Deep Learning Algorithms

Edited by: **Zoran Gacovski**

ARCLER PRESS

Deep Learning Algorithms

Deep Learning Algorithms

Edited by:

Zoran Gacovski

ARCLER PRESS

www.arclerpress.com

Deep Learning Algorithms

Zoran Gacovski

Arcler Press
224 Shoreacres Road
Burlington, ON L7L 2H2
Canada
www.arclerpress.com
Email: orders@arclereducation.com

e-book Edition 2022

ISBN: 978-1-77469-362-9 (e-book)

ISBN: 978-1-77469-183-0 (Hardcover)

Arcler Press publishes wide variety of books and eBooks. For more information about Arcler Press and its products, visit our website at www.arclerpress.com

DECLARATION

Some content or chapters in this book are open access copyright free published research work, which is published under Creative Commons License and are indicated with the citation. We are thankful to the publishers and authors of the content and chapters as without them this book wouldn't have been possible.

ABOUT THE EDITOR

Dr. Zoran Gacovski has earned his PhD degree at Faculty of Electrical engineering, Skopje. His research interests include Intelligent systems and Software engineering, fuzzy systems, graphical models (Petri, Neural and Bayesian networks), and IT security. He has published over 50 journal and conference papers, and he has been reviewer of renowned Journals. Currently, he is a professor in Computer Engineering at European University, Skopje, Macedonia.

TABLE OF CONTENTS

Section 2: Deep Learning Techniques Applied in Biology

LIST OF CONTRIBUTORS

Md Nazmus Saadat
University of Kuala Lumpur, Malaysia

Muhammad Shuaib
University of Kuala Lumpur, Malaysia

Junta Wu
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518071, China

Huiyun Li
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518071, China

Soohyun Park
Korea University, Seoul, Republic of Korea

Dohyun Kim
Naver Webtoon Corporation, Seongnam, Republic of Korea

Joongheon Kim
Korea University, Seoul, Republic of Korea

Qi Yue
Xi'an Institute of Optics and Precision Mechanics, CAS, Xi'an 710119, China
University of Chinese Academy of Sciences, Beijing 100039, China
Xi'an University of Posts and Telecommunications, Xi'an 710121, China

Caiwen Ma
Xi'an Institute of Optics and Precision Mechanics, CAS, Xi'an 710119, China

Xi-liang Chen
Institute of Command Information System, PLA University of Science and Technology, No. 1, Hai Fu Road, Guang Hua Road, Qin Huai District, Nanjing City, Jiangsu Province 210007, China

Lei Cao
Institute of Command Information System, PLA University of Science and Technology, No. 1, Hai Fu Road, Guang Hua Road, Qin Huai District, Nanjing City, Jiangsu Province 210007, China

Chen-xi Li
Institute of Command Information System, PLA University of Science and Technology, No. 1, Hai Fu Road, Guang Hua Road, Qin Huai District, Nanjing City, Jiangsu Province 210007, China

Zhi-xiong Xu
Institute of Command Information System, PLA University of Science and Technology, No. 1, Hai Fu Road, Guang Hua Road, Qin Huai District, Nanjing City, Jiangsu Province 210007, China

Jun Lai
Institute of Command Information System, PLA University of Science and Technology, No. 1, Hai Fu Road, Guang Hua Road, Qin Huai District, Nanjing City, Jiangsu Province 210007, China

Suxia Cui
Department of Electrical and Computer Engineering, Prairie View A&M University, Prairie View, TX 77446, USA

Yu Zhou
Department of Electrical and Computer Engineering, Prairie View A&M University, Prairie View, TX 77446, USA

Yonghui Wang
Department of Computer Science, Prairie View A&M University, Prairie View, TX 77446, USA

Lujun Zhai
Department of Electrical and Computer Engineering, Prairie View A&M University, Prairie View, TX 77446, USA

Keke Zhang
College of Engineering, Northeast Agricultural University, Harbin 150030, China

Qiufeng Wu
College of Science, Northeast Agricultural University, Harbin 150030, China

Anwang Liu
College of Engineering, Northeast Agricultural University, Harbin 150030, China

Xiangyan Meng
College of Science, Northeast Agricultural University, Harbin 150030, China

Yu Sun
School of Information Science and Technology, Beijing Forestry University, Beijing 100083, China

Yuan Liu
School of Information Science and Technology, Beijing Forestry University, Beijing 100083, China

Guan Wang
School of Information Science and Technology, Beijing Forestry University, Beijing 100083, China

Haiyan Zhang
School of Information Science and Technology, Beijing Forestry University, Beijing 100083, China

Ngoc Giang Nguyen
Graduate School of Natural Science and Technology, Kanazawa University, Kanazawa, Japan;

Dau Phan
Graduate School of Natural Science and Technology, Kanazawa University, Kanazawa, Japan;

Favorisen Rosyking Lumbanraja
Graduate School of Natural Science and Technology, Kanazawa University, Kanazawa, Japan;

Mohammad Reza Faisal
Graduate School of Natural Science and Technology, Kanazawa University, Kanazawa, Japan;

Bahriddin Abapihi
Graduate School of Natural Science and Technology, Kanazawa University, Kanazawa, Japan;

Bedy Purnama
Graduate School of Natural Science and Technology, Kanazawa University, Kanazawa, Japan;

Mera Kartika Delimayanti
Graduate School of Natural Science and Technology, Kanazawa University, Kanazawa, Japan;

Kunti Robiatul Mahmudah
Graduate School of Natural Science and Technology, Kanazawa University, Kanazawa, Japan;

Mamoru Kubo
Institute of Science and Engineering, Kanazawa University, Kanazawa, Japan

Kenji Satou
Institute of Science and Engineering, Kanazawa University, Kanazawa, Japan

Awwal Muhammad Dawud
Department of Computer Engineering, Cyprus International University, Nicosia, Cyprus

Kamil Yurtkan
Department of Computer Engineering, Cyprus International University, Nicosia, Cyprus

Huseyin Oztoprak
Department of Computer Engineering, Cyprus International University, Nicosia, Cyprus

Hai Hu
Applied Nuclear Technology in Geosciences Key Laboratory of Sichuan Province, Chengdu University of Technology, Chengdu, China

Yang Shao
Applied Nuclear Technology in Geosciences Key Laboratory of Sichuan Province, Chengdu University of Technology, Chengdu, China

Shijie Hu
Applied Nuclear Technology in Geosciences Key Laboratory of Sichuan Province, Chengdu University of Technology, Chengdu, China

Eduardo Ribeiro
Department of Computer Sciences, University of Salzburg, Salzburg, Austria
Department of Computer Sciences, Federal University of Tocantins, Palmas, TO, Brazil

Andreas Uhl
Department of Computer Sciences, University of Salzburg, Salzburg, Austria

Georg Wimmer
Department of Computer Sciences, University of Salzburg, Salzburg, Austria

Michael Häfner
St. Elisabeth Hospital, Vienna, Austria

Asif Mansoor
National University of Sciences and Technology, Islamabad, Pakistan

Muhammad Waleed Usman
National University of Computer and Emerging Sciences, Islamabad, Pakistan

Noreen Jamil
National University of Computer and Emerging Sciences, Islamabad, Pakistan

M. Asif Naeem
National University of Computer and Emerging Sciences, Islamabad, Pakistan

Lei Zhu
Training Center of Xinjiang Air Traffic Management Bureau, Urumqi, China

Guodong Zhu
College of Atmospheric Science, Nanjing University, Nanjing, China
Meteorological Center of Xinjiang Air Traffic Management Bureau, Urumqi, China

Lei Han
Meteorological Center of Xinjiang Air Traffic Management Bureau, Urumqi, China

Nan Wang
Meteorological Center of Xinjiang Air Traffic Management Bureau, Urumqi, China

Haijun Zhang
Guangdong Provincial Key Laboratory of Conservation and Precision Utilization of Characteristic Agricultural Resources in Mountainous Areas, Meizhou, China
School of Computing, Jiaying University, Meizhou, China

Yinghui Chen
Guangdong Provincial Key Laboratory of Conservation and Precision Utilization of Characteristic Agricultural Resources in Mountainous Areas, Meizhou, China
School of Mathematics, Jiaying University, Meizhou, China

Wenling Li
College of Science, Yanbian University, Yanji, China

Bo Jin
College of Science, Yanbian University, Yanji, China

Yu Quan
Department of Economics and Management of Yanbian University, Yanji, China

Ruzhang Yang
Shanghai Foreign Language School, Shanghai, China

Xiangxi Jiang
Barstow School of Ningbo, Ningbo, China

LIST OF ABBREVIATIONS

ANN	Artificial neural network
AUV	Autonomous underwater vehicle
BA	Boltzmann addition
BM	Boltzmann multiplication
CPU	Central processing unit
CAD	Computer-aided diagnosis
CT	Computer tomography
CV	Computer vision
CNN	Convolution neural network
DBN	Deep belief network
DCNN	Deep convolution neural network
DNNs	Deep neural network
DRL	Deep reinforcement learning
DRBM	Deep restricted Boltzmann machine
DPG	Deterministic policy gradient
GRU	Gated recurrent units
GPU	Graphical processing unit
HBPNNs	Hybrid BP neural networks
ICH	Intracranial haemorrhage
JELSR	Joint embedding learning and sparse regression
LR	Logistic regression

LSTM	Long short-term memory network
MV	Majority voting
MDP	Markov decision processes
MRSF	Minimum redundancy spectral feature selection
MVEP	Motion-onset visual evoked potential
MLP	Multi-layer perceptron
NIRS	Near-infrared spectroscopy
PMI	Pointwise mutual information
PFCL	Prior fully connected layers
QDA	Quadratic discriminant analysis
RBF	Radial Basis Function
RH	Relative humidity
RBM	Restricted Boltzmann machine
RMSE	Root mean square error
SVD	Singular value decomposition
SCP	Slow cortical potentials
TORCS	The open racing car simulator

PREFACE

The Deep learning is a branch of machine learning based on data presentation via complex representations with high degree of abstraction - that are obtained by applying learned nonlinear transformations. Deep learning methods find their application in important areas of artificial intelligence, such as: computer vision, natural language processing, speech and sound comprehension, as well as in bioinformatics. Deep learning is a class of machine learning algorithms that:

- uses multilayer nonlinear processor units to extract and transform features. Each subsequent layer takes as input the output elements of the previous layer.
- learns in a supervised and / or unsupervised manner.
- learns a number of levels of representation - corresponding to different degrees of abstraction.
- uses some form of descending gradient algorithm to train through error backpropagation.

The layers used in deep programming include the hidden layers of the artificial neural network and a multitude of statement formulas.

This book covers the most important discriminant and generative deep models with special emphasis on practical implementations. We cover the key elements of classical neural networks and provides an overview of the building blocks, regularization techniques, and learning methods that are specific to deep models. Also we consider the deep convolutional models and illustrates their application in image classification and natural language processing.

The generative deep models are often used in computer vision applications and natural language processing. Sequence modeling by deep feedback neural networks can be applied in the field of natural language processing. Practical implementations of deep learning are made in modern dynamic languages (Python, Lua or Julia), and also with application frameworks for deep learning (e.g. Theano, TensorFlow, Torch).

This edition covers different topics from deep learning algorithms, including: methods and approaches for deep learning, deep learning applications in biology, deep learning applications in medicine, and deep learning applications in pattern recognition systems.

Section 1 focuses on methods and approaches for deep learning, describing advancements in deep learning theory and applications - perspective in 2020 and beyond; deep ensemble reinforcement learning with multiple deep deterministic policy gradient algorithm; dynamic decision-making for stabilized deep learning software

platforms; deep learning for hyperspectral data classification through exponential momentum deep convolution neural networks; and ensemble network architecture for deep reinforcement learning.

Section 2 focuses on deep learning applications in biology, describing fish detection using deep learning; deep learning identification of tomato leaf disease; deep learning for plant identification in natural environment; and applying deep learning models to mouse behavior recognition.

Section 3 focuses on deep learning applications in medicine, describing application of deep learning in neuroradiology: brain hemorrhage classification using transfer learning; a review of the application of deep learning in brachytherapy; exploring deep learning and transfer learning for colonic polyp classification; and deep learning algorithm for brain-computer interface.

Section 4 focuses on deep learning applications in pattern recognition systems, describing application of deep learning in airport visibility forecast; hierarchical representations feature deep learning for face recognition; review of research on text sentiment analysis based on deep learning; classifying hand written digits with deep learning; and bitcoin price prediction based on deep learning methods.

SECTION 1:

Methods and Approaches for Deep Learning

CHAPTER 1

ADVANCEMENTS IN DEEP LEARNING THEORY AND APPLICATIONS: PERSPECTIVE IN 2020 AND BEYOND

Md Nazmus Saadat and Muhammad Shuaib
University of Kuala Lumpur, Malaysia

ABSTRACT

The aim of this chapter is to introduce newcomers to deep learning, deep learning platforms, algorithms, applications, and open-source datasets. This chapter will give you a broad overview of the term deep learning, in context to deep learning machine learning, and Artificial Intelligence (AI) is also introduced. In Introduction, there is a brief overview of the research achievements of deep learning. After Introduction, a brief history of deep learning has been also discussed. The history started from a famous scientist called Allen Turing (1951) to 2020. In the start of a chapter after

Citation: Md Nazmus Saadat and Muhammad Shuaib (December 9th 2020). Advancements in Deep Learning Theory and Applications: Perspective in 2020 and beyond, Advances and Applications in Deep Learning, Marco Antonio Aceves-Fernandez, IntechOpen, DOI: 10.5772/intechopen.92271.

Introduction, there are some commonly used terminologies, which are used in deep learning. The main focus is on the most recent applications, the most commonly used algorithms, modern platforms, and relevant open-source databases or datasets available online. While discussing the most recent applications and platforms of deep learning, their scope in future is also discussed. Future research directions are discussed in applications and platforms. The natural language processing and auto-pilot vehicles were considered the state-of-the-art application, and these applications still need a good portion of further research. Any reader from undergraduate and postgraduate students, data scientist, and researchers would be benefitted from this.

Keywords:- Deep learning, machine learning, artificial intelligence, neural networks

INTRODUCTION

Deep learning is focusing comprehensively on video, image, text and audio recognition, autonomous driving, robotics, healthcare, etc. [1]. Deep learning is a result orientated field of study that why getting very much attention from researcher and academicians. The Rina Dechter introduced the word of deep learning in 1986, the main motivation behind the advent of field deep learning was making an intelligent machine that mimic the human brain. In humans, the brain is the most important and decision-making organ; brain takes decision based on sight, smell, touch, and sounds. The brain also can store memory and solve complex problems based on their experience.

For the last few decades, the researchers dreamed of making a machine that is as intelligent as, like our brains, they started studying the biological structure and working of the human brain. Making a robot that performs certain duties and self-driving cars is to reduce roadside incidents. Because according to the World Health Organization (WHO), 1.35 million people die every year in road incidents [2] and approximately 90% of the incidents are due to human errors [3]. To develop state-of-the-art devices for the applications listed above, ones need to think in a different way of programming a device to make it artificially intelligent. Deep learning is one of the most innovative paradigms that make it possible up to some extent. In deep learning, the word deep indicates the number of layers through which data are converted from input to the desired output. It is difficult for a new researcher or student to recognize any project whether it is from artificial intelligence machine learning or deep learning because all these overlap

each other some way or the other. Machine learning is any sort of computer program that can learn by their own without having specially programmed by the programmer. There are two types of machine learning: supervised learning and unsupervised learning. In supervised learning, you teach or train the machine with a fully labeled data, the machine learns from the labeled data and then anticipate the unforeseen data. In supervised learning, the machine can only give you correct output when the input is already experienced in training phase; it is based on experience; the more is the training dataset or experience of your machine the higher is the chances of getting the actual output. It is a time-consuming process and also required a lot of expertise in data science. On the other hand, in unsupervised learning, supervision of a model is not needed, rather the model work on its own catches new data and discovers the information inside the data. It usually deals with label-less data; compared to supervised learning, unsupervised learning is more complicated. It is usually used to find features and unknown patterns.

Deep learning models are agile and result oriented in terms of complicated abstractions. Deep learning models are mostly based on ANN, categorically CNNs, although there are deep belief networks, generative models, propositional formulas and Boltzmann machine also play their part (Figure 1).

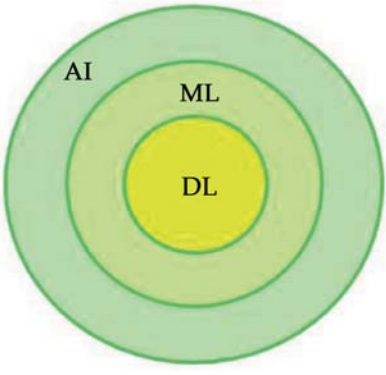


Figure 1. Deep learning a subset of machine learning and AI.

Deep learning has been evaluated as a game-changer in AI and computer vision. Today, state-of-the-art object detection is possible only due to deep learning [4]; traditional methods of object detection are not enough to cater with detection so smartly. To understand the whole image of object detection, it is not necessary to only focus on image classification, but to precisely

calculate the concept and locations of the objects in every image, that is, object detection which is based on face detection, pedestrian detection, and skeleton detection [5]. Deep learning has cutting-edge technology and has application in every field of life ranging from computational to healthcare. It has a very deep impact on the life of the people or societies because its application is always the need of the day. The deep learning also gains significant importance due to new and flourishing field called big data analytics. Big data analytics is the number of complicated processes examining large and varied data sets, or it is also defined as techniques and methods used to identify the hidden patterns, unknown correlations market trends, and customer preference from huge dataset. Big data analytics can offer various business benefits, that is, more effective marketing strategies, better customer service, improved operational efficiency, etc.

Deep learning is an emerging area of research and modern application. The deep learning is a very widespread and demanding field now-days, it covers industry, business, and healthcare; it combines all the hot research-oriented fields, that is, IoT, e-health-care, cybersecurity, bioinformatics, optimization, and cyber-physical systems; these all are seen interdependent. Gartner has proposed top ten technology trends for 2020, some of them are, hyper-automation, human augmentation, AI Security, IoT, Autonomous things; etc.; all are related to AI, machine learning, and deep learning some way or the other. Surely, deep learning will bring a bunch of innovations to everywhere whether it is industry, health-care or business intelligence. According to Ref. [6], machine learning and AI will be used more in 2020 experts says in the survey conducted by the computer-world.

In 2019, many researchers, academicians, and teachers claimed that deep learning is over because it cannot do common-sense reasoning; Rodney Brooks a professor in MIT says that some popular press started stories that the deep learning will be over by 2020. In 2020, hybrid, interdisciplinary, collaborative, and open-minded research is expected to add more contribution. The topics that are expected to be more prevalent in 2020 are common-sense reasoning, active learning and life-long learning, multi-modal and multi-task learning, open-domain dialogue conversation, medical applications and autonomous vehicles, ethics that includes privacy, confidentiality, and biases, and finally robotics.

There are two most common deep learning platforms: TensorFlow and PyTorch; these two platforms compete; and this competition is very fruitful for the community; TensorFlow is easy to use, integrated with Keras; while

on the other hand, Pytorch has TPU support, etc. In 2020, it is expected to have a platform which can easily transform a TensorFlow model to Pytorch and vice versa. There is a need to develop an actively developed stable reinforcement learning framework. The higher layers of abstractions are expected in 2020 like Keras, so that machine learning is used outside the machine learning fields.

History

Deep learning is a sub branch of machine learning, and machine learning is a sub branch of artificial intelligence. Deep learning is a set of algorithms that processes large set of data and imitates the thinking process. The history of deep leaning is started from 1943, when Warren McCulloch and Walter Pitts created a neural network-based computer model. There basic aim was to mimic thought process of human brain; they used algorithms and mathematics to make the threshold logic to mimic human thought process. Alan Turing called the father of AI concluded in 1951 that the machines would not take much time in started thinking of their own; at some point of time, they would be able to talk to each other; and it is also expected that they would take the control of the universe. In context to this, the frank Rosenblatt introduced single and multi-layer artificial neural network (1957–1962). The history amazed us when the world champion of chess player called Kasparov was defeated by deep blue computer in 1997. In 1957–62, the single layer and multi-layer perceptron's was introduced. The first deep feedforward general purpose learning algorithm multilayer perceptron's by Alexey Icakhnenko and Lapa was published in 1967. In 1971, a deep network with eight layers trained by the group method of data handling algorithm was described already. The idea of backpropagation, Recurrent Neural Network (RNN), and restricted Boltzmann machine (RBM) was introduced in 1970–1986. In 1979-1998, the Convolution Neural Network (CNN), Bidirectional RNN, and long short-term memory (LSTM) were the state of the art. The deep belief network (DBN) was introduced by Geoff Hinton in 2006. The data sets called ImageNet and AlexNet that was created in 2009. Generative Adversarial Network (GAN) is a class of machine learning system invented by Ian Goodfellow and his colleagues in 2014. Coming up in history in 2016 Google DeepMind challenge match between Alpha Go versus Lee Sedol, the AlphaGo win all the matches from a world champion Lee Sedol. AlfaGo and AlfaZero are computer programs developed by artificial intelligence

research company called DeepMind in (2016–2017); it plays the board game Go. The transformer introduced in 2017–19 a deep learning model used specially used for Natural language Processing (NLP). Although there is a lot of community contributed to the deep learning but Yann LeCun, Geoffrey Hinton, and Yoshua Bengio have received Turing awards in 2018.

DEEP NETWORK TOPOLOGIES

Deep neural network (DNN)

In DNN, there is multilayer perceptron or hidden layer between the input and output. All the layers are connected to previous layers; by going through each layer, the network estimates the exact output based on the weights and activation function. Through DNN, we can model any complex non-linear relation. The backbone of the DNN is the characteristic of learning about the feature that is most relevant to the targets [7]. The DNN has research gap in model selection, training dynamics, by using graph convolution neural network combination optimization, and Bayesian neural network for estimation of uncertainty. There are a lot of applications for DNN, that is, computer vision, machine translation, social network filtering, playing board, video games, and medical diagnosis (Figure 2).

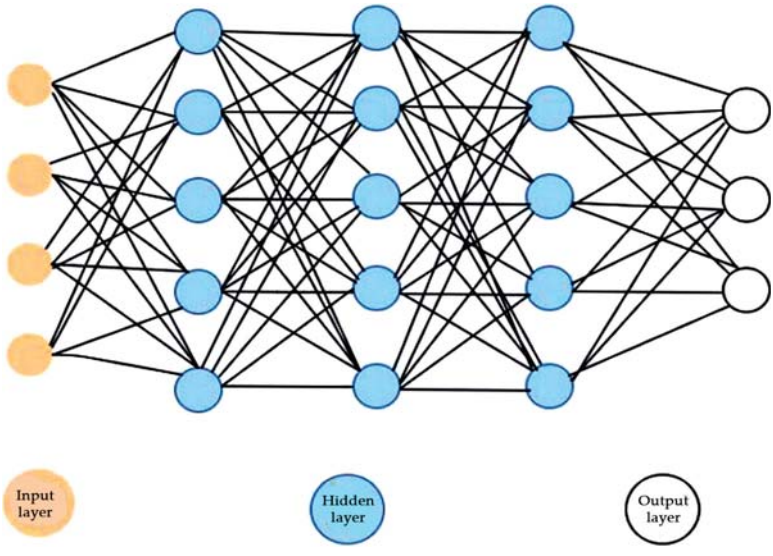


Figure 2. Deep neural network.

Recurrent neural network (RNN)

RNN is a type of deep learning network that is used specifically when there is sequential data or time-series, that is, video, speech, etc. The RNN usually maintained the data from the previous state to the next state. It is called recurrent because it performs the same function for each input, while the output is different because it also depends on past calculations. The state-of-the-art topic of deep learning with RNN is Long Short-Term Memory Network (LSTM). RNN provides the solution to many problems, that is, intelligent transportation system [8], solving time-varying matrix inversion [9], and many more. The RNN is famous for sentence evaluation and linguistic data processing (Figure 3).

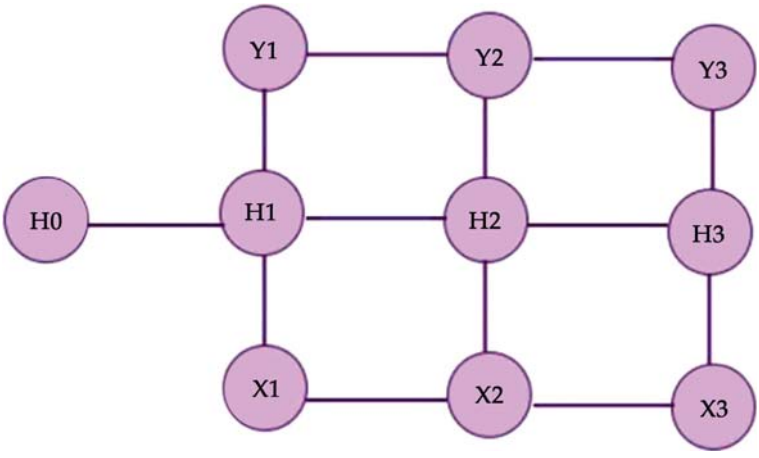


Figure 3. Recurrent neural network.

Deep belief network (DBN)

DBN is a probabilistic unsupervised deep learning algorithm. It has many layers of hidden variables. To solve the more complex problems, it needs more hidden layers; each layer is a special statistical relation with the other layer. DBN can learn probabilistically; after learning, BDN needs training under supervisor to perform classification. The DBN is used to recognize clusters and generates images, video sequences, and motion-capture data (Figure 4).

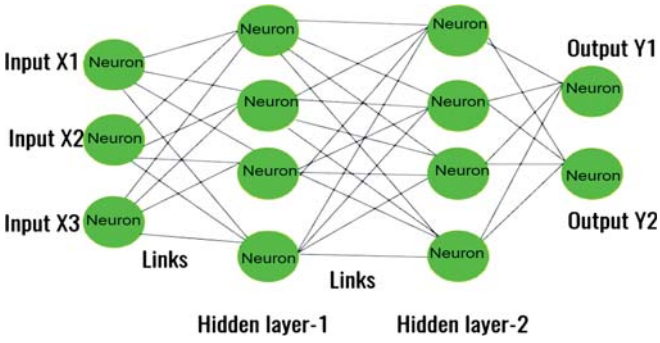


Figure 4. Deep belief network.

Boltzmann machine (BM)

The BM is a network that is a uniformly attached, neuron-like unit, which is responsible for taking decisions stochastically about whether to be off or on. Computational problems are solved through BM like search, optimization, and learning problem. Many features are uncovered in learning algorithm that shows very complex behavior in training dataset. Boltzmann machine is used for classification and dimensionality reduction.

Restricted Boltzmann machine (RBM)

RBM introduced in 1986 by Smolensky: two layers visible and hidden units, while there is no connection between visible-visible and hidden-hidden. It can learn a probability distribution over a collection of datasets. The applications of RBM are features learning, collaborative filtering, dimensionality reduction, and classification.

Convolutional neural network (CNN)

In CNN, the layers are delicately connected to input layer as well as each other. There is a specific function for each neuron of the subsequent layer like it is only responsible for only a part of the input. CNN is now widely used for remote sensing, computer vision, audio, and text processing [10].

Deep auto-encoder

Just like others, deep auto-encoder has also many hidden layers. The difference between a simple auto-encoder and deep-auto-encoder is the simple auto-encoder that has one hidden layer, while the deep-auto-encoder has many

hidden layers. In deep-auto-encoder, the training is complex normally, you need to train one hidden layer first to reconstruct the structure of the input data, and this input data are further used to train other hidden layers and so on. Some applications of deep auto-encoder are image extraction, image generation recommendation system, and sequence to sequence prediction.

Gradient descent (GD)

GD is used to reduce the overall cost function; it is considered as an optimization algorithm and is widely used for determination of coefficient function in machine learning. When there is not possible to estimate the parameters analytically, then GD is used to calculate the desired parameters. Using the GD weight of the model is updated for every epoch. It is used for supervised machine learning.

Stochastic gradient descent (SGD)

Just like GD, SGD is also an optimization algorithm but GD is used when the datasets are small, while SGD is usually used when the datasets are large, and SD becomes very costly if used for a large number of datasets.

APPLICATION OF DEEP LEARNING

Deep learning is new and state-of-the-art technology used for large scale applications now-days. Deep learning (also called differential programming or structure learning) is member of a large family of machine learning class. It is edge-cutting technology used for many different new research fields which are stated below.

Deep learning in automatic speech recognition

The automatic speech recognition is the convincing application of deep learning. Speech recognition means making speech as in input to a machine that can make the input process very easy and has a hundred of other advantages as well, that is, illiterate people can also use technology, speech coding, text to speech synthesis, speech recognition, speaker recognition, speech enhancement, speech segmentation, language identification, and many more [11]. The speech is the natural form of communication, hence it is considered a very convincing application.

Image recognition

Image recognition based on deep learning becomes very famous and accurate result-oriented technology based on the training and experience of machine. Deep learning plays a very important part in image recognition and image classification in underwater target recognition [12] although the images from underwater are always noisy and deteriorated. MNIST is one of the most renowned examples used for image classification, below is the simple of dataset of MNIST dataset (Figure 5).

Figure 5. Image example of handwritten digits from the MNIST dataset.

Natural language processing

LSTM helps a lot in language modeling and machine translation [13]; language modeling task is to understand the language. To implement the language, models' neural networks are used. Google translate is the most famous and widely used application in this regard; Google translate is used for more than 100 languages all over the world. It also used LSTM; and it learns from millions of examples and translates the whole sentence rather than word by word translation. BERT (Google) is one of the most common technologies in this field achieved a lot of benchmarks, that is, sentence classification, sentence pair classification, sentence pair similarity, sentence tagging, create contextualized words embedding, question answering, and multiple-choice questions. There are some other transformer-based language models developed in 2019, which are XLNet (Google/CMU), RoBERTa (Facebook), Distil BERT (hugging Face), CTRL (Salesforce), GPT-2 (Open-AI), ALBERT (Google), and Magatron (NVIDIA). Magatron is the largest transformer model ever trained. It has 8.3 million parameters transformer language model. XLNet is the best transformer in terms of performance; XLNet outperforms BERT on 20 tasks often by a large margin. ALBERT developed by Google is used to reduce the parameters via cross-layer

parameters sharing. The state of the artwork in this domain is about multi-domain task-oriented dialogue system [14]. In 2020, it expected to combine common sense reasoning with language models, extending language model context to thousands of words and to have more focus on open-domain dialogue (Figure 6).

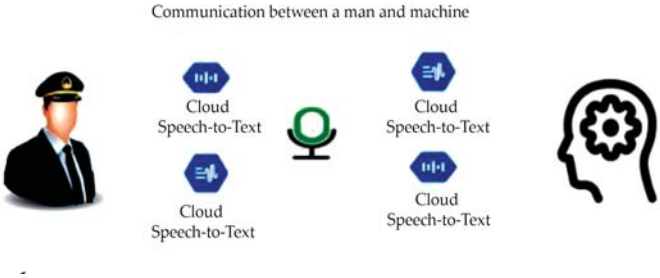


Figure 6. NLP and deep learning.

Games and robotics

Robots are the agents who are artificially intelligent and working in the real-world replacing humans. OpenAI and Dota 2 are popular games; in 2017, 1v1 bot beats top professional Dota 2 players; in 2018, OpenAI five lost two games against top Dota 2 player, while in 2019, OpenAI five beat OG team (the world champion in 2018). The OpenAI five win in 2019 is only because of the more training compute; the current version of OpenAI has consumed 800 petaflops/day and experiences about 45,000 years of dota self-play over 10 real-time months.

The current version has 99.9%-win rate versus the 2018 version. It is one of the best experiences in deep learning that systems that learn to play with each other and incrementally improving. OpenAI Rubiks Cube Manipulation is another example from Robotics. The researchers are expecting in 2020 to implement reinforcement-learning methods in the manipulation of real-world interaction tasks.

In games, experts are loss from different machines, using these machines to assist human experts in discovering new strategies. Waymo a company that is focusing on developing auto-pilot like Tesla in October 2018; they have 10 million miles on road and now in 2020 they have 20 million miles on road 20,000 of classes for structure test, also initiated testing without having a safety driver.

Financial fraud detection

Deep learning is playing a very important role in financial fraud detection. With the advent of technology and a significant amount of e-commerce platforms, the number of e-payments is increasing day by day chances of financial fraud, which is also a source of headache for banks and other financial institutions. Thus, focusing on fraud detection is a hot area of research. The author of [15] used auto-encoder for financial fraud detection [16]. This research uses deep learning model for fraud detection, while [17] proposed a solution to fraud detection using machine learning approach.

Deep learning in health-care

In this modern era of computing, deep learning also produced best results medical and health care, that is, deep learning is used for cancer cell coordination, organ segmentation, protein folding, lesion detection, and image enhancement in the field of medicine. There are several other issues like [18, 19, 20, 21] and much more where deep learning is directly involved in the suggestion of the ultimate solution to the problem in healthcare.

Military

Deep learning is used for making many different military devices used in wars or other spy services. The military is also working on robots to train the robots to handle the critical situation through these robots. The militaries of some countries are making their weapons more intelligent using AI. In a war zone, AI can be embedded in the robots for remote surgical support in healthcare.

Cybersecurity

Cybersecurity is also one of the hot research areas; deep learning models are used for the cybersecurity of the Internet of Things (IoT) [22]. The IoT devices are usually low power devices having power-constrained that's why always vulnerable to external threats. Deep learning models can detect threats more accurately than any other technology. The author of [23] used deep learning and machine learning for intrusion, spam, and malware detection.

MODERN DEEP LEARNING PLATFORMS

Open-sources deep learning platforms discussed in this section. It will provide a quick review of the open-source platforms for beginners and mediocre because every platform has its pros and cons.

TensorFlow

The TensorFlow is new and open-source platform for differential programming; it was developed by Google team called Google brain and was first released in 2015 [24]. In February 2017, they released version 1.0.0; TensorFlow can work on CPU and GPU; it is available for Mac, Linux, and windows and also for mobile computing platform android and iOS. It is the most famous machine learning library in the world today. Its best-supported client language is python but there is also interface available in C++, Java, and GO. It is easy to use and have Keras integration. TensorFlow has many of its versions available like for mobiles TensorFlow lite, for industry TensorFlow Serving, etc.

Pytorch

Pytorch is also machine learning and deep learning library, based on torch library. It was initially released by Facebook's AI Research lab (FAIR) in 2016. Pytorch has two high-level features, Tensor computing with graphics processing units (GPU), and auto-diff based deep neural network. It is too easy in Pytorch to move tensors to and from GPU. Pytorch Mobile is the version of Pytorch used for mobiles. There are some key features of Pytorch; the first feature is called imperative programming; most of the python code is imperative; this type of programming is more flexible. The other feature of Pytorch is dynamic computation graphs, it run time the system generates the graph structure, dynamic graph work well for dynamic networks like RNN, dynamic graph also makes debugging very easy. The Pytorch provides maximum flexibility and speed during implementing and building deep neural network.

Theano

Theano is designed by Montreal Institute for Learning Algorithms (MILA), which is very famous after their deployment, but unfortunately, there is no support after version 1.0.0 (November 2017). It is a python library designed for code compilation optimization [25]; it is primarily used for mathematical operations like multi-dimensional arrays. Theano was far better than other python libraries like Numpy in terms of speed, computing symbolic graphs, and stability optimizations. Tensor operations, GPU computation, and parallelism are also supported by Theano.

Microsoft cognitive toolkit (CNTK)

CNTK is used for commercial-grade distributed deep learning. It can be used as a standalone tool for machine learning or also can be included as a library in C++ programs, python, and C#; its model evaluation functionality can be also used from Java programs. It supports ONNX that allows sharing model with frameworks Caffe2, MXNet, and PyTorch [26]. CNTK can be used only on Linux and Windows. The CNTK is considered as a powerful machine learning platform similar surge of performance as compared to other widely used platforms [27].

Keras

Keras is a powerful library written in python; it uses TensorFlow, Theano, and CNTK as a framework because it does not have their framework. Keras can work on GPUs and CPUs and can also support RNNs and CNNs. The beauty of Keras is it has the ability of fast and easy prototyping; Keras is user-friendly. It has been ranged one of the most cited API in 2018 and has enough number of users on board.

Deep learning 4J

It is distributed open-source, robust deep learning framework for Java designed by Skymind [28] which is added a lot to Java ecosystem and eclipse foundation. It has compatibility with Clojure and Scala APIs just like Keras; it is also able to work with both CPUs and GPUs. It is widely used for academics and industrial applications.

Torch

It is a scientific computing open-source machine learning framework released in October 2002; it is not able to work on CPUs; it is only made to focus on GPUs accelerated computing. It is developed in programming language C and based on Lua, a contribute in a LuaJIT, a scripting language. Max OSX and Ubuntu 12+ can use this framework, although they have Platform for Windows, but their implementations are not supported officially [29].

Caffe and Caffe2

CAFFE (Convolutional Architecture for Fast Feature Embedding) created by Berkeley AI Research (BAIR) is a framework for deep learning. It is developed in C++ with a python interface. Caffe2 was introduced by the research group of Facebook in 2017, but Caffe2 was merged in PyTorch in

March 2018. It supports multiple platforms, that is, Mac OS X, Windows, Linux, iOS, and Android [30].

Apache MXNet

An MXNet is a fast-scalable deep learning platform that supports many programming languages, i.e., Scala, Julia, C++, R, Python, Gluon API, and Perl APIs. Like Torch, it is also made only for GPUs, and it is very competent in multi GPUs implementations. The Apache MXNet is scalable flexible and portable, and due to these qualities, it attracts many users.

TRAINING ALGORITHMS

One of the most important parts of deep learning is learning algorithms. The deep neural network can be differentiated only through the number of layers; if the number of layers increases, the network becomes deeper and more complex. Each layer has its specific function or can detect or help in the detection of the special feature.

According to the author [31], if the problem is face recognition, the first layer has the responsibility to recognize edges, the second has to detect higher features such as the nose, eye, ears, etc., the next layer can further dig out the features, and so on. Thus, each layer is developed earlier to the development of training algorithm like gradient descent; that's why these kinds of classifiers are not suitable for a dataset with huge volume or variation. This was discussed by Yann et al. [32]; they further concluded that a system with less manual and more automatic design can give better results in pattern recognition.

Backpropagation is the solution; it takes information from the data without going through classifiers and finds the representation needed for recognition. List of few famous training algorithms is listed below.

Gradient descent

In statistics, data science, and machine learning, we optimize a lot of stuffs; when we fit a line with linear regression, we optimize the intercept and slope; when we use logistic regression, we optimize a squiggle; when we use t-SNE, we optimize clusters. The gradient descent is used to optimize all these and tons of others as well.

Gradient descent algorithm is similar to Newton's roots finding algorithm of 2D function. The methodology is very simple; just pick a point randomly

on a curve and move toward the right or left along x-axis depending on the positive and negative value of the slope of the function at the given point up-till the value of y-axis, that is, function or f(x) becomes zero. There is the same concept behind the gradient descent; we move or traverse along a specific path in many-dimensional space weight when the error rate is reduced to your limits than we stop. It is one of the underlying concepts for most of deep learning and machine learning algorithms.

$$C = \frac{1}{2}(Yexpected - Yactual)^2 \quad (1)$$

Stochastic gradient descent

A method used for optimizing an objective function with the iterative method is called stochastic gradient descent. It can also be called gradient descent optimization. Stochastic gradient descent would randomly pick one sample for each step and from that, just use this one sample to calculate the derivatives, thus in super sample example, stochastic gradient descent reduced the number of terms by computed by 3.If we had one million samples than the stochastic gradient descent would reduce the number of terms by computed by factor of one million. In stochastic gradient descent, when minibatch of the number of samples finished running than updates are applied, in here update of weights is more frequent, so we reach a global minimum in less time (Figure 7).

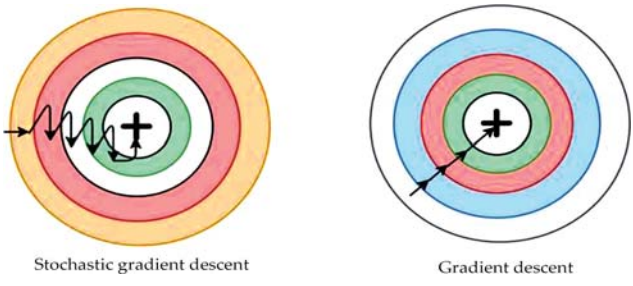


Figure 7. Comparison of GD and SGD.

Momentum

In stochastic gradient descent to update the weight or to calculate step size, a fixed multiplier is used as a learning rate; this can cause the update to overshoot a potential-minima; if the gradient is too steep or delay, the convergence of the gradient is noisy. The concept of momentum used in

Physics is velocity exponentially decreasing an average of gradient [33]. This prevents the descent going in the wrong direction.

Levenberg-Marquardt algorithm

This type of algorithm is used for curve fitting or non-linear least-squares problems. This algorithm is also called as deep least-square; these kinds of issues arise usually in the least-squares curve fitting. It was first introduced by Kenneth Levenberg in 1944, although it was rediscovered by statistician called Donald Marquardt in 1963.

Backpropagation through time

It is one of the famous and standard methods used to train the recurrent neural network. It was developed independently by several researchers. Unlike general-purpose optimization techniques, it is faster in training RNN. The backpropagation through time also has issues with local optima [34].

ROUTINE CHALLENGES OF DEEP LEARNING

According to Google trends graph more and more expert and professionals have attracted toward deep learning in last five year; the percentage of professionals increased from 12 to 100% [35, 36]. Deep learning is used everywhere, that is, bio-informatics, computer vision, IoT security, health-care, e-commerce, digital marketing, natural language processing, and many more [37, 38]. Because of the very hot research area, there must have some challenges which are enlisted below.

Non-contributing columns or inputs

When dealing with data or making a model, several inputs are not necessary for finding any feature, so it is advised to drop un-necessary attributes. There is also necessary to find one best column and make it separate from the dataset; it can be done using numpy array in Keras; but it is difficult and challenging to find best match attribute.

Number of hidden layers

The number of hidden layers is directly propositional to computational complexity and deepness of the network. To deal with a large number of layers require a high computational cost, difficult to manage a large number of neurons.

Optimization algorithms

In model optimizations, gradient descent optimizer helps to make the model cost minimum by adjusting the value; choosing an optimizer is also a challenging task to do, because sometimes it makes your cost of model high rather than decreasing the model cost.

Loss function

Is from the name indicate loss function, it estimates the loss or the difference between the expected outcome and the actual outcome the formula for loss function is listed below.

$$\text{Floss}=\text{Expected outcome}-\text{actual outcome} \quad (1)$$

There are many different ways to calculate the loss function; choosing a loss function is also one of the essential and challenging tasks of deep learning

Activation function

There are many different activation functions; every activation function does not produce the same results; sigmoid activation function shows good results with binary classification problem. One needs to be careful about Tanh activation function because of the vanishing gradient problem. In multi-labeled classification, softmax is the best option; Relu should be used when there is much zeros in the input side because Relu is good in dead neuron generation. It is also a point to use the required activation function.

Epoch

When the dataset is passed backwards and forward through the whole neural network, it is called one epoch, as after every epoch value of weights assigned is analyzed to make model. The weights are changed, checked, and tested in every cycle for the same dataset simulation. The main memory is keeping the record of all the training data; sometimes it is not possible to keep all the record in main memory, like for larger datasets, so the epoch is brought to memory in divided or batches form, and finally the result is represented as an epoch output. Dealing with epoch is also a challenging task in deep learning.

AVAILABLE OPEN-SOURCE DATASETS

Research in machine learning and deep learning is started since last many decades hence significant improvement it brings to the society in terms of various application-based on deep learning and machine learning. There are many freely available datasets on the web which can be used by researchers for various purposes.

Image datasets (Table 1):

Table 1. Open source image datasets

Pascal VOC	**MS COCO**
MNIST handwritten digits	NORB
CIFAR10/CIFAR100 color images data set with	COIL100
Caltech101	Google's Open Images
Caltech 256	COIL 20
The dataset of street view	LabelMe
STL-10	ImageNet

Geospatial datasets available online:

- NEXRAD
- OpenstreetMAP
- Landsat8

Dataset available for text (Table 2):

Table 2. Text open-source datasets

Google books Ngrams	**Yelp open dataset**	**20 newsgroups**	
UCI's Spambase (Older)	Prediction	UCI machine learning repository	Text classification datasets
SQuAD	Google books Ngrams	Broadcast news	WikiText
Penn Treebank	Reuters news dataset	Billion words dataset:	Common crawl

Artificial datasets:

- Arcade Universe
- Dataset inspired from baby-AIschool
- All images and question datasets
- Deep vs. shallow comparison ICML
- Background correlation
- Rectangles data
- Mnist variations

Facial datasets (Table 3):

Table 3. Databases for face recognitions.

Labeled faces in the wild	**UMD faces annotated dataset**	**CASIA WebFace facial**
MS-Celeb-1M	Olivetti	Multi-Pie
JACFEE	FERET	mmifacedb
Indian face database	The Yale face database	Mut1nyFace/head segmentation dataset

Recent additions of datasets (Table 4):

Table 4. Free databases developed recently

The UZH-FPV drone racing dataset	**North Korean missile test database**	**Flickr-Faces-HQ Dataset (FFHQ)**
Hotels-50K	MIMIC-CXR	Google Audioset
Two new evaluation datasets	Open-source biometric data recognition	Uber 2B trip data
Yelp Open Dataset	Core50	Data portals
Open data monitor	Quandl data portal	Mutiny face/head segmentation dataset
Awesome public dataset	Head CT scan dataset	Open datasets
WAPo	Chess dataset	NLP datasets

REFERENCES

1. Aliper A, Plis S, Artemov A, Ulloa A, Mamoshina P, Zhavoronkov A. Deep learning applications for predicting pharmacological properties of drugs and drug repurposing using transcriptomic data. Molecular Pharmaceutics. 2016;13(7):2524-2530
2. World Health Organization. Global status report on road safety. Available from: https://www.who. int/violence_injury_prevention/ road_safety_status/2018/en/ [Accessed: 31 Janaury 2018]
3. U. D. O. Transportation. Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash
4. Causation Survey. Washington, DC: National Center for Statistics and Analysis; 2015
5. Zhao Z-Q , Zheng P, Xu S-T, Wu X. Object detection with deep learning: A review. IEEE Transactions on Neural Networks and Learning Systems. 2019;30(11):3212-3232
6. Kobatake H, Yoshinaga Y. Detection of spicules on mammogram based on skeleton analysis. IEEE Transactions on Medical Imaging. 1996;15(3):235-245
7. Top 3 enterprise tech trends to watch in 2020. 2020. Available from: https://www.computerworld.com/ article/3512109/top-3-enterprise-techtrends-to-watch-in-2020.html
8. Zhong S, Hu J, Fan X, Yu X, Zhang H. A deep neural network combined with molecular fingerprints (DNN-MF) to develop predictive models for hydroxyl radical rate constants of water contaminants. Journal of Hazardous
9. Materials. 2020;383(5)
10. Kong J, Huang J, Yu H, Deng H, Gong J, Chen H. RNN-based default logic for route planning in urban environments. Neurocomputing. 2019;338:307-320
11. Zhang Z, Zheng L, Wang M. An exponential-enhancedtype varying-parameter RNN for solving time-varying matrix inversion. Neurocomputing. 2019;338:126-138
12. Konstantinidis D, Argyriou V, Stathaki T. A modular CNN-based building detector for remote sensing
13. images. Computer Networks. 2020;138:107034

14. Karpagavalli S, Chandra EH. A Review on Automatic Speech Recognition Architecture and Approaches. International Journal of Signal Processing, Image Processing and Pattern Recognition. 2016;9(4):393-404
15. Jin L, Liang H. Deep Learning for Underwater Image Recognition in Small Sample Size Situations. Aberdeen UK: OCEANS; 2017
16. Jozefowicz R, Vinyals O, Schuster M, Shazeer N, Wu Y. Exploring the Limits of Language Modeling. California, USA: Google Brain; 2016
17. Wu CS, Madotto A, Hosseini-Asl E, Xiong C, Socher R, Fung P. Transformable multi domain state
18. generator for task oriented dialogue system. In: arXiv preprint arXiv; 2019
19. Zamini M, Montazer G. Credit Card Fraud Detection using autoencoder based clustering. In: 9th International Symposium on Telecommunications (IST). Tehran, Iran; 2018. pp. 486-491
20. Mubalaike AM, Adali E. Deep learning approach for intelligent financial fraud detection system. In: 3rd International Conference on Computer Science and Engineering (UBMK). Sarajevo, Bosnia; 2018
21. Vidanelag HMMH, Tasnavijitvong T, Suwimonsatein P, Meesad P. Study on machine learning techniques with conventional tools for payment fraud detection. In: 11th International Conference on Information Technology and Electrical Engineering (ICITEE). Pattaya, Thailand; 2019
22. Subiksha K. Improvement in analyzing healthcare systems using deep learning architecture. In: 4th International Conference on Computing Communication and Automation (ICCCA). Greater Noida, India; 2018
23. Hajjo R. The ethical challenges of applying machine learning and artificial intelligence in cancer care. In: 1st International Conference on Cancer Care Informatics (CCI). Amman, Jordan; 2018
24. Nugroho H, Harmanto D, Hassan Al-Absi HR. On the development of smart home care: Application of deep learning for pain detection. In: IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES). Sarawak, Malaysia; 2018
25. Shickel B, Tighe PJ, Bihorac A, Rashidi P. Deep EHR: A survey of recent advances in deep learning techniques for electronic health record (EHR) analysis. IEEE Journal of Biomedical and Health Informatics. 2017;22(5):1589-1604

26. Roopak M, Tian GY, Chambers J. Deep learning models for cyber security in IoT networks. In: IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC). Las Vegas USA; 2019
27. Apruzzese G, Colajanni M, Ferretti L, Guido A, Marchetti M. On the effectiveness of machine and deep learning for cyber security. In: 10th International Conference on Cyber Conflict (CyCon). Tallinn, Estonia; 2018
28. Hatcher WG, Yu W. A survey of deep learning: Platforms, applications and emerging research trends. Human-Centered Smart Systems and Technologies. 2018;6:24411-24432
29. E. A. The Theano Development. Theano: A Python framework for fast computation of mathematical expressions. In: arXiv preprint arXi; 2016
30. The Microsoft Cognitive Toolkit. 2017. Available from: https://docs.microsoft.com/en-us/cognitive-toolkit/
31. Shi S, Wang Q , Xu P, Chu X. Benchmarking state-of-the-art deep learning software tools. In: 7th International Conference on Cloud Computing and Big Data (CCBD). Macau, China; 2017
32. Keras: The Python Deep Learning Library. 2017. Available from: https:// keras.io/
33. Torch: A Scientific Computing Framework for LuaJIT. 2017. Available from: http://torch.ch/
34. Giang N, Dlugolinsky S, Bobák M, Tran V, García L, Heredia I, et al. Machine learning and deep learning frameworks and libraries for large-scale data mining. Artificial Intelligence Review. 2019;52(1):77-124
35. Maryam NM, Villanustre F, Khoshgoftaar TM, Seliya N, Wald R, Muharemagic E. Deep learning applications and challenges in big data analytics. Journal of Big Data. 2015;2(1):1
36. Yann L, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE. 1998;86(11):2278-2324
37. Ian G, Bengio Y, Courville A. Deep learning. In: Adaptive Computation And Machine Learning, Cambridge. Cambridge USA: MIT Press; 2016
38. Mahmood A, Shrestha A. Review of deep learning algorithms and architectures. IEEE Access. 2019;7:53040-53065

39. Kumar OS, Joshi N. Rule power factor a new interest measure in associative classification. Procedia Computer Science. 2016;93:12-18
40. Sharma O, Kumar S, Joshi N. Significant rule power an algorithm for new interest measure. In: Smart Computing and Informatics. Singapore: Smart Innovation, Systems and Technologies; 2018
41. Dong C, Loy CC, He K, Tang X. Learning a deep convolutional network for image super-resolution. In: European Conference on Computer Vision. Cham; 2014
42. Seonwoo M, Lee B, Yoon S. Deep learning in bioinformatics. Briefings in Bioinformatics. 2017;18(5):851-869
43. Pathmind. 2019. Available from: https://pathmind.com/wiki/ open-datasets

CHAPTER 2

DEEP ENSEMBLE REINFORCEMENT LEARNING WITH MULTIPLE DEEP DETERMINISTIC POLICY GRADIENT ALGORITHM

Junta Wu and Huiyun Li

Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518071, China

ABSTRACT

Deep deterministic policy gradient algorithm operating over continuous space of actions has attracted great attention for reinforcement learning. However, the exploration strategy through dynamic programming within the Bayesian belief state space is rather inefficient even for simple systems. Another problem is the sequential and iterative training data with autonomous

Citation: Junta Wu, Huiyun Li, "Deep Ensemble Reinforcement Learning with Multiple Deep Deterministic Policy Gradient Algorithm", Mathematical Problems in Engineering, vol. 2020, Article ID 4275623, 12 pages, 2020. https://doi.org/10.1155/2020/4275623.

vehicles subject to the law of causality, which is against the i.i.d. (independent identically distributed) data assumption of the training samples. This usually results in failure of the standard bootstrap when learning an optimal policy. In this paper, we propose a framework of m-out-of-n bootstrapped and aggregated multiple deep deterministic policy gradient to accelerate the training process and increase the performance. Experiment results on the 2D robot arm game show that the reward gained by the aggregated policy is 10%–50% better than those gained by subpolicies. Experiment results on the open racing car simulator (TORCS) demonstrate that the new algorithm can learn successful control policies with less training time by 56.7%. Analysis on convergence is also given from the perspective of probability and statistics. These results verify that the proposed method outperforms the existing algorithms in both efficiency and performance.

INTRODUCTION

Reinforcement learning is an active branch of machine learning, where an agent tries to maximize the accumulated reward when interacting with a complex and uncertain environment [1, 2]. Reinforcement learning combining deep neural network (DNN) technique [3, 4] had gained some success in solving challenging problems. One of the most noticeable results was achieved through the deep Q-network (DQN), which exploited deep neural networks to achieve maximum accumulated reward [5]. DQN has performed well over 50 different Atari games and inspired many deep reinforcement learning (DRL) algorithms [6–8].

However, DQN only deals with the tasks with small, discrete state and action spaces while many reinforcement learning tasks have large, continuous, real-valued state and action spaces. Although such tasks could be solved with DQN by discretizing the continuous spaces, the instability of the control system may be increased. For overcoming this difficulty, deterministic policy gradient (DPG) algorithm [9] with the DNN technique was proposed, producing deep deterministic policy gradient (DDPG) algorithm [10]. Unfortunately, DDPG suffers from inefficient exploration and unstable training [11]. Many existed works attempted to solve the problems. Gu et al. proposed the Q-prop method, a Taylor expansion of the off-policy critic as a control variant to stabilize DDPG [12]. Q-Prop combines the on-policy Monte Carlo and the off-policy DPG; it achieves the advantages of sample efficiency and stability. Mnih et al. proposed A3C to stabilize the training process of DDPG, by training the parallel agents

with asynchronously accumulated updates [13]. Interactive learning with the environment in multiple threads is performed at the same time, and each thread summarizes the learning results and stores them in a common place. In this way, A3C avoids the problem of too strong correlation of empirical playback and achieves an asynchronous concurrent learning model. This method consumes considerable computation resources. When the implementation complexity is not a strong limit, we can use any of these policy gradient-related methods to generate subpolicies to further improve our method, where the centralized experience replay buffer stores and shares experiences from all subpolicies, enabling more knowledge gained from the environment.

Additionally, researchers attempted to overcome the disadvantage of unstable training of DDPG and speed up the convergence of DDPG with bootstrap technique recently [14]. Osband et al. developed bootstrapped DQN as the critic of DDPG [15]. Yang et al. employed a multiactor architecture for multitask purpose [16]. DBDDPG [11] and MADDPG [17] both used multiactor-critic structure to improve the exploration efficiency and increase the training stability. Shi et al. introduced deep soft policy gradient (DSPG) [18], an off-policy and stable model-free deep RL algorithm by combining policy and value-based methods under maximum entropy RL framework. The authors discover that the standard bootstrap is likely to fail when learning an optimal policy, since in most reinforcement learning tasks, the sequential and iterative training data subject to the law of causality, which is against the i.i.d. (independent identically distributed) assumption of the training samples. Hence, a novel bootstrap technique is needed for achieving the optimal policy.In consideration of the above shortcomings of the previous work, this paper introduces a simple DRL algorithm with m-out-of-n bootstrap technique [19, 20] and aggregated multiple DDPG structures. The control policy will be gained by averaging all learned subpolicies. Additionally, the proposed algorithm uses the centralized experience replay buffer to improve the exploration efficiency. Since m-out-of-n bootstrap with random initialization produces reasonable uncertainty estimates at low computational cost, this helps in the convergence of the training. The proposed bootstrapped and aggregated DDPG can substantially reduce the learning time. The remainder of this paper is organized as follows. Section 2 presents a brief background. Section 3 introduces the proposed method in detail and analyses the convergence of the algorithm. The experimental results of the proposed method are presented in Section 4. The paper is concluded in Section 5.

BACKGROUND

Reinforcement Learning

In a classical scenario of reinforcement learning, an agent aims at learning an optimal policy according to the reward function by interacting with the environment E in discrete time steps, where policy is a map from the state space to action space [1]. At each time step, the environment state s_t is observed by the agent, and then it executes the action at by following the policy π. Afterwards, a reward $r(s_t, a_t)$ is received immediately. The following equation defines the accumulated reward that the agent receives from step t:

$$R_t = \sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i), \tag{1}$$

where $\gamma \in [0, 1]$ is a discount factor. As the agent maximizes the expected accumulated reward $E[R_t]$ from the initial state, the optimal policy π^* will be gained finally

Deterministic Policy Gradient Algorithm

Policy gradient (PG) algorithms optimize a policy directly by maximizing the performance function with the policy gradient. Deterministic policy gradient algorithm which is originated from deterministic policy gradient theorem [9] is one of the policy gradient methods. It learns deterministic policies $\mu(\bullet \mid \theta): S \longmapsto A$ with the actor-critic framework, while the critic estimates the action-value function and the actor represents the deterministic policy function. /e updates for the action-value function and the policy function are given below

$$\begin{aligned} w &= \arg\min_{w} E_{s\sim\rho^{\mu}(\bullet), a\sim\mu(\bullet\mid\theta)}\Big[\big(r(s,a) + \gamma Q(s', \mu(s' \mid \theta) \mid w) \\ &\quad - Q(s, a \mid w)\big)^2 \Big], \\ \theta &= \arg\max_{\theta} E_{s\sim\rho^{\mu}(\bullet)}[Q(s, \mu(s \mid \theta) \mid w)], \end{aligned} \tag{2}$$

where $\rho^{\mu}(\bullet)$ denotes the discounted state distribution [9]. Since full optimization is expensive, stochastic gradient optimization is usually used instead. /e following equation shows the deterministic policy gradient [9] which is used to update the parameter of the deterministic policy:

$$\nabla_{\theta} J = \mathrm{E}_{s\sim\rho^{\mu}(\bullet), a\sim\mu(\bullet\mid\theta)}\left[\nabla_{\theta}\mu(s \mid \theta)\nabla_{a} Q(s, a \mid w)\big|_{a=\mu(s\mid\theta)}\right]. \tag{3}$$

DDPG Algorithm

DDPG applies the DNN technique onto the deterministic policy gradient algorithm [10], which approximates deterministic policy function μ and actionvalue function Q with neural network, as shown in Figure 1.

There are two sets of weights in DDPG. w, θ are weights for main networks while w′, θ′ are weights for target networks which are introduced in [5] for generating the Q-learning targets. We use $Q(\bullet,\bullet \mid w)$ and $\mu(\bullet \mid \theta)$ to denote the main networks while $Q'(\bullet,\bullet \mid w)$ and $\mu'(\bullet \mid \theta)$ represent the target networks. As equations (4) and (5) shows, weights of the main networks are updated according to the stochastic gradient, while weights of target networks are updated with "soft" updating rule [10], as shown in equation (6):

$$w \longleftarrow w + \alpha_w \left[\left(r(s,a) + \gamma Q' \right) \left(s', \mu'(s' \mid \theta) \mid w' \right) - Q(s,a \mid w) \right)^2 \Big], \tag{4}$$

$$\theta \longleftarrow \theta + \alpha_\theta \nabla_\theta \mu(s \mid \theta) \nabla_a Q(s,a \mid w)\big|_{a=\mu(s|\theta)}, \tag{5}$$

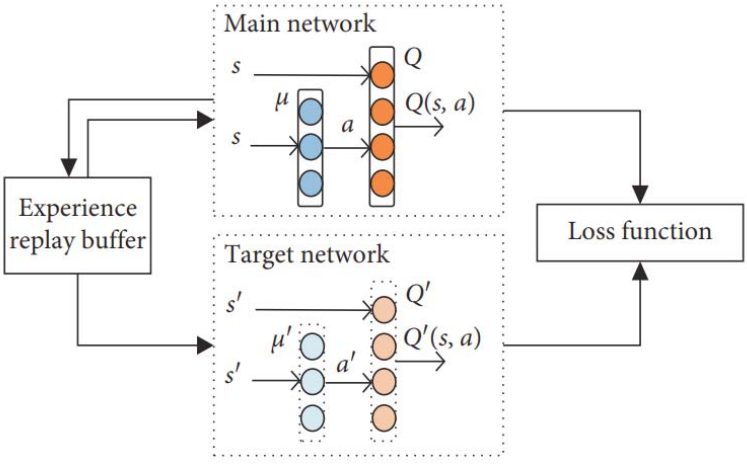


Figure 1. Diagram of deep deterministic policy gradient.

$$\begin{aligned} w' &= \tau w + (1-\tau) w', \\ \theta' &= \tau \theta + (1-\tau) \theta'. \end{aligned} \tag{6}$$

DDPG utilizes the experience replay technique [10] to break training samples' temporal correlation, keeping them subject to the i.i.d. (independent identically distributed) assumption. Furthermore, the "soft" updating rule is used to increase the stability of the training process. DDPG updates the

main actor network with the policy gradient, while the main critic network is updated with the idea of combining the supervised learning and Q-learning which is used in DQN. After training, the main actor network converges to the optimal policy.

METHODS

Structure of Multi-DDPG

Compared with DQN, DDPG is more appropriate for reinforcement learning tasks with continuous action spaces. However, it takes long time for DDPG to converge to the optimal policy. We propose multiDDPG structure and bootstrap technique to train several subpolicies in parallel so as to cut down the training time. We randomly initialize N main critic networks $Q_i(s, a |w_i)$ and main actor networks $\mu_i(s| \theta_i)$ with weights w_i and θ_i (i = 1, 2, ... , N), and then, we initialize N target networks Q_i' and μ_i' with weights $w_i' \longleftarrow w_i, \theta_i' \longleftarrow \theta_i$ (I = 1, 2, ... , N) and initialize the centralized experience replay buffer R.

The structure of multi-DDPG with the centralized experience replay buffer is shown in Figure 2. We name the proposed method which utilizes the multi-DDPG structure and bootstrap technique as bootstrapped aggregated multiDDPG (BAMDDPG). Figure 3 demonstrates that BAMDDPG averages all action outputs of trained subpolicies to achieve the final aggregated policy. For clarity, the terms agent, main actor network, and subpolicy refer to the same thing and are interchangeable in this paper. Algorithm 1 presents the entire algorithm of BAMDDPG.

In Algorithm 1, "#Env" means the number of environment modules while "#selected DDPG" represents the number of selected DDPG components. During the training process, each DDPG component which exploits the actorcritic framework is responsible for training the corresponding subpolicy. Figure 2 demonstrates the training process of a DDPG component, containing the interaction procedure and the update procedure.

In the interaction procedure, the main actor network which represents an agent interacts with the environment. It receives the current environment state s_t and outputs an action a_t. The environment gives the immediate reward r_t and the next state st+1 after executing the action. Then the transition tuple $(s_t, a_t, r_t, s_{t+1})_t$ is stored into the central experience replay buffer. To efficiently explore the environment, noise sampled from an Ornstein–

Uhlenbeck process N is added to the action.In the update procedure, a random minibatch of transitions used for updating weights is sampled from the central experience replay buffer. e main critic network is updated by minimizing the loss function which is based on the Q-learning method [1], while the target networks are updated by having them slowly track the main networks. Weights of the main actor network are updated with the policy gradient along which the overall performance increases. By following such an update rule, each subpolicy of BAMDDPG gradually improves. The centralized experience replay buffer stores experiences from all subpolicies.

Figure 3 illustrates the aggregation details of subpolicies. We denote subpolicies approximated by main actor networks with $\mu_1(\bullet), \mu_2(\bullet), \ldots, \mu_N(\bullet)$ and the outputs of these subpolicies with a_1, a_2, ... , a_N. In addition, the aggregated policy's output is denoted as a.

In practice, we train multiple subpolicies by setting a maximum number of episodes. Since episodes in BAMDDPG terminate earlier than that of the original DDPG algorithm with less steps, the training time of subpolicies is less than the optimal policy. It can be predicted that the performance of less-trained subpolicies will be worse than the optimal policy to some degree, but we can aggregate the trained subpolicies to increase the performance and get the optimal policy. Furthermore, we use the average method as aggregation strategy in consideration of the equal status and real-valued outputs of all subpolicies. Specically, the outputs of all subpolicies are averaged to produce the final output.As Figure 2 demonstrates, the interaction procedure of a DDPG requires an environment component to interact with the agent Therefore, multi-DDPG structure requires multiple environment modules. However, for some reinforcement learning tasks, the environment module does not support being copied for multiple DDPGs. In such case, the environment component interacts with only one subpolicy in each time step. BAMDDPG supports reinforcement learning tasks with both one environment module and multiple environment modules by choosing one subpolicy or multiple subpolicies to interact with the environments in each time step. All subpolicies are then updated simultaneously with sampled minibatch from the centralized experience replay buffer. In the end, all trained subpolicies are averaged to form the final policy. Algorithm 1 presents the BAMDDPG algorithm.

Additionally, from the perspective of intuition, the centralized experience replay technique exploited in BAMDDPG enables each agent to use experiences encountered by other agents. /is makes the training of

subpolicies of BAMDDPG more efficient since each agent owns a wider vision and more environment information.

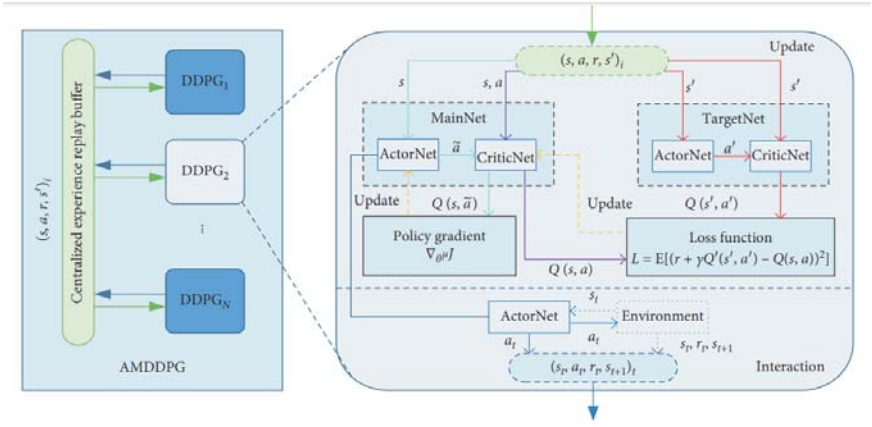


Figure 2. Structure of BAMDDPG.

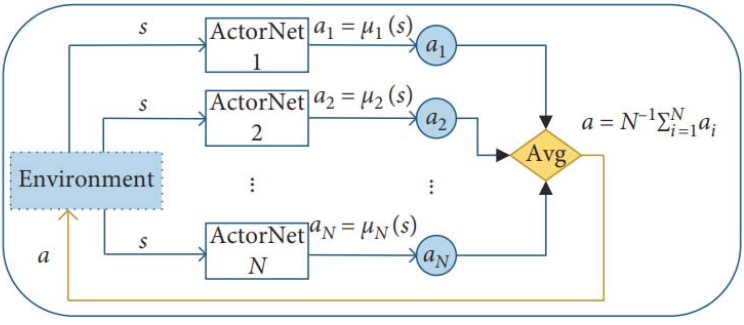


Figure 3. Aggregation of subpolicies.

Randomly initialize N main critic networks $Q_i(s, a \mid w_i)$ and main actor networks $\mu i(s \mid \theta_i)$ with weights w_i and θ_i $(i = 1, 2, \ldots, N)$ Initialize N target networks Q_i' and μ_i' with weights $w_i' \longleftarrow w_i, \theta_i' \longleftarrow \theta_i (i = 1, 2, \ldots, N)$

Initialize centralized experience replay buffer R

for episode = 1, M do

Initialize an Ornstein–Uhlenbeck process N for action exploration

if #Env == 1 do

Alternately select Q_i and μ_i among multiple DDPGs to interact with the environment

else do

Select all Q_i and μ_i, each DDPG is bound with one environment
end if

for t = 1, T do

for #selected DDPG do

Receive state s_t from its bound environment

Execute action $a_t = \mu_i(s_t \mid \theta_i) + \mathcal{N}_t$ and observe reward rt and new state st+1
Store experience (s_t, a_t, r_t, s_{t+1}) in R

end for

for i = 1, N do

Update w_i, θ_i, w_i', and θ_i' according to equations (4)–(6)

end for

end for

end for

Get final policy by aggregating subpolicies: $\mu(s) = N^{-1}\sum_{i=1}^{N}\mu_i(s \mid \theta_i)$

Algorithm 1. Bootstrapped and aggregated multi-DDPG (BAMDDPG).

Analysis on Convergence with Bootstrap and Aggregation

For ease of description, we suppose BAMDDPG trains N subpolicies simultaneously and denote these subpolicies with μ_i (i = 1, 2, · · · N). /e aggregated policy is denoted as μ which can be formulated as

$$\overline{\mu} = \mathrm{Avg}[\mu_i] = N^{-1}\sum_{i=1}^{N}\mu_i, \tag{7}$$

where $\overline{\mu}$ represents the aggregation of subpolicies. Let the optimal policy denoted as μ^*. Then the following formula holds [20]

$$\mathrm{Avg}\left[(\mu_i - \mu^*)^2\right] \geq (\overline{\mu} - \mu^*)^2. \tag{8}$$

where $\mathrm{Avg}[(\mu_i - \mu^*)^2]$ means the average bias of subpolicies and the optimal policy while $(\overline{\mu} - \mu^*)^2$ represents bias of the aggregated policy and the optimal policy.

Equation (8) demonstrates that the aggregated policy has better performance than subpolicies and approximates the optimal policy more closely than any subpolicy. Under this conclusion, the aggregated policy

approximates the optimal policy quickly as subpolicies are trained to a certain extent [21].

Further, we analyze the convergence from the perspective of probability and statistics [22]. Assume all policies are from the policy space U. The subpolicies $\mu_1, \mu_2, \ldots, \mu_N$ are sampled according to a distribution function F in U. Let $\widehat{F}$ denote the empirical cumulative distribution function

$$\widehat{F}(x) = N^{-1} \sum_{i=1}^{N} I(u_i \le x), \tag{9}$$

) where N is the number of the sampled subpolicies. $I[\bullet]$ is an indicator function which outputs 1 when the condition is satisfied, otherwise 0. The operator " $\le$ " in "$u_i \le x$" means x is a better policy than u_i in U, which indicates the agent acting by following the policy x is able to gain more reward than those only adopting u_i. According to the rule of Dvoretzky–Kiefer–Wolfowitz inequality [23], we get

$$P\left(\sup_{x \in U} |\widehat{F}(x) - F(x)| > \delta\right) \le 2e^{-2n\delta^2}, \tag{10}$$

where P(·) represents probability, sup(·) means upper bound, and δ is an arbitrary small positive integer.

Equation (10) shows that $\widehat{F}$ converges uniformly to the true distribution function exponentially fast in probability. Suppose we are interested in the mean $\mu = \mathrm{T}(F)$, then the unbiasedness of the empirical measure extends to the unbiasedness of linear functions of the empirical measure. Actually, empirical cumulative distribution can be seen as a discrete distribution with equal probability for each component, which means we can get a policy from the empirical cumulative distribution by averaging multiple policies. Therefore, the aggregating policy $\overline{\mu}$ subjects to empirical cumulative distribution and it subjects to true distribution.

Since $\overline{\mu}$ is a better policy than $\mu_i (i = 0, \ldots, n)$ in U, $\overline{\mu}$ converges to the optimal policy of U.

The m-out-of-n Bootstrap

Bootstrap [14] is a significant resample technique in statistics, which generally works by random sampling with the replacement process. In this paper, we try to train multiple DDPG components with bootstrap. It is analyzed that such requirement can be simply attained by initializing the network weights of different DDPG components with different methods [15]. Therefore, we

adopt this technique as a prior and multiple DDPG components are trained in parallel on different subdataset from experience replay buffer.

However, standard bootstrap fails as the training data subject to a long-tail distribution, rather than the usual normal distribution, as the i.i.d. assumption implies. A valid technique is m-out-of-n bootstrap method [19], where the number of bootstrap samples is much smaller than that of the training dataset. More specifically, we draw subsamples without replacement and use these subsamples as new training datasets. Multiple DDPG components are then trained with this newly produced training dataset

RESULTS AND DISCUSSION

2D Robot Arm

In order to illustrate the effectiveness of aggregation, we use BAMDDPG to learn a control policy for a 2D robot arm task.

Benchmark and Reward Function

As Figure 4 demonstrates, a 2D robot arm contains a two-link arm with one joint which is attempting to get to the blue block. Th first link rotates around the root point while the second link rotates around the joint point. /e action of an agent consists of two real-valued numbers denoting angular increment. We construct the reward according to the distance between the finger point of the arm (endpoint) and the blue block. /e farther away the finger point being from the blue block, the lesser the reward is. Additionally, the reward adds one when the distance $\sqrt{dx^2 + dy^2}$ is less than the threshold δ. When the finger point stops within the blue block for a while (more than 50 iterations), the reward adds ten. /e following equation presents the reward:

$$r = -\frac{\sqrt{dx^2 + dy^2}}{200} + I\left[\sqrt{dx^2 + dy^2} < \delta\right] + 10 * I[\eta > 50], \tag{11}$$

where I[·] is an indicator function which outputs 1 when the condition is satisfied, otherwise

Performance of Aggregated Policy

During the training process of BAMDDPG, each agent interacts with its corresponding environment, producing multiple learning curves. Figure 5 demonstrates learning curves of 3 subpolicies with shared experience on 2D

robot arm benchmark. /e curve depicts the moving average of episode reward while the shaded area depicts the moving average ± partial standard deviation. As Figure 5 shows, the training process of BAMDDPG's subpolicies is better than that of DDPG. /e centralized experience replay buffer stores and shares experiences from all subpolicies, enabling more knowledge gained from the environment. /erefore, BAMDDPG's subpolicies can gain more reward during the training process. After about 1000 episodes, the subpolicies of BAMDDPG and the policy of the original DDPG both converge.

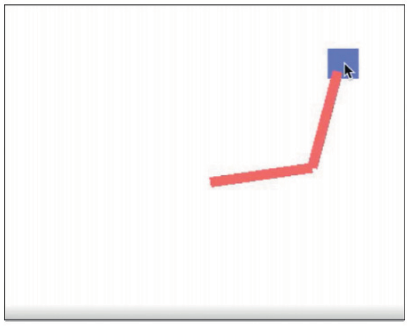


Figure 4. 2D robot arm benchmark.

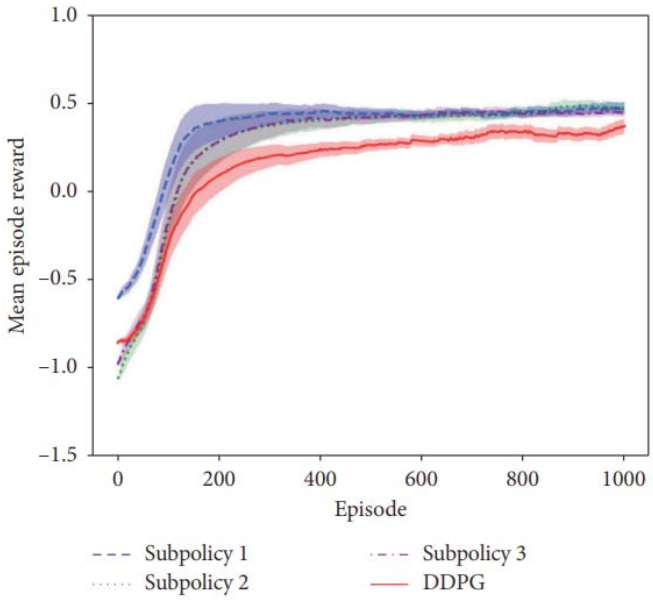


Figure 5. Comparison between sub-DDPGs of BAMDDPG and DD.

The key of BAMDDPG is the aggregation of subpolicies. In this section, we show the comparison of performance between the aggregated policy and subpolicies so as to illustrate the effectiveness of aggregation. Suppose the action given by the i^{th} subpolicy is $a_i = [ai_1, ai_2]$, then the immediate reward of the i^{th} subpolicy is given by:

$$IR_i = -\frac{f(a_i)}{200} + I[f(a_i) < \delta] + 10 * I[\eta > 50], \tag{12}$$

where $f(a_i)$ denotes the distance between the finger point of the arm and the blue block after executing action a_i while it is an implicit function. The immediate reward of the aggregated policy can be expressed in the same way:

$$IR = -\frac{f(\sum_{i=1}^{n} a_i)}{200} + I\left[f\left(\sum_{i=1}^{n} a_i\right) < \delta\right] + 10 * I[\eta > 50], \tag{13}$$

where $\sum_{i=1}^{n} a_i$ represents the action taken by the aggregated policy.

Table 1 shows the performance comparison of subpolicies and aggregated policy of BAMDDPG. /e result demonstrates that reward gained by the aggregated policy is 10%~50% better than those gained by subpolicy.

TORCS

Benchmark and Reward Function

The Open Racing Car Simulator (TORCS) is a car driving simulation software with high portability, which takes the client-server architecture [24, 25]. It realistically simulates real cars by modeling the physical dynamic models of the car engines, brakes, gearboxes, clutches, etc. It is a commonly used DRL benchmark and is appropriate for test of self-driving techniques. Using TORCS, a developer is able to easily access a simulated car's sensor information. Therefore, the controller of the simulated car is able to get the current environment state and follow a policy to send controlling instructions, including control of steering, brake, and throttle. Figure 6 presents TORCS's client-server architecture. /e controller connects to the race server through the user datagram protocol (UDP). At each time step, the information of the current driving environment state is perceived by the simulated car and is sent to the controller. /e server then waits for an instruction from the controller for 10 ms. The simulated car executes the corresponding actions according to the current instruction, or last instruction if no new instruction

is sent.Designing a suitable reward function is a key for using TORCS as the platform to test BAMDDPG, which helps to learn a good policy to control the simulated car. We describe the details of designing the reward function in this section. As the driving environment state of TORCS can be perceived by various sensors of the simulated car, we can create the reward function using these sensor data which is shown in Table

Equation (14) presents our constructed reward function, which restricts the behavior of the simulated car in TORCS. Each time the simulated car interacts with the driving environment of TORCS, we expect to gain as large reward as possible through the following equation:

$$r = \left(v \times \left(\frac{1 - |v - \beta|}{\alpha} \right)^{\mathbb{I}[\![d_1 \le 10]\!]} \right) \times \cos\phi \times (1 - |\sin\phi|) \times (1 - |d_2|) \times \left(\frac{|d_1|}{50} \right)^{\mathbb{I}[\![d_1 \le 50]\!]}, \tag{14}$$

Table 1. Performance comparison of subpolicies and the aggregated policy

Policy	Episodes	Total reward	Average reward
Subpolicy1	20	720.69	36.03
Subpolicy2	20	538.28	26.91
Subpolicy3	20	463.98	23.20
Aggregated policy	20	829.17	41.46

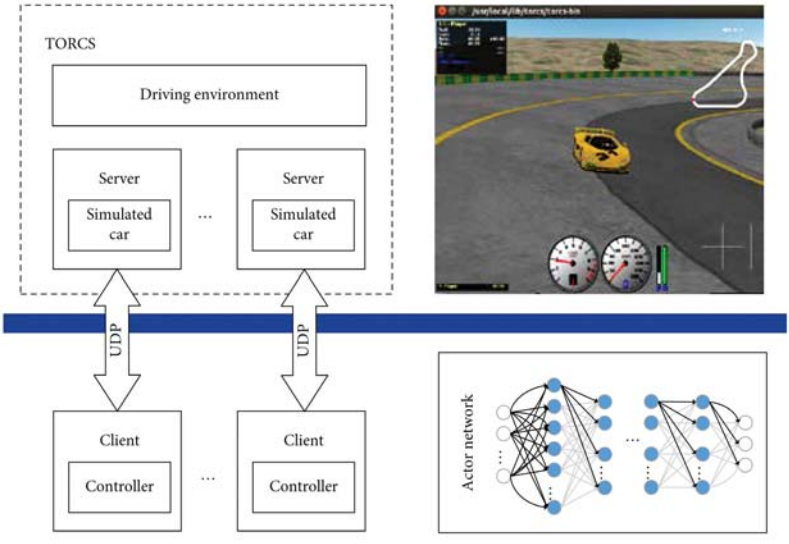


Figure 6. Diagram of the client-server architecture of TORCS.

Table 2. Information of sensor data for creating the reward function

Name	Range (unit)	Description
ϕ	$[-\pi, +\pi]$ (rad)	Angle between track direction and car's forward direction
v	$[-\infty, +\infty]$ (km/h)	Speed of the car along the direction of the car's longitudinal axis
d_1	[0, 200] (m)	Distance between the track edge ahead and the car
d_2	$[-\infty, +\infty]$	Distance between the track axis and the car. $\lvert d_2\rvert = 1$ means the car is on the right or left edge of the track, $d_2 = 0$ means the car is right on the track axis, and $\lvert d_2\rvert > 1$ means the car is outside of the track

where the term v represents the car is expected to run as fast as possible so as to maximize the reward. /e terms cos ϕ and (1 − |sin ϕ|) mean ϕ is expected as zero so that the car can run along the track all the time. /e term (1 − $|d_2|$) represents the car is on the track axis. I⟦·⟧ represents an indicator function whose value is 1 or 0 depending on whether the condition is met or not. /e following equation reformulates the first term of equation (14):

$$v \times \left(\frac{1 - |v - \beta|}{\alpha}\right)^{I[\![d_1 \le 10]\!]} = \begin{cases} v \times \left(\frac{1 - |v - \beta|}{\alpha}\right), & d_1 \le 10, \\ v, & d_1 > 10. \end{cases} \tag{15}$$

Equation (15) takes into account the speed constraints of the car whether the car encounters a turn or not. /e car slows down when a turn is encountered and drives as fast as possible along a straight route. Here, $d_1 = 10$ is set to be the threshold of encountering a turn. /e car is at a turn when $d_1 < 10$ and the corresponding reward is a quadratic function with respect to the speed of the car. Note that α and β are hyper parameters needing to fine-tune. Figure 7 illustrates the graph of the quadratic function when α = 120, β = 180. /e quadratic function reaches the maximum value when v = 90.5, which means the expected speed of the car at a turn is 90.5 km/h and the car will decelerate automatically when it encounters a turn.

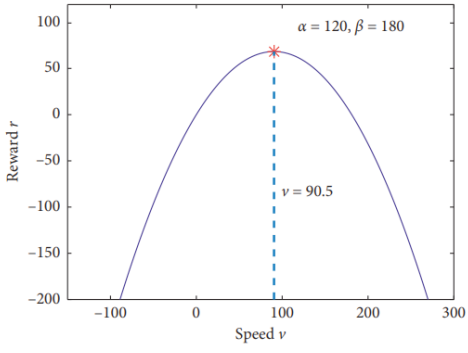


Figure 7. Graph of speed constrained function.

Equation (16) reformulates the last term in equation (14). It restricts the distance between the track edge ahead and the car. This term means that the turn should be observed by the car in advance and the steering angles should be adjusted according to the turn:

$$\left(\frac{|d_1|}{50}\right)^{I[d_1 \le 50]} = \begin{cases} \frac{|d_1|}{50}, & d_1 \le 50, \\ 1, & d_1 > 50. \end{cases} \tag{16}$$

Learning Curve and Training Time

We successfully achieve the optimal self-driving policy with BAMDDPG by aggregating multiple subpolicies in TORCS. During one episode of the training process, one subpolicy is selected. /e corresponding agent perceives the driving environment state through various sensors and executes the action by following the selected subpolicy. Table 3 presents the detailed description of the action commands, including steering, brake, and throttle.

After the interaction, all subpolicies were updated using the minibatch from the centralized experience replay buffer. We have argued that less training time is demanded by BAMDDPG than DDPG. Figure 8(a) illustrates the comparison of learning curve between BAMDDPG and DDPG while Figure 8(b) demonstrates the comparison of training time.

In our experiments on TORCS, the simulated car was trained 6000 episodes with the Aalborg track using BAMDDPG and DDPG, respectively. Figure 8(a) illustrates the learning curve comparison of DDPG and BAMDDPG. /e curve depicts the moving average of episode reward while the shaded area depicts the mean ± the standard deviation. Figure 8(a) demonstrates that BAMDDPG and DDPG both converge and oscillate around a specific mean episode reward after being trained 6000 episodes. Figure 8(b) demonstrates that BAMDDPG takes less time to train since the aggregated policy quickly approximates the optimal policy as subpolicies are trained to a certain extent. It takes 22.84 hours for BAMDDPG to be trained 6000 episodes, but 52.77 hours for DDPG, which demonstrates BAMDDPG can cut down the training time by 56.7%.

Figure 8(b) also shows that training time spent by BAMDDPG and DDPG is not so different in the first 1500 episodes. /e reason is that the attention is mostly paid on environment exploring by the simulated car at first and these initial episodes finish quickly. Exploring time spent by BAMDDPG and DDPG is nearly the same. From the perspective of network

training, the first 1500 episodes can be considered as the initialization of the corresponding networks.

4.2.3. Effectiveness of Aggregation

The ability of the BAMDDPG algorithm to reduce training time is based on policy aggregation. Section 3.3 illustrated the conclusion that the performance of the aggregated policy is better than that of subpolicies through theoretical analysis. In addition, Section 4.1 has shown the effectiveness of aggregation on 2D robot arm benchmark. In this section, we are to further illustrate the effectiveness of aggregation on TORCS.

In order to avoid the influence of too many subpolicies on the conciseness and contrast of expression, only three subpolicies are trained by the BAMDDPG algorithm in this experiment. /e trained subpolicies and the aggregated policy control the same simulated car on the same track, Aalborg track, within one lap. Then, we observe the total reward and whether the car can finish one lap on the track or not. Table 4 illustrates the simulated car controlled by the aggregated policy finishes the Aalborg track and gained much larger total reward than subpolicies, but the cars controlled by subpolicies all left the track and are not able to complete the track, which indicates that aggregation technique does increase the performance of subpolicies.

Figure 9 further illustrates the difference in total reward between subpolicies and the aggregated policy. As shown by the real line, the total reward of the aggregated policy is in a steady upward trend as the number of steps increases. However, the total rewards of subpolicy 2 and subpolicy 3 increases steadily in the initial stage and then stops rising because the car pulled out of the track at some point. /e performance of subpolicy 1 is the worst, and its total reward is always the lowest and ultimately remains unchanged due to the car leaving the track.

Effect from Number of Subpolicies

The final policy gained by BAMDDPG is based on the aggregation of subpolicies, but the algorithm does not give specific number of subpolicies. In theory, when there is large enough number of subpolicies, the aggregated policy successfully approximates the optimal policy. However, aggregating a large number of subpolicies is inefficient in consideration of computing and storage resource consumption in practice.

Under the consideration of balancing efficiency and performance, this section explores the appropriate number range of subpolicies through experiment. We choose the numbers of subpolicies within 30 and get the appropriate number of subpolicies by comparing the performance of the aggregated policies with different number of subpolicies. These aggregated policies are tested on the Aalborg track, and we then compare their training time, total reward within 5000 steps. Furthermore, we compare the generalization performance of the aggregated policies by testing them on the CG1 track and CG2 track. Experimental results are demonstrated in Figure 10 and Table 5.

Table 3. Description of action commands

Commands	Range	Description
Steering	[−1, +1]	−1 is full right while +1 is full left
Brake	[0, 1]	Brake pedal (1 is full brake while 0 is no brake)
Throttle	[0, 1]	Gas pedal (1 is full gas while 0 is no gas)

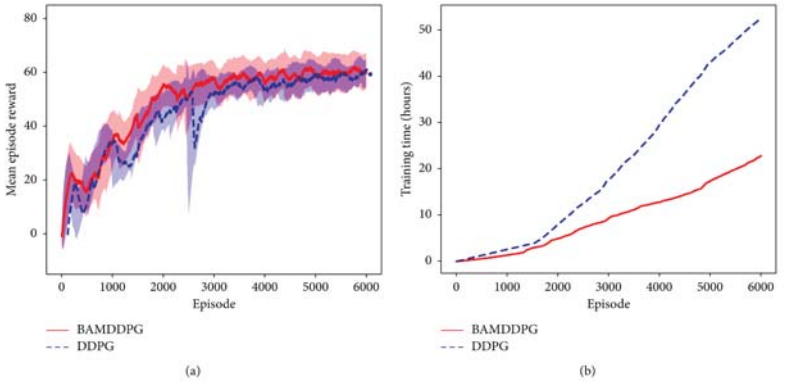


Figure 8. (a) Learning curve and (b) training time comparison of BAMDDPG and DDPG.

Table 4. Performance comparison of the aggregated policy and subpolicies

Policy	Steps	Total reward (points)	Complete one lap
Subpolicy1	246	16690.60	No
Subpolicy2	246	15413.12	No
Subpolicy3	102	−1252.46	No
Aggregated policy	457	31603.37	Yes

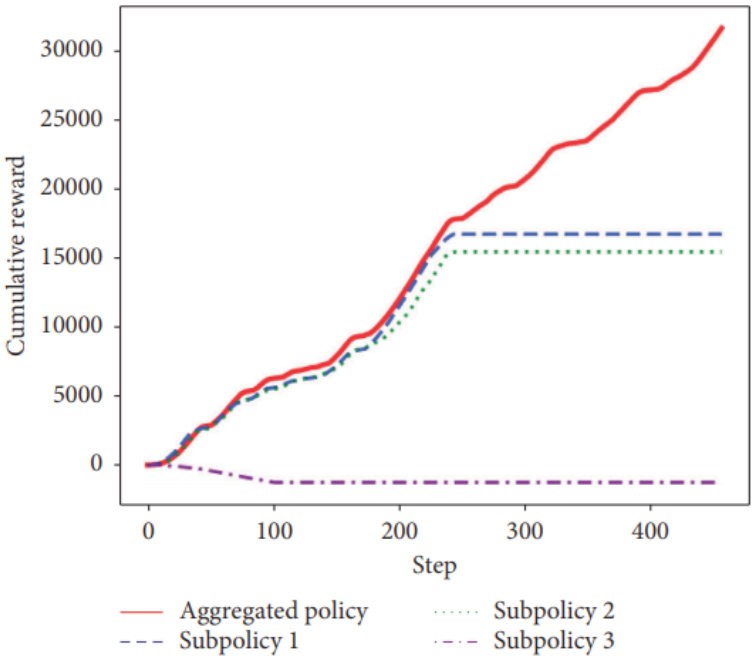


Figure 9. Performance comparison of the aggregated policy and subpolicies.

Figure 10 illustrates the comparison of total reward gained by aggregated policies with different number of subpolicies on the Aalborg track. Since the episode of TORCS may not terminate, we set the maximum number of steps to be 5000 in one episode. /e aggregated policies with 3–10 subpolicies are able to reach the maximum number of steps while others terminate early in one episode. Therefore, they gained much larger reward than those aggregated policies with over 10 subpolicies.

Table 5 demonstrates, for policies aggregating from different numbers of subpolicies within 30, no large difference appears in training time, but the performances of different policies vary from each other. /e policies aggregating from 3 to 10 subpolicies can achieve the maximum interaction number of 5000 steps on the Aalborg track, complete the training Aalborg track with larger total reward than the aggregated policies with over 10 subpolicies, and pass the test track CG1 and CG2 safely.

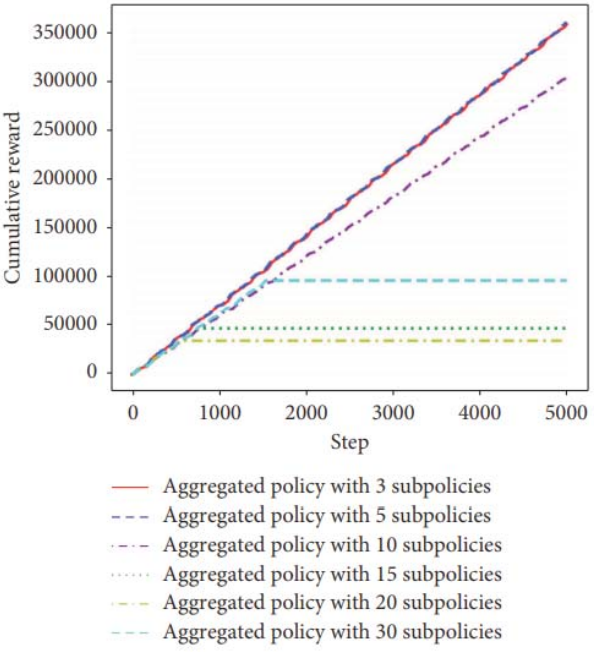


Figure 10. Reward comparison of aggregated policies with different numbers of subpolicies on Aalborg.

Table 5. Comparison of aggregated policies with different numbers of subpolicies

Number of subpolicies	Training time (hours)	Total steps	Total reward	Pass Aalborg	Pass CG1	Pass CG2
3	22.84	5000	331086.10	Yes	Yes	Yes
5	24.40	5000	360804.43	Yes	Yes	Yes
10	24.16	5000	303678.65	Yes	Yes	Yes
15	22.09	771	47121.87	Yes	No	Yes
20	20.49	567	34343.05	Yes	No	Yes
30	21.74	1541	97146.37	Yes	No	Yes

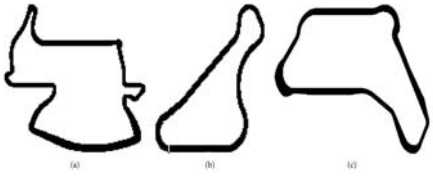


Figure 11. Maps of training and test tracks. (a) Aalborg; (b) CG1; (c) CG2.

Generally speaking, when the number of subpolicies is 3–10, the corresponding aggregated policies perform well and have better

generalization performance than the aggregated policies with over 10 subpolicies, which means 3–10 is the appropriate number of subpolicies for BAMDDPG in practical application.

However, the aggregated policies with over 10 subpolicies cannot reach the maximum steps on the Aalborg track and are not able to finish the CG1 track. /e reason why the aggregated policies with over 10 subpolicies performed worse mainly lies in the limit of the centralized experience replay buffer. During the training time, we fixed the size of the centralized experience replay buffer to 100, 000 transition tuples (s_t, a_t, r_t, $_{st+1}$), by considering the feasibility and efficiency of implementation. However, this buffer could not manage to share all experiences with more than 10 subpolicies. As a result, the aggregated policies with over 10 subpolicies gained less knowledge and performed not well. /e experiment with a larger buffer size will display a better performance with aggregation of 10 subpolicies. But the memory setting has a nonmonotonic effect on the reinforcement learning (RL) performance [26]. /e influence of the memory setting in RL arises from the trade-off between the correct weight update direction and the wrong direction.

Table 6. Generalization performance of the aggregated policy

Track name	Total reward (points)	Complete
Aalborg	30007.61	Yes
CG1	23755.62	Yes
CG2	35602.09	Yes

Generalization Performance

Generalization performance is a research hotspot in the field of machine learning, and it is also a key evaluation index for the performance of algorithms. An overtrained model often performs well in the training set, while it performs poorly in the test set. In our experiments, self-driving policies are learned successfully on the Aalborg track using BAMDDPG. /e car controlled by these policies has good performance on the training track. However, the generalization performance of the learned policies is not known. Hence, we test the performance of the aggregated policy learned with BAMDDPG on both the training and test tracks, including Aalborg, CG1, and CG2, whose maps are illustrated in Figure 11.

The total reward of the aggregated policy shown in Table 6 differs in different tracks since the length of different tracks is not the same. On a long

track, the car travels for a longer time, and the total reward will be larger. In our experiment, route CG2 is the longest and CG1 is the shortest.

Table 6 illustrates that the car controlled by the aggregated policy passes the test tracks successfully. It demonstrates that the learned aggregated policy from BAMDDPG achieves a good generalization performance.

CONCLUSIONS

This paper proposed a deep reinforcement learning algorithm, by aggregating multiple deep deterministic policy gradient algorithm and an m-out-of-n bootstrap sampling method. /is method is effective to the sequential and iterative training data, where the data exhibit long-tailed distribution, rather than the norm distribution implicated by the i.i.d. data assumption. The method can learn the optimal policies with much less training time for tasks with continuous space of actions and states.

Experiment results on the 2D robot arm game show that the reward gained by the aggregated policy is 10%~50% better than those gained by the nonaggregated subpolicies. Experiment results on TORCS demonstrate the proposed method can learn successful control policies with less training time by 56.7%, compared to the normal sampling method and nonaggregated subpolicies.

REFERENCES

1. R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, MIT press, Cambridge, MA, USA, 1998.
2. K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: a brief survey," IEEE Signal Processing Magazine, vol. 34, no. 6, pp. 26–38, 2017.
3. A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in Proceedings of the 25th International Conference on Neural Information Processing Systems, pp. 1097–1105, Lake Tahoe, NV, USA, March 2012.
4. Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, 2015.
5. V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529–533, 2015.
6. H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-Learning," in Proceedings of the 30th AAAI Conference on Artificial Intelligence, Phoenix, AZ USA, March 2016.
7. T. Schaul, J. Quan, I. Antonoglou et al., "Prioritized experience replay," 2015, https://arxiv.org/abs/1511.05952.
8. Z. Wang, T. Schaul, M. Hessel et al., "Dueling network architectures for deep reinforcement learning," in Proceedings of the 33rd International Conference on Machine Learning, vol. 4, pp. 2939–2947, New York, NY, USA, 2016.
9. D. Silver, G. Lever, N. Heess et al., "Deterministic policy gradient algorithms," in Proceedings of the 31st International Conference on Machine Learning, pp. 387–395, Bejing, China, June 2014.
10. T. P. Lillicrap, J. J. Hunt, A. Pritzel et al., "Continuous control with deep reinforcement learning," Computer Science, vol. 8, no. 6, p. A187, 2015.
11. Z. Zheng, C. Yuan, Z. Lin et al., "Self-adaptive double bootstrapped DDPG," in Proceedings of the 27th International Joint Conference on Artificial Intelligence, pp. 3198–3204, Stockholm, Sweden, July 2018.
12. S. Gu, T. Lillicrap, Z. Ghahramani et al., "Q-prop: sampleefficient policy gradient with an off-policy critic," in Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, May 2017.

13. V. Mnih, A. P. Badia, M. Mirza et al., “Asynchronous methods for deep reinforcement learning,” in Proceedings of the International Conference on Machine Learning, pp. 1928–1937, San Juan, PR, USA, May 2016.
14. B. Efron and R. J. Tibshirani, An Introduction to the Bootstrap, CRC Press, Boca Raton, FL, USA, 1994.
15. I. Osband, C. Blundell, A. Pritzel et al., “Deep exploration via bootstrapped DQN,” in Proceedings of the Advances in Neural Information Processing Systems, pp. 4026–4034, Barcelona, Spain, December 2016.
16. Z. Yang, K. E. Merrick, H. A. Abbass et al., “Multi-task deep reinforcement learning for continuous action control,” in Proceedings of the 26th International Joint Conference on Artificial Intelligence, pp. 3301–3307, Melbourne, Australia, August 2017.
17. R. Lowe, Y. Wu, A. Tamar et al., “Multi-agent actor-critic for mixed cooperative-competitive environments,” Advances in Neural Information Processing Systems, 2017.
18. W. Shi, S. Song, and C. Wu, “Soft policy gradient method for maximum entropy deep reinforcement learning,” 2019, https://arxiv.org/abs/1909.03198.
19. R. Davidson and E. Flachaire, “Asymptotic and bootstrap inference for inequality and poverty measures,” Journal of Econometrics, vol. 141, no. 1, pp. 141–166, 2007.
20. H. Ishwaran, L. F. James, and M. Zarepour, “An alternative to the out of bootstrap,” Journal of Statistical Planning and Inference, vol. 139, no. 3, pp. 788–801, 2009.
21. J. Wu and H. Li, “Aggregated multi-deep deterministic policy gradient for self-driving policy,” in Proceedings of 5th International Conference on Internet of Vehicles, vol. 11253, pp. 179–192, Paris, France, November 2018.
22. A. M. F. Mood, Introduction to the 5eory of Statistics, McGraw-Hill Education, New York, NY, USA, 1950.
23. A. Dvoretzky, J. Kiefer, and J. Wolfowitz, “Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator,” 5e Annals of Mathematical Statistics, vol. 27, no. 3, pp. 642–669, 1956.
24. B. Wymann, E. Espie, C. Guionneau et al., ´ Torcs: 5e Open Racing Car Simulator, SourceForge, 2015.

25. D. Loiacono, L. Cardamone, and P. L. Lanzi, Simulated Car Racing Championship: Competition Software Manual, Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria, Milan, Italy, 2013.
26. R. Liu and J. Zou, "/e effects of memory replay in reinforcement learning," in Proceedings of the ICML 2017 Workshop on Principled Approaches to Deep Learning, Sydney, Australia, 2017, https://arxiv.org/pdf/1710.06574.pdf.

CHAPTER 3

DYNAMIC DECISION-MAKING FOR STABILIZED DEEP LEARNING SOFTWARE PLATFORMS

Soohyun Park[1], Dohyun Kim[2] and Joongheon Kim[1]

[1]Korea University, Seoul, Republic of Korea

[2]Naver Webtoon Corporation, Seongnam, Republic of Korea

ABSTRACT

This chapter introduces a dynamic and low-complexity decision-making algorithm which aims at time-average utility maximization in real-time deep learning platforms, inspired by Lyapunov optimization. In deep learning computation, large delays can happen due to the fact that it is computationally expensive. Thus, handling the delays is an important issue for the commercialization of deep learning algorithms. In this chapter, the proposed algorithm observes system delays at first formulated by queue-

Citation: Soohyun Park, Dohyun Kim and Joongheon Kim (September 2nd 2020). Dynamic Decision-Making for Stabilized Deep Learning Software Platforms, Advances and Applications in Deep Learning, Marco Antonio Aceves-Fernandez, IntechOpen, DOI: 10.5772/intechopen.92971.

backlog, and then it dynamically conducts sequential decisionmaking under the tradeoff between utility (i.e., deep learning performance) and system delays. In order to evaluate the proposed decision-making algorithm, the performance evaluation results with real-world data are presented under the applications of super-resolution frameworks. Lastly, this chapter summarizes that the Lyapunov optimization algorithm can be used in various emerging applications.

Keywords:- Lyapunov optimization, stochastic optimization, real-time computing, deep learning platforms, computer vision platforms

INTRODUCTION

Nowadays, many machine learning and deep learning algorithms have been developed in various applications such as computer vision, natural language processing, and so forth. Furthermore, the performances of the algorithms are getting better. Thus, the developments of machine learning and deep learning algorithms become mature. However, the research contributions which are focusing on the real-world implementation of the algorithms are relatively less than the developments of the algorithms themselves. In order to operate the deep learning algorithms in real-world applications, it is essential to think about the real-time computation.

Thus, the consideration of delay handling is desired because deep learning algorithm computation generally introduces large delays [1].In communications and networks research literature, there exists a well-known stochastic optimization algorithm which is for utility function maximization while maintaining system stability.

Here, the stability is modeled with queue, and then the algorithm aims at the optimization computation while stabilizing the queue dynamics. In order to formulate the stability, the queue is mathematically modeled with Lyapunov drift [2].This algorithm is designed inspired by Lyapunov control theory, and thus, it is named to Lyapunov optimization theory [2]. In this chapter, the basic theory, examples, and discussions of the Lyapunov optimization theory are presented. Then, the use of Lyapunov optimization theory for real-time computer vision and deep learning platforms is discussed. Furthermore, the performance evaluation results with real-world deep learning framework computation (e.g., real-world image super-resolution computation results with various models) are presented in various aspects. Finally, the emerging applications will be introduced.

STABILIZED CONTROL FOR RELIABLE DEEP LEARNING PLATFORMS

In this section, Lyapunov optimization theory which is for time-average optimization subject to stability is introduced at first (refer to Section 2.1), and then example-based explanation is presented (refer to Section 2.2). Finally, related discussions are organized (refer to Section 2.3).

Theory

In this section, we introduce the Lyapunov optimization theory which aims at time-average penalty function minimization subject to queue stability. Notice that the time-average penalty function minimization can be equivalently converted to time-average utility function maximization. The Lyapunov optimization theory can be used when the tradeoff exists between utility and stability. For example, it can be obviously seen that the tradeoff exists when current decision-making is optimal in terms of the minimization of penalty function, whereas the operation of the decision takes a lot of time, i.e., thus it introduces delays (i.e., queue-backlog increases in the system). Then, the optimal decision can be dynamically time-varying because focusing on utility maximization (i.e., penalty function minimization) is better when the delay in the current system is not serious (i.e., queueing delay is small or marginal). On the other hand, the optimal decision will be for the delay reduction when the delay in the current system is large. In this case, the decision should be for delay reduction while sacrificing certain amounts of utility maximization (or penalty function minimization).

Suppose that our time-average penalty function is denoted by P (α [t]) and it should be minimized and our control action decision-making is denoted by α [t]. Then, the queue dynamics in the system, i.e., Q[t], can be formulated as follows:

$$Q[t+1] = \max\{Q[t] + a(\alpha[t]) - b(\alpha[t]), 0\} \quad (1)$$

$$Q[0] = 0 \quad (2)$$

where $a(\alpha[t])$ is an arrival process at $Q[t]$ at t when our control action decision-making is $\alpha[t]$. In (1), $b(\alpha[t])$ t is a departure/service process at $Q[t]$ when our control action decision-making is $\alpha[t]$ at t.

In this section, control action decision-making should be made in each unit time for time-average penalty function minimization subject to queue stability. Then, the mathematical program for minimizing time-average

penalty function, P(α[t]) where the control action decision-making at t is α[t], can be presented as follows:

$$\min : \lim_{t\to\infty} \sum_{\tau=0}^{t-1} P(\alpha[\tau]) \tag{3}$$

Subject to queue stability:

$$\lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} Q[\tau] < \infty. \tag{4}$$

In (3), P(α[t]) stands for the penalty function when a control action decision-making is α[t] at t.

As mentioned, the Lyapunov optimization theory can be used when tradeoff between utility maximization (or penalty function minimization) and delays exists. Based on this nature, drift-plus-penalty (DPP) algorithm [2, 3, 4] is designed for maximizing the time-average utility subject to queue stability. Here, the Lyapunov function is defined as $L(Q[t]) = \frac{1}{2}(Q[t])^2$, and let Δ(.) be a conditional quadratic Lyapunov function which is formulated as E[L(Q[t+1])−L(Q[t])|Q[t]], which is called as the drift on t. According to [2], this dynamic policy is designed to achieve queue stability by minimizing an upper bound of our considering penalty function on DPP which is given by

$$\Delta(Q[t]) + V\mathbb{E}[P(\alpha[t])]. \tag{5}$$

where V is a tradeoff coefficient. The upper bound on the drift of the Lyapunov function at t is derived as follows:

$$L(Q[t+1]) - L(Q[t]) = \frac{1}{2}\left(Q\left([t+1]^2 - Q[t]^2\right)\right) \tag{6}$$

$$\leq \frac{1}{2}\left(a(\alpha[t])^2 + b(\alpha[t])^2\right) + Q[t](a(\alpha[t]) - b(\alpha[t])). \tag{7}$$

Therefore, the upper bound of the conditional Lyapunov drift can be derived as follows:

$$\begin{aligned}\Delta(Q(t)) &= \mathbb{E}[L(Q[t+1]) - L(Q[t])|Q[t]] \\ &\leq C + \mathbb{E}[Q[t](a(\alpha[t]) - b(\alpha[t])|Q[t]],\end{aligned} \tag{8}$$

where C is a constant given by

$$\frac{1}{2}\mathbb{E}\left[a(\alpha[t])^2 + b(\alpha[t])^2|Q[t]\right] \leq C, \tag{9}$$

which supposes that the arrival and departure process rates are upper bounded. Due to the fact that C is a constant, minimizing the upper bound on DPP is as follows:

$$V\mathbb{E}[P(\alpha[t])] + \mathbb{E}[Q[t] \cdot (a(\alpha[t]) - b(\alpha[t]))] \tag{10}$$

Algorithm 1. Stabilized Time-Average Penalty Function Minimization

Initialize:
$t \leftarrow 0$;
$Q[t] \leftarrow 0$;
Decision Action: $\forall \alpha[t] \in \mathcal{A}$
Time-Average Penalty Function Minimization subject to Stability
while $t \leq T$ **do** // T: operation time
    Observe $Q[t]$;

    $\mathcal{T}^* \quad \infty$;
    for $\alpha[t] \in \mathcal{A}$ **do**
        $\mathcal{T} \leftarrow V \cdot P(\alpha[t]) + Q[t] \cdot (a(\alpha[t]) - b(\alpha[t]))$;
        if $\mathcal{T} \leq \mathcal{T}^*$ **then**
            $\mathcal{T}^* \leftarrow \mathcal{T}$;
            $\alpha^*[t+1] \quad \alpha[t]$;
        end if
    end for
end while

Finally, the dynamic control action decision-making α[t] in each unit time t for time-average penalty function P(α[t]) minimization subject to queue stability can be formulated as follows based on the Lyapunov optimization theory:

$$\boxed{\alpha^*[t+1] \quad \arg\min_{\alpha[t] \in \mathcal{A}} [V \cdot P(\alpha[t]) + Q[t] \cdot (a(\alpha[t]) - b(\alpha[t]))]} \tag{11}$$

where A is the set of all possible control actions and α*[t+1] is the optimal control action decision-making for the next time slot.

In order to verify whether (11) works correctly or not, following two example cases can be considerable:

- *Case 1:* Suppose Q[t]≈∞. Then

$$\alpha^*[t+1] \quad \arg\min_{\alpha[t] \in \mathcal{A}} [V \cdot P(\alpha[t]) + Q[t] \cdot (a(\alpha[t]) - b(\alpha[t]))] \tag{12}$$

$$\approx \arg\min_{\alpha[t] \in \mathcal{A}} [a(\alpha[t]) - b(\alpha[t])]. \tag{13}$$

Then, (13) shows that control action decision-making should works as follows, i.e., (i) the arrival process should be minimized, and (ii) the

departure process should be maximized. Both cases are for stabilizing the queue, i.e., it should be beneficial when Q[t]≈∞.

- *Case 2:* Suppose Q[t]=0. Then

$$\alpha^*[t+1] \quad \arg\min_{\alpha[t]\in\mathcal{A}} [V \cdot P(\alpha[t]) + Q[t] \cdot (a(\alpha[t]) - b(\alpha[t]))] \tag{14}$$

$$= \arg\min_{\alpha[t]\in\mathcal{A}} V \cdot P(\alpha[t]). \tag{15}$$

Then, (15) shows that control action decision-making should work for minimizing the given penalty function. This is semantically reasonable because focusing on our main objective is possible because stability does not need to be considered because Q[t]=0.

The pseudo-code of the proposed time-average penalty function minimization algorithm is presented in Algorithm 1. From line 1 to line 3, all variables and parameters are initialized. The algorithm works in each unit time as shown in line 4. In line 5, current queue-backlog Q[t] is observed to be used in (11). From line 7 to line 13, the main computation procedure for (11) is described.

Up to now, the time-average penalty function minimization is considered. Based on the theory, the dynamic control action decision-making α[t] in each unit time t for time-average utility function U(α[t]) maximization subject to queue stability can be formulated as follows:

$$\boxed{\alpha^*[t+1] \quad \arg\max_{\alpha[t]\in\mathcal{A}} [V \cdot U(\alpha[t]) - Q[t] \cdot (a(\alpha[t]) - b(\alpha[t]))]} \tag{16}$$

where A is the set of all possible control actions and α*[t+1] is the optimal control action decision-making for the next time slot.

2.2 Example: multicore scheduling in mobile devices

In this section, the Lyapunov optimization-based stabilized time-average optimization algorithm is introduced with one simple toy model. In this example, dynamic core allocation decision-making algorithm is designed which is for time average energy consumption minimization subject to queue stability.

As illustrated in Figure 1, mobile smartphone is with the processor which is equipped with multiple cores. For example, ARM big.LITTLE processors are with multiple little and big heterogeneous cores.

In this system, the task events will be generated when users generate events, which are denoted by a[t] in Figure 1. Then, the events will be located in the task queue (i.e., Q[t] in Figure 1). Then, the events can be processed by the multicore processor. In this case, if many/more cores are allocated in order to process the events from the queue, the processing can be accelerated which is beneficial in terms of queue stability. However, it is not good in terms of our main objective, i.e., energy consumption minimization. On the other hand, if less cores are allocated, the processing becomes slow which is harmful in terms of queue stability but is beneficial in terms of our main objective, i.e., energy consumption minimization. Finally, the tradeoff can be observed between energy consumption minimization (i.e., our main objective) and stability. Then, it can be confirmed that Lyapunov optimization-based algorithm can be used.

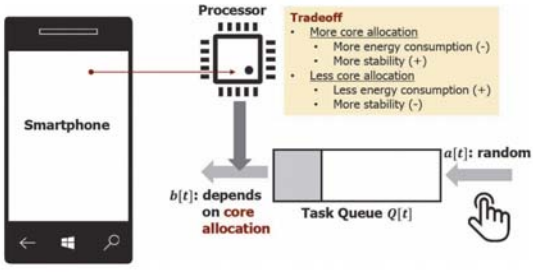


Figure 1. Mobile devices with multicore processors.

In order to design the dynamic core allocation decision-making, α[t] in each unit time t for time-average energy consumption E(α[t]) minimization subject to queue stability can be formulated as follows based on (11):

$$\alpha^*[t+1] \quad \arg\min_{\alpha[t]\in\mathcal{A}} [V \cdot E(\alpha[t]) + Q[t] \cdot (a(\alpha[t]) - b(\alpha[t]))] \tag{17}$$

where A is the set of all possible core allocation combinations and α*[t+1] is the optimal core allocation decision-making for the next time slot. Here, it is obvious that the arrival process is not controllable (i.i.d. random events); thus, it can be ignored. Then, the final form of the dynamic decision-making algorithm can be defined as follows:

$$\alpha^*[t+1] \quad \arg\min_{\alpha[t]\in\mathcal{A}} [V \cdot E(\alpha[t]) - Q[t] \cdot b(\alpha[t])]. \tag{18}$$

In order to check whether the derived Eq. (18) is correct or not, two example cases can be considered, i.e., (i) Q[t]≈∞, and (ii) Q[t]=0:

- Busy queue case (Q[t]≈∞): in this case

$$\alpha^*[t+1] \quad \arg\min_{\alpha[t]\in\mathcal{A}} [V \cdot E(\alpha[t]) - Q[t] \cdot b(\alpha[t])], \tag{19}$$

$$= \arg\min_{\alpha[t]\in\mathcal{A}} [-b(\alpha[t])] = \arg\max_{\alpha[t]\in\mathcal{A}} b(\alpha[t]), \tag{20}$$

Thus, the departure process should be accelerated, i.e., more cores should be allocated. This is semantically true because the fast processing events from the queue is desired if overflow situations happen.

- Busy queue case (Q[t]=0): In this case

$$\alpha^*[t+1] \quad \arg\min_{\alpha[t]\in\mathcal{A}} [V \cdot E(\alpha[t]) - Q[t] \cdot b(\alpha[t])], \tag{21}$$

$$= \arg\min_{\alpha[t]\in\mathcal{A}} V \cdot E(\alpha[t]), \tag{22}$$

Thus, less cores should be allocated for energy consumption minimization which is our main objective. This is semantically true because the given main objective should be desired if the system is stable, i.e., Q[t]=0.

As discussed with examples, the proposed Lyapunov optimization-based dynamic core allocation decision-making algorithm works as desired.

Discussions in stabilized control

The proposed dynamic super-resolution model selection algorithm is beneficial in various aspects, as follows.

Hardware/system-independent self-adaptation

Suppose that this proposed algorithm is implemented in supercomputer-like high-performance computing machines. In this case, the processing should be fast; thus, the queue-backlog is always low. Therefore, the system has more chances to focus on our main objective, i.e., penalty function minimization or utility function maximization. On the other hand, if the hardware itself is performance/resource limited (e.g., mobile devices), then the processing speed is also limited due to the low specifications in processors. Thus, the queue-backlog can be frequently busy because it may not be able to process many data with the queue even though it utilizes the fastest model. Therefore,

it can be finally observed that the proposed algorithm is self-adaptive which can adapt depending on the given hardware/system specifications. It automatically adapts the models based on the given hardware/system; thus, it does not require system engineer's trial-and-error tuning. Furthermore, the proposed algorithm is reliable according to the fact that the self-adaptation is for maximizing its utility while maintaining stability.

Low-complexity operation

As shown in Algorithm 1, the computation procedure is iterative for solving closed-form equation, i.e., (11) and (16). Thus, the computational complexity of the proposed algorithm is polynomial time, i.e., O(N), where N is the number of the given control actions. Thus, it guarantees low-complexity operations.

THE USE OF LYAPUNOV OPTIMIZATION FOR DEEP LEARNING PLATFORMS

As explained, the Lyapunov optimization theory is a scalable, self-configurable, low-complexity algorithm which can be used in many applications. In this section, the use of Lyapunov optimization for deep learning and computer platforms is discussed in two different ways, i.e., departure process control (refer to Section 3.1) and arrival process control (refer to Section 3.2). Finally, its related performance evaluation results are presented (refer to Section 3.3).

Lyapunov control over departure processes

As illustrated in Figure 2, stabilized real-time computer vision platforms should be equipped with queues in order to handle bursty traffics. If the queue is busy or near-overflow, the departure process should be accelerated. Thus, the simplest model should be used for reducing the corresponding computation. On the other hand, if the queue is empty, deep learning computation accuracy can be improved with more sophisticate models because we have enough time to conduct the computation. Thus, multiple models are desired in order to select one depending on queue backlog.

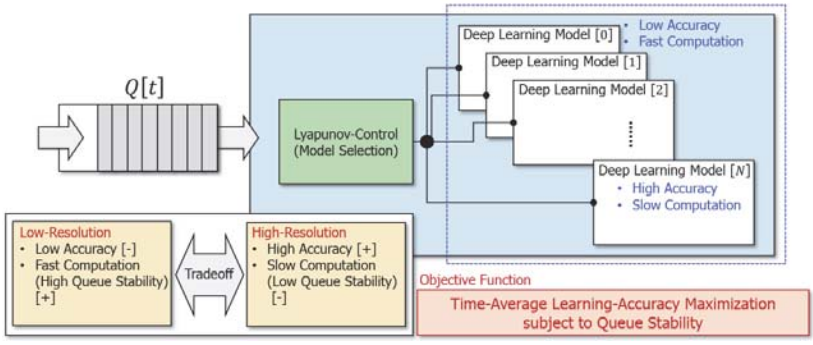


Figure 2. Lyapunov control over departure processes in real-time computer vision platforms for time-average learning accuracy maximization subject to queue stability.

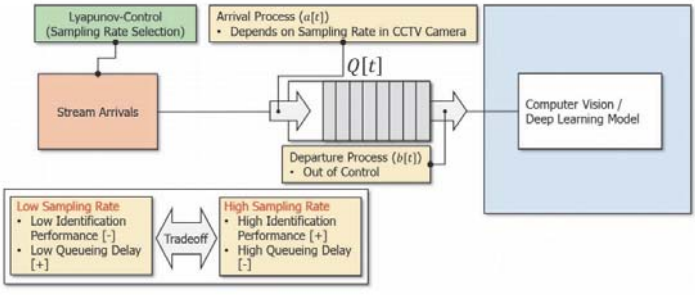


Figure 3. Lyapunov control over arrival processes in real-time computer vision platforms for time-average learning accuracy maximization subject to queue stability.

In Figure 2, multiple models exist, and it can be seen that the simplest model (i.e., low-resolution model) is able to conduct fast computation, but it presents low learning accuracy. On the other hand, the most sophisticate model (i.e., high-resolution model) is good for accurate learning performance, but it introduces computation delays. Thus, the tradeoff exists between performance and delays, i.e., Lyapunov optimization theory-based dynamic model selection decision-making algorithm can be designed as follows:

$$\alpha^*[t+1] \leftarrow \arg\max_{\alpha[t] \in \mathcal{A}} [V \cdot A(\alpha[t]) - Q[t] \cdot (a(\alpha[t]) - b(\alpha[t]))] \tag{23}$$

and this can be reformulated as follows due to the fact that the arrival process is out of control:

$$\alpha^*[t+1] \quad \arg\max_{\alpha[t]\in\mathcal{A}} [V \cdot A(\alpha[t]) + Q[t] \cdot b(\alpha[t])] \tag{24}$$

where A(α[t]) stands for the learning-accuracy when the model selection decision is α[t] at t. Here, A is the set of all possible deep learning models, and α∗[t+1] is the optimal control action decision-making for next time slot.

Lyapunov control over arrival processes

The stabilized real-time computer vision platform in Section 3.1 is novel and scalable; however it has burden because multiple deep learning models should be implemented in a single platform.

Thus, a new dynamic control algorithm with a single deep learning model is also needed for resource-limited systems. As illustrated in Figure 3, our considering system has a single computer vision and deep learning model in computing platforms. In addition, the queue is in front of the system. Thus, the departure process is not controllable anymore. In this case, the arrival process should be controllable in order to control the queue dynamics for stability. Therefore, the arrival image/video streams should be controlled by handling sample rates. If high-frequency sampling is available, more signals will be generated, and then the results will be enqueued. Thus, the arrival process increases. This is beneficial because it increases computer vision performance due to the fact that more images/videos can be obtained especially in surveillance applications. On the other hand, i.e., if low-frequency sampling is conducted, the computer vision performance can be degraded, whereas the number of arrival process data decreases which is beneficial in terms of stability. Eventually, the tradeoff between computer vision performance and delays can be observed. Finally, Lyapunov optimization theory-based sampling rate selection decision-making algorithm can be designed as follows:

$$\alpha^*[t+1] \quad \arg\max_{\alpha[t]\in\mathcal{A}} [V \cdot A(\alpha[t]) - Q[t] \cdot (a(\alpha[t]) - b(\alpha[t]))] \tag{25}$$

and this can be reformulated as follows due to the fact that the departure process is out of control:

$$\alpha^*[t+1] \quad \arg\max_{\alpha[t]\in\mathcal{A}} [V \cdot A(\alpha[t]) - Q[t] \cdot a(\alpha[t])] \tag{26}$$

where A(α[t]) stands for the learning accuracy when the sample rate selection decision is α[t] at t. Here, A is the set of all possible sample rates, and α*[t+1] is the optimal control action decision-making for next time slot.

Performance evaluation and discussions

In this section, the performance evaluation results of the proposed algorithm in Section 3.1 are presented. The data-intensive simulation-based evaluation is performed, and then the results are presented in Figure 4. In addition, Table 1 shows the performance of super-resolution depending on the number of hidden layers. If the number of hidden layers is maximum (i.e., 20 in this research), the PSNR and structural similarity (SSIM, one of the widely used performance metrics in super-resolution) values are maximum. However, the computation times (for CPU-only and CPU-GPU) become slow.

As illustrated in Figure 4, if the models are static (i.e., deep or shallow), the curves show that the two models are not efficient. The deep model cannot handle the overflow situations; thus, the queue diverges. On the other hand, the shallow model is too fast; thus, the queue is always empty. This is obviously positive for stability where the performance in terms of super-resolution performance is the lowest. Thus, it might be better if the algorithm allows certain amounts of delays in order to enhance the quality of super-resolution. The proposed algorithm is initially follows deep model because the queue is idle during the initial phases. If the queue becomes filled with certain amounts of images (i.e., near threshold), it starts the control, i.e., self-adaptive, near the unit time of 5800. Thus, the proposed algorithm starts to select super-resolution models which can handle delays. Thus, it is true that the proposed algorithm is better than the other two static algorithms.

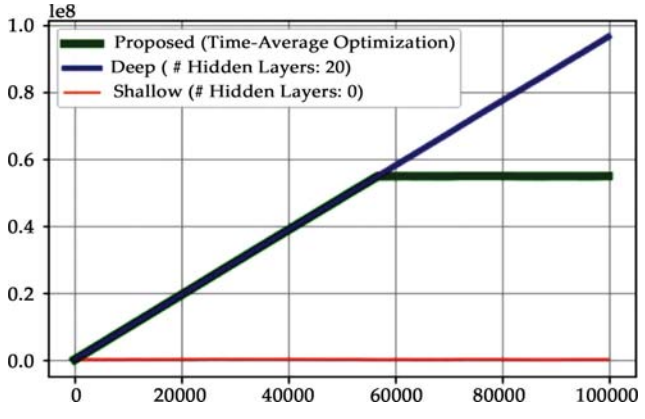


Figure 4. Performance evaluation: Queue-backlog (x-axis, unit time; x-axis, queue occupancy (unit: Bits)).

Table 1. Tradeoff between utility and delay obtained from super-resolution performance measurement results (processing time have measured on 512 768 images)

Depth (# of hidden layers)	0	4	6	8	11	14	17	20
PSNR (dB)	30.400	32.560	33.010	33.229	33.379	33.435	33.495	33.523
SSIM	0.8682	0.9100	0.9160	0.9180	0.9200	0.9200	0.9210	0.9220
Processing time (CPU – only)	0.0020	0.3210	0.5468	0.7725	0.9940	1.3170	1.6220	1.9600
Processing time (CPU + GPU)	0.0010	0.0100	0.0120	0.0152	0.0189	0.0224	0.0262	0.0305

For the proposed self-adaptive stabilized algorithm, the evaluation with two processing capabilities (CPU-only platform vs. CPU-GPU platform), it can be observed that the CPU-GPU platform selects the maximum performance superresolution model (i.e., 20 hidden layers in Table 1) 4:36 times more than the CPU only platform. It means that the proposed algorithm is self-adaptive depending on the hardware/platform requirements. This is obviously beneficial in terms of system engineers because they do not need to conduct trial-and-error-based system parameter tuning anymore.

(a) (b)

(c) (d)

Figure 5. Super-resolution computation results. Note that the model for low-resolution is bicubic which has no hidden layers. (a) Image #1 (low-resolution), (b) image #1 (high-resolution), (c) image #2 (low-resolution), (d) image #2 (high-resolution), (e) image #3 (low-resolution) and (f) image #3 (high-resolution).

In order to confirm the performance of super-resolution models, Figure 5 shows the super-resolution computation results with real-world images. As can be seen in the figures, the super-resolution models show better performances if they have more hidden layers, as shown in Figure 5b, Figure 5d, and Figure 5f. For the superresolution computation without hidden layers, this paper uses bicubic interpolation, as shown in Figure 5a, Figure 5c, and Figure 5e. Finally, these results show that our considering Lyapunov control algorithms for adaptive deep learning platforms can make different super-resolution performance depending on queue-backlog size information.

EMERGING APPLICATIONS

As presented, the Lyapunov optimization framework is for time-average utility maximization while achieving queue stability; and this theory is scalable; thus it is widely applicable [2]. Therefore, there exist many applications based on this algorithm as follows.

Adaptive video streaming

Kim et al. [3, 5] design a dynamic control algorithm for time-average streaming quality (i.e., peak-signal-to-noise ratio (PSNR)) maximization subject to transmit buffer stability in wireless video networks. Koo et al. [6, 7] also propose a novel dynamic adaptive streaming over HTTP (DASH)-based mechanism for video streaming quality maximization under the consideration of battery status, LTE data quota, and stability in hybrid LTE and WiFi networks.

Networks

Neely et al. [8] proposed a novel dynamic multi-hop routing algorithm which is for energy-efficient data/packet forwarding in wireless ad hoc and sensor networks subject to queue stability.

Security applications: surveillance monitoring

Mo et al. [9] design a deep learning framework for CCTV-based distributed surveillance applications. In the system, multiple deep learning frameworks exist; and each deep learning model is with its own configurations. In this situation, there exists a tradeoff between complexity and performance. Therefore, the proposed CCTV-based surveillance algorithm adaptively selects a deep learning model depending on queue-backlog in the system for recognition performance maximization subject to CCTV queue stability. Kim et al. [10] also design a novel face identification deep learning frameworks for CCTV-based surveillance platforms. Instead of having multiple deep learning models, this system has one learning system (based on OpenFace open-source software library) and controls the sampling rates of the CCTV camera. Finally, the proposed decision-making algorithm dynamically selects CCTV sampling rates for recognition performance maximization subject to CCTV queue stability.

Others

The application of Lyapunov optimization-based dynamic control algorithm for dynamic reinforcement learning policy design is illustrated in [11]. In addition, the adaptive control algorithms using the Lyapunov optimization framework in stock market pricing and smart grid are introduced in [12, 13].

CONCLUSIONS

This chapter introduces a dynamic control decision-making algorithm, inspired by Lyapunov optimization theory under the situation where the tradeoff between utility/performance and delays exists. Thus, the dynamic decision-making algorithms aim at time-average utility maximization (or penalty minimization) in real-time deep learning platforms. As discussed, the Lyapunov optimization-based algorithms are scalable, hardware/system-independent, self-configurable, and lowcomplexity. Thus, it can be used in various emerging applications such as video streaming, wireless networks, security applications, and smart grid applications.

ACKNOWLEDGEMENTS

This work is supported by the National Research Foundation of Korea (2019R1A2C4070663, 2019M3E4A1080391). J. Kim is a corresponding author (e-mail: joongheon@korea.ac.kr).

REFERENCES

1. Kim D, Kwon J, Kim J. Lowcomplexity online model selection with Lyapunov control for reward maximization in stabilized real-time deep learning platforms. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC '18); 7–10 October, 2018; Miyazaki, Japan: IEEE; 2018. pp. 4363-4368
2. Neely M. Stochastic Network Optimization with Application to Communication and Queueing Systems. Vermont, USA: Morgan & Claypool; 2010
3. Kim J, Caire G, Molisch A. Qualityaware streaming and scheduling for device-to-device video delivery. IEEE/ ACM Transactions on Networking. 2016;24:2319-2331. DOI: 10.1109/ TNET.2015.2452272
4. Choi M, Kim J, Moon J. Adaptive detector selection for queue-stable word error rate minimization in connected vehicle receiver design. IEEE Transactions on Vehicular Technology. 2018;67:3635-3639. DOI: 10.1109/ TVT.2017.2776327
5. Kim J, Meng F, Chen P, Egilmez H, Bethanabhotla D, Molisch A, et al. Demo: Adaptive video streaming for device-todevice mobile platforms. In: Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom '13), 30 September–4 October, 2013; Miami, FL, USA: IEEE; 2013
6. Koo J, Yi J, Kim J, Hoque M, Choi S. REQUEST: Seamless dynamic adaptive streaming over HTTP for multi-homed smartphone under resource constraints. In: Proceedings of the ACM International Conference on Multimedia (MM '17), 23–27 October, 2017; Mountain View, CA, USA: IEEE; 2017
7. Koo J, Yi J, Kim J, Hoque M, Choi S. Seamless dynamic adaptive streaming in LTE/Wi-fi integrated network under smartphone resource constraints. IEEE Transactions on Mobile Computing. 2019;18:1647-1660. DOI: 10.1109/ TMC.2018.2863234
8. Neely M. Energy optimal control for time varying wireless networks. IEEE Transactions on Information Theory. 2006;52:2915-2934. DOI: 10.1109/ TIT.2006.876219
9. Kim J, Mo YJ, Lee W, Nyang D. Dynamic security-level maximization for stabilized parallel deep learning architectures in surveillance applications. In: Proceedings of the IEEE Symposium on Privacy-Aware Computing (PAC '07); 1–3 August, 2017; Washington DC, USA: IEEE; 2017. pp. 192-193

10. Kim D, Kim J, Bang J. A reliable, selfadaptive face identification framework via Lyapunov optimization. In: Proceedings of ACM Symposium on Operating Systems Principles (SOSP) AI Systems Workshop (AISys‘17), 28 October, 2017; Shanghai, China: ACM; 2017
11. Neely M, Supittayapornpong S. Dynamic Markov decision policies for delay constrained wireless scheduling. IEEE Transactions on Automatic Control. 2013;58:1948-1961. DOI: 10.1109/TAC.2013.2256682
12. Neely M. Stock market trading via stochastic network optimization. In: Proceedings of IEEE Conference on Decision and Control (CDC ‘10), 15–17 December, 2010; Atlanta, GA, USA: IEEE; 2010
13. Neely M, Tehrani A, Dimakis A. Efficient algorithms for renewable energy allocation to delay tolerant consumers. In: Proceedings of IEEE International Conference on Smart Grid Communication (SmartGridComm ‘10), 4–6 October, 2010; Gaithersburg, MD, USA: IEEE; 2010

CHAPTER 4

DEEP LEARNING FOR HYPERSPECTRAL DATA CLASSIFICATION THROUGH EXPONENTIAL MOMENTUM DEEP CONVOLUTION NEURAL NETWORKS

Qi Yue[1,2,3] and Caiwen Ma[1]

[1]Xi'an Institute of Optics and Precision Mechanics, CAS, Xi'an 710119, China

[2]University of Chinese Academy of Sciences, Beijing 100039, China

[3]Xi'an University of Posts and Telecommunications, Xi'an 710121, China

ABSTRACT

Classification is a hot topic in hyperspectral remote sensing community. In the last decades, numerous efforts have been concentrated on the classification problem. Most of the existing studies and research efforts are following the conventional pattern recognition paradigm, which is based on

Citation: Qi Yue, Caiwen Ma, "Deep Learning for Hyperspectral Data Classification through Exponential Momentum Deep Convolution Neural Networks", Journal of Sensors, vol. 2016, Article ID 3150632, 8 pages, 2016. https://doi.org/10.1155/2016/3150632.

complex handcrafted features. However, it is rarely known which features are important for the problem. In this paper, a new classification skeleton based on deep machine learning is proposed for hyperspectral data. The proposed classification framework, which is composed of exponential momentum deep convolution neural network and support vector machine (SVM), can hierarchically construct high-level spectral-spatial features in an automated way. Experimental results and quantitative validation on widely used datasets showcase the potential of the developed approach for accurate hyperspectral data classification.

INTRODUCTION

Recent advances in optics and photonics have allowed the development of hyperspectral data detection and classification, which is widely used in agriculture [1], surveillance [2], environmental sciences [3, 4], astronomy [5, 6], and mineralogy [7]. In the past decades, hyperspectral data classification methods have been a hot research topic. A lot of classical classification algorithms, such as *k*-nearest neighbors, maximum likelihood, parallelepiped classification, minimum distance, and logistic regression (LR) [8, 9], have been proposed. However, there are several critical problems in the classification of hyperspectral data: (1) high dimensional data, which would lead to curse of dimensionality; (2) limited number of labeled training samples, which would lead to Hughes effect; (3) large spatial variability of spectral signature [10].

Most of the existing work, concerning the classification of hyperspectral data, follows the conventional paradigm of pattern recognition and complex handcrafted features extraction from the raw data and classifiers training. Classical feature extraction methods include the following: principle component analysis, singular value decomposition, projection pursuit, self-organizing map, and fusion feature extraction method. Many of these methods extract features in a shallow manner, which do not hierarchically extract deep features automatically. In contrast, the deep machine learning framework can extract high-level abstract features, which has rotation, scaling, and translation invariance characteristics [11, 12].

In recent years, the deep learning model, especially the deep convolution neural network (CNN), has been shown to yield competitive performance in many fields including classification or detection tasks which involve image [13–15], speech [16], and language [17]. However, most of the CNN network input data are original image without any preprocessing based

on the prior knowledge. Such manner directly extends the CNN network training time and the feature extraction time [18, 19]. Besides, the traditional CNN network has too many parameters, which is difficult to initialize. And the training algorithm based on gradient descend technique may lead to entrapment in local optimum and gradient dispersion. Moreover, there is little study on the convergence rate and smoothness improvement of CNN at present.

In this paper, we propose an improved hyperspectral data classification framework based on exponential momentum deep convolution neural network (EM-CNN). And an innovative method for updating parameters of the CNN on the basis of exponential momentum gradient descendent is proposed aiming at the problem of gradient diffusion of deep network.

The rest of the paper is organized into four sections. Section 2 describes the feature learning and deep learning. The proposed EM-CNN framework is introduced in Section 3, while Section 4 details the new way of exponential momentum gradient descent method, which yields the highest accuracy compared with homologous parameters momentum updating methods. Section 5 is the experiment results. Section 6 summarizes the results and draws a general conclusion.

FEATURE LEARNING

Feature extraction is necessary and useful in the real-world for that the data such as images, videos, and sensor measurement data is usually redundant, highly variable, and complex. Traditional handcrafted feature extraction algorithms are time-consuming and laborious and usually rely on the prior knowledge of certain visual task. In contrast, feature learning allows a machine to both learn at a specific task and learn the features themselves.

Deep learning is part of a broader family of machine learning based on learning representations of data. It attempts to model high-level abstractions in data by using a deep graph with multiple processing layers, composed of multiple linear and nonlinear transformations. Typical deep learning models include autoencoder (AE) [20], deep restricted Boltzmann machine (DRBM) [21], deep Boltzmann machines (DBM) [22], deep belief networks (DBN) [23], stacked autoencoder (SAE) [24], and deep convolutional neural networks (DCNN) [25].

The deep convolution neural network (DCNN), a kind of neural network, is an effective method for feature extraction, which can potentially lead to

progressively more abstract and complex features at higher layers, and the learnt features are generally invariant to most local changes of the input. It has been shown to yield competitive performance in many fields, such as object detection [13–15], speech simultaneous interpretation [16], and language classification [17]. As the performance of classification highly depends on the features [26], we adopt deep convolution neural network (DCNN) as the part of our hyperspectral data classification framework.

STRUCTURE DESIGN OF HYPERSPECTRAL DATA CLASSIFICATION FRAMEWORK

In deep convolutional neural network, input data, convolution kernel, and threshold parameter are the three most important issues [27–29]. The input data is the basis of feature extraction, which determines the final classification performance.

The size of the convolution kernel determines the degree of abstraction of the feature. If convolution kernel size is too small, the effective local features are difficult to extract. Otherwise, the extraction feature would exceed the feature range that convolution kernel can express. Threshold parameter is mainly used to control the degree of response of characteristic submode. Besides, the network depth and dimension of output layer can also influence the quality of feature extraction.

The deeper network layers indicate stronger feature expression ability, while they would lead to overfitting and poor real-time ability. The dimension of output layer directly determines the convergence speed of network.

When the sample sets are limited, over lower dimension of the output layer cannot guarantee the validity of features, while over higher feature of the output layer will produce feature redundancy.

Since the traditional CNN input the original image directly into the deep network and the input data play a crucial part in the final feature extraction [28, 29], three images obtained by image data preprocessing are used as inputs to improve the convergence speed and specific pattern classification performance. In order to obtain better extraction features, the sizes of convolution layer filter are 9×9, 5×5 and 3×3, respectively, and the depth of network is seven according to the results of the experiments.

Besides, the lower sampling applies Max-pooling and the nonlinear mapping function is LREL function, which is shown in the following formula:

$$h^{(i)} = \begin{cases} w^{(i)T}x, & w^{(i)T}x > 0, \\ \varepsilon \times w^{(i)T}x, & w^{(i)T}x \le 0, \end{cases} \quad (1)$$

where ε is nonzero small constant and w is the weight of neuron. The setting of ε ensures that inactive neurons receive a nonzero gradient value, so that the neuron has the possibility of being activated.

Based on the above analysis, a deep network framework for hyperspectral data classification based on deep convolutional neural network is proposed in Figure 1.

In the proposed deep CNN model, the first layer, the third layer, and the fifth layer are convolution layers, which realized feature extraction from lower level to higher level. The second layer, the fourth layer, and the sixth layer are lower sampling layers, used for feature dimension reduction. The final layer is the output layer which is whole connection layer and output of the final extraction features.

EXPONENTIAL MOMENTUM GRADIENT DESCENT ALGORITHM

Error Transfer

Error transmission descends by two steps through forward propagation and reverse gradient, to conduct weight generation and adjustment. Using gradient descent method to update weight is shown in formula (2), and bias updating method is shown in formula (3) [30]:

$$w^l_{\text{new}} = w^l_{\text{old}} + \eta\left(-\frac{\partial E}{\partial w^l_{\text{old}}}\right), \quad (1)$$

$$b^l_{\text{new}} = b^l_{\text{old}} + \eta\left(-\frac{\partial E}{\partial b^l_{\text{old}}}\right). \quad (2)$$

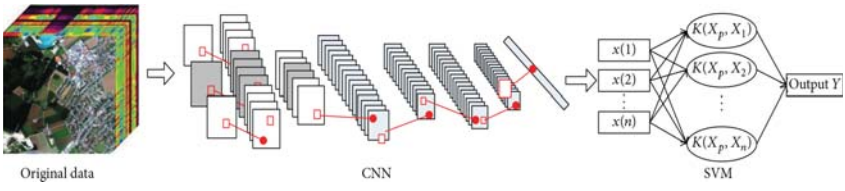


Figure 1. Classification framework based on EM-CNN.

In the formula, η is the learning rate, $-\partial E/\partial w^{l}_{\text{old}}$ is the gradient of error to weight, and $-\partial E/\partial b^{l}_{\text{old}}$ is the gradient of error to bias, namely, the sensitivity of parameter adjustment. In order to achieve weight and bias optimizing, the gradient of error to weight and the gradient of error to bias must be first obtained.

For convolution layer, its output is shown as the following formula:

$$x_j^l = f\left(\sum_{i\in M_j} M_j * K_{ij}^l + b_j\right), \tag{3}$$

where b_j is the bias of jth type of feature diagram, $\square_j$ is the block of input feature diagram, and K_{ij}^l is convolution kernel. According to derivation formula of sensitivity function, the sensitivity of convolution layer can be represented by the following formula:

$$\delta_j^l = \beta_j^{l+1}\text{up}\left(\delta_j^{l+1}\right)\circ f'\left(u^l\right), \tag{4}$$

where β_j^{l+1} is the convolution kernel of l+1 sampling layer, up represents upper sampling, and δ_j^{l+1} is 1/4 of δ_j^l, so upper sampling should be conducted.∘symbol represents the multiplication of corresponding elements

Thus, the gradient of convolution layer error to bias is shown in formula (6). In the formula, (u, V) is the element location of sensitivity matrix:

$$\frac{\partial E}{\partial b^l_{\text{old}}} = \frac{\partial E}{\partial b_j} = \sum_{(u,v)}\left(\delta_j^l\right)_{uv}. \tag{5}$$

The gradient of convolution layer error to weight is shown in formula (7). In the formula, p_i^{l-1} is the convolution block of x_i^{l-1} and convolution kernel Kij, (u, V) is the element location of the block:

$$\frac{\partial E}{\partial w^l_{\text{old}}} = \frac{\partial E}{\partial K_{ij}^l} = \sum_{(u,v)}\left(\delta_j^l\right)_{uv}\left(p_i^{l-1}\right)_{uv}. \tag{6}$$

Substitute formula (5), (6) into formula (1), (2) and obtain the updated value of convolution layer's weight.

The output of sampling layer's neural network can be expressed by formula (8), in which β_j^l and b_j^l, respectively, represent multiplicative bias and additive bias. Multiplicative bias is generally set as 1:

$$x_j^l = f\left(\beta_j^l \text{down}\left(x_j^{l-1}\right) + b_j^l\right). \tag{7}$$

According to the sensitivity of calculating formula of gradient descent, the sensitivity of sampling layer obtained is shown as the following formula:

$$\delta_j^l = \delta_j^{l+1} w_j^{l+1} \circ f'\left(u^l\right). \tag{8}$$

Whereby the bias updating formula of sampling layer can be obtained, as is shown in formula (10). According to formula (3), bias value updating can be obtained:

$$\frac{\partial E}{\partial b_{\text{old}}^l} = \frac{\partial E}{\partial b_j} = \sum_{(u,v)} \left(\delta_j^l\right)_{uv}. \tag{9}$$

Exponential Momentum Training Algorithm

The traditional gradient descent method only transmits gradient error between single layers, which lead to slow convergence rate of the network. Increasing the learning rate η is a good way to improve the convergence speed.

But it not only improves the convergence speed but also causes unstable problem of the network, namely, "oscillation." Faced with this situation, paper [19] proposes the momentum method, which increases the convergence speed by adding momentum factor. Paper [31] proposes the self-adaptive momentum method based on paper [19].

However, neither of these methods considers the relation between oscillation, convergence, and momentum. And the momentum factor does not promote convergence and enhance learning performance.This paper applies error exponential function of gradient to adjust the pace of momentum factor.

The function can increase the momentum factor at the flat region, which can accelerate the network convergence speed and can decrease the momentum factor at the steep region of error curve, which can avoid excessive network convergence. Such method can improve the convergence rate of the algorithm and avoid oscillation of convergence process.

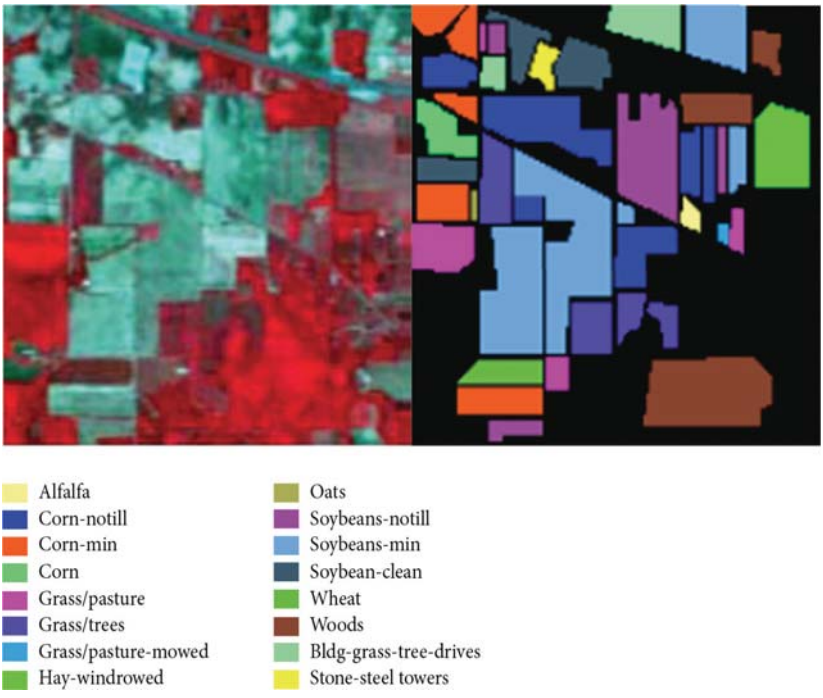


Figure 2. Indian Pines hyperspectral imagery and ground truth of classification.

The updating formula of momentum factor is the following formula:

$$a = \exp\left(-\lambda_1 \left\| D_k \right\| - \lambda_2\right) \frac{D_k}{\left\| \Delta w^{k-1} \right\|^2},$$

$$D_k = -\frac{\partial E}{\partial w\,(k)}. \tag{11}$$

In the formula, $\Delta w^k = w^{k+1} - w$, and Dk represents the gradient of error to weight.

EXPERIMENT AND ANALYSIS

In this section, the performance of the proposed algorithm is evaluated on AVIRIS and ROSIS hyperspectral dataset. The overall accuracy, generalized accuracy, and kappa parameters, the most three important criteria, are used to evaluate the performance of the proposed framework.

Data Description

In our experiments, we experimented and validated the proposed framework with AVIRIS and ROSIS hyperspectral datasets. AVIRIS hyperspectral data 92AV3C was obtained by the AVIRIS sensor in June 1992. ROSIS hyperspectral datasets were gathered by a sensor known as the reflective optics system imaging spectrometer (ROSIS-3) over the city of Pavia, Italy. In particular, we employed the Indian Pines dataset, which depicts Indiana and consists of 145 × 145 data size and 224 spectral bands in the wavelength range 0.4 to 2.510^{-6} meters. It contains a total of 16 categories, as shown in Table 1. Its true mark is shown in Figure 2. The other datasets we employed are the Pavia University datasets, whose number of spectral bands are 102. Nine land cover classes are selected, which are shown in Figure 3. The numbers of samples for each class are displayed in Table 2.

Table 1. Sixteen classes of Indian Pines dataset

Class code	Ground object class	Number of training samples	Number of testing samples
C1	Alfalfa	30	24
C2	Corn-notill	734	700
C3	Corn-min	434	400
C4	Corn	134	100
C5	Grass/Pasture	297	200
C6	Grass/Trees	397	300
C7	Grass/pasture-mowed	16	10
C8	Hay-windrowed	289	200
C9	Oats	10	10
C10	Soybeans-notill	538	430
C11	Soybeans-min	1268	1200
C12	Soybean-clean	314	300
C13	Wheat	112	100
C14	Woods	694	600
C15	Bldg-Grass-Tree-Drives	190	190
C16	Stone-steel towers	50	45

For investigating the performance of the proposed methods, experiments were organized step by step. The influence of the convolution kernel size and the depth of the network on the classification results was first analyzed. Then, we verified the performance of exponential momentum training algorithm. Finally, classifications based on CNN framework were conducted.

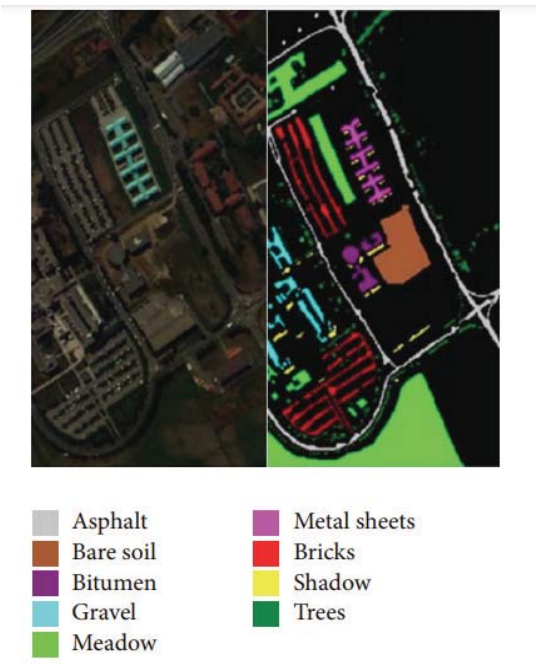


Figure 3. Pavia hyperspectral imagery and ground truth of classification.

Effect of Kernel Size and Depth

The influence of the kernel size and the network depth on the classification performance of the proposed framework is analyzed in this section. The deep convolution neural network is trained by a series of different kernel size and network depth under fixed network structure and algorithm parameters.

The results are shown in Tables 3 and 4. Table 3 suggested that the convolution kernel size is less affected by the overall accuracy of the method, and it better be consistent with the features size of the image data. Table 4 results shows that the deeper structures can get better classification accuracy.

Exponential Momentum Training Algorithm

In this section, we verified the general accuracy and the convergence speed of the algorithm. We select adaptive momentum [31] and elastic momentum [32] as the comparative method to observe the iteration round change of loss

function of training objectives. It can be easily seen from Figure 4 that the convergence point of adaptive momentum is 14, the convergence point of elastic momentum is 8, and the convergence point of exponential momentum is 7. So the convergence of iteration times of exponential momentum is the minimum, and its consumption of the training time is also the minimum.

For the general accuracy test experiment, the LeNet5 neural network [33] and standard multiple neural network [34] are chosen for comparison. The accuracy results obtained are shown in Table 5. It can be seen from the table that, compared with the corresponding training models of the standard momentum and adaptive momentum, the exponential momentum training method can elevate the classification accuracy on different network.

Table 2. Nine classes of Pavia dataset.

Class code	Ground object class	Number of training samples	Number of testing samples
C1	Alfalfa	30	24
C2	Bare soil	734	700
C3	Bitumen	434	400
C4	Gravel	134	100
C5	Meadows	297	200
C6	Metal sheets	397	300
C7	Bricks	16	10
C8	Shadow	289	200
C9	Trees	10	10

Table 3. Accuracy comparison of different kernel size

Kernel size	3	7	9	11	15	17
Accuracy	35.2	95.2	98.7	98.2	90.5	50.2

Table 4. Accuracy comparison of different depth

Depth	3	4	5	6	7	9
Accuracy	40.5	85.3	89.5	94.5	98.8	99.6

Table 5. Accuracy comparison of algorithms' image recognition

Momentum method	EFM-CNN network	LeNet5 neural network	Multiple neural network
Exponential momentum	98.69	98.24	73.03
Elastic momentum	97.65	97.5	72.5
Adaptive momentum	97.10	97.00	65.50

Comparing with Other Methods

Comparing with Other Feature Extraction Methods

We verify the effectiveness of the proposed feature extraction method from the sense of classification, by comparing our algorithm with other classical feature extraction methods, involving principle component analysis- (PCA-) SVM, kernel PCA- (KPCA-) logistic regression (LR), independent component analysis- (ICA-) SVM, nonnegative matrix factorization- (NMF-) LR, and factor analysis- (FA-) SVM

All the logistic regression classifiers are set to have learning rate 0.1 and are iterated on the training data for 8000 epochs. The result is shown in Figure 5. Experiments show that, by combining with SVM, the proposed method outperforms all other feature extraction methods and gets the highest accuracy.

Table 6. Accuracy comparison of different classifier

Datasets	Measurements	SAE-LR		RBP-SVM		EFM-CNN-SVM		PCA-RBP-SVM	
		Spatial	Joint	Spatial	Joint	Spatial	Joint	Spatial	Joint
Pavia	Overall accuracy	0.9514	0.9852	0.9455	0.9845	0.9625	**0.9869**	0.9448	0.9791
	average accuracy	0.9401	0.9732	0.9370	0.9711	0.9522	**0.9790**	0.9281	0.9689
	Kappa coefficient	0.9358	0.9850	0.9289	0.9794	0.9431	**0.9859**	0.9246	0.9758
Indian	Overall accuracy	0.9589	0.9735	0.9564	0.9722	0.9653	**0.9876**	0.9514	0.9701
	average accuracy	0.9475	0.9669	0.9412	0.9615	0.9517	**0.9795**	0.9375	0.9599
	Kappa coefficient	0.9594	0.9750	0.9539	0.9439	0.9605	**0.9862**	0.9547	0.9418

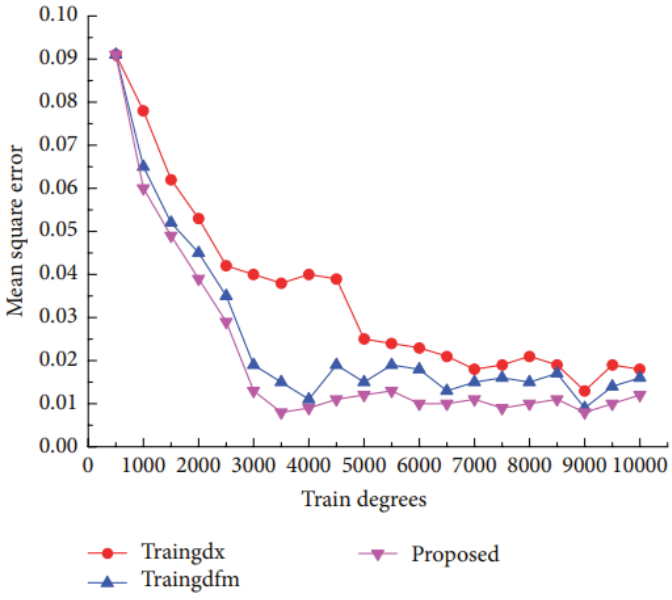


Figure 4. Convergence curve.

Comparing with Other Classification Methods

We examine the classification accuracy of EFM-CNN-SVM framework by comparing proposed framework with spatialdominated methods, such as radial basis function- (RBF-) linear SVM, principle component analysis- (PCA-) RBFSVM, and stacked autoencoder- (SAE-) logistic regression (LR). By putting both the spectral and spatial information together to form a hybrid input and utilizing the deep classification framework detailed in Section 3, we get the highest classification accuracy we have ever attained. The experiments were performed with same parameter settings above 100. The results are shown in Table 6 and Figure 6. From Table 6, we can see that the EFM-CNN-SVM method turns out to be better on all other methods. And the joint features yield higher accuracy than spectral features in terms of mean performance. In Figure 6, we look into the classification accuracy from a visual perspective. It can be seen that classification results of proposed method are closest to the ideal classification results other than RBF-SVM and linear SVM methods.

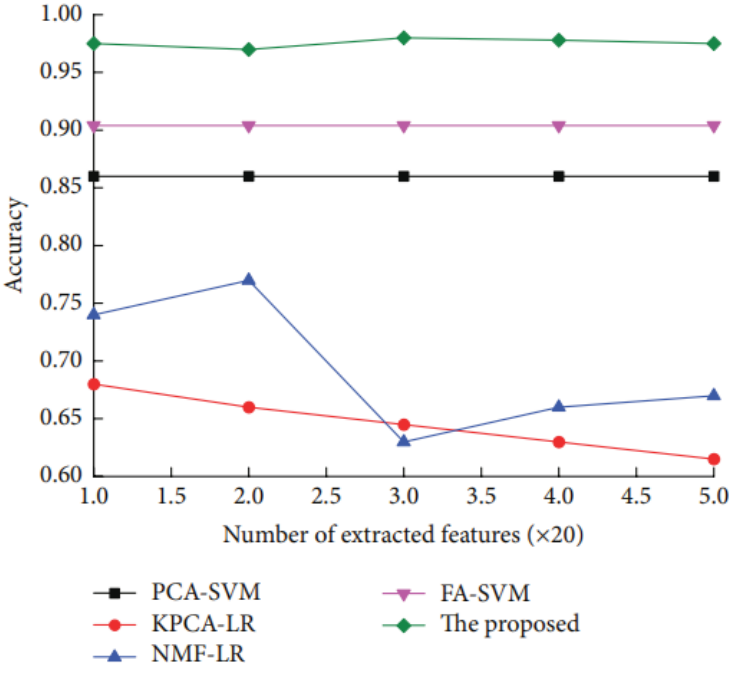


Figure 5. Comparison with other feature extraction methods.

CONCLUSION

In this paper, a hyperspectral data classification framework is proposed based on deep CNN features extraction architecture. And an improved error transmission algorithm, selfadaptive exponential momentum algorithm, is proposed. Experiments results show that the improved error transmission algorithm converged quickly compared to homologous error optimization algorithm such as adaptive momentum and elastic momentum. And proposed EFM-CNN-SVM framework has been proven to provide better performance than PCA-SVM, KPCA-SVM, and SAE-LR frameworks. Our experimental results suggest that deeper layers always lead to higher classification accuracies, though operation time and accuracy are contradictory. It has shown that the deep architecture is useful for classification and the high-level spectral-spatial feature, increasing the classification accuracy. When the data scale is larger, the extracted feature has better recognition ability.

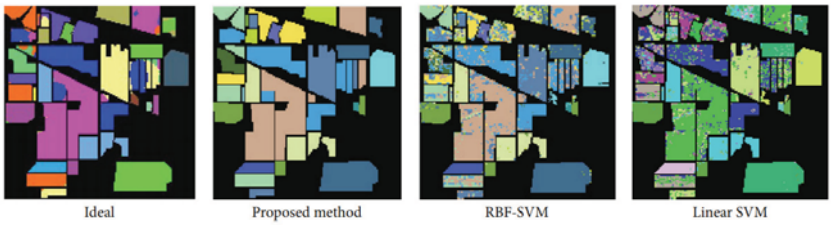


Figure 6. Classification of Indian Pines dataset based on EFM-CNN-SVM.

ACKNOWLEDGMENTS

This work is supported by the National 863 High Tech Research and Development Program (2010AA7080302).

REFERENCES

1. F. M. Lacar, M. M. Lewis, and I. T. Grierson, "Use of hyperspectral imagery for mapping grape varieties in the Barossa Valley, South Australia," in Proceedings of the 2001 International Geoscience and Remote Sensing Symposium (IGARSS '01), pp. 2875–2877, IEEE, Sydney, Australia, July 2001.
2. P. W. T. Yuen and M. Richardson, "An introduction to hyperspectral imaging and its application for security, surveillance and target acquisition," Imaging Science Journal, vol. 58, no. 5, pp. 241–253, 2010.
3. T. J. Malthus and P. J. Mumby, "Remote sensing of the coastal zone: an overview and priorities for future research," International Journal of Remote Sensing, vol. 24, no. 13, pp. 2805–2815, 2003.
4. J. M. Bioucas-Dias, A. Plaza, G. Camps-Valls, P. Scheunders, N. Nasrabadi, and J. Chanussot, "Hyperspectral remote sensing data analysis and future challenges," IEEE Geoscience & Remote Sensing Magazine, vol. 1, no. 2, pp. 6–36, 2013.
5. M. T. Eismann, A. D. Stocker, and N. M. Nasrabadi, "Automated hyperspectral cueing for civilian search and rescue," Proceedings of the IEEE, vol. 97, no. 6, pp. 1031–1055, 2009.
6. E. K. Hege, W. Johnson, S. Basty et al., "Hyperspectral imaging for astronomy and space surviellance," in Imaging Spectrometry IX, vol. 5159 of Proceedings of SPIE, pp. 380–391, January 2004.
7. F. V. D. Meer, "Analysis of spectral absorption features in hyperspectral imagery," International Journal of Applied Earth Observation & Geoinformation, vol. 5, no. 1, pp. 55–68, 2004.
8. S. Rajan, J. Ghosh, and M. M. Crawford, "An active learning approach to hyperspectral data classification," IEEE Transactions on Geoscience and Remote Sensing, vol. 46, no. 4, pp. 1231– 1242, 2008.
9. Q. Lu and M. Tang, "Detection of hidden bruise on kiwi fruit ¨ using hyperspectral imaging and parallelepiped classification," Procedia Environmental Sciences, vol. 12, no. 4, pp. 1172–1179, 2012.
10. G. M. Foody and A. Mathur, "A relative evaluation of multiclass image classification by support vector machines," IEEE Transactions on Geoscience & Remote Sensing, vol. 42, no. 6, pp. 1335– 1343, 2004.
11. Y. Chen, X. Zhao, and X. Jia, "Spectral-spatial classification of hyperspectral data based on deep belief network," IEEE Journal of

Selected Topics in Applied Earth Observations & Remote Sensing, vol. 8, no. 6, pp. 2381–2392, 2015.

12. Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, "Deep learningbased classification of hyperspectral data," IEEE Journal of Selected Topics in Applied Earth Observations & Remote Sensing, vol. 7, no. 6, pp. 2094–2107, 2014.
13. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS '12), pp. 1097–1105, Lake Tahoe, Nev, USA, December 2012.
14. G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," Science, vol. 313, no. 5786, pp. 504–507, 2006.
15. Z. Zhu, C. E. Woodcock, J. Rogan, and J. Kellndorfer, "Assessment of spectral, polarimetric, temporal, and spatial dimensions for urban and peri-urban land cover classification using Landsat and SAR data," Remote Sensing of Environment, vol. 117, pp. 72–82, 2012.
16. D. Yu, L. Deng, and S. Wang, "Learning in the deep structured conditional random fields," in Proceedings of the Neural Information Processing Systems Workshop, pp. 1–8, Vancouver, Canada, December 2009.
17. A.-R. Mohamed, T. N. Sainath, G. Dahl, B. Ramabhadran, G. E. Hinton, and M. A. Picheny, "Deep belief networks using discriminative features for phone recognition," in Proceedings of the 36th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '11), pp. 5060–5063, Prague, Czech Republic, May 2011.
18. B. H. M. Sadeghi, "A BP-neural network predictor model for plastic injection molding process," Journal of Materials Processing Technology, vol. 103, no. 3, pp. 411–416, 2000.
19. [19] P. Baldi and K. Hornik, "Neural networks and principal component analysis: learning from examples without local minima," Neural Networks, vol. 2, no. 1, pp. 53–58, 1989.
20. Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in Proceedings of the 20th Annual Conference on Neural Information Processing Systems (NIPS '06), pp. 153–160, Cambridge, Mass, USA, December 2006.

21. G. E. Hinton, “Apractical guide to training restricted Boltzmann machines,” Tech. Rep. UTML TR2010-003, Department of Computer Science, University of Toronto, Toronto, Canada, 2010.
22. R. Salakhutdinov and G. E. Hinton, “Deep Boltzmann machines,” in Proceedings of the International Conference on Artificial Intelligence and Statistics, pp. 448–455, Clearwater Beach, Fla, USA, April 2009.
23. G. E. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” Neural Computation, vol. 18, no. 7, pp. 1527–1554, 2006.
24. Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, “Greedy layer-wise training of deep networks,” in Proceedings of the Neural Information Processing Systems, pp. 153–160, Cambridge, Mass, USA, 2007.
25. M. D. Zeiler and R. Fergus, “Stochastic pooling for regularization of deep convolutional neural networks,” https://arxiv.org/ abs/1301.3557.
26. H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio, “An empirical evaluation of deep architectures on problems with many factors of variation,” in Proceedings of the 24th International Conference on Machine Learning (ICML ’07), pp. 473–480, Corvallis, Ore, USA, June 2007.
27. Y. Bengio, G. Guyon, V. Dror et al., “Deep learning of representations for unsupervised and transfer learning,” in Proceedings of the Workshop on Unsupervised & Transfer Learning, Bellevue, Wash, USA, July 2011.
28. Y. Bengio, A. Courville, and P. Vincent, “Representation learning: a review and new perspectives,” IEEE Transactions on Pattern Analysis & Machine Intelligence, vol. 35, no. 8, pp. 1798– 1828, 2013.
29. W. Ouyang and X. Wang, “Joint deep learning for pedestrian detection,” in Proceedings of the 14th IEEE International Conference on Computer Vision (ICCV ’13), pp. 2056–2063, December 2013.
30. N. B. Karayiannis, “Reformulated radial basis neural networks trained by gradient descent,” IEEE Transactions on Neural Networks, vol. 10, no. 3, pp. 657–671, 1999.
31. S. S. Agrawal and V. Yadava, “Modeling and prediction of material removal rate and surface roughness in surface-electrical discharge diamond grinding process of metal matrix composites,” Materials and Manufacturing Processes, vol. 28, no. 4, pp. 381– 389, 2013.

32. W. Tan, C. Zhao, H. Wu, and R. Gao, “A deep learning network for recognizing fruit pathologic images based on flexible momentum,” Nongye Jixie Xuebao/Transactions of the Chinese Society for Agricultural Machinery, vol. 46, no. 1, pp. 20– 25, 2015.
33. N. Yu, P. Jiao, and Y. Zheng, “Handwritten digits recognition base on improved LeNet5,” in Proceedings of the 27th Chinese Control and Decision Conference (CCDC ’15), pp. 4871–4875, May 2015.
34. D. Shukla, D. M. Dawson, and F. W. Paul, “Multiple neuralnetwork,” IEEE Transactions on Neural Networks, vol. 10, no. 6, pp. 1494–1501, 1999.

CHAPTER 5

ENSEMBLE NETWORK ARCHITECTURE FOR DEEP REINFORCEMENT LEARNING

Xi-liang Chen , Lei Cao , Chen-xi Li, Zhi-xiong Xu, and Jun Lai

Institute of Command Information System, PLA University of Science and Technology, No. 1, Hai Fu Road, Guang Hua Road, Qin Huai District, Nanjing City, Jiangsu Province 210007, China

ABSTRACT

The popular deep Q learning algorithm is known to be instability because of the Q-value's shake and overestimation action values under certain conditions. These issues tend to adversely affect their performance. In this paper, we develop the ensemble network architecture for deep reinforcement learning which is based on value function approximation. The temporal ensemble stabilizes the training process by reducing the variance of target approximation error and the ensemble of target values

Citation: Xi-liang Chen, Lei Cao, Chen-xi Li, Zhi-xiong Xu, Jun Lai, "Ensemble Network Architecture for Deep Reinforcement Learning", Mathematical Problems in Engineering, vol. 2018, Article ID 2129393, 6 pages, 2018. https://doi.org/10.1155/2018/2129393.

reduces the overestimate and makes better performance by estimating more accurate Q-value. Our results show that this architecture leads to statistically significant better value evaluation and more stable and better performance on several classical control tasks at OpenAI Gym environment.

INTRODUCTION

Reinforcement learning (RL) algorithms [1, 2] are very suitable for learning to control an agent by letting it interact with an environment. In recent years, deep neural networks (DNN) have been introduced into reinforcement learning, and they have achieved a great success on the value function approximation. The first deep Q-network (DQN) algorithm which successfully combines a powerful nonlinear function approximation technique known as DNN together with the Q-learning algorithm was proposed by Mnih et al. [3]. In this paper, experience replay mechanism was proposed. Following the DQN work, a variety of solutions have been proposed to stabilize the algorithms [3–9].The deep Q-networks classes have achieved unprecedented success in challenging domains such as Atari 2600 and some other games.

Although DQN algorithms have been successful in solving many problems because of their powerful function approximation ability and strong generalization between similar state inputs, they are still poor in solving some issues. Two reasons for this are as follows: (a) the randomness of the sampling is likely to lead to serious shock and (b) these systematic errors might cause instability, poor performance, and sometimes divergence of learning. In order to address these issues, the averaged target DQN (ADQN) [10] algorithm is implemented to construct target values by combining target Q-networks continuously with a single learning network, and the Bootstrapped DQN [11] algorithm is proposed to get more efficient exploration and better performance with the use of several Q-networks learning in parallel. Although these algorithms do reduce the overestimate, they do not evaluate the importance of the past learned networks. Besides, high variance in target values combined with the max operator still exists.

There are some ensemble algorithms [4, 12] solving this issue in reinforcement learning, but these existing algorithms are not compatible with nonlinearly parameterized value functions.

In this paper, we propose the ensemble algorithm as a solution to this problem. In order to enhance learning speed and final performance, we combine multiple reinforcement learning algorithms in a single agent with

several ensemble algorithms to determine the actions or action probabilities. In supervised learning, ensemble algorithms such as bagging, boosting, and mixtures of experts [13] are often used for learning and combining multiple classifiers. But in RL, ensemble algorithms are used for representing and learning the value function.

Based on an agent integrated with multiple reinforcement learning algorithms, multiple value functions are learned at the same time. The ensembles combine the policies derived from the value functions in a final policy for the agent. The majority voting (MV), the rank voting (RV), the Boltzmann multiplication (BM), and the Boltzmann addition (BA) are used to combine RL algorithms. While these methods are costly in deep reinforcement learning (DRL) algorithms, we combine different DRL algorithms that learn separate value functions and policies. Therefore in our ensemble approaches we combine the different policies derived from the update targets learned by deep Q-networks, deep Sarsa networks, double deep Q-networks, and other DRL algorithms. As a consequence, this leads to reduced overestimations, more stable learning process, and improved performance.

RELATED WORK

Reinforcement Learning

Reinforcement learning is a machine learning method that allows the system to interact with and learn from the environment to maximize cumulative return rewards. Assume that the standard reinforcement learning setting where an agent interacts with the environment $\mathcal{E}$. We can describe this process with Markov Decision Processes (MDP) [2, 9]. It can be specifed as a tuple (S, A, π, R, γ). At each step t, the agent receives a state s_t, and select an action a_t from the set of legal actions A according to the policy π, where π is a policy mapping sequences to actions. The action is passed to the environment E. In addition, the agent receives the next state S_{t+1} and a reward signal r_t. Tis process continues until the agent reaches a terminal state.The agent seeks to maximize the expected discounted return, where we define the future discounted return at time t as $R_t = \sum_{k=0}^{\infty} \gamma^k r^{t+k}$ with discount factor $\gamma \in (0, 1]$. The goal of the RL agent is to learn a policy which makes the future discounted return maximize. For an agent behaving according to a stochastic policy π, the value of the state action pair can be defined as follows: $Q^{\pi}(s, a) = E\{R_t \mid s_t = s, a_t = a, \pi\}$. Te optimal action-value function

Q satisfes the Bellman equation $Q^*(s,a) = E_{s'\sim\varepsilon}[r + \gamma \max_{a'} Q^*(s',a') \mid s,a]$. The reinforcement learning algorithms estimate the action value function by iteratively updating the Bellman equation $Q^*(s,a) = E_{s'\sim\varepsilon}[r + \gamma \max_{a'} Q^*(s',a') \mid s,a]$. When $t\to\infty$, the algorithm makes Q-value function converge to the optimal action value function [1]. If the optimal Q- function Q∗ is known, the agent can select optimal actions by selecting the action with the maximal value in a state: $\pi^* = \text{argmax}_a Q^*(s,a)$.

Target Deep Q Learning

RL agents update their model parameters while they observe a stream of transitions like $(s_t, a_t, r_{t+1}, s_{t+1})$. They discard the incoming data after a single update. There are two issues with this method. The first one is that there are strong correlations among the incoming data, which may break the assumption of many popular stochastic gradient-based algorithms. Secondly, the minor changes in the Q function may result in a huge change in the policy, which makes the algorithm difficult to converge [7, 9, 14, 15].

As for the deep Q-networks algorithms proposed in (Mnih et al., 2013), two aspects are improved. On the one hand, the action value function is approximated by the DNN, DQN uses the DNN with a parameter θ to approximate the value function, $Q(s,a;\theta) \approx Q^*(s,a;\theta)$; on the other hand, the experience replay mechanism is adopted. The algorithm learns from sampled transitions from an experience buffer, rather than learning fully online. Tis mechanism makes it possible to break the temporal correlations by mixing more and less recent experience for updating and training. Tis model free reinforcement learning algorithm solves the problem of "model disaster" and uses the generalized approximation method of the value function to solve the problem of "dimension disaster. The convergence issue was mentioned in 2015 by Schaul et al. [14]. The above Q-learning update rules can be directly implemented in a neural network. DQN uses the DNN with parameters θ to approximate the value function∘. The parameter θ updates from transition $(s_t, a_t, r_{t+1}, s_{t+1})$ are given by the following [11]:

$$\theta_{t+1} \longleftarrow \theta_t + \alpha\left(y_t^Q - Q(s_t,a_t;\theta_t)\right)\nabla_\theta Q(s_t,a_t;\theta_t), \tag{1}$$

with $y_i^{\text{DQN}} = r_t + \gamma \max_a \widetilde{Q}(s_{t+1},a;\theta_t^-)$

The update targets for Sarsa can be described as follows:

$$y_i^{\text{Sarsa}} = r_t + \gamma\widetilde{Q}(s_{t+1},a_{t+1};\theta_t^-), \tag{2}$$

where α is the scalar learning rate. θ^- are target network parameters which are fixed to$\theta^- = \theta^t$. In case the squared error is taken as a loss function $L_i(\theta_i) = E_{s' \sim \varepsilon}(y_i - Q(s, a; \theta_i))^2$.

In general, experience replay can reduce the amount of experience required to learn and replace it with more computation and more memory, which are often cheaper resources than the RL agent's interactions with its environment [14].

Double Deep Q Learning

In Q-learning and DQN, the max operator uses the same values to both select and evaluate an action. Tis can therefore lead to overoptimistic value estimates (van Hasselt, 2010). To mitigate this problem, the update targets value of double Q-learning error can then be written as follows:

$$y_i^{\text{DDQN}} = r_t + \gamma \widetilde{Q}\left(s_{t+1}, \arg\max_{a'} \widetilde{Q}\left(s_{t+1}, a'; \theta_t\right); \theta_t^-\right) \tag{3}$$

DDQN is the same as for DQN [8], but with the target y_i^{DQN} replaced with y_i^{DDQN}.

ENSEMBLE METHODS FOR DEEP REINFORCEMENT LEARNING

As DQN classes use DNNs to approximate the value function, it has strong generalization ability between similar state inputs. The generalization can cause divergence in the case of repeated bootstrapped temporal difference updates. So we can solve this issue by integrating different versions of the target network. In contrast to a single classifier, ensemble algorithms in a system have been shown to be more effective. They can lead to a higher accuracy. Bagging, boosting, and Ada Boosting are methods to train multiple classifiers. But in RL, ensemble algorithms are used for representing and learning the value function. They are combined by major voting, Rank Voting, Boltzmann Multiplication, mixture model, and other ensemble methods. If the errors of the single classifiers are not strongly correlated, this can significantly improve the classification accuracy.

Temporal Ensemble

As described in Section 2.2, the DQN classes of deep reinforcement learning algorithms use a target network with parameters θ^- copied from θ^t every

C steps. Temporal Ensemble method is suitable for the algorithms which use a target network for updating and training. Temporal ensemble uses the previous K learned networks to produce the value estimate and builds up $K \in$N complete networks with K distinct memory buffers. The recent Q-value function is trained according to its own target network $Q(s,a;\theta_t)$. So each one of Q-value functions $Q_1, Q_2, \ldots, Q_k$ represents temporally extended estimate of Q- value function.

Note that the more recent target network is likely to be more accurate at the beginning of the training and the accuracy of the target networks is increasing as the training goes on. So we denote a learning rate parameter $\lambda \in (0,1]$here for target network. The weight of ith target network is $w_i = \lambda^{i-1} / \sum_{i=1}^{N} \lambda^{i-1}$.

So the learned Q-value function by temporal ensemble can be described as follows:

$$Q^T(s,a;\theta) = \sum_{i=1}^{N} \left(\frac{\lambda^{i-1} Q_i(s,a;\theta_i)}{\sum_{i=1}^{N} \lambda^{i-1}} \right). \tag{4}$$

As $\lim_{\lambda \to 1} \lambda^{i-1} / \sum_{i=1}^{N} \lambda^{i-1} = 1/N$, we can see that the target networks have the same weights when λ equals 1. Tis formula indicates that the closer the target networks are, the greater the target networks' weight is. As target networks become more accurate, their weights become equal. The loss function remains the same as in DQN and so does the parameter update equation:

$$y_i^T = r_t + \gamma \max_{a'} \sum_{i=1}^{N} w_i Q^T \left(s', a'; \theta^T \right), \tag{5}$$

$$\theta_{t+1} \longleftarrow \theta_t + \alpha \left(y_t^T - Q(s_t, a_t; \theta_t) \right) \nabla_\theta Q(s_t, a_t; \theta_t).$$

In every iteration, the parameters of the oldest ones are removed from the target network buffer and the newest ones are added to the buffer. Note that the Q-value functions are inaccurate at the beginning of training. So the parameter λ may be a function of time and even the state space.

Ensemble of Target Values

The traditional ensemble reinforcement learning algorithms maintain multiple tabular algorithms in memory space [4, 16], and majority voting, rank voting, Boltzmann addition, and so forth are used to combine these

tabular algorithms. But deep reinforcement learning uses neutral networks as function approximators. Te use of multiple neural networks is very computationally expensive and inefcient. In contrast to previous researches, we combine diferent DRL algorithms that learn separate value functions and policies. Terefore in our ensemble approaches we combine the diferent policies derived from the update targets learned by deep Q-networks, deep Sarsa networks, double deep Q-networks, and other DRL algorithms as follows:

$$y_i^{\mathrm{DQN}} = r_t + \gamma \max_a \widetilde{Q}\left(s_{t+1}, a; \theta_t^-\right),$$

$$y_i^{\mathrm{Sarsa}} = r_t + \gamma \widetilde{Q}\left(s_{t+1}, a_{t+1}; \theta_t^-\right),$$

$$y_i^{\mathrm{DDQN}} = r_t + \gamma \widetilde{Q}\left(s_{t+1}, \operatorname*{argmax}_{a'} \widetilde{Q}\left(s_{t+1}, a_{t+1}; \theta_t\right); \theta_t^-\right). \tag{6}$$

Besides these update targets formula, other algorithms based on value function approximators can be also used to combine. The update targets according to the algorithm k at time t will be denoted by $y_t = \sum_{i=1}^{k} \beta_i y_t^i$.

The loss function remains the same as in DQN and so does the parameter update equation:

$$\theta_{t+1} \longleftarrow \theta_t + \alpha\left(y_t^E - Q\left(s_t, a_t; \theta_t\right)\right) \nabla_\theta Q\left(s_t, a_t; \theta_t\right). \tag{7}$$

The Ensemble Network Architecture

The temporal and target values ensemble algorithm (TEDQN) is an integrated architecture of the value-based DRL algorithms. As shown in Sections 3.1 and 3.2, the ensemble network architecture has two parts to avoid divergence and improve performance.

The architecture of our ensemble algorithm is shown in Figure 1; these two parts are combined together by evaluated network.

The temporal ensemble stabilizes the training process by reducing the variance of target approximation error [10]. Besides, the ensemble of target values reduces the overestimate and makes better performance by estimating more accurate Q-value. The temporal and target values ensemble algorithm are given by Algorithm 1.

As the ensemble network architecture shares the same input-output interface with standard Q-networks and target networks, we can recycle all learning algorithms with Q-networks to train the ensemble architecture.

EXPERIMENTS

Experimental Setup

So far, we have carried out our experiments on several classical control and Box2D environments on OpenAI Gym: CartPole-v0, MountainCar-v0, and LunarLander-v2 [15]. We use the same network architecture, learning algorithms, and hyperparameters for all these environments.

We trained the algorithms using 10,000 episodes and used the Adaptive Moment Estimation (Adam) algorithm to minimize the loss with learning rate $\mu = 0.00001$ and set the batch size to 32.

The summary of the configuration is provided below. The target network updated each 300 steps. The behavior policy during training was ε-greedy with ε annealed linearly from 1 to 0.01 over the first five thousands steps and fixed at 0.01 thereafter. We used a replay memory of ten thousands most recent transitions

We independently executed each method 10 times, respectively, on every task.

For each running time, the learned policy will be tested 100 times without exploration noise or prior knowledge by every 100 training episodes to calculate the average scores. We report the mean and standard deviation of the convergence episodes and the scores of the best policy.

(1) Initialize action-value network Q with random weights θ
(2) Initialize the target neural network buffer $(Q_i)_{i=1}^{L}$
(3) For episode 1, M do
(4) For $t = 1, T$ do
(5) With probability ε select a random action a_t, otherwise $a_t = \text{argmax}_a Q(s_t, a; \theta)$
(6) Execute action a_t in environment and observe reward r_t and next state s_{t+1}, and store transition (s_t, a_t, r_t, s_{t+1}) in D
(7) Sample random *minibatch* of transition (s_t, a_t, r_t, s_{t+1}) from D
(8) set $w_i = \lambda^{i-1} / \sum_{i=1}^{N} \lambda^{i-1}$
(9) Ensemble Q-learner $\widetilde{Q}(s, a; \theta) = \sum_{i=1}^{N} w_i Q_i(s, a; \theta_i)$
(10) set $y_i^{\text{DQN}} = r_t + \gamma \max_a \widetilde{Q}(s_{t+1}, a; \theta_t^-)$
(11) set $y_i^{\text{Sarsa}} = r_t + \gamma \widetilde{Q}(s_{t+1}, a_{t+1}; \theta_t^-)$
(12) set $y_i^{\text{DDQN}} = r_t + \gamma \widetilde{Q}(s_{t+1}, \text{argmax}_{a'} \widetilde{Q}(s_{t+1}, a_{t+1}; \theta_t); \theta_t^-)$
(13) Set $y_i = \{r_j$, if episode terminates at step $j+1$; $\sum_{i=1}^{k} \beta_i y_t^j$, otherwise$\}$
(14) $\theta_i = \underset{\theta}{\text{argmin}}\, E\left[\left(y_{(s,a)}^j - Q(s, a; \theta)\right)^2\right]$
(15) Every C steps reset $\widetilde{Q} = Q$
(16) End for
(17) End for

Algorithm 1. The temporal and target values ensemble algorithm.

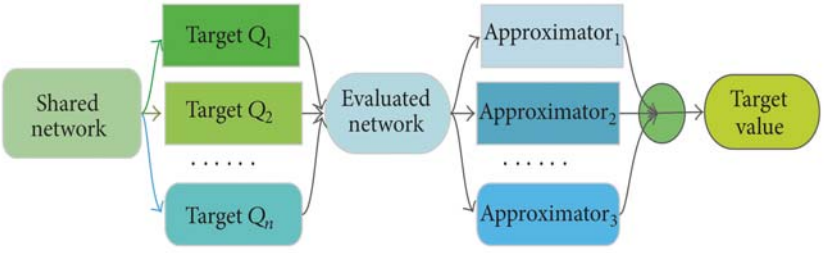


Figure 1. The architecture of the ensemble algorithm.

Results and Analysis

We consider three baseline algorithms that use target network and value function approximation, namely, the version of the DQN algorithm from the Nature paper [8], DSN that reduce over estimation [17], and DDQN that substantially improved the state-of-the-art by reducing the overestimation bias with double Q-learning [9].

Using this 10 no-ops performance measure, it is clear that the ensemble network does substantially better than a single network. For comparison we also show results for DQN, DSN, and DDQN. Figure 2 shows the improvement of the ensemble network over the baseline single network of DQN, DSN, and DDQN. Again, we see that the improvements are often very dramatic.

The results in Table 1 show that algorithms we presented can successfully train neural network controllers on the classical control domain on OpenAI Gym.

A detailed comparison shows that there are several games in which TE DQN greatly improves upon DQN, DSN, and DDQN. Noteworthy examples include CartPole-v0 (performance has been improved by 13.6%, 79.5%, and 7.8%, and variance has been reduced by 100%, 100%, and 100%),

MountainCar-v0 (performance has been improved by 26.7%, 21.2%, and 24.8%, and variance has been reduced by 31.6%, 77.9%, and 8.4%), and LunarLander-v2 (performance has been improved by 28.3%, 32.8%, and 50.5%, and variance has been reduced by 19.2%, 46.4%, and 50.5%).

CONCLUSION

We introduced a new learning architecture, making temporal extension and the ensemble of target values for deep Q learning algorithms, while sharing a common learning module. The new ensemble architecture, in combination with some algorithmic improvements, leads to dramatic improvements over existing approaches for deep RL in the challenging classical control issues. In practice, this ensemble architecture can be very convenient to integrate the RL methods based on the approximate value function.

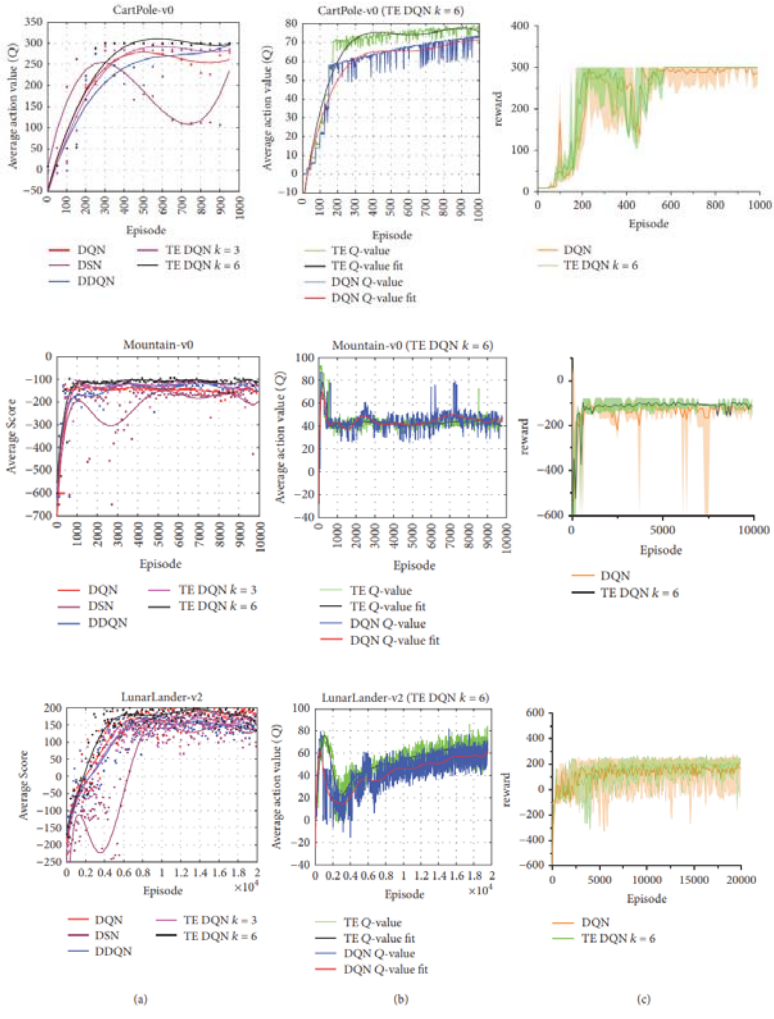


Figure 2. Training curves tracking the agent's average score and average predicted action-value. (a) Performance comparison of all algorithms in terms of

the average reward on each task. (b) Average predicted action-value on a held-out set of states on each task. Each point on the curve is the average of the action-value Q computed over the held-out set of states. (c) The performance of DQN and TEDQN on each task. The darker line shows the average scores of each algorithm, and the orange shaded area shows the two extreme values of DQN and the green shaded area shows TE DQN.

Table 1. The columns present the average performance of DQN, DSN, DDQN, EDQN, and TE-DQN after 10000 episodes, using ε-greedy policy with $\varepsilon = 0.0001$ after 10000 steps. The standard variation represents the variability over seven independent trials. Average performance improved with the number of averaged networks

Task (AVG score, Std.)	CartPole-v0	MountainCar-v0	LunarLander-v2
DQN	(264.9, 21.7)	(−148.2, 17.4)	(159.3, 16.7)
DSN	(167.1, 61.6)	(−137.7, 53.9)	(153.9, 25.2)
Double DQN	(278.2, 31.8)	(−144.2, 16.8)	(135.8, 11.8)
TE DQN $K = 3$	(299.1, 1.3)	(−115.6, 21.4)	(186.9, 19.1)
TE DQN $K = 6$	(300, 0)	(−108.4, 11.9)	(204.4, 13.5)

Although the ensemble algorithms are superior to a single reinforcement learning algorithm, it is noted that the computational complexity is higher. The experiments also show that the temporal ensemble makes the training process more stable, and the ensemble of a variety of algorithms makes the estimation of the Q-value more accurate. The combination of the two ways enables the training to achieve a stable convergence. This is due to the fact that ensembles improve independent algorithms most if the algorithms predictions are less correlated. So that the output of the Q-network based on the choice of action can achieve balance between exploration and exploitation. In fact, the independence of the ensemble algorithms and their elements is very important on the performance for ensemble algorithms. In further works, we want to analyze the role of each algorithm and each Q-network in different stages, so as to further enhance the performance of the ensemble algorithm.

REFERENCES

1. S. Mozer and M. Hasselmo, “Reinforcement learning: an introduction,” IEEE Transactions on Neural Networks and Learning Systems, vol. 16, no. 1, pp. 285-286, 2005.
2. L. P. Kaelbling, M. L. Littman, and A.W. Moore, “Reinforcement learning: a survey,” Journal of Artifcial Intelligence Research, vol. 4, pp. 237–285, 1996.
3. V. Mnih, K. Kavukcuoglu, D. Silver et al., “Playing Atari with deep reinforcement learning [EB/OL],” https://arxiv.org/abs/ 1312.5602.
4. M. A. Wiering and H. van Hasselt, “Ensemble algorithms in reinforcement learning,” IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 38, no. 4, pp. 930–936, 2008.
5. S. Whiteson and P. Stone, “Evolutionary function approximation for reinforcement learning,” Journal of Machine Learning Research (JMLR), vol. 7, pp. 877–917, 2006.
6. P. Preux, S. Girgin, and M. Loth, “Feature discovery in approximate dynamic programming,” in Proceedings of the 2009 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning, ADPRL 2009, pp. 109–116, April 2009.
7. T. Degris, P. M. Pilarski, and R. S. Sutton, “Model-Free reinforcement learning with continuous action in practice,” in Proceedings of the 2012 American Control Conference, ACC 2012, pp. 2177–2182, June 2012.
8. V. Mnih, K. Kavukcuoglu, D. Silver et al., “Human-level control through deep reinforcement learning,” Nature, vol. 518, no. 7540, pp. 529–533, 2015.
9. H. Van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double Q-Learning,” in Proceedings of the 30th AAAI Conference on Artifcial Intelligence, AAAI 2016, pp. 2094– 2100, February 2016.
10. O. Anschel, N. Baram, N. Shimkin et al., “Averaged-DQN: Variance Reduction and Stabilization for Deep Reinforcement Learning [EB/OL],” https://arxiv.org/abs/1611.01929.
11. I. Osband, C. Blundell, A. Pritzel et al., “Deep Exploration via Bootstrapped DQN [EB/OL],” https://arxiv.org/abs/1602.04621.
12. S. Faußer and F. Schwenker, “Ensemble Methods for Reinforcement Learning with Function Approximation,” in Multiple Classifer Systems, pp. 56–65, Springer, Berlin, Germany, 2011.

13. A. K. Jain, R. P. W. Duin, and J. Mao, "Statistical pattern recognition: a review," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 1, pp. 4–37, 2000.
14. T. Schaul, J. Quan, I. Antonoglou et al., "Prioritized Experience Replay [EB/OL]," https://arxiv.org/abs/1511.05952.
15. I. Zamora, N. G. Lopez, V. M. Vilches et al., "Extending the OpenAI Gym for robotics: a toolkit for reinforcement learning using ROS and Gazebo [EB/OL]," https://arxiv.org/abs/1608.05742.
16. D. Ernst, P. Geurts, and L. Wehenkel, "Tree-based batch mode reinforcement learning," Journal of Machine Learning Research (JMLR), vol. 6, no. 2, pp. 503–556, 2005.
17. M. Ganger, E. Duryea, and W. Hu, "Double Sarsa and Double Expected Sarsa with Shallow and Deep Learning," Journal of Data Analysis and Information Processing, vol. 04, no. 04, pp. 159–176, 2016.

Section 2:
Deep Learning Techniques Applied In Biology

CHAPTER 6

FISH DETECTION USING DEEP LEARNING

Suxia Cui[1], Yu Zhou[1], Yonghui Wang[2], and Lujun Zhai[1]

[1]Department of Electrical and Computer Engineering, Prairie View A&M University, Prairie View, TX 77446, USA

[2]Department of Computer Science, Prairie View A&M University, Prairie View, TX 77446, USA

ABSTRACT

Recently, human being's curiosity has been expanded from the land to the sky and the sea. Besides sending people to explore the ocean and outer space, robots are designed for some tasks dangerous for living creatures. Take the ocean exploration for an example. There are many projects or competitions on the design of Autonomous Underwater Vehicle (AUV) which attracted many interests. Authors of this article have learned the necessity of platform upgrade from a previous AUV design project, and would like to share the

Citation: Suxia Cui, Yu Zhou, Yonghui Wang, Lujun Zhai, "Fish Detection Using Deep Learning", Applied Computational Intelligence and Soft Computing, vol. 2020, Article ID 3738108, 13 pages, 2020. https://doi.org/10.1155/2020/3738108.

experience of one task extension in the area of fish detection. Because most of the embedded systems have been improved by fast growing computing and sensing technologies, which makes them possible to incorporate more and more complicated algorithms. In an AUV, after acquiring surrounding information from sensors, how to perceive and analyze corresponding information for better judgment is one of the challenges. The processing procedure can mimic human being's learning routines. An advanced system with more computing power can facilitate deep learning feature, which exploit many neural network algorithms to simulate human brains. In this paper, a convolutional neural network (CNN) based fish detection method was proposed. The training data set was collected from the Gulf of Mexico by a digital camera. To fit into this unique need, three optimization approaches were applied to the CNN: data augmentation, network simplification, and training process speed up. Data augmentation transformation provided more learning samples; the network was simplified to accommodate the artificial neural network; the training process speed up is introduced to make the training process more time efficient. Experimental results showed that the proposed model is promising, and has the potential to be extended to other underwear objects.

INTRODUCTION

The ocean is full of mystery and the underwater exploration has always been an exciting topic. Nowadays, robotics has been widely adopted into our daily lives. The AUV is one type of robot, which is gaining more and more attention [1, 2]. It must be equipped with a sophisticate onboard computer, Inertial Measurement Unit (IMU), and other sensors to be able to support a preprogrammed navigation system [1]. Authors have experience on design and function of an AUV [3, 4] for competitions. The AUV, as shown in Figure 1, is featured with an i7-based industrial motherboard plus an ARM microcontroller. Detail hardware layout and mechanical balancing scheme are introduced in [3, 4]. It passed the qualification and became one of the eleven finalists at the 2017 IEEE Singapore AUV Challenge [5]. This competition was hosted in a swimming pool of clear water. The tasks did not need a high-resolution camera, so the major processor was not chosen to be of high performance. After this the AUV retired from the competition, authors realized it was time to revise the system to conquer real life tasks. As of now, most of the robot control platforms were shifting to Systems-On-Chip (SOC) [6, 7]. To move forward and add more functionalities to

the AUV, one goal is to switch from a clear swimming pool environment to a real ocean water condition. Therefore, the hardware has to be upgraded to high resolution digital camera along with a powerful onboard computer, such as NVIDIA JETSON AGX XAVIER development board. So, before upgrading the whole system with integrated vision, research on an off-line simulation of the computer vision module was conducted. Fishes of many kinds were chosen to be the objects to build up the training and testing data set. Ocean water conditions vary from place to place. In the Gulf of Mexico where the authors reside, the water is not as clear as in the east or west coast of the United States. Thus, how to identify fish from the blurred sea water is most challenging in this research. One of the solutions is to adopt ultrasonic technology [8, 9]. To some extent, it was proved to be effective for the fish industry where a rough quantity of fish is sufficient enough. However, because of low resolution, it is difficult to differentiate objects in a complex environment that has mixed fishes, turtles, etc. The goal of this research is to investigate the object detection scheme under real sea water through an AUV build-in digital camera. Researchers have successfully adopted the digital camera as a tool for capturing images from the ocean to improve underwater robot vision [10], but the vehicle was remotely operated (ROV) instead of an AUV.

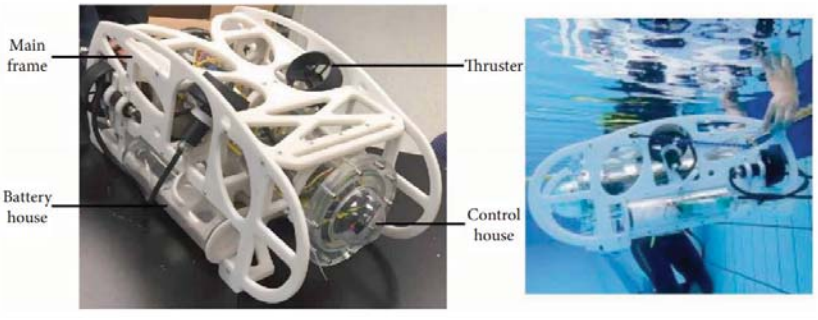


Figure 1. AUV and competition environment.

LITERATURE REVIEW

The main contribution of this research is to introduce deep learning methodology to accomplish fish identification in blurry ocean water. As a result, the approach improved computer vision into an AUV system through an applicable neural network.

Computer Vision

Computer vision uses computers with imaging sensors to imitate human visual functions that extract features from obtained data set, analyse and classify them to assist in decision making. It usually involves many fields of knowledge such as high-level computer programming, image processing, artificial intelligence (AI), and so on. For example, manufacture industry uses it to check the defection or improve the quality from large quantities of products [11, 12]. There are mature applications on face detection and emotion observation at the airport and other security checking points [13–15]. Medical doctors' use certain diagnose software to assist in identifying tumours and other abnormal tissues from medical imaging [16]. The agricultural industry adopts the computer vision to decision making system for predicting the yield from the field [17]. Google is designing its own self-driving car with a visual range of about 328 feet and the car can recognize traffic signs and avoid pedestrians [18]. Many state-of-the-art examples indicate that computer vision is changing our daily lives. To improve the performance, besides traditional image processing skills, deep learning algorithms which imitate our brain are widely adopted.

Deep Learning

The concepts of deep learning with neural network has arisen decades ago. It was originally developed by researcher LeCun et al. in 1998 [19]. He designed a five-layer classifier named LeNet5 using a Convolutional Neural Network (CNN). Due to dramatic improvement in computing power and the explosion of big data, deep learning is able to make tremendous achievements in the past several years. Deep learning is based on big data collected in a certain field. Learning resources from massive data are extremely important. Deep means that a neural network has lots of layers for imitating our brain. With the advent of high-performance GPU, ASIC accelerators, cloud storage, and powerful computing facility, it is now possible to collect, manage, and analyse big data sets. Because only with data sets large enough, can overfitting problems be solved in deep learning. And the enhanced computing power can accelerate the speed of time-consuming training process.

Deep leaning based approaches are increasingly applied in many fields, and have significant advantage over traditional algorithms in computer vision and object detection. The performance of many robotics systems has been improved by incorporating deep learning. Take Google's AlphaGo as

an example, it studied human's learning behavior and in return compete with the famous Go player [20].To be able to foster deep learning in computer vision, enough examples from images collected beforehand is critical. ImageNet is a good example [21]. One contribution of this research includes developing a database of fish in ocean water to support training and testing. Nevertheless, a learning algorithm is important as well. Traditional computer vision and image processing approaches suffered from the accuracy of feature extraction, while deep learning method can be utilized to improve the technique through neural network.

Neural Network

Over the past few years, neural networks in deep learning were getting increasingly popular. In 2012, researcher Krizhevsky et al. adopted CNN to accomplish images classification in the ImageNet Large Scale Visual Recognition Challenge [22, 23], and the test accuracy was significantly higher than traditional algorithms. Due to this achievement, the interest in deep learning with neural network has been raised [24]. In 2014, Ross et al. proposed an algorithm called Fast R-CNN which aims to convert object identification into a regression problem [25]. The mean average precision was improved by almost 30% compared to the previous best result 53.3% on ImageNet Large Scale Visual Recognition Challenge in 2012. The amount of calculation was massive because features from different sizes of thousands of proposals in each image would be extracted. Since Faster R-CNN reduced the computational burden dramatically, it has been widely adopted recently in computer vision which involves target detection, image classification, and object identification. YOLO proposed in Facebook is also a milestone for corresponding research [26, 27].

MATERIALS AND METHODS

In this paper, a CNN model with image segmentation is introduced for fish detection from in blurry ocean water. Specific data set was developed to support this research. The data augmentation transformation scheme was adopted to obtain more learning resources because the original images in the particular environment are not sufficient for training purpose. To solve the overfitting problem, the dropout algorithm is applied. Because our goal is to incorporate this system into an AUV which requires real-time applications, some trade-offs were discussed to reduce processing time. In this section, detail system design with optimization approaches is addressed.

CNN Architecture

A CNN model usually consists of many layers, such as an input layer, convolutional layers with nonlinear units, and fully connected layers [28, 29]. An example of CNN is demonstrated in Figure 2. The first layer is the input layer which receives image information as learning resources from the outside world. The following layers are convolutional layers, which are responsible for extracting features from images. Convolution operation is one of the common mathematical operations. The convolution formula of two discrete functions is shown in Equation (1):

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]. \quad (1)$$

Figure 2. Convolutional operation on a RGB color image [30].

The data set consists of 256 levels of RGB color images. The 3×3 matrix, W0 below, is called a kernel or a filter. In practice, convolutional operations are performed on R, G, and B channels respectively, and then the results are summed up to obtain each element in the feature map as shown in Figure 2.

In order to extract the features of an object more accurately, a lot of filters are used in each convolutional layer. For example, to extract features, such as edges, texture, etc., corresponding filters are available as shown in Figure 3.

When performing the convolutional operation, the size of the feature map is under consideration. There are three main factors that influence its size: depth, stride, and padding. Figure 4 illustrates the feature map where depth is 3, stride is 1, and with zero padding.For a complex neural network, usually there are two types of connections between two adjacent layers. They are the fully connected layer and locally connected neural layer respectively as illustrated in Figure 5. For a fully connected neural net, all pixels in the input layer are connected with each neuron in the hidden layer as shown in Figure 5(a). It is common that the last two layers in a CNN are fully connected layers. They are the somax and output layer, respectively. Because a huge number of parameters will increase the amount of computation and delay the processing. For a locally connected neural network, only a portion of pixels in the input layer are connected with the following neuron in the hidden layer as shown in Figure 5(b). This type of connection will reduce the number of connections and speed up the system.

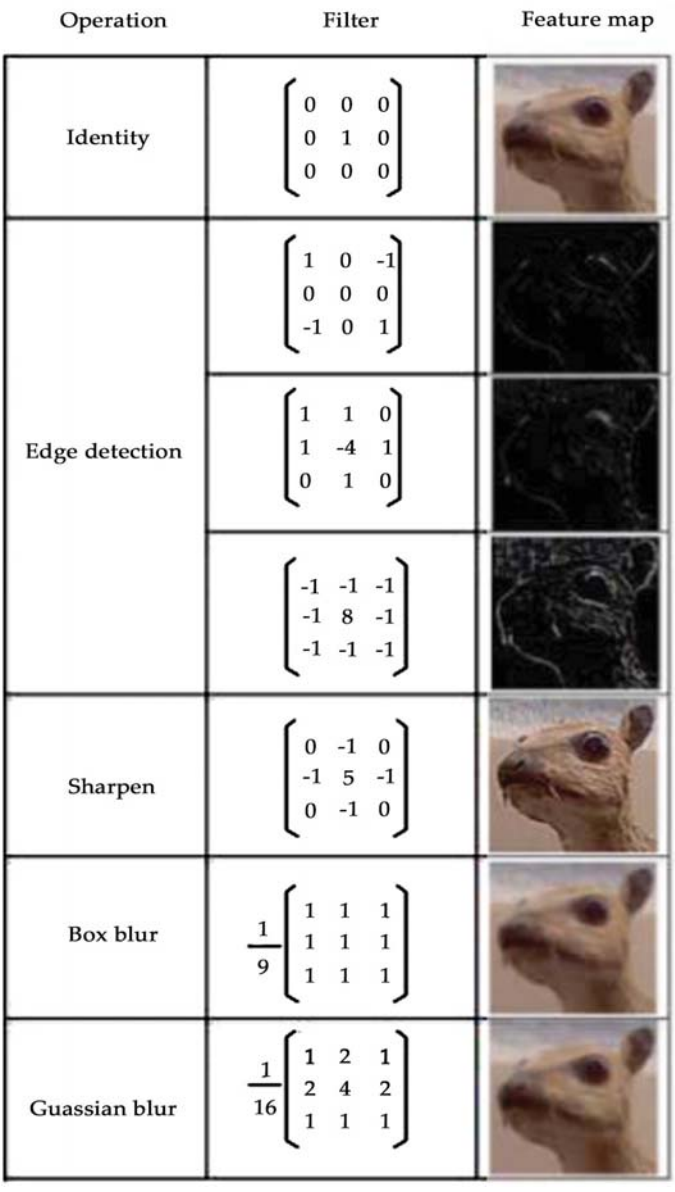


Figure 3. Convolution operation application in image feature exaction [31].

The Convolutional layer in CNN uses local connections as shown in Figure 6. For example, the value-8 in the feature map is only connected with a 3×3 matrix [0, 0, 0; 0, 1, 1; 0, 1, 2] from input image and has nothing to do with the remaining parts of the input image pixels.

The parameters for fully or local connectivity for all the layers in this CNN are listed in Table 1.

The system can be visualized with simplification in Figure 7.

System Validation Using ImageNET Dataset

Before applying this system into the ocean fish data set developed in this research, authors downloaded images from the well-known ImageNet ILSVRC [21] to do a system validation testing through object classification. There are 500 images with 20 classes ranging from fish, coral, sea turtle, frog, ship, etc. Here all the RGB images are rescaled to 448×448. Ground truths images are obtained from operate labeling soware manually. Each image is divided into a grid of 7×7 cells. Each cell will predict the two bounding boxes location information and class information made up of a 1×1×30 vector. This vector consists of object center coordinates (X, Y), the width w, and height h of the bounding box confidence scores, and predicted probabilities of fish, as shown in Figure 8.To predict the target location of an image, the target is displayed in a bounding box. There are always errors between the ground truth and the predictions. Loss function was developed to measure errors consisting of three parts: coordinate error, (Intersection over union) IoU error, and class error. Equation (2) gives the mathematics form of the loss function.

$$Loss = \sum_{i=0}^{S^2} coordError + IoUError + classError. \tag{2}$$

Here, IoU is used to measure position accuracy as shown in Figure 9.

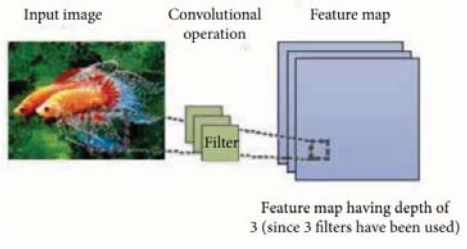


Figure 4. Feature extractor using convolutional operation.

Each grid cell in an image will predict 꽨 bounding boxes that encloses an object to predict the object localization and class. In addition, there is a confidence with each bounding box. Confidence score has nothing to do with the class of object. It just depicts how certain it is that the predicted box actually encloses the real object.

$$\text{Confidence} = \Pr(\text{object}) \times \text{IoU}, \tag{3}$$

where Pr(object) represents the probability of the object of interest. If there is an object in the grid cell, the Pr(object) is 1; otherwise, it is 0.

Usually, loss function is in the form of the sum of squared errors as shown below [33]. It consists of three parts which are localization errors, confidence errors, and probabilities errors.

$$\begin{aligned} Loss = & \sum_{i=0}^{S^2} \sum_{j=0}^{B} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (w_i - \hat{w}_i)^2 + \left(h_i - \hat{h}_i\right)^2 \right] \\ & + \sum_{i=0}^{S^2} \sum_{j=0}^{B} \left(C_i - \hat{C}_i\right)^2 + \sum_{i=0}^{S^2} \sum_{C \in class} (P_i(c) - P_i(\hat{c}))^2, \end{aligned} \tag{4}$$

where x_i, y_i are the ground truth coordinates of objects center; w_i , h_i are the width and height of the ground truth bounding box; $\hat{x}_i$, $\hat{y}_i$ are the predicted coordinates of the objects center; $\hat{w}_i$, $\hat{h}_i$ are the width and height of predicted bounding box. Above Figure 10 shows one set of output with confidence values from different classes.

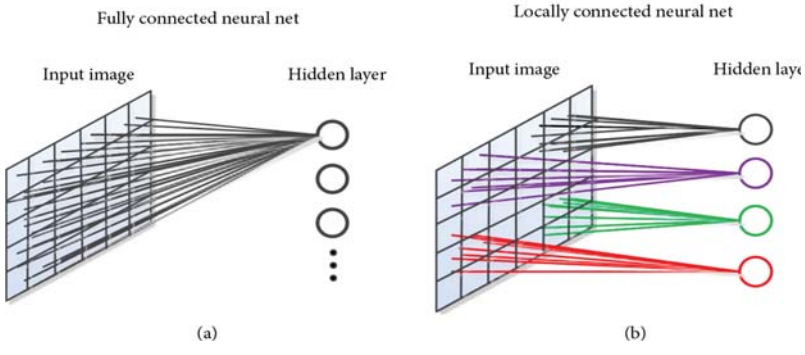


Figure 5. Fully connected neural net and locally connected neural net [32].

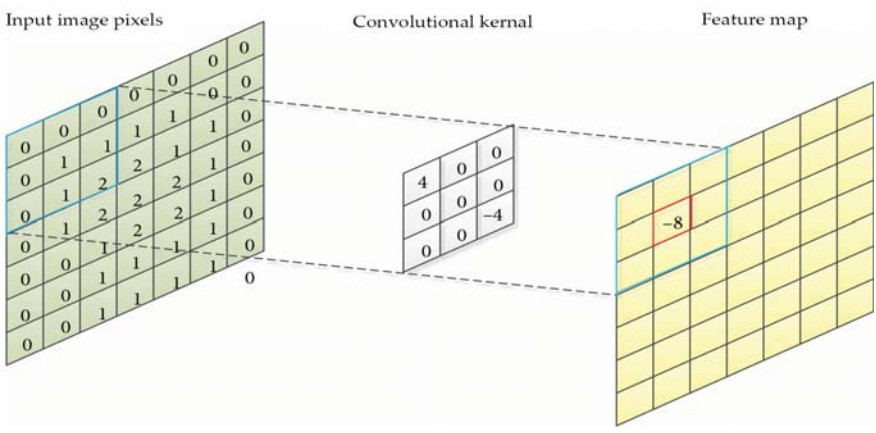


Figure 6. Convolutional operation using local connections.

Ground Truth Preparation for Real Ocean Environment

After testing the CNN system with perfect images without noise. Next step is to build our own dataset of fish in the ocean. Because it is difficult to obtain images from other kinds of objects such as the sea turtle, coral and so on. is part of the research, fish is the only object to be detected. For the collection of 410 images, many of them have multiple fish in one image, so the detection is challenging. The same method was chosen to create ground truth image. And all the parameters introduced before remain the same, only the class information is made up of a 1×1×18 vector instead of 1×1×30 because of reduce in the classes. Figure 11 illustrates one labeled image example. It is obvious that this data set is totally different from the ideal images from ImageNET.

DATA AUGMENTATION

Since deep learning is based on large training dataset for the system to learn and build up the identification knowledge, enough data has to be provided as learning resources to extract object features [34].

The lacking of data would bring overfitting problem. Images were collected in real underwater environment from the Gulf of Mexico and are going to be used to attract object features. However, the number of original images collected from a particular environment is not large enough to train the system.

Therefore, data augmentation transformation was performed geometrically, which changed the pixels location while the images features remained unchanged as shown in Figure 12. Four types of data augmentation transformation were adopted, which doubled the number of original images to make the training dataset sufficient.

(1) Rotation: rotate the images by random angles;

(2) Scale: the images are scaled to different sizes according to scale factors set;

(3) Crop: crop patches of images;

(4) Mirror symmetry: flip the original images horizontally or vertically

Dropout Algorithm

One of the common problems in deep learning is overfitting, which refers to the fact that the testing accuracy is much lower than the training accuracy. In this case, a model with high performance feature is built using real world training data.

If it turns out to have overfitting issue, the robustness of the model is worth consideration. Apart from the lacking of learning data, which will also cause overfitting problem, the huge amount of parameters in a neural network would lead to the overfitting issue. Therefore, the dropout algorithm [35] was introduced into the system to simplify the model, which is depicted in Figure 5.

Table 1. Parameters in CNN model with image segmentation

Layer	Input size	Filter size	Stride	Output size
Conv. 1	[$N \times 448 \times 448 \times 3$]	[$7 \times 7 \times 3 \times 64$]	2	[$N \times 224 \times 224 \times 64$]
Maxpool 1	[$N \times 224 \times 224 \times 64$]	[2×2]	2	[$N \times 112 \times 112 \times 32$]
Conv. 2	[$N \times 112 \times 112 \times 32$]	[$3 \times 3 \times 32 \times 192$]	1	[$N \times 112 \times 112 \times 192$]
Maxpool 2	[$N \times 112 \times 112 \times 192$]	[2×2]	2	[$N \times 56 \times 56 \times 96$]
Conv. 3	[$N \times 56 \times 56 \times 96$]	[$3 \times 3 \times 128 \times 256$]	1	[$N \times 56 \times 56 \times 256$]
Conv. 4	[$N \times 56 \times 56 \times 128$]	[$3 \times 3 \times 128 \times 256$]	1	[$N \times 56 \times 56 \times 256$]
Conv. 5	[$N \times 56 \times 56 \times 256$]	[$1 \times 1 \times 256 \times 256$]	1	[$N \times 56 \times 56 \times 256$]
Conv. 6	[$N \times 56 \times 56 \times 256$]	[$3 \times 3 \times 256 \times 512$]	1	[$N \times 56 \times 56 \times 512$]
Maxpool 3	[$N \times 56 \times 56 \times 512$]	[2×2]	2	[$N \times 28 \times 28 \times 256$]
Conv. 7	[$N \times 28 \times 28 \times 256$]	[$1 \times 1 \times 256 \times 256$]	1	[$N \times 28 \times 28 \times 256$]
Conv. 8	[$N \times 28 \times 28 \times 256$]	[$1 \times 1 \times 256 \times 256$]	1	[$N \times 28 \times 28 \times 256$]
Conv. 9	[$N \times 28 \times 28 \times 256$]	[$1 \times 1 \times 256 \times 256$]	1	[$N \times 28 \times 28 \times 256$]
Conv. 10	[$N \times 28 \times 28 \times 256$]	[$1 \times 1 \times 256 \times 256$]	1	[$N \times 28 \times 28 \times 256$]
Conv. 11	[$N \times 28 \times 28 \times 256$]	[$3 \times 3 \times 256 \times 512$]	1	[$N \times 28 \times 28 \times 512$]
Conv. 12	[$N \times 28 \times 28 \times 512$]	[$3 \times 3 \times 512 \times 512$]	1	[$N \times 28 \times 28 \times 512$]
Conv. 13	[$N \times 28 \times 28 \times 512$]	[$3 \times 3 \times 512 \times 512$]	1	[$N \times 28 \times 28 \times 512$]
Conv. 14	[$N \times 28 \times 28 \times 512$]	[$3 \times 3 \times 512 \times 512$]	1	[$N \times 28 \times 28 \times 512$]

Conv. 15	[N × 28 × 28 × 512]	[1 × 1 × 512 × 512]	1	[N × 28 × 28 × 512]
Conv. 16	[N × 28 × 28 × 512]	[3 × 3 × 512 × 1024]	1	[N × 28 × 28 × 1024]
Maxpool 4	[N × 28 × 28 × 1024]	[2 × 2]	2	[N × 14 × 14 × 512]
Conv. 17	[N × 14 × 14 × 512]	[1 × 1 × 512 × 256]	1	[N × 14 × 14 × 256]
Conv. 18	[N × 14 × 14 × 256]	[3 × 3 × 256 × 1024]	1	[N × 14 × 14 × 1024]
Conv. 19	[N × 14 × 14 × 1024]	[1 × 1 × 1024 × 512]	1	[N × 14 × 14 × 512]
Conv. 20	[N × 14 × 14 × 512]	[3 × 3 × 512 × 1024]	1	[N × 14 × 14 × 512]
Conv. 21	[N × 14 × 14 × 512]	[3 × 3 × 512 × 1024]	1	[N × 14 × 14 × 1024]
Conv. 22	[N × 14 × 14 × 1024]	[3 × 3 × 1024 × 1024]	1	[N × 7 × 7 × 1024]
Conv. 23	[N × 7 × 7 × 1024]	[3 × 3 × 1024 × 1024]	2	[N × 7 × 7 × 1024]
Conv. 24	[N × 7 × 7 × 1024]	[3 × 3 × 1024 × 1024]	1	[N × 7 × 7 × 1024]
Fully conn.1	[N × 7 × 7 × 1024]	[1024 × 4]	Multi	[1 × 4096]
Fully conn. 2	[1 × 4096]	[4096 × 7 × 7 × 30]	Multi	[7 × 7 × 30]

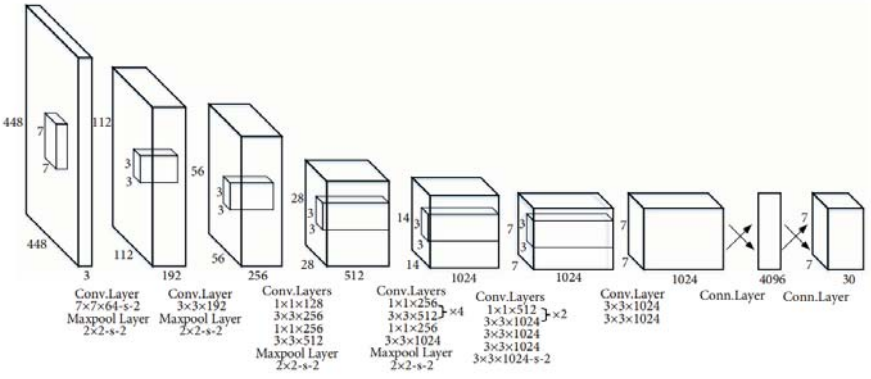


Figure 7. CNN architecture for object identification [26].

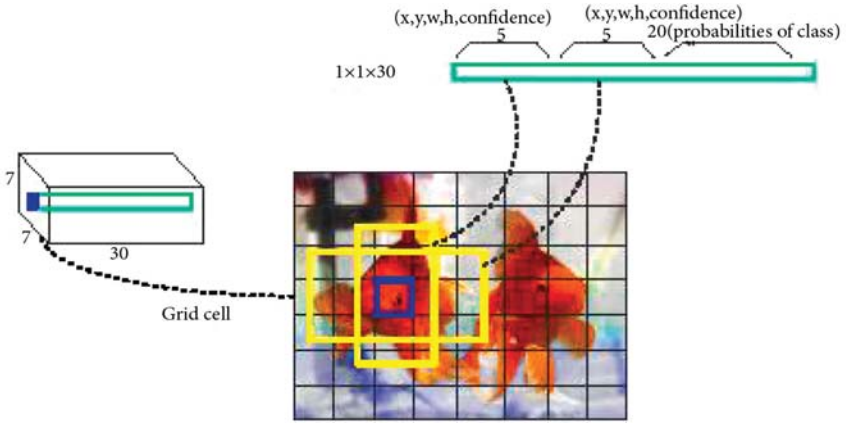


Figure 8. Output of CNN model using ImageNET.

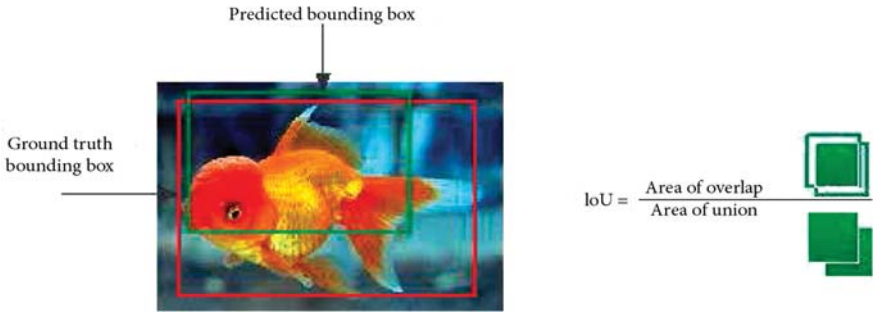


Figure 9. Intersection over union.

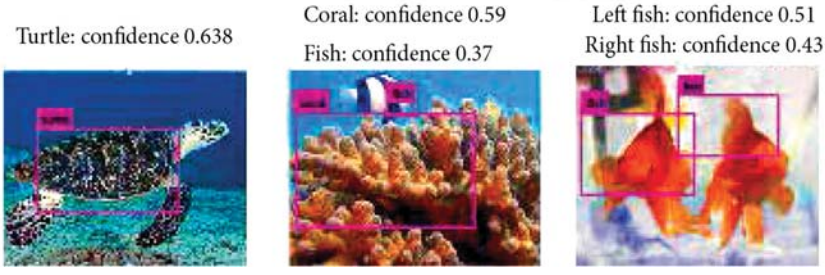


Figure 10. Object classification results from ImageNET data set.

Figure 11. Labeled ground truth.

Dropout means that we remove some nodes temporarily from the network according to the probability setting in the process of learning. In practice, some features can be extracted only when some hidden relationships exist, which decreased the robustness of the deep learning model. On the other hand, dropout removes a hidden fixed relationship between nodes, anti-interference ability can be improved and overfitting problem can be solved to some extent. L1 and L2 regularization are achieved by modifying the cost

function, while dropout is implemented by modifying the neural network itself, which is a technique used when training the network. In each iteration of the training process, authors randomly drop some of the neurons, and set the probability of eliminating nodes in the neural network for each layer of the network. For example, the value is set to 0.5, as shown in Figure 13 on the left. The neurons are discarded, then the connections from the node are removed, and finally a network with fewer nodes and smaller scales is obtained. e network structure during this training will be simplified to the one shown in Figure 13 on the right.

Figure 12. Data augmentation transformation.

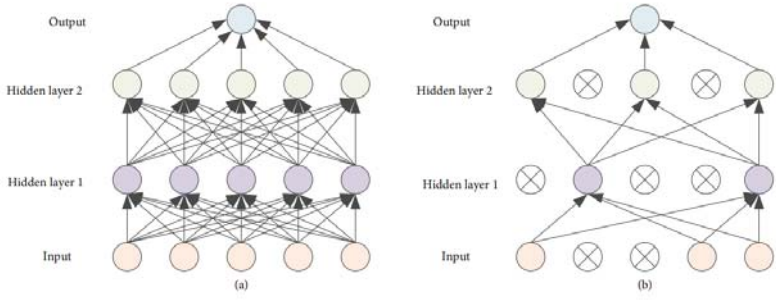


Figure 13. A standard neural network model (a) and a network model with dropout (b) [35].

Refine Loss Function

YOLO improved loss function from Equations (4) and (5), [26]. Three coefficients were placed before the error terms in proportion to its

contribution to the loss. As shown in Equation (5), the first two terms are related to coordinate the identified object with x and y to denote the object location, while w and h refer to the width and height of the bounding box. In order to have more weight in the first two terms, $\gamma_{cord}$was assigned to be the largest number, which had a value of 5. Thus, the weight of localization error got enhanced. In terms of IoU error computation, when the object center falls in this cell, the weight of IoU error should be increased in order to predict location accurately. The value of $\gamma_{noobj}$is set to be 0.5 to refine the IoU error. For the same error value, the effect of large object error on the detection should be less than the effect of small object error on the detection. This is because the same bias accounts for the proportion of large objects is much smaller than the proportion of the same deviation to small objects. Therefore, it is supposed to increase the contribution to loss due to bigger object IoU error. Square roots of width and height were chosen to replace their original forms. For same bias value, the square root error from the big box is smaller than the small one.

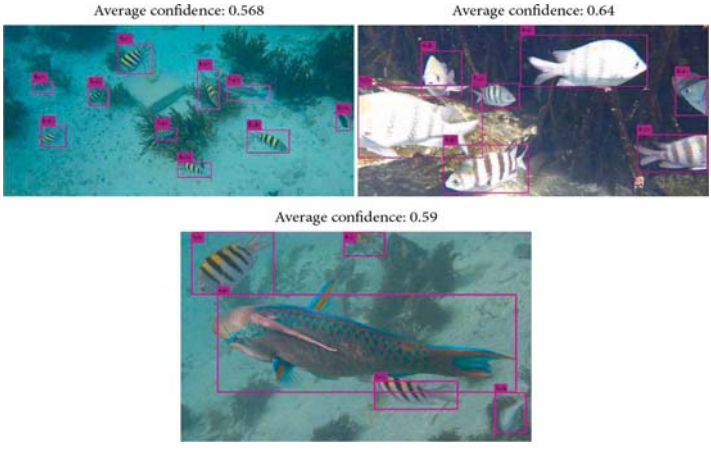


Figure 14. Enhancement of identification accuracy with data augmentation.

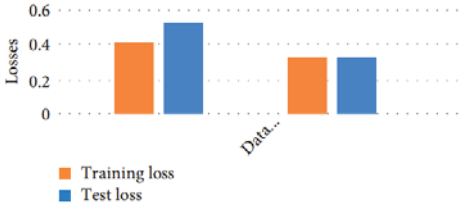


Figure 15. Comparison of training loss.

$$
\begin{aligned}
Loss =& \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \left[1_{ij}^{obj} (x_i - \hat{x}_l)^2 + (y_i - \hat{y}_l)^2 \right] \\
&+ \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_l} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_l} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left(C_i - \hat{C}_l \right)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{nobj} \left(C_i - \hat{C}_l \right)^2 \\
&+ \sum_{i=0}^{S^2} 1_{ij}^{obj} \sum_{C \in class} \left(P_i(c) - P_l(\hat{c}) \right)^2 .
\end{aligned} \tag{5}
$$

In this paper, authors refined the loss function to fit for multiple fish application. The proposed loss function is regularized to reduce the small dataset and overfitting problem, L2 regularization is to add a regularization function after the cost function which is listed in the Equations (6) and (7).

$$
\begin{aligned}
Loss =& \sum_{i=0}^{S^2} (coordError + IoUError + classError) \\
&+ \text{L2 Regularization},
\end{aligned} \tag{6}
$$

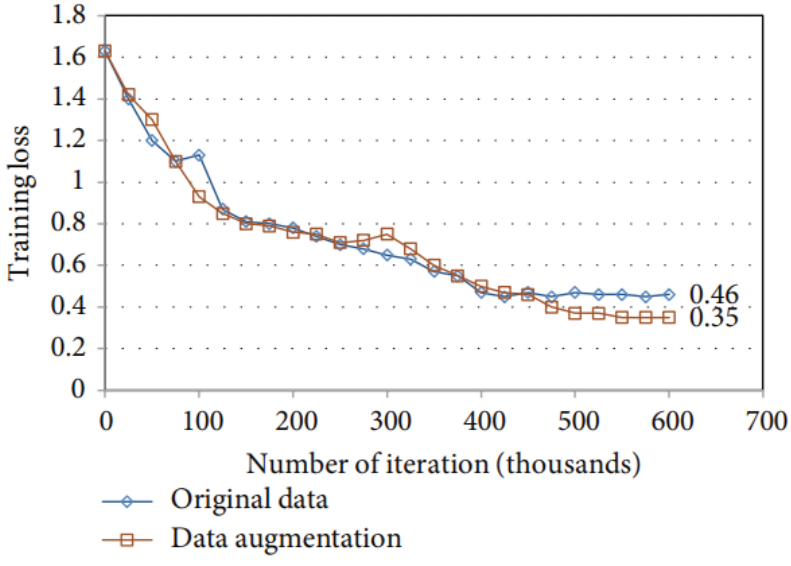


Figure 16. Effect of data augmentation on overfitting.

$$
\begin{aligned}
Loss =& \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \left[1_{ij}^{obj} (x_i - \hat{x}_l)^2 + (y_i - \hat{y}_l)^2 \right] \\
&+ \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_l} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_l} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left(C_i - \hat{C}_l \right)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{nobj} \left(C_i - \hat{C}_l \right)^2 \\
&+ \sum_{i=0}^{S^2} 1_{ij}^{obj} \sum_{C \in class} \left(P_i(c) - P_l(\hat{c}) \right)^2 + \frac{\lambda}{2n} \sum_{w} w^2.
\end{aligned}
\tag{7}
$$

The last term is the L2 regularization term, which is the sum of the squares of all the parameters w, divided by the sample size n of the training set. λ is the coefficient of the regular term, which weighs the proportion of the regular term and the other terms. There is also a coefficient 1/2, 1/2 is oen seen, mainly for the convenience of the results of the latter, the latter will produce a 2, multiplied by 1/2 just rounded up. The principle and procedure of how L2 reduce overfitting is in the reference [31].

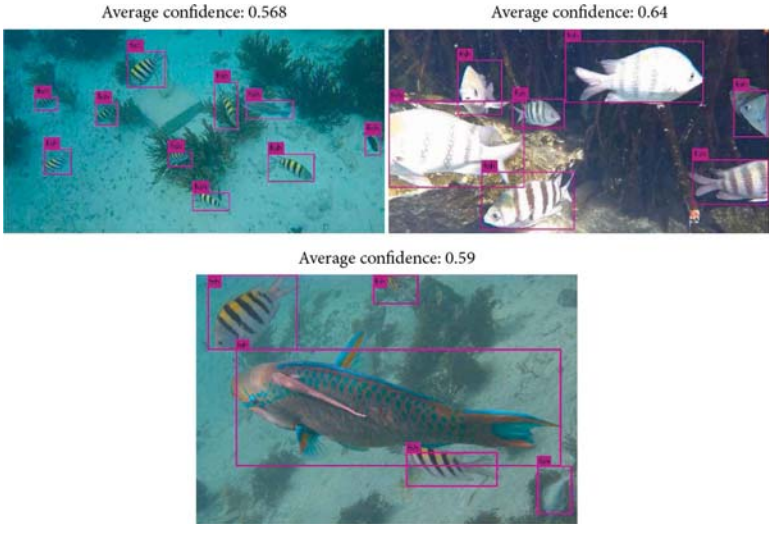


Figure 17. Enhancement of identification accuracy with dropout.

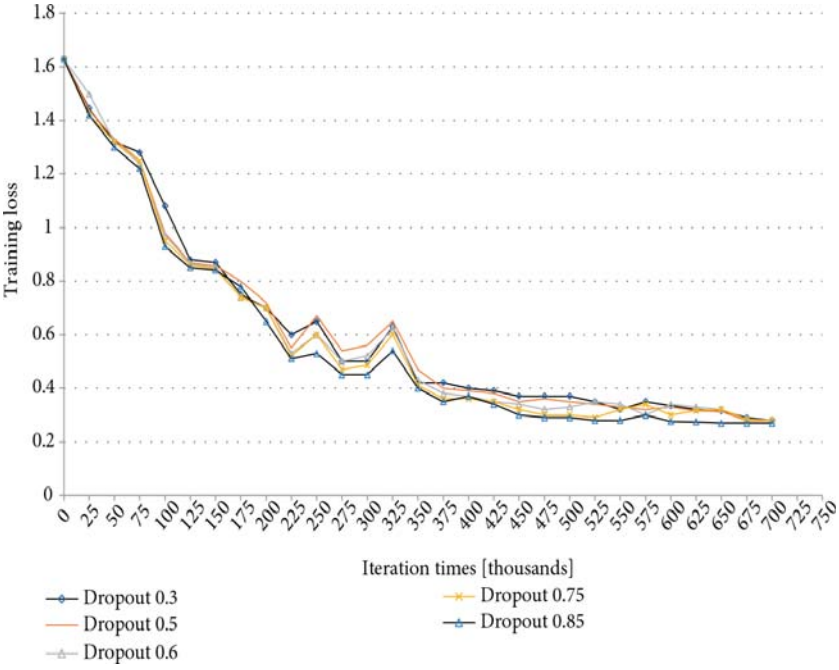


Figure 18. Effect of dropout on the training loss.

RESULTS AND DISCUSSION

The experiment implementations are based on the publicly available Tensorflow toolbox and Python language programming. THe hardware platform is built on a GeForce GTX 745 GPU with 4G memory. The experiment is carried out by the three criterions aforementioned, the performance, training loss, and testing loss are compared respectively.

Experimental Results from Data Augmentation

The number of images is doubled by taking data augmentation transformation approach, which means this methodology could assist the machine to learn the feature more accurately. We used the test dataset to evaluate our model. It turns out that with data augmentation, the machine can identify the objects of interest more accurately than the result without data augmentation, the experiment result is clearly shown in Figure 14.

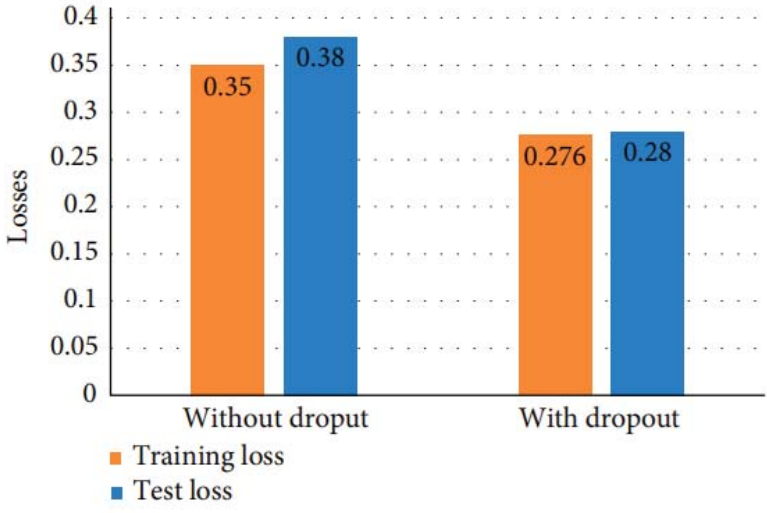


Figure 19. Effect of dropout on overfitting.

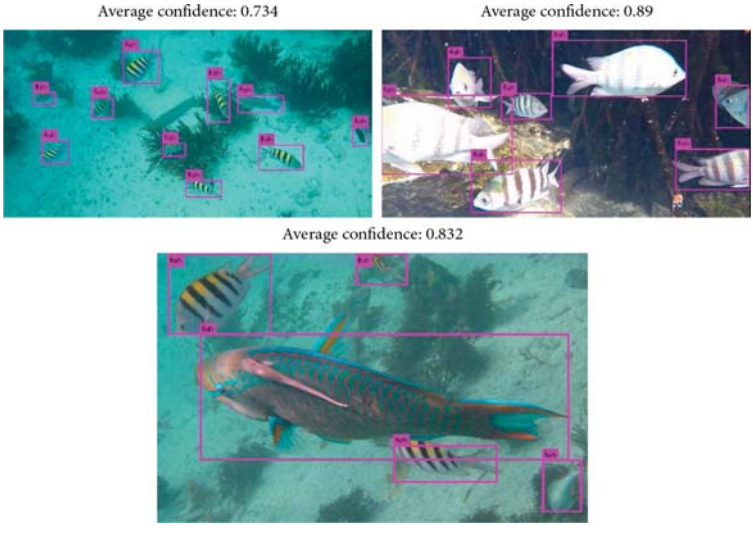


Figure 20. Enhancement of identification accuracy with the refinement of loss function.

From Figure 14, it is clear that the fish in the image is identified accurately; the average confidence is 0.568, 0.65, and 0.59 respectively for the three sample images.

From Figure 15, it is observed that the final training loss of the proposed neural network model using original data is 0.35 while the final training loss using data augmentation is 0.46. This clearly demonstrates that the data augmentation transformation is much helpful to reduce the training loss.

Figure 16 illustrates the training loss with the increase on the number of iterations. The iteration times are set from 0 to 600. e difference between training loss and test loss is decreased from 1.6 to 0.46 and 0.35 respectively. The overfitting problem was solved to a great extent.

Experimental Results from Dropout

In this test, neurons in the hidden layers were randomly selected to be removed from this network. In this wary, a simplified deep neural network is obtained. Figure 17 shows the average confidence for each sample image is greatly improved; the effectiveness of identification by dropout approach is highly enhanced.

The training loss of dropout is illustrated in Figure 18. As we can see, the final training loss is 0.28 with dropout, and 0.35 without. The dropout algorithm can help reduce the training loss. In this specific application, when dropout is 0.85, the training loss is smallest all the time (Figure 18).After conducted dropout approach, we can see from Figure 19, the difference between training loss and test loss decreased from 0.03 to 0.004. There is only a slight difference between training loss and test loss with dropout. The overfitting problem got resolved to a great extent. Therefore, the model we built is applicable.

Experimental Results from Loss Function Om Algorithm Performance

Authors used the CNN model with image segmentation and back-propagated the gradients of refined loss function and update the parameters in the network. With the refinement of loss function, the prediction gets more accurate as clearly shown in Figure 20.

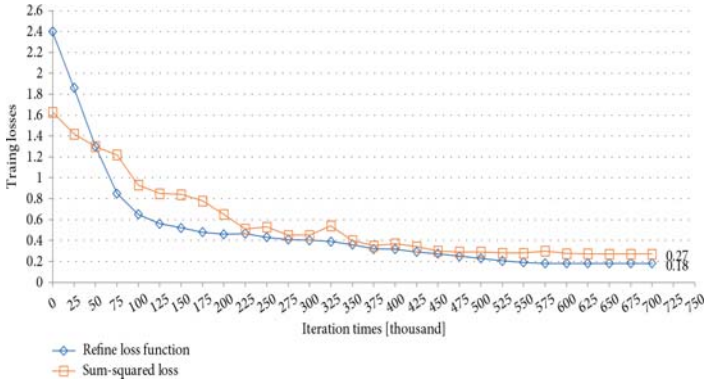


Figure 21. Effect of loss function on training loss.

As shown in Figure 21, when the iteration times are 575, it converges. The final training loss is 0.18. However, with sum squared loss function, the convergent point is 0.27 when the iteration times are 650.

Discussions

With the design and the choices of optimization, a deep learning based fish detection module was designed and simulated. With the improved accuracy and reduced processing time, it is very promising to adopt the proposed method to an AUV for implementation. The Tensorflow toolbox and Python programming interface are compatible with current advanced microcontroller platforms.

CONCLUSION

In this paper, authors built a neural network model to accomplish fish detection. To support the training process with enough dataset, the data augmentation approach was conducted. Dropout algorithm was selected to solve the overfitting problem. Moreover, loss function was refined to update the parameters inside the network. By these approaches, both the training time and the training loss were reduced dramatically. To summarize the contribution of this article: (1) Establish the data set to include real blur ocean water condition; (2) Revise loss function and other parameters in CNN to explore an applicable solution for fish detection; (3) The system is targeted at an embedded system for AUV design with all possible optimizations.

ACKNOWLEDGMENTS

This work was sponsored by the United States NSF grants #1332566, #1827243, and #1411260.

REFERENCES

1. R. B. Wynn, V. A. I. Huvenne, T. P. Le Bas et al., "Autonomous underwater vehicles (AUVs): their past, present and future contributions to the advancement of marine geoscience," Marine Geology, vol. 352, pp. 451–468, 2014. M. Dinc and C. Hajiyev, "Integration of navigation systems for autonomous underwater vehicles," Journal of Marine Engineering & Technology, vol. 14, no. 1, pp. 32–43, 2015.
2. Y. Zhou, S. Cui, Y. Wang, and C. Ai, "Design of autonomous underwater vehicle (AUV) control unit," in 2015 ASEE GulfSouthwest Annual Conference, pp. 25–27, ASEE Gulf-South, San Antonio, TX, 2015.
3. Y. Zhou, S. Cui, Y. Wang, and L. Zhai, "A refined attitude algorithm for AUV based on IMU," in 15th International Conference on Scientific Computing (CSC'17), pp. 16–22, CSREA Press ©, Las Vegas, NV, 2017.
4. SAUVC, https://sauvc.org/#competition.
5. A. Cadena, P. Teran, G. Reyes, J. Lino, V. Yaselga, and S. Vera, "Development of a hybrid autonomous underwater vehicle for benthis monitoring," in Proceedings of 2018 4th International Conference on Control, Automation and Robotics (ICCAR), pp. 20–23, IEEE, Auckland, New Zealand, 2018.
6. M. Eichhorn, H. C. Woithe, and U. Kremer, "Parallel of path planning algorithms for auvs concepts, opportunities, and program – technical implementation," in 2012 Oceans - Yeosu, MTS/IEEE, Yeosu, South Korea, 2012.
7. H. Taka, T. Sasaki, and M. Wada, "Supporting system for fishry resource management utilizing convolutional neural network," in 20th International Symposium on Woreless Personal Multimedia Communications (WPMC), pp. 442–447, IEEE, Bali, Indonesia, 2017.
8. J. Kim, H. Cho, J. Pyo, and B Yu S-C Kim, "e convolution neural network based agent vehicle detection using forwardlooking sonar image," in OCEANS 2016 MTS/IEEE Monterey, IEEE, CA, USA, 2016.
9. F. Xu, X. Ding, J. Peng et al., "Real-time detecting method of marine small object with underwater robot vision," in 2018 OCEANS – MTS/IEEE Kobe Techno-Oceans (OTO), IEEE, Kobe, Japan, 2018.
10. C.-F. Chien, Y.-J. Chen, Y.-T. Han et al., "AI and big data analytics for wafer fab energy saving and chiller optimization to empower

intelligent manufacturing," in Proceedings of 2018 e-Manufacturing & Design Collaboration Symposium (eMDC), pp. 1–4, IEEE, Hsinchu, Taiwan, 2018.

11. J. Jia, "A machine vision application for industrial assembly inspection," in Proceedings of 2009 Second International Conference on Machine Vision, pp. 172–176, IEEE, Dubai, UAE, 2009.
12. S. Biswas, Y. Wang, and S. Cui, "Surgically altered face detection using log-gabor wavelet," in Proceedings of the 12th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), pp. 154–157, IEEE, Chengdu, China, 2015.
13. S. Cui, O. Ekwonnah, and Y. Wang, "e analysis of emotions over keystroke dynamics," in 2018 International Conference on Information, Electronic and Communication Engineering (IECE2018), pp. 28–29, DEStech Publications, Beijing, China, 2018.
14. S. Cui, Y. Wang, and O. Ekwonnah, "Keystroke dynamics on user authentication," in 4th International Conference on Cybernetics (CYBCONF), pp. 5–7, Beijing, China, 2019.
15. Eman Abdel-Maksoud, Mohammed Elmogy, and Rashid AlAwadi, "Brain tumor segmentation based on a hybrid clustering technique," Egyptian Informatics Journal, vol. 16, no. 1, pp. 71–81, Cairo University, 2015.
16. Y. Wang, Y. Lan, Y. Zheng, K. Lee, S. Cui, and J. Lian, "A UGVbased laser scanner system for measuring tree geometric characteristics," vol. 8905, in Proceedings of 2013 SPIE International Symposium on Photoelectronic Detection and Imaging, SPIE, Beijing China, 2013.
17. B. Marr, "Key milestones of Waymo – Google's self-driving cars," https://www.forbes.com/sites/bernardmarr/2018/09/21/keymilestones-of-waymo-googles-self-driving-cars/#3831b2965369.
18. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.
19. BBC News, "Artificial intelligence: Google's AlphaGo beats Go master Lee Se-dol," 2016.
20. About ImageNet, http://image-net.org/about-overview.
21. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Advances in Neural

Information Processing Systems, vol. 25, pp. 1106–1114, 2012.

22. J. Deng, A. Berg, S. Satheesh, H. Su, A. Khosla, and F. Li, "ImageNet large scale visual recognition competition 2012 (ILSVRC2012)," http://www.imagenet.org/challenges/LSVRC/2012/.
23. Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with deep learning: a review," IEEE Transactions on Neural Networks and Learning Systems, vol. 30, no. 11, pp. 3212–3232, 2019.
24. R. Girshick, "Fast R-CNN," in 2015 IEEE International Conference on Computer Vision (ICCV), pp. 1440–1448, IEEE, Santiago, Chile, 2015.
25. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: unified, real-time object detection," in Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 779–788, IEEE, Las Vegas, NV, USA, 2016.
26. Q. Peng, W. Luo, G. Feng et al., "Pedestrian detection for transformer substation based on gaussian mixture model and YOLO," in 2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), pp. 562– 565, IEEE, Hangzhou, China, 2016.
27. S. Hassairi, R. Ejbali, and M. Zaied, "A deep convolutional neural wavelet network to supervised Arabic letter image classification," in 2015 15th International Conference on Intelligent Systems Design and Applications (ISDA), pp. 207–212, IEEE, Marrakech, Morocco, 2015.
28. D. Zhang, G. Kopanas, C. Desai, S. Chai, and M. Piacentino, "Unsupervised underwater fish detection fusing flow and objectiveness," in 2016 IEEE Winter Applications of Computer Vision Workshops (WACVW), pp. 1–7, IEEE, Lake Placid, NY, USA, 2016.
29. Convolutional neural networks for recognition, https:// en.wikipedia.org/wiki/Kernel_(image_processing).
30. Kernel (image processing), https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/.
31. Deep learning and machine learning, https://ireneli. eu/2016/02/03/deep-learning-05-talk-about-convolutionalneural-network.
32. M. Hu, Y. Yang, F. Shen, L. Zhang, H. Shen, and X. Li, "Robust web image annotation via exploring multi-facet and structural knowledge," IEEE Transactions on Image Processing, vol. 26, no. 10, pp. 4871–4884, 2017.

33. J. Gaya, L. T. Gonçalves, A. Duarte, B. Zanchetta, P. Drews, and S. Botelho, "Vision-based obstacle avoidance using deep learning," in 2016 XIII Latin American Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR), pp. 7–12, IEEE, Recife, Brazil, 2016.
34. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," Journal of Machine Learning Research, vol. 15, pp. 1929–1958, 2014.

CHAPTER 7

CAN DEEP LEARNING IDENTIFY TOMATO LEAF DISEASE?

Keke Zhang[1], Qiufeng Wu[2]., Anwang Liu[1], and Xiangyan Meng[2]

[1]College of Engineering, Northeast Agricultural University, Harbin 150030, China

[2]College of Science, Northeast Agricultural University, Harbin 150030, China

ABSTRACT

This paper applies deep convolutional neural network (CNN) to identify tomato leaf disease by transfer learning. AlexNet, GoogLeNet, and ResNet were used as backbone of the CNN. The best combined model was utilized to change the structure, aiming at exploring the performance of full training and fine-tuning of CNN. The highest accuracy of 97.28% for identifying tomato leaf disease is achieved by the optimal model ResNet with stochastic gradient descent (SGD), the number of batch size of 16, the number of

Citation: Keke Zhang, Qiufeng Wu, Anwang Liu, Xiangyan Meng, "Can Deep Learning Identify Tomato Leaf Disease?", Advances in Multimedia, vol. 2018, Article ID 6710865, 10 pages, 2018. https://doi.org/10.1155/2018/6710865.

iterations of 4992, and the training layers from the 37 layer to the fully connected layer (denote as "fc"). The experimental results show that the proposed technique is effective in identifying tomato leaf disease and could be generalized to identify other plant diseases.

INTRODUCTION

Tomato is a widely cultivated crop throughout the world, which contains rich nutrition, unique taste, and health effects, so it plays an important role in the agricultural production and trade around the world. Given the importance of tomato in the economic context, it is necessary to maximize productivity and product quality by using techniques. Corynespora leaf spot disease, early blight, late blight, leaf mold disease, septoria leaf spot, two-spotted spider mite, virus disease, and yellow leaf curl disease are 8 common diseases in tomato [1–8]; thus, a real time and precise recognition technology is essential.

Recently, since CNN has the self-learned mechanism, that is, extracting features and classifying images in the one procedure [9], CNN has been successfully applied in various applications, such as writer identification [10], salient object detection [11, 12], scene text detection [13, 14], truncated inference learning [15], road crack detection [16, 17], biomedical image analysis [18], predicting face attributes from web images [19], and pedestrian detection [20], and achieved the better performance. In addition, CNN is able to extract more robust and discriminative features with considering the global context information of regions [10], and CNN is scarcely affected by the shadow, distortion, and brightness of the natural images. With the rapid development of CNN, many powerful architectures of CNN emerged, such as AlexNet [21], GoogLeNet [22], VGGNet [23], Inception-V3 [24], Inception-V4 [25], ResNet [26], and DenseNets [27].

Training deep neural networks from scratch needs amounts of data and expensive computational resources. Meanwhile, we sometimes have a classification task in one domain, but we only have enough data in other domains. Fortunately, transfer learning can improve the performance of deep neural networks by avoiding complex data mining and data-labeling efforts [28]. In practice, transfer learning consists of two ways [29]. One option is to fine-tune the networks weights by using our data as input; it is worth nothing that the new data must be resized to the input size of the pretrained network. Another way is to obtain the learned weights from the pretrained network and apply the weights to the target network.

In this work, first, we compared the performance between SGD [30] and Adaptive Moment Estimation (Adam) [30, 31] in identifying tomato leaf disease. These optimization methods are based on the pretrained networks AlexNet [21], GoogLeNet [22], and ResNet [26]. Then, the network architecture with the highest performance was selected and experiments on effect of two hyperparameters (i.e., batch size and number of iterations) on accuracy were carried out. Next, we utilized the network with the suitable hyperparameters, which was obtained from the previous experiments, to discuss the impact of different network structures on recognition tasks. We believe this makes sense for researchers who choose to fine-tune pretrained systems for other similar issues.

The rest of this paper is organized as follows. Section 2 displays an overview of related works. Section 3 introduces the dataset and three deep convolutional neural networks, i.e., AlexNet, GoogLeNet, and ResNet. Section 4 presents the experiments and results in this work. Section 5 concludes the paper.

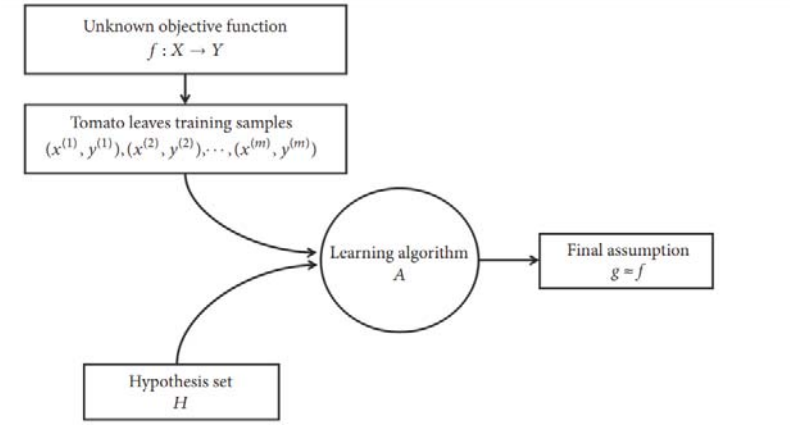


Figure 1. Proposed workfow diagram.

RELATED WORK

The research of agricultural disease identification based on computer vision has been a hot topic. In the early years, the traditional machine learning methods and shallow networks were extensively adopted in the agricultural field.

Sannakki et al. [32] proposed to use k-means based clustering performed on each image pixel to isolate the infected spot. They obtained the result that the Grading System they built by machine vision and fuzzy logic is very useful for grading the plant disease. Samanta et al. [33] proposed a novel histogram based scab diseases detection of potato and applied color image segmentation technique to exact intensity pattern. They got the best classification accuracy of 97.5%. Pedro et al. [34] applied fuzzy decision-making to identify weed shape, with fuzzy multicriteria decision-making strategy; they achieved the best accuracy of 92.9%. Cheng and Matson [35] adopted Decision Tree, Support Vector Machine (SVM), and Neural Network to identify weed and rice; the best accuracy they achieved is 98.2% by using Decision Tree. Sankaran and Ehsani [36] used quadratic discriminant analysis (QDA) and k-nearest neighbour (kNN) to classify citrus leaves infected with canker and Huanglongbing (HLB) from healthy citrus leaves; they got the highest overall accuracy of 99.9% by kNN.

Recently, deep learning methods have been applied in identifying plant disease widely. Cheng et al. [37] used ResNet and AlexNet to identify agricultural pests. At the same time, they carried out comparative experiments with SVM and BP neural networks; finally, they got the best accuracy of 98.67% by ResNet-101. Ferreiraa et al. [38] utilized ConvNets to perform weed detection in soybean crop images and classify these weeds among grass and broadleaf. The best accuracy they achieved is 99.5%. Sladojevic et al. [39] built a deep convolutional neural network to automatically classify and detect 15 categories of plant leaf diseases. Meanwhile, their model was able to distinguish plants from their surroundings. They got an average accuracy of 96.3%. Mohanty et al. [40] trained a deep convolutional neural network based on the pretrained AlexNet and GoogLeNet to identify 14 crop species and 26 diseases. They achieved an accuracy of 99.35% on a held-out test set. Sa et al. [41] proposed a novel approach to fruit detection by using deep convolutional neural networks. They adapted Faster Region-based CNN (Faster R-CNN) model, through transfer learning. They got the F1 score with 0.83 in a field farm dataset.

MATERIALS AND METHODS

This paper concentrates on identifying tomato leaf disease by deep learning. In this section, the abstract mathematical model about identifying tomato leaf disease is displayed at first. Meanwhile, the process of typical CNN is described with formulas. Then, the dataset and data augmentation are

presented. Finally, we introduced three powerful deep neural networks adopted in this paper, i.e., AlexNet, GoogLeNet, and ResNet.

The main process of tomato leaf disease identification in this work can be abstracted as a mathematical model (see Figure 1). First, we assume the mapping function from tomato leaves to diseases is $f : X \longrightarrow Y$ and then send the training samples to the optimization method. The hypothesis set H means possible objective functions with diferent parameters; through a series of parameters update, we can get the final assumption $g \approx f$.

Figure 2. Raw tomato leaf images.

The typical CNN process can be represented with following formulas. Firstly, send the training samples (i.e., training tomato leaf images) to the classifer (i.e., AlexNet, GoogLeNet, and ResNet). Ten, convolution operation is carried out; that is, a number of filters slide over the feature map of the previous layer, and the weight matrices do dot product.

$$M_j^l = f\left(\sum_{i \epsilon N_j} M_i^{l-1} * w_j^l + b_j^l\right) \tag{1}$$

where f (·) is activation function, typically a Rectifer Linear Unit (ReLU) [42] function:

$$f(x) = \max(x, 0) \tag{2}$$

N_j is the number of kernels of the certain layer, M_i^{l-1} represents the feature map of the previous layer, w_j^l is the weight matrix, and b_j^l is the bias term. Max-pooling or average pooling is conducted after the convolution opera-

tion. Furthermore, the learned features are sent to the fully connected layer. The softmax regression always follows the final fully connected layer, an input x will get the probability of belonging to class i.

$$p(y = i \mid x;\theta) = \frac{e^{\theta_i^T x}}{\sum_{j=1}^{k} e^{\theta_i^T x}} \quad (3)$$

where y is the response variable (i.e., predict label), k is the number of categories, and θ is the parameters of our model.

Raw Dataset

The raw tomato leaf dataset utilized in this work comes from an open access repository of images, which focus on plant health [43]. Health and other 8 diseases categories are included (see Table 1, Figure 2), i.e., early blight (pathogen: Alternaria solani) [1], yellow leaf curl disease (pathogen: Tomato Yellow Leaf Curl Virus (Tylcv), Family Geminiviridae, Genus Begomovirus) [2], corynespora leaf spot disease (pathogen: Corynespora cassiicola) [3], leaf mold disease (pathogen: Fulvia fulva) [4], virus disease (pathogen: Tomato Mosaic Virus) [5], late blight (pathogen: Phytophthora Infestans)[6], septoria leaf spot (pathogen: Septoria lycopersici) [7], and two-spotted spider mite (pathogen: Tetranychus urticae) [8]. Te total dataset is 5550.

Data Augmentation

Deep convolutional neural networks contain millions of parameters; thus, massive amounts of data is required. Otherwise, the deep neural network may be overftting or not robust. The most common method to reduce overftting on image dataset is to enlarge the dataset manually and conduct label-preserving transformations [21, 44].

In this work, at first, the raw image dataset was divided into 80% training samples and 20% testing samples, and then the data augmentation procedure was conducted: (1) fipping the image from left to right; (2) fipping the image from top to bottom; (3) fipping the image diagonally; (4) adjusting the brightness of image, setting the max delta to 0.4; (5) adjusting the contrast of image, setting the ratio from 0.2 to 1.5; (6) adjusting the hue of image, setting the max delta to 0.5; (7) adjusting the saturation of image, setting the ratio from 0.2 to 1.5; (8) rotating the image by 90° and 270° , respectively. The final dataset is shown in Table 2, and the label in the first row represents the disease categories which are given in Table 1.

Table 1. The raw tomato leaf dataset

Label	Category	Number	Leaf symptoms	Illustration
1	Corynespora leaf spot disease	547	Small brown spots appear, leaf spots have yellow halo.	See Figure 1 first row No.1-No.5
2	Early blight	405	Black or brown spots appear, leaf spots often have yellow or green concentric ring pattern.	See Figure 1 first row No.6-No.10
4	Late blight	726	Water-soaked area appears and rapidly enlarges to form purple-brown, oily-appearing blotches.	See Figure 1 second row No.1-No.5
5	Leaf mold disease	480	Irregular yellow or green area appears.	See Figure 1 second row No.6-No.10
6	Septoria leaf spot	734	Round spots, marginal brown, chlorotic yellow, appear.	See Figure 1 third row No.1-No.5
7	Two-spotted spider mite	720	Show white or yellow spots, blade back netting.	See Figure 1 third row No.6-No.10
8	Virus disease	481	Develop yellow or green, slightly shrinking.	See Figure 1 forth row No.1-No.5
9	Yellow leaf curl disease	814	Develop small and curl upward, crumpling, and marginal yellowing, bushy appearance.	See Figure 1 forth row No.6-No.10
3	Health	643		See Figure 1 fifth row
Total		5550		

Deep Learning Models

AlexNet

AlexNet is the winner of ImageNet LargeScale Visual Recognition Challenge (ILSVRC) 2012, a deep convolutional neural network, which has 60 million parameters and 650,000 neurons [21]. The architecture of AlexNet utilized in this paper is displayed in Figure 3. Te AlexNet architecture consists of five convolutional layers (i.e., conv1, conv2, and so on), some of which are followed by maxpooling layers (i.e., pool1, pool2, and pool5), three fully connected layers (i.e., fc6, fc7, and fc8), and a liner layer with sofmax activation in output. In order to reduce overftting in the fully connected layers, a regularization method called "dropout" is used (i.e., drop6, drop7) [21]. The ReLU activation function is applied to each of the first seven layers (i.e., relu1, relu2, and so on) [45]. In Figure 3, the notation $m \times m \times n$ in each convolutional layer represents the size of the feature map for each layer, 4096 represents the number of neurons of the first two fully connected layers. The number of neurons of the final fully connected layer was modified to 9, since the classification problem in this work has 9 categories. In addition, the size of input images must be shaped to 227×227, which meets the input pixel size requirement of AlexNet.

GoogLeNet

GoogLeNet is an inception architecture [22], which is the winner of ILSVRC 2014 and owns roughly 6.8 million parameters. Te architecture of GoogLeNet is presented in Figure 4. Te inception module is inspired by the network in network [46] and uses a parallel combination of 1×1, 3×3, and 5×5 convolutional layer along with 3×3 max-pooling layer [45]; the 1×1 convolutional layer before 3×3 and 5×5 convolutional layer reduces the spatial dimension and limits the size of GoogLeNet. Te whole architecture of GoogLeNet is stacked by inception module on top of each other (See Figure 4), which has nine inception modules, two convolutional layers, four max-pooling layers, one average pooling layer, one fully connected layer, and a linear layer with sofmax function in the output. GoogLeNet uses dropout regularization in the fully connected layer and applies the ReLU activation function in all of the convolutional layers [29]. In this work, the last three layers of GoogLeNet were replaced by a fully connected layer, a sofmax layer, and a classifcation layer; the fully connected layer was modifed to 9 neurons, which is equal to the categories in the tomato leaf disease identifcation problem. The size requested of input image of GoogLeNet is 224 × 224.

Table 2. The final tomato leaf dataset.

Labels	Label1	Label2	Label3	Label4	Label5	Label6	Label7	Label8	Label9	Total
Training set	3933	2916	4626	5229	3456	5283	5184	3465	5859	39951
Testing set	110	81	129	145	96	147	144	161	163	1176

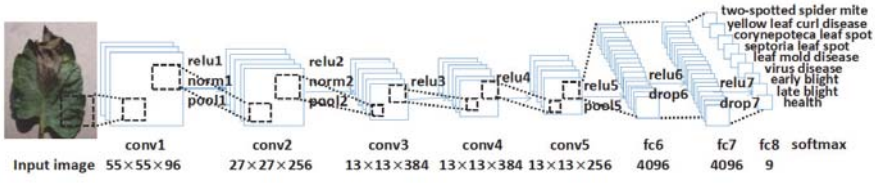


Figure 3. The architecture of AlexNet in this work.

ResNet

The deep residual learning framework is proposed for addressing the degradation problem. ResNet consists of many stacked residual units, which won the first place in ILSVRC 2015 and COCO 2015 classification challenge with error rate of 3.57% [26]. Each unit can be expressed in the following formulas [47]:

$$y_l = h(x_l) + F(x_l, W_l) \tag{4}$$

$$x_{l+1} = f(y_l) \tag{5}$$

where x_l and x_{l+1} are input and output of the l-th unit, and F is a residual function. In [26] $h(x_l) = x_l$ is an identity mapping and f is a ReLU function [42]. A "bottleneck" building block is designed for ResNet (See Figure 5) and comprises two 1×1 convolutions with a 3×3 convolution in between and a direct skip connection bypassing input and output. The 1×1 layers are responsible for changing in dimensions. ResNet model has three types of layers with 50, 101, and 152. For saving computing resources and training time, we choose the ResNet50, which also has high performance. In this work, at first, the last three layers of ResNet were modifed by a fully connected layer, a softmax layer, and a classification layer, the fully connected layer was replaced to 9 neurons, which is equal to the categories of the tomato leaf disease. We changed the structure of ResNet subsequently. The size of input image of ResNet should satisfy 224 × 224.

EXPERIMENTS AND RESULTS

In this section, we reveal the experiments and discuss the experimental results. All the experiments were implemented in Matlab under Windows 10, using the GPU NVIDIA GTX1050 with 4G video memory or NVIDIA GTX1080Ti with 11G video memory. In this paper, overall accuracy was regarded as the evaluation metric in every experiment on tomato leaf disease detection, which means the percentage of samples that are correctly classified:

$$\text{accuracy} = \frac{\text{true positive} + \text{true negative}}{\text{positive} + \text{negative}} \tag{6}$$

where "true positive" is the number of instances that are positive and classified as positive, "true negative" is the number of instances that are negative and classified as negative, and the denominator represents the total number of samples. In addition, the training time was regarded as an additional performance metric of the network structure experiment.

Experiments on Optimization Methods

The first experiment is designed for seeking the suitable optimization

method between SGD [30] and Adam [30, 31] in identifying tomato leaf diseases, combining with the pretrained network AlexNet, GoogLeNet, and ResNet, respectively. In this experiment, the hyperparameters were set as follows for each network: the batch size was set to 32, the initial learning rate was set to 0.001 and dropped by a factor of 0.5 every 2 epochs, and the max epoch was set to 5; i.e., the number of iterations is 6240. So far as SGD optimization method, the momentum was set to 0.9. For Adam, the gradient decay rate β_1 was set to 0.9, the squared gradient decay rate β_2 was set to 0.999, and the denominator offset ε was set to 10−8 [31]. The accuracy of different networks is displayed in Table 3. In addition, we choose the better results in each deep neural network to show the training loss against number of iterations during the fine-tuning process (See Figure 6). The words inside parenthesis indicate the corresponding optimization method.

In Table 3, the ResNet with SGD optimization method gets the highest test accuracy 96.51%. In identifying tomato leaf diseases, the performance of Adam optimization method is inferior to the SGD optimization method, especially in combining with AlexNet. In the following paper, AlexNet (SGD), GoogLeNet (SGD), and ResNet (SGD) are referred to as AlexNet, GoogLeNet, and ResNet, respectively.

As it can be seen in Figure 6, the training loss of ResNet drops rapidly in the earlier iterations and tends to stable afar 3000 iterations. Consistent with Table 3, the performance of AlexNet and GoogLeNet is similar and both inferior to the ResNet.

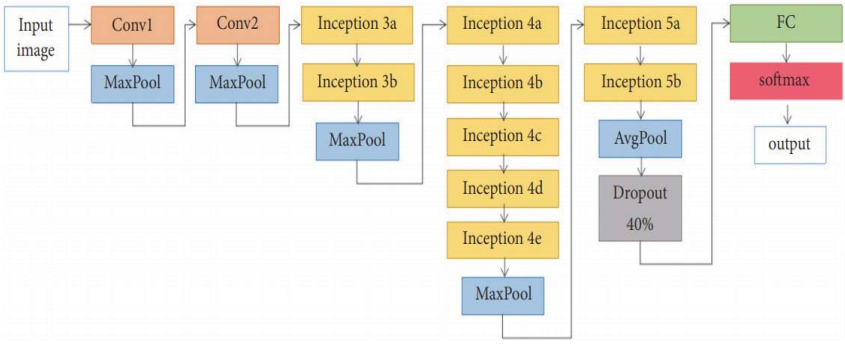


Figure 4. The architecture of GoogLeNet [22, 45].

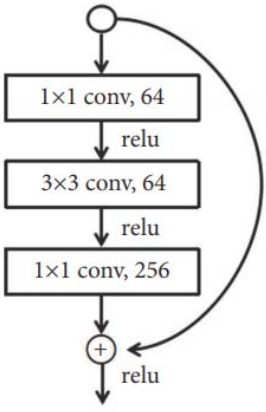


Figure 5. ResNet bottleneck residual building block [26].

Table 3. Model recognition accuracy.

Model	Accuracy
AlexNet (SGD)	95.83%
AlexNet (Adam)	13.86%
GoogLeNet (SGD)	95.66%
GoogLeNet (Adam)	94.06%
ResNet (SGD)	**96.51%**
ResNet (Adam)	94.39%

Experiments on Batch Size and Number of Iterations

From the experiment on optimization methods, the ResNet obtains the highest classification accuracy. Next, we evaluated the effects of batch size and the number of iterations on the performance of the ResNet. Te batch size was set to 16, 32, and 64, respectively. Meanwhile, the number of iterations was set to 2496, 4992, and 9984. The classification accuracy of different training scenarios is given in Table 4. At the same time, the classification accuracy of each label's representative leaf disease category (See Table 1) is given. In this experiment, the initial learning rate was set to 0.001 and dropped by a factor of 0.5 every 2496 iterations.

In Table 4, the best overall classifcation accuracy 97.19% is got by the ResNet combining with batch size 16 and

Iterations 4992. As shown in Table 4, whether increasing the number of iterations or batch size, the performance of corresponding models has not been improved significantly in identifying tomato leaf disease. A small batch size with a medium number of iterations is quite effective in this work. Moreover, a larger batch size and number of iterations increases the training duration. We have not tried higher or lower values for the attempted parameters, since different classification task may have various suitable parameters, and it is hard to give a certain rule in setting hyperparameters.

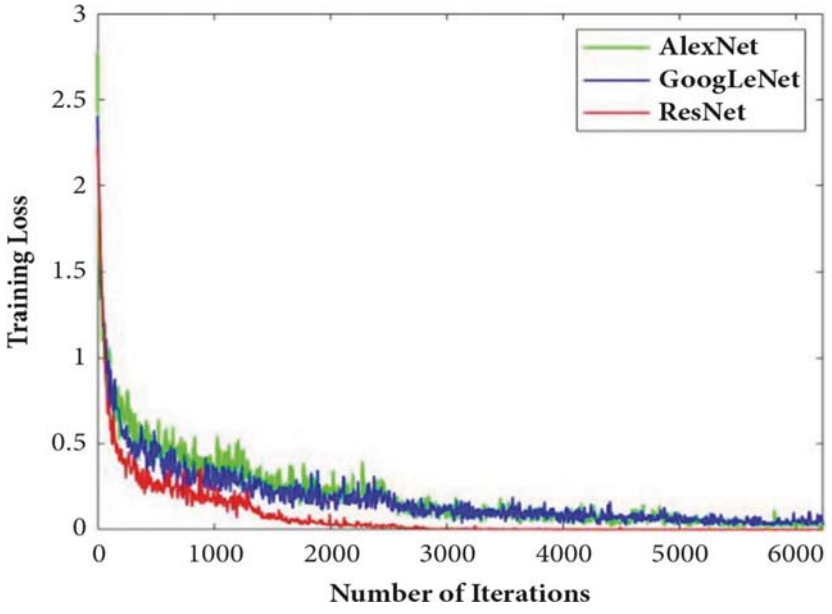


Figure 6. The training loss during the fine-tuning process.

Experiments on Full Training and Fine-Tuning of ResNet

This section is designed for exploring the performance of CNN by changing the structure of the models. In practical, a deep CNN always owns a large size which means a large number of parameters. Thus, full training of a deep CNN requires extensive computational resources and is time-consuming. In addition, full training of a deep CNN may led to overftting when the training data is limited. So we compared the performance of the pretrained CNN through full training and fine-tuning their structures.

We changed the structure of ResNet, and combination of the best parameters from the front experiments was utilized.

Table 4. Classification accuracies with different parameters during fine-tuning of the ResNet. The numbers inside parenthesis indicate batch size and number of iterations.

Networks	label1	label2	label3	label4	label5	label6	label7	label8	label9	overall
ResNet (16,2496)	90.91%	88.89%	100%	100%	96.88%	100%	90.28%	88.20%	97.55%	94.98%
ResNet (16,4992)	**98.18%**	**98.77%**	**100%**	**98.62%**	**96.88%**	**100%**	**96.53%**	**88.82%**	**98.77%**	**97.19%**
ResNet (16,9984)	98.18%	97.53%	100%	98.62%	96.88%	100%	97.22%	86.96%	98.77%	96.94%
ResNet (32,2496)	97.27%	95.06%	100%	97.93%	96.88%	100%	95.14%	86.34%	99.39%	96.34%
ResNet (32,4992)	97.27%	95.06%	100%	97.24%	96.88%	100%	96.53%	86.96%	99.39%	96.51%
ResNet (32,9984)	96.36%	96.30%	100%	99.31%	96.88%	100%	94.44%	88.20%	98.77%	96.60%
ResNet (64,2496)	93.64%	93.83%	100%	97.24%	96.88%	100%	94.44%	87.58%	99.39%	95.92%
ResNet (64,4992)	94.55%	95.29%	100%	96.55%	95.83%	100%	95.83%	86.96%	99.39%	95.83%
ResNet (64,9984)	95.45%	93.83%	100%	97.93%	96.88%	100%	94.44%	87.58%	99.39%	96.17%

ResNet50 has 177 layers if the layers for each building block and connection are calculated. In this experiment, the last three layers of ResNet were modified to a fully connected layer (denoted as “fc”), a softmax layer, and a classification layer, and the fully connected layer owns 9 neurons. The structure was changed by freezing the weights of a certain number of layers in the network by setting the learning rate in those layers to zero. During training, the parameters of the frozen layers are not updated. Full training and fine-tuning are defined by the number of training layers, i.e., full training (1-“fc”), fine-tuning (37-“fc”, 79-“fc”, 111-“fc”, 141- “fc”, 163-“fc”). The accuracy and training time of different network structure are presented in Table 5. At first, the batch size and 4992 iterations were combined; the initial learning rate was set to 0.001 and dropped by a factor of 0.1 every 2496 iterations. In order to get more convincing conclusions, ResNet (16, 9984), which gets the second place in Table 4, was also used to execute the experiments.

In Table 5, the accuracy and training time of different network structures are presented. In two cases, i.e., the 4992 iterations and 9984 iterations of ResNet, the accuracy of the model from the 37 layer fne-tuning structure are higher than that of the full training model. In the case where the number of iterations is 4992, the accuracy of the model from the 79 layer fine-tuning structure is equal to that of the full training model.The fonal column of the Table 5 represents the training time of the corresponding network, and it is clear that the training time of the fine-tuning models is greatly lowered than the full training model. Because the gradients of the frozen layers do not need to be computed, freezing the weights of initial layers can speed up network training. We observe that the moderate fine-tuning models (37-“fc”, 79-“fc”, 111-“fc”) always led to a performance superior or approximately equal to the full training models. Thus, we suggest that, for practical application,

the moderate fine-tuning models may be a good choice. Especially for the researcher who holds massive data, the fine-tuning models may achieve good performance while saving computational resources and time.

Moreover, the features of the final fully connected layer of ResNet (16, 4992, 37-"fc") were examined by utilizing the t-distributed Stochastic Neighbor Embedding (t-SNE) algorithm (see Figure 7) [48]. 1176 test images were used to extract the features. In Figure 7, different colors represent different labels; the corresponding disease categories of the labels were listed in Table 1. As shown in Figure 7, 9 different color points are clearly separated, which indicates that the features learned from the ResNet with the optimal structure can be used to classify the tomato leaf disease precisely.

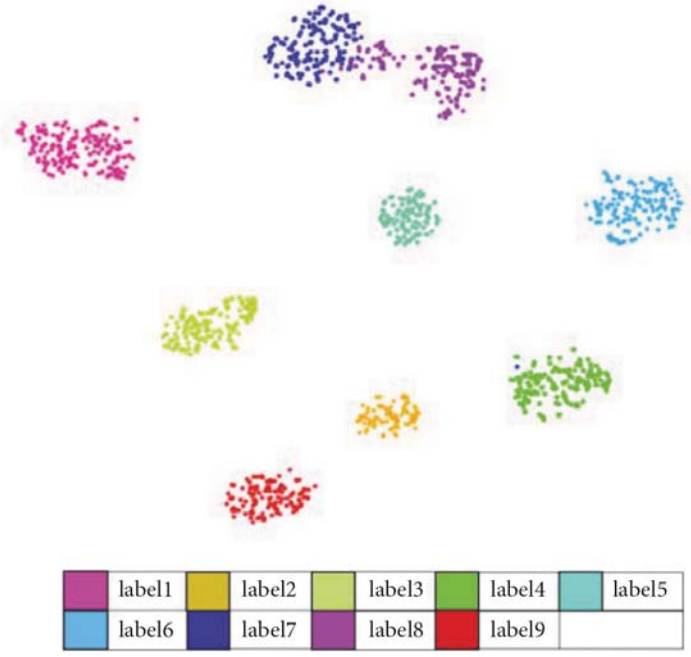


Figure 7. Two-dimensional scatter plot of high-dimensional features generated with t-SNE.

CONCLUSION

Tis paper concentrates on identifying tomato leaf disease using deep convolutional neural networks by transfer learning. The utilized networks are based on the pretrained deep learning models of AlexNet, GoogLeNet, and ResNet. First we compared the relative performance of these networks by using SGD and Adam optimization method, revealing that the ResNet with SGD optimization method obtains the highest result with the best accuracy, 96.51%. Ten, the performance evaluation of batch size and number of iterations affecting the transfer learning of the ResNet was conducted. A small batch size of 16 combining a moderate number of iterations of 4992 is the optimal choice in this work. Our findings suggest that, for a particular task, neither large batch size nor large number of iterations may improve the accuracy of the target model. The setting of batch size and number of iterations depends on your data set and the utilized network. Next, the best combined model was used to fine-tune the structure. Fine-tuning ResNet layers from 37 to "fc" obtained the highest accuracy 97.28% in identifying tomato leaf disease. Based on the amount of available data, layer-wise fine-tuning may provide a practical way to achieve the best performance of the application at hand. We believe that the results obtained in this work will bring some inspiration to other similar visual recognition problems, and the practical study of this work can be easily extended to other plant leaf disease identification problems.

Table 5. Accuracies and training time in different network structures. The values inside parenthesis denote batch size, number of iterations, and training layers.

Network topology	Accuracy	Time (min:sec)
ResNet (16, 4992, 1-"fc")	96.43%	59min 30sec
ResNet (16, 4992, 37-"fc")	**97.28%**	**44min 13sec**
ResNet (16, 4992, 79-"fc")	96.43%	37min 27sec
ResNet (16, 4992, 111-"fc")	95.75%	30min 6sec
ResNet (16, 4992, 141-"fc")	95.32%	24min 15sec
ResNet (16, 4992, 163-"fc")	92.69%	19min 31sec
ResNet (16, 9984, 1-"fc")	96.94%	118min 32sec
ResNet (16, 9984, 37-"fc")	**97.02%**	**92min 53sec**
ResNet (16, 9984, 79-"fc")	96.77%	72min 23sec
ResNet (16, 9984, 111-"fc")	96.26%	58min 40sec
ResNet (16, 9984, 141-"fc")	95.75%	47min 22sec
ResNet (16, 9984, 163-"fc")	93.96%	39min 32sec

ACKNOWLEDGMENTS

This study was supported by the National Science and technology support program (2014BAD12B01-1-3), Public Welfare Industry (Agriculture) Research Projects Level-2 (201503116- 04-06), Postdoctoral Foundation of Heilongjiang Province (LBHZ15020), Harbin Applied Technology Research and Development Program (2017RAQXJ096), and Economic Decision Making and Early Warning of Soybean Industry in Technology Collaborative Innovation System of Soybean Industry in Heilongjiang Province (20170401).

REFERENCES

1. R. Chaerani and R. E. Voorrips, "Tomato early blight (Alternaria solani): Te pathogen, genetics, and breeding for resistance," Journal of General Plant Pathology, vol. 72, no. 6, pp. 335–347, 2006.
2. A. M. Dickey, L. S. Osborne, and C. L. Mckenzie, "Papaya (Carica papaya, Brassicales: Caricaceae) is not a host plant of tomato yellow leaf curl virus (TYLCV; family Geminiviridae, genus Begomovirus)," Florida Entomologist, vol. 95, no. 1, pp. 211–213, 2012.
3. G. Wei, L. Baoju, S. Yanxia, and X. Xuewen, "Studies on pathogenicity diferentiation of corynespora cassiicola isolates, against cucumber, tomato and eggplant," Acta Horticulturae Sinica, vol. 38, no. 3, pp. 465–470, 2011.
4. P. Lindhout, W. Korta, M. Cislik, I. Vos, and T. Gerlagh, "Further identifcation of races of Cladosporium fulvum (Fulvia fulva) on tomato originating from the Netherlands France and Poland," Netherlands Journal of Plant Pathology, vol. 95, no. 3, pp. 143–148, 1989.
5. K. Kubota, S. Tsuda, A. Tamai, and T. Meshi, "Tomato mosaic virus replication protein suppresses virus-targeted posttranscriptional gene silencing," Journal of Virology, vol. 77, no. 20, pp. 11016–11026, 2003.
6. M. Tian, B. Benedetti, and S. Kamoun, "A second Kazallike protease inhibitor from Phytophthora infestans inhibits and interacts with the apoplastic pathogenesis-related protease P69B of tomato," Plant Physiology, vol. 138, no. 3, pp. 1785–1793, 2005.
7. L. E. Blum, "Reduction of incidence and severity of Septoria lycopersici leaf spot of tomato with bacteria and yeasts," Ci^encia Rural, vol. 30, no. 5, pp. 761–765, 2000.
8. E. A. Chatzivasileiadis and M. W. Sabelis, "Toxicity of methyl ketones from tomato trichomes to Tetranychus urticae Koch," Experimental and Applied Acarology, vol. 21, no. 6-7, pp. 473– 484, 1997.
9. M. Anthimopoulos, S. Christodoulidis, L. Ebner, A. Christe, and S. Mougiakakou, "Lung Pattern Classifcation for Interstitial Lung Diseases Using a Deep Convolutional Neural Network," IEEE Transactions on Medical Imaging, vol. 35, no. 5, pp. 1207–1216, 2016.
10. Y. Tang and X. Wu, "Text-independent writer identifcation via CNN features and joint Bayesian," in Proceedings of the 15th International Conference on Frontiers in Handwriting Recognition, ICFHR 2016, pp. 566–571, Shenzhen, China, October 2016.

11. Y. Tang and X. Wu, “Saliency Detection via Combining RegionLevel and Pixel-Level Predictions with CNNs,” in Proceedings of the European Conference on Computer Vision (ECCV), pp. 1608– 05186, 2016.
12. Y. Tang and X. Wu, “Salient object detection with chained multi-scale fully convolutional network,” ACM Multimedia (ACMMM), pp. 618–626, 2017.
13. Y. Tang and X. Wu, “Scene text detection and segmentation based on cascaded convolution neural networks,” IEEE Transactions on Image Processing, vol. 26, no. 3, pp. 1509–1520, 2017.
14. Y. Tang and X. Wu, “Scene Text Detection using Superpixel based Stroke Feature Transform and Deep Learning based Region Classifcation,” IEEE Transactions on Multimedia, vol. 20, no. 9, pp. 2276–2288, 2018.
15. Y. Yao, X. Wu, Z. Lei, S. Shan, and W. Zuo, “Joint Representation and Truncated Inference Learning for Correlation Filter based Tracking,” in Proceedings of the European Conference on Computer Vision (ECCV), pp. 1–14, 2018.
16. L. Zhang, F. Yang, Y. Daniel Zhang, and Y. J. Zhu, “Road crack detection using deep convolutional neural network,” in Proceedings of the 23rd IEEE International Conference on Image Processing, ICIP 2016, pp. 3708–3712, Phoenix, AZ, USA, September 2016.
17. D. Xie, L. Zhang, and L. Bai, “Deep learning in visual computing and signal processing,” Applied Computational Intelligence and Sof Computing, vol. 2017, Article ID 1320780, 13 pages, 2017.
18. Z. Zhou, J. Shin, L. Zhang, S. Gurudu, M. Gotway, and J. Liang, “Fine-tuning convolutional neural networks for biomedical image analysis: Actively and incrementally,” in Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, pp. 4761–4772, USA, July 2017.
19. Z. Liu, P. Luo, X. Wang, and X. Tang, “Deep learning face attributes in the wild,” in Proceedings of the 15th IEEE International Conference on Computer Vision, ICCV 2015, pp. 3730–3738, Santiago, Chile, 2015.
20. W. Ouyang and X. Wang, “Joint deep learning for pedestrian detection,” in Proceedings of the 14th IEEE International Conference on Computer Vision (ICCV ’13), pp. 2056–2063, Sydney, Australia, December 2013.

21. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classifcation with deep convolutional neural networks," in Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS '12), pp. 1097–1105, Lake Tahoe, Nev, USA, December 2012.
22. C. Szegedy, W. Liu, Y. Jia et al., "Going deeper with convolutions," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '15), pp. 1–9, IEEE, Boston, Mass, USA, June 2015.
23. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," https://arxiv.org/ abs/1409.1556, 2015.
24. C. Szegedy, V. Vanhoucke, S. Iofe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," https://arxiv.org/ abs/1512.00567, 2015.
25. C. Szegedy, S. Iofe, V. Vanhoucke, and A. Alemi, "Inceptionv4, inception-ResNet and the impact of residual connections on learning," https://arxiv.org/abs/1602.07261, 2016.
26. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, pp. 770–778, July 2016.
27. G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," in Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2016.
28. S. J. Pan and Q. Yang, "A survey on transfer learning," IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 10, pp. 1345–1359, 2010. [29] M. Mehdipour Ghazi, B. Yanikoglu, and E. Aptoula, "Plant identifcation using deep neural networks via optimization of transfer learning parameters," Neurocomputing, vol. 235, pp. 228–235, 2017.
29. S. Ruder, "An overview of gradient descent optimization algorithms," https://arxiv.org/abs/1609.04747, 2017. [31] D. P. Kingma and J. L. Ba, "Adam: a method for stochastic optimization," https://arxiv.org/ abs/1412.6980, 2017.
30. S. S. Sannakki, V. S. Rajpurohit, V. B. Nargund, R. Arun Kumar, and S. Prema Yallur, "Leaf disease grading by machine vision and fuzzy logic," International Journal of Computer Technology and Applications, vol. 2, no. 5, pp. 1709–1716, 2011.

31. D. Samanta, P. P. Chaudhury, and A. Ghosh, “Scab diseases detection of potato using image processing,” International Journal of Computer Trends and Technology, vol. 3, pp. 109–113, 2012.
32. P. J. Herrera, J. Dorado, and A. Ribeiro, “A novel approach for ´ weed type classifcation based on shape descriptors and a fuzzy decision-making method,” Sensors, vol. 14, no. 8, pp. 15304– 15324, 2014.
33. B. Cheng and E. T. Matson, “A feature-based machine learning agent for automatic rice and weed discrimination,” International Conference on Artifcial Intelligence and Sof Computing, pp. 517– 527, 2015.
34. S. Sankaran and R. Ehsani, “Comparison of visible-near infrared and mid-infrared spectroscopy for classifcation of Huanglongbing and citrus canker infected leaves,” Agricultural Engineering International: CIGR Journal, vol. 15, no. 3, pp. 75–79, 2013.
35. X. Cheng, Y. Zhang, Y. Chen, Y. Wu, and Y. Yue, “Pest identifcation via deep residual learning in complex background,” Computers and Electronics in Agriculture, vol. 141, pp. 351–356, 2017.
36. A. dos Santos Ferreira, D. Matte Freitas, G. Gonc¸alves da Silva, H. Pistori, and M. Teophilo Folhes, “Weed detection in soybean crops using ConvNets,” Computers and Electronics in Agriculture, vol. 143, pp. 314–324, 2017.
37. S. Sladojevic, M. Arsenovic, A. Anderla, D. Culibrk, and D. Stefanovic, “Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classifcation,” Computational Intelligence and Neuroscience, vol. 2016, Article ID 3289801, 11 pages, 2016.
38. S. P. Mohanty, D. P. Hughes, and M. Salath´e, “Using deep learning for image-based plant disease detection,” Frontiers in Plant Science, vol. 7, article no. 1419, 2016.
39. I. Sa, Z. Ge, F. Dayoub, B. Upcrof, T. Perez, and C. McCool, “Deepfruits: A fruit detection system using deep neural networks,” Sensors, vol. 16, article no. 1222, no. 8, 2016.
40. V. Nair and G. E. Hinton, “Rectifed linear units improve Restricted Boltzmann machines,” in Proceedings of the 27th International Conference on Machine Learning (ICML ’10), pp. 807–814, Haifa, Israel, June 2010.
41. D. P. Hughes and M. Salathe, “An open access repository of images on plant health to enable the development of mobile disease diagnostics,” https://arxiv.org/abs/1511.08060, 2016.

42. D. Ciresan, U. Meier, J. Masci, and J. Schmidhuber, “A committee of neural networks for trafc sign classifcation,” in Proceedings of the 2011 International Joint Conference on Neural Networks (IJCNN 2011 - San Jose), San Jose, CA, USA, July 2011.

43. P. Pawara, E. Okafor, O. Surinta, L. Schomaker, and M. Wiering, “Comparing Local Descriptors and Bags of Visual Words to Deep Convolutional Neural Networks for Plant Recognition,” in Proceedings of the 6th International Conference on Pattern Recognition Applications and Methods, pp. 479–486, Porto, Portugal, Feburary 2017.

44. M. Lin, “Network in Nnetwork,” https://arxiv.org/abs/1312.4400, 2014.

45. K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deep residual networks,” in Proceedings of the European Conference on Computer Vision, pp. 630–645, 2016.

46. L. van der Maaten and G. Hinton, “Visualizing data using tSNE,” Journal of Machine Learning Research, vol. 9, pp. 2579– 2625, 2008.

CHAPTER 8

DEEP LEARNING FOR PLANT IDENTIFICATION IN NATURAL ENVIRONMENT

Yu Sun, Yuan Liu, Guan Wang, and Haiyan Zhang

School of Information Science and Technology, Beijing Forestry University, Beijing 100083, China

ABSTRACT

Plant image identification has become an interdisciplinary focus in both botanical taxonomy and computer vision. The first plant image dataset collected by mobile phone in natural scene is presented, which contains 10,000 images of 100 ornamental plant species in Beijing Forestry University campus. A 26-layer deep learning model consisting of 8 residual building blocks is designed for large-scale plant classification in natural environment. The proposed model achieves a recognition rate of 91.78% on the BJFU100 dataset, demonstrating that deep learning is a promising technology for smart forestry.

Citation: Yu Sun, Yuan Liu, Guan Wang, Haiyan Zhang, "Deep Learning for Plant Identification in Natural Environment", Computational Intelligence and Neuroscience, vol. 2017, Article ID 7361042, 6 pages, 2017. https://doi.org/10.1155/2017/7361042.

INTRODUCTION

Automatic plant image identification is the most promising solution towards bridging the botanical taxonomic gap, which receives considerable attention in both botany and computer community. As the machine learning technology advances, sophisticated models have been proposed for automatic plant identification. With the popularity of smartphones and the emergence of Pl@ntNet mobile apps [1], millions of plant photos have been acquired. Mobile-based automatic plant identification is essential to real-world social-based ecological surveillance [2], invasive exotic plant monitor [3], ecological science popularization, and so on. Improving the performance of mobile-based plant identification models attracts increased attention from scholars and engineers.

Nowadays, many efforts have been conducted in extracting local characteristics of leaf, flower, or fruit. Most researchers use variations on leaf characteristic as a comparative tool for studying plants, and some leaf datasets including Swedish leaf dataset, Flavia dataset, and ICL dataset are standard benchmark. In [4], Söderkvist extracted shape characteristics and moment features of the leaves and analyzed the 15 different Swedish tree classes using back propagation for the feed-forward neural network. In [5], Fu et al. chose the local contrast and other parameters to describe the characteristics of the surrounding pixels of veins. The artificial neural network was used to segment the veins and other leaves. The experiment shows that the neural network is more effective in identifying the vein images. Li et al. [6] proposed an efficient leaf vein extraction method by combining snakes technique with cellular neural networks, which obtained satisfactory results on leaf segmentation. He and Huang used the probabilistic neural network as a classifier to identify the plant leaf images, which has a better identification accuracy comparing to BP neural network [7]. In 2013, the idea of natural-based leaf recognition was proposed, and the method of contour segmentation algorithm based on polygon leaf model was used to obtain contour image [8]. With the deep learning becoming a hot spot in the field of image recognition, Liu and Kan proposed texture features in combination with shape characteristics, using deep belief network architecture as a classifier [9]. Zhang et al. designed a deep learning system which includes eight layers of Convolution Neural Network to identify leaf images and achieved a higher recognition rate. Some researchers focus on the flowers. Nilsback and Zisserman proposed a method of bag of visual word to describe the color, shape, texture features, and other characteristics [10]. In [11], Zhang et al. combined Harr features with SIFT features of

flower image, coding them with nonnegative sparse coding method and classifying them by *k*-nearest neighbor method. In [12], they raised a method of recognizing the picking rose by integrating BP neural network. The studies of identifying plants by fruit are relatively rare. Li et al. proposed the method of multifeature integration using preference Ainet as the recognition algorithm [13]. After so many years continued exploration into plant recognition technology, the dedicated mobile applications such as LeafSnap [14], Pl@ntNet [1], or Microsoft Garage's Flower Recognition app [15] can be conveniently used for identify plants.

Although the research on automatic plant taxonomy has yield fruitful results, one must note that those models are still far from the requirements of a fully automated ecological surveillance scenario [3]. The aforesaid datasets lack the mobile-based plant images acquired in natural scene which vary greatly in contributors, cameras, areas, periods of the year, individual plants, and so on. The traditional classification models rely heavily on preprocessing to eliminate complex background and enhance desiring features. What is more, the handcraft feature engineering is incapable of dealing with large-scale datasets consisting of unconstrained images.

To overcome aforementioned challenges and inspired by the deep learning breakthrough in image recognition, we acquired the BJFU100 dataset by mobile phone in natural environment. The proposed dataset contains 10,000 images of 100 ornamental plant species in Beijing Forestry University campus. A 26-layer deep learning model consisting of 8 residual building blocks is designed for uncontrolled plant identification. The proposed model achieves a recognition rate of 91.78% on the BJFU100 dataset.

PROPOSED BJFU100 DATASET AND DEEP LEARNING MODEL

Deep learning architectures are formed by multiple linear and nonlinear transformations of input data, with the goal of yielding more abstract and discriminative representations [16]. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection, and many other domains such as drug discovery and genomics [17]. The deep convolutional neural networks proposed in [18] demonstrated outstanding performance in the large-scale image classification task of ILSVRC-2012 [19]. The model was trained on more than one million images and has achieved a winning top-5 test error rate of 15.3% over 1,000 classes. It almost halved the error rates of the best competing approaches.

This success has brought about a revolution in computer vision [17]. Recent progress in the field has advanced the feasibility of deep learning applications to solve complex, real-world problems [20].

BJFU100 Dataset

The BJFU100 dataset is collected from natural scene by mobile devices. It consists of 100 species of ornamental plants in Beijing Forestry University campus. Each category contains one hundred different photos acquired by smartphone in natural environment. The smartphone is equipped with a prime lens of 28 mm equivalent focal length and a RGB sensor of 3120 × 4208 resolution.

For tall arbors, images were taken from a low angle at ground as shown in Figures 1(a)–1(d). Low shrubs were shot from a high angle, as shown in Figures 1(e)–1(h). Other ornamental plants were taken from a level angle. Subjects may vary in size by an order of magnitude (i.e., some images show only the leaf, others an entire plant from a distance), as shown in Figures 1(i)–1(l).

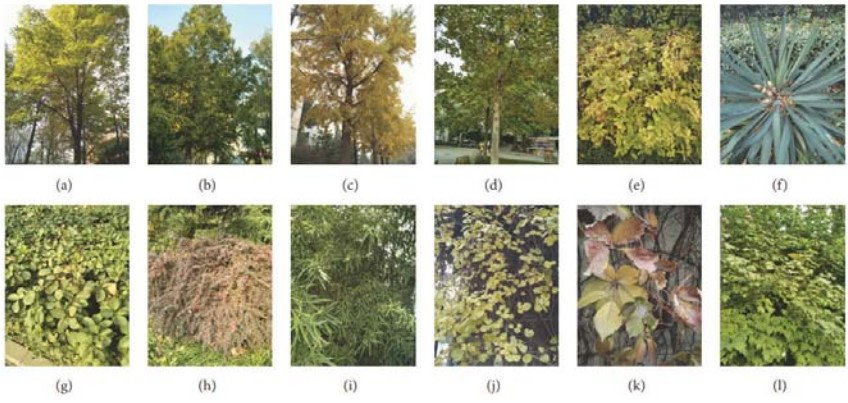


Figure 1. Example images of the BJFU100 dataset. (a) Chinese buckeye, (b) metasequoia, (c) Ginkgo biloba, (d) hybrid tulip tree, (e) Weigela florida cv. red-prince, (f) Yucca gloriosa, (g) Euonymus kiautschovicus Loes, (h) Berberis thunbergii var. atropurpurea, (i) mottled bamboo, (j) Celastrus orbiculatus, (k) Parthenocissus quinquefolia, and (l) Viburnum opulus.

The Deep Residual Network

With the network depth increasing, traditional methods are not as expected to improve accuracy but introduce problems like vanishing gradient

and degradation. The residual network, that is, ResNet, introduces skip connections that allow the information (from the input or those learned in earlier layers) to flow more into the deeper layers [23, 24]. With increasing depth, ResNets give better function approximation capabilities as they gain more parameters and successfully contribute to solving vanishing gradient and degradation problems. Deep residual networks with residual units have shown compelling accuracy and nice convergence behaviors on several large-scale image recognition tasks, such as ImageNet [23] and MS COCO [25] competitions.

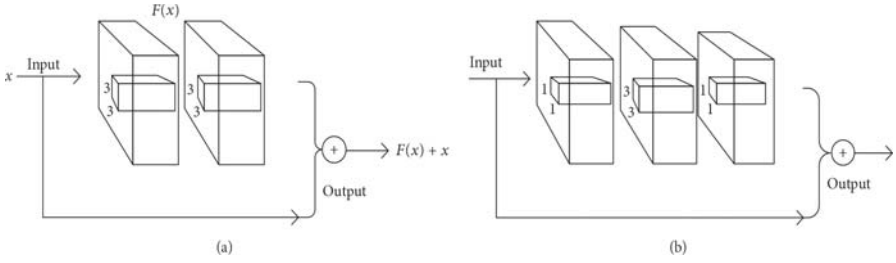


Figure 2. (a) A basic building block. (b) A "bottleneck" building block of deep residual networks.

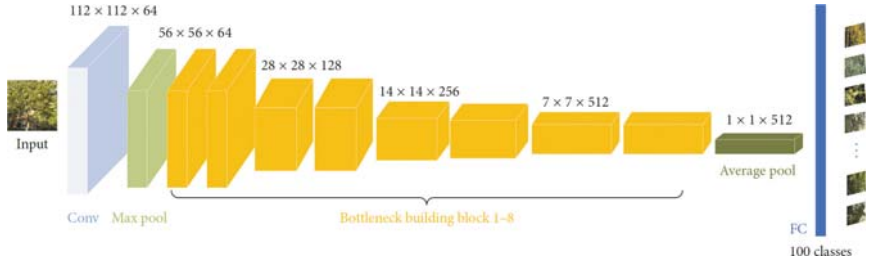


Figure 3. Architecture of 26-layer ResNet model for plant identification.

Residual Building Blocks

Residual structural unit utilizes shortcut connections with the help of identity mapping. Shortcut connections are those skipping one or more layers. The original underlying mapping can be realized by feed forward neural networks with shortcut connections. The building block illustrated in Figure 2 is defined as

$$y = F(x, \{W_i\}) + x,$$
$$F = W_2\sigma(W_1 x),$$
$$\sigma(a) = \max(0, a), \quad (1)$$

where x and y are the input and output vectors of stacked layers, respectively. The function $(x, \{Wi\})$ represents the residual mapping that needs to be learned. The function (a) denotes ReLU [26] and the biases are omitted for simplifying notations. The dimensions of x and F must be equal to perform the element-wise addition. If this is not the case, a linear projection Ws is applied to match the dimensions of x and F:

$$y = F(x, \{W_i\}) + W_s x. \quad (2)$$

The baseline building block is shown in Figure 2(a). A shortcut connection is added to each pair of 3 × 3 filters. Concerning the training time on deeper nets, a bottleneck building block is designed as in Figure 2(b). The three layers are 1 × 1, 3 × 3, and 1 × 1 convolutions, where the 1 × 1 layers are responsible for reducing and then restoring dimensions, leaving 3 × 3 layer a bottleneck with smaller input/output dimensions [23]. Bottleneck building blocks use fewer parameters to obtain more abstraction of layers.

The overall network architecture of our 26-layer ResNet, that is, ResNet26, model is depicted in Figure 3. As Figure 3 shows, the model is mainly designed by using bottleneck building blocks. The input image is fed into a 7 × 7 convolution layer and a 3 × 3 max pooling layer followed by 8 bottleneck building blocks. When the dimensions increase, 1 × 1 convolution is used in bottleneck to match dimensions. The 1 × 1 convolution enriches the level of abstraction and reduces the time complexity. The network ends with a global average pooling, a fully connected layer, and a softmax layer. We adopt batch normalization (BN) [27] right after each convolution layer and before ReLU [26] activation layer. Downsampling is performed by the first convolution layer, the max pooling layer, and the 3, 5, and 7 bottleneck building blocks.

EXPERIMENTS AND RESULTS

Implementation and Preprocess

The model implementation is based on the open source deep learning framework keras [28]. All the experiments were conducted on a Ubuntu

16.04 Linux server with a 3.40 GHz i7-3770 CPU (16 GB memory) and a GTX 1070 GPU (8 GB memory). The 100 samples of each class are split into 80 training samples and 20 test samples. Compared with conventional classification methods, data preprocess on deep learning approaches is much simpler. In this paper, the inputs to the network are RGB color images. All the images only need to be rescaled to 224 × 224 pixels and then per-pixel value is divided by 255.

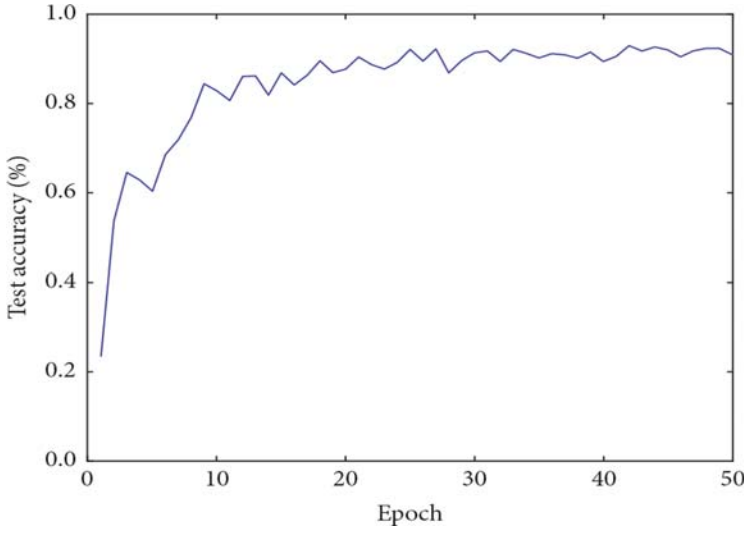


Figure 4. Evolution of classification accuracy in the test set.

Training Algorithm

During the back propagation phase, the model parameter is trained by the stochastic gradient descent (SGD) algorithm, with the categorical cross entropy loss function as optimization object. The SGD can be expressed as follows:

$$\delta_x = w_{x+1}\left(\sigma'\left(w_{x+1} \bullet c_x + b_{x+1}\right) \circ \mathrm{up}\left(\delta_{x+1}\right)\right),$$
$$\Delta w_x = -\eta \bullet \sum_{i,j}\left(\delta_x \circ \mathrm{down}\left(S_{x-1}\right)\right), \tag{3}$$

where δx is sensitivity, wx+1 is multiplicative bias, ∘ indicates that each element is multiplied, up is upsampling, down is downsampling,Δwx represents the weight update of the layer, and η is the learning rate. The cross-entropy loss function is defined to be

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right), \tag{4}$$

where f_j is the jth element in the classification score vector f.

After some preliminary training experiments, the base learning rate is set to 0.001, which is gradually reduced at each epoch. The decay rate is 10^{-6} and the momentum is 0.9. Figure 4 shows the training process of ResNet26 model. Test accuracy improves quickly since the first epochs and stabilizes after 40 epochs.

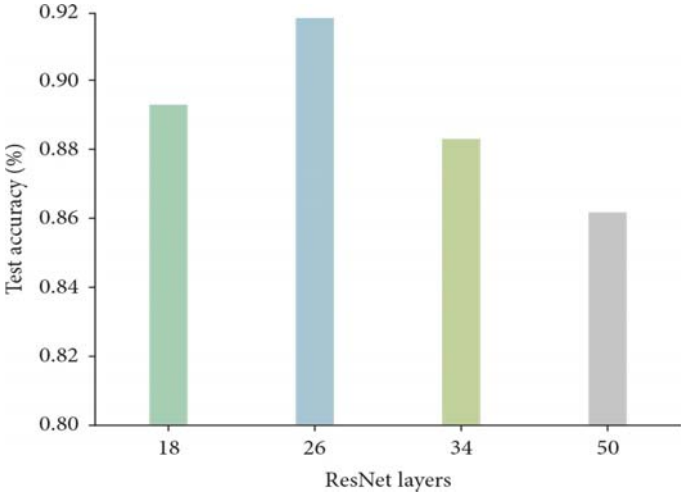


Figure 5. Test accuracy of the ResNet18, ResNet34, ResNet50 [23], and ResNet26 model. The proposed ResNet26 outperforms the best reference ResNet by 2.51%.

Results Analysis

To find the best deep residual network, a series of experiments have been conducted on BJFU100 dataset. Figure 5 shows the comparison of test accuracy among the proposed ResNet26 model and the original ResNet model of 18, 34, and 50 layers [23] designed for ImageNet. The ResNet18, ResNet34, and ResNet50 yield a test accuracy of 89.27%, 88.28%, and 86.15%, respectively. The proposed ResNet26 results in 91.78% accuracy which increases the overall efficiency up to 2.51%.

The ResNet26 is the best tradeoff between model capacity and optimization difficulty. For the size of BJFU100, ResNet26 contains enough trainable parameter to learn the discriminative feathers, which prevents underfitting.

Compared to larger model, ResNet26 results in fast and robust convergence during SGD optimization, which prevents overfitting or falls into local optimum.

RESNET26 ON FLAVIA DATASET

To show the effectiveness of the proposed ResNet26 model, a series of experiments have been performed on the publicly available Flavia [29] leaf dataset. It comprises 1907 images of 1600 × 1200 pixels, with 32 categories. Some of the samples are shown in Figure 6. We randomly select 80% of the dataset for training and 20% for testing.

All the images are doubled and resized to 224 × 224 pixels. Per-pixel value is divided by the maximum value and subtracted the mean values of the data.

The training algorithm is exactly the same as that applied to the BJFU100 dataset. Figure 7 shows the training process of ResNet26 model. Test accuracy improves quickly since the first epochs and stabilizes after 30 epochs.

The test accuracy of each model is estimated by 10- fold cross-validation, as visualized in Figure 8. The ResNet18, ResNet34, and ResNet50 achieve a test accuracy of 99.44%, 98.95%, and 98.60%, respectively. The proposed ResNet26 gains 99.65% accuracy which increases the overall efficiency up to 0.21%. Table 1 summarizes our result and other previously published results on Flavia [29] leaf dataset. The ResNet26 model achieves a 0.28% improvement compared with the best-performing method.

Figure 6. Example images of the Flavia dataset.

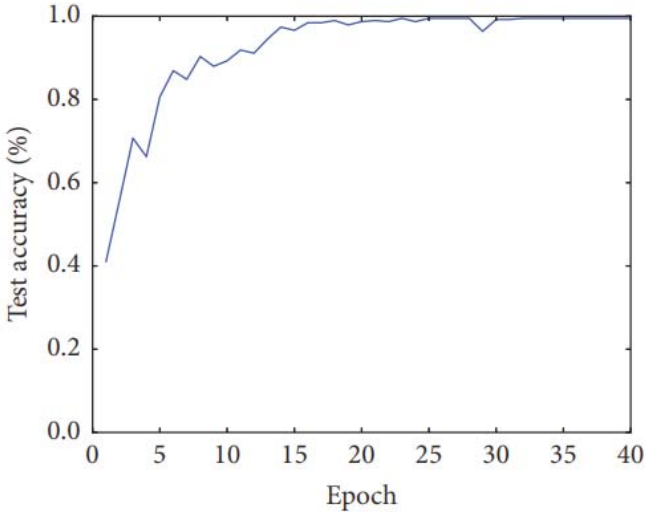


Figure 7. Evolution of classification accuracy in the test set.

CONCLUSION

The first mobile device acquired BJFU100 dataset containing 10,000 images of 100 plant species which provides data pillar stone for further plant identification study. We continue to expand the BJFU100 dataset by wider coverage of species and seasons. The dataset is open for academic community, which is available at http://pan.baidu.com/s/1jILsypS.This work also studied a deep learning approach to automatically discover the representations needed for classification, allowing use of a unified end-to-end pipeline for recognizing plants in natural environment. The proposed model ResNet26 results in 91.78% accuracy in test set, demonstrating that deep learning is the promising technology for large-scale plant classification in natural environment.

In future work, the BJFU100 database will be expanded by more plant species at different phases of life cycle and more detailed annotations. The deep learning model will be extended from classification task to yield prediction, insect detection, disease segmentation, and so on.

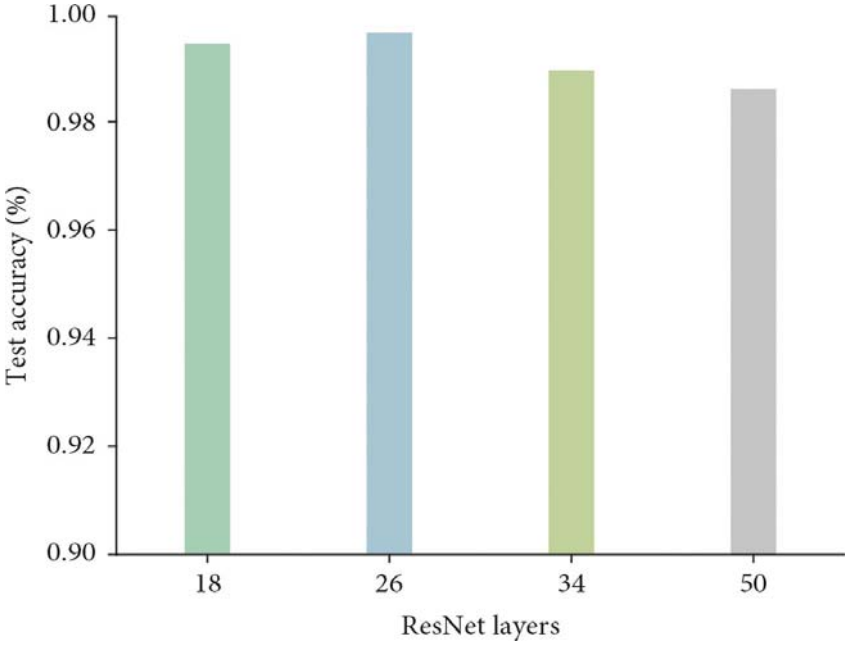


Figure 8. Test accuracy of the ResNet18, ResNet34, ResNet50 [23], and ResNet26 model on Flavia dataset. The proposed ResNet26 outperforms the best reference ResNet by 0.21%.

Table 1. Recognition rate comparison on Flavia dataset

Method	Recognition rate
PBPNN [21]	93.82%
SVM [22]	96.00%
DBNs (with "dropout") [9]	99.37%
Our work	99.65%

ACKNOWLEDGMENTS

This work was supported by the Fundamental Research Funds for the Central Universities: YX2014-17 and TD2014- 01.

REFERENCES

1. A. Joly, H. Goeau, P. Bonnet et al., “Interactive plant identifica- ¨ tion based on social image data,” Ecological Informatics, vol. 23, pp. 22–34, 2014.
2. H. Goeau, P. Bonnet, and A. Joly, “LifeCLEF plant identification ¨ task 2015,” in Proceedings of the Conference and Labs of the Evaluation Forum (CLEF ’15), 2015.
3. H. Goeau, P. Bonnet, and A. Joly, “Plant identification in an ¨ open-world (lifeclef 2016),” in Proceedings of the CLEF working notes, vol. 2016, 2016.
4. O. Soderkvist, ¨ Computer Vision Classification of Leaves from Swedish Trees, 2001.
5. H. Fu, Z. Chi, J. Chang, and C. Fu, “Extraction of leaf vein features based on artificial neural network—Studies on the living plant identification I,” Chinese Bulletin of Botany, vol. 21, pp. 429–436, 2003.
6. Y. Li, Q. Zhu, Y. Cao, and C. Wang, “A leaf vein extraction method based on snakes technique,” in Proceedings of the International Conference on Neural Networks and Brain (ICNN&B ’05), pp. 885–888, 2005.
7. P. He and L. Huang, “Feature extraction and recognition of plant leaf,” Journal of Agricultural Mechanization Research, vol. 6, p. 52, 2008.
8. G. Cerutti, L. Tougne, J. Mille, A. Vacavant, and D. Coquin, “Understanding leaves in natural images - a model-based approach for tree species identification,” Computer Vision and Image Understanding, vol. 117, no. 10, pp. 1482–1501, 2013.
9. N. Liu and J.-M. Kan, “Plant leaf identification based on the multi-feature fusion and deep belief networks method,” Journal of Beijing Forestry University, vol. 38, no. 3, pp. 110–119, 2016.
10. M.-E. Nilsback and A. Zisserman, “Delving deeper into the whorl of flower segmentation,” Image and Vision Computing, vol. 28, no. 6, pp. 1049–1062, 2010.
11. C. Zhang, J. Liu, C. Liang, Q. Huang, and Q. Tian, “Image classification using Harr-like transformation of local features with coding residuals,” Signal Processing, vol. 93, no. 8, pp. 2111– 2118, 2013.
12. Y. J. Wang, Y. W. Zhang, D. L. Wang, X. Yin, and W. J. Zeng, “Recognition algorithm of edible rose image based on neural network,” Journal of China Agricultural University, vol. 19, no. 4, pp. 180–186, 2014.

13. X. Li, L. Li, Z. Gao, J. Zhou, and S. Min, "Image recognition of camellia fruit based on preference for aiNET multi-features integration," Transactions of the Chinese Society of Agricultural Engineering, vol. 28, no. 14, pp. 133–137, 2012.
14. N. Kumar, P. N. Belhumeur, A. Biswas et al., "Leafsnap: a computer vision system for automatic plant species identification," in Proceedings of the Computer Vision—ECCV 2012, pp. 502– 516, 2012.
15. https://www.microsoft.com/en-us/research/project/flowerreco/.
16. Y. Bengio, A. Courville, and P. Vincent, "Representation learning: a review and new perspectives," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 8, pp. 1798–1828, 2013.
17. Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, 2015.
18. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in neural information processing systems, pp. 1097–1105, 2012. http://www.image-net.org/challenges/LSVRC/2012/.
19. B. Huval, T. Wang, S. Tandon et al., "An empirical evaluation of deep learning on highway driving," https://arxiv.org/abs/ 1504.01716.
20. A. Kulkarni, H. Rai, K. Jahagirdar, and P. Upparamani, "A leaf recognition technique for plant classification using RBPNN and Zernike moments," International Journal of Advanced Research in Computer and Communication Engineering, vol. 2, pp. 984– 988, 2013.
21. C. Sari, C. B. Akgul, and B. Sankur, "Combination of gross shape ¨ features, fourier descriptors and multiscale distance matrix for leaf recognition," in Proceedings of the 55th International Symposium (ELMAR '13), pp. 23–26, September 2013.
22. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '16), pp. 770– 778, Las Vegas, Nev, USA, June 2016.
23. K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in Proceedings of the European Conference on Computer Vision, pp. 630–645, 2016.
24. J. Dai, K. He, and J. Sun, "Instance-aware semantic segmentation via multi-task network cascades," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '16), pp. 3150–3158, Las Vegas, Nev, USA, June 2016.

25. V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in Proceedings of the 27th International Conference on Machine Learning (ICML ’10), pp. 807–814, June 2010.
26. S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” https:// arxiv.org/abs/1502.03167.
27. https://keras.io/.
28. S. G. Wu, F. S. Bao, E. Y. Xu, Y.-X. Wang, Y.-F. Chang, and Q.- L. Xiang, “A leaf recognition algorithm for plant classification using probabilistic neural network,” in 2007 IEEE International Symposium on Signal Processing and Information Technology, pp. 11–16, Giza, Egypt, December 2007.

CHAPTER 9

APPLYING DEEP LEARNING MODELS TO MOUSE BEHAVIOR RECOGNITION

Ngoc Giang Nguyen[1], Dau Phan[1], Favorisen Rosyking Lumbanraja[1], Mohammad Reza Faisal[1], Bahriddin Abapihi[1], Bedy Purnama[1], Mera Kartika Delimayanti[1], Kunti Robiatul Mahmudah[1], Mamoru Kubo[2], and Kenji Satou[2]

[1]Graduate School of Natural Science and Technology, Kanazawa University, Kanazawa, Japan;

[2]Institute of Science and Engineering, Kanazawa University, Kanazawa, Japan

ABSTRACT

In many animal-related studies, a high-performance animal behavior recognition system can help researchers reduce or get rid of the limitation of human assessments and make the experiments easier to reproduce. Recently, although deep learning models are holding state-of-the-art performances

Citation: Nguyen, N., Phan, D., Lumbanraja, F., Faisal, M. , Abapihi, B., Purnama, B., Delimayanti, M., Mahmudah, K., Kubo, M. and Satou, K., (2019), Applying Deep Learning Models to Mouse Behavior Recognition. Journal of Biomedical Science and Engineering, 12, 183-196. doi: 10.4236/jbise.2019.122012.

in human action recognition tasks, these models are not well-studied in applying to animal behavior recognition tasks. One reason is the lack of extensive datasets which are required to train these deep models for good performances. In this research, we investigated two current state-of-the-art deep learning models in human action recognition tasks, the I3D model and the R(2 + 1)D model, in solving a mouse behavior recognition task. We compared their performances with other models from previous researches and the results showed that the deep learning models that pre-trained using human action datasets then fine-tuned using the mouse behavior dataset can outperform other models from previous researches. It also shows promises of applying these deep learning models to other animal behavior recognition tasks without any significant modification in the models' architecture, all we need to do is collecting proper datasets for the tasks and fine-tuning the pre-trained models using the collected data.

INTRODUCTION

Researchers widely use many animals from fruit flies, mice to primates for studying biology, psychology or for developing new therapies or medicines. In many researches, observing the behaviors of the animals is a crucial step to get the data which is needed for answering research questions. Since watching and annotating the behaviors of these animals in hours of video clips are hard works, it's necessary to have a reliable and automated behavior recognition system to delegate these works to computers. With a well-performed system, we could not only solve the problem of the limitation of human assessments but also make the experiments easier to reproduce.

Many studies reported works in creating such systems for animal behavior recognition tasks. In the paper of Jhuang H. et al. [1], they created a system to automatically analyze behaviors of mice in home-cages. The system contains two modules: a feature extraction module and a classification module. In the feature extraction module, for each frame, they calculated the mouse's position and velocity based features and combined them with motion features which are extracted from the adjacent frames using an algorithm in [2]. These features then fed into an SVMHMM (Support Vector Machine-Hidden Markov Models [3]) to assess the action in the frame. In another research [4], Jiang J. et al. also used a similar approach but with a different feature extractor and classifier. For the feature extraction module, they detected interest points using a modified version of the algorithm in [5], then they extracted contextual and visual features from these points.

And they fed these extracted features into a shallow neural network that has only one hidden layer to assess the actions in the frames. The changes in the feature extraction method and the classification algorithm slightly improved the performance of the system in comparison to the previous paper. And it showed that the design of the feature extraction module can affect the performance of the whole system. However, creating good feature extractors is not an easy task. It requires much expert knowledge and carefulness and it is not always successful. And the abilities of these created systems are highly limited to the problems they were designed to solve. For example, an automated mouse behavior recognition system may not work well in a raccoon behavior recognition task, although the two animals are sharing many similarities in their appearance.

We could solve the above problem by using deep learning models which have the ability of automated learning to extract useful features from given data. Because of having this ability, deep learning models are widely used in many application fields from computer vision, speech recognition to natural language processing and often become state-of-the-art models in the field they applied to. And we can use the same models for tasks that have similarities without significant changes in the architecture of the models.

Though its high performance, it is not easy to apply deep learning models for whatever tasks we have because these models have too many parameters that it requires an extensive amount of data to train these parameters. And it is one of the reasons why deep learning models have very high performances in human action recognition tasks but not well-studied in applying to animal behavior recognition tasks.

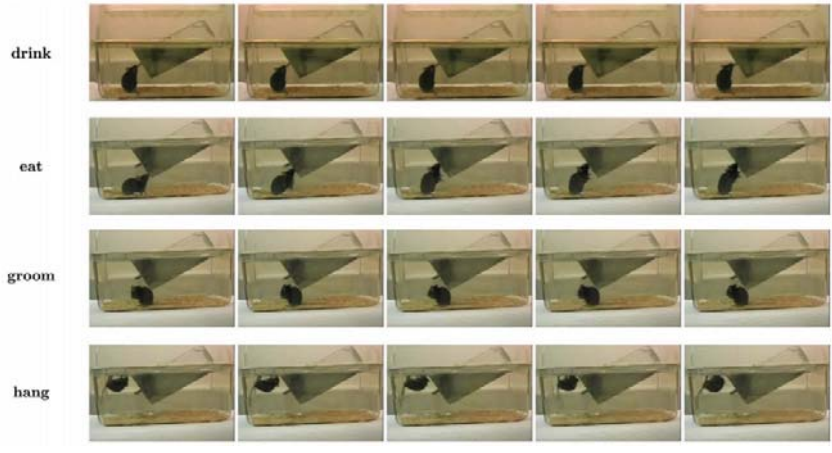
To fill in this gap, in this research, we investigated two current state-of-the-art human action recognition deep learning models in applying to a mouse behavior recognition task. The first model we investigated is the I3D model [6], which implements an inflated version of the inception module architecture [7]. The most important features of inception module are the utilization of the combined effects of filters of different sizes and pooling kernels all in one layer; and the usage of 1×1 convolutional filters which not only help to reduce the number of parameters but also introduce new combinations of features to its next layers. The second model we investigated in this research is the R(2 + 1)D model [8] which implements a 3D version of the residual module architecture [9]. This architecture allows the model to go deeper by solving the vanishing of information when training deep models.

To deal with the scarcity of training data, we did not train the models from randomly-initialized parameters but we used the parameters that were pre-trained on human action recognition tasks. By doing so, we can transfer knowledge that related to action recognition from human's tasks to the new models [10]. In the next section, we show the dataset which we used to evaluate the performances of the two deep learning models in the mouse behavior recognition task. In Section 3, we describe in detail experiments and results of the evaluating process. Finally, we give some conclusions in Section 4.

THE MOUSE BEHAVIOR DATASET

In the research of H. Jhuang, et al. [9], they introduced a task of neurobehavioral analysis of mouse phenotypes by monitoring the mouse's behaviors over long periods of time. In this experiment, each mouse is put in a transparent home cage, and there behaviors are recorded from a perpendicular angle to the side of the cages using consumer grade cameras.

In order to create a machine learning system to automatically analyze mouse's behaviors, Jhuang and his colleges have created a mouse behavior dataset by annotating the mouse's behaviors in over 10 hours of recorded videos. In their dataset, they have annotated 8 types of behavior: drinking, eating, grooming, hanging, rearing, walking, resting and micro-movements of the head. Example scenes of these behaviors are shown in Figure 1, and descriptions of these behaviors are shown in Table 1.



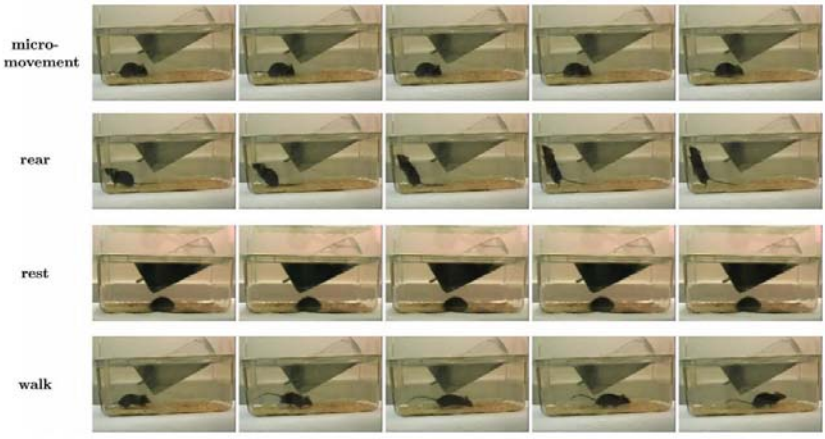


Figure 1. Example scenes of mouse's behaviors.

In the created mouse behavior dataset, among totally more than 9000 short clips, only 4200 clips that are most unambiguous were selected to create the "clipped database". It includes about 285,000 frames and corresponds to about 2.5 hours of recorded videos. In this research, in order to properly evaluate the performance of the deep learning models, we decided to use only this subset to eliminate the ambiguous in the data that even human cannot declare. The distribution of number of frames of each behavior in the "clipped database" is shown in Figure 2.

EXPERIMENTS AND RESULTS

Data Preparation

To generate optical-flow data from RGB data, we used the implementation of the TV-L1 algorithm from the research of [11] in OpenCV library. For each RGB frame, we input its previous frame and itself to the algorithm, and the algorithm outputs one optical-flow frame that has the same size as the inputs and contains two channels for horizontal and vertical movements respectively.

For data augmentation, we used the same method that used in the research of Carreira, J. and Zisserman, A. [6]. Each video frame in the dataset has a size of 320 × 240 pixels. For the I3D model, we resized them to size 255 × 255 pixels. Then we randomly cropped them to a size of 224 × 224 pixels and randomly horizontal flip them to create input frames. For R(2 + 1)D

model, we resized the images to 128 × 128 pixels then randomly cropped them to a size of 112 × 112 pixels. This data augmentation method helps us to increase the accuracy of prediction of about 3%.

Table 1. Behaviors description

No.	Behavior name	Description
1	*drink*	The mouse drinks water from the water-feed nipple.
2	*eat*	The mouse eats food from the food-feed door.
3	*groom*	The mouse grooms its coat.
4	*hang*	The mouse hangs on the top of the cage.
5	*micro-movement*	The mouse slightly moves its head around.
6	*rear*	The mouse rears on the side of the cage.
7	*rest*	The mouse stays stable or sleeps. There is no movement at all.
8	*walk*	The mouse walks or runs inside the cage.

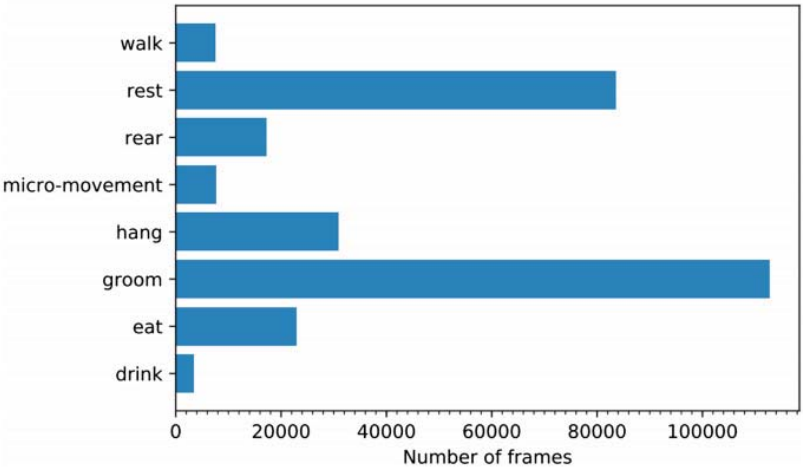


Figure 2. Distribution of number of frames of each behavior in the "clipped database".

The Models The

I3D models are derived from Inception-V1 models [7]. To benefit from the 2D architecture, all filters and pooling kernels of 2D models were inflated to 3D by endowing them with an additional temporal dimension, i.e. N × N filters become N × N × N filters. Then, the weights of 2D filters are repeated N times along the temporal dimension to bootstrap parameters from pre-trained 2D models to the 3D models. We showed the architecture of an

inflated inception module used in I3D models in Figure 3 and the detail of the architecture of the I3D model we used in this research in Figure 5.

The R(2 + 1)D models are derived from 2D versions [9] by replacing each 2D convolutional layer with two 3D convolutional layers, one for 2D image dimensions which have filters with size 1 × N × N and one for the time dimension which has filters with size M × 1 × 1. In some layers of the R(2 + 1)D models, to keep the total number of parameter to be the same as the 2D versions, the number of filter in these layers are calculated using the formula shown in Figure 4. The detail of the architecture of the R(2 + 1)D model we used in this research is shown in Figure 5.

For both models, we used 16 successive frames as an input (current frame, its 8 previous frames and its 7 next frames).

To initialize parameters of the model, for the I3D models, we used weights from model-checkpoints that were pre-trained on ImageNet data [12]; and for the R(2 + 1)D models, we used weights from model-checkpoints that were pre-trained on Sport1M [13] and Kinetics data [14].

To fine-tune the models, we used momentum optimizer from the TensorFlow framework with momentum value equal to 0.9 and a learning rate start from 1e−3 and decay to 5e−5 after several thousands of iterations. We also used dropout in fully connected layers with keep-probability of 36% to reduce the effect of overfitting when fine-tuning the models.

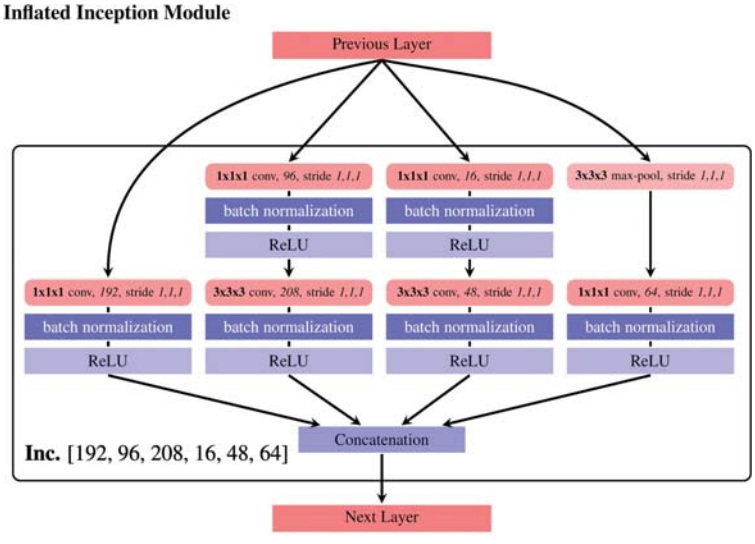


Figure 3. Architecture of an inflated inception module.

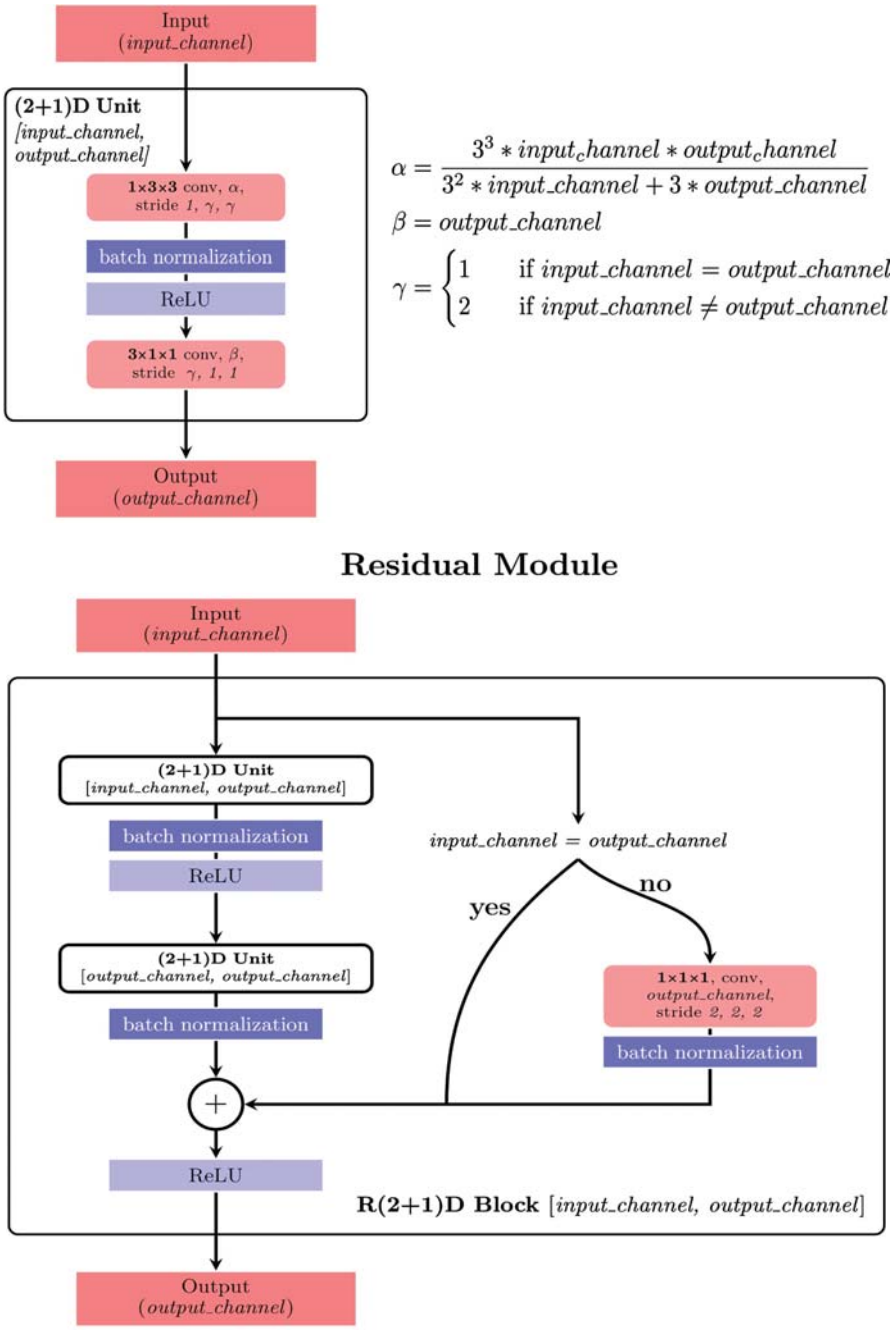
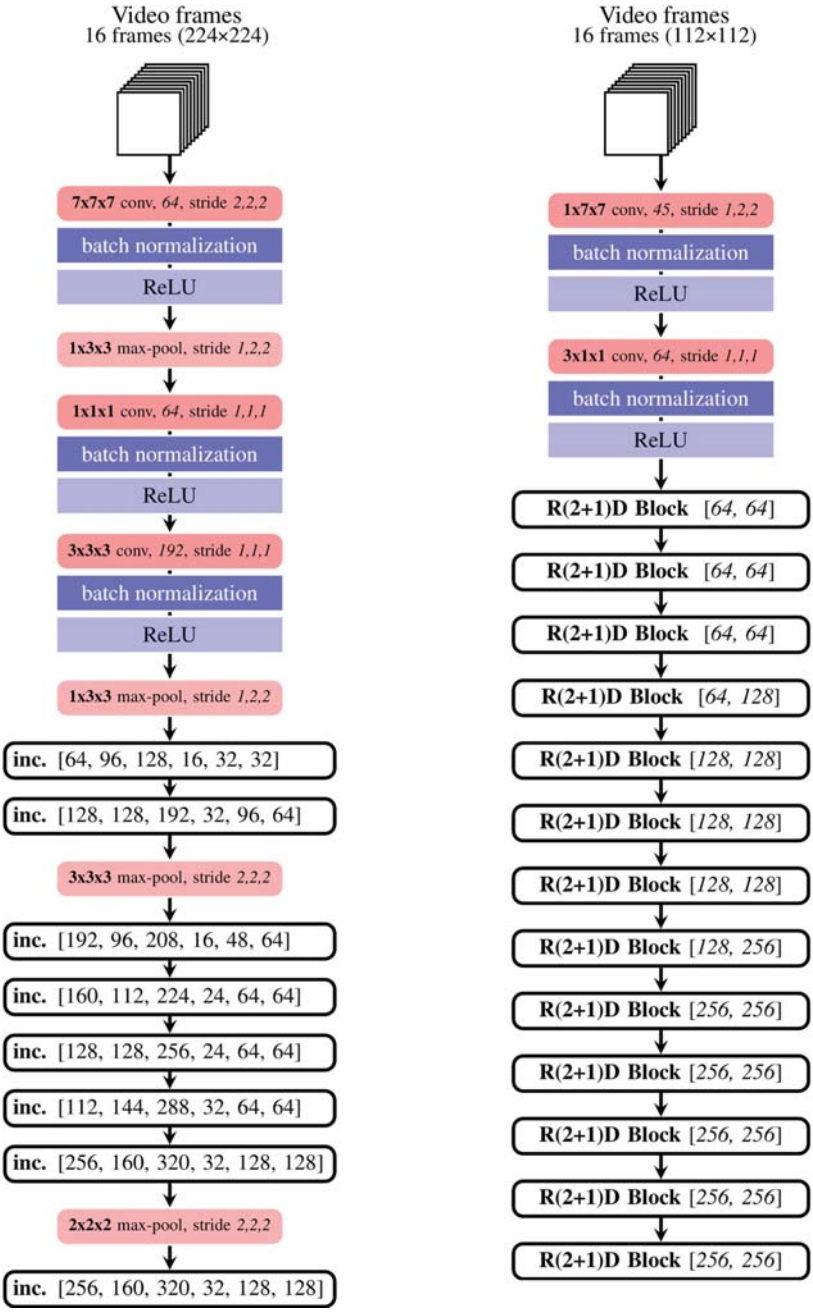


Figure 4. Architecture of a (2 + 1)D residual module.


Video frames
16 frames (224×224)
7x7x7 conv, 64, stride 2,2,2
batch normalization
ReLU
1x3x3 max-pool, stride 1,2,2
1x1x1 conv, 64, stride 1,1,1
batch normalization
ReLU
3x3x3 conv, 192, stride 1,1,1
batch normalization
ReLU
1x3x3 max-pool, stride 1,2,2
inc. [64, 96, 128, 16, 32, 32]
inc. [128, 128, 192, 32, 96, 64]
3x3x3 max-pool, stride 2,2,2
inc. [192, 96, 208, 16, 48, 64]
inc. [160, 112, 224, 24, 64, 64]
inc. [128, 128, 256, 24, 64, 64]
inc. [112, 144, 288, 32, 64, 64]
inc. [256, 160, 320, 32, 128, 128]
2x2x2 max-pool, stride 2,2,2
inc. [256, 160, 320, 32, 128, 128]
Video frames
16 frames (112×112)
1x7x7 conv, 45, stride 1,2,2
batch normalization
ReLU
3x1x1 conv, 64, stride 1,1,1
batch normalization
ReLU
R(2+1)D Block [64, 64]
R(2+1)D Block [64, 64]
R(2+1)D Block [64, 64]
R(2+1)D Block [64, 128]
R(2+1)D Block [128, 128]
R(2+1)D Block [128, 128]
R(2+1)D Block [128, 128]
R(2+1)D Block [128, 256]
R(2+1)D Block [256, 256]
R(2+1)D Block [256, 256]
R(2+1)D Block [256, 256]
R(2+1)D Block [256, 256]
R(2+1)D Block [256, 256]

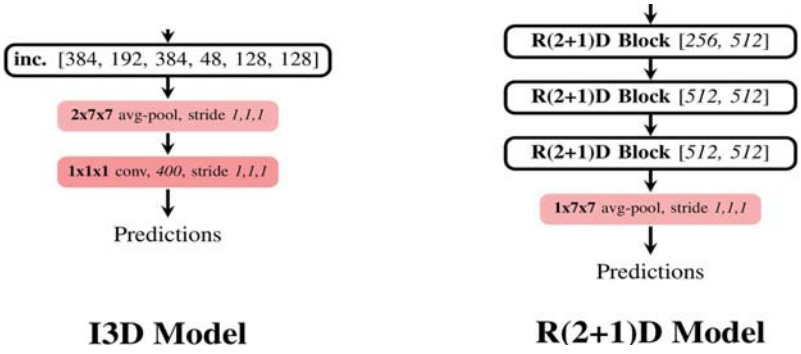


Figure 5. Architecture of the I3D model and the R(2 + 1)D model.

As discussed in the paper of Carreira, J. and Zisserman, A. [6], although I3D models can learn motion features from RGB input videos, using optical-flow as inputs can introduce some recurrent sense to the models and significantly improved the performances. In this research, we also use the same fusion method to combine output predictions of I3D models and R(2 + 1)D models. The two-stream fusion method is illustrated in Figure 6. To investigate the effects of different two-stream fusion ratios in prediction performances, we tested various fusion ratios of the two models by setting different values for rgb_weight and flow_weight in the Two-stream fusion module. For example, if only using 30% of RGB data fine-tuned model's output and 70% of optical-flow data fine-tuned model's output then rgb_ weight is set to 0.3 and flow_weight is set to 0.7.

Because frames of the dataset come from 12 different videos, we used leave-one-videos-out cross-validation to properly evaluate the performance of the models. For each video, we used all the frames extracted from it as testing data and all the frames extracted from the other videos as training data. We used the training data to fine-tune the models and used the fine-tuned models to predict labels for testing data. Then we count the total number of correct and incorrect prediction and calculate the accuracy.

Results

Figure 7 shows the results of using different fusion ratio of RGB and optical-flow data fine-tuned models on accuracies of prediction of each behavior. And Figure 8 and Figure 9 show confusion matrixes of correct and incorrect prediction ratio of behaviors in combinations of rgb_weight and flow_ weight. In Figure 7, we can see that for "drink" behaviors, combinations

with more portion of RGB fine-tuned models have better performance than combinations with more portion of optical-flow fine-tuned models for both I3D models and R(2 + 1)D models. And the performance of R(2 + 1)D models are better than the performance of the I3D models in this behaviors.

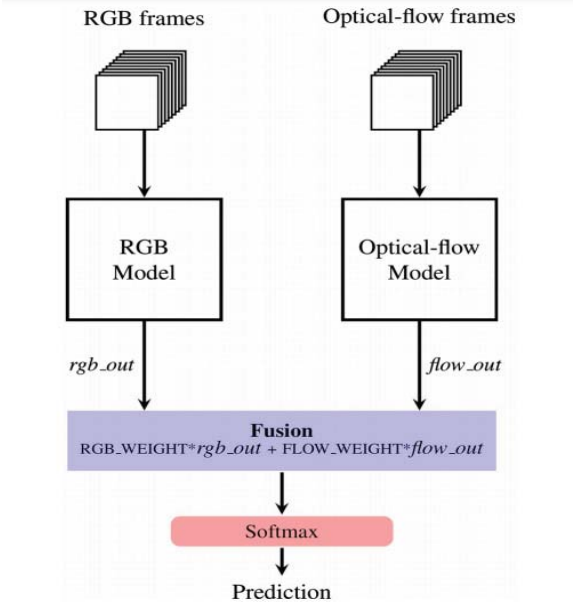


Figure 6. The two-stream fusion method.

In Figure 8 and Figure 9, from confusion matrixes of both I3D model and R(2 + 1)D model, we can see that almost false predicted samples of "drink" behaviors are misclassified as "eat" behaviors. We can explain that as the water-feed nipple and food-feed door are quite close to each other; Sometimes the two behaviors look very similar and the dataset is also imbalanced with the ratio of the number of "drink" frames to the number of "eat" frames is about 1:6.85. Therefore, the models tend to predict "drink" behaviors as "eat" behaviors in many cases. It also explains why models fine-tuned using RGB data are more precise in distinguishing the two behaviors than models fine-tuned using optical-flow data. RGB data fine-tuned models can utilize the information of the mouse's mouth contact with water-feed nipple or food-feed door. However, this information is lost in optical-flow data because there is no motion of water-feed nipple or food-feed door in the scenes.

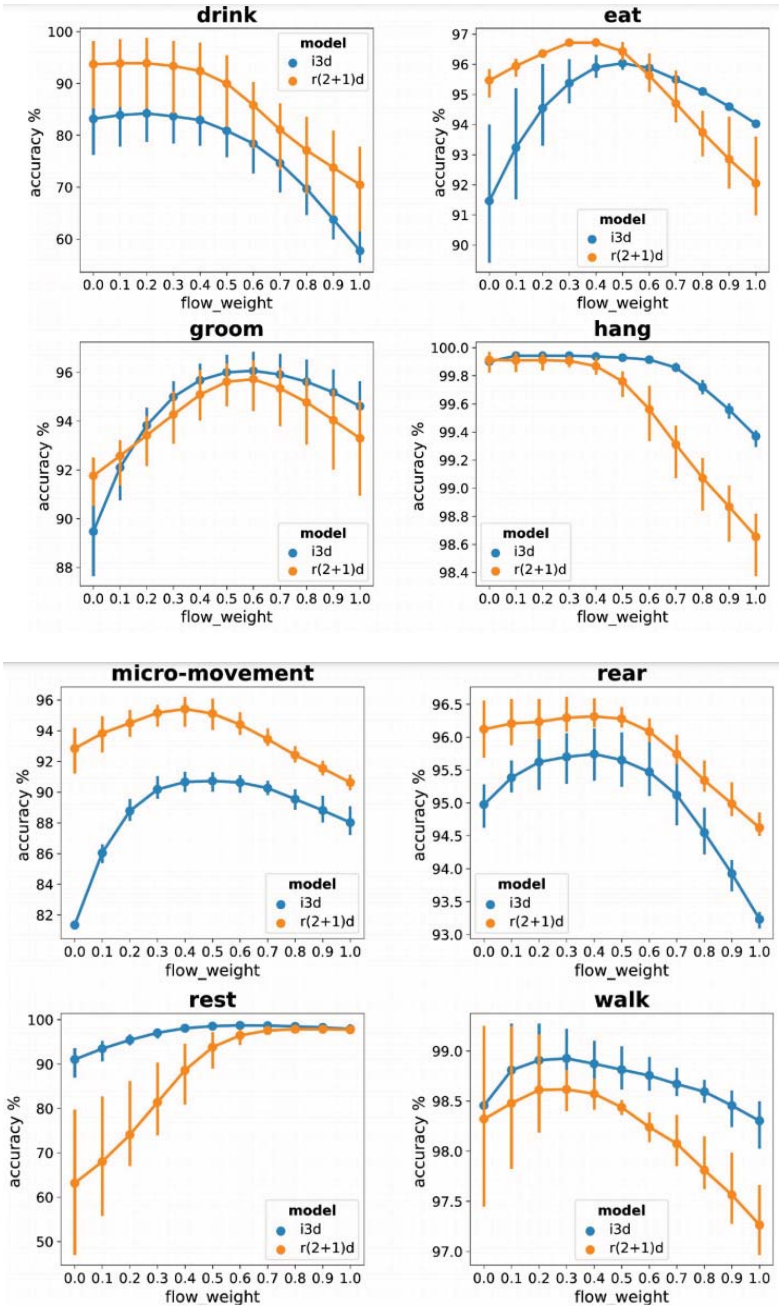


Figure 7. Accuracies of the prediction of each behavior with different two-stream fusion ratios.

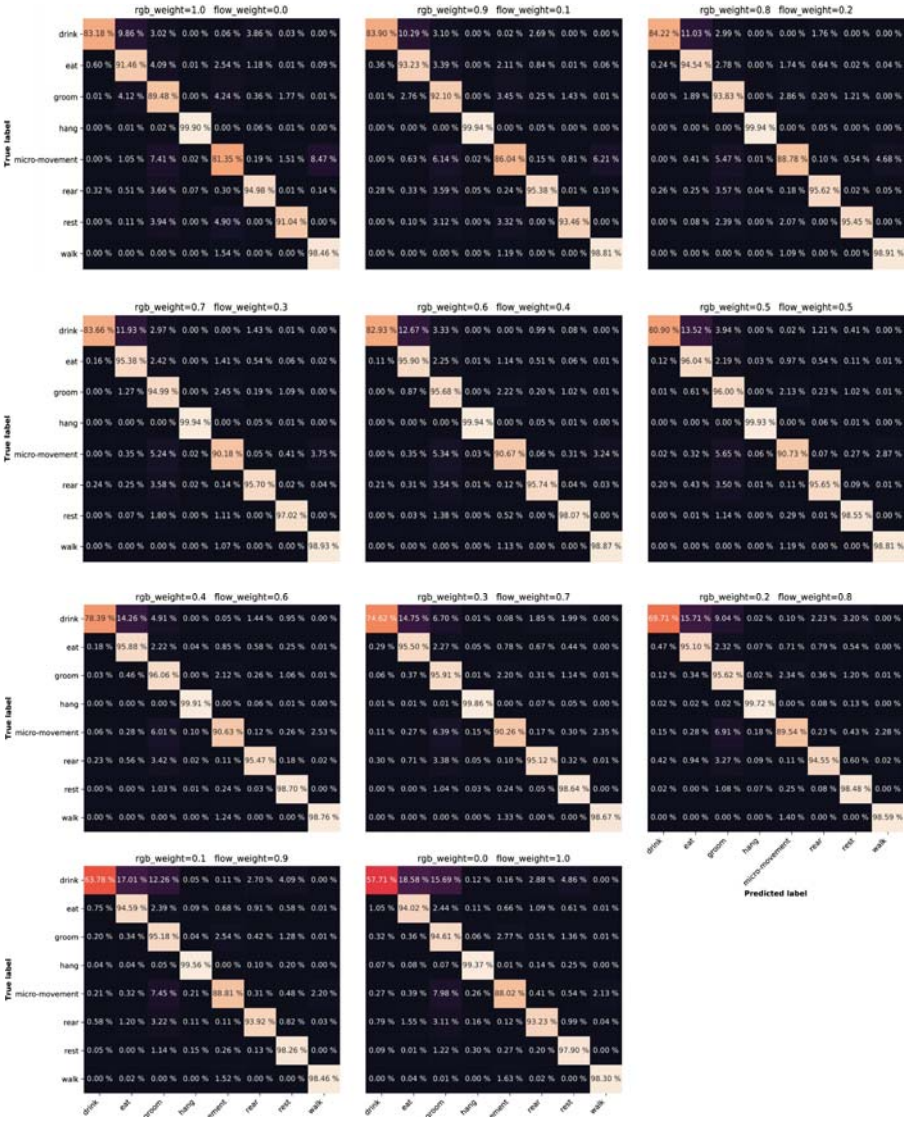


Figure 8. Confusion matrices of predictions of I3D models with different two-stream fusion ratios.

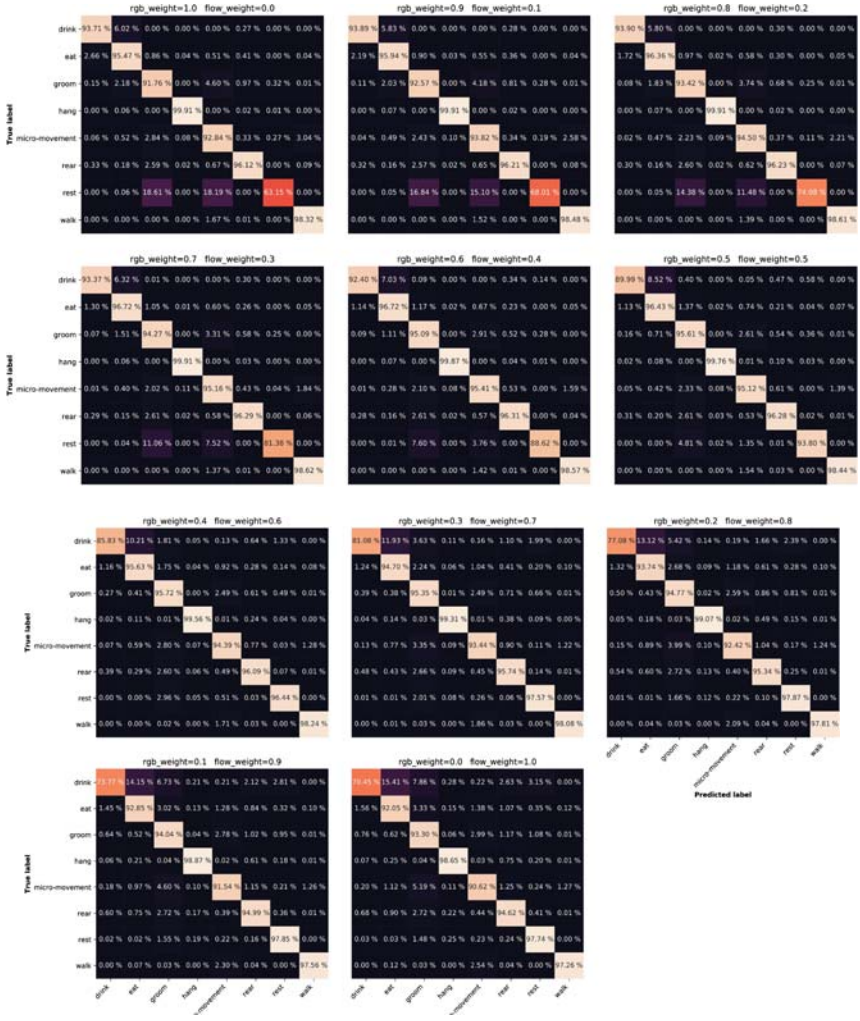


Figure 9. Confusion matrices of predictions of R(2 + 1)D models with different two-stream fusion ratios.

For "eat", "groom", "micro-movement", "rear", and "walk" behaviors, we can see that using a right combination of RGB data fine-tuned models and optical-flow data fine-tuned models give a better performance than using these models only. The R(2 + 1)D model outperforms I3D model in classifying "micro-movement" and "rear" behaviors but the I3D model is better in classifying "walk" behaviors.

The two models work very well on classifying "hang" behaviors and their performances just slightly reduce when we use a high portion of optical-

flow data fine-tuned models because of the lack of the mouse surrounding environment in the optical-flow data.

And for the "rest" behaviors, it is easy to understand why using a high portion of optical-flow data fine-tuned models give better performance as "rest" behaviors are different from other behaviors that they have no movement in the scenes.

Overall, the two deep learning models we investigated in this research outperform the previous research model in the Mouse behavior dataset as shown in Table 2. The accuracies of the two models with different fusion ratio are shown in Figure 10. Both models have best performances at fusion ratio of 40% RGB data fine-tuned models and 60% optical-flow data fine-tuned models. The I3D model achieves 96.9% of accuracy and the R(2 + 1)D achieves 96.3% of accuracy.

Table 2. Comparison of performance of models

No	Model	Accuracy (%)
1	MF + SVMHMM [1]	93.0
2	FV + NN [4]	95.9
3	I3D	**96.9**
4	R(2 + 1)D	96.3

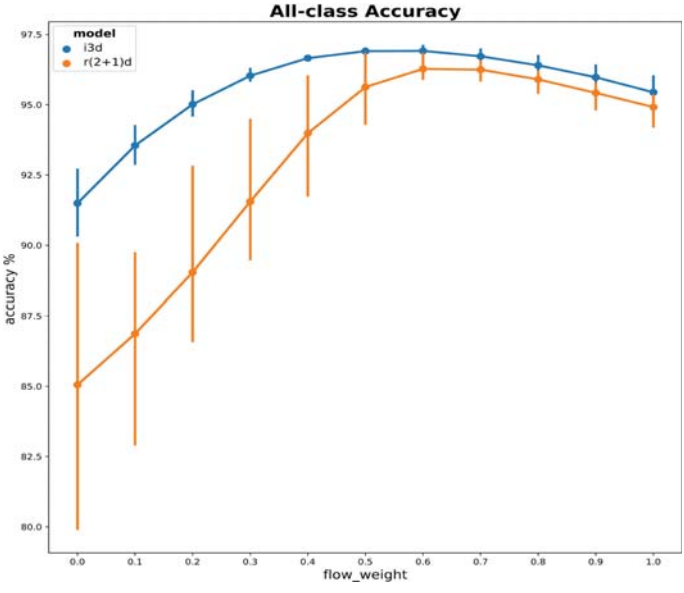


Figure 10. Accuracies of I3D models and R(2 + 1)D models with different two-stream fusion ratios.

CONCLUSIONS

We have investigated two current state-of-the-art deep learning models for human action recognition in a mouse behavior recognition task. Both models outperformed the models from previous researches. It proves that our approach of utilizing deep learning models that pre-trained on human action datasets and fine-tuning them for animal behavior recognition tasks is efficient despite the scarcity of training data. We also showed the effect of two-stream fusion ratios on the predictions. The fine-tuned models can precisely recognize most of behaviors they learned from the mouse behavior dataset. But there are some difficulties in classifying behaviors that are ambiguous or similar to other behaviors. Our proposal to solve the problem is to collect more data on difficult-to-classify behaviors. And we can redesign experimental environment such as changing the camera position or the cage configuration in order to minimize the ambiguity between behaviors. For further research, we will collect behavior data of other animals. Then we will use them to fine-tune the fine-tuned models we achieved from this research to experiment if we can really efficiently utilize deep learning models for animal behavior recognition tasks without any requirements of extensive data for training these models.

ACKNOWLEDGEMENTS

In this research, the super-computing resource was provided by Human Genome Center, the Institute of Medical Science, The University of Tokyo. Additional computation time was provided by the super computer system in Research Organization of Information and Systems (ROIS), National Institute of Genetics (NIG). This work was supported by JSPS KAKENHI Grant Number JP18K11525.

REFERENCES

1. Jhuang, H., Garrote, E., Yu, X., Khilnani, V., Poggio, T., Steele, A.D. and Sere, T. (2010) Automated Home-Cage
2. Behavioural Phenotyping of Mice. Nature communications, 1, Article Number: 68. https://doi.org/10.1038/ncomms1064
3. Jhuang, H., Serre, T., Wolf, L. and Poggio, T. (2007) A Biologically Inspired System for Action Recognition. 2007 IEEE 11th International Conference of Computer Vision, Rio de Janeiro, 14-21 October 2007, 716-725. https://doi.org/10.1109/ICCV.2007.4408988
4. Altun, Y., Tsochantaridis, I. and Hofmann, T. (2003) Hidden Markov Support Vector Machines. International Conference on Machine Learning, Washington DC, 21-24 August 2003, 3-10.
5. Jiang, Z., Crokes, D., Green, B.D., Zhang, S. and Zhou, H. (2017) Behaviour Recognition in Mouse Videos Using Contextual Features Encoded by Spatial-Temporal Stacked Fisher Vectors. International Conference on Pattern
6. Recognition Applications and Methods, 259-269. https://doi.org/10.5220/0006244602590269
7. Dollar, P., Rabaud, V., Cottrell, G. and Belongie, S. (2005) Behavior Recognition via Sparse Spatio-Temporal Feature. IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, Beijing, 15-16 October 2005, 65-72. https://doi.org/10.1109/VSPETS.2005.1570899
8. Carreira, J. and Zisserman, A. (2018) Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. 2017 IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, 21-26 July 2017, 4724-4733. https://doi.org/10.1109/CVPR.2017.502
9. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A. (2015) Going Deeper with Convolutions. 2015 IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, 7-12 June 2015, 1-9. https://doi.org/10.1109/CVPR.2015.7298594
10. Tran, D., Wang, H., Torrensani, L., Ray, J., LeCun, Y. and Paluri, M. (2018) A Closer Look at Spatiotemporal Convolutions for Action Recognition. Computer Vision and Pattern Recognition.
11. https://arxiv.org/abs/1711.11248

12. 9. He, K., Zhang, X., Ren, S. and Sun, J. (2015) Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, 27-30 June 2016, 770-778. https://doi.org/10.1109/CVPR.2016.90
13. 10. Torrey, L. and Shavlik, J. (2009) Transfer Learning. In: Soria, E., Martin, J., Magdalena, R., Martinez, M. and Serrano, A., Eds., Handbook of Research on Machine Learning Applications, IGI Global, 242-264.
14. 11. Zach, C., Pock, T. and Bischof, H. (2007) A Duality Based Approach for Realtime TV-L1 Optical Flow. Proceeding of 29th DAGM Symposium of Pattern Recognition, 4713, 214-223. https://doi.org/10.1007/978-3-540-74936-3_22
15. 12. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C. and Fei, L.F. (2015) ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision, 115, 211-252. https://doi.org/10.1007/s11263-015-0816-y
16. 13. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R. and Fei, L.F. (2014) Large-Scale Video Classification with Convolutional Neural Networks. 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, 23-28 June 2014, 1725-1732. https://doi.org/10.1109/CVPR.2014.223
17. 14. Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., Suleyman, M. and Zisserman, A. (2017) The Kinetics Human Action Video Dataset. Computer Vision and Pattern Recognition. https://arxiv.org/abs/1705.06950

Section 3:
Deep learning Applications in Medicine

CHAPTER 10

APPLICATION OF DEEP LEARNING IN NEURORADIOLOGY: BRAIN HEMORRHAGE CLASSIFICATION USING TRANSFER LEARNING

Awwal Muhammad Dawud , Kamil Yurtkan , and Huseyin Oztoprak

Department of Computer Engineering, Cyprus International University, Nicosia, Cyprus

ABSTRACT

In this paper, we address the problem of identifying brain haemorrhage which is considered as a tedious task for radiologists, especially in the early stages of the haemorrhage. The problem is solved using a deep learning approach where a convolutional neural network (CNN), the well-known AlexNet

Citation: Awwal Muhammad Dawud, Kamil Yurtkan, Huseyin Oztoprak, "Application of Deep Learning in Neuroradiology: Brain Hemorrhage Classification Using Transfer Learning", Computational Intelligence and Neuroscience, vol. 2019, Article ID 4629859, 12 pages, 2019. https://doi.org/10.1155/2019/4629859.

neural network, and also a modified novel version of AlexNet with support vector machine (AlexNet-SVM) classifier are trained to classify the brain computer tomography (CT) images into haemorrhage or nonhaemorrhage images. The aim of employing the deep learning model is to address the primary question in medical image analysis and classification: can a sufficient fine-tuning of a pretrained model (transfer learning) eliminate the need of building a CNN from scratch? Moreover, this study also aims to investigate the advantages of using SVM as a classifier instead of a three-layer neural network. We apply the same classification task to three deep networks; one is created from scratch, another is a pretrained model that was fine-tuned to the brain CT haemorrhage classification task, and our modified novel AlexNet model which uses the SVM classifier. The three networks were trained using the same number of brain CT images available. The experiments show that the transfer of knowledge from natural images to medical images classification is possible. In addition, our results proved that the proposed modified pretrained model "AlexNet-SVM" can outperform a convolutional neural network created from scratch and the original AlexNet in identifying the brain haemorrhage.

INTRODUCTION

Intracranial haemorrhage (ICH) reveals as a bleeding within the intracranial vault [1]. Weak blood vessels, hypertension, trauma, and drug abuse are generally what trigger such a medical condition. ICH is a neurologic emergency in which it can have several subtypes such as basal ganglia, caudate nucleus, or pons. The types of haemorrhage are generally dependent on the anatomic location of bleeding [2]. According to the American Heart Association and American Stroke Association, the early and timely diagnosis of ICH is significant as this condition can commonly deteriorate the affected patients within the first few hours after occurrence [3]. Noncontrast head computer tomography (CT) is the imaging modality used to detect haemorrhage due its wide availability and speed. This modality has shown a high sensitivity and specificity in detecting acute haemorrhage [2].

Recently, deep learning has risen rapidly and effectively. Deep learning-based networks have shown a great generalization capability when applied to solve challenging medical problems such as medical image classification [4, 5], medical image analysis [6], medical organs detection [7], and disease detection [8]. Convolutional neural networks were the most effective networks among deep networks, for they own the paradigms of

more biologically inspired structures than other traditional networks [9]. Eventually, various convolutional neural networks were developed such as AlexNet [10], VGG-NET [11], and ResNet [12]; these deep networks are all extensively trained on a large database named ImageNet, Large-Scale Visual Recognition Challenge [13], and they were considered as the state of the art in image classification [11–13]. These networks are considered as machine learning methods that can learn features hierarchically from lower level to higher level by building a deep architecture of the input data.

The rise in deep convolutional neural networks performance, due to their abstractions of different levels of features, motivated many researchers to transfer the knowledge acquired by these networks, when trained on millions of images into new tasks such as medical image classification, to benefit from their learned parameters, in particular, weights.

These convolutional neural networks models use fully connected layers, which represent a feedforward neural network trained using the conventional backpropagation algorithm. This means that these models may have the same drawbacks of the conventional simple neural network.

An effective neural network model is the one that performs well during both training and testing datasets; a good balance between variance error and bias error must be struck [14]. For simple models, a high bias and a low variance situation reveals when training these models; that is called underfitting. For more complex neural network models, the progress of training may let the model enter a region of low variance and bias; this can be considered as a good fit. However, as the training progresses further (more complex models), the model may go through a high variance and low bias, that is called overfitting. This is considered a major problem in training a complex neural network model.

There are many approaches for alleviating this problem [15]. These approaches include early stopping, weights penalization, weights pretraining, and dropout of hidden neurons. However, in our study, we ought to avoid these problems by replacing the SoftMax neural network with a multiclass SVM that acts as a classifier for both pretrained employed models. There have been many conducted studies [16–18] that attempt to find an alternative to SoftMax function for classification tasks. All these studies concluded that the support vector machine (SVM) might be the appropriate alternative as it may slightly boost the performance of neural network compared to the conventional SoftMax function.

Thus, in this paper, we aim to transfer the knowledge acquired by AlexNet into a new target task: classifying the CT brain haemorrhage into haemorrhage or nonhaemorrhage images. Moreover, a CNN is created from scratch and a modified AlexNet combined with SVM are also employed to perform the same classification task. The goal of employing one CNN created from scratch and fine-tuning a pretrained model for the same classification task is to show that transfer learning-based network can perform better when data are not much. Also, it is aimed to show that sufficient fine-tuning of a pretrained model can eliminate the need for training a deep CNN from scratch which usually takes long time and requires large number of images to learn. Note that in this research, the CNN created from scratch is denoted as CNN, the pretrained model that uses original AlexNet architecture is denoted as AlexNet, and the modified model is denoted as AlexNet-SVM.

The paper is structured as follows: Section 1 is an introduction of the work. Section 3 is a brief explanation of the convolutional neural networks basics, while Section 4 explains the transfer learning concept including AlexNet. Section 5.3 discusses the training of the two employed deep networks in which the data used for training are described. Section 6 discusses the networks performances and compares the results of both models. Finally, Section 8 is conclusion of the paper.

RELATED WORK

Convolutional neural networks have been employed to overcome big medical challenges like image segmentation [19] and control for people with disabilities [20]. Hussain et al. [19] have developed a convolutional neural network designed for the segmentation of the most common brain tumor, i.e., glioma tumor. The authors proposed a system composed of two networks, stacked together to form a new ILinear nexus architecture. This new architecture was capable of achieving the best results among all the proposed and related architectures. Another study by Abiyev and Arslan [20] showed that convolutional neural networks can also be used as supporting elements for people with disabilities. The authors proposed a human-machine interface based on two convolutional neural networks designed for disabled people with spinal cord, to control mouse by eye movements. Their work was validated and tested by a handcrafted dataset, and results showed that the network's performance outscored many other related works.

Furthermore, deep learning techniques were employed by Helwan et al. [21] to classify brain computer tomography (CT) images into haemorrhage

or healthy. The authors used autoencoders and deep convolutional neural networks to perform this task. As authors claimed, the employed models performed differently when trained and tested on 2527 images. It was found that the stacked autoencoder used in their paper consists of three hidden layers and outperformed other employed networks, where it achieved the highest classification rate and the lowest MSE. The authors concluded that the possible reason of this outperformance on the stacked autoencoder over convolutional neural network is due to the small number of data used for training, as a CNN needs large amount of training examples in order to converge.

In another study by Mahajan and Mahajan [22], brain haemorrhage was examined in more refined manner by feeding using the watershed algorithm along with artificial neural network (ANN) for CT identification of brain haemorrhage type. The authors of this work used features extraction before feeding images to the neural classifier, in which different features were extracted using grey-level co-occurrence matrix (GLCM). Features were then classified by a conventional backpropagation neural network used to identify the type of haemorrhage. They found that adequate image processing techniques such as noise removal and high segmentation methods are required for accurate identification of haemorrhage.

Furthermore, Gong et al. [23] focused on dividing brain CT images into regions, where each region could either be normal or haemorrhage. For images containing haemorrhage, the regions which did not include haemorrhage were treated as normal regions resulting in a highly imbalanced dataset. The researcher had utilized an image segmentation scheme that used ellipse fitting, background removal, and wavelet decomposition technique. The weighted precision and recall value for this approach were approximately 83.6% and 88.5%, respectively.

CONVOLUTIONAL NEURAL NETWORK

Convolutional neural network (CNN) is a well-employed network for several tasks in machine vision and medicine [24, 25]. Generally, the CNN relies on architectural features which include the receptive field, weight sharing, and pooling operation to take into account the 2D characteristic of structured data such as images [26]. The concept of weight sharing for convolution maps drastically reduces model parameters; this has the important implications that the model is less prone to overfitting as compared to fully connected models of comparable size. The pooling operation essentially reduces the spatial

dimension of input maps and allows the CNN to learn some invariance to moderate distortions in the training; this feature enhances the generalization of the CNN at test time as the model is more tolerant to moderate distortion in the test data [27]. The typical CNN is shown in Figure 1. Essentially, convolution layers, pooling layers, and the fully connected layers are shown. For example, layer 1 employs n convolution filters of size $a \times a$ to generate a bank of n convolution maps (C1) of size $i \times i$; this is followed by a pooling (subsampling) operation on the convolution maps with a window size of $b \times b$. Therefore, the pooling layer (S1) composes n feature maps of size $j \times j$, where, $j = i/b$ [25]. The convolution layer performs feature extraction on the incoming inputs via a convolution filter of specified size. The pooling operation pools features across input maps using a window of specified size; common pooling operations used in applications are the average and max pooling [28]. In average pooling, the average value of the inputs captured by the pooling window is taken, while, in max pooling, the maximum value of the inputs captured by the pooling window is taken. For learning the classifier model, features are forward-propagated through the network to the fully connected layer with an output layer of units. Then, the backpropagation learning algorithm can be employed to update the model parameters via the gradient descent update rule [29].

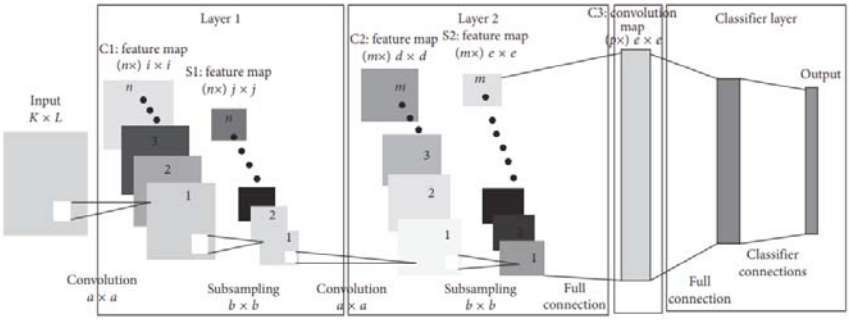


Figure 1. Convolutional neural network.

TRANSFER LEARNING

In medical image analysis and processing, a most common issue is that the number of available data for research purposes is limited and small. Hence, training a fully deep network structure like CNN with small number of data may result in overfitting, which is usually the reason of low performance

and generalization power [30]. Transfer learning is a solution to this problem where the learned parameters of effective and well-trained networks on a very large dataset are shared. The concept of transfer learning is the use of a pretrained model that is already trained on large datasets and transfers its pretrained learning parameters, in particular weights, to the targeted network model. To be able to use the network for another problem, the last fully connected layers are then trained with initial random weights on the new dataset. Although the dataset is different than the one that the network was trained on, the low-level features are similar. Thus, the parameters' transfer of the pretrained model may provide the new target model with a powerful feature extraction capability and reduce its training computations and memory cost. Transfer learning has been used extensively in medical imaging, and it showed a great efficacy in terms of accuracy, training time, and error rates [10, 31, 32]. In this paper, we present a modified pretrained model, AlexNet, that has been employed for the classification of CT brain haemorrhage images into normal and abnormal classes.

AlexNet

AlexNet is the first convolutional neural network that achieved the highest classification accuracy at the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012 [10]. This deep structure is comprised of eight main layers; the first five layers are mainly convolutions, while the last three are fully connected layers. Each convolutional layer is followed by an activation function layer, i.e., rectified linear units layer (ReLU), proposed to improve the performance of the network by making the training faster than equivalents of "tanh" activation functions [10]. After each convolution layer, a max pooling is used in AlexNet, in order to reduce the network size. Moreover, a dropout layer is added after the first two fully connected layer which helps to reduce the number of neurons and prevent overfitting [33]. Finally, a layer is added after the last layer to classify the input given data. Figure 1 shows the structure of the AlexNet.

MATERIALS AND METHODS

This work addresses the problem of the classification of the CT brain images into normal or haemorrhage, which can be a hard task for some junior radiologists and doctors. The problem is addressed by the implementation of a deep learning network trained extensively to acquire the power of extracting low to high levels of features from normal brain CT images and

others with haemorrhage medical conditions using its designed and trained filters. These features are then what distinguishes the class of the brain images, i.e., haemorrhage or not. Nonetheless, the transfer of knowledge from original to target task, which is here Haemorrhage identification, is also considered by transferring the knowledge of a pretrained model known as AlexNet, into a new classification task and testing it by the same number of images used for testing the CNN created from scratch.

In this manner, we aim to address the central issue in medical image analysis and diagnosis: training deep CNN from scratch is not needed; instead, use a pretrained modified AlexNet by adding SVM classifier to transfer its knowledge to a new target task with sufficient fine-tuning. Our conducted experiment on the CT brain haemorrhage classification using a CNN created from scratch and the pretrained models will demonstrate the truth and accuracy behind this central issue.

Data

The two employed models are trained and tested using normal and diseased brain computer tomography (CT) images collected from the Aminu Kano Teaching Hospital, Nigeria [34]. It is important to note that the abnormal images collected from this database are of different types of haemorrhage, but they were all labeled as haemorrhage, because this work aims to classify whether the CT slice contains haemorrhage or not; haemorrhage identification from set of images regardless of the haemorrhage pathology type it may have is feasible [35].

Data Augmentation

Deep networks are data-hungry systems [36], hence the more data you feed them, the more powerful and accurate they become. Therefore, in this work we decided to use data augmentation in order to multiply the number of images collected for the database, which can help in preventing the overfitting that may be encountered during training [37].

Thus, each image is first rotated left and right and then flipped 70, 160, and 270 degrees. Overall, a total number of 12635 normal and haemorrhage CT brain images are obtained. Note that 70% of the data are used for training the employed networks while 30% are used for testing, i.e., 8855 and 3790 images, respectively. Table 1 shows the learning scheme that is used in this work.

Table 1. Learning scheme of the networks.

Total number of images	
Train	8855
Test	3790
Total	12635

Figure 2 shows some normal and haemorrhage CT slices of the brain that are the used for training and testing the deep networks.

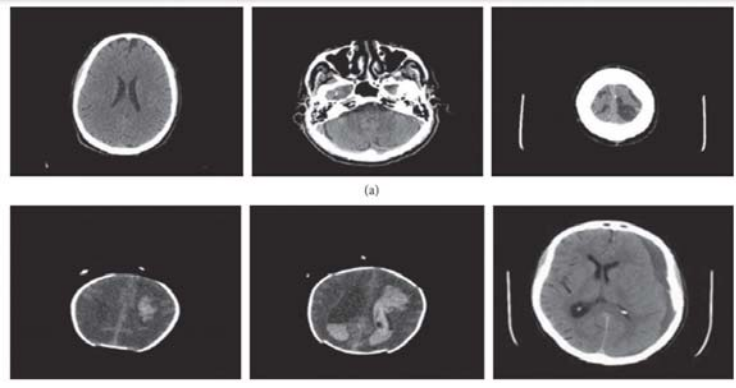


Figure 2. Sample of the databases training and validating images. (a) Haemorrhage images; (b) normal images.

The images of this database are originally of size $1024 * 1024 * 1$ pixels; hence, they were first downsampled to $227 * 227 * 1$ pixels to fit the input layer of the pretrained model: AlexNet which does not accept other input data sizes. Note that we decided to use the same input images size for the CNN created from scratch, only for networks performance comparison purposes, although any size could be used. Moreover, the images of the database are of grayscale type, and since the AlexNet model requires 3-channels input data, images were all converted to RGB by concatenating their grayscale channel for three times to become $227 * 227 * 3$.

Training the Network Models

The two employed deep models are simulated using MATLAB environment. The networks were trained on a Windows 64-bit desktop computer with an Intel Core i7 4770 central processing unit (CPU) and 16 GB random access memory. It is important to mention that there was no graphical processing unit (GPU) available in the used desktop.

The performance evaluation of the networks was carried out using a held-out test set 30% of the data. The calculation of the loss and accuracy was achieved as follows:

$$\text{Loss} = -\left(\frac{1}{n}\right)\sum_{i=1}^{n}\log P(C),$$
$$\text{Accuracy} = \frac{C}{N}, \tag{1}$$

where P(C) is the probability of the correctly classified images, n is the number of images, while N is the total number of images during the training and/or testing phases.

CNN Training

The model architecture and training settings for the CNN employed to perform the classification of brain haemorrhage are presented in this section. Extensive tests are performed to determine the best learning parameters that optimize the neural network.

Note that out of the retrieved 12635 brain CT images, 8855 images are used for training and 3790 images are used for validating the trained network. The CNN architecture employed for the classification of brain haemorrhage images is shown in Figure 3, where "Conv" denotes a convolution layer, "BN" denotes batch normalization, "FM" denotes feature maps, and "FC" denotes fully connected layer.

In this paper, all convolution operations are performed using convolution filters of size 3×3 with zero padding; all pooling operations are performed using max pooling windows of size 2×2; the input images to the model are of size 32×32.

For designing the proposed architecture, we take into consideration the size of available (i.e., limited) training data for constructing a learning model that is considerably regularized.

For example, we employ batch normalization and dropout training schemes which have been shown to improve model generalization [38–40]. For optimizing the proposed model, we employ minibatch optimization via gradient descent; we use a batch size of 60. In addition, we use a learning rate of 0.001 and train the model for 100 epochs. The learning curve for the trained CNN is shown in Figure 4; a validation accuracy of 90.65% is achieved.

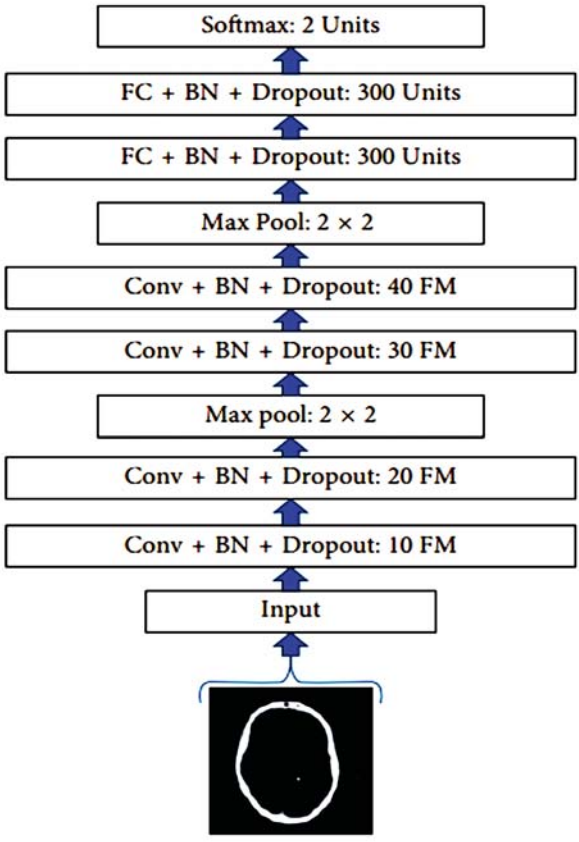


Figure 3. Proposed CNN architecture.

In addition, we observe a slight drop in validation performance when dropout and batch normalization are not employed for training the model; a validation accuracy of 87.33% is obtained. e overall proposed system for brain haemorrhage identification is tested using few CT brain haemorrhage images obtained from di erent sources available online. From the aforementioned database, we collect CT brain images of subjects with di erent haemorrhage conditions as test images. i.e., Figure 5. Experimental results show that the developed haemorrhage identification deep framework is capable of e ectively classifying the haemorrhage within the test images with an accuracy of 87.13%.

We note that in contrast to other works that train and test the proposed approach on the same dataset, the proposed pipeline in this paper has been trained and validated on one dataset and achieved promising results when

tested again on a completely di erent dataset. This shows the robustness of the deep CNN that is designed for such classification task.

AlexNet Training

AlexNet is the pretrained model selected to be used in this research because of its effective power in feature extraction. As can be seen in Figure 5, this deep convolutional neural network is comprised of 5 convolutional layers denoted as CONV1 to CONV5. These layers are followed by 3 fully connected layers denoted as FC1 to FC3, along with a Softmax activation function in the output layer (multinomial logistic regression).

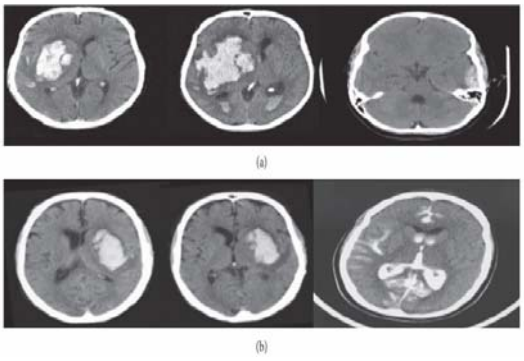
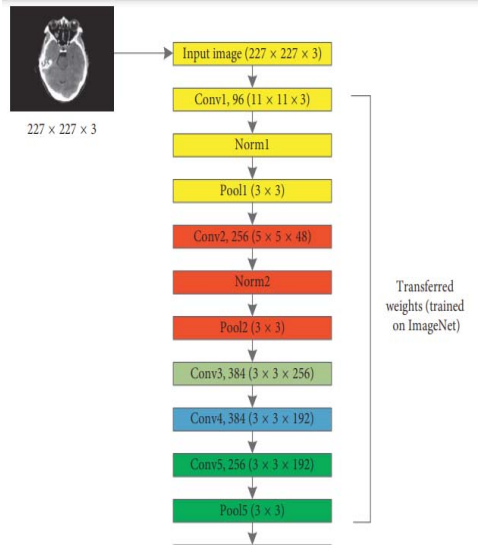


Figure 4. A sample of the brain images collected from the Internet to test the robustness of the system [41].



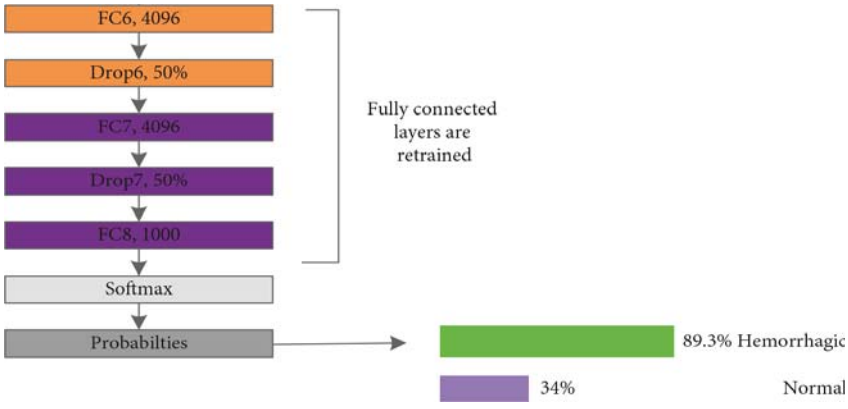


Figure 5. AlexNet proposed transfer learning network for the haemorrhage classification.

In this research, the publicly available weights of the network trained against the ILSVRC12 are used. As a pretrained model is employed (AlexNet), the final fully connected layer (FC8) was disconnected in order to add a new layer having 2 output neurons corresponding to the two CT brain images' categories. Note that the weights of this layer are initialized at random.

Contrarily, the remaining five convolutional layers are kept in the network for sharing the learned parameters, in particular, weights. These weights are already trained on large datasets, ImageNet, to extract high-level features of the input data. +us, when transferring the knowledge of AlexNet to haemorrhage classification task, these weights can act as a powerful extractor of different levels of abstractions from input data features.

The network is trained using minibatch of size 200 images of each iteration via stochastic gradient descent SGD [42]. Also, an initial learning rate is set to 0.01 to the fully connected layers (FC6, FC7, and FC8) and a reducing factor of 0.1 after 2000 iterations. Wherefore, this may fasten the learning of the network for the final fully connected layer (FC8). Table 2 shows the networks parameters during training and the result of the classification task. As seen, AlexNet has reached average training and testing accuracy of 94.12% and 92.13%, respectively.

An image from the test dataset is selected to evaluate the performance of the network in the classification pathway. Table 3 shows the mean square error (MSE) loss after each convolutional layer being trained.

Proposed AlexNet-SVM Training

Figure 6 shows the architecture of the modified version of AlexNet, in which an SVM classifier is used instead of a neural network. Similarly, this modified network, AlexNet-SVM, is also trained with the same conditions and same number of images except for the number of iterations which is here 140.

As seen in Figure 6 AlexNet-SVM's training parameters were similar to the parameters of AlexNet; however, it is noted that their performance was different. AlexNet-SVM was trained and it reached a lower MSE (0.054) compared to other networks. In addition, AlexNet-SVM achieved higher accuracies during training and testing with values of 96.34% and 93.48%, respectively.

RESULTS AND DISCUSSION

Once trained, all network models are tested on 30% of the available data. Table 4 shows the performances of each model during testing. As can be seen, the CNN, AlexNet, and AlexNet-SVM achieved different accuracies of 90.65%, 92.13%, and 93.48%, respectively. AlexNet-SVM was capable of achieving more accurate generalizing power on unseen data. However, a larger number of epochs was required to achieve such accuracy, which is relatively higher than that needed for CNN and AlexNet to achieve their highest accuracy. It is also noted that AlexNet-SVM reached a lower mean square error (MSE) (0.054) than that reached by AlexNet (0.087) and CNN (0.092); however, this also required longer training time. +e learning curves of the trained models are shown in Figures 7–9. The figures show the variations accuracy with respect to the increase of the number of epochs. Consequently, it is seen that all models are trained well, but the increase of depth of AlexNet and AlexNet-SVM makes it more difficult to train, i.e., it required longer time and more epochs to reach the minimum square error (MSE) and converge. Furthermore, it is important to mention that due to this difference in time and epoch number, the classifier of AlexNet-SVM resulted in a lower MSE and higher recognition rate than that scored by AlexNet and CNN. As a result, to understand the learning performance of networks, we have an insight into the different levels features learned by the employed models, by visualizing the learned kernels or features in the convolutional layers, shown in Figures 10 and 11.

Table 2. Models learning parameters

	CNN	AlexNet	AlexNet-SVM
Learning parameters	Values	Values	Values
Training ratio (%)	80	80	80
Initial learning rates	0.001	0.01	0.01
Number of epochs	100	200	140
Training accuracy (%)	92.89	94.12	96.34
Testing accuracy (%)	90.65	92.13	93.48
Achieved mean square error (MSE)	0.092	0.087	0.054

Table 3. Loss at each convolutional layer of CNN

Layer	CONV1	CONV2	CONV3	CONV4	CONV5
Loss	0.186	0.341	0.412	0.46	0.51

Figures 10 and 11 show the learned features of CNN and AlexNet, respectively. From Figure 6, it can be seen that neurons in the first convolution layer are the mostly active neurons in capturing good features in the training data. However, from Figure 11, it is seen that the neurons of the last convolutional layer of AlexNet are the most active neurons in capturing descriptive and different levels features. In addition, compared to CNN, this layer has an improved activity as observed in the learned features. Lastly, it can be noted that the neurons of the first and last convolutional layers of both networks have learned different and interesting representation of the input images. Generally, networks that tend to learn more descriptive and different levels features tend to perform better at run time, as the good knowledge acquired in the unsupervised pretraining contributes to better fine-tuning and classification.

Table 5 shows a comparison of the developed networks with some previous works that were proposed to classify brain haemorrhage using deep learning. Note that we ought to compare our approach with the deep networks and pretrained model researches that provide explicitly achieved accuracies and number of data. Firstly, a general analysis of the table shows that the pretrained models (transfer learning-based networks) achieved higher accuracies when compared to those that were created from scratch. The proposed AlexNet_SVM employed in this research achieved more powerful generalization capabilities than other AlexNet that use neural

network classifiers like the networks employed in this research and also in other researches [43]. Moreover, AlexNet-SVM outperformed the networks that were created from scratch such as convolutional neural networks and autoencoders [21]. Furthermore, it is seen that the employed pretrained model (AlexNet) achieved a higher recognition rate (92.13%) than other earlier research works such as CNN created from scratch on less number of images [21]. Also, this model has outperformed other types of deep networks such as autoencoder (88.3%) and stacked autoencoder (90.9%) [21].

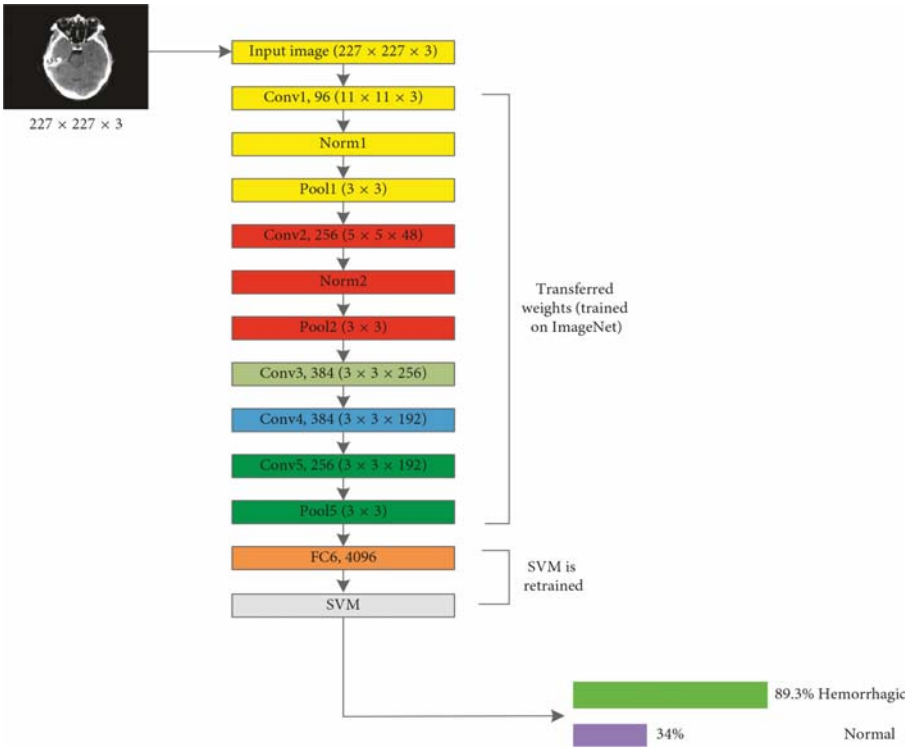


Figure 6. Modified AlexNet (AlexNet-SVM).

Table 4. Performance comparison of the employed networks

	CNN	AlexNet	AlexNet-SVM
Testing images	3790	3790	3790
Number of correctly classified images	3436	3492	3543
Accuracy (%)	90.65	92.13	93.48

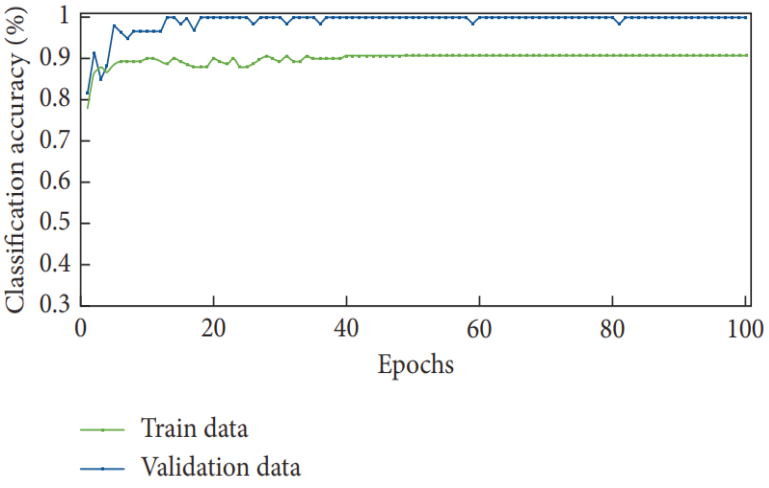


Figure 7. Learning curve for the trained CNN.

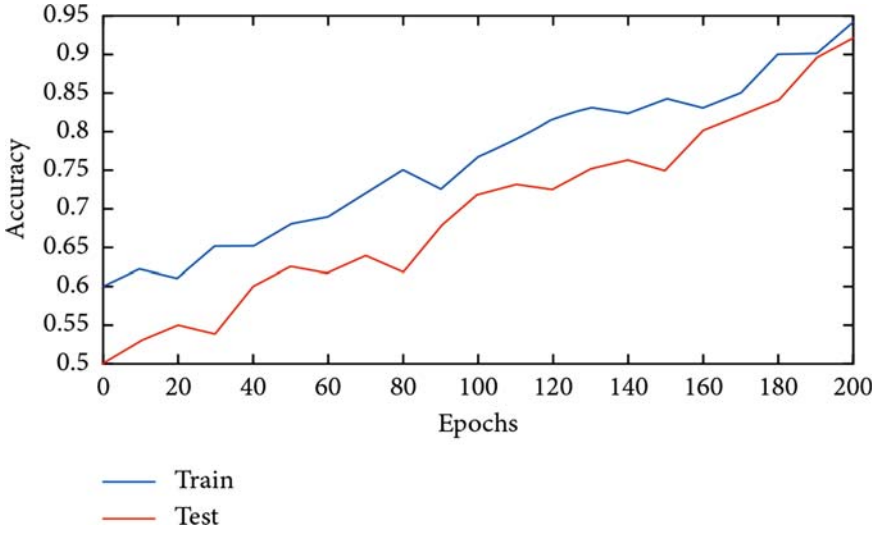


Figure 8. Learning curves of AlexNet.

This can probably be due to the deficiency of newly born networks in extracting the important features from input images which is a result of the small number of images used for training them in addition to their depth.

Overall, the application of pretrained models to solve haemorrhage classification challenge can end up with satisfying results since these deep structures have gained powerful feature extraction capabilities as they were

trained using huge databases such as ImageNet [13]. The obtained results of applying the proposed AlexNet-SVM, AlexNet and CNN in this research show that applying deep CNNs to the problem of brain haemorrhage is promising, in a way that a haemorrhage can be identified by a deep neural network with low margins of error.

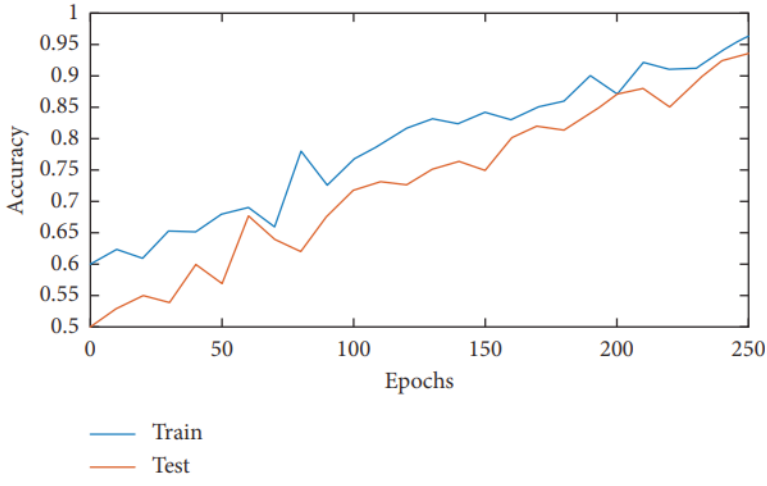


Figure 9. Learning curves of AlexNet-SVM.

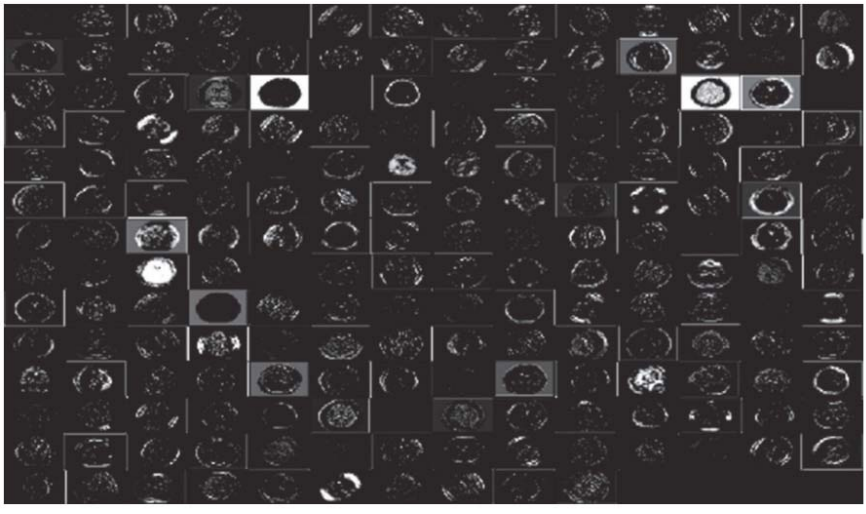


Figure 10. Learned kernels of CNN.

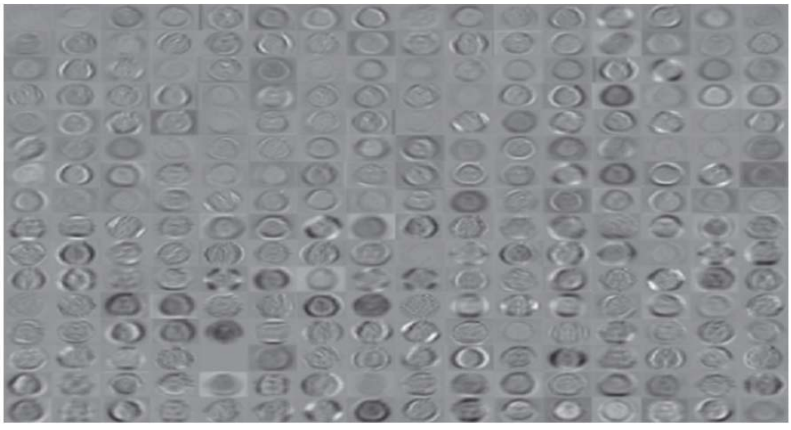


Figure 11. Learned kernels of AlexNet

Table 5. Performance metrics of the networks

Network model	CNN	AlexNet	AlexNet-SVM
Accuracy (%)	98	96	96
Sensitivity (%)	96	95	95
Specificity (%)	98	98	96
Misclassified (%)	11	9	7

Table 6. Results comparison with earlier works

Network models	CNN	AlexNet	AlexNet-SVM	AlexNet [43]	CNN [91]	AE [91]	SAE [91]
Number of images	15632	15632	15632	11,088	5251	5251	5251
Accuracy (%)	90.82	95.13	94.48	96	98.9	88.3	90.6

Performance Evaluation Metrics

These metrics are derived from classification of the tested sampling images, as shown in Table 6, being derived by a contingency table which is called confusion matrix [13]. Accuracy indicates the percentage of rightly classified image samples, without considering their class labels. For a binary classification that concludes on positive and negative classes, sensitivity is the percentage of correctly classified samples and specificity is the number of correctly negative samples classified:

$$\text{Accuracy} = \frac{(TP + TN)}{TN + TP + FP + FN},$$
$$\text{Sensitivity} = \frac{TP}{TP + FN},$$
$$\text{Specificity} = \frac{TN}{TN + FP}. \quad (2)$$

Models Comparison

In this section, the comparison of the conventional AlexNet and the proposed AlexNet-SVM is explained, in order to show the advantages of the fusion of AlexNet and SVM, in addition to the possible reasons of AlexNet-SVM outperformance. As seen in Table 5, the fusion of AlexNet and SVM resulted in a slight boost of accuracy by 0.934. +is outperformance is mainly due to the use of a different optimization criterion that the SVM uses. This algorithm is used to minimize the prediction loss on the training set of the neural network. However, in practice, there are two challenges with this risk. First is the convexity; it is not convex which means that many local minimums may exist. Second problem is the smoothness; it is not smooth, which means it may not be practically minimized. In contrast, SVM aims to minimize the generalization error by using structural risk minimization principles for the testing set. As a result of a maximized margin, the generalization ability of SVM is greater than that of the other classifiers.

LIMITATIONS

The effectiveness of deep learning in medical applications is great and improving with time; however, it still encounters some drawbacks, in particular, the availability data. +e variability of data (e.g., contrast, noise, and resolution) can be one of the main barriers of the adaptation of deep learning in medicine. +ese intelligent models can suffer from poor generalization if data contain some noise and when they are generated from different modalities. Moreover, deep learning models are data-driving systems; the more the data, the more efficient they become. +e problem is very few data are not publicly available in the medical field due to privacy issues as in most cases, the data contain sensitive information. +us, we and many other researchers prefer to use transfer learning based models which usually require less number of data to learn, as they are already trained using large amounts of data. Hence, the system is capable of learning different levels of features, which helps in adapting the new task accurately, even if the data are not large.

CONCLUSION

In this research, the detection of brain haemorrhage in CT images problem is solved using neural networks and the results sound robust and promising. One of the motivations behind this research is to address and attempt to overcome the difficulties that radiologists might encounter when diagnosing brain haemorrhage suspected images. Hence, we investigated the use of a potential deep convolutional neural network that can help the medical experts in making more accurate decisions. As a result, this may reduce the diagnosis error and boost the accuracy of haemorrhage identification made by medical experts. The paper proposes a pretrained modified network "AlexNet-SVM" for the same classification task. +e three models including the proposed model were trained on a relatively small database in order to examine the network performance. It is obvious that the application of deep learning networks in medical image analysis encounters several challenges. +e most common challenge is the lack of large training data sets which can be considered as an obstacle. +e experiments conducted in this study demonstrated that the transfer of knowledge into medical images can be possible, even though the deep networks are originally trained on natural images. +e proposed model using the SVM classifier helps in improving the performance of AlexNet. Moreover, it was manifested that small number of data can be enough for fine-tuning a pretrained model, in contrast to a CNN created from scratch which needs a large number of data to be trained. +us, the proposed model's performance is an indicator of how transfer learning-based networks can be considered in brain haemorrhage identification.

REFERENCES

1. U. Balasooriya and M. S. Perera, "Intelligent brain haemorrhage diagnosis using artificial neural networks," in Proceedings of the Business Engineering and Industrial Applications Colloquium (BEIAC), pp. 128–133, IEEE, Kuala Lumpur, Malaysia, September 2012.
2. R. Badenes and F. Bilotta, "Neurocritical care for intracranial haemorrhage: a systematic review of recent studies," British Journal of Anaesthesia, vol. 115, no. 2, pp. 68–74, 2015.
3. L. B. Morgenstern, J. C. Hemphill, C. Anderson et al., "Guidelines for the management of spontaneous intracerebral haemorrhage: a guideline for healthcare professionals from the American Heart Association/ American Stroke Association," Stroke, vol. 46, pp. 2032–2060, 2010.
4. R. H. Abiyev and M. K. S. Ma'aitah, "Deep convolutional neural networks for chest diseases detection," Journal of Healthcare Engineering, vol. 2018, Article ID 4168538, 11 pages, 2018.
5. A. Helwan and R. Abiyev, "Shape and texture features for the identification of breast cancer," in Proceedings of the World Congress on Engineering and Computer Science, vol. 2, pp. 19–21, San Francisco, USA, October 2016.
6. S. U. Akram, J. Kannala, L. Eklund, and J. Heikkila, "Cell¨ segmentation proposal network for microscopy image analysis," in Proceedings of the International Workshop on LargeScale Annotation of Biomedical Data and Expert Label Synthesis, pp. 21–29, Springer International Publishing, Athens, Greece, October 2016.
7. A. Helwan and D. Uzun Ozsahin, "Sliding window based machine learning system for the left ventricle localization in MR cardiac images," Applied Computational Intelligence and Soft Computing, vol. 2017, Article ID 3048181, 9 pages, 2017.
8. O. K. Oyedotun, E. O. Olaniyi, A. Helwan, and A. Khashman, "Hybrid auto encoder network for iris nevus diagnosis considering potential malignancy," in Proceedings of the 2015 International Conference on Advances in Biomedical Engineering (ICABME), pp. 274–277, Beirut, Lebanon, September 2015.
9. A. Mnih and G. E. Hinton, "Ascalable hierarchical distributed language model," in Proceedings of the Advances in Neural Information Processing Systems, pp. 1081–1088, Vancouver, Canada, December 2009.

10. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Proceedings of the Advances in Neural Information Processing Systems, pp. 1097–1105, Lake Tahoe, NV, USA, December 2012.
11. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, http:// arxiv.org/abs/1409.1556.
12. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778, Las Vegas, NV, USA, June 2016.
13. O. Russakovsky, J. Deng, H. Su et al., "ImageNet large Scale visual recognition challenge," International Journal of Computer Vision (IJCV), vol. 115, no. 3, pp. 211–252, 2015.
14. C. M. Bishop, Pattern Recognition and Machine Learning, Springer-Verlag, New York, USA, 2006.
15. O. K. Oyedotun, E. O. Olaniyi, and A. Khashman, "A simple and practical review of over-fitting in neural network learning," International Journal of Applied Pattern Recognition, vol. 4, no. 4, pp. 307–328, 2017.
16. Abien Fred Agarap, "A neural network architecture combining gated recurrent unit (GRU) and support vector machine (SVM) for intrusion detection in network traffic data," 2017, http://arxiv.org/abs/1709.03082.
17. A. Alalshekmubarak and L. S. Smith, "A novel approach combining recurrent neural network and support vector machines for time series classification," in Proceedings of the 2013 9th International Conference on Innovations in Information Technology (IIT), pp. 42–47, IEEE, Abu Dhabi, UAE, March 2013.
18. Y. Tang, "Deep learning using linear support vector machines," 2013, http://arxiv.org/abs/1306.0239.
19. [19] S. Hussain, S. M. Anwar, and M. Majid, "Segmentation of glioma tumors in brain using deep convolutional neural network," Neurocomputing, vol. 282, pp. 248–261, 2018.
20. R. H. Abiyev and M. Arslan, "Head mouse control system for people with disabilities," Expert Systems, 2019.
21. A. Helwan, G. El-Fakhri, H. Sasani, and D. Uzun Ozsahin, "Deep networks in identifying CT brain hemorrhage," Journal of Intelligent & Fuzzy Systems, vol. 35, no. 2, pp. 2215–2228, 2018.

22. R. Mahajan and P. M. Mahajan, "Survey on diagnosis of brain haemorrhage by using artificial neural network," International Journal of Scientific Research Engineering & Technology, vol. 5, no. 6, pp. 378–381, 2016.
23. T. Gong, R. Liu, C. L. Tan et al., "Classification of CT brain images of head trauma," in Proceedings of the IAPR International Workshop on Pattern Recognition in Bioinformatics, pp. 401–408, Springer, Melbourne, Australia, October 2007.
24. O. K. Oyedotun and A. Khashman, "Deep learning in visionbased static hand gesture recognition," Neural Computing and Applications, vol. 28, no. 12, pp. 3941–3951, 2017.
25. H.-C. Shin, H. R. Roth, M. Gao et al., "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning," IEEE transactions on medical imaging, vol. 35, no. 5, pp. 1285–1298, 2016.
26. O. K. Oyedotun and K. Dimililer, "Pattern recognition: invariance learning in convolutional auto encoder network," International Journal of Image, Graphics and Signal Processing, vol. 8, no. 3, pp. 19–27, 2016.
27. M. D. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," 2013, http:// arxiv.org/ abs/1301.3557.
28. D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?," Journal of Machine Learning Research, vol. 11, pp. 625–660, 2010.
29. O. Abdel-Hamid, A.-R. Mohamed, H. Jiang, and G. Penn, "Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition," in Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'12), pp. 4277–4280, IEEE, Kyoto, Japan, March 2012.
30. M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," in Proceedings of the 32nd International Conference on Machine Learning, pp. 97–105, Lille, France, July 2015.
31. P. M. Cheng and H. S. Malhi, "Transfer learning with convolutional neural networks for classification of abdominal ultrasound images," Journal of digital imaging, vol. 30, no. 2, pp. 234–243, 2017.

32. H. Lei, T. Han, F. Zhou et al., "A deeply supervised residual network for HEp-2 cell classification via cross-modal transfer learning," Pattern Recognition, vol. 79, pp. 290–302, 2018.
33. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Stacked denoising autoencoder and dropout together to prevent overfitting in deep neural network," Journal of Machine Learning Research, vol. 15, pp. 1929–1058, 2014.
34. Aminu Kano Teaching Hospital, Nigeria, http://akth.org.ng.
35. M. Al-Ayyoub, D. Alawad, K. Al-Darabsah, and I. Aljarrah, "Automatic detection and classification of brain hemorrhages," WSEAS Transactions on Computers, vol. 12, no. 10, pp. 395–405, 2013.
36. Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, 2015.
37. J. Weston, F. Ratle, H. Mobahi, and R. Collobert, "Deep learning via semi-supervised embedding," in Neural Networks: Tricks of the Trade, pp. 639–655, Springer, Berlin, Heidelberg, 2012.
38. S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in Proceedings of the International Conference on Machine Learning, pp. 448–456, Lille, France, June 2015. K. He and J. Sun, "Convolutional neural networks at constrained time cost," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'15), pp. 5353–5360, Boston, MA, USA, June 2015.
39. S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: a convolutional neural-network approach," IEEE Transactions on Neural Networks, vol. 8, no. 1, pp. 98–113, 1997.
40. F. Gaillard, "Intracranial hemorrhage, radiology reference article," 2018, https://radiopaedia.org/articles/intracranialhaemorrhage.
41. R. G. J. Wijnhoven and P. H. N. de With, "Fast training of object detection using stochastic gradient descent," in Proceedings of the International Conference on Pattern Recognition (ICPR), pp. 424–427, Istanbul, Turkey, August 2010.
42. V. Desai, A. E. Flanders, and P. Lakhani, "Application of deep learning in neuroradiology: automated detection of basal ganglia haemorrhage using 2D-convolutional neural networks," 2017, http://arxiv.org/abs/1710.03823.

CHAPTER 11

A REVIEW OF THE APPLICATION OF DEEP LEARNING IN BRACHYTHERAPY

Hai Hu, Yang Shao, and Shijie Hu

Applied Nuclear Technology in Geosciences Key Laboratory of Sichuan Province, Chengdu University of Technology, Chengdu, China

ABSTRACT

Objective

The automation of brachytherapy is the direction of future development. This article retrospectively studied the application of deep learning in brachytherapy of cervical cancer and clarified the status quo of development.

Method

This survey reviewed the application of machine learning and deep learning in brachytherapy for cervical cancer in the past 10 years. The survey

Citation: Hu, H., Shao, Y. and Hu, S. (2020), A Review of the Application of Deep Learning in Brachytherapy. Open Access Library Journal, 7, 1-9. doi: 10.4236/oalib.1106589.

retrieved and reviewed electronic journal articles in scientific databases such as Google Scholar and IEEE. The three sets of keywords used 1) deep learning, brachytherapy, 2) machine learning, brachytherapy, 3) automation, brachytherapy.

Results

Through research on the application of deep learning in brachytherapy, it is found that the U-net model is basically based on convolutional neural networks or some attention mechanisms are added to it, and it is applied to brachytherapy of prostate or cervical cancer. The automatic segmentation and reconstruction of the mid-source applicator (interpolation needle), target area delineation, optimization in the treatment planning system and dose calculation have achieved good results, proving that deep learning can be applied to the clinical treatment of brachytherapy.

Conclusion

The research on the application of deep learning in brachytherapy confirmed that deep learning can effectively promote the development of brachytherapy.

Keywords:- Deep Learning, Brachytherapy, Machine Learning, Automation

INTRODUCTION

Brachytherapy technology is a method of placing a radioactive source into the tumor area through an applicator or directly implanting it into the tumor tissue for radiotherapy. Because of its inherent physical and biological characteristics, brachytherapy can give a high absorbed dose to the focused irradiation of the tumor, and the dose around the source drops rapidly, which can effectively increase the local irradiation of the tumor and protect the normal tissue around the tumor. The local control rate and survival rate have significantly reduced the complications associated with brachytherapy. When the patient moves or the tumor moves in the body, the relative position of the radiation source and the tumor can remain unchanged, and the tumor obtains high dose conformity. These advantages make brachytherapy widely used in clinical applications, often used in the cervix and uterus. The treatment of tumors in the body, vagina, nasopharynx, esophagus, rectum, breast, prostate, skin and other parts is also applicable to the treatment of tumors in many other parts [1] [2] [3].

The basic procedures of brachytherapy include: target area delineation, applicator reconstruction, dose calculation and dose optimization [4]. The specific process is shown in Figure 1. This review revolves around the brachytherapy process.

In recent years, with the development of computer hardware, the rapid calculation and innovation of large amounts of data and neural network algorithms brought by social digitization, deep learning technology has developed rapidly, and deep learning has also gradually emerged in the medical field [5] mainly applied in the following aspects: organ delineation, applicator reconstruction, dose calculation and treatment planning system [6] [7] [8]. How to closely integrate emerging deep learning with traditional brachytherapy technology to promote the development of three-dimensional brachytherapy technology is the problem considered in this article, but also provides a broader perspective in this field and finds new problems. This article reviews the application of deep learning in the brachytherapy automation process [9], which is reported as follows.

ORGAN DELINEATION AND SEGMENTATION

In brachytherapy, organ delineation and segmentation undoubtedly play an important role in the treatment plan. The patient performs layer-by-layer scanning through CT or MRI to obtain multiple medical images [10]. The doctor sketches the primary target area, the medium-risk target area and the organs at risk according to the patient's condition. In clinical practice, manual sketching is still the main method. However, there are hundreds of CT images of a single patient, which is a lot of work for doctors. The automatic segmentation of the target area of medical images also plays an increasingly important role in helping doctors delineate tumor areas and endanger organs. Many segmentation methods have been developed, and the effects are different according to the type of application and the image studied. Now some of them have mature segmentation methods integrated in commercial treatment planning system [6].

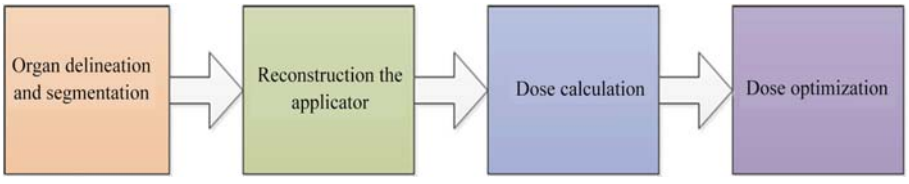


Figure 1. The basic procedures of brachytherapy.

Derek Allman [11] et al. used convolutional neural networks in 2018 to locate and classify the sources and artifacts in the K-wave simulation formation data. The experimental results show that, unlike geometry-based beamforming, the use of convolutional neural networks can be effective to eliminate metal artifacts in the image. In 2018, Xia Huang [12] et al. constructed a convolutional neural network and combined residual learning to eliminate metal artifacts on CT images. The final signal-to-noise ratio on the test set was 38.09. In 2019, Davood Karimi [13] et al. used Convolutional Neural Networks (CNN) to develop an automated accurate and stable segmentation method for the clinical target volume of transrectal ultrasound imaging of the prostate in brachytherapy, and proposed two different strategies to Improving the accuracy of image segmentation, research shows that this method can significantly improve the performance of medical image segmentation. Yang Lei [14] et al. built the supervised learning model V-net in 2019 to complete the segmentation of the prostate in the ultrasound image, and marked the prostate tissue through deep supervised learning. Finally, the segmented prostate volume is reconstructed and refined through contours. Experiments prove that the proposed technique can be used for the diagnosis and treatment of the prostate. The results show that the Days similarity coefficient, Hausdorff distance, and average surface distance of prostate segmentation are 0.92, 3.94, and 0.60, respectively. Nathan Orlando [15] et al. completed the automatic segmentation of the prostate in ultrasound-guided prostate cancer by constructing U-net and V-net networks in 2020, proving that the constructed model achieved good results in segmentation, and the evaluation parameters all indicated that the segmentation effect was better good. Qin Nannan [16] et al. completed the automatic delineation of clinical target areas and endangered organs in brachytherapy of cervical cancer by building a U-net network in 2020. The average value of the Dess similarity coefficient of the automatically delineated target area is 0.898, Hausdorff distance. The average value is within 5.3 mm, which proves that it can be used in clinic and can greatly improve the efficiency of doctors.

SEGMENTATION AND RECONSTRUCTION OF THE APPLICATOR (INTERSTITIAL NEEDLES)

The applicators serve as a bridge between the radiation source and the patient in brachytherapy, and are extremely important in brachytherapy. Due to the difference in density and patient tissue, highlight features are displayed on the CT image. In the formulation of the treatment plan, it

is especially important to re-construct the contour of the applicator. The subsequent source distribution and dose optimization are all based on the accurate reconstruction of the applicator. Deep learning and reconstruction of the applicator are also the direction of development at this stage. Nlls Gessert [17] et al. built a deep learning model based on spatial continuity in 2019 to estimate the position of the tip when the interpolation needle is inserted, to solve problems in clinical applications, and proved that this model can also be used to adjust the position of the calibration interpolation needle. The relationship coefficient is 0.9997, which is significantly better than other methods. In 2019, William T. Hrinivich [18] et al. reconstructed the reconstruction of ring-shaped applicators and oval applicators in MRI-guided cervical cancer through a model-to-image registration algorithm, proving that the accuracy and time of the algorithm fully meet the clinical needs and make treatment The automation of the plan is a step forward. Hyunuk Jung [19] [20] et al. completed the segmentation of the applicator in high-dose cervical cancer by constructing a U-net network, and then generated the trajectory of the applicator through the voxel clustering algorithm to complete the reconstruction of the applicator, and The model was evaluated using Hausdorff distance, Days similarity coefficient, and the average difference in needle tip positions. On the basis of this work, the segmentation and reconstruction of the interpolation needle in brachytherapy for cervical cancer were also studied by the same algorithm. Experiments have proved that the reconstruction of interpolation needles and applicators in brachytherapy for cervical cancer is of great help to the clinic. Paolo Zaffino [21] et al. proposed an algorithm based on Convolutional Neural Network (CNN) in 2019 for fully automatic segmentation of multiple closely spaced applicators in nuclear magnetic images. The average error distance of the final segmentation is 2.0 ± 3.4 mm, the proportion of false positive and false negative applicators were 6.7% and 1.5%, respectively. By combining Attention gate and U-net in 2020, Xianjin Dai [22] et al. used a total variation (TV) regularization to construct a model to complete the detection and reconstruction of interpolation needles in brachytherapy for prostate cancer with high dose rate MRI, and evaluate the model through tip deviation and movement deviation. Yupei Zhang [23] et al. studied the reconstruction of the position of interpolation needles in prostate cancer guided by ultrasound images through the combination of U-net network and attention in 2020. Similar to the work of Xianjin Dai, they are all interpolation needles for brachytherapy of prostate cancer. For positioning and reconstruction, the only difference is that the ultrasound and MRI guided

images are reconstructed separately, and the reconstruction results are good enough to meet the clinical needs. Fuyue Wang [24] et al. also applied the automatic segmentation of interpolation needles in brachytherapy of prostate cancer by constructing U-net network in 2020, and proved that the model can accurately reconstruct the trajectory of interpolation needles.

In the past two or three years, great attention has been paid to target area delineation and applicator reconstruction. Different network models have also been constructed for reconstruction of applicators or interpolation pins. However, the change is inseparable, basically based on the idea of deep learning, using U-net network or its variants to reconstruct or segment the applicator. Target area sketching is also done using supervised neural networks.

DOSE CALCULATION

Deep learning has not done much research on dose calculation. It is still relatively blank at this stage, but dose calculation is an important step in brachytherapy, and further research is needed [25]. Marc Morcos [26] et al. calculated and studied the dose of nuclear magnetically guided brachytherapy for cervical cancer based on Monte Carlo in 2020, and evaluated the effect of different rotation angles on the intensity-modulated radiotherapy on the dose, laid the foundation for the treatment of complex cervical cancer by intensity modulated radiotherapy. In 2020, Ximeng Mao [27] et al. built a fast brachytherapy deep learning model through convolutional neural networks for dose calculation in brachytherapy planning. The results show that the accuracy is similar to the results obtained by the Monte Carlo algorithm, but the calculation speed is much faster and can be extended to other tumor sites.

APPLICATION OF TREATMENT PLANNING SYSTEM

The treatment plan is formulated by the physicist by the treatment planning system. Many semi-automatic or fully automatic treatment planning systems have been developed to improve the quality of treatment planning while reducing planning time. Some of them have been integrated and successfully tested in commerce. Therefore, deep learning methods are also suitable for the automation of treatment planning [6]. In 2011, Timmy Siauw [28] et al. proposed an optimization model and fast heuristic algorithm for calculating

HDR brachytherapy dose planning, inverse planning (IPIP), and evaluated the measured dose and compared it with the standard dose. Studies have shown that the prostate dose obtained by the algorithm used in this study has clinical significance. C Guthier [29] et al. studied a reverse planning for low-dose-rate brachytherapy in 2015, applied the idea of compressed sensing, developed a reverse planning algorithm, and optimized it to adapt this algorithm to the current best reverse Algorithms are compared and faster. The algorithm can also effectively reduce the cost of intervention. Alexandru Nicolae [30] et al. 2016 used machine learning algorithms to automatically generate a low dose rate brachytherapy plan for the prostate, and compared the pre- and post-implantation treatment plans generated by the machine learning algorithm with the plans made by the physicist. The results showed that the machine-generated plans quality was same with the plan made by the physicist, but it can reduce the planning time and resources. In 2019, Chenyang Shen [31] et al. developed a weight adjustment strategy network based on reverse optimization after high-dose-rate cervical cancer in order to observe the planned dose volume histogram and adjust the organ weighting factor in real time. The experimental results prove that the quality is improved by 10.7% compared to the plan made by the physicist. Maryam Golshan [32] et al. built a model through convolutional neural networks in 2019 to complete the automatic detection of seeds in brachytherapy under the guidance of three-dimensional ultrasound images. The results show that the estimated time for each needle is 1 minute, and the total time is less than 15 minutes. Compared with manual, the model obtained higher accuracy. By 2020, Alexandru Nicolae [33] et al. randomly compared the treatment plan based on machine learning with the traditional manual treatment plan to evaluate the total planning time between the two groups and the dose measurement results after 30 days of implantation, compared with the traditional Manual planning has a great advantage in the time of treatment planning based on machine learning.

OTHERS

In addition to the application of deep learning technology in the above aspects [34], some scholars have also conducted a series of studies in toxicity prediction and other aspects. Several research centers have confirmed the value of machine learning methods in prediction, and now have used deep learning to study the toxicity of lungs, prostate, etc. In 2017, Xin Zhen [34] et al. study on the prediction of rectal toxicity in cervical cancer

radiotherapy based on transfer learning deep convolutional neural network. Convolutional neural network was used to analyze rectal dose distribution and predict rectal toxicity. 42 patients were collected and overcome by transfer learning for quantity problems, training on VGG-16, fine-tuning of the patient's rectal display dose map, and comparison with traditional dose volume parameters, studies have shown that pre-trained CNN can simulate rectal dose distribution and predict rectal toxicity after cervical cancer radiotherapy. There are also some studies on the prediction of the survival rate of deep learning in the next 5 years.

In terms of toxicity research, the application of deep learning is not so extensive for the time being. Future research areas in this area have better prospects. Of course, there are other areas worthy of improvement.

CONCLUSIONS

From the above, we can see that deep learning is becoming more and more widely used in brachytherapy, especially in the treatment of cervical cancer and prostate cancer has been widely developed. Often focusing on target area delineation, applicator reconstruction, radiotherapy planning system and dose calculation, etc., it provides new assistance for improving the clinical treatment effect and the automation of the treatment planning system.

The automation of the brachytherapy plan is one of the future development directions. It is worthwhile to conduct a series of studies to construct different networks through deep learning to solve the problems in brachytherapy. At this stage, target area delineation and applicator reconstruction are both moving in the direction of automation. Future dose calculation and dose optimization are also worthy of attention.

REFERENCES

1. 李龙婕, 邓晓琴. 宫颈癌近距离放射治疗进展[J]. 大连医科大学学报, 2019, 41(3): 193-198.
2. 王金花, 宋金维, 王建东. 人工智能在宫颈癌筛查中的研究进展[J]. 癌症进展, 2019, 17(13): 1503-1505.
3. Lee, J.H., Ha, E.J. and Kim, J.H. (2019) Application of Deep Learning to the Diagnosis of Cervical Lymph Node Metastasis from Thyroid Cancer with CT. European Radiology, 29, 5452-5457. https://doi.org/10.1007/s00330-019-06098-8
4. Doyle, L.A., Yondorf, M., Peng, C., Harrison, A.S. and Den, R.B. (2018) Process Mapping and Time Study to Improve Efficiency of New Procedure Implementation for High-Dose Rate Prostate Brachytherapy. Journal of Healthcare Quality, 40, 19-26.
5. Meyer, P., Noblet, V., Mazzara, C. and Lallement, A. (2018) Survey on Deep Learning for Radiotherapy. Computers in Biology and Medicine, 98, 126-146. https://doi.org/10.1016/j.compbiomed.2018.05.018
6. William, W., Ware, A., Basaza-Ejiri, A.H. and Obungoloch, J. (2018) A Review of Image Analysis and Machine Learning Techniques for Automated Cervical Cancer Screening from Pap-Smear Images. Computer Methods and Programs in Biomedicine, 164, 15-22.
7. Chen, J., Remulla, D., Nguyen, J.H., Aastha, D., Liu, Y., Dasgupta, P., Hung, A.J. (2019) Current Status of Artificial intelligence Applications in Urology and Their Potential to Influence Clinical Practice. BJU International, 124, 567-577. https://doi.org/10.1111/bju.14852
8. Cunha, J.A.M., Flynn, R., Bélanger, C., et al. (2020) Brachytherapy Future Directions. Seminars in Radiation Oncology, 30, 94-106. https://doi.org/10.1016/j.semradonc.2019.09.001
9. Tajbakhsh, N., Jeyaseelan, L., Li, Q., Chiang, J.N., Wu, Z.H. and Ding, X.W. (2020) Embracing Imperfect Datasets: A Review of Deep Learning Solutions for Medical Image Segmentation. Medical Image Analysis, 63, Article ID: 101693. https://doi.org/10.1016/j.media.2020.101693
10. Allman, D., Reiter, A. and Bell, M.A.L. (2018) Photoacoustic Source Detection and Reflection Artifact Removal Enabled by Deep Learning. IEEE Transactions on Medical Imaging, 37, 1464-1477.
11. Huang, X., Wang, J., Tang, F., Zhong, T. and Zhang, Y. (2018) Metal Artifact Reduction on Cervical CT Images by Deep Residual Learning.

BioMedical Engineering Online, 17, Article No. 175. https://doi.org/10.1186/s12938-018-0609-y

12. Karimi, D., Zeng, Q., Mathur, P., et al. (2019) Accurate and Robust Deep Learning-Based Segmentation of the Prostate Clinical Target Volume in Ultrasound Images. Medical Image Analysis, 57, 186-196. https://doi.org/10.1016/j.media.2019.07.005
13. Lei, Y., Tian, S., He, X., et al. (2019) Ultrasound Prostate Segmentation Based on Multidirectional Deeply Supervised V-Net. Medical Physics, 46, 3194-3206. https://doi.org/10.1002/mp.13577
14. Orlando, N., Gillies, D.J, Gyacskov, I., Romagnoli, C., D'Souza, D. and Fenster, A. (2020) Automatic Prostate Segmentation Using Deep Learning on Clinically Diverse 3D Transrectal Ultrasound Images. Medical Physics, 47, 2413-2426. https://doi.org/10.1002/mp.14134
15. 秦楠楠, 薛旭东, 吴爱林, 等. 基于 U-net 卷积神经网络的宫颈癌临床靶区和危 及器官自动勾画的研究[J]. 中国医学物理学杂志, 2020, 37(4): 524-528.
16. Gessert, N., Priegnitz, T., Saathoff, T., et al. (2019) Spatio-Temporal Deep Learning Models for Tip force Estimation during Needle Insertion. International Journal of Computer Assisted Radiology and Surgery, 14, 1485-1493. https://doi.org/10.1007/s11548-019-02006-z
17. Hrinivich, W.T., Morcos, M., Viswanathan, A. and Lee, J.H. (2019) Automatic Tandem and Ring Reconstruction Using MRI for Cervical Cancer Brachytherapy. Medical Physics, 46, 4324-4332. https://doi.org/10.1002/mp.13730
18. Jung, H., Shen, C.Y., Gonzalez, Y., Albuquerque, K. and Jia, X. (2019) Deep-Learning Assisted Automatic Digitization of Interstitial Needles in 3D CT Image Based High Dose-Rate Brachytherapy of Gynecological Cancer. Physics in Medicine & Biology, 64, Article ID: 215003. https://doi.org/10.1088/1361-6560/ab3fcb
19. Jung, H., Gonzalez, Y., Shen, C., Klages, P. and Albuquerque, K. (2019) Deep Learning Assisted Automatic Digitization of Applicators in 3D CT Image-Based High-Dose-Rate Brachytherapy of Gynecological Cancer. Brachytherapy, 18, 841-851. https://doi.org/10.1016/j.brachy.2019.06.003
20. Zaffino, P., Pernelle, G., Mastmeyer, A., et al. (2019) Fully Automatic Catheter Segmentation in MRI with 3D Convolutional Neural Networks: Application to MRI-Guided Gynecologic Brachytherapy.

Physics in Medicine & Biology, 64, Article ID: 165008. https://doi.org/10.1088/1361-6560/ab2f47

21. Dai, X., Lei, Y., Zhang, Y., et al. (2020) Automatic Multi-Catheter Detection Using Deeply Supervised Convolutional Neural Network in MRI-Guided HDR Prostate Brachytherapy. Medical Physics. https://doi.org/10.1002/mp.14307
22. Zhang, Y., Lei, Y., Qiu, R.L.J., et al. (2020) Multi-Needle Localization with Attention U-Net in US-Guided HDR Prostate Brachytherapy. Medical Physics.
23. Wang, F., Xing, L., Bagshaw, H., Buyyounouski, M. and Han, B. (2020) Deep Learning Applications in Automatic Needle Segmentation in Ultrasound-Guided Prostate Brachytherapy. Medical Physics. https://doi.org/10.1002/mp.14328
24. [Wang, X., Wang, P., Li, C., et al. (2018) An Automated Dose Verification Software for Brachytherapy. Journal of Contemporary Brachytherapy, 10, 478-482. https://doi.org/10.5114/jcb.2018.79396
25. Morcos, M. and Enger, S.A. (2020) Monte Carlo Dosimetry Study of Novel Rotating MRI-Compatible Shielded Tandems for Intensity Modulated Cervix Brachytherapy. European Journal of Medical Physics, 71, 178-184. https://doi.org/10.1016/j.ejmp.2020.02.014
26. Mao, X.M., Pineau, J., Keyes, R. and Enger, S.A. (2020) RapidBrachyDL: Rapid Radiation Dose Calculations in Brachytherapy via Deep Learning. International Journal of Radiation Oncology, Biology, Physics.
27. Siauw, T., Cunha, A., Atamtürk, A., Hsu, I.-C., Pouliot, J. and Goldberg, K. (2011) IPIP: A New Approach to Inverse Planning for HDR Brachytherapy by Directly Op timizing Dosimetric Indices. Medical Physics, 38, 4045-4051. https://doi.org/10.1118/1.3598437
28. Guthier, C., Aschenbrenner, K.P., Buergy, D., et al. (2015) A New Optimization Method Using a Compressed Sensing Inspired Solver for Real-Time LDR-Brachytherapy Treatment Planning. Physics in Medicine and Biology, 60, 2179-2194. https://doi.org/10.1088/0031-9155/60/6/2179
29. Nicolae, A., Morton, G., Chung, H., et al. (2017) Evaluation of a Machine-Learning Algorithm for Treatment Planning in Prostate Low-Dose-Rate Brachytherapy. International Journal of Radiation Oncology, Biology, Physics, 97, 822-829. https://doi.org/10.1016/j.ijrobp.2016.11.036

30. Shen, C., Gonzalez, Y., Klages, P., et al. (2019) Intelligent Inverse Treatment Planning via Deep Reinforcement Learning, a Proof-of-Principle Study in High Dose-Rate Brachytherapy for Cervical Cancer. Physics in Medicine & Biology, 64, Article ID: 115013. https://doi.org/10.1088/1361-6560/ab18bf
31. Golshan, M., Karimi, D., Mahdavi, S., et al. (2020) Automatic Detection of Brachytherapy Seeds in 3D Ultrasound Images Using a Convolutional Neural Network. Physics in Medicine & Biology, 65, Article ID: 35016. https://doi.org/10.1088/1361-6560/ab64b5
32. Nicolae, A., Semple, M., Lu, L., et al. (2020) Conventional vs. Machine Learning-Based Treatment Planning in Prostate Brachytherapy: Results of a Phase I Randomized Controlled Trial. Brachytherapy, 19, 470-476. https://doi.org/10.1016/j.brachy.2020.03.004
33. Tian, Z., Yen, A., Zhou, Z., et al. (2019) A Machine-Learning-Based Prediction Model of Fistula Formation after Interstitial Brachytherapy for Locally Advanced Gynecological Malignancies. Brachytherapy, 18, 530-538. https://doi.org/10.1016/j.brachy.2019.04.004
34. Zhen, X., Chen, J.W., Zhong, Z.C., Hrycushko, B., Zhou, L.H., Jiang, S., Albuquerque, K. and Gu, X.J. (2017) Deep Convolutional Neural Network with Transfer Learning for Rectum Toxicity Prediction in Cervical Cancer Radiotherapy: A Feasibility Study. Physics in Medicine & Biology, 62, 8246-8263.

CHAPTER 12

EXPLORING DEEP LEARNING AND TRANSFER LEARNING FOR COLONIC POLYP CLASSIFICATION

Eduardo Ribeiro[1,2], Andreas Uhl[1], Georg Wimmer[1], and Michael Häfner[3]

[1]Department of Computer Sciences, University of Salzburg, Salzburg, Austria

[2]Department of Computer Sciences, Federal University of Tocantins, Palmas, TO, Brazil

[3]St. Elisabeth Hospital, Vienna, Austria

ABSTRACT

Recently, Deep Learning, especially through Convolutional Neural Networks (CNNs) has been widely used to enable the extraction of highly representative features. This is done among the network layers by filtering, selecting, and using these features in the last fully connected layers for pattern classification. However, CNN training for automated endoscopic image

Citation: Eduardo Ribeiro, Andreas Uhl, Georg Wimmer, Michael Häfner, "Exploring Deep Learning and Transfer Learning for Colonic Polyp Classification", Computational and Mathematical Methods in Medicine, vol. 2016, Article ID 6584725, 16 pages, 2016. https://doi.org/10.1155/2016/6584725.

classification still provides a challenge due to the lack of large and publicly available annotated databases. In this work we explore Deep Learning for the automated classification of colonic polyps using different configurations for training CNNs from scratch (or full training) and distinct architectures of pretrained CNNs tested on 8-HD-endoscopic image databases acquired using different modalities. We compare our results with some commonly used features for colonic polyp classification and the good results suggest that features learned by CNNs trained from scratch and the "off-the-shelf" CNNs features can be highly relevant for automated classification of colonic polyps. Moreover, we also show that the combination of classical features and "off-the-shelf" CNNs features can be a good approach to further improve the results.

INTRODUCTION

The leading cause of deaths related to the intestinal tract is the development of cancer cells (polyps) in its many parts. An early detection (when the cancer is still at an early stage) and a regular exam to everyone over an age of 50 years can reduce the risk of mortality among these patients. More specifically, colonic polyps (benign tumors or growths which arise on the inner colon surface) have a high occurrence and are known to be precursors of colon cancer development.

Endoscopy is the most common method for identifying colon polyps and several studies have shown that automatic detection of image regions which may contain polyps within the colon can be used to assist specialists in order to decrease the polyp miss rate [1, 2].

The automatic detection of polyps in a computer-aided diagnosis (CAD) system is usually performed through a statistical analysis based on color, shape, texture, or spatial features applied to the videos frames [3–6]. The main problems for the detection are the different aspects of color, shape, and textures of polyps, being influenced, for example, by the viewing angle, the distance from the capturing camera, or even by the colon insufflation as well as the degree of colon muscular contraction [5].

After detection, the colonic polyps can be classified into three different categories: hyperplasic, adenomatous, and malignant. Kudo et al. [7] proposed the so-called "pit-pattern" scheme to help in diagnosing tumorous lesions once suspicious areas have been detected. In this scheme, the mucosal surface of the colon can be classified into 5 different types designating the size, shape, and distribution of the pit structure [8, 9].

As can be seen in the Figures 1(a)–1(d), these five patterns also allow the division of the lesions into two main classes: (1) normal mucosa or hyperplastic polyps (healthy class) and (2) neoplastic, adenomatous, or carcinomatous structures (abnormal class). This approach is quite relevant in clinical practice as shown in a study by Kato et al. [10].

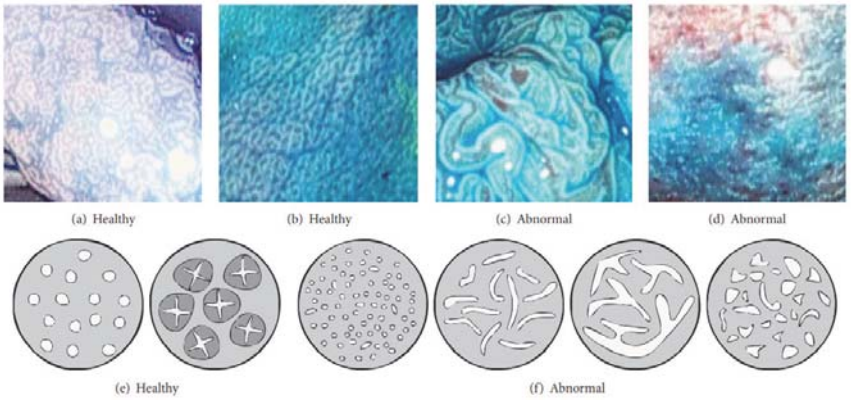


Figure 1. Example images of the two classes (a–d) and the pit-pattern types of these two classes (e–f).

In the literature, existing computer-aided diagnosis techniques generally make use of feature extraction methods of color, shape, and texture in combination with machine learning classifiers to perform the classification of colon polyps [9, 11, 12]. For example, the dual-tree complex wavelet transform DT-CWT features proved to be quite suitable for the distinction of different types of polyps as can be seen in many works like, for example, [13–15]. Other features were also proved to be quite suitable for colonic polyp classification as the Gabor wavelets [16], vascularization features [17], and directional wavelet transform features [18]. Particularly, in the work of Wimmer et al. [18], using the same 8 colonic polyp databases of this work, an average accuracy of 80.3% was achieved in the best scenario. In this work, we achieve an average accuracy of 93.55% in our best scenario.

The main difficulty of the feature extraction methods is the proper characterization of these patterns due to several factors as the lack or excess of illumination, the blurring due to movement or water injection, and the appearance of polyps [5, 9]. Also, to find a robust and a global feature extractor that summarizes and represents all these pit-pattern structures in a single vector is very difficult and Deep Learning can be a good alternative to surpass these problems. In this work we explore the use of Deep Learning

through Convolutional Neural Networks (CNNs) to develop a model for robust feature extraction and efficient colonic polyp classification.

To achieve this, we test the use of CNNs trained from scratch (or full training) and off-the-shelf CNNs (or pretrained) using them as medical imaging feature extractors. In the case of the CNN full training we assume that a feature extractor is formed during the CNN training, adapting to the context of the database and particularly in the case of off-the-shelf CNNs we consider that the patterns learned in the original database can be used in colonoscopy images for colonic polyp classification. In particular, we explore two different architectures for the training from scratch and six different off-the-shelf architectures, describing and analyzing the effects of CNNs in different acquisition modes of colonoscopy images (8 different databases). This study was motivated by recent studies in computer vision addressing the emerging technique of Deep Learning presented in the next section.

MATERIALS AND METHODS

Using CNNs on Small Datasets

Some researchers propose replacing handcrafted feature extraction algorithms with Deep Learning approaches that act as features extractor and image classifier at the same time [19]. For example, the Deep Learning approach using CNNs takes advantage of many consecutive convolutional layers followed by pooling layers to reduce the data dimensionality making it, concomitantly, invariant to geometric transformations. Such convolution filters (kernels) are built to act as feature extractors during the training process and recent research indicates that a satisfactorily trained CNN with a large database can perform properly when it is applied to other databases, which can mean that the kernels can turn into a universal feature extractor [19]. Also, Convolutional Neural Networks (CNNs) have been demonstrated to be effective for discriminative pattern recognition in big data and in real-world problems, mainly to learn both the global and local structures of images [20].

Many strategies exploiting CNNs can be used for medical image classification. These strategies can be employed according to the intrinsic characteristics of each database [21] and two of them, mostly used when it comes to CNN training, are described in the following part.

When the available training database is large enough, diverse, and very different from the database used in all the available pretrained CNNs (in a case of transfer learning), the most appropriate approach would be to initialize the CNN weights randomly (training the *CNN trained from scratch*) and train it according to the medical image database for the kernels domain adaptation, that is, to find the best way to extract the features of the data in order to classify the images properly. The main advantage of this approach is that the same method can be used for the extraction of strong features that are invariant to distortion and position at the same time of the image classification. Finally, the Neural Network Classifier can make use of these inputs to delineate more accurate hyperplanes helping the generalization of the network.

This strategy, although ideal, is not widely used due to the lack of large and annotated medical image database publicly available for training the CNN. However, some techniques can assist the CNN training from scratch with small datasets and the most used approach is data augmentation. Basically, in data augmentation, transformations are applied to the image making new versions of it to increase the number of samples in the database. These transformations can be applied in both the training and the testing phase and can use different strategies such as cropping (overlapped or not), rotation, translation, and flipping [22]. Experiments show that using these techniques can be effective to combat overfitting in the CNN training and improve the recognition and classification accuracy [22, 23].

Furthermore, when the database is small, the best alternative is to use an *off-the-shelf CNN* [21]. In this case, using a pretrained CNN, the last or next-to-last linear fully connected layer is removed and the remaining pretrained CNN is used as a feature extractor to generate a feature vector for each input image from a different database. These feature vectors can be used to train a new classifier (such as a support vector machine, SVM) to classify the images correctly. If the original database is similar to the target database, the probability that the high-level features describe the image correctly is high and relevant to this new database. If the target database is not so similar to the original, it can be more appropriate to use higher-level features, that is, features from previous layers of CNN.

In this work, besides using a CNNs trained from scratch, we consider the knowledge transfer between natural images and medical images using off-the-shelf pretrained CNNs. The CNN will project the target database samples into a vector space where the classes are more likely to be separable.

This strategy was inspired by the work of Oquab et al. [24], which uses a pretrained CNN on a large database (ImageNet) to classify images in a smaller database (Pascal VOC dataset) with improved results. Unlike that work, rather than copy the weights of the original pretrained CNN to the target CNN with additional layers, we use the pretrained CNN to project data into a new feature space through the propagation of the colonic polyp database into the CNN getting the resultant vector from the last CNNs layer, obtaining a new representation for each input sample. Subsequently, we use the feature vector set to train a linear classifier (e.g., support vector machines) in this representation to evaluate the results as used in [25, 26].

CNNs and Medical Imaging

In recent years there has been an increased interest in machine learning techniques that is based not on hand-engineered feature extractors but using raw data to learn the representations [19].

Among the development of efficient parallel solvers together with GPUS, the use of Deep Learning has been extensively explored in the last years in different fields of application. Deep Learning is intimately related to the use of raw data to do high-level representations of this knowledge through a large volume of annotated data. However, when it comes to the medical area, this type of application is limited by the problem of the lack of large, annotated, and publicly available medical image databases such as the existing natural image databases. Additionally, it is a difficult and costly task to acquire and annotate such images and due to the specific nature of different medical imaging modalities which seems to have different properties according to each modality the situation is even aggravated [21, 27].

Recently, works addressing the use of Deep Learning techniques in medical imaging have been explored in many different ways mainly using CNNs trained from scratch. In biomedical applications, examples include mitosis detection in digital breast cancer histology [28] and neuronal segmentation of membranes in electron microscopy [29]. In Computer-Aided Detection systems (CADe systems), examples include a CADe of pulmonary embolism [30], computer-aided anatomy detection in CT volumes [31], lesion detection in endoscopic images [32], detection of sclerotic spine metastases [33], and automatic detection of polyps in colonoscopy videos [27, 34, 35]. In medical image classification, CNNs are used for

histopathological image classification [36], digestive organs classification in wireless capsule endoscopy images [37, 38], and automatic colonic polyp classification [39]. Besides that, CNNs have also been explored to improve the accuracy of CADe systems knee cartilage segmentation using triplanar CNNs [40].

Other recent studies show the potential for knowledge transfer from natural images to the medical imaging domain using off-the-shelf CNNs. Examples include the identification and pathology of X-ray and computer tomography modalities [25], automatic classification of pulmonary perifissural nodules [41], pulmonary nodule detection [26], and mammography mass lesion classification [42]. Moreover, in [26], Van Ginneken et al. show that the combination of CNNs features and classical features for pulmonary nodule detection can improve the performance of the model.

CNNs Trained from Scratch: Architecture

In this section we briefly describe the components of a CNN and how it can be used to perform the CNN from scratch.

A CNN is very similar to traditional Neural Networks in the sense of being constructed by neurons with their respective weights, biases, and activation functions. The structure is basically formed by a sequence of convolution and pooling layers ending in a fully connected Neural Network as shown in Figure 2. Generally, the input of a CNN is $m \times m \times d$ image (or patch), where $m \times m$ is the dimension of the image and d is the number of channels (depth) of the image. The convolutional layer consists of k learnable filters (also called kernels) with size $n \times n \times d$ where $n \leq m$ which are convolved with the input image resulting in the so-called activation maps or feature maps. As classic Neural Networks, the convolution layer outputs are submitted to an activation function, for example, the ReLU rectifier function $f(x) = \max(0, x)$, where x is the neuron input. After the convolution, a pooling layer is included to subsample the image by average functions (mean) or max-pooling over regions of size $p \times p$. These functions are used to reduce the dimensionality of the data in the following layers (upper layers) and to provide a form of invariance to translation thus making overfitting control. In the convolution and pooling layers the stride has to be specified; the larger the stride, the smaller the overlapping, decreasing the output volume dimensions.

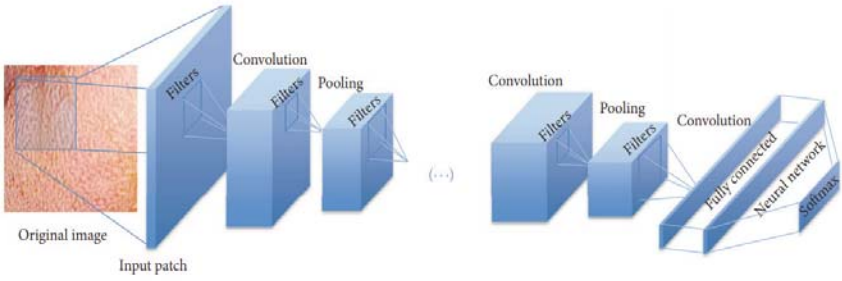


Figure 2. An illustration of the CNN architecture for colonic polyp classification.

At the end of the CNN there is a fully connected layer as a regular Multilayer Neural Network with the Softmax function that generates a well-formed probability distribution on the outputs. After a supervised training, the CNN is ready to be used as a classifier or as a feature extractor in the case of transfer learning.

CNNs and Transfer Learning

Transfer learning is a technique used to improve the performance of machine learning by harnessing the knowledge obtained by another task. According to Pan and Yang [43], transfer learning can be defined by the following model. We give a domain D having two components: a feature space $X = \{x_1, x_2, \ldots, x_n\}$ and a probabilistic distribution $P(X)$; that is, $D = \{X, P(X)\}$. Also, we give a task T with two components: a ground truth $Y = \{y_1, y_2, \ldots, y_n\}$ and an objective function $T = \{Y, f(\cdot)\}$ assuming that this function can be learned through a training database. Function $f(\cdot)$ can be used to predict the correspondent class $f(x)$ of a new instance x. From a probabilistic point of view, $f(x)$ can be written as $P(y \mid x)$. In colonic polyp classification, usually, a feature extractor is used to generate the feature space. A given training database X associated to the ground truth Y consisting of the pairs $\{x_i, y_i\}$ is used to train and "learn" the function $f(\cdot)$ or $P(y \mid x)$ until it reaches a defined and acceptable error rate between the result of the function $f(x)$ and the ground truth Y.

In case of transfer learning, given a source domain $D_S = \{(x_{S_1}, y_{S_1}), (x_{S_2}, y_{S_2}), \ldots, (x_{S_n}, y_{S_n})\}$ and the learning task T_s and the target domain $D_T = \{(x_{T_1}, y_{T_1}), (x_{T_2}, y_{T_2}), \ldots, (x_{T_m}, y_{T_m})\}$ and the learning task , transfer learning aims to help improve the learning of the target predictive function $f_T(\cdot)$ using the knowledge in D_S and T_S, where $D_T \neq D_S$ and $T_T \neq T_S$.

Among the various categories of transfer learning, one, called inductive transfer learning, has been used with success in the pattern recognition area. In the inductive transfer learning approach an annotated database is necessary for the source domain as well as for the target domain. In this work, we apply transfer learning between two very different tasks using different labels $(Y_T \neq Y_S)$ and different distributions $(P(Y_T \mid X_T) \neq P(Y_S \mid X_S))$. To bypass the difference between the probability distribution of the images $P(X_S)$, the last layer from the original function $f_S(\cdot)$ directly connected to the classification is removed being replaced by other linear function (as SVM) to adapt it to the new task T_T turning into the function $f_T(\cdot)$. In the following sections the functions $f_T(\cdot)$ used in this work are presented. Also, the use of transfer learning using pretrained CNNs can help to avoid the problem of lack of data in the medical field. The works of Razavian et al. [19] and Oquab et al. [24] suggest that the use of CNNs intermediate layer outputs can be used as input features to train other classifiers (such as support vector machines) for a number of other applications different from the original CNN obtaining a good performance.

Despite the difference between natural and medical images, some feature descriptors designed especially for natural images are used successfully in medical image detection and classification, for example, texture-based polyp detection [3], Fourier and Wavelet filters for colon classification [18], shape descriptors [44], and local fractal dimension [45] for colonic polyp classification. Additionally, recent studies show the potential of the knowledge transfer between natural and medical images using pretrained (off-the-shelf) CNNs [34, 46].

Experimental Setup

Data

The use of an integrated endoscopic apparatus with high-resolution acquisition devices has been an important object of research in clinical decision support system area. With high-magnification colonoscopies it is possible to acquire images up to 150-fold magnified, revealing the fine surface structure of the mucosa as well as small lesions. Recent work related to classification of colonic polyps used highly-detailed endoscopic images in combination with different technologies divided into three categories: high-definition endoscope (with or without staining the mucosa) combined with the i-Scan

technology (1, 2, and 3) [18], high-magnification chromoendoscopy [8], and high-magnification endoscopy combined with narrow band imaging [47].

Specifically, the i-Scan technology (Pentax) used in this work is an image processing technology consisting of the combination of surface enhancement and contrast enhancement aiming to help detect dysplastic areas and to accentuate mucosal surfaces and applying postprocessing to the reflected light being called virtual chromoendoscopy (CVC) [44].

There are three i-Scan modes available: i-Scan1, which includes surface enhancement and contrast enhancement, i-Scan2 that includes surface enhancement, contrast enhancement, and tone enhancement, and i-Scan3 that, besides including surface, contrast, and tone enhancement, increases lighting emphasizing the features of vascular visualization [18]. In this work we use an endoscopic image database (CC-i-Scan Database) with 8 different imaging modalities acquired by an HD endoscope (Pentax HiLINE HD+ 90i Colonoscope) with images of size 256×256 extracted from video frames either using the i-Scan technology or without any computer virtual chromoendoscopy $(\neg CVC)$.

Table 1 shows the number of images and patients per class in the different i-Scan modes. The mucosa is either stained or not stained. Despite the fact that the frames were originally in high-definition, the image size was chosen (i) to be large enough to describe a polyp and (ii) small enough to cover just one class of mucosa type (only healthy or only abnormal area). The image labels (ground truth) were provided according to their histological diagnosis.

Table 1. Number of images and patients per class of the CC-i-Scan databases gathered with and without CC (staining) and computed virtual chromoendoscopy (CVC)

i-Scan mode	No staining				Staining			
	¬CVC	i-Scan1	i-Scan2	i-Scan3	¬CVC	i-Scan1	i-Scan2	i-Scan3
Non-neoplastic								
Number of images	39	25	20	31	42	53	32	31
Number of patients	21	18	15	15	26	31	23	19
Neoplastic								
Number of images	73	75	69	71	68	73	62	54
Number of patients	55	56	55	55	52	55	52	47
Total number of images	112	100	89	102	110	126	94	85

Employed CNN Techniques

Due to the limitation of colonic polyp images to train a good CAD system from scratch, the main elements of the proposed method are defined in order to (1) extract and preprocess images aiming to have a database with a suitable size, (2) use CNNs for learning representative features with good generalization, and (3) enable the use of methods to avoid overfitting in the training phase.

To test the application of a CNN trained from scratch we used the i-Scan1 database without chromoscopy (staining the mucosa) that presents a good performance in the tests using classical features and pretrained CNNs (on average) and subsequently applying the best configuration to the i-Scan3 without chromoscopy database that presented the best results among the classical features results.

In the first experiment of CNN full training, it is proposed that an architecture should be trained with subimages of size $227 \times 227 \times 3$ based on the work of [20] to fit into the chosen architecture. Usually, some simple preprocessing techniques are necessary for the image feature generation. In this experiment we apply normalization by subtracting the mean and dividing by the standard deviation of its elements as in [48] corresponding to local brightness and normalization contrast. We also perform data augmentation by flipping each original image horizontally and vertically and rotating the original image 90° to the right and left. Besides that, we flipped horizontally the rotated images, and then we flipped vertically the horizontally flipped image, totalizing 7 new samples for each original image. After the data augmentation (resulting in 800 images), we randomly extract 75 subimages of size $227 \times 227 \times 3$ from each healthy image and 25 subimages from each abnormal image for the training set to balance the number of images in each class.

Also, in this experiment, to be able to compare the different architectures in a faster way, we used cross-validation evaluation with 10 different CNNs for each architecture. In nine of them, we removed 56 patients for training and used 6 for tests and, in one of them, we removed 54 patients for training and used 8 for test to assure that all the 62 patients are tested. The accuracy result given for each architecture is the average accuracy from each of the 10 CNNs trained based on the final classification of each image between the two classes.

For the second experiment in the CNN full training we propose to extract subimages of size 128×128 from the original images using the same

approach as in the first experiment. In this case, we explore the hypothesis that the colonic polyp classification with the CNN can be done only with a part of the image, and then we trained the network with smaller subimages instead of the entire image. This helps to reduce the size of the network reducing its complexity and can allow different polyp classifications in the same image using different subimages in different parts of the image. Additionally, choosing smaller regions in a textured image can diminish the degree of intraimage variances in the dataset as the neighborhood is limited.

Besides the different architectures for the training from scratch, we mainly explore six different off-the-shelf CNN architectures trained to perform classification on the ImageNet ILSVRC challenge data. The input of all tested pretrained CNNs has size of 224 × 224 × 3 and the descriptions as well as the details of each CNN are given as follows:(i)The *CNN VGG-VD* [49] uses a large number of layers with very small filters (3 × 3) divided into two architectures according to the number of their layers. The CNN *VGG-VD16* has 16 convolution layers and five pooling layers while the CNN *VGG-VD19* has 19 convolution layers, adding one more convolutional layer in three last sequences of convolutional layers. The fully connected layers have 4096 neurons followed by a Softmax classifier with 1000 neurons corresponding to the number of classes in the ILSVRC classification. All the layers are followed by a rectifier linear unit (ReLU) layer to induce the sparsity in the hidden units and reduce the gradient vanishing problem.(ii) The *CNN-F* (also called Fast CNN) [22] is similar to the CNN used by Alex et al. [20] with 5 convolutional layers. The input image size is 224 × 224 and the fast processing is granted by the stride of 4 pixels in the first convolutional layer. The fully connected layers also have 4096 neurons as the CNN VGG-VD. Besides the original implementation, in this work, we also used the MatConvNet implementation (beta17 [50]) of this architecture trained with batch normalization and minor differences in its default hyperparameters and called here *CNN-F MCN*.(iii)The *CNN-M* architecture (Medium CNN) [22] also has 5 convolutional layers and 3 pooling layers. The number of filters is higher than the Fast CNN: 96 instead of 64 filters in the first convolution layer with a smaller size. We also use the MatConvNet implementation called *CNN-M MCN*.(iv)The *CNN-S* (Slow CNN) [22] is related to the "accurate" network from the Overfeat package [51] and also has smaller filters with a stride of 2 pixels in the first convolutional layer. We also use the MatConvNet implementation called *CNN-S MCN*.(v) The *AlexNet* CNN [20] has five convolutional layers, three pooling layers (after layers 2 and 5), and two fully connected layers. This architecture is

similar to the CNN-F, however, with more filters in the convolutional layers. We also use the MatConvNet implementation called *AlexNet MCN*.(vi) The *GoogleLeNet* [52] CNN has the deepest and most complex architecture among all the other networks presented here. With two convolutional layers, two pooling layers, and nine modules also called "inception" layers, this network was designed to avoid patch-alignment issues introducing more sparsity in the inception modules. Each module consists of six convolution layers and one pooling layer concatenating these filters of different sizes and dimensions into a single new filter.

In order to form the feature vector using the pretrained CNNs, all images are scaled using bicubic interpolation to the required size for each network, in the case of this work, $224 \times 224 \times 3$. The vectors obtained by the linear layers of the CNN have size of 1024×1 for the GoogleLeNet CNN and of 4096×1 for the other networks due to their architecture specificities.

Classical Features

To allow the CNN features comparison and evaluation, we compared them with the results obtained by some state-of-the-art feature extraction methods for the classification of colonic polyps [18] shortly explained in the next items.(i)*BSAG-LFD*. The Blob Shape adapted Gradient using Local Fractal Dimension method combines BA-LFD features with shape and contrast histograms from the original and gradient image [45].(ii)*Blob SC*. The Blob Shape and Contrast algorithm [44] is a method that represents the local texture structure of an image by the analyses of the contrast and shape of the segmented blobs.(iii)*Shearlet-Weibull*. Using the Discrete Shearlet Transform this method adopts regression to investigate dependencies across different subband levels using the Weibull distribution to model the subband coefficient distribution [53].(iv)*GWT Weibull*. The Gabor Wavelet Transform function can be dilated and rotated to get a dictionary of filters with diverse factors [18] and its frequency using different orientations is used as a feature descriptor also using the Weibull distribution.(v)*LCVP*. In the Local Color Vector Patterns approach, a texture operator computes the similarity between neighboring pixels constructing a vector field from an image [12].(vi)*MB-LBP*. In the Multiscale Block Local Binary Pattern approach [54], the LBP computation is done based on average values of block subregions. This approach is used for a variety image processing applications including endoscopic polyp detection and classification [12].

For the classical features, the classification accuracy is also computed using an SVM classifier, however, with the original images (without resizing) trained using the leave-one-patient-out cross-validation strategy assuring that there are no images from patients of the validation set in the training set as in [55] to make sure the classifier generalizes to unseen patients.

This cross-validation is applied to the classical feature extraction methods from the literature as well as to the full training and off-the-shelf CNNs features. The accuracy measure is used to allow an easy comparability of results due to the high number of methods and databases to be compared.

RESULTS AND DISCUSSION

CNNs Trained from Scratch

In the first experiment for the CNN full training, we first use the configuration similar to [20] that can be seen in Table 2 and it can be concluded that the accuracy result was not satisfactory (79%).

This can be explained by the fact that Neural Networks involving a large number of inputs require a great amount of computation in training, requiring more data to avoid overfitting (which is not available given the size of our dataset).

Table 2. CNN configuration for input subimages of size 227 × 227 × 3 and its respective accuracy in %

Size of inputs	Number of convolutional filters/size								Connected layer
	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6	Layer 7	Layer 8	
227 × 227 × 3	96/11 × 11	256/5 × 5	384/3 × 3	384/3 × 3	256/3 × 3	384/3 × 3	384/3 × 3	4096/6 × 6	4096
Accuracy: 79.00									

For the second experiment, the hyperparameters presented in Table 3 were selected based on the works [48, 56] and empirical adjustment tests in the architecture such as changing the size and number of filters as well as the number of units in the fully connected layer were made and are also shown in Table 3. It can be seen that the architecture CNN-05 obtained the best results, therefore, chosen to perform the subsequent tests.

Table 3. Accuracy results from different CNN configurations for inputs of size 128 × 128 × 3 in %

Network index	Number of convolutional filters/size			Connected layer	Acc
	Layer 1	Layer 2	Layer 3		
CNN-01	48/7 × 7	72/4 × 4	512/5 × 5	512	76.00
CNN-02	48/11 × 11	72/5 × 5	512/6 × 6	512	84.00
CNN-03	24/11 × 11	48/5 × 5	1024/6 × 6	1024	86.00
CNN-04	24/11 × 11	72/4 × 4	2048/5 × 5	2048	80.00
CNN-05	48/11 × 11	72/5 × 5	1024/6 × 6	1024	*87.00*

In the third experiment, with the CNN-05 configuration, we trained one CNN for each patient from the database (leave-one-patient-out (LOPO) cross-validation). Specifically, the results from the CNNs presented in Table 4 are the mean values of the validation set from 62 different CNNs, one for each patient, implemented using the MatConvNet framework [50]. After training the CNN, in the evaluation phase, the final decision for a 256 × 256 pixel image of the dataset is obtained by majority voting of the decisions of all 128 × 128 pixel subimages (patches). One of the advantages of this approach is the opportunity to have a set of decisions available to acquire the final decision for one image. Also, the redundancy of overlapping subimages can increase the system accuracy likewise to give the assurance of certainty for the overall decision.

Table 4. Accuracy of different strides for overlapping subimages in the CNN-05 evaluation for i-Scan1 database in %

Stride	Number of subimages	Accuracy
1	16384	89.00
5	676	89.00
20	49	90.00
32	25	*91.00*
48	9	87.00
Random	9	87.00
Random	25	89.00
Random	49	89.00

As it can be seen in Table 4, first we tested with a stride of 1 extracting the maximum number of 128 × 128 subimages available, totalizing 16384 subimages for each image, resulting in an accuracy of 89.00%. This evaluation is very computationally expensive to perform, so we decided to evaluate with different strides resulting in different number of subimages as it is shown in Table 4. We also perform a random patch extraction and it can

be concluded that there is not much difference between 16384 subimages or just 25 cropped subimages (accuracy of 91.00%), saving considerable computation time and achieving good results. Besides that, using the same procedure we evaluate the architecture CNN-05 for the i-Scan3 database without staining the mucosa that presented the best results among the classical features and results are presented in Table 5.

Table 5. Accuracy of CNN-05 architecture comparing to classical features for the i-Scan1 and i-Scan3 databases in %

Methods	i-Scan1	i-Scan3
CNN-05	*91.00*	*89.00*
CNN-05 + SVM – LFCL	83.00	72.55
CNN-05 + SVM – PFCL	80.00	66.67
BSAG-LFD	86.87	82.87
Blob SC	83.33	75.22
Shearlet-Weibull	76.67	86.80
GWT-Weibull	78.67	84.28
LCVP	66.00	77.12
MB-LBP	80.67	83.37

For a better comparability of results, we trained an SVM with the extracted vectors from the last fully connected layers (LFCL) and from the prior fully connected layers (PFCL) of CNN-05 as we make in the transfer learning approach explained in the next section. The vectors are extracted from 25 cropped subimages of size 128 × 128 (with stride of 32 pixels) feedforwarded into the CNN-05 subsequently used to train a support vector machine also using the LOPO cross-validation [55]. The results from this approach using the CNN-05 architecture trained with the i-Scan1 and i-Scan3 without staining the mucosa databases are presented in Table 5. As it can be seen, using the last-layer vectors to train an SVM does not improve the results, mainly because the amount of data is not sufficient to generate representative features to be applied into a linear classifier. However, when the CNN is fully trained, the results surpass the classical features results as can be seen also in Table 5 mostly because the last layers are more suitable to design nonlinear hyperplanes in the classification phase. However, the problem of lack of data still is an issue and using all the information in the image would be better than using cropped patches. The significance comparison between the methods will be explored in the next section. Therefore, in order to try solving this problem, we also propose the use of transfer learning by pretrained CNNs that will be also explained in the next section.

Pretrained CNNs

In this section we present the experiments made exploring the 11 different off-the-shelf CNN architectures with the classical features trying to achieve better results than the CNN trained from scratch. As well as in the CNN trained from scratch, we use the i-Scan1 without staining the mucosa database for the first experiments.

In the first experiment, we tested the use of more samples from the same image using overlapping patches by randomly cropping 25 images of size 224 × 224 × 3 of each original image of size 256 × 256 × 3 (resized using bicubic interpolation for the tests presented in Table 8) increasing the database from 100 to 2500 images. The obtained results after the feature extraction performed by the CNN and after the SVM training also using the LOPO cross-validation are presented in Table 6.

Table 6. Results from i-Scan1 database with images resized to 224 × 224 and cropped in 25 patches of size 224 × 224

	CNN-F	CNN-M	CNN-S	CNN-F MCN	CNN-M MCN	CNN-S MCN	Google LeNet	VGG VD16	VGG VD19	AlexNet	AlexNet MCN	$\overline{X}$
Resizing image	89.33	90.67	90.00	82.00	90.67	91.42	90.67	85.33	82.67	87.33	84.67	**87.70**
Cropping 25 images	84.00	82.67	84.67	78.67	84.67	88.67	91.29	89.67	78.67	85.33	85.33	84.87

It can be observed that, in this case, the use of more samples from the same image does not provide any significant improvement in the results. On the average, resizing the images produces an accuracy of 87.70% while cropping the images produces an average of 84.87%. One of the explanations for this is that, in case of resized images, there is more information about the polyp to provide to the network, so the CNN can abstract more information and form a more robust and intrinsic vector from the actual features of the lesion. However, in three cases (GoogleLeNet, VGG-VD16, and AlexNet MCN), the results using smaller cropped images surpassed the results using the entire image.

In the second experiment, still using i-Scan1 without staining the mucosa database, we also tested the use of other layers of CNNs to extract features. Table 7 shows the results obtained when the vectors are extracted from the last fully connected layer and when the vectors are from the prior fully connected layer. In the case of the last layer, the results are worse (87.70% against 85.75% on average) because the vectors from the prior fully connected layer are more related to high-level features describing the natural images used for training the original CNNs that are very different

from the features to describe colonic polyp images. However, in this case, the results from CNN-F and AlexNet CNN are better using the features from the last fully connected layers.

Table 7. Results from i-Scan1 database with images resized to 224 × 224 using the last fully connected layer and the prior fully connected layer

	CNN-F	CNN-M	CNN-S	CNN-F MCN	CNN-M MCN	CNN-S MCN	Google LeNet	VGG VD16	VGG VD19	AlexNet	AlexNet MCN	$\overline{X}$
Prior fully connected layer	89.33	90.67	90.00	82.00	90.67	91.42	90.67	85.33	82.67	87.33	84.67	**87.70**
Last fully connected layer	90.67	84.67	85.33	78.67	88.00	89.33	90.67	84.67	79.33	81.33	90.67	85.75

Table 8. Accuracies of the methods for the CC-i-Scan databases in %

Methods	No staining				Staining				
	¬CVC	i-Scan1	i-Scan2	i-Scan3	¬CVC	i-Scan1	i-Scan2	i-Scan3	$\overline{X}$
1: CNN-F	86.16	89.33	80.65	88.41	86.52	81.40	84.22	80.62	84.66
2: CNN-M	87.45	90.67	81.38	83.58	87.99	89.55	87.40	90.53	87.31
3: CNN-S	88.03	90.00	87.01	77.33	87.25	82.68	87.40	75.54	84.41
4: CNN-F MCN	88.84	82.00	73.15	90.73	85.78	89.55	89.72	83.15	85.36
5: CNN-M MCN	89.53	90.67	*88.88*	*94.66*	86.97	89.29	87.40	90.53	**89.74**
6: CNN-S MCN	90.12	*91.42*	81.38	79.85	89.18	*93.49*	81.10	84.77	86.41
7: GoogleLeNet	79.65	90.67	72.43	74.51	88.27	80.46	75.60	84.08	80.70
8: VGG-VD16	87.45	85.33	86.38	79.65	*92.47*	89.80	*95.26*	*92.38*	88.59
9: VGG-VD19	83.49	82.67	83.88	87.71	*92.47*	83.98	94.46	85.59	86.78
10: AlexNet	*91.40*	87.33	75.65	89.32	87.71	83.03	84.22	79.24	84.73
11: AlexNet MCN	89.42	84.67	78.88	83.78	89.36	83.55	81.10	78.32	83.63
$\overline{X}$	87.41	87.70	80.88	84.50	88.54	86.07	86.17	84.06	85.67
12: BSAG-LFD	*86.27*	*86.87*	*84.60*	82.87	70.20	*80.63*	78.78	71.39	**80.20**
13: Blob SC	77.67	83.33	82.10	75.22	59.28	78.83	66.13	59.83	72.79
14: Shearlet-Weibull	73.72	76.67	79.60	*86.80*	*81.30*	69.91	72.38	*83.63*	78.00
15: GWT-Weibull	79.75	78.67	70.25	84.28	*81.30*	74.54	77.17	83.39	78.66
16: LCVP	76.60	66.00	47.75	77.12	77.45	79.00	70.01	69.56	70.43
17: MB-LBP	78.26	80.67	81.38	83.37	69.29	70.60	77.22	78.32	77.38
$\overline{X}$	78.71	78.70	74.28	81.61	73.13	75.58	73.61	74.35	76.24
Fusion 5/8	88.84	85.33	83.88	92.14	93.12	90.49	96.88	94.00	90.58
Fusion 5/12	92.79	*92.67*	88.88	*96.98*	87.71	90.49	88.26	90.53	91.03
Fusion 5/8/12	*95.94*	90.00	88.88	92.14	92.30	91.43	97.63	*97.46*	93.22
Fusion 5/8/14	91.51	88.67	87.10	93.75	*94.68*	91.43	*98.44*	95.85	92.67
Fusion 5/8/15	90.91	90.00	88.88	92.14	93.94	89.80	96.88	95.61	92.27
Fusion 5/8/12/14	93.38	88.00	*91.38*	93.75	93.49	*92.12*	97.63	94.92	93.08
Fusion 5/8/12/17	93.38	90.00	*91.38*	93.75	92.75	*92.12*	97.63	*97.46*	**93.55**
CNN-05	—	91.00	—	89.00	—	—	—	—	—
CNN-05 + SVM	—	83.00	—	72.55	—	—	—	—	—

Based on the results from the two experiments explained before, we tested the methods with all the other databases using the inputs resized to size 224 × 224 × 3 by bicubic interpolation and extracting the features from the prior fully connected layer. The accuracy results for the colonic polyp classification for the 8 different databases are reported in Table 8. As can be seen, the results in Table 8 are divided into three groups: off-the-shelf features, classical features, and the fusion between off-the-shelf features and classical features that will be explained as follows.

Among the 11 pretrained CNNs investigated, the CNNs that present lower performance were GoogleLeNet, CNN-S, and AlexNet MCN. These results may indicate that such networks themselves are not sufficient to be considered off-the-shelf feature extractors for the polyp classification task.

As it can be seen in Table 8, the pretrained CNN that presents the best result on average for the different imaging modalities $(\overline{X})$ is the CNN-M network trained with the MatConvNet parameters (89.74%) followed by the CNN VGG-VD16 (88.59%). These deep models with smaller filters generalize well with other datasets as it is shown in [49], including texture recognition, which can explain the better results in the colonic polyp database. However, there is a high variability in the results and thus it is difficult to draw general conclusions.

Many results obtained from the pretrained CNNs surpassed the classic feature extractors for colonic polyp classification in the literature. The database that presents the best results using off-the-shelf features is the database staining the mucosa without any i-Scan technology (¬CVC, 88.54% on average). In the case of classical features, the database with the best result on average is the database using the i-Scan3 technology without staining the mucosa (81.61%).

To investigate the differences in the results we assess the significance of them using the McNemar test [57]. By means of this test we analyze if the images from a database are classified differently or similarly when comparing two methods. With a high accuracy it is supposed that the methods will have a very similar response, so the significance level α must be small enough to differentiate between classifying an image as correct or incorrect.

The test is carried out on the databases i-Scan3 and i-Scan1 without staining the mucosa using significance level $\alpha = 0.01$ with all the off-the-shelf CNNS, all the classical features, and the CNN-05 architecture trained from scratch. The results are presented in Figure 3. It can be observed by the black squares (indicating significantly differences) that, among the pretrained CNNs, in the i-Scan1 database the results are not significantly different and in the i-Scan3 database the CNN-M MCN and GoogleLeNet present the most significantly different results comparing to the other CNNs. It also can be seen that the CNN-05 does not have significantly different results comparing to the other CNNs in the i-Scan1 database and has significantly different results with CNN-M MCN and GoogleLeNet in the i-Scan3 database.

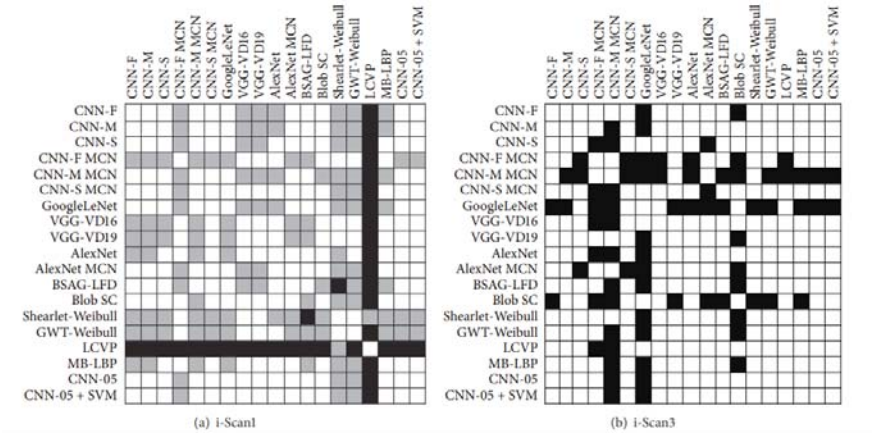


Figure 3. Results of the McNemar test for the i-Scan1 (a) and i-Scan3 (b) databases without staining. A black square in the matrix means that the methods are significantly different with significance level and a grey square in (a) means that the methods are significantly different with significance level . If the square is white then there is no significant difference between the methods.

Also, in Figure 3, when comparing the classical feature extraction methods with the CNNs features it can be seen that there is a quite different response among the results in i-Scan3 database, especially for CNN-M MCN that is significantly different from all the classical methods with the exception of the Shearlet-Weibull method.

The CNN-05 and CNN-05 + SVM did not present significantly different results with the classical features (except with LCVP in i-Scan1 database) and with the pretrained CNNs (except with CNN-M and GoogleLeNet in i-Scan3 database). Likewise, the methods with high accuracy in the i-Scan3 database (BSAG-LFD, VGG-VD16, and VGG-VD19) are not found to be significantly different.

In the i-Scan1 database, with the significance level $\alpha = 0.05$, the results are not significantly different in general (except for LCVP features). However, with the significance level $\alpha = 0.01$, the significance results represented by the grey squares in Figure 3(a) show that the two databases presented different correlation between methods which means that it is difficult to predict a good feature extractor that can satisfy both databases at the same time.

Observing the methods that presented significantly different results in Figure 3 and with good results in Table 8 we decided to produce a feature

level fusion in the feature vectors concatenating them to see if the features can complement each other. It can be seen in Figure 3 that the two most successful CNNs CNN-M MCN and VGG-VD16 are significantly different from each other in both databases and the feature level fusion of these two vectors improve the results from 89.74% and 88.59%, respectively, to an accuracy of 90.58% in average as can be seen in Table 8 (Fusion 5/8).

In Figure 3(b) it can also be observed that the results from CNN-M MCN are significantly different to the classical features BSAG-LFD in the i-Scan3 database. With the feature level fusion of these two features the accuracy increases to 91.03% on average.

Concatenating the three feature vectors (CNN-M MCN, VGG-VD16, and BSAG-LFD) leads to an even better accuracy: 93.22%. It is interesting to note that in both databases the results from CNN-M MCN and VGG-VD16 are significantly different.

Besides that, BSAG-LFD results are significantly different to VGG-VD16 in database i-Scan1. Furthermore, BSAG-LFD results are significantly different to CNN-M MCN in database i-Scan3 which can explain the improvement in the feature level fusion between these three methods.

Making the fusion with these two off-the-shelf CNNs (CNN-M MCN and VGG-VD16) to other classical feature vectors also increases the accuracy as it can be seen in Table 8 (Fusion 5/8/14 and Fusion 5/8/15).

When we add to the vector Fusion 5/8/12 one more classical feature (MB-LBP) that is also significantly different to CNN-M MCN in database i-Scan3 and at the same time significantly different to BSAG-LFD in database i-Scan1, the result outperforms all the previous approaches: 93.55% as it can be seen in Table 8.

In Figure 4 we present some example images from the classification results of all the methods used in the McNemar test with the higher agreement for each prediction outcome.

The percentage above each image shows the average classification rate of the prediction. For example, in the i-Scan1 database and i-Scan3 database (Figures 4(a) and 4(b)), the two images presented in the true positive box were classified as such in all classifiers. However, from i-Scan3 database, in the case of the false negative box, one image had 44% of misclassification and another 15% of misclassification in average.

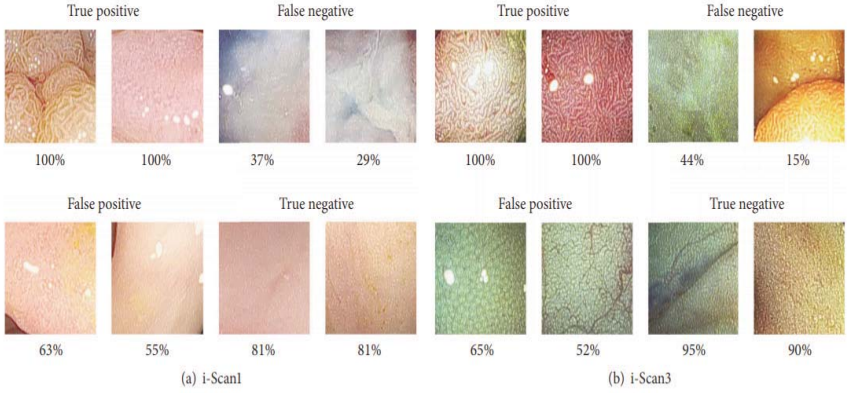


Figure 4. Example results of the classification in agreement from the methods tested in the McNemar test for each prediction outcome.

Comparing the results from all off-the-shelf CNNs and classical features with the CNN-05 trained from scratch using the databases i-Scan1 and i-Scan3 in Table 8 it can be observed that the full training CNN outperformed the results obtained by the classical features and some of the pretrained CNNs. This approach can be considered an option for automatic colonic polyp classification, although the training time and processing complexity are not worthwhile if comparing to the off-the-shelf features.

CONCLUSION

In this work, we propose to explore Deep Learning and Transfer Learning approach using Convolutional Neural Networks (CNNs) to improve the accuracy of colonic polyp classification based on the fact that databases containing large amounts of annotated data are often limited for this type of research. For the training of CNNs from scratch, we explore data augmentation with image patches to increase the size of the training database and consequently the information to perform the Deep Learning. Different architectures were tested to evaluate the impact of the size and number of filters in the classification as well as the number of output units in the fully connected layer.

We also explored and evaluated several different pretrained CNNs architectures to extract features from colonoscopy images by knowledge transfer between natural and medical images providing what is called off-

the-shelf CNNs features. We show that the off-the shelf features may be well suited for the automatic classification of colon polyps even with a limited amount of data.

Besides the fact that the pretrained CNNs were trained with natural images, the 4096 features extracted from CNN-M MCN and VGG-16 provided a good feature descriptor of colonic polyps. Some reasons for the success of the classification include the training with a large range of different images providing a powerful extractor joining the intrinsic features from the images such as color, texture, and shape in the same architecture reducing and abstracting these features in just one vector. Also, the combination of classical features with off-the-shelf features yields the best prediction results complementing each other. It can be concluded that Deep Learning using Convolutional Neural Networks is a good option for colonic polyp classification and the use of pretraining CNNs is the best choice to achieve the best results being improved by feature level fusion with classical features. In future work we plan to use this strategy to also test the detection of colonic polyps directly into video frames and evaluate the performance in real time applications as well as to use this strategy in other endoscopic databases such as automatic classification of celiac disease.

ACKNOWLEDGMENTS

This research was partially supported by CNPq, Brazil, for Eduardo Ribeiro under Grant no. 00736/2014-0.

REFERENCES

1. J. Bernal, J. Sánchez, and F. Vilariño, "Towards automatic polyp detection with a polyp appearance model," *Pattern Recognition*, vol. 45, no. 9, pp. 3166–3182, 2012.
2. Y. Wang, W. Tavanapong, J. Wong, J. H. Oh, and P. C. de Groen, "Polyp-alert: near real-time feedback during colonoscopy," *Computer Methods and Programs in Biomedicine*, vol. 120, no. 3, pp. 164–179, 2015.
3. S. Ameling, S. Wirth, D. Paulus, G. Lacey, and F. Vilarino, "Texture-based polyp detection in colonoscopy," in *Bildverarbeitung für die Medizin 2009, Informatik Aktuell*, pp. 346–350, Springer, Berlin, Germany, 2009.
4. S. Y. Park, D. Sargent, I. Spofford, K. G. Vosburgh, and Y. A-Rahim, "A colon video analysis framework for polyp detection," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 5, pp. 1408–1418, 2012.
5. W. Yi, W. Tavanapong, J. Wong, J. Oh, and P. C. de Groen, "Part-based multiderivative edge cross-sectional profiles for polyp detection in colonoscopy," *IEEE Journal of Biomedical and Health Informatics*, vol. 18, no. 4, pp. 1379–1389, 2014.
6. N. Tajbakhsh, S. R. Gurudu, and J. Liang, "Automated polyp detection in colonoscopy videos using shape and context information," *IEEE Transactions on Medical Imaging*, vol. 35, no. 2, pp. 630–644, 2016.
7. S. Kudo, S. Hirota, T. Nakajima et al., "Colorectal tumours and pit pattern," *Journal of Clinical Pathology*, vol. 47, no. 10, pp. 880–885, 1994.
8. M. Häfner, R. Kwitt, A. Uhl, A. Gangl, F. Wrba, and A. V{\'e} csei, "Feature extraction from multi-directional multi-resolution image transformations for the classification of zoom-endoscopy images," *Pattern Analysis and Applications*, vol. 12, no. 4, pp. 407–413, 2009.
9. M. Häfner, M. Liedlgruber, A. Uhl, A. Vécsei, and F. Wrba, "Delaunay triangulation-based pit density estimation for the classification of polyps in high-magnification chromo-colonoscopy," *Computer Methods and Programs in Biomedicine*, vol. 107, no. 3, pp. 565–581, 2012.
10. S. Kato, K. I. Fu, Y. Sano et al., "Magnifying colonoscopy as a non-biopsy technique for differential diagnosis of non-neoplastic and

neoplastic lesions," *World Journal of Gastroenterology*, vol. 12, no. 9, pp. 1416–1420, 2006.

11. S. Gross, S. Palm, J. Tischendorf, A. Behrens, C. Trautwein, and T. Aach, *Automated Classification of Colon Polyps in Endoscopic Image Data*, SPIE, Bellingham, Wash, USA, 2012.
12. M. Häfner, M. Liedlgruber, A. Uhl, A. Vécsei, and F. Wrba, "Color treatment in endoscopic image classification using multi-scale local color vector patterns," *Medical Image Analysis*, vol. 16, no. 1, pp. 75–86, 2012.
13. M. Häfner, M. Liedlgruber, and A. Uhl, "Colonic polyp classification in high-definition video using complex wavelet-packets," in *Bildverarbeitung für die Medizin 2015*, Informatik Aktuell, pp. 365–370, Springer, Berlin, Germany, 2015.
14. M. Häfner, A. Gangl, M. Liedlgruber, A. Uhl, A. Vécsei, and F. Wrba, "Pit pattern classification using extended local binary patterns," in *Proceedings of the 9th International Conference on Information Technology and Applications in Biomedicine (ITAB '09)*, pp. 1–4, Larnaca, Cyprus, November 2009.
15. M. Häfner, A. Uhl, A. Vécsei, G. Wimmer, and F. Wrba, "Complex wavelet transform variants and discrete cosine transform for scale invariance in magnification-endoscopy image classification," in *Proceedings of the 10th IEEE International Conference on Information Technology and Applications in Biomedicine (ITAB '10)*, pp. 1–5, Corfu, Greece, November 2010.
16. Y. Yuan and M. Q.-H. Meng, "A novel feature for polyp detection in wireless capsule endoscopy images," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '14)*, pp. 5010–5015, Chicago, Ill, USA, September 2014.
17. T. Stehle, R. Auer, S. Gros et al., "Classification of colon polyps in NBI endoscopy using vascularization features," in *Medical Imaging 2009: Computer-Aided Diagnosis*, N. Karssemeijer and M. L. Giger, Eds., vol. 7260 of *Proceedings of SPIE*, Orlando, Fla, USA, February 2009.
18. G. Wimmer, T. Tamaki, J. J. W. Tischendorf et al., "Directional wavelet based features for colonic polyp classification," *Medical Image Analysis*, vol. 31, pp. 16–36, 2016.
19. A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: an astounding baseline for recognition," in *Proceedings*

of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW '14), pp. 512–519, Columbus, Ohio, USA, June 2014.

20. K. Alex, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the Advances in Neural Information Processing Systems 25 (NIPS '12)*, pp. 1097–1105, Curran Associates, Denver, Colo, USA, 2012.
21. H. Shin, H. R. Roth, M. Gao et al., "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.
22. K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: delving deep into convolutional nets," in *Proceedings of the 25th British Machine Vision Conference (BMVC '14)*, Nottingham, UK, September 2014.
23. J. Guo and S. Gould, "Deep CNN ensemble with data augmentation for object detection," https://arxiv.org/abs/1506.07224
24. M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proceedings of the 27th IEEE Conference on Computer Vision and Pattern Recognition (CVPR '14)*, pp. 1717–1724, Columbus, Ohio, USA, June 2014.
25. Y. Bar, I. Diamant, L. Wolf, S. Lieberman, E. Konen, and H. Greenspan, "Chest pathology detection using deep learning with non-medical training," in *Proceedings of the IEEE 12th International Symposium on Biomedical Imaging (ISBI '15)*, pp. 294–297, April 2015.
26. B. Van Ginneken, A. A. A. Setio, C. Jacobs, and F. Ciompi, "Off-the-shelf convolutional neural network features for pulmonary nodule detection in computed tomography scans," in *Proceedings of the 12th IEEE International Symposium on Biomedical Imaging (ISBI '15)*, pp. 286–289, Brooklyn, NY, USA, April 2015.
27. N. Tajbakhsh, S. R. Gurudu, and J. Liang, "Automatic polyp detection in colonoscopy videos using an ensemble of convolutional neural networks," in *Proceedings of the 12th IEEE International Symposium on Biomedical Imaging (ISBI '15)*, pp. 79–83, New York, NY, USA, April 2015.

28. D. C. Cireşan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Mitosis detection in breast cancer histology images with deep neural networks," in *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2013*, K. Mori, I. Sakuma, Y. Sato, C. Barillot, and N. Navab, Eds., vol. 8150 of *Lecture Notes in Computer Science*, pp. 411–418, Springer, Berlin, Germany, 2013.
29. D. C. Cireşan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Deep neural networks segment neuronal membranes in electron microscopy images," in *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS '12)*, pp. 2843–2851, December 2012.
30. N. Tajbakhsh, M. B. Gotway, and J. Liang, "Computer-aided pulmonary embolism detection using a novel vessel-aligned multi-planar image representation and convolutional neural networks," in *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015: 18th International Conference, Munich, Germany, October 5–9, 2015, Proceedings, Part II*, vol. 9350 of *Lecture Notes in Computer Science*, pp. 62–69, Springer, Berlin, Germany, 2015.
31. H. R. Roth, L. Lu, A. Seff et al., *A New 2.5D Representation for Lymph Node Detection Using Random Sets of Deep Convolutional Neural Network Observations*, Springer International, Cham, Switzerland, 2014.
32. R. Zhu, R. Zhang, and D. Xue, "Lesion detection of endoscopy images based on convolutional neural network features," in *Proceedings of the 8th International Congress on Image and Signal Processing (CISP '15)*, pp. 372–376, Shenyang, China, October 2015.
33. H. Roth, J. Yao, L. Lu, J. Stieger, J. Burns, and R. Summers, Detection of sclerotic spine metastases via random aggregation of deep convolutional neural network classifications, CoRR, abs/1407.5976, 2014
34. N. Tajbakhsh, J. Y. Shin, S. R. Gurudu et al., "Convolutional neural networks for medical image analysis: full training or fine tuning?" *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1299–1312, 2016.
35. N. Tajbakhsh, S. R. Gurudu, and J. Liang, "A comprehensive computer-aided polyp detection system for colonoscopy videos," in *Proceedings of the 24th International Conference on Information Processing in Medical Imaging (IPMI '15)*, pp. 327–338, Sabhal Mor Ostaig, Isle of Skye, UK, June-July 2015.

36. N. Hatipoglu and G. Bilgin, "Classification of histopathological images using convolutional neural network," in *Proceedings of the 4th International Conference on Image Processing Theory, Tools and Applications (IPTA '14)*, pp. 1–6, October 2014.

37. Y. Zou, L. Li, Y. Wang, J. Yu, Y. Li, and W. J. Deng, "Classifying digestive organs in wireless capsule endoscopy images based on deep convolutional neural network," in *Proceedings of the IEEE International Conference on Digital Signal Processing (DSP '15)*, pp. 1274–1278, IEEE, Singapore, July 2015.

38. J. S. Yu, J. Chen, Z. Q. Xiang, and Y. X. Zou, "A hybrid convolutional neural networks with extreme learning machine for WCE image classification," in *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO '15)*, pp. 1822–1827, IEEE, Zhuhai, China, December 2015.

39. E. Ribeiro, A. Uhl, and M. Häfner, "Colonic polyp classification with convolutional neural networks," in *Proceedings of the IEEE 29th International Symposium on Computer-Based Medical Systems (CBMS '16)*, pp. 253–258, Dublin, Ireland, June 2016.

40. A. Prasoon, K. Petersen, C. Igel, F. Lauze, E. Dam, and M. Nielsen, "Deep feature learning for knee cartilage segmentation using a triplanar convolutional neural network," in *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2013—16th International Conference, Nagoya, Japan, September 2013, Proceedings, Part II*, pp. 246–253, Springer, 2013.

41. F. Ciompi, B. de Hoop, S. J. van Riel et al., "Automatic classification of pulmonary peri-fissural nodules in computed tomography using an ensemble of 2D views and a convolutional neural network out-of-the-box," *Medical Image Analysis*, vol. 26, no. 1, pp. 195–202, 2015

42. J. Arevalo, F. A. Gonzalez, R. Ramos-Pollan, J. L. Oliveira, and M. A. Guevara Lopez, "Convolutional neural networks for mammography mass lesion classification," in *Proceedings of the 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC '15)*, pp. 797–800, IEEE, Milan, Italy, August 2015.

43. S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

44. M. Häfner, A. Uhl, and G. Wimmer, "A novel shape feature descriptor for the classification of polyps in HD colonoscopy," in *Medical Computer Vision. Large Data in Medical Imaging*, B. Menze, G. Langs, A. Montillo, M. Kelm, H. Müller, and Z. Tu, Eds., vol. 8331 of *Lecture Notes in Computer Science*, pp. 205–213, Springer, Berlin, Germany, 2014.

45. M. Häfner, T. Tamaki, S. Tanaka, A. Uhl, G. Wimmer, and S. Yoshida, "Local fractal dimension based approaches for colonic polyp classification," *Medical Image Analysis*, vol. 26, no. 1, pp. 92–107, 2015.

46. H. R. Roth, L. Lu, J. Liu et al., "Improving computer-aided detection using convolutional neural networks and random view aggregation," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1170–1181, 2015.

47. M. Ganz, X. Yang, and G. Slabaugh, "Automatic segmentation of polyps in colonoscopic narrow-band imaging data," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 8, pp. 2144–2151, 2012.

48. A. Coates, H. Lee, and A. Y. Ng, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the 4th International Conference on Artificial Intelligence and Statistics (AISTATS '11)*, vol. 15 of *JMLR*, pp. 215–223, JMLR W&CP, Fort Lauderdale, Fla, USA, 2011.

49. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," CoRR, abs/1409.1556, 2014

50. A. Vedaldi and K. Lenc, "Matconvnet—convolutional neural networks for MATLAB," CoRR, abs/1412.4564, 2014

51. P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," CoRR, abs/1312.6229, 2013

52. C. Szegedy, W. Liu, Y. Jia et al., "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '15)*, pp. 1–9, Boston, Mass, USA, June 2015.

53. Y. Dong, D. Tao, X. Li, J. Ma, and J. Pu, "Texture classification and retrieval using shearlets and linear regression," *IEEE Transactions on Cybernetics*, vol. 45, no. 3, pp. 358–369, 2015.

54. S. Liao, X. Zhu, Z. Lei, L. Zhang, and S. Li, "Learning multi-scale block local binary patterns for face recognition," in *Advances in Biometrics*, vol. 4642 of *Lecture Notes in Computer Science*, pp. 828–837, Springer, Berlin, Germany, 2007.

55. M. Häfner, M. Liedlgruber, S. Maimone, A. Uhl, A. Vécsei, and F. Wrba, "Evaluation of cross-validation protocols for the classification of endoscopic images of colonic polyps," in *Proceedings of the 25th IEEE International Symposium on Computer-Based Medical Systems (CBMS '12)*, pp. 1–6, Rome, Italy, June 2012.

56. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Intelligent Signal Processing*, pp. 306–351, IEEE Press, Piscataway, NJ, USA, 2001.

57. Q. McNemar, "Note on the sampling error of the difference between correlated proportions or percentages," *Psychometrika*, vol. 12, no. 2, pp. 153–157, 1947.

CHAPTER 13

DEEP LEARNING ALGORITHM FOR BRAIN-COMPUTER INTERFACE

Asif Mansoor[1], Muhammad Waleed Usman[2], Noreen Jamil[2], and M. Asif Naeem[2]

[1]National University of Sciences and Technology, Islamabad, Pakistan

[2]National University of Computer and Emerging Sciences, Islamabad, Pakistan

ABSTRACT

Electroencephalography-(EEG-) based control is a noninvasive technique which employs brain signals to control electrical devices/circuits. Currently, the brain-computer interface (BCI) systems provide two types of signals, raw signals and logic state signals. The latter signals are used to turn on/off the devices. In this paper, the capabilities of BCI systems are explored, and a survey is conducted how to extend and enhance the reliability and

Citation: Asif Mansoor, Muhammad Waleed Usman, Noreen Jamil, M. Asif Naeem, "Deep Learning Algorithm for Brain-Computer Interface", Scientific Programming, vol. 2020, Article ID 5762149, 12 pages, 2020. https://doi.org/10.1155/2020/5762149.

accuracy of the BCI systems. A structured overview was provided which consists of the data acquisition, feature extraction, and classification algorithm methods used by different researchers in the past few years. Some classification algorithms for EEG-based BCI systems are adaptive classifiers, tensor classifiers, transfer learning approach, and deep learning, as well as some miscellaneous techniques. Based on our assessment, we generally concluded that, through adaptive classifiers, accurate results are acquired as compared to the static classification techniques. Deep learning techniques were developed to achieve the desired objectives and their real-time implementation as compared to other algorithms.

INTRODUCTION

Background

Brain-computer interface (or BCI) is basically setting up a connection between the human brain and the computer device to control or to perform certain activity using brain signals. These brain signals are translated as an action for a device. The interface thus provides a one-to-one communication pathway between the brain and the target.

The technology has advanced from mechanical devices and touch systems, and now, world is approaching towards use of neural waves as the input. Even though it is not widely applied for now, it has a promising future. Especially for the physically impaired people who face difficulties in performing physical activities and lose their brain signal to move their muscles, it is the only way to function.

A BCI system includes a device with electrodes that act as sensors and measure brain signals, an amplifier to raise the weak neural signals, and a computer which decodes the signals into controlling signals to operate devices. Mostly, the BCI device is a headset which is portable and wearable.

The BCI device has two functions. Firstly, it records the data reviewed at its electrodes, and secondly, it interprets or decodes neural signals.

Nervous system resembles an electrical system which passes nerve impulses as a message. This means neurons (brain cells) communicate by transmitting and receiving very small electrical waves, merely in range of microvolts. Now, to sense and record these signals, we require precise and advanced sensors.

Electrodes are set directly on the scalp or embedded in the brain which requires surgical procedure. The nonsurgical method of electrode placement though does not damage the brain, it yields poor-quality brain signals. Those that are recorded directly from the scalp yield better results but at the risk of surgery that may induce damage in the brain. The risk of damaging brain tissues exceeds the quality obtained through the surgical method. BCI is therefore a better pathway for neurorehabilitation for paralyzed people. Apart from these, other techniques include functional MRI (fMRI) and magnetoencephalography (MEG). fMRI maps brain activity with an MRI scanner, while MEG is a brain imaging process that identifies brain activity. Electric currents flowing through the brain produce magnetic field, and these are sensed by highly sensitive magnetometers. Both fMRI and MEG techniques use large and expensive machines. Another noninvasive methodology is near-infrared spectroscopy (NIRS). In this process, neural signals are recorded by passing NI light through the head. The quality of the brain activity measurement is not adequate for the brain computer interface.

In case of healthy people, the brain transmits signals from the central nervous system to the muscles, and thus, they can move the muscles of the body. However, in case of people suffering from stroke or neuromuscular illness, the transmission of signals between the brain and the rest of body muscles is distorted. The patient's body becomes paralyzed or losses the capability to control muscle movement, like cerebral palsy. It is observed that a patient may not be able to move a muscle, but a brain can transmit the neural signal. This means that the neural signal is transmitted from the CNS but not received by target muscles. A BCI can be designed to utilize those commands to control devices or computer programs.

Each part of the body is controlled by a particular part of the brain as shown in the figure. Using BCI techniques, it is observed which part of the brain is active and transmitting the signal. Through this, the BCI system can predict the muscle locomotion from the brain activity [1].

BCI systems can be advanced, and multiple new applications can be developed using a fact that a variety of other brain activities can also be recognized. For instance, while one performs a numeric calculation, the frontal lobe is activated, and when one comprehends a language, Wernicke's area is activated.

Currently, numerous groups are contributing to the evolution of BCIs so as to develop numerous applications, specific for each category of the consumer. Each day, scientists and engineers are improving algorithms, BCI

sensor devices, and techniques for quality attainment of data and improved accuracy of systems.

The problem is which method is optimal to analyze these complex time-varying neural responses and map them accordingly to the output response desired. These signals are merely in the range of microvolts. So, these electrical signals are passed through several processes to remove noise and to gather useful signals. Next, algorithms and classification techniques are applied to the data obtained [2].

Preliminaries

To attain a better understanding of BCI systems and the processes that undergo within them, an explanation of the terminologies and the said processes is presented as follows.

Brain Waves

Brain waves are oscillating voltages bearing amplitudes from microvolts to some millivolts; there are 5 widely known brain waves bearing different frequency ranges exhibiting states of the brain as shown in Table 1 [3].

Table 1. Brain waves and associated frequencies

Frequency band	Frequency	Brain states
Gamma	>35 Hz	Concentration
Alpha	12–35 Hz	Anxiety, relaxed, external attention
Beta	8–12 Hz	Very relaxed, passive attention
Theta	4–8 Hz	Deeply relaxed, inward focus
Delta	0.5–4 Hz	Sleep

Brain Activity Recording Methods for the BCI

The neural activity of the brain can be analyzed and understood based on the recording methods used. Recording methods of the BCI can be categorized as follows:

(1) *Invasive Recording Techniques*. Invasive recording methods are those in which the electrodes are inserted deep in the brain using surgical methods, and the quality of the signal generated is better as compared to its noninvasive counterpart; however, issues arise from long-term stability, and protection is required to hinder them from creating infections. One such example is electrocorticography (ECoG), which measures the brain activity from the neural cortex.

(2) *Noninvasive Recording Techniques*. Noninvasive techniques do not require any surgical treatment and thus safe from causing any sort of infections; though their signal quality is low, it is still a popular means of brain signal acquisition.

These techniques include electroencephalography (EEG) in which the electrical activity is recorded from the scalp of the brain and magnetoencephalography (MEG) in which magnetic properties exhibited due to the difference in oxygenated and deoxygenated hemoglobin are recorded.

For our project, we will opt for an EEG-based signal recording technique and explain its characteristics in the following [1].

Electroencephalography (EEG)

Introduced by Hans Berger in 1929, EEG is a measurement of voltage levels that underlines the activity of the brain in response to an event or a stimulus. EEG method comprises electrodes placed on the scalp of the brain at different locations as specified in Figure 1 with temporary glue. The electric signals are generated due to the ionic content present in the brain consisting of Na+, Ca++, K+, and Cl− ions; the transportation of these ions invokes the electric potential used in EEG.

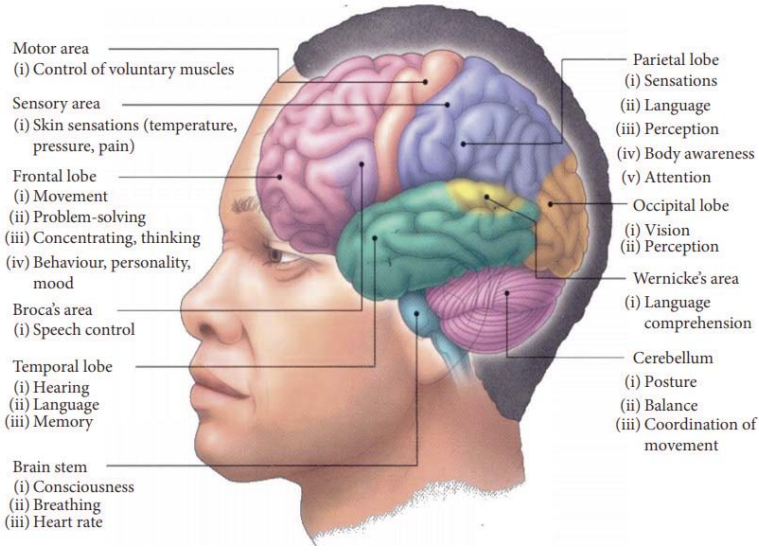


Figure 1. Brain anatomy.

The EEG signals are of low quality because of different layers of tissues between the EEG cap and the signal source as shown in Figure 2. The potential created is in a range of tens of microvolts, and these electrodes need to have powerful amplifiers in order to acquire meaningful signals.

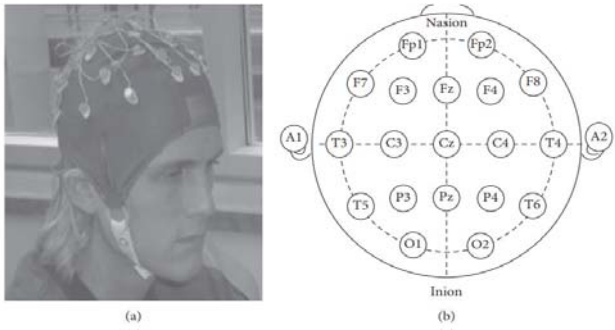


Figure 2. EEG: (a) subject wearing a 32-electrode EEG cap; (b) standardized electrode placements.

Need of BCI

Brain-computer interface-based technology is a developing field, and it has been under focus by many industries to innovate and make everyday life tasks easier. One of the questions which arises in the mind is why we need BCI systems? BCI system is a complex technology, no doubt, however, leading to a simpler life.

Following are the main reasons why we need to focus on this technology:(i) Control of devices can be made easy through just our thoughts(ii)Making a decision and then performing a task takes time, while operating a device using thoughts or technically our brain waves is easier(iii)Re-establishing communication path from the brain to artificial limbs and assisting those affected by brain-related diseases

Individuals in Need of a BCI to Re-Establish Motor Control and Communication

A wide range of neurological diseases such as motor neuron diseases and spinal cord injury may lead to severe paralysis of the motor muscles, restricting the patient to control artificial devices through only a few muscles and thereby termed as “locked-in,” while people who have completely lost their motor control are termed as “completely locked-in.”

Evident that the normal communication channel from the brain to the limbs is lost, BCI is used to re-establish the communication through an alternative route.

Even being applicable to a healthy person, BCI systems can be used to employ numerous tasks from the users using the signals generated from the brains to control applications as presented in the following [4]:

(1) *Noninvasive Brain-Computer Interface Research at the Wadsworth Center*. The research conducted at the Wadsworth Center was to study different approaches employed in the BCI to control a computer screen cursor to analyze their advantages and disadvantages; one approach was sensory-based rhythm control in which the selected features in the frequency domain were based on the potentials created by motor imagery and linear regression was employed so that they can be converted as control signals to move the cursor.

The other procedure was the P300-based cursor control in which the user focuses attention on the desired symbol and is provided with a $6 * 6$matrix to produce time-varying stimuli and linear regression is utilized to allow these signals as a control input to move the cursor.

The research suggested that the BCI is an application-oriented approach and depends entirely on user training; the EEG features dictate the BCI system for speed, accuracy, bit rate, and usefulness.

Sensorimotor Rhythms (SMR) is an approach employing better results for control tasks such as controlling a screen cursor, while the P300-BCI system was slower as compared to the SMR-BCI.

(2) *The Berlin Brain-Computer Interface: Machine Learning-Based Detection of User-Specific Brain States*. The researchers for the Berlin brain-computer interface employed sensory motor rhythms, i.e., thinking of moving the left hand or right hand and used machine learning-based detection of the user specific brain states. While testing their trained model, they achieved an information transfer rate above 35 bits per minute (bpm), and overall spelling seed was 4.5 letters per minute including correcting the mistakes, using 128-channel EEG and using feedback control for untrained users in order to properly train the machine learning algorithms, thereby reducing the training user time used in the voluntary control approach [2].

Structure of the BCI

The steps of the brain computer interface system include the following:(1) Brain activity measurement/recording methods of the BCI(2)Preprocessing techniques(3)Feature extraction(4)Machine learning implementation/ classification(5)Translation to control signal

Preprocessing

In BCI, preprocessing techniques consist of data acquisition of brain signals to check the signal quality without losing important information; the recorded signals are cleaned and conduct noise removal to acquire relevant information encoded in the signal. As mentioned above, the EEG signals are of poor quality; even the commercial 50 Hz frequency, due to nearby appliances, can corrupt the EEG signals, and the users are also advised not to think anything else apart from the stimuli as presented. In preprocessing, using Fourier transform or Fourier series, the signals are taken into the frequency domain and studied what frequency content is present in the signal. The undesired 60 Hz frequency signal and undesired signal produced by performing actions other than the said stimuli are then filtered out using a notch filter as mentioned in Figure 3.

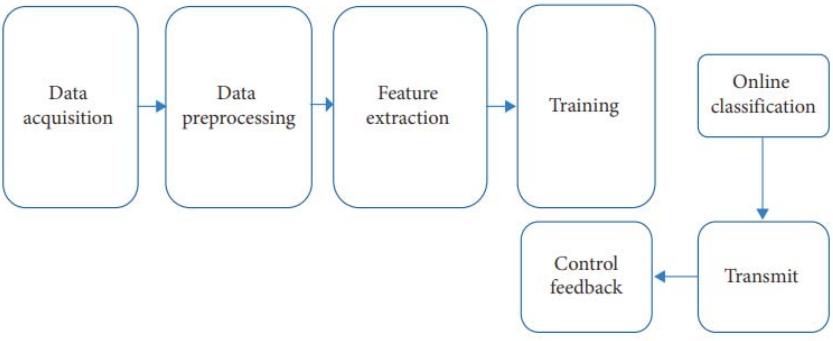


Figure 3. BCI block diagram.

Feature Extraction

Feature extraction plays a vital role in brain-computer interface applications; the raw EEG signals are nonstationary signals that are corrupted by noise or due to artifacts present in the environment where they are being recorded, but still meaningful information can be extracted from them. The data

dimensionality is also reduced to process it better, and machine learning models are applied. This method is essential to increase the classification accuracy of the BCI system.

EEG signal is a time-domain nonstationary signal, and the relevant information such as signal energy is analyzed as a function of time or frequency later; relevant statistic measures are adapted to properly explain the characteristics of the signal.

Some of the commonly used feature extraction techniques are listed as follows.

Short-time Fourier transform is a frequency-domain feature extraction technique in which the EEG signal is convolved with a window function to extract the relevant frequency features of the brain which are broken down as sinusoids at different frequency ranges.

Mathematically, it is represented as

$$Xstft[m,n] = \sum_{k=0}^{L-1} x[k]w[n-k]e^{-(j2\pi nk/L)}, \quad (1)$$

where $x[k]$ is the input EEG and w$[n-k]$ is the window multiplied to extract the frequency features as shown in the figure [4].

Discrete-time and continuous-time wavelet transform is a time frequency-based feature extraction technique that allows better temporal and spatial resolution in which the EEG signals are produced in the form of wavelets at different frequency ranges of interest as shown in Figure 4.

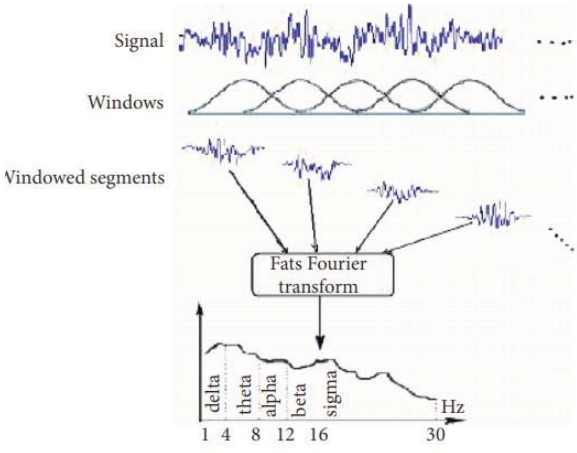


Figure 4. Feature extraction using short-time Fourier transform.

The technique of wavelet transforms as adapted in the literature is a filtering continuous application of high-pass and low-pass filters to extract the wavelets of the signal which, when added together, constitute the original signal and downsampled by a factor of 2 as shown in Figure 5.

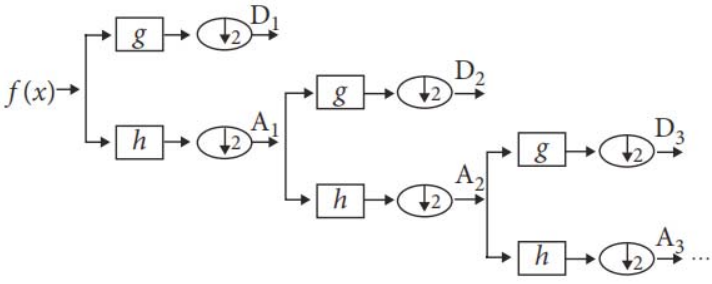


Figure 5. Wavelet transform.

g and *h* are consecutive high-pass and low-pass filters which produce the relevant detailed and approximated coefficients of the EEG signals [5].

Neural Networks

Before starting to explain what deep learning is, it is first beneficial to explain the role of deep learning and its fundamental blocks.

Deep learning is a classification tool used in a variety of daily applications which is composed of speech recognition and computer vision to natural language processing in the context of the BCI; the input features which are different brain frequency bands are classified according to what activity the user is performing at the moment.

Neural Network. A neural network is a model similar to that of a neuron in our brain that has input nodes and output nodes; the mathematical model for a neural network is given by the following equation:

$$v = wx + b, \tag{2}$$

where v is the weighted sum of the inputs and the bias term which will be fed at the output node, *b* is a bias term which is mostly set to 1, and w is the random weights assigned that are multiplied with the input in order to reach closer to the desire output.

The neural network is shown in Figure 6.

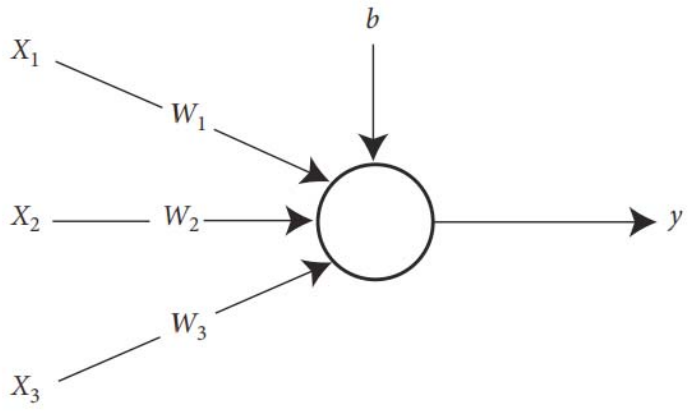


Figure 6. Neural network.

These calculations are often preceded in the form of matrices; the input, the weight terms, the output, and the bias are as follows:

$$v = (w_1 \times x_1) + (w_2 \times x_2) + (w_3 \times x_3) + b. \tag{3}$$

Finally, the output node is passed to an activation function and provides the final output; the activation function calculates the characteristic of the node. The activation function acts to map the corresponding inputs to the right output y present at the output node:

$$y = \varphi(v). \tag{4}$$

The neural network does not get its right output at the first attempt. It needs to be trained a lot, and so a training rule is assigned to neural networks to get the right output. Many training rules are adapted, but one of the most commonly used is the delta rule, and the rule is expressed using the following equation:

$$w_{ij} \longleftarrow w_{ij} + \alpha e_i x_j, \tag{5}$$

where x_j represents the number of inputs, e_i is the error generated at the output node, and α is the learning rule between $(0 < \alpha < 1)$.

The training rule is summarized as follows:(1)Assign adequate values to the weights.(2)Obtain the input from the training data and feed it into the neural network which will give an output d; subtract the output d to obtain the correct output at the output node.

$$e_i = d_i - y_i. \tag{6}$$

(3)Calculate the weight updates:

$$\Delta w_{ij} = \alpha e_i x_j. \tag{7}$$

(4) Adjust the weights accordingly until the correct output or that has small tolerance is obtained:

$$w_{ij} \longleftarrow w_{ij} + w_{ij}. \tag{8}$$

The above explanation was presented for a single-layer neural network; the architecture of neural networks is becoming better with the cost of greater memory, but with higher classification accuracy, we use deep neural networks which are the same as the single-layer neural network but with hidden layers added in between the input and output nodes, as shown in Figure 7.

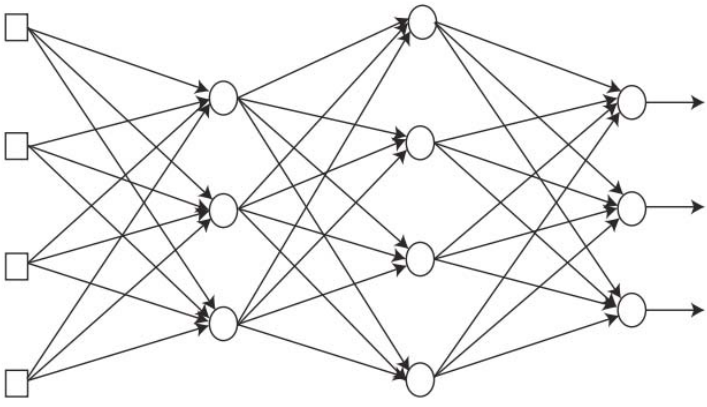


Figure 7. Structure of the deep neural network.

The concepts are similar to those of a single neural network but with the adjustments of added hidden layers and a different training rule because the delta rule has a drawback of not propagating the output to the hidden layers, thereby the weights are not adjusted.

To explain how the deep neural network works, the above explained single neural network is set as basis.

In Figure 8, given a multiple-layered neural network, the weighted sum obtained at the first hidden layer is presented as

$$\begin{bmatrix} v_1^{(1)} \\ v_2^{(1)} \end{bmatrix} = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \triangleq W_1 x. \tag{9}$$

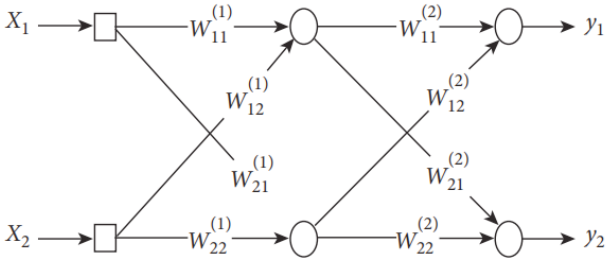


Figure 8. Multilayered network.

The outputs are calculated via the sigmoid activation function:

$$\begin{bmatrix} y_1^{(1)} \\ y_2^{(1)} \end{bmatrix} = \begin{bmatrix} \varphi\left(v_1^{(1)}\right) \\ \varphi\left(v_2^{(1)}\right) \end{bmatrix}. \tag{10}$$

The process is repeated, and the outputs obtained are treated as the inputs to the other nodes, and we get the outputs as

$$\begin{bmatrix} v_1^{(1)} \\ v_2^{(1)} \end{bmatrix} = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} \end{bmatrix} \begin{bmatrix} y_1^{(1)} \\ x_2^{(1)} \end{bmatrix} \triangleq W_1 y^{(1)}. \tag{11}$$

And lastly, the weighted sum is being inserted into the activation function, and we return our final output:

Deep learning training rule is given in the following.

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \varphi(v_1) \\ \varphi(v_2) \end{bmatrix}. \tag{12}$$

Backpropagation algorithm is commonly used as the training instruction for the deep neural networks; the procedure is summarized as follows:

(1) Assign adequate values to the weights.

(2) Take the input from the training data and feed it into the neural network which will give an output d. Subtract the output d to

obtain the correct output at the output node and the delta (δ) of the output nodes:

$$e = d - y,$$
$$\delta = \varphi'(v)e. \tag{13}$$

(3) Propagate the delta back towards the hidden nodes, and determine respective delta δ of nodes:

$$e^k = W^T\delta,$$
$$\delta^{(k)} = \varphi'\left(v^{(k)}\right)e^{(k)}. \tag{14}$$

(4) Repeat until it reaches the input nodes.

(5) Modify the weights according to the rule:

$$\Delta w_{ij} = \alpha\delta_i x_j,$$
$$w_{ij} \longleftarrow w_{ij} + \Delta w_{ij}. \tag{15}$$

(6) These steps are repeated until the neural network is utterly trained as shown in Figure 6.

Now, the alpha-beta ranges are extracted, and consecutive energies are calculated. The features are fed into the deep learning neural network, and using the backpropagation learning rule, the model is trained, yielding an accuracy of 97.83%, as shown in Figure 9 [5].

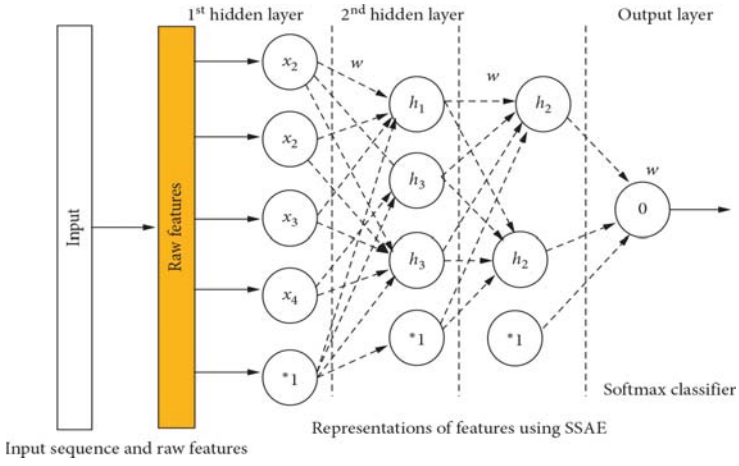


Figure 9. General diagram of neural networks in the BCI.

CRITICAL REVIEW OF THE RELATED LITERATURE

A brain-computer interface involves various stages, and development in each stage leads to an improved and efficient system. Here, the literature review of major steps including data acquisition, feature extraction, classification algorithm, and applications is presented.(1)Improvement of EEG signal acquisition: An electrical aspect for state of the art of front end. Computational intelligence and neuroscience: a research paper published by Ali Bulent Usakli, the NCO Academy, Turkey, presented some applicable concerns for acquiring quality EEG signals which are proven helpful for users and design engineers. One of the most important considerations is selecting suitable electrodes and headset. In the EEG-based BCI, electrodes, signal processing components including mental and environmental conditions, filtration of noise, amplification, signal translation, and data storage affect the recording process. The data acquisition is an important step in any machine learning procedure. Brain signals need to be cleaned and preprocessed so that a good result can be obtained [1].(2)P300 wave detection using Emotiv Epoc+ headset: effects of matrix size, flash duration, and colors: Saleh Ibrahim Alzahrani conducted research on P300 wave detection using the Emotive Epoc+ headset to study the effects of the size of the matrix, flash duration, and colors. In this study, P300 signals were obtained from five subjects with Emotive EPOC+ using all 14 channels. For this research, EEG signals obtained were communicated to software OpenViBE through a USB dongle. A sample was taken every 8 seconds at a rate of 128 samples per second. The configured sampling rate provides ample samples for the four frequency bands, containing significant ERP data. During process of signal recording, the subjects were shown a matrix of 6×6 or 3×3 on the computer screen. They were instructed to stay calm, avoid any needless movement, and set all on letter which they desire to spell. It is reported in the study that one of the advantages of using the Emotiv EPOC+ headset is that it takes merely two to three minutes for preparation as compared to other headsets that take more than ten minutes. The quality of sensors is verified through Emotiv Xavier SDK which provides feedback report of each sensor. To decrease the contact impedance, saline liquid was applied to the sensor surface. Primary objective was to assess the potential of Emotiv EPOC+ to perceive P300 signals. Finding the electrodes proficient at providing target signals helps minimize the number of channels which makes signal processing a lot easier. The results of this experiment provide evidence of the capability of Emotiv EPOC+ to detect the P300 signals from two channels, O1 and O2

[2].(3)Automatic seizure detection in SEEG using high frequency activities in wavelet domain. Medical engineering and physics: in this research paper, the researcher has found the method for detection of seizures using the high-frequency analysis in the wavelet domain. This method is used highly in the high-frequency domain. Because of seizer detection, the method is usually done using high frequency in the range of 80–250 Hz. Also, it can handle fast ripples in the range of 250 to 500 Hz. The biggest advantage in the seizure detection is that it can detect the seizure offset. The methodology consists of the Continuous Wavelet Transform (CWT), which was computed by convolving the SEEG signal which has to make the feature extraction, and it also includes the complex conjugate of the wavelet basis function (Ayoubian, 2013).(4)Classification of epilepsy EEG signals using DWT-based envelope analysis and neural network ensemble: envelope detection is a very efficient method for detecting the impact of the signals which are based on the biological change or diagnosis. In this paper, the researchers used the Hilbert transform which has a good impact on the resultant signals so that the signals are then widespread using the DWH technique which has a unique behavior regarding the biological change; the researchers detected the changes using this method [5].(5)Feature selection for motor imagery EEG classification based on firefly algorithm and learning automata: in this research paper, the researchers implemented spectral linear discriminant analysis for the classification of motor imagery signals. Feature extraction method used was basically common spatial patterns; the advantage of using this feature extraction method is basically of two-class discrimination problems. This maximizes the variance of one class and decreases the variance of the other class, which is the advantage, but the disadvantage is because of the multiclass overlap structure in this method, it is not used for multiclass prediction [6].(6)Unsupervised adaptation of electroencephalogram signal processing based on fuzzy C-means algorithm. International Journal of Adaptive Control and Signal Processing: this research paper presented the techniques of brain mapping with emphasis on multichannel EEG and functional brain imaging techniques. During training and testing, the concentration of the subject on the target object is one of the concerns which signifies the capacity to operate a device. They have used different algorithms such as LDA and fuzzy C-means. Fuzzy C-means is an adaptive classifier, and this is probably used where the device behavior is not synchronized

with the classification model [7].(7)A review of classification algorithms for EEG-based brain-computer interfaces: a 10-year update: this paper is the latest review of the BCI classification techniques; there are some algorithms discussed in this paper taken from different papers, and their accuracies, optimization, the method of feature extraction used were compared, and every algorithm has its own advantages and disadvantages. In this paper, they have used the event-related potentials. Feature extraction method they have used is based on spatial filtering which is the most optimized filter; it can further be optimized by using the calibration [8].(8)Sequential non-stationary dynamic classification with sparse feedback: in this research paper, basically, they have discussed the technique for the classification of the nonstationary and nonlinear signals. As we know that the BCI signals are nonstationary in nature, sparse feedback can be used for the stability of the brain signals and classifying them using the RBF classifiers which are radial basis functions. The signals are acquired in a nonlinear manner, and we can apply linear models to them, but again, multiclass prediction is not able to be performed. This is because of the sparse feedback matrices involved [9].(9) Motor imagery and direct brain-computer communication: Gert Pfurtscheller and Neuper researched about the technique for the motor imaginary signals by the imagination of the left hand, right hand, and foot movements. In the neurofeedback method, real-time prediction of brain signals is difficult to achieve. We have nonlinear signals at the input of the neurofeedback method so that we can use the Hidden Markov Method (HMM) to make predictions in real time but the accuracy is a tradeoff [10].(10)Toward unsupervised adaptation of LDA for brain-computer interfaces. IEEE Transactions on Biomedical Engineering: the firefly algorithm (FA) is an efficient algorithm for selecting the most appropriate subset of features and helps improving accuracy of classification. When the problem of entrapping of FA in the local optimum arises, a procedure for combining the firefly algorithm and learning automata (LA) is proposed which optimizes feature selection for motor imagery EEG. For the expected outcome of the high-dimensional feature set, a process of combining the common spatial pattern (CSP) and local characteristic-scale decomposition (LCD) algorithms is used. It is further classified by the use of the spectral regression discriminant analysis (SRDA) classifier [11].

COMPARISON OF CLASSIFICATION ALGORITHMS

Table 2 shows the comparison of classification algorithms.

Table 2. Comparison of classification algorithms

Title	Algorithm	Input features	Efficiency	Advantages	Drawbacks/tradeoff
Vidaurre et al. [12]. Toward unsupervised adaptation of LDA for brain-computer interfaces. IEEE transactions on biomedical engineering, 587–597.	Linear discriminant analysis uses hyperplanes for different classes, assuming normal distribution, with equal covariance matrix for both classes; to solve an NC class problem, several hyperplanes are used.	Separating hyperplane is obtained by seeking the projection that maximizes the distance between two classes' means and minimizes the interclass variance.	Suitable for online BCI and provides generally good result, and fluctuations in the training data set do not affect much.	Very low computational requirement so suitable for online BCI system.	Linearity that can provide poor results on complex nonlinear EEG data (not immune to noise).
Li et al. [5]. Classification of epilepsy EEG signals using DWT-based envelope analysis and neural network ensemble. Biomedical signal processing and control, 357–365.	Support vector machine uses a support vector hyperplane to identify classes.	Selected hyperplane is the one that maximizes margins, i.e., the distance from nearest training points.	Enables classification using linear decision boundaries, (linear SVM) has been applied, generalization capabilities, to be insensitive to overtraining and to the curse of dimensionality.	Maximizing margins and regularization are known to increase the accuracy.	These advantages are gained at the expense of a low speed of execution.
Lotte et al. [4]. A review of classification algorithms for EEG-based brain-computer interfaces: a 10-year update. Journal of neural engineering, 031005.	Neural network consists of at least three layers of nodes. Except for the input nodes, each node is a neuron that uses a nonlinear activation function which utilizes a supervised learning technique called backpropagation for training.	Neurons of the output layer determine the class of the input feature vector.	Applied to almost all BCI applications.	Because universal approximators are composed of enough neurons and hidden layers, they can approximate and classify any continuous signal.	Sensitive to overtraining, especially with such noisy and nonstationary data as EEG; therefore, careful architecture selection and regularization is required.
Usakli [1]. Improvement of EEG signal acquisition: An electrical aspect for state of the art of front end. Computational intelligence and neuroscience.	Consists of the noninvasive technique for recording brain signals which is based on the electromagnetic resonance signals compared to that of the EEG scalp signals.	Brain signals as an electrical pulse coming from the brain.	Suitable for all BCI systems. They are based on the noninvasive technique in real-time monitoring of signals.	Proven helpful for users and design engineers. One of the most important considerations is selecting suitable electrodes and headset.	Costly design because of the gold electrodes. They are costly, and everyone cannot afford that system.
Alzahrani [2]. P300 wave detection using Emotive Epoc+ headset: Effects of matrix size, flash duration,	Emotive Epoc+ 14 channel sensor was used which has 14 channels for EEG and a neutral channel as well.	Input features consist of P300 steady-state evoked potentials.	Suitable for the brain signals which are collected by the Emotive Epoc+ sensor. The signals are carried out using Emotive Epoc.	Advantage is basically being optimized, and the device is very cheap in price with affordable accuracy.	For more number of class predictions, the accuracy becomes low, and the output is affected.

Title	Algorithm	Input features	Efficiency	Advantages	Drawbacks/tradeoff
Ayoubian, L. a. (2013). Automatic seizure detection in SEEG using high frequency activities in wavelet domain. Medical engineering and physics, 35, 319–328.	Based on the continuous wavelet transform, the brain signals were computed by convolving the SEEG signal.	Brain signals were collected from the Stellate Harmonie system for EEG monitoring purpose. These signals were passed through a band-pass filter.	Suitable for the detection of the seizure. A seizure onset is added to the signals and then compared with the normal brain signal.	For automatic seizure detection, it is very useful, and it can be used for the patients who are not able to calculate when they have seizure.	The disadvantage is basically for some high-frequency seizures. The high-frequency seizures are not detected easily because of the band-pass filter.
Liu et al. [6]. Feature selection for motor imagery EEG classification based on firefly algorithm and learning automata. Sensors.	Spectral regression discriminant analysis (SRDA) is widely used in the feature classification; in this paper, they have implemented this algorithm.	Separating hyperplane is obtained by seeking the projection that maximizes distance between two classes' means and minimizes the interclass variance.	Suitable for online BCI and provides generally good result, and fluctuations in the training data set do not affect much.	Very low computational requirement so suitable for online BCI system, simple to use, and generally provides good results.	Linearity that can provide poor results on complex nonlinear EEG data (not immune to noise).
Lowne et al. [9]. Sequential non-stationary dynamic classification with sparse feedback. Pattern recognition, 897--905.	Spectral regression discriminant analysis (SRDA) is widely used in the feature classification; in this paper, they have implemented this algorithm.	Separating hyperplane is obtained by seeking the projection that maximizes distance between two classes' means and minimizes the interclass variance.	Suitable for online BCI and provides generally good result, and fluctuations in the training data set do not affect much.	Very low computational requirement so suitable for online BCI system, simple to use, and generally provides good results.	Linearity that can provide poor results on complex nonlinear EEG data (not immune to noise).
Liu et al. [6]. Unsupervised adaptation of electroencephalogram signal processing based on fuzzy C-means algorithm. International journal of adaptive control and signal processing.	Common spectral patterns were used for the feature extraction and the linear discriminant analysis, and fuzzy C-means was used for the feature classification.	The maximum distance was calculated for each fuzzy C-means, and then the mean was calculated; after that, the features were classified.	They are suitable for nonlinear EEG signals having different amplitudes for different people.	Very low computational requirement so suitable for online BCI system, simple to use, and generally provides good results.	Fuzzy behavior can be seen in the output when the frequency changes at the input.
Pfurtscheller and Neuper [10]. Motor imagery and direct brain-computer communication. Proceedings of the IEEE, 1123–1134.	Hidden Markov model was used for the classification of EEG signals as they are nonlinear in nature, so we can tune the Markov model accordingly.	The input signals were obtained from the two channels, and these signals were transformed into the HMM network.	For two channels, EEG signals, this is good, and it has a fast classification.	The method used in this paper uses low computational power, and the model functions are optimized.	Output accuracy depends on the linear behavior of the signals. When the frequency fluctuates the output, control will also change.

DISCUSSION

There is a large range of classifiers developed by scientists and engineers around the world. These classification algorithms can be divided into four groups.

Adaptive Classifiers

Adaptive classifiers are listed as those classifiers in which parameters are progressively recalculated and also updated with procurement of new EEG data signals which are nonstationary, and adaptive classifiers are capable to follow the change in the feature distribution.

As mentioned in [10], a model for a motor-imagery-based self-paced BCI structure for operating a robot was proposed. They used a basic synchronous BCI, devised earlier for recording data for offline training classification before conducting the online self-paced procedure. They extracted logarithmic band power as features, and features were extracted from EEG signals. Feature selection was manual so as to gain quality frequency bands. To extract the features, chosen frequency bands were digitally bandpass-filtered, squared, and averaged over 1 second sliding window, and natural log was applied. Utilizing the features and associated labels, two linear discriminant analysis (LDA) classifiers were trained, with one to recognize left imagery from right and the other to isolate right imagery movement development from others.

Features related to subject's control brain signals are extracted and constantly classified by the offline LDA algorithm. These features are then used to control the robot. They used parameters of the LDA algorithm in place of accommodating threshold and dwell features. Subject's control intention and timing is the basis for adaptation for online training. The methodology proposed entailed information about the user's control needed to train and adapt which could show promise of improving the accuracy and performance. The paper used Kalman filters for online parameters to be classified using LDA. Event label assignment was introduced with a slower learning process [12].

One of the advantages of implementing adaptive classifiers is that they can employ both supervised and unsupervised. This means that even if there is no information of true labels of data being received, they can be executed. From multiple research studies, it is inferred that unsupervised learning has yielded better results than static classifiers. Now, most of the real-world applications of BCI do not present class labels, and for this purpose, unsupervised adaption classifiers require more development [7].

Matrix and Tensor Classifiers

The approach which the researchers have used in this paper holds fewer stages for classification as compared to the classical machine learning algorithms, and they are simpler as well. Compared to other standard classifiers,

Riemannian classifier does not need any parameter-tuning techniques such as cross validation to properly train and verify the accuracy of the produced model, which makes it far more robust and accurate all due to its logarithmic tendencies. Likewise, the inborn Riemannian separation for the SPD matrix is invariant both to inversion of the matrix and to any direct invertible change of the information, for example, any outside interference added to the EEG sources does not change the separations among the witnessed covariance matrices. These properties partially clarify why Riemannian classification techniques give a decent speculation ability, which empowered analysts to set up adjustment-free versatile ERP-BCIs utilizing basic subject-to-subject and session-to-session exchange learning methodologies [9].

It is shown that several approaches were implemented and gave higher performance than CSP+LDA procedures on motor imagery EEG data. The biggest advantage is quality performance. However, this is a gain at tradeoff between performance and greater number of weights because of elevated expansion in input feature dimensionality which makes suitable regularization a much-needed step [12].

EEG data can be represented in the form of tensors and are treated as analysis tools for EEG data tools for EEG data analysis which includes feature extraction, data clustering, and data classification in the BCI. EEG data are represented in more than one dimension; this includes time, frequency space, and trails and hence, these are presented by the tensor order. EEG data that have time frequency and space can be represented by 3-dimensional tensor. Tensors have been used frequently for motor imagery-based analysis even with a large amount of data containing different categories which can be represented by the tensor and its order [10].

Transfer Learning and Deep Learning

Transfer learning is a crucial tool when it comes to session-to-session and subject-to-subject decoding performance. If transfer learning is improved enough, BCI system can be operated without calibration, and this will revolutionize the BCI systems forever.

It is observed that calibration sessions are strenuous and mentally exhausting for subjects. It is explained that accepting the input from the earliest starting point of their BCI system is promising for started subjects.

Deep learning is categorized as a ML algorithm, where features and the classifier are collectively learned straight from EEG data. There exist multiple deep learning algorithms. One of the most explored and commonly

used is deep neural networks (DNNs). DNN is also performed online for slow cortical potentials (SCP) and motion-onset visual evoked potential (MVEP) which are not commonly used. The very first research conducted and paper was published on the P300-based BCI by Cicotte et al. Two convolutional layers were constructed followed by completely connected layer. One convolutional layer has a purpose to learn spatial filters and the second one was to learn temporal filters. The model was proven better than the P300 experiment, but the SVM model had more accuracy [2].

Deep extreme learning machine is used for Slow Cortical Potentials (SCP) classifications. This technique consists of multiple layers, and the last one was Kernal ELM. However, in this project, number of units, network structure, hyperparameter, and input features were not reasoned. This did not prove to be better than multiplayer ELM or standard ELM [4].

Miscellaneous Classifiers

In order to classify more than two mental tasks, two main approaches can be used to obtain a multiclass classification function. The first approach consists in directly estimating the class using multiclass techniques such as decision trees, multilayer perceptron, naive Bayes classifiers, or *k*-nearest neighbors. The second approach consists of decomposing the problem into several binary classification problems.

Multiclass and multilabel approaches therefore aim to recognize more than two commands. It is therefore necessary to choose carefully the mapping between mental commands and corresponding labels. However, the errors may be possible during the classification. In particular, the set of estimated labels, sometimes, may not correspond to any class, and several classes may be at equal distances, thus causing class confusion [13].

METHODOLOGY

Here are some methods which are discussed in the research papers for the past few years in brain-computer interface systems, as shown in Table 3.

Table 3. Summary of various methodologies in BCI systems

Classification methods	Input EEG pattern	Features	References
Adaptive classifiers	Motor imaginary-based	Frequency band power, EEG time	[10, 12, 14]
Matrix and tensor classifiers	Steady-state visual evoked potentials, P300	Frequency band power, raw data	[2, 5, 7, 15]
Transfer learning and deep learning	Motor imaginary-based, P300, SSVEP	EEG time points, frequency band power, raw EEG data	[2, 4, 6, 7]
Miscellaneous classifiers	Motor imaginary-based, P300	Not specified	[1, 4, 15]

CONCLUSION

During this course of work, a question arises whether it is possible to create a brain computer interface which is affordable, with high accuracy and optimization. So, after reviewing different papers, the conclusion is that if we need an optimized model with high accuracy for the noninvasive technique of brain signals, the artificial neural network has a high accuracy and is optimal. However, there are some tradeoffs as well that are model compatibility with the brain signals. From 10 years of BCI review, we have obtained that the ANN has a high response and accuracy; after all, it optimizes the system as well. However, further research studies have been done to make it more accurate because this has to be used in health care.

Due to the fast processing that ANN allows, a form of guidance could be provided during training that enables a user to improve the classifier's performance and let it reflect his/her intents more often. This guidance was very useful for one of the three subjects.

Also, the statistical test was examined on whether it performs in a way that can be expected. Therefore, after the offline classification, it yielded an accuracy above 90 percent.

The control provided by the developed system has been sufficient to conclude that it provides enough control that a user can command an arbitrary computerized device. Also, it showed to be easily trainable.

In the future, the proposed model can provide support on multiplatforms. This can be achieved by developing applications which can help humanity and make everyday tasks easier. Furthermore, the system can be controlled with a smartphone that can override EEG headset commands. This will act as fail-safe if the BCI system experiences any malfunctioning. On the basis of this development, better EEG can be designed with higher efficiency and which is less dependent on offline classification.

REFERENCES

1. A. B. Usakli, "Improvement of EEG signal acquisition: an electrical aspect for state of the art of front end," *Computational Intelligence and Neuroscience*, vol. 2010, Article ID 630649, p. 12, 2010.
2. S. I. Alzahrani, "P300 Wave Detection Using Emotiv Epoc+ Headset: Effects of Matrix Size, Flash Duration, and Colors," Colorado State University, Fort Collins, CO, USA, 2016, Doctoral dissertation.
3. F. Yger, M. Berar, and F. Lotte, "Riemannian approaches in brain-computer interfaces: a review," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 10, pp. 1753–1762, 2016.
4. F. Lotte, L. Bougrain, A. Cichocki et al., "A review of classification algorithms for EEG-based brain–computer interfaces: a 10 year update," *Journal of Neural Engineering*, vol. 15, no. 3, Article ID 031005, 2018.
5. M. Li, W. Chen, and T. Zhang, "Classification of epilepsy EEG signals using DWT-based envelope analysis and neural network ensemble," *Biomedical Signal Processing and Control*, vol. 31, pp. 357–365, 2017.
6. A. Liu, K. Chen, Q. Liu, Q. Ai, Y. Xie, and A. Chen, "Feature selection for motor imagery EEG classification based on firefly algorithm and learning automata," *Sensors*, vol. 17, no. 11, p. 2576, 2017.
7. G. Liu, D. Zhang, J. Meng, G. Huang, and X. Zhu, "Unsupervised adaptation of electroencephalogram signal processing based on fuzzy C-means algorithm," *International Journal of Adaptive Control and Signal Processing*, vol. 26, no. 6, pp. 482–495, 2012.
8. A. Andreev, A. Barachant, F. Lotte, and M. Congedo, *Recreational Applications of OpenViBE: Brain Invaders and Use-The-Force*, Wiley Online Library, Hoboken, NY, USA, 2016.
9. D. R. Lowne, S. J. Roberts, and R. Garnett, "Sequential non-stationary dynamic classification with sparse feedback," *Pattern Recognition*, vol. 43, no. 3, pp. 897–905, 2010.
10. G. Pfurtscheller and C. Neuper, "Motor imagery and direct brain-computer communication," *Proceedings of the IEEE*, vol. 89, no. 7, pp. 1123–1134, 2001.
11. A. Schlögl, C. Vidaurre, and K. R. Müller, *Adaptive Methods in BCI Research-an Introductory Tutorial. In Brain-Computer Interfaces*, Springer, Berlin, Heidelberg, Germany, 2009.

12. C. Vidaurre, M. Kawanabe, P. von Bünau, B. Blankertz, and K. R. Müller, "Toward unsupervised adaptation of LDA for brain–computer interfaces," *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 3, pp. 587–597, 2010.
13. A. B. R. Suleiman and T. A. H. Fatehi, *Features Extraction Techniqes of EEG Signal for BCI Applications*, University of Mosul, Mosul, Iraq, 2007.
14. I. A. Fouad and F. E. Z. M. Labib, "Using emotiv EPOC neuroheadset to acquire data in brain-computer interface," *International Journal*, vol. 3, no. 11, pp. 1012–1017, 2015.
15. G. A.-R. Dornhege, *Toward Brain-Computer Interfacing*, MIT Press, Cambridge, MA, USA, 2007.

Section 4:

Deep Learning in Pattern Recognition Tasks

CHAPTER 14

THE APPLICATION OF DEEP LEARNING IN AIRPORT VISIBILITY FORECAST

Lei Zhu[1], Guodong Zhu[2,3], Lei Han[3], and Nan Wang[3]

[1]Training Center of Xinjiang Air Traffic Management Bureau, Urumqi, China

[2]College of Atmospheric Science, Nanjing University, Nanjing, China

[3]Meteorological Center of Xinjiang Air Traffic Management Bureau, Urumqi, China

ABSTRACT

This paper uses Urumqi International Airport's hourly observation from 2007 to 2016 and builds regression prediction model for airport visibility with deep learning method. From the results we can see: the absolute error of hourly visibility is 706 m. When the visibility ≤ 1000 m, the absolute error is 325 m, and this method can predict visibility's trend. So we can use this method to provide the airport visibility's objective forecast guidance

Citation: Zhu, L., Zhu, G., Han, L. and Wang, N. (2017), The Application of Deep Learning in Airport Visibility Forecast. Atmospheric and Climate Sciences, 7, 314-322. doi: 10.4236/acs.2017.73023.

products for aviation meteorological services in the future. In this paper, the Urumqi area is as the research object, to explore the depth of learning in the field of weather forecasting applications, providing a new visibility return forecast for weather forecast personnel so as to improve the visibility of the level of visibility to ensure the safe and stable operation of the airport.

Keywords:- Deep Learning, Airport Visibility, Regression Prediction

INTRODUCTION

With the rapid development of the national economy and the increasing popularity of civil aviation transport, airport operation on the visibility is becoming increasingly prominent. A long, low-visibility weather caused by fog, haze and other weather can cause a wide range of airport delays and cancellations. This not only has brought huge losses for the airlines and the airport, but also affects the public travel. At the same time visibility and flight safety are closely related. Low visibility is also one of the most common causes of flight accidents. Urumqi International Airport is the hub of the Xinjiang region airport. It is responsible for the Xinjiang region and Central Asia flight operations. The existing climate data show that Urumqi airport visibility was below 1000 m the average number of days for 60 days [1] [2]. Most of the low visibility days occurred in the winter half (November to March), up to 57 days. The weather phenomenon that causes low visibility is mainly fog and smoke.

Improving the level of visibility is an important measure to ensure the safe and stable operation of the airport. At present, the low visibility forecast for the smoke, fog and other weather, is still based on empirical forecasts and statistical forecasts. Although with the development of numerical forecasting, there are also numerical and fog model predictions and many experiments have shown that the fog model has only a certain degree of analytical use and is difficult to predict. Therefore, the study of atmospheric visibility is still a difficult and hot spot in meteorological forecast in recent years [3]-[11].

DEEP LEARNING

Deep Learning (DNNs) is also known as deep neural network (DNNs), which is the sub-field of machine learning. Its concept originated in the Artificial Neural Network (ANN). In essence, it refers to a class of neural networks with deep structure of the effective training methods. It uses a

multi-layer representation to model the complex relationships between the data [12] [13] [14]. Deep Learning can be used in sorting, regression and information retrieval and other specific issues.

THE ESTABLISHMENT OF PREDICTION MODEL

Data Preprocessing

This article uses the Urumqi Airport from 2007 to October 2016 to March the following year 24 hours a day observation data. Contains hourly dominant visibility, temperature, dew point temperature, relative humidity, average wind direction and average wind speed. By sorting and controlling data quality, 43,752 data records were received. Since each factor is composed of different meteorological elements, in order to avoid the difference in magnitude between the various factors, it needs to be normalized before the input factor as the depth neural network, so that its value is limited to [0, 1]. See Equation (1) for the specific algorithm.

$$\text{data} = \left(\text{data} - \min\left(\text{data}\right)\right) / \left(\max\left(\text{data}\right) - \min\left(\text{data}\right)\right) \tag{1}$$

Prediction of Forecasting Factors

For the time series regression prediction, the simplest way is to build a nonlinear function based on historical data. Combined with the prediction of dominant visibility, we construct two types of factors: the first type of forecasting factor contains only the dominant visibility (Vis) in the past. See Equation (2) for details. Because the dominant visibility is related to the factors such as wind, temperature and relative humidity, the second type of forecasting factor not only includes the dominant visibility in the past, but also the temperature (T), dew point temperature (TD), relative humidity (RH), wind direction (WD) and wind speed (WS). See Equation (3) for details.

$$\text{Vis}_t = \text{f}\left(\text{Vis}_{t-1}, \text{Vis}_{t-2}, \cdots, \text{Vis}_{t-n}\right) \tag{2}$$

$$\begin{aligned}\text{Vis}_t = \text{f}\big(&(\text{Vis}, \text{T}, \text{TD}, \text{RH}, \text{WD}, \text{WS})_{t-1}, (\text{Vis}, \text{T}, \text{TD}, \text{RH}, \text{WD}, \text{WS})_{t-2}, \cdots, \\ &(\text{Vis}, \text{T}, \text{TD}, \text{RH}, \text{WD}, \text{WS})_{t-n}\big)\end{aligned} \tag{3}$$

where the dominant visibility is Vis_t for the current dominant visibility. In order to verify the effect of the different length of time on the dominant visibility, here are a number of ns for modeling operations which is used to

evaluate the difference in the prediction effect of the model under different time length samples.

Build a Sample Sequence

This article takes hourly dominant visibility of Urumqi Airport as the forecast object. Using observations from October to March of the following ten years. According to the selection of forecast factors, two kinds of 28 sample sequences were established. See Table 1 for details. In each of the sample sequences, 80% of the 43,752 historical data were randomly selected as training samples for the prediction model training and the predictive effect test. The rest are test samples. In Table 1, Vis is visibility, T is temperature, TD is Temperature difference, RH is Relative humidity, WD is Wind Direction, WS is Wind speed.

According to the 28 sample sequences which were established the above two types of factor selection methods, the MLP model (Multilayer Perceptron Model) in Keras was used to model the operation. And using test samples to test it. Finally 28 different models of dominant visibility prediction were obtained. The prediction effect of the model is discussed in detail below.

Table 1. The number of two types of forecasting factors at different times

Number	Value at past n-hour	Number of the first type of forecasting factor	Number of the second type of forecasting factor
		Containing Vis	Containing Vis, T, TD, RH, WD, WS
1	n = 1	1	6
2	n = 3	3	18
3	n = 6	6	36
4	n = 9	9	54
5	n = 12	12	72
6	n = 24	24	144
7	n = 36	36	216
8	n = 48	48	288

9	n = 60	60	360
10	n = 72	72	432
11	n = 84	84	504
12	n = 96	96	576
13	n = 108	108	648
14	n = 120	120	720

PREDICTIVE EFFECT TEST

This paper constructs the forecasting object for Urumqi airport hourly dominant visibility, which contains a total of nearly 45,000 records. Through the analysis of the forecast object, you can see the Urumqi airport visibility changes in the range of 0 to 10,000 meters. 26.9% of the dominant visibility is 10,000 meters. While the impact of the operation of the civil aviation airport to dominate the visibility of less than 1000 meters accounted for 12.3% of the record. Specific distribution is shown in Figure 1. By using the MLP model to predict the dominant visibility, an hourly dominant visibility prediction result is obtained. Here we examine the predictive effect of this method from two different types of forecasting factors.

A Model with Dominant Visibility as a Predictor

This type of model contains only the dominant visibility of the past, without adding other meteorological elements. Constructing a predictive model that dominates the dominant time at historical time is based on the dominant visibility at the current time. The prediction model was established by using the dominant visibility of the past 1, 3, 6, 9, 12, 24, 36, 48, 60, 72, 84, 96, 108, 120 hours.

The results of the predictive results shown in Figure 2 show that the average absolute error of dominant visibility is between 966 m and 706 m using historical dominant visibility as a predictor for different time lengths. It is best to use the forecast for 1 hour before the current time. The average absolute error is 706.98 m. The prediction effect of the model with different time length is different, and the error increases slowly with the increase of the length of time.

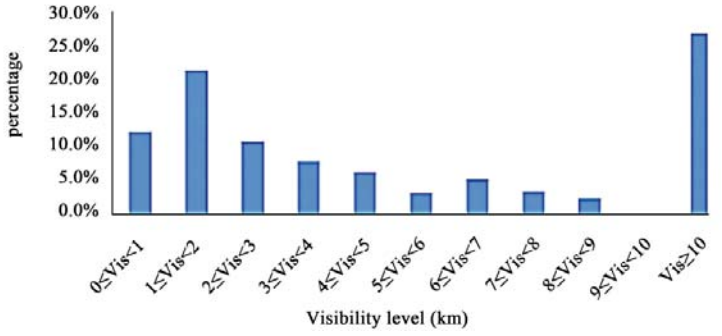


Figure 1. Urumqi airport visibility distribution from November to next March.

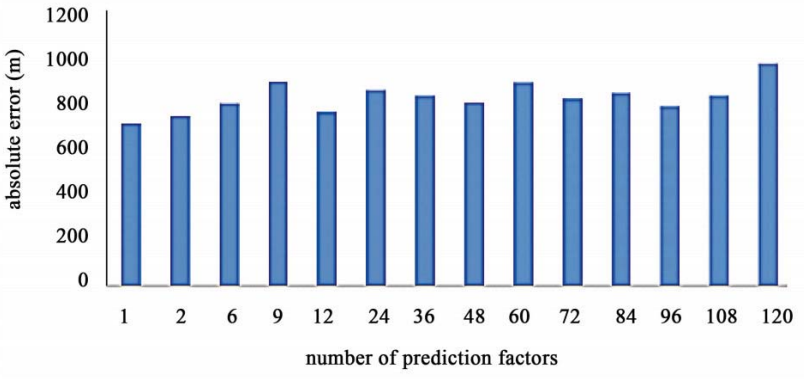


Figure 2. The results using the dominant visibility of different time lengths.

In order to fully test the different factors of the model to dominate the visibility, the following statistics by 5000 m within the dominant visibility to predict the average absolute error. And analyzing the ability of the model to predict the dominant visibility of different scales. It can be seen from Table 2 that this model has a mean absolute error of less than 1000 m between 325 m and 520 m, and the average absolute error between the training sample and the test sample is small. Which used the past 1, 3 hours two kinds of factors to predict the effect is better. As the length of time in the predictor increasing, the mean absolute error of dominant visibility increases. In addition, as the size of the predicted object increasing, the average absolute error of the dominant visibility in the [4000, 5000] interval is significantly greater than the average absolute error in the range [0, 1000], to about 1200 m.

Table 2. Average absolute error (in m) for visibility below 5000 m

		0 ≤ Vis < 1000	1000 ≤ Vis < 2000	2000 ≤ Vis < 3000	3000 ≤ Vis < 4000	4000 ≤ Vis < 5000
1	train	327.29	423.56	663.63	840.48	1133.94
	test	326.47	378.32	726.58	897.89	1230.10
3	train	325.97	431.51	690.45	852.26	1134.11
	test	328.46	380.70	741.47	908.79	1208.05
6	train	342.56	446.40	705.54	870.03	1151.82
	test	340.53	398.67	751.12	926.55	1220.44
9	train	456.03	581.39	850.16	991.67	1284.93
	test	428.65	502.76	887.82	1102.37	1337.66
12	train	344.53	452.17	708.21	883.09	1184.77
	test	334.67	406.62	762.61	950.00	1272.34
24	train	380.97	473.22	705.61	865.83	1139.36
	test	367.18	429.12	752.18	938.26	1248.22
36	train	570.81	652.72	887.72	1018.68	1310.13
	test	517.37	591.00	972.87	1170.40	1423.56
48	train	436.89	518.80	754.84	897.60	1173.12
	test	402.15	476.46	821.20	1009.72	1271.40
60	train	437.95	518.15	751.58	893.35	1162.32
	test	408.69	475.60	831.60	996.01	1267.75
84	train	451.67	517.76	736.56	884.41	1153.88
	test	415.40	471.45	812.99	1003.08	1258.19
96	train	523.64	531.87	735.77	858.77	1119.00
	test	457.69	488.49	819.98	981.49	1234.37
108	train	519.76	542.62	752.20	874.10	1136.56
	test	459.54	503.04	847.17	1017.79	1262.01
120	train	492.18	509.75	711.91	841.43	1092.73
	test	439.76	480.97	793.34	956.49	1200.19

A Multi-Meteorological Element Used as a Model of Forecasting Factors

This paper attempts to extend the forecasting factor to the dominant visibility, temperature, dew point temperature, wind direction, and wind direction of the past, due to the fact that there are many reasons for the occurrence of low visibility weather and visibility and temperature, relative humidity and other factors. The model is used to predict the dominant visibility. Through the analysis of the forecast results can be found, using the multi-factor predictor model, the average absolute error in models of different time lengths is predicted from 799 m to 827 m. The average absolute error of the different models is about 10 m. The model used in the past 24-hour multi-factor forecasting factor is best. Its absolute error is 798.87 m. See Figure 3 for details.

In order to fully test the different factors of the model to dominate the visibility, the following statistics the dominant visibility within 5000 m to predict the average absolute error. The ability of the model to predict the dominant visibility of different orders of magnitude. The model has the best predictive effect on the dominant visibility within 1000 m. Its average absolute error is between 450 m and 550 m. Which used the past 72,120 hour factor to build the model to predict the effect is better. With the increase of the dominant visibility level, the error of the model prediction is gradually increased to about 1100 m. The detailed data statistics table is omitted here.

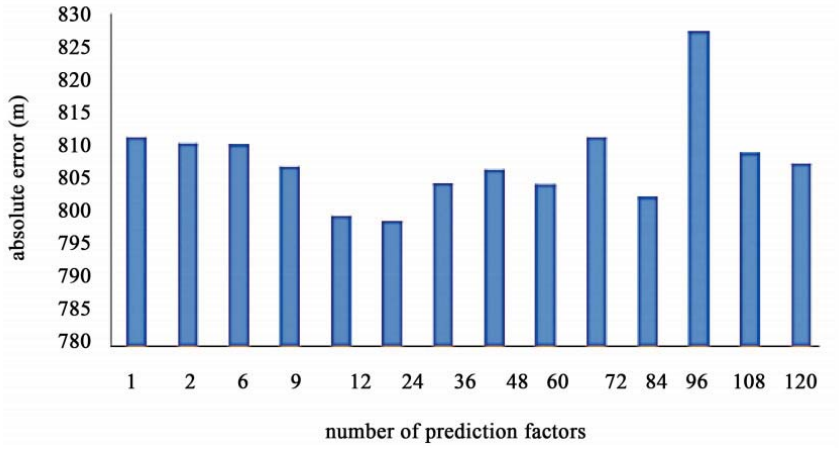


Figure 3. The effect using the dominant visibility of multiple factors for different time lengths.

Suggestion of Practical Predictive

Effect Based on the model predictive effect constructed by the two kinds of forecasting factors discussed in this paper, the forecasting model of single factor construction is better for the prediction with less than 1000 m visibility. The forecasting model of multi-factor construction is more stable than the forecasting effect of more than 2000 m. So here at the same time use these two models. And taking into account the continuity of the change in visibility, we choose the model using the forecast factor of the past 12 h to make the actual forecast.

Here we select Urumqi Airport December 31, 2016 visibility to predict. The day the airport visibility changes greatly, before 11 hours to maintain more than 1000 m, then quickly decreased to maintain two hours of 100 m, then 16:00 suddenly improved to 2000 m, and then down to maintain at 100 m, see Figure 4. The low visibility process includes persistent low visibility and a sudden improvement in visibility. This has a high test of the ability of the model to predict.

It can be seen from the model predictions that both models can predict the trend of decreasing the visibility and the turn of the day. When the real visibility is greater than 1500 m, the prediction error of the multi-factor model is relatively small. When the dominant visibility is less than 1000 m, the prediction effect of the single factor is relatively better, especially when the long-term continuous visibility is less than 300 m, the average absolute error of the visibility of the single factor is 86 m. Therefore, by using the two types of forecasting models, it is possible to provide a quantitative reference for the forecasting staff to predict the visibility. However, it can be seen from the simulation of the actual case that the depth learning model has a certain hysteresis when the dominant visibility is good or worse, and the error increases obviously when the visibility is greater than 1500 m.

CONCLUSIONS

Due to the high incidence of low visibility weather, the impact of the system is more complex. Especially the forecasts of low visibility of the starting and ending time are more difficult. So how to as much as possible predict low visibility weather and dissipation of the time are the keys of Urumqi Airport winter service guarantee.

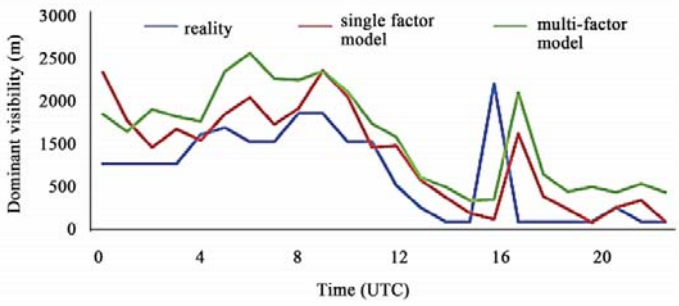


Figure 4. The effect of different models on the forecast of dominant visibility on December 31, 2016.

This paper attempts to use the depth neural network for airport visibility prediction, using nearly ten years' data for model prediction. The results of the model forecast show that it can reflect the trend of the dominant visibility of Urumqi Airport. The average absolute error can reach a minimum of 706 m. Where the minimum absolute error of less than 1000 m, it is as low as 325 m. At the same time, we can see that the prediction results of multi-factor factor prediction model are stable. Next, we will try to put this model into the actual business and conduct continuous testing to improve the quantitative forecasting capability of this method in leading visibility.

Although the method has better prediction effect, in the detailed analysis of its forecast results also found some shortcomings, such as predictive visibility turn or turn bad times have a certain lag. The results show that the average absolute error is greater than 2000 m above the dominant visibility, and the prediction effect is less than 1000 m. Then, we will cooperate with low weather conditions, and try to combine the ability to reflect low-level stratification conditions, high-altitude wind field and ground pressure field and other factors as a forecast factor to ensure that the forecasting factor can better contain the low visibility weather conditions to improve the prediction effect of the model.

REFERENCES

1. Zhu, L. and Zhu, G.D. (2012) Analysis of Low Visibility Weather Characteristics of Urumqi Airport in Recent 30 Years. Journal of Civil Aviation Flight Administration of China, 23, 27-30.
2. Zhu, L. and Zhu. G.D. (2010) Application of Support Vector Machine Method in Visual Field Forecast of Airport Runway. Journal of Stroke, 29, 171-175.
3. Li, P., Wang, S.G., Shang, K.H., et al. (2012) Beijing Area Visibility Prediction Based on Neural Network Step-by-Step Classification Modeling. Journal of Lanzhou University (Natural Science Edition), 48, 52-57. (In Chinese)
4. Wang, K., Zhao, H., Liu, A.X., et al. (2009) Prediction of Atmospheric Visibility Based on Risk Neural Network. China Environmental Science, 29, 1029-1033.
5. Chen, H.Z., Hao, L.P., Xie, N., et al. (2009) Digitalization of Information and Refinement Prediction of Fog. Journal of Natural Disasters, 18, 151-156.
6. Zhu, G.D. (2011) Multi-Element Prediction of Urumqi International Airport based on SVM Method. Development and Oasis Meteorology, 5, 40-43.
7. Peng, S.T., Hu, Y.D., Zhou, R., et al. Application of Artificial Neural Network in Urban Short-term Forecast of Visibility. Proceedings of the 6th Annual Conference of Chinese Society for Particles and Cross-Strait Particle Technology Symposium, 606-609.
8. Shao, Z.P. (2014) Zhengzhou Airport Visibility Changes and the Causes of Fog Analysis. Meteorological and Environmental Science, 37, 75-82.
9. He, H. and Luo, H. (2009) Fark Forecasting Method Based on Support Vector Machine Pattern Recognition. Meteorological Science and Technology, 37, 149-151.
10. Cao, Z.J., Wu, D., Wu, X.J., et al. (2008) Climate Characteristics of Fog in China from 1961 to 2005. Meteorological Science and Technology, 36, 556-560.
11. Li, X.l., Chen, K.J., Wang, K., et al. (2008) Classification and Statistical Analysis of Fog in the Capital Airport. Meteorological Science and Technology, 36, 717-723.

12. Bengio, Y. (2009) Learning Deep Architectures for AI. Foundations and Trends in Machine Learning, 2, 1-127. https://doi.org/10.1561/2200000006
13. Hinton, G., Osindero, S. and Teh, Y. (2006) A Fast Learning Algorithm for Deep Belief Nets. Neural Computation, 18, 1527-1554. https://doi.org/10.1162/neco.2006.18.7.1527
14. Liu, J.W., Liu, Y., Luo, X.L., et al. (2014) Progress in the Study of Depth. Application Research of Computers, 31, 1921-1930, 1942.

CHAPTER 15

HIERARCHICAL REPRESENTATIONS FEATURE DEEP LEARNING FOR FACE RECOGNITION

Haijun Zhang[1,2] andYinghui Chen[1,3]

[1]Guangdong Provincial Key Laboratory of Conservation and Precision Utilization of Characteristic Agricultural Resources in Mountainous Areas, Meizhou, China

[2]School of Computing, Jiaying University, Meizhou, China

[3]School of Mathematics, Jiaying University, Meizhou, China

ABSTRACT

Most modern face recognition and classification systems mainly rely on hand-crafted image feature descriptors. In this paper, we propose a novel deep learning algorithm combining unsupervised and supervised learning named deep belief network embedded with Softmax regress (DBNESR) as a natural source for obtaining additional, complementary hierarchical representations, which helps to relieve us from the complicated hand-crafted

Citation: Zhang, H. and Chen, Y. (2020), Hierarchical Representations Feature Deep Learning for Face Recognition. Journal of Data Analysis and Information Processing, 8, 195-227. doi: 10.4236/jdaip.2020.83012.

feature-design step. DBNESR first learns hierarchical representations of feature by greedy layer-wise unsupervised learning in a feed-forward (bottom-up) and back-forward (top-down) manner and then makes more efficient recognition with Softmax regress by supervised learning. As a comparison with the algorithms only based on supervised learning, we again propose and design many kinds of classifiers: BP, HBPNNs, RBF, HRBFNNs, SVM and multiple classification decision fusion classifier (MCDFC)—hybrid HBPNNs-HRBFNNs-SVM classifier. The conducted experiments validate: Firstly, the proposed DBNESR is optimal for face recognition with the highest and most stable recognition rates; second, the algorithm combining unsupervised and supervised learning has better effect than all supervised learning algorithms; third, hybrid neural networks have better effect than single model neural network; fourth, the average recognition rate and variance of these algorithms in order of the largest to the smallest are respectively shown as DBNESR, MCDFC, SVM, HRBFNNs, RBF, HBPNNs, BP and BP, RBF, HBPNNs, HRBFNNs, SVM, MCDFC, DBNESR; at last, it reflects hierarchical representations of feature by DBNESR in terms of its capability of modeling hard artificial intelligent tasks.

Keywords:- Face Recognition, Unsupervised, Hierarchical Representations, Hybrid

INTRODUCTION

Face recognition (FR) is one of the main areas of investigation in biometrics and computer vision. It has a wide range of applications, including access control, information security, law enforcement and surveillance systems. FR has caught the great attention from large numbers of research groups and has also achieved a great development in the past few decades [1] [2] [3]. However, FR suffers from some difficulties because of varying illumination conditions, different poses, disguise and facial expressions and so on [4] [5] [6]. A plenty of FR algorithms have been designed to alleviate these difficulties [7] [8] [9]. FR includes three key steps: image preprocessing, feature extraction and classification. Image preprocessing is essential process before feature extraction and also is the important step in the process of FR. Feature extraction is mainly to give an effective representation of each image, which can reduce the computational complexity of the classification algorithm and enhance the separability of the images to get a higher recognition rate. While classification is to distinguish those extracted

features with a good classifier. Therefore, an effective face recognition system greatly depends on the appropriate representation of human face features and the good design of classifier [10].

To select the features that can highlight classification, many kinds of feature selection methods have been presented, such as: spectral feature selection (SPEC) [11], multi-cluster feature selection (MCFS) [12], minimum redundancy spectral feature selection (MRSF) [13], and joint embedding learning and sparse regression (JELSR) [14]. In addition, wavelet transform is popular and widely applied in face recognition system for its multi-resolution character, such as 2-dimensional discrete wavelet transform [15], discrete wavelet transform [16], fast beta wavelet networks [17], and wavelet based feature selection [18] [19] [20].

After extracting the features, the following work is to design an effective classifier. Classification aims to obtain the face type for the input signal. Typically used classification approaches include polynomial function, HMM [21] [22], GMM [23], K-NN [23], SVM [24], and Bayesian classifier [25]. In addition, random weight network (RWN) is proposed in some articles [26] [27] and there are also other kinds of neural networks used as the classifier for FR [28] [29].

In this paper, we first make image preprocessing to eliminate the interference of noise and redundant information, reduce the effects of environmental factors on images and highlight the important information of images. At the same time, in order to compensate the deficiency of geometric features, it is well known that the original face images often need to be well represented instead of being input into the classifier directly because of the huge computational cost. So PCA and 2D-PCA are used to extract geometric features from preprocessed images, reduce their dimensionality for computation and attain a higher level of separability. At last, we propose a novel deep learning algorithm combining unsupervised and supervised learning named deep belief network embedded with Softmax regress (DBNESR) to learn hierarchical representations for FR; as a comparison with the algorithms only based on supervised learning, again design many kinds of other classifiers and make experiments to validate the effectiveness of the algorithm.

The proposed DBNESR has several important properties, which are summarized as follows: 1) Through special learning, DBNESR can provide effective hierarchical representations [30]. For example, it can capture the intuition that if a certain image feature (or pattern) is useful in some

locations of the image, then the same image feature can also be useful in other locations or it can capture higher-order statistics such as corners and contours, and can be tuned to the statistics of the specific object classes being considered (e.g., faces). 2) DBNESR is similar to the multiple nonlinear functions mapping, which can extract complex statistical dependencies from high-dimensional sensory inputs (e.g., faces) and efficiently learn deep hierarchical representations by re-using and combining intermediate concepts, allowing it to generalize well across a wide variety of computer vision (CV) tasks, including face recognition, image classification, and many others. 3) Further, an end system making use of deep learning hierarchical representations features can be more readily adapted to new domains.

The analysis and experiments are performed on the precise rate of face recognition. The conducted experiments validate: Firstly, the proposed DBNESR is optimal for face recognition with the highest and most stable recognition rates; Second, the deep learning algorithm combining unsupervised and supervised learning has better effect than all supervised learning algorithms; Third, hybrid neural networks has better effect than single model neural network; Fourth, the average recognition rate and variance of these algorithms in order of largest to smallest are respectively shown as DBNESR, MCDFC, SVM, HRBFNNs, RBF, HBPNNs, BP and BP, RBF, HBPNNs, HRBFNNs, SVM, MCDFC, DBNESR; At last, it reflects hierarchical representations of feature by DBNESR in terms of its capability of modeling hard artificial intelligent tasks.

The remainder of this paper is organized as follows. Section 2 reviews the images preprocessing. Section 3 introduces the feature extraction methods. Section 4 designs the classifiers of supervised learning. Section 5 gives and designs the classifier combining unsupervised and supervised learning proposed by us. Experimental results are presented and discussed in Section 6. Section 7 gives the concluding remarks.

IMAGES PREPROCESSING

Images often appear the phenomenon such as low contrast, being not clear and so on in the process of generation, acquisition, input, etc. of images due to the influence of environmental factors such as the imaging system, noise and light conditions so on. Therefore it needs to make images preprocessing. The purpose of the preprocessing is to eliminate the interference of noise and redundant information, reduce the effects of environmental factors on images and highlight the important information of images [31].

Images preprocessing usually includes gray of images, images filtering, gray equalization of images, standardization of images, compression of images (or dimensionality-reduced) and so on [32]. The process of images preprocessing is as following.

1) ***Face images filtering:-*** We use median filtering to make smoothing denoising for images. This method not only can effectively restrain the noise but also can very well protect the boundary. Median filter is a kind of nonlinear operation, it sorts a pixel point and all others pixel points within its neighborhood as the size of grey value, sets the median of the sequence as the gray value of the pixel point, as shown in Equation (1).

$$f'(i,j) = \underset{s}{Med}\left\{f(i,j)\right\} \tag{1}$$

where, s is the filter window. Using the template of 3 × 3 makes median filtering for the experiment in the back.

2) ***Histogram equalization:-*** The purpose of histogram equalization is to make images enhancement, improve the visual effect of images, make redundant information of images after preprocessing less and highlight some important information of images.

Set the gray range of image A(x,y) as [0, L] , image histogram for $H_A(r)$, Therefore, the total pixel points are:

$$A_0 = \int_0^L H_A(r)\,\mathrm{d}r \tag{2}$$

Making normalization processing for the histogram, the probability density function of each grey value can be obtained:

$$p(r) = \frac{H_A(r)}{A_0} \tag{3}$$

The probability distribution function is:

$$P(r) = \int_0^L p(r)\,\mathrm{d}r = \frac{1}{A_0}\int_0^L H_A(r)\,\mathrm{d}r \tag{4}$$

Set the gray transformation function of histogram equalization as the limited slope not reduce continuously differentiable function s = (T_r), input it into A(x, y) to get the output B (x, y) . H_B (r) is the histogram of output image, it can get

$$H_B(s)\,\mathrm{d}s = H_A(r)\,\mathrm{d}r \tag{5}$$

$$H_B(s) = \frac{H_A(r)}{\mathrm{d}s/\mathrm{d}r} = \frac{H_A(r)}{T'(r)} \tag{6}$$

where, $T'(r) = \mathrm{d}s/\mathrm{d}r$. Therefore, when the difference between the molecular and denominator of H_B (r) is only a proportionality constant, H_B (r) is constant. Namely

$$T'(r) = \frac{C}{A_0} H_A(r) \tag{7}$$

$$s = T(r) = \frac{C}{A_0} \int_0^r H_A(r)\mathrm{d}r = CP(r) \tag{8}$$

In order to make the scope of s for [0, L] , can get C = L. For discrete case the gray transformation function is as following:

$$s = T(r) = CP(r_k) = C\sum_{i=0}^{k} p(r_i) = C\sum_{i=o}^{k} \frac{n_i}{n} \tag{9}$$

where, r_k is the kth grayscale, n_k is the pixel number of k_r, n is the total pixels number of images, the scope of k for [0, L −1] .

We make the histogram equalization experiment for the images in the back.

3) ***Compression of images (or dimensionality-reduced):-*** It is well known that the original face images often need to be well represented instead of being input into the classifier directly because of the huge computational cost. As one of the popular representations, geometric features are often extracted to attain a higher level of separability. Here we employ multi-scale two-dimensional wavelet transform to generate the initial geometric features for representing face images.

We make the multi-scale two-dimensional wavelet transform experiment for the images in the back.

FEATURE EXTRACTION

There are two main purposes for feature extraction: One is to extract characteristic information from the face images, the feature information can classify all the samples; The second is to reduce the redundant information of the images, make the data dimensionality being on behalf of human faces as far as possibly reduce, so as to improve the speed of subsequent operation process. It is well known that image features are usually classified into four

classes: Statistical-pixel features, visual features, algebraic features, and geometric features (e.g. transform-coefficient features).

1) ***Extract features with PCA:-*** Suppose that there are N facial images $\{X_i\}_{i=1}^{N}$, X_i is column vector of M dimension. All samples can be expressed as following:

$$X = (X_1, X_2, \cdots, X_N)^{\mathrm{T}} \tag{10}$$

Calculate the average face of all sample images as following:

$$\bar{X} = \frac{1}{N}\sum_{i=1}^{N} X_i \tag{11}$$

Calculate the difference of faces, namely the difference of each face with the average face as following:

$$d_i = X_i - \bar{X}, i = 1, 2, \cdots, N \tag{12}$$

Therefore, the images covariance matrix C can be represented as following:

$$C = \frac{1}{N}\sum_{i=1}^{N} d_i d_i^{\mathrm{T}} = \frac{1}{N} AA^{\mathrm{T}}$$
$$A = (d_1, d_2, \cdots, d_N) \tag{13}$$

Using the theorem of singular value decomposition (SVD) to calculate the eigenvalue λ_i and orthogonal normalization eigenvector ν_i of $A^{\mathrm{T}}A$, through Equation (14) the eigenvalues of covariance matrix C can be calculated.

$$u_i = \frac{1}{\sqrt{\lambda_i}} A v_i, (i = 1, 2, \cdots, N) \tag{14}$$

Making all the eigenvalues $[\lambda_1, \lambda_2, \cdots, \lambda_N]$ order in descend according to the size, through the formula as following:

$$t = \min_k \left\{ \frac{\sum_{j=1}^{k} u_j}{\sum_{j=1}^{N} u_j} > \alpha, k \le t \right\} \tag{15}$$

where, usually set a = 90% , can get the eigenvalues face subspace $U = (u_1, u_2, \cdots, u_t)$. All the samples project to subspace U, as following:

$$Z = U^{\mathrm{T}} X \tag{16}$$

Therefore, using front t principal component instead of the original vector X, not only make the facial features parameter dimension is reduced, but also won't loss too much feature information of the original images.

2) ***Extract features with 2D-PCA:-*** Suppose sample set is $\left\{S_j^i \in R^{m\times n}, i=1,2,\cdots,N; j=1,2,\cdots,M\right\}$, i is the category, j is the sample of the ith category, N is the total number of category, M is the total number of samples of each category, K = NM · is the number of all samples.

Let $\overline{S}$ be average of all samples as follows:

$$\overline{S}=\frac{1}{K}\sum_{i=1}^{N}\sum_{j=1}^{M}S_j^i \tag{17}$$

Therefore, the images covariance matrix G can be represented as follows:

$$G=\frac{1}{K}\sum_{i=1}^{N}\sum_{j=1}^{M}\left(S_j^i-\overline{S}\right)^{\mathrm{T}}\left(S_j^i-\overline{S}\right) \tag{18}$$

and the generalized total scattered criterion J (X) can be expressed by:

$$J(X)=X^{\mathrm{T}}GX \tag{19}$$

Let X_{opt} be the unitary vector such that it maximizes the generalized total scatter criterion J (X) , that is:

$$X_{opt}=\arg\max_X J(X) \tag{20}$$

In general, there is more than one optimal solution. We usually select a set of optimal solutions $\{X_1,\cdots,X_t\}$ subjected to the orthonormal constraints and the maximizing criterion J (X), where, t is smaller than the dimension of the coefficients matrix. In fact, they are those orthonormal eigenvectors of the matrix G corresponding to t largest eigenvalues.

Now for each sub-band coefficient matrix Si, compute the principal component of the matrix S_i as follows:

$$y_{ij}=A_i x_j, j=1,2,\cdots,t \tag{21}$$

Then we can get its reduced features matrix $Y_i=[y_{i1},\cdots,y_{it}],\ i=1,2,\cdots,m$.

We extract features respectively with PCA and 2D-PCA and compare their effects for the images in the back experiment.

DESIGNING THE CLASSIFIERS OF SUPERVISED LEARNING

Usually the classifiers based on supervised learning are often used for FR, in the paper we design two types of classifiers. One is the type of supervised learning classifiers and the other is the classifiers combining unsupervised and supervised learning [33].

1) ***BP neural network:-*** BP neural network is a kind of multilayer feed-forward network according to the back-propagation algorithm for errors, is currently one of the most widely used neural network models [34]. Recognition and classification of face images is an important application for BP neural network in the field of pattern recognition and classification.

The network consists of L layers as shown in Figure 1. Its training algorithm consists of three steps, illustrated as follows [35].

2) ***Hybrid BP neural networks (HBPNNs):-*** When the number scale of human face images isn't big, generalization ability and operation time of single model BP neural network are ideal, and with the increase of numbers of identification species, the structure of BP network will become more complicated, which causes the time of network training to become longer, slower convergence rate, easy to fall into local minimum and poorer generalization ability and so on.

In order to eliminate these problems we design the hybrid BP neural networks (HBPNNs) composed of multiple single model BP networks to replace the complex BP network for FR. Hybrid networks have better fault tolerant and generalization than single model network, and can implement distributed computing to greatly shorten the training time of network [36].

The core idea of designing hybrid networks classifier is to divide a K-class pattern classification into K independent 2-class pattern classification. That is to make a complex classification problem decomposed into some simple classification problems. In the paper multiple single model BP networks are combined into a hybrid network classifier, namely make K BP networks of multiple inputs single output integrated, a BP network is a child network only being responsible for identifying one of K-class model category and parallel to each other between different subnets. In reference of Figure 1 the model figure of HBPNNs is shown in Figure 2.

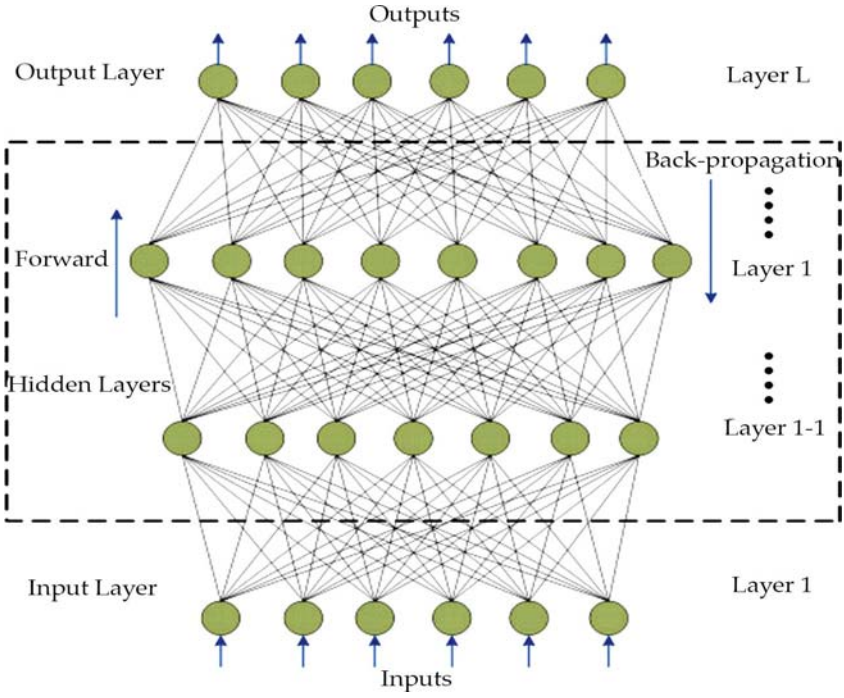


Figure 1. Single model BP neural network.

BP neural network only having a hidden layer and with sufficient hidden neurons is sufficient for approximating the input-output relationship [37]. Therefore, it selects standard three-layer BP neural network as the subnets for hybrid networks. For each subnets of hybrid networks, the number of neurons of input layer corresponds to the dimensions of face feature extraction, the number of neurons of output layer is 1. The number of neurons of hidden layer is calculated by the following empirical formula:

$$h = \sqrt{n+m} + a \tag{22}$$

where, m are the number of neurons of output layer, n are the number of neurons of input layer, a is constant between 1 - 10 [38]. If the dimensions of face feature extraction are X, the structure of each subnets of the hybrid networks is as following:

$$X \to \left(\sqrt{X+1} + a\right) \to 1 \tag{23}$$

The structure of BP neural network is as following:

$$X \to \left(\sqrt{X+K} + a\right) \to K \tag{24}$$

The structure of subnets is simpler than the structure of single model BP neural network. When the structure of networks is complex, every increasing a neural the training time will greatly increase. In addition, with the size of networks gradually becoming larger, more and more complex network structure is easy to have slow convergence, prone to fall into local minimum, to have poor generalization ability and so on. By contrast, the hybrid networks based on some subnets can obtain more stable and efficient classifiers in the shorter period of time of training.

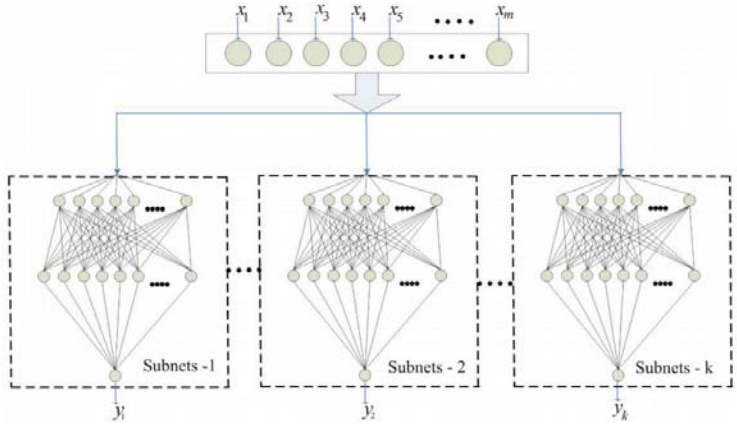


Figure 2. Hybrid BP neural networks (HBPNNs).

3) ***RBF neural network:-*** Radial Basis Function (RBF) simulates the structure of neural network of the adjustment and covering each other of receiving domain of human brain, can approximate any continuous function with arbitrary precision. With the characteristics of fast learning, won't get into local minimum.

The expression of RBF is as following [39]:

$$\phi(x) = \phi(\|x - c\|) \tag{25}$$

Where, $x, c \in R^n$, Euclidean distance of x to c is $\|x - c\|$. The radial basis function most commonly used is the Gaussian function for RBF neural network as following:

$$\phi(x) = \exp\left(-\frac{\|x - c\|^2}{\sigma^2}\right) \tag{26}$$

where, σ is the width of the function. Radial basis function is often used to construct the function as following:

$$y(x)=\sum_{i=1}^{M} w_i \phi(\|x-c_i\|) \quad (27)$$

There are some different for c_i of each radial basis function and the weight w_i. The concrete process of training RBF is as follows.

For the set of sample data $\{(x_i,d_i)\}_{i=1}^{N}$, we use Equation (27) with M hidden nodes to classify those sample data.

$$\{(x_i,d_i)\}_{i=1}^{N} \quad (28)$$

The number of hidden nodes is chosen to be a small integer initially in applications. If the training error is not good, we can increase hidden nodes to reduce it. Considering the testing error simultaneously, there is a proper number of hidden nodes in applications. The model figure of RBF is shown in Figure 3.

4) ***Hybrid RBF neural networks (HRBFNNs):-*** The hybrid RBF neural networks (HRBFNNs) are composed of multiple RBF networks to replace RBF network for FR. Hybrid networks have better fault tolerant, higher convergence rate and stronger generalization than a single model network, and can implement distributed computing to greatly shorten the training time of network [40].

If the dimensions of face feature extraction are n, the structure of each subnets of the hybrid networks is as following:

$$n \to m \to 1 \quad (29)$$

The structure of RBF neural network is as following:

$$n \to m \to k \quad (30)$$

The structure of subnets is simpler than the structure of RBF neural network. In addition, when the structure of networks is complex, every increasing a neural the training time and amount of calculation will greatly increase. The model figure of the HRBFNNs is shown in Figure 4.

5) ***Support Vector Machine (SVM):-*** SVM is a novel machine learning technique based on the statistical learning theory that aims at finding the optimal hyper-plane among different classes (usually to solve binary classification problem) of input data or training data in high dimensional feature space, and new test data can be classified by the separating hyper-plane [41] [42].

Supposing there are two classes of examples (positive and negative), the label of positive example is +1 and negative example is −1. The number of positive and negative examples respectively is n and m. The set $\{x_i\}_{i=1}^{n+m}$ are given positive and negative examples for training. The set $\{y_i\}_{i=1}^{n+m}$ are the labels of i x , in which $\{y_i = +1\}_{i=1}^{n}$ and $\{y_i = -1\}_{i=n+1}^{n+m}$.

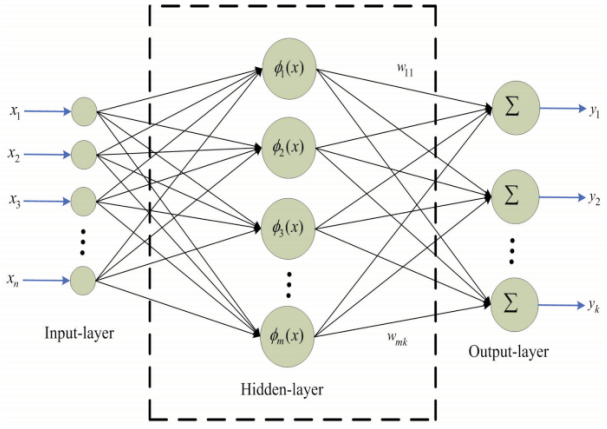


Figure 3. RBF neural networks.

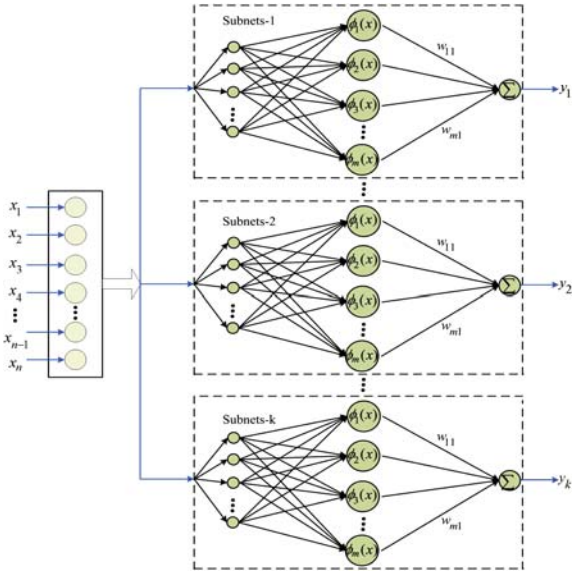


Figure 4. Hybrid RBF neural networks (HRBFNNs).

SVM is to learn a decision function to predict the label of an example. The optimization formulation of SVM is:

$$\begin{aligned}&\min \frac{\|w\|^2}{2}+G\sum_{i=1}^{n+m}\xi_i,\\&\text{s.t. } wx_i+b\geq 1-\xi_i, i=1,\cdots,n,\\&\quad wx_i+b\leq -1+\xi_i, i=n+1,\cdots,n+m\end{aligned} \tag{31}$$

where, ξ_i is the slack variables and G controls the fraction on misclassified training examples. This is a quadratic programming problem, use Lagrange multiplier method and meet the KKT conditions, can get the optimal classification function for the above problems:

$$f(x)=\operatorname{sgn}\left\{w\cdot x+b^*\right\}=\operatorname{sgn}\left\{\sum_{i=1}^{n}a_i^*y_i(x_i\bullet x)+b^*\right\} \tag{32}$$

where, a_i^* and b^* are to the parameters to determine the optimal classification surface. $(x_i\bullet x)$ is the dot product of two vectors.

For the nonlinear problem SVM can turn it into a high dimensional space by the nonlinear function mapping to solve the optimal classification surface. Therefore, the original problem becomes linearly separable. As can be seen from Equation (32) if we know dot product operation of the characteristics space the optimal classification surface can be obtained by simple calculation. According to the theory of Mercer, for any ϕ (x) ≠ 0 if:

$$\begin{cases}\int\varphi^2(x)\mathrm{d}x<\infty \quad \text{and}\\ \iint K(x_i,x_j)\varphi(x_i)\varphi(x_j)\mathrm{d}x_i\mathrm{d}x_j>0\end{cases} \tag{33}$$

The arbitrary symmetric function $K(x_i,x_j)$ will be the dot product of a certain transformation space. Equation (32) will be corresponding to:

$$f(x)=\operatorname{sgn}\left\{\sum_{i=1}^{n}a_i^*y_iK(x_i\bullet x)+b^*\right\} \tag{34}$$

This is SVM. There are a number of categories of the kernel function K (x, x_i):

- The linear kernel function $K(x,x_i)=(x\bullet x_i)$;;
- The polynomial kernel function $K(x,x_i)=\left(s(x\bullet x_i)+c\right)^d$,where s, c and d are parameters;
- The radial basis kernel function, $K(x,x_i)=\exp\left(-\gamma|x-x_i|^2\right)$,where, γ is the parameter;
- The Sigmoid kernel function $K(x,x_i)=\tanh\left(s(x\bullet x_i)+c\right)$, where, s and c are parameters.

The model figure of SVM [43] [44] [45] is shown in Figure 5. SVM is essentially the classifier for two types. Solving multiple classification problems needs to make more appropriate classifier.

There are two main methods for SVM to structure the classifier for multiple classifications. One is the direct method, namely modify the objective function to use an optimization problem to solve the multiple classification parameters. This method is of high computational complexity. Another method is the indirect method. Combining multiple two-classifier constructs multiple classification classifiers.

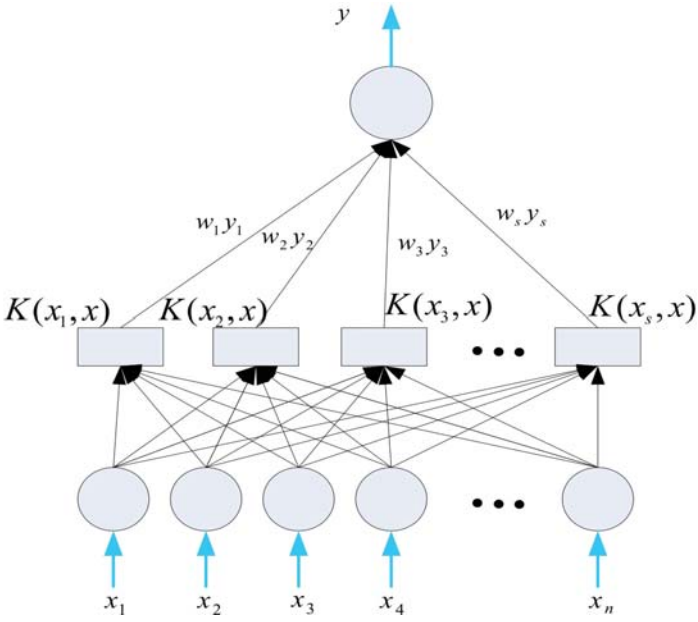


Figure 5. Support Vector Machine (SVM).

The method has two ways:

- One-Against-One: Build a hyper-plane between any two classes, to the problem of k classes needing to build $k\times(k-1)/2$ classification planes.
- One-Against-the-Rest: The classification plane is built between one category and other multiple categories, to the problem of k classes only needing to build k classification planes.

We will use two methods of "One-Against-One" and "One-Against-the-Rest" for the experiment and choose the method with better effect to construct the multiple classification classifiers of SVM.

6) Multiple classification decision fusion classifier (MCDFC)—hybrid HBPNNs HRBFNNs-SVM classifier:- The different classifiers have different performance. Fusion of multiple classifiers integrating their respective characteristics can make classification effect and robustness further improvement.

Feature fusion and decision-making fusion are of two main methods of classifier fusion. Feature fusion has large computation to be not easy to achieve, therefore, we adopt the decision-making fusion. The model figure of MCDFC is shown in Figure 6.

We use the weighted voting for decision fusion of each classifier:

$$w_i = \begin{cases} \log\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right), & \varepsilon_i \le 0.5 \\ 0, & \varepsilon_i > 0.5 \end{cases} \tag{35}$$

where, w_i is the weight of each classifier for the vote of classification result, ε_i is variable. The final classification result is concluded by each classifier according to the following weighted voting formula:

$$f_t(x) = \arg\max_{y \in Y} \sum_{i=1}^{n} w_i \left[f_i(x) = y \right] \tag{36}$$

where, $f_t(x)$ is the final classification result and corresponding to the category y with the maximum, $f_i(x)$ is the classification result of the ith classifier, x is the input, y Y □ and Y is the category set. $[f_i(x) = y]$ indicates that the classification result of the ith classifier meeting the conditions is the category y and combines with the voting weight w_i of the classifier

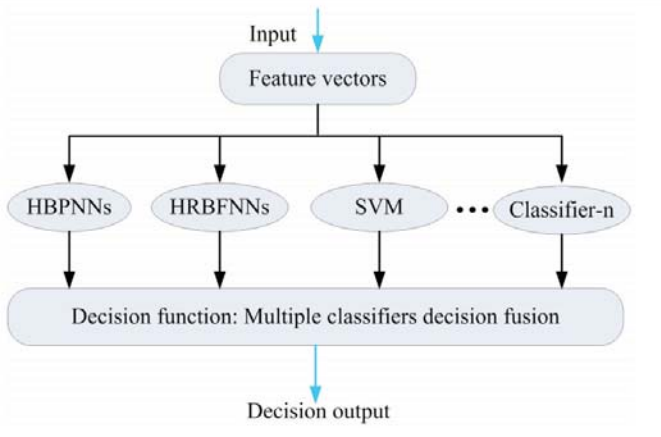


Figure 6. Multiple classification decision fusion classifier (MCDFC).

DESIGNING THE CLASSIFIER COMBINING UNSUPERVISED AND SUPERVISED LEARNING

Supervised learning systems are domain-specific and annotating a large-scale corpus for each domain is very expensive [46]. Recently, semi-supervised learning, which uses a large amount of unlabeled data together with labeled data to build better learners, has attracted more and more attention in pattern recognition and classification [47]. In the paper we design a novel classifier of semi-supervised learning, namely combining unsupervised and supervised learning—deep belief network embedded with Softmax regress (DBNESR) for FR. DBNESR first learns hierarchical representations of feature by greedy layer-wise unsupervised learning in a feed-forward (bottom-up) and back-forward (top-down) manner [48] and then makes more efficient classification with Softmax regress by supervised learning. Deep belief network (DBN) is a representative deep learning algorithm, has deep architecture that is composed of multiple levels of non-linear operations [49], which is expected to perform well in semi-supervised learning, because of its capability of modeling hard artificial intelligent tasks [50]. Softmax regression is a generalization of the logistic regression in many classification problems.

1) Problem formulation:- The dataset is represented as a matrix:

$$X = \left[X^1, X^2, \cdots, X^{N+M}\right] = \begin{bmatrix} x_1^1 & x_1^2 & \cdots & x_1^{N+M} \\ x_2^1 & x_2^2 & \cdots & x_2^{N+M} \\ \vdots & \vdots & \ddots & \vdots \\ x_D^1 & x_D^2 & \cdots & x_1^{N+M} \end{bmatrix} \tag{37}$$

where, N is the number of training samples, M is the number of test samples, D is the number of feature values in the dataset. Each column of X corresponds to a sample X. A sample which has all features is viewed as a vector in $\mathbb{R}^D$, where the jth coordinate corresponds to the jth feature.

Let Y be a set of labels correspond to L labeled training samples and is denoted as:

$$Y^L = \left[Y^1, Y^2, \cdots, Y^L\right] = \begin{bmatrix} y_1^1 & y_1^2 & \cdots & y_1^L \\ y_2^1 & y_2^2 & \cdots & y_2^L \\ \vdots & \vdots & \ddots & \vdots \\ y_C^1 & y_C^2 & \cdots & y_C^L \end{bmatrix} \tag{38}$$

where, C is the number of classes. Each column of Y is a vector in $\mathbb{R}^C$, where, the jth coordinate corresponds to the jth class:

$$y_j = \begin{cases} 1 & \text{if } X \in j\text{th class} \\ 0 & \text{if } X \notin j\text{th class} \end{cases} \tag{39}$$

We intend to seek the mapping function $X \to Y^L$ using all the samples in order to determine Y when a new X comes.

2) Softmax regression:- Softmax regression is a generalization of the logistic regression in many classification problems [51]. Logistic regression is for binary classification problems, class tag $Y^{(i)} \in \{0,1\}$. The hypothesis function is as following:

$$h_\phi(X) = \frac{1}{1+\exp\left(-\phi^{\mathrm{T}} X\right)} \tag{40}$$

Training model parameters vector $\phi \in \mathbb{R}^{D+1}$, which can minimize the cost function:

$$J(\phi) = -\frac{1}{L}\left[\sum_{i=1}^{L} Y^{(i)} \log h_\phi\left(X^{(i)}\right) + \left(1-Y^{(i)}\right)\log\left(1-h_\phi\left(X^{(i)}\right)\right)\right] \tag{41}$$

Softmax regression is for many classification problems, class tag $Y^{(i)} \in \{1,2,\cdots,k\}$. It is used for each given sample X, using hypothesis function to estimate the probability value $p(Y=j \mid X)$ for each category j. The hypothesis function is as following:

$$h_\phi\left(X^{(i)}\right) = \begin{bmatrix} p\left(Y^{(i)}=1 \mid X^{(i)};\phi\right) \\ p\left(Y^{(i)}=2 \mid X^{(i)};\phi\right) \\ \vdots \\ p\left(Y^{(i)}=k \mid X^{(i)};\phi\right) \end{bmatrix} = \frac{1}{\sum_{j=1}^{k} e^{\phi_j^T X^{(i)}}} \begin{bmatrix} e^{\phi_1^T X^{(i)}} \\ e^{\phi_2^T X^{(i)}} \\ \vdots \\ e^{\phi_k^T X^{(i)}} \end{bmatrix} \tag{42}$$

where, $\phi_1, \phi_2, \cdots, \phi_k \in \mathbb{R}^{D+1}$ denote model parameters vector, the cost function is as following:

$$J(\phi) = -\frac{1}{L}\left[\sum_{i=1}^{L}\sum_{j=1}^{k} 1\left\{Y^{(i)}=j\right\} \log \frac{e^{\phi_j^T X^{(i)}}}{\sum_{l=1}^{k} e^{\phi_l^T X^{(i)}}}\right] \tag{43}$$

where, $1\{\cdot\}$ denotes:

$1\{\text{The value of expression is true}\} = 1$ or $1\{\text{The value of expression is false}\} = 0$ (44)

There are no closed form solutions to minimize the cost function Equation (43) at present. Therefore, we use the iterative optimization algorithm (for example, gradient descent method or L-BFGS). After derivation we get gradient formula is as following:

$$\nabla_{\phi_j} J(\phi) = -\frac{1}{L}\sum_{i=1}^{L}\left[X^{(i)}\left(1\left\{Y^{(i)} = j\right\} - p\left(Y^{(i)} = j \mid X^{(i)};\phi\right)\right)\right] \tag{45}$$

Then make the following update operation:

$$\phi_j := \phi_j - \alpha\nabla_{\phi_j} J(\phi), j = 1,\cdots,k \tag{46}$$

where, α denotes learning rate.

3) Deep belief network embedded with Softmax regress (DBNESR):- DBN uses a Markov random field Restricted Boltzmann Machine (RBM) [52] [53] of unsupervised learning networks as building blocks for the multi-layer learning systems and uses a supervised learning algorithm named BP (back propagation) for fine-tuning after pre-training. Its architecture is shown in Figure 7. The deep architecture is a fully interconnected directed belief nets with one input layer v^1, $W = \{W, W, \cdots, W\}$ hidden layers $h^1, h^2, \cdots, h^N$ and one labeled layer at the top. The input layer v^1 has D units, equal to the number of features of samples. The label layer has C units, equal to the number of classes of label vector Y. The numbers of units for hidden layers, currently, are pre-defined according to the experience or intuition. The seeking of the mapping function, here, is transformed to the problem of finding the parameter space $W = \{W^1, W^2, \cdots, W^N\}$ for the deep architecture [54].

The semi-supervised learning method based on DBN architecture can be divided into two stages: First, DBN architecture is constructed by greedy layer-wise unsupervised learning using RBM as building blocks. All samples are utilized to find the parameter space W with N layers. Second, DBN architecture is trained according to the log-likelihood using gradient descent method. As it is difficult to optimize a deep architecture using supervised learning directly, the unsupervised learning stage can abstract the hierarchical representations feature effectively, and prevent over-fitting of the supervised training. The algorithm BP is used pass the error top-down for fine-tuning after pre-training.

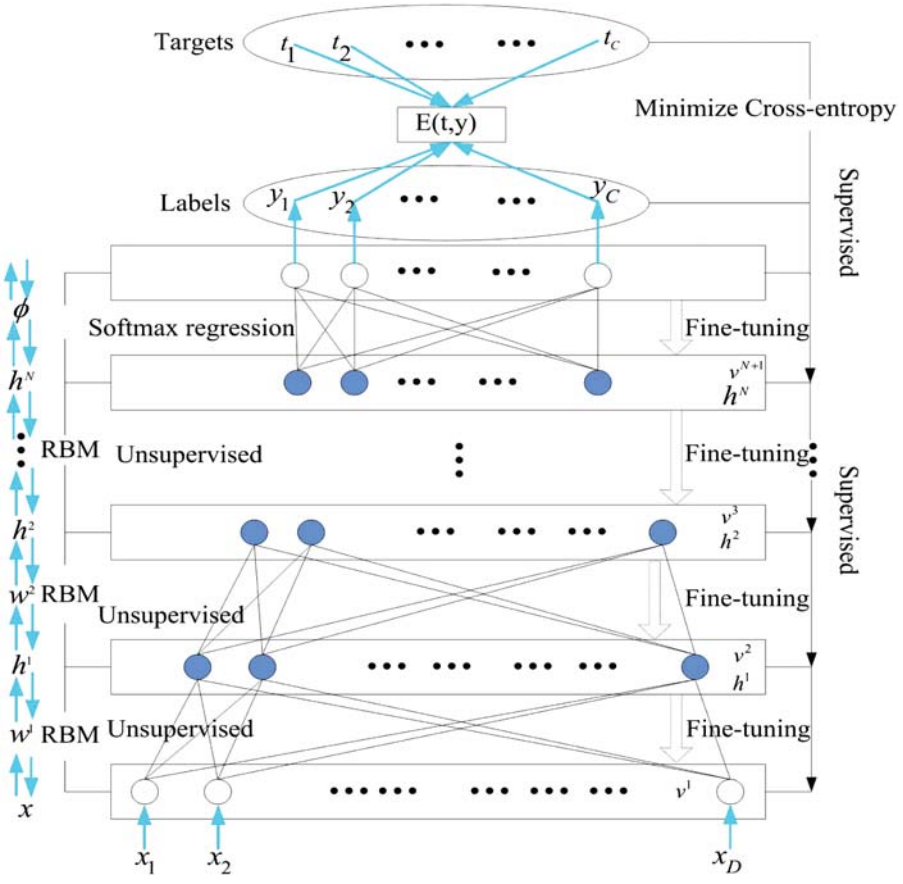


Figure 7. Architecture of deep belief network embedded with Softmax regress (DBNESR).

For unsupervised learning, we define the energy of the joint configuration (h^{k-1}, h^k)as [50]:

$$\left(h^{k-1}, h^k; \theta\right) = -\sum_{i=1}^{D_{k-1}} \sum_{j=1}^{D_k} w_{ij}^k h_i^{k-1} h_j^k - \sum_{i=1}^{D_{k-1}} b_i^{k-1} h_i^{k-1} - \sum_{j=1}^{D_k} c_j^k h_j^k \tag{47}$$

where, $\theta = (W, b, c)$are the model parameters: w_{ij}^k is the symmetric interaction term between unit i in the layer $h^{k-1}$and unit j in the layer h^k, $k = 1, \cdots, N-1$. $b_i^{k-1}$is the ith bias of layer h^{k-1} and $c_j^k$is the jth bias of layer h^k. D^k is the number of units in the kth layer. The network assigns a probability to every possible data via this energy function. The probability of a training data can be raised by adjusting the weights and biases to lower the energy of

that data and to raise the energy of similar, confabulated data that h^k would prefer to the real data. When we input the value of h^k, the network can learn the content of $h^{k-1}$by minimizing this energy function.

$$P\left(h^{k-1};\theta\right)=\frac{1}{Z(\theta)}\sum_{h^k}\exp\left(-E\left(h^{k-1},h^k;\theta\right)\right) \tag{48}$$

$$Z(\theta)=\sum_{h^{k-1}}\sum_{h^k}\exp\left(-E\left(h^{k-1},h^k;\theta\right)\right) \tag{49}$$

where, Z(θ) denotes the normalizing constant. The conditional distributions over h^k and $h^{k-1}$are given as:

$$p\left(h^k\mid h^{k-1}\right)=\prod_j p\left(h_j^k\mid h^{k-1}\right) \tag{50}$$

$$p\left(h^{k-1}\mid h^k\right)=\prod_i p\left(h_i^{k-1}\mid h^k\right) \tag{51}$$

The probability of turning unit j is a logistic function of the states h^{k-1} and w_{ij}^k :

$$p\left(h_j^k=1\mid h^{k-1}\right)=sigm\left(c_j^k+\sum_i w_{ij}^k h_i^{k-1}\right) \tag{52}$$

The probability of turning unit i is a logistic function of the states of h^k and w_{ij}^k :

$$p\left(h_i^{k-1}=1\mid h^k\right)=sigm\left(b_i^{k-1}+\sum_j w_{ij}^k h_j^k\right) \tag{53}$$

where, the logistic function been chosen is the sigmoid function:

$$sigm(x)=1/\left(1+e^{-x}\right) \tag{54}$$

The derivative of the log-likelihood with respect to the model parameter w^k can be obtained from Equation (48):

$$\frac{\partial\log p\left(h^{k-1}\right)}{\partial w_{ij}^k}=\left\langle h_i^{k-1}h_j^k\right\rangle_{p_0}-\left\langle h_i^{k-1}h_j^k\right\rangle_{p_{Model}} \tag{55}$$

where, $\langle\cdot\rangle_{p_0}$ denotes an expectation with respect to the data distribution and Model $\langle\cdot\rangle_{p_{Model}}$ denotes an expectation with respect to the distribution defined by the model [55]. The expectation $\langle\cdot\rangle_{p_{Model}}$ cannot be computed analytically. In practice, $\langle\cdot\rangle_{p_{Model}}$is replaced by $\langle\cdot\rangle_{p_1}$, which denotes a distribution of samples when the feature detectors are being driven by reconstructed h^{k-1}. This is an approximation to the gradient of a different objective function, called the

contrastive divergence (CD) [56] [57] [58] [59]. Using Kullback-Leibler distance to measure two probability distribution "diversity", represented by $KL(P \| P')$, is shown in Equation (56):

$$CD_n = KL(p_0 \| p_\infty) - KL(p_n \| p_\infty) \tag{56}$$

where, p_0 denotes joint probability distribution of initial state of RBM network, p_n denotes joint probability distribution of RBM network after n transformations of Markov chain Monte Carlo(MCMC), $p\infty$ denotes joint probability distribution of RBM network at the ends of MCMC. Therefore, CDn can be regarded as a measure location for n p between p_0 and p_∞. It constantly assigns p_n to p_n and gets new p_0 and p_n. The experiments show that CD_n will tend to zero and the accuracy is approximate of MCMC after making slope for r times for correction parameter θ. The training process of RBM is shown in Figure 8.

We can get Equation (57) by training process of RBM using contrastive divergence:

$$\Delta w_{ij}^k = \eta\left(\left\langle h_i^{k-1} h_j^k \right\rangle_{P_0} - \left\langle h_i^{k-1} h_j^k \right\rangle_{P_1}\right) \tag{57}$$

where, η is the learning rate. Then the parameter can be adjusted through:

$$w_{ij}^k = \mu w_{ij}^k + \Delta w_{ij}^k \tag{58}$$

where, μ is the momentum.

The above discussion is based on the training of the parameters between hidden layers with one sample x. For unsupervised learning, we construct the deep architecture using all samples by inputting them one by one from layer h^0, train the parameters between h^0 and h^1. Then 1 h is constructed, the value of h^1 is calculated by h^0 and the trained parameters between h^0 and h^1. We also can use it to construct the next layer h^2 and so on. The deep architecture is constructed layer by layer from bottom to top. In each time, the parameter space W^K is trained by the calculated data in the (k −1th) layer. Accord to the W^K calculated above, the layer h^k is obtained as below for a sample x fed from layer h^0:

$$h_j^k(x) = sigm\left(c_j^k + \sum_{i=1}^{D_{k-1}} w_{ij}^k h_i^{k-1}(x)\right), \quad j = 1, \cdots, D_k; k = 1, \cdots, N-1 \tag{59}$$

For supervised learning, the DBM architecture is trained by C labeled data. The optimization problem is formulized as:

$$argminerr = -\sum_k p_k \log \hat{p}_k - \sum_k (1-p_k)\log(1-\hat{p}_k) \tag{60}$$

namely, to minimize cross-entropy. Where, p_k denotes the real label probability and $\hat{P}_k$ denotes the model label probability

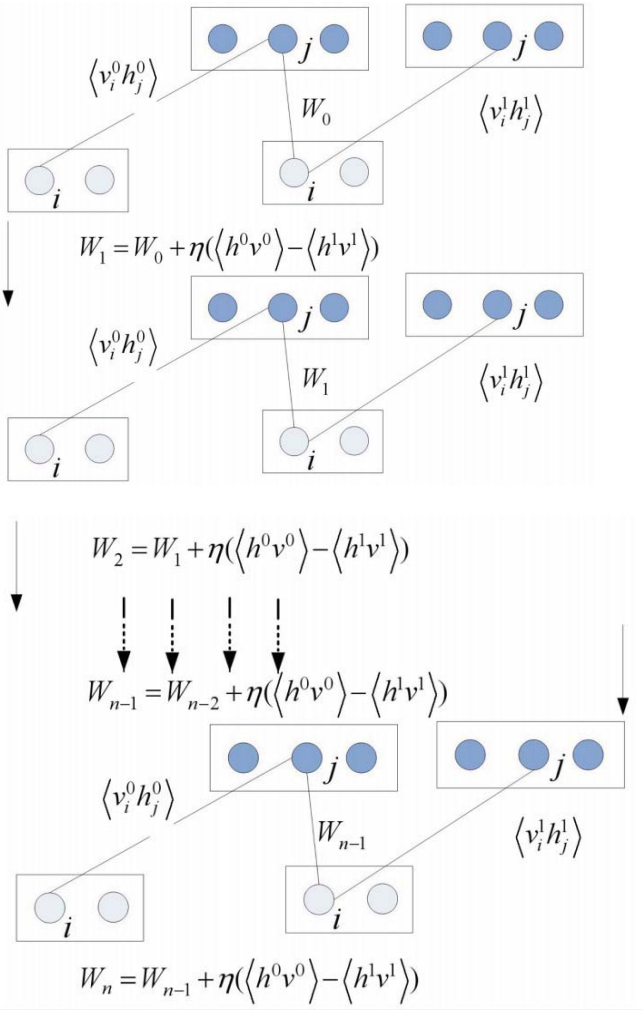


Figure 8. Training process of RBM using contrastive divergence.

The greedy layer-wise unsupervised learning is just used to initialize the parameter of deep architecture, the parameters of the deep architecture are updated based on Equation (58). After initialization, real values are used in all the nodes of the deep architecture. We use gradient-descent through the whole deep architecture to retrain the weights for optimal classification.

EXPERIMENTS

1) ***Face Recognition Databases:-*** We selected some typical databases of images, for example ORL Face Database, which consists of 10 different images for each of the 40 distinct individuals. Each people is imaged in different facial expressions and facial details under varying lighting conditions at different times. All the pictures are captured with a dark background and the individuals are in an upright and frontal position; the facial gestures are not identical, expressions, position, angle and scale are some different; The depth rotation and plane rotary can be up to 20˚, the scale of faces also has as much as 10% change. For each face database as above, we randomly choose a part of images as training data and the remaining as testing data. In this paper, in order to reflect the universality and high efficiency of all classification algorithms we randomly choose about 50% of each individual image as training data and the rest as testing data. At first all images will be made preprocessing and feature extraction.

All the experiments are carried out in MATLAB R2010b environment running on a desktop with Intel® Core™2 TM2 Duo CPU T6670 @2.20GHz and 4.00 GB RAM.

2) ***Relevant experiments:-*** *Experiment 1*. In this experiment, we use median filtering to make smoothing denoising for images preprocessing and get the sample Figure 9 as following: Seeing from the comparison of face images, the face images after filtering eliminate most of noise interference.

Experiment 2. In this experiment, we make histogram equalization for the images preprocessing and get the sample figures as following:

From Figure 10 and Figure 11 we can see: After histogram equalization, the distribution of image histogram is more uniform, the range of gray increases some and the contrast has also been stronger. In addition, the image after histogram equalization basically eliminated the influence of illumination, expanded the representation range of pixel gray, improved the contrast of image, made the facial features more evident and is conducive to follow-up feature extraction and FR

Figure 9. Face images with median filtering versus no median filtering.

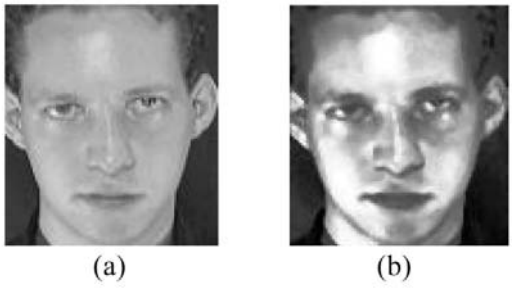
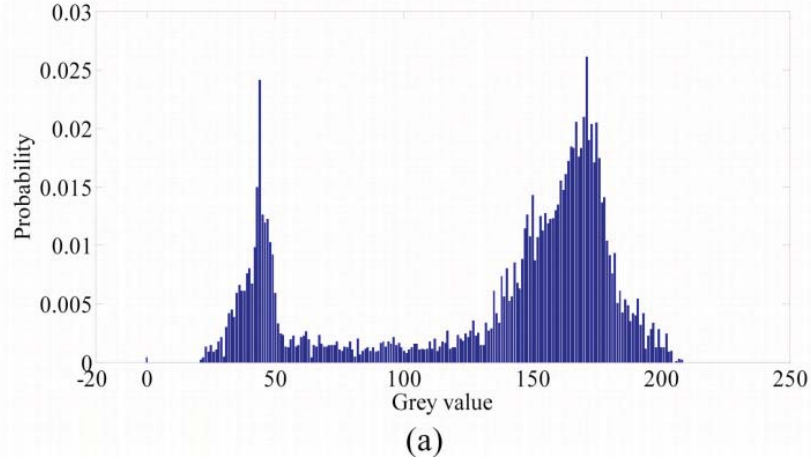


Figure 10. Face images before histogram equalization versus after histogram equalization. (a) Original image; (b) Image after histogram equalization.



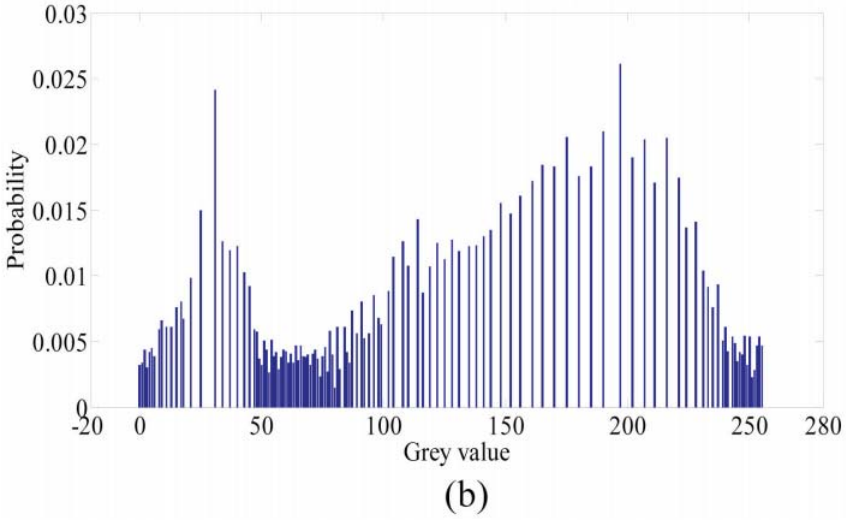


Figure 11. Histogram of original image versus histogram of image after histogram equalization. (a) Histogram of original image; (b) Histogram of image after histogram equalization.

Experiment 3. In this experiment, we employ multi-scale two-dimensional wavelet transform to generate the initial geometric features for representing face images. By the experiment we get the sample figures as following: From Figure 12 we can see: Although for compression of images (or dimensionality-reduced), LL sub-graph information capacity has decreased some, but still has very high resolution and the energy of wavelet domain did not decrease a lot. LL sub-graph can be well made for the follow-up feature extraction.

Experiment 4. In this experiment, we extract features respectively with PCA and 2D-PCA and compare their effects as following:

From Figure 13 we can see that the first several principal components contribution rates extracted with 2D-PCA are higher than the first several principal components contribution rates extracted with PCA. From Figure 14 we can see when the principal components are extracted for 20, the principal component contribution rate of 2D-PCA is greater than 90%, while the principal component contribution rate of PCA is less than 80%. Accordingly, 2D-PCA can use less principal component to better describe the image than PCA.

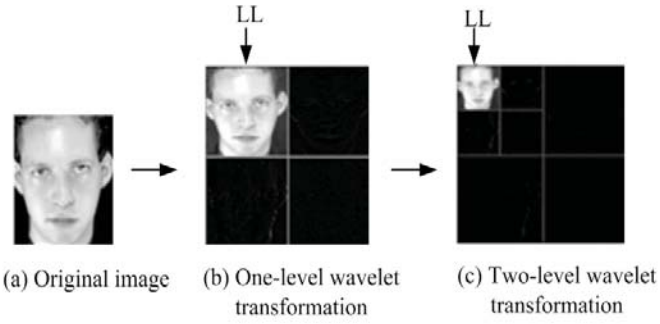


Figure 12. Multi-scal two-dimensional wavelet transform.

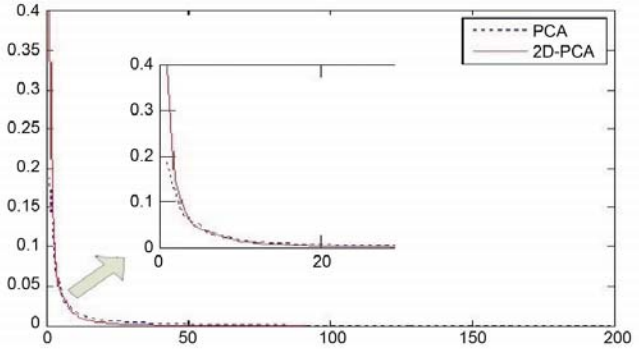


Figure 13. Principal component energy figure. Abscissa: principal components; ordinate: energy value.

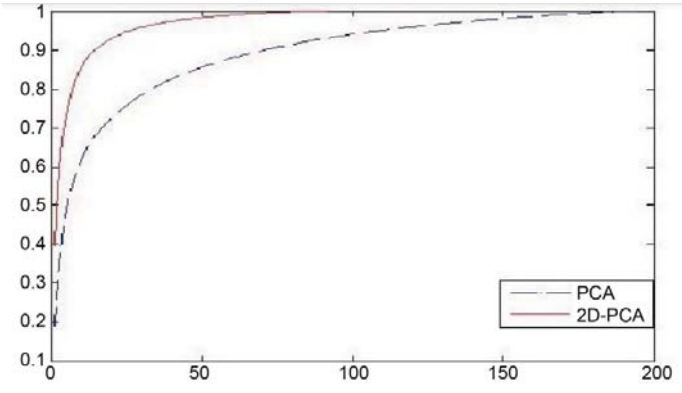


Figure 14. Principal component accumulation contribution rate. Abscissa: principal components; ordinate: total energy value.

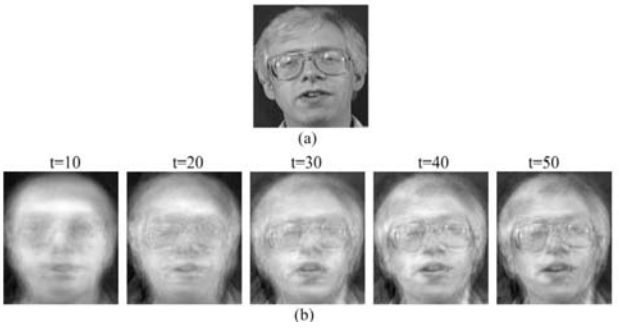
Figure 15 is the comparing results image of reconstruction with the feature respectively extracted with PCA and 2D-PCA. We can see that the images of reconstruction by 2D-PCA are clearer than the images of reconstruction by PCA when extracting same number of principal components. The reconstruction face extracted 50 principal components by 2D-PCA is almost same clear with the original image. 2D-PCA has better effect than PCA.

Experiment 5. In this experiment, we compare the recognition rate of the methods respectively based on PCA + BP, WT + PCA + BP, PCA + HBPNNs and WT + PCA + HBPNNs. The experiment is repeated many times and takes the average recognition rate. The experimental results are shown in Table 1. As shown in Table 1, Recognition rates of HBPNNs are improved very greatly being compared to BP, in the same classifier (BP or HBPNNs) recognition rates of the methods based on WT + PCA are higher than them based on PCA.

Experiment 6. This experiment compares the recognition rate of the methods respectively based on WT + 2D-PCA + RBF and WT + 2D-PCA + HRBFNNs. The experiment is repeated for many times and takes the average recognition rate. The experimental results are shown in Table 2.

As shown in Table 2, Recognition rates of HRBFNNs are improved very greatly being compared to RBF. Therefore, HRBFNNs being used for FR is more feasible.

Experiment 7. Because SVM is essentially the classifier for two types, solving the multiple classification problems needs to reconstruct more appropriate classifier. We will use two methods of "One-Against-One" and "One-Against-the Rest" for the experiment and choose the method with better effect to construct the multiple classification classifiers of SVM. The experiment is repeated for 20 times and takes the average recognition rate. The experimental results are shown in Table 3.



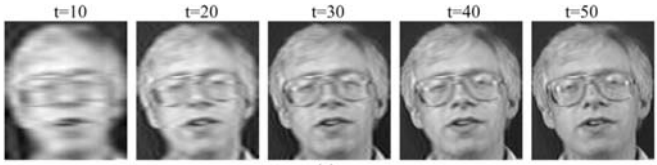


Figure 15. Reconstructed images with 2D-PCA and PCA versus original image (t: principal component number). (a) Original image; (b) PCA principal component reconstruction images; (c) 2D-PCA principal component reconstruction images.

Table 1. Average recognition rates of different recognition methods.

Serial number	Recognition method	Recognition rate/%
1	PCA + BP	66.2
2	WT + PCA + BP	67.29
3	PCA + HBPNNs	91.7
4	WT + PCA + HBPNNs	93.3

Table 2. Average recognition rates of different recognition methods.

Serial number	Recognition method	Recognition rate/%
1	WT + 2D-PCA + RBF	90.5
2	WT + 2D-PCA + HRBFNNs	95.5

As shown in Table 3, "One-Against-One" SVM has higher recognition rate than "One-Against-the-Rest" SVM and at the same time has lower wrong number. Therefore, we use the way of "One-Against-One" to reconstruct the SVM classifier to realize FR.

Experiment 8. In the paper we construct the multiple classification decision fusion classifier (MCDFC)—hybrid HBPNNs-HRBFNNs-SVM classifier. In this experiment, in order to show the efficiency of MCDFC, we first make recognition experiment respectively based on HBPNNs, HRBFNNs and SVM, then use the decision function to make fusions for classification results of three classifiers and get classification results of MCDFC. The experiment is repeated for 20 times and the experimental results are shown in Table 4 and in Figure 16.

As shown in Figure 16, the recognition effect of MCDFC is always not lower than the average level of other three kinds of classifiers and in almost all cases the effect of MCDFC is optimal.

To eliminate the error of single experiment and greatly reduce the random uncertainty, Table 5 lists the average recognition rates of each classifier for 20 times and the variance of each classifier. It can be seen from the experimental results that the multiple classification decision fusion classifier (MCDFC)—hybrid HBPNNs-HRBFNNs-SVM classifier has the best effect for FR, has the minimum variance, can effectively improve the generalization ability and has high stability.

Experiment 9. In this experiment, in order to validate the performance of our proposed algorithm—DBNESR is optimal for FR, we compare our proposed algorithm with some other methods such as BP, HBPNNs, RBF, HRBFNNs, SVM and MCDFC.

Table 3. Average recognition rates of different recognition methods

Serial number	Recognition method	Recognition rate/%	Wrong number
1	One-Against-One SVM	95.05	9.9
2	One-Against-the-Rest SVM	90.45	19.1

Table 4. Recognition rates of different recognition methods for 20 times

Algorithm	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
HBPNNs	0.92	0.915	0.905	0.875	0.905	0.905	0.93	0.95	0.91	0.9	0.935	0.9	0.9	0.91	0.91	0.91	0.9	0.925	0.925	0.91
HRBFNNs	0.935	0.905	0.945	0.9	0.95	0.94	0.97	0.935	0.945	0.935	0.95	0.945	0.94	0.95	0.92	0.94	0.935	0.95	0.935	0.945
SVM	0.93	0.92	0.945	0.91	0.945	0.915	0.955	0.945	0.94	0.915	0.95	0.955	0.92	0.955	0.935	0.935	0.945	0.945	0.94	0.92
MCDFC	0.93	0.93	0.945	0.915	0.95	0.94	0.97	0.94	0.95	0.935	0.955	0.94	0.94	0.955	0.925	0.94	0.94	0.95	0.94	0.94

Table 5. Average recognition rates and variances of different recognition methods.

Serial number	Recognition method	Average recognition rate/%	Variance
1	HBPNNs	91.2	0.0002537
2	HRBFNNs	93.85	0.0002476
3	SVM	93.6	0.0002147
4	MCDFC	94.15	0.0001424

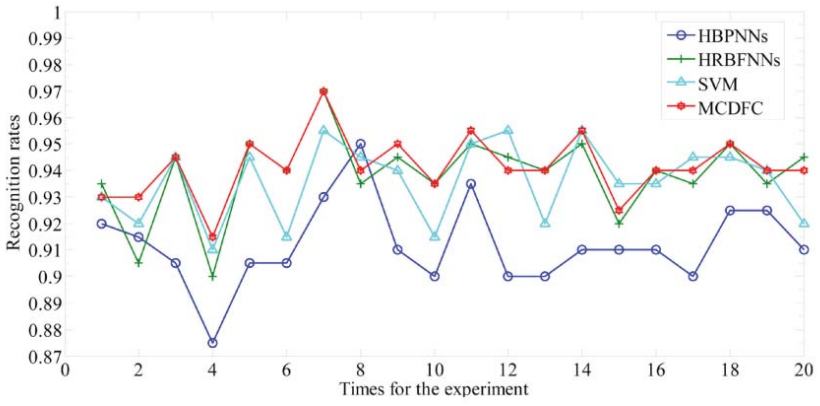


Figure 16. The curve figures of recognition rates of different recognition methods.

In the experiment we set up different hidden layers and each hidden layer with different neurons. The architecture of DBNESR is similar with DBN, but with a different loss function introduced for supervised learning stage.

For greedy layer-wise unsupervised learning we train the weights of each layer independently with the different epochs, we also make fine-tuning supervised learning for the different epochs.

All DBNESR structures and learning epochs used in this experiment are separately shown in Table 6. The number of units in input layer is the same as the feature dimensions of the dataset.

Almost all the recognition rates of these DBNESR structures are more than 90%, in particular the effects of the models of 500-1000-40 and 1000-500-40 are best and most stable.

Therefore, the DBNESR structures used in this experiment are 1000-500-40, which represents the number of units in output layer is 40, and in 2 hidden layers are 1000 and 500 respectively.

The learning rate is set to dynamic value, which the initial learning rate is set to 0.1 and becomes smaller as the training error becoming smaller. The experimental results are shown in Table 7, Table 8 and in Figures 17-19.

Table 6. Different hidden layers of DBNESR and learning epochs used in this experiment

Serial number	DBNESR structures	Unsupervised learning epochs	Supervised learning epochs
1	400-200-100-50-20-40	10	1000
2	400-200-100-100-50-40	50	100
3	400-200-300-100-50-40	100	20
4	400-200-300-100-40	50	50
5	400-200-300-200-40	100	20
6	200-200-300-400-40	100	100
7	200-300-400-40	100	100
8	400-300-200-40	100	200
9	400-200-300-40	200	100
10	500-400-40	200	200
11	500-1000-40	200	200
12	1000-500-40	200	200

Table 7. Recognition rates of different recognition methods for 20 times

Algorithm	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
BP	0.65	0.655	0.68	0.675	0.645	0.645	0.805	0.64	0.665	0.635	0.635	0.68	0.625	0.625	0.7	0.8	0.635	0.65	0.628	0.74
HBPNNs	0.92	0.915	0.905	0.875	0.905	0.905	0.93	0.95	0.91	0.9	0.935	0.9	0.9	0.91	0.91	0.91	0.9	0.925	0.925	0.91
RBF	0.905	0.9	0.9	0.875	0.88	0.88	0.915	0.92	0.92	0.915	0.93	0.935	0.9	0.905	0.895	0.895	0.93	0.85	0.91	0.94
HRBFNNs	0.935	0.905	0.945	0.9	0.95	0.94	0.97	0.935	0.945	0.935	0.95	0.945	0.94	0.95	0.92	0.94	0.935	0.95	0.935	0.945
SVM	0.93	0.92	0.945	0.91	0.945	0.915	0.955	0.945	0.94	0.915	0.95	0.955	0.92	0.955	0.935	0.935	0.945	0.945	0.94	0.92
MCDFC	0.93	0.93	0.945	0.915	0.95	0.94	0.97	0.94	0.95	0.935	0.955	0.94	0.94	0.955	0.925	0.94	0.94	0.95	0.94	0.94
DBNESR	0.95	0.95	0.96	0.965	0.945	0.95	0.95	0.96	0.965	0.96	0.95	0.965	0.945	0.95	0.96	0.965	0.95	0.96	0.96	0.965

Table 8. Average recognition rates and variances of different recognition methods

Serial number	Recognition method	Average recognition rate/%	Variance
1	BP	67.06	0.0028
2	HBPNNs	91.2	0.0002537
3	RBF	90.5	0.0005
4	HRBFNNs	93.85	0.0002476
5	SVM	93.6	0.0002147
6	MCDFC	94.15	0.0001424
7	DBNESR	95.63	0.0000523

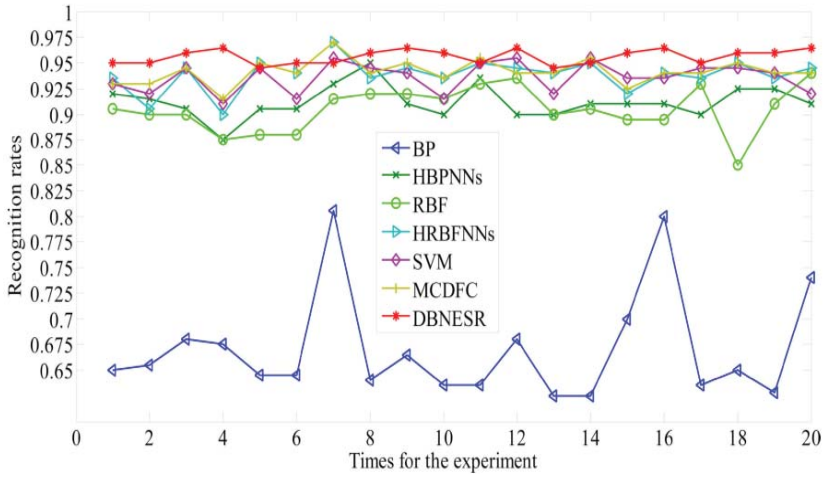


Figure 17. The curve figures of recognition rates of different recognition methods.

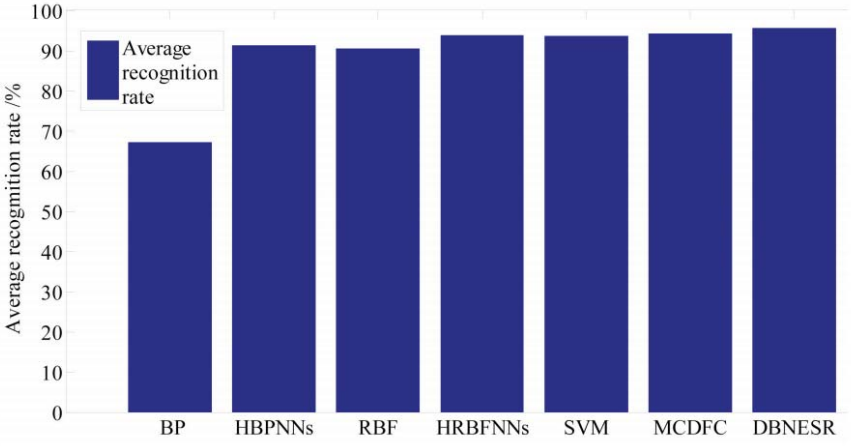


Figure 18. The bar charts of average recognition rate of different recognition methods.

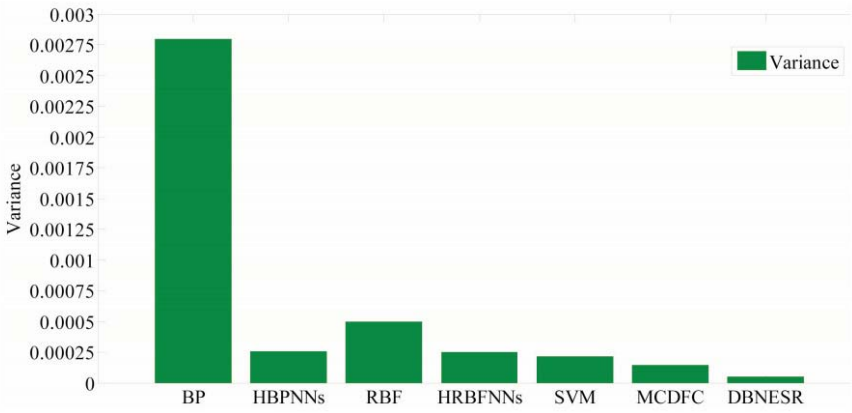


Figure 19. The bar charts of variance of different recognition methods.

As shown in Table 7, Table 8 and in Figures 17-19, our proposed algorithm—DBNESR is optimal for FR, in almost all cases the recognition rates of DBNESR is highest and most stable, namely there is the largest average recognition rate and the smallest variance.

CONCLUSION

The conducted experiments validate that the proposed algorithm DBNESR is optimal for face recognition with the highest and most stable recognition rates, that is, it successfully implements hierarchical representations' feature deep learning for face recognition. You can also be sure that it reflects hierarchical representations of feature by DBNESR in terms of its capability of modeling other artificial intelligent tasks, which is also what we're going to do in the future.

ACKNOWLEDGEMENTS

This research was funded by the National Natural Science Foundation (Grand 61171141, 61573145), the Public Research and Capacity Building of Guangdong Province (Grand 2014B010104001), the Basic and Applied Basic Research of Guangdong Province (Grand 2015A030308018), the Main Project of the Natural Science Fund of Jiaying University (grant number 2017KJZ02) and the key research bases being jointly built by

provinces and cities for humanities and social science of regular institutions of higher learning of Guangdong province (Grant number 18KYKT11), the cooperative education program of Ministry of Education (Grant number 201802153047), the college characteristic innovation project of Education Department of Guangdong province in 2019 (Grant number 2019KTSCX169), the authors are greatly thanks to these grants.

REFERENCES

1. Wright, J., Ma, Y., Mairal, J., et al. (2010) Sparse Representation for Computer Vision and Pattern Recognition. Proceedings of the IEEE, 98, 1031-1044. https://doi.org/10.1109/JPROC.2010.2044470
2. Wang, S.J., Yang, J., Sun, M.F., et al. (2012) Sparse Tensor Discriminant Color Space for Face Verification. IEEE Transactions on Neural Networks and Learning Systems, 23, 876-888. https://doi.org/10.1109/TNNLS.2012.2191620
3. Xu, Y., Zhong, A., Yang, J. and Zhang, D. (2010) LPP Solution Schemes for Use with Face Recognition. Pattern Recognition, 43, 4165-4176. https://doi.org/10.1016/j.patcog.2010.06.016
4. Fan, Z.Z., Xu, Y., Zuo, W.M., Yang, J., et al. (2014) Modified Principal Component Analysis: An Integration of Multiple Similarity Subspace Models. IEEE Transactions on Neural Networks and Learning Systems, 25, 1538-1552. https://doi.org/10.1109/TNNLS.2013.2294492
5. Yang, W.K., Sun, C.Y. and Zhang, L. (2011) A Multi-Manifold Discriminant Analysis Method for Image Feature Extraction. Pattern Recognition, 44, 1649-1657. https://doi.org/10.1016/j.patcog.2011.01.019
6. Xu, Y., Li, X., Yang, J., et al. (2013) Integrating Conventional and Inverse Representation for Face Recognition. IEEE Transactions on Cybernetics, 44, 1738-1746.
7. Wang, S.J., Zhou, C.G., Chen, Y.H., et al. (2011) A Novel Face Recognition Method Based on Sub-Pattern and Tensor. Neurocomputing, 74, 3553-3564. https://doi.org/10.1016/j.neucom.2011.06.017
8. Zhang, H.Z., Zhang, Z., Li, Z.M., Chen, Y. and Shi, J. (2014) Improving Representation Based Classification for Robust Face Recognition. Journal of Modern Optics, 61, 961-968. https://doi.org/10.1080/09500340.2014.915064
9. Wang, S.J., Chen, H.L., et al. (2014) Face Recognition and Micro-Expression Recognition Based on Discriminant Tensor Subspace Analysis Plus Extreme Learning Machine. Neural Processing Letters, 39, 25-43. https://doi.org/10.1007/s11063-013-9288-7
10. Wan, W.G., Zhou, Z.H., Zhao, J.W. and Cao, F.L. (2015) A Novel Face Recognition Method: Using Random Weight Networks and Quasi-Singular Value Decomposition. Neurocomputing, 151, 1180-1186. https://doi.org/10.1016/j.neucom.2014.06.081

11. Zhao, Z. and Liu, H. (2007) Spectral Feature Selection for Supervised and Unsupervised Learning. Proceedings of the 24th International Conference on Machine Learning, Corvails, June 2007, 1151-1157. https://doi.org/10.1145/1273496.1273641
12. Cai, D., Zhang, C.Y. and He, X.F. (2010) Unsupervised Feature Selection for Multi-Cluster Data. Proceedings of the 16th SIGKDD International Conference on Knowledge Discovery and Data Mining, July 2010, Washington DC, 333-342. https://doi.org/10.1145/1835804.1835848
13. Zhao, Z., Wang, L. and Liu, H. (2010) Efficient Spectral Feature Selection with Minimum Redundancy. Proceedings of the 24th AAAI Conference on Artificial Intelligence, July 2010, Atlanta, 673-678.
14. Hou, C.P., Nie, F.P. and Li, X.L. (2011) Joint Embedding Learning and Sparse Regression: A Framework for Unsupervised Feature Selection. IEEE Transactions on Cybernetics, 44, 793-804.
15. Ghazali, K.H., Mansor, M.F. and Mustafa, M.M. (2007) A Feature Extraction Technique Using Discrete Wavelet Transform for Image Classification. Proceedings of the 5th Student Conference on Research and Development, Selangor, Malaysia, 12-11 December 2007, 1-4. https://doi.org/10.1109/SCORED.2007.4451366
16. Hu, H.F. (2011) Variable Lighting Face Recognition Using Discrete Wavelet Transform. Pattern Recognition Letters, 32, 1526-1534. https://doi.org/10.1016/j.patrec.2011.06.009
17. Jemai, O., Zaied, M., Amar, C.B. and Alimi, A.M. (2010) FBWN: An Architecture of Fast Beta Wavelet Networks for Image Classification. Proceedings of the 2010 International Joint Conference on Neural Networks (IJCNN), Barcelona, 18-23 July 2010, 1-8. https://doi.org/10.1109/IJCNN.2010.5596876
18. Huang, K. and Aviyente, S. (2008) Wavelet Feature Selection for Image Classification. IEEE Transactions on Image Processing, 17, 1709-1719. https://doi.org/10.1109/TIP.2008.2001050
19. Zhao, M., Li, P. and Liu, Z. (2008) Face Recognition Based on Wavelet Transform Weighted Modular PCA. 2008 Congress on Image and Signal Processing, Sanya, 27-30 May 2008, 589-593. https://doi.org/10.1109/CISP.2008.138
20. [20] Zhang, B.L., Zhang, H.H. and Ge, S.S. (2004) Face Recognition by Applying Wavelet Subband Representation and Kernel Associative Memory. IEEE Transactions on Neural Networks, 15, 166-177.

21. https://doi.org/10.1109/TNN.2003.820673
22. Nefian A.V., Hayes, M.H. (1998) Face Detection and Recognition Using Hidden Markov Models. Proceedings 1998 International Conference on Image Processing, ICIP98 (Cat. No. 98CB36269), Chicago, 7-7 October 1998, 141-145. https://doi.org/10.1109/ICIP.1998.723445
23. Vlasenko, B., Prylipko, D., Böck, R. and Wendemuth, A. (2013) Modeling Phonetic Pattern Variability in Favor of the Creation of Robust Emotion Classifiers for Real-Life Applications. Computer Speech & Language, 28, 483-500. https://doi.org/10.1016/j.csl.2012.11.003
24. He, L., Lech, M., Maddage, N.C. and Allen, N.B. (2011) Study of Empirical Mode Decomposition and Spectral Analysis for Stress and Emotion Classification in Natural Speech. Biomedical Signal Processing and Control, 6, 139-146. https://doi.org/10.1016/j.bspc.2010.11.001
25. Suykens, J.A.K. and Vandewalle, J. (1999) Least Squares Support Vector Machine Classifiers. Neural Processing Letters, 9, 293-300. https://doi.org/10.1023/A:1018628609742
26. Lee, C.-C., Mower, E., Busso, C., Lee, S. and Narayanan, S. (2011) Emotion Recognition Using a Hierarchical Binary Decision Tree Approach. Speech Communication, 53, 1162-1171. https://doi.org/10.1016/j.specom.2011.06.004
27. Igelnik, B. and Pao, Y.H. (1995) Stochastic Choice of Basis Functions in Adaptive Function Approximation and the Functional-Link Net. IEEE Transactions on Neural Networks, 6, 1320-1329. https://doi.org/10.1109/72.471375
28. Pao, Y.H., Park, G.H. and Sobajic, D.J. (1994) Learning and Generalization Characteristics of the Random Vector Functional-Link Net. Neurocomputing, 6, 163-180. https://doi.org/10.1016/0925-2312(94)90053-1
29. Xu, Y., Zhang, X.F. and Gai, H.C. (2011) Quantum Neural Networks for Face Recognition Classifier. Procedia Engineering, 15, 1319-1323. https://doi.org/10.1016/j.proeng.2011.08.244
30. Reddy, K.R.L., Babu, G.R. and Kishore, L. (2010) Face Recognition Based on Eigen Features of Multi Scaled Face Components and an Artificial Neural Network. Procedia Computer Science, 2, 62-74. https://doi.org/10.1016/j.procs.2010.11.009
31. Suka, H.-I., Lee, S.-W., Shen, D.G. and the Alzheimer's Disease Neuroimaging Initiative (2014) Hierarchical Feature Representation

and Multimodal Fusion with Deep Learning for AD/MCI Diagnosis. Neuroimage, 101, 569-582. https://doi.org/10.1016/j.neuroimage.2014.06.077

32. Han, H., Shan, S.G., Chen, X.L. and Gao, W. (2013) A Comparative Study on Illumination Preprocessing in Face Recognition. Pattern Recognition, 46, 1691-1699. https://doi.org/10.1016/j.patcog.2012.11.022
33. Chao, S. (2013) Research and Implement of Face Recognition Based on Neural Network. South China University of Technology, Guangzhou.
34. Längkvist, M., Karlsson, L. and Loutfi, A. (2014) A Review of Unsupervised Feature Learning and Deep Learning for Time-Series Modeling. Pattern Recognition Letters, 42, 11-24. https://doi.org/10.1016/j.patrec.2014.01.008
35. Xu, Y.J., You, T. and Du, C.L. (2015) An Integrated Micromechanical Model and Bp Neural Network for Predicting Elastic Modulus of 3-D Multi-Phase and Multi-Layer Braided Composite. Composite Structures, 122, 308-315. https://doi.org/10.1016/j.compstruct.2014.11.052
36. Andrew, N. and Ngiam, J., et al. (2014) http://ufldl.stanford.edu/wiki/index.php/Backpropagation_Algorithm
37. Zhang, Y.X., Gao, X.D. and Katayama, S. (2015) Weld Appearance Prediction with BP Neural Network Improved by Genetic Algorithm during Disk Laser Welding. Journal of Manufacturing Systems, 34, 53-59.
38. https://doi.org/10.1016/j.jmsy.2014.10.005
39. Sundararajan, N. and Saratchandran, P. (1998) Parallel Architecture for Artificial Neural Networks: Paradigms and Implementations. IEEE Computer Society Press, 412.
40. Han, L.Q. (2007) Artificial Neural Networks Tutorial. Beijing University of Posts and Telecommunications Press of China, Beijing, 47-83.
41. Karami, A. and Guerrero-Zapata, M. (2015) A Hybrid Multiobjective RBF-PSO Method for Mitigating DoS Attacks in Named Data Networking, Neurocomputing, 151, 1262-1282. https://doi.org/10.1016/j.neucom.2014.11.003
42. Reiner, P. and Wilamowski, B.M. (2015) Efficient Incremental Construction of RBF Networks Using Quasi-Gradient Method. Neurocomputing, 150, 349-356. https://doi.org/10.1016/j.neucom.2014.05.082

43. Liu, X.F., Bo, L. and Luo, H.L. (2015) Bearing Faults Diagnostics Based on Hybrid LS-SVM and EMD Method. Measurement, 59, 145-166. https://doi.org/10.1016/j.measurement.2014.09.037

44. Wang, Z.G., Zhao, Z.S., Weng, S.F. and Zhang, C.S. (2015) Solving One-Class Problem with Outlier Examples by SVM. Neurocomputing, 149,100-105. https://doi.org/10.1016/j.neucom.2014.03.072

45. Al-Hadeethi, H., Abdulla, S., Diykh, M., Deo, R.C. and Green, J.H. (2020) Adaptive Boost LS-SVM Classification Approach for Time-Series Signal Classification in Epileptic Seizure Diagnosis Applications. Expert Systems with Applications, 161, Article ID 113676. https://doi.org/10.1016/j.eswa.2020.113676

46. Yin, H.P., Jiao, X.G., Chai, Y. and Fang, B. (2015) Scene Classification Based on Single-Layer SAE and SVM. Expert Systems with Applications, 42, 3368-3380. https://doi.org/10.1016/j.eswa.2014.11.069

47. Liu, X.F. and Bo, L. (2015) Identification of Resonance States of Rotor-Bearing System Using RQA and Optimal Binary Tree SVM. Neurocomputing, 152, 36-44. https://doi.org/10.1016/j.neucom.2014.11.021

48. Dasgupta, S. and Ng, V. (2009) Mine the Easy, Classify the Hard: A Semi-Supervised Approach to Automatic Sentiment Classification. Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and 4th International Joint Conference on Natural Language Processing of the AFNLP, Suntec, Singapore, August 2009, 701-709. https://doi.org/10.3115/1690219.1690244

49. Zhu, X. (2007) Semi-Supervised Learning Literature Survey. Technical Report, University of Wisconsin Madison, Madison.

50. Schmidhuber, J. (2015) Deep Learning in Neural Networks: An Overview. Neural Networks, 61, 85-117. https://doi.org/10.1016/j.neunet.2014.09.003

51. Bengio, Y. (2007) Learning Deep Architectures for AI. Technical Report, IRO, Universite de Montreal, Montreal.

52. Hinton, G.E. and Salakhutdinov, R. (2006) Reducing the Dimensionality of Data with Neural Networks. Science, 31, 3504-3507. https://doi.org/10.1126/science.1127647

53. Hu, W.P., Qian, Y., Soong, F.K. and Wang, Y. (2015) Improved Mispronunciation Detection with Deep Neural Network Trained

Acoustic Models and Transfer Learning Based Logistic Regression Classifiers. Speech Communication, 67, 154-166. https://doi.org/10.1016/j.specom.2014.12.008

54. Fischera, A. and Igelb, C. (2014) Training Restricted Boltzmann Machines: An Introduction. Pattern Recognition, 47, 25-39. https://doi.org/10.1016/j.patcog.2013.05.025
55. Lopes, N. and Ribeiro, B. (2014) Towards Adaptive Learning with Improved Convergence of Deep Belief Networks on Graphics Processing Units. Pattern Recognition, 47, 114-127. https://doi.org/10.1016/j.patcog.2013.06.029
56. Zhou, S.S., Chen, Q.C. and Wang, X.L. (2014) Fuzzy Deep Belief Networks for Semi-Supervised Sentiment Classification. Neurocomputing, 131, 312-322. https://doi.org/10.1016/j.neucom.2013.10.011
57. Salakhutdinov, R. and Murray, I. (2008) On the Quantitative Analysis of Deep Belief Networks. Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland, August 2008, 872-879. https://doi.org/10.1145/1390156.1390266
58. Hinton, G.E. (2002) Training Products of Experts by Minimizing Contrastive Divergence. Neural Computation, 14, 1771-1800. https://doi.org/10.1162/089976602760128018
59. Zhang, H.-J., Zhang, N. and Xiao, N.-F. (2015) Fire Detection and Identification Method Based on Visual Attention Mechanism. Optik, 126, 5011-5018. https://doi.org/10.1016/j.ijleo.2015.09.167
60. Zhang, H.-J. and Xiao, N.-F. (2016) Parallel Implementation of Multilayered Neural Networks Based on Map-Reduce on Cloud Computing Clusters. Soft Computing, 20, 1471-1483. https://doi.org/10.1007/s00500-015-1599-3
61. Zhang, H.-J. and Xiao, N.-F. (2015) Learning Hierarchical Representations for Face Recognition Using Deep Belief Network Embedded with Softmax Regress and Multiple Neural Networks. Proceedings of the 2015 2nd International Workshop on Materials Engineering and Computer Sciences (IWMECS), 1-7 https://doi.org/10.2991/iwmecs-15.2015.1

CHAPTER 16

REVIEW OF RESEARCH ON TEXT SENTIMENT ANALYSIS BASED ON DEEP LEARNING

Wenling Li[1], Bo Jin[1], and Yu Quan[2]

[1]College of Science, Yanbian University, Yanji, China

[2]Department of Economics and Management of Yanbian University, Yanji, China

ABSTRACT

Sentiment analysis is part of the field of natural language processing (NLP), and its purpose is to dig out the process of emotional tendencies by analyzing some subjective texts. With the development of word vector, deep learning develops rapidly in natural language processing. Therefore, the text emotion analysis based on deep learning has also been widely studied. This article is mainly divided into two parts. The first part briefly introduces the traditional methods of sentiment analysis. The second part introduces several typical

Citation: Li, W., Jin, B. and Quan, Y. (2020), Review of Research on Text Sentiment Analysis Based on Deep Learning. Open Access Library Journal, 7, 1-8. doi: 10.4236/oalib.1106174.

methods of sentiment analysis based on deep learning. The advantages and disadvantages of sentiment analysis are summarized and analyzed, which lays a foundation for the in-depth research of scholars.

Keywords:- Deep Learning, Sentiment Analysis, Convolutional Neural Network, Recurrent Neural Network

INTRODUCTION

Text sentiment analysis is also known as opinion mining and tendency analysis. In short, it is the process of analyzing, processing, inducing, and inferring subjective text with emotion. It has a wide range of applications in public opinion monitoring, stock and movie box office forecasting, and consumer preference analysis [1]. Traditional affective analysis methods are mainly based on affective dictionary and machine learning, but there are some difficulties in using these two methods for affective analysis. Firstly, the text is unstructured. The length of the text is difficult to fit the classic machine learning classification model. Secondly, feature extraction is difficult. The text may be talking about a certain topic, or it may be talking about a person, a product, or an event. Not only does it take a lot of effort to extract features manually, but the results are not good. Thirdly, there is a link between words, and it is also difficult to incorporate this part of the information into the model. “How to reduce manual work to a greater extent, and can quickly mine valuable information and perform sentiment analysis”—as a result of thinking about the above issues, deep learning successfully entered everyone’s field of vision.

Deep learning is a general term for a series of machine learning algorithms based on feature self-learning and deep neural networks (DNN). Its advantages are its strong discriminative ability and feature self-learning ability. It is very suitable for high-dimensional, unlabeled, and big data features. This article divides text sentiment analysis based on deep learning into the following research tasks: 1) Briefly introduce and compare several classic methods of text sentiment analysis, and point out the advantages of deep learning; 2) Introduce several existing mature deep learning methods and make relevant notes; 3) Summarize the existing problems in text sentiment analysis based on deep learning, and put forward suggestions and prospects.

BRIEF REVIEW ON THE RESEARCH PROGRESS OF TEXT SENTIMENT ANALYSIS

Text sentiment analysis is also called sentiment mining. The core of sentiment analysis is to classify the data you have, The first is the subjective and objective classification of text to reduce the interference caused by objective text to the analysis, and the other is to classify subjective texts [2], and dividing emotions into several categories according to people's emotional expressions for analysis of certain situations; In addition, the analyzed text can be divided into chapter-level, paragraph-level, and sentence-level. Different text lengths will result in different methods used in processing text. The following mainly introduces some mainstream methods of subjective information sentiment classification, and points out the problems and deficiencies in the current stage of sentiment classification research.

Emotion Dictionary-Based Approach

The sentiment dictionary-based sentiment analysis method is an unsupervised analysis method, and usually requires the method of "affective dictionary + manual judgment" for analysis. Turney [3] divides emotions into two categories: excellent and poor, and then introduces the method of pointwise mutual information (PMI) to calculate the semantics between the selected word and the excellent or poor words, respectively. The similarity is used to find the semantic orientation (SO) of the candidate words. The formula is as follows:

$$SO(\text{phrase}) = PMI(\text{phrase}, \text{“excellent”}) - PMI(\text{phrase}, \text{“poor”}) \quad (1)$$

Alistair et al. [4] believe that it is necessary to consider the polarity transition factor of each sentiment word in the current context (CVS); in 2012, Jinan et al. [5] studied two different sentiment dictionaries and three different scoring methods are used for sentiment analysis. The scoring method includes the commonly used weighting techniques for retrieving data, word frequency-inverse text frequency (TF-IDF), and potential Dirichlet allocation (LDA) strategy. However, the above methods are all based on artificial dictionaries, with limited coverage and artificial errors. In recent years, with the explosion of network data and the continuous increase of network language, this single method has been unable to solve the problems of a large number of unknown words and complex ambiguous words. But for small amounts of text, its accuracy is very high, so we can consider using it in combination with other methods.

Machine Learning Methods

The core of sentiment analysis based on machine learning is effective feature extraction, and then using classifiers for emotion classification. In 2002, Pang [6] and others first used machine learning algorithms for sentiment classification tasks, and proposed Naive Bayes (NB), Maximum Entropy (ME), and Support Vector Machine (SVM) and other models for sentiment classification of text. Here only introduces an algorithm, taking the Weibo comment of an event as an example, The NB algorithm is that given several sentiment categories, it is assumed that the target data is independent between several sentiments, and then input text data to find the maximum probability of the target data appearing in each text category, which is the corresponding text categories to solve text classification problems; In recent years, machine learning-based sentiment classification models have been widely studied, which has led to rapid development of machine learning in sentiment analysis. The machine learning-based method runs faster, but still requires a lot of manual annotation and other operations. High-quality data integration is costly and time-consuming. Its classification performance is also limited by the design of complex features, and has poor adaptability in different fields.

INTRODUCTION TO TEXT SENTIMENT ANALYSIS BASED ON DEEP LEARNING

In 2006, the concept of deep learning was proposed [7], and in 2011, Socher [8] introduced a model based on recursive autoencoders to perform sentiment analysis on movie evaluation, and the effect is more obvious than traditional methods. In recent years, CNN, RNN, LSTM and other methods have been gradually applied to sentiment analysis, and their effects have been significant. This chapter will summarize the characteristics of deep learning methods and introduce the characteristics of several deep neural networks and their applicability in sentiment analysis of texts.

Features of Deep Learning Methods

Compared with the sentiment dictionary method and machine learning method, deep learning method is not perfect. It also has advantages and disadvantages for different types of text. In order to make it play a better role, the following summarizes and discusses its advantages and disadvantages.

Firstly, deep learning methods can automatically learn multi-level features, replacing the tedious manual feature extraction in machine learning, and because of the powerful learning and expression capabilities of deep neural networks, the results are often more accurate than traditional methods. However, due to its powerful expression ability, many useless parameters will be generated at runtime, which requires a large number of data samples for network training. It can be seen that this method is more suitable for sentiment analysis of large amounts of data, and traditional methods are more accurate for sentiment analysis of small volumes of data.

Secondly, the focus of traditional machine learning methods and dictionary construction methods is how to build a mathematical model and what features to extract. However, the focus of deep learning methods is to design a more efficient network structure and how to train more accurate network parameters.

Thirdly, due to the powerful autonomous learning function, deep neural networks can automatically adjust the weights of network parameters to achieve the desired effect as much as possible. The same model and training method may be applied to different problems, but for different problems, the network structure and parameter weights are different, the whole structure is like a function, the input and output are one-to-one corresponding. Because of this, deep learning can be applied to many different fields and has achieved good results. However, due to the diversity and complexity of the language text, it is easy to make the emotional evaluation deviate, especially for the Chinese language, which is also the key to further improve the deep learning.

Characteristics and Applicability of Several Deep Networks

In recent years, deep network models have been continuously innovated and developed. Different network structures have made their respective characteristics and functions different. It is mainly reflected in the type of text (for example, long text and short text), the granularity and scale of the problem, and the type of the problem. In the following, some of the more classic deep network models are briefly analyzed and summarized in terms of text sentiment analysis.

Based on CNN (Convolutional Neural Network Model)

Convolutional neural network is a kind of feed forward neural network [9]. In recent years, it has been widely used in natural language processing,

speech recognition, and image processing. Its structure is mainly composed of an input layer, a convolutional layer, a pooling layer, a fully connected layer, and an output layer. The structure is shown below in Figure 1:

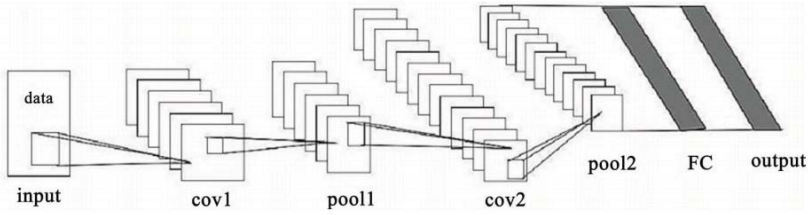


Figure 1. Structure of a classic convolutional neural network (Note: picture reference [10]).

As shown in Figure 1, taking short text as an example, the input layer is a vector representation of the input data, where the matrix is represented as:

$$e \in R^{n \times k} \tag{2}$$

Among them, n is the word length of the sentence and k is the dimension of the word vector.

Next, the convolution layer performs a convolution operation on the input matrix and vectorizes the input data to extract local features. The result can be expressed as:

$$c_i = f\left(W \cdot X_{i:i+h-1} + b\right) \tag{3}$$

Among them, c_i represents the i-th eigenvalue corresponding to the convolution operation; W represents the weight matrix; b represents the bias; f represents the activation function; $X_{i:i+h-1}$ represents the length of the i to $i+h-1$words in the sentence. After performing the convolution operation on the input matrix, the convolution kernel feature vector map is obtained as:

$$C = [c_1, c_2, \cdots, c_{n-h+1}] \tag{4}$$

among them, $c \in R^{n-h+1}$.

The pooling layer is an important layer in the network structure. It can extract important features from the feature vector map obtained from the previous layer. In more operations, the maximum pooling method is used for sampling. The obtained features are expressed as:

$$c = \max\left(c_1, c_2, \cdots, c_{n-h+1}\right) \tag{5}$$

The convolution operation is used to obtain the vectorization of the sentence through the vectorization of the words, and then learn the vector representation of the sentence as a feature, which makes it more suitable as a way to deal with the sentiment analysis problem of short text. Not only can multiple channels be used for multi view feature extraction, but also the number of parameters can be reduced by sharing weights, but the main disadvantage is that the complexity is high when processing long text, and with the increase of convolution layer, there will be problems such as gradient disappearance.

Based on RNN (Recurrent Neural Network Model)

Recurrent neural network mainly includes input layer, hidden layer and output layer. For some text data, there may be a relationship between the front and back, that is, there is a temporal relationship between the data. The "memory function" of the recurrent neural network is reflected here. Compared to ordinary fully connected neural networks, each neuron of the recurrent neural network will remember the output value of the previous moment, and affect the calculation of the output value of the current moment to a certain extent. The structure of the recurrent neural network is shown below in Figure 2.

Calculated as follows:

$$O_t = g\left(V \cdot S_t\right) \tag{6}$$

$$S_t = f\left(U \cdot x_t + W \cdot s_{t-1}\right) \tag{7}$$

Among them, x is the value of the input layer; s is the output of the hidden layer; U is the weight parameter when calculating from x to s; V is the weight parameter when calculating the hidden layer to the input layer; W represents the weight parameter of the influence of the value of the hidden layer before calculation on the value of the hidden layer at the current moment; O represents the value of the output layer.

But the recurrent neural network has its own shortcomings. During data training, if a longer sequence appears, the gradient will disappear or the gradient cannot be updated. Therefore, RNNs have a poor ability to capture long text information. Based on traditional RNNs, they are more suitable for sentence-level sentiment analysis problems (such as Weibo reviews).

Hochreiter [12] and others proposed long-short-term memory networks (LSTMs), and Cho [13] and others proposed gated recurrent units (GRU). These recurrent neural network variant structures effectively solve the problem of long-term dependence by introducing gate layers such as forget gates to process input data. Text sentiment analysis belongs to a type of natural language processing. Words are related to each other and depend on each other. Therefore, the "memory function" of the recurrent neural network shows its advantages. It can analyze the feature associations between the words before and after in the sentence to extract more accurate features. With the introduction of LSTM, GRU and other models, the problem of long text gradient disappearance has been solved, making recurrent neural networks widely used in the field of sentiment analysis.

Based on FNN (Fuzzy Neural Network Model)

FNN networks, the initial text representations are generally BOW and VSM models with great sparsity, which is more suitable for processing text-level sentiment analysis problems at the chapter level. Because the text set of the same size will cause the initial representation of the short text to be too sparse, the problem will not be obvious. Therefore, the short text can be processed by controlling the size of the text set. Model training generally combines unsupervised pre-training and supervised parameter adjustment; accordingly it can use a large amount of unlabeled data, which is also its advantage.

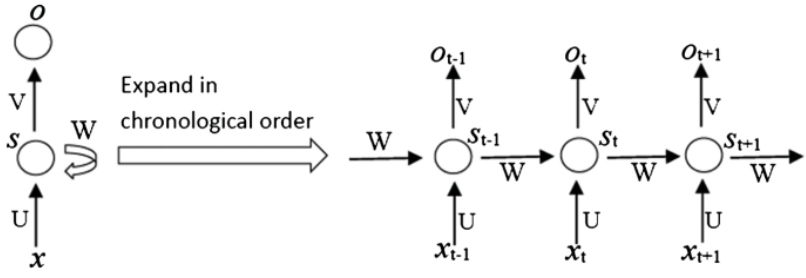


Figure 2. RNN structure based on time. (Note: picture reference [11]).

SUMMARY AND PROSPECT

This article briefly reviews and analyzes traditional methods of text sentiment analysis. It mainly introduces several different deep learning methods

and text data for different categories, and further summarizes and analyzes their unique advantages and applicability. Deep learning method saves a lot of complicated process of complicated feature extraction compared with machine learning method, but it has its own shortcomings. If there is supervised deep learning, it still needs to label a large number of data sets for model training. In the case of unsupervised deep learning, the requirements for semantic association are very strict. But the understanding of semantics is diverse and often causes ambiguity, which affects the degree of relevance. Therefore, the sentiment analysis of text based on deep learning still needs further research, and the author will continue to work hard in this direction.

REFERENCES

1. Zhu, X.X. (2019) Summarization of Text Sentiment Analysis Based on Topic Mining Technology.
2. Yang, L.G., Zhu, J. and Tang, S.P. (2013) A Review of Text Sentiment Analysis. Journal of Computer Applications, 33, 1574-1607. https://doi.org/10.3724/SP.J.1087.2013.01574
3. Turney (2002) Thumbs Up or Thumbs Down? Semantic Orientation Applied Unsupervised Classification of Reviews. Meeting on Association for Computational Linguistics. Association for Computational Linguistics, ACM Press, Philadelphia, PA, 417-424. https://doi.org/10.3115/1073083.1073153
4. Alistair, K. and Diana, I. (2006) Sentiment Classification of Movie Reviews Using Contextual Valence Shifters. Computational Intelligence, 22, 110-125. https://doi.org/10.1111/j.1467-8640.2006.00277.x
5. Jinan, F., Osama, M., Sabah, M., et al. (2012) Opinion Mining over Twitter Space Classifying Tweets Programmatically Using the R Approach. Proceedings of the 7th International Conference on Digital Information Management, Macau, China, 313-319.
6. Pang, B., Lee, L. and Vaithyanathan, S. (2002) Thumbs up? Sentiment Classification Using Machine Learning Techniques. In: Proceedings of Empirical Methods in Natural Language Processing, MIT Press, Cambridge, MA, 79-86. https://doi.org/10.3115/1118693.1118704
7. Hinton, G.E. and Salakhutdiony, R.R. (2006) Reducing the Dimensionality of Data with Neural Networks. Science, 313, 504-507. https://doi.org/10.1126/science.1127647
8. Socher, R., Cliff, C.L., Andrew, Y., et al. (2011) Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In: Getoor, L. and Scheffer, T., Eds., Proceedings of the 28th International Conference on Machine Learning Bellevue, Omni Press, Madison, WI, 129-136.
9. Si, X.H. and Wang, Y. (2019) Analysis of Short Text Sentiment Orientation with CNN and BLSTM. Journal of Software, 18, 15-20.
10. Lu, C. (2017) Research on Sentiment Analysis Methods Based on Deep Learning. Hunan University, Changsha.
11. Ji, L.Z. (2019) Research on Text Sentiment Analysis Technology Based on Deep Learning. Beijing University of Posts and Telecommunications, Beijing.

12. Hochreiter, S. and Schmidhuber, J. (1997) Long Short-Term Memory. Neural Computation, 9, 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735
13. Cho, K., Van Merrienboer, B., Gulcehre, C., et al. (2014) Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. arXiv Preprint arXiv:1406.1078. https://doi.org/10.3115/v1/D14-1179

CHAPTER 17

CLASSIFYING HAND WRITTEN DIGITS WITH DEEP LEARNING

Ruzhang Yang

Shanghai Foreign Language School, Shanghai, China

ABSTRACT

Recognizing digits from natural images is an important computer vision task that has many real-world applications in check reading, street number recognition, transcription of text in images, etc. Traditional machine learning approaches to this problem rely on hand crafted feature. However, such features are difficult to design and do not generalize to novel situations. Recently, deep learning has achieved extraordinary performance in many machine learning tasks by automatically learning good features. In this paper, we investigate using deep learning for hand written digit recognition. We show that with a simple network, we achieve 99.3% accuracy on the MNIST dataset. In addition, we use the deep network to detect images with multiple digits. We show that deep networks are not only able to classify digits, but they are also able to localize them.

Citation: Yang, R. (2018), Classifying Hand Written Digits with Deep Learning. Intelligent Information Management, 10, 69-78. doi: 10.4236/iim.2018.102005.

Keywords:- Digit Classification, Deep Network, Gradient Descent

INTRODUCTION

Text recognition from images is an important task that has multiple real-world applications such as text localization [1] [2], transcription of text into digital format [3] [4], car plate reading [5] [6] [7] [8], automatic check reading [9], classifying text from unlabeled/partially labeled documents [10], recognizing road signs and house number [11] [12], etc. Traditionally hand designed features are used to for image classification [13] [14] [15] [16] [17]. However, these techniques require a huge amount of engineering effort, and often do not generalize to novel situations.

Recent techniques in deep learning have allowed efficient automatic learning of features that are superior to hand designed features. As a result, we are able to train classifier that is significantly more accurate compared to previous methods. In this paper, we investigate using deep learning to classify handwritten digits, and show that with a simple deep network, we can classify digits with near-perfect accuracy.

We test our methods on the MNIST dataset [18]. This dataset consists of 50,000 training digit images and 10,000 testing images and is an important benchmark for deep learning methods. Samples images from the dataset are shown in Figure 1. On this dataset, we achieve an accuracy of 99.3% on the test set.

We also investigate classifying multiple digits, where more than one digit is present in an image. An example of this task is shown in Figure 2. We design a novel method of applying classifier of a single digit to an image with multiple digits. Though the number of digits and their location is unknown a-priori, our method is able to accurately localize and classify all the digits in the image.

DIGIT CLASSIFICATION WITH DEEP NETWORKS

Supervised Learning

A supervised learning task consists of two components, the input x and label y. For example, the input can be images of handwritten digits, or image of natural objects, and the label is the corresponding digit class or object class. The goal is to learn the correct mapping f from input x to label y. To accomplish this a learner is provided with examples of the correct mapping

$(x_i, y_i), i = 1, \cdots, N$ where x_i is an example input and y_i is the corresponding label provided by human annotators. Ideally after learning, f should map each input in the dataset $(x_i, y_i), i = 1, \cdots, N$ to the correct label, i.e. $f(x_i) = y_i$. The hope is that the learner can learn the correct mapping between x and y based on these examples, so that on unseen data, the learner f can also correctly classify.

Usually f is selected from a class of functions indexed by a parameter θ. For example, the class of functions can be quadratic functions

$$f(x) = ax^2 + bx + c$$

Figure 1. Samples from the MNIST dataset.

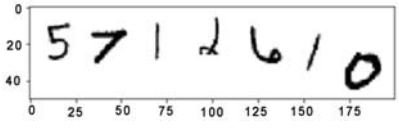


Figure 2. Classifying multiple digits.

in this example, $\theta = (a, b, c)$ are the parameters. We will denote the function selected by a parameter choice as f_θ.

To encourage the learner to select a f_θ that maps each x_i to the correct y_i we define a loss function such as

$$\mathcal{L}_\theta = \sum_{i=1}^{N} (f_\theta(x_i) - y_i)^2$$

In general any loss function that takes a smaller value when $f(x_i)$ is closer to y_i can be used. For classification tasks we use a special class of functions f_θ that outputs a probability distribution. That is for each x_i, $f_\theta^j(x_i)$ is the probability the input belongs to the j-th class. Then we can use the cross-entropy loss

$$L_\theta = \sum_{j=1}^{K} I(y_i = j) \log f_\theta^j(x_i)$$

where K is the number of classes, and $I(y_j = f) = 1$ if $y_i = j$ and equals to 0 otherwise.

To train the model, we use gradient descent on the loss function L_θ. This is described by the following process:

1) We start from a random parameter θ that can be arbitrarily chosen.
2) We compute the gradient of the loss function $\nabla_\theta L_\theta$. Computation of this gradient is discussed in the next section.
3) We update the parameters by $\theta := \theta - \alpha \nabla_\theta L_\theta$. This changes θ in the direction that minimizes the loss L_θ. α is the learning rate that controls the step size. The larger the step size, the faster θ changes. However, step size that is too large may lead to instability or even divergence. Therefore, the learning rate α is an important hyperparameter that is selected based on the specific problem.
4) We repeat from Step 2 until θ stops changing.

The above algorithm reduces L_θ during each iteration. The hope is that when L_θ is minimized, $f_\theta(x_i)$ will be close to y_i, that is, the function f_θ we selected can correctly predict the label yi given xi on the training set.

However, even if f_θ correctly predicts every example we provided, this does not mean that f_θ will classify correctly on new data. For example, f_θ may have only memorized the training dataset. Therefore we need additional examples $(x_j, y_j),\ j = 1, \cdots, M$ that the learner has not seen during training. The learner should only be able to classify these new examples correctly if it has learned the correct mapping between x and y. We can compute the testing accuracy by dividing the number of examples f_θ correctly classifies by the total number of examples. This is the final measurement of performance that we use to evaluate our learner.

Deep Networks

In the previous section we left an open question: which class of functions $\{f_\theta\}$ to select from during training. This section introduces an important function class of deep networks [19] [20] [21].

The key idea of deep learning is to compose very simple functions $g_{\theta_1}^1, g_{\theta_2}^2, \cdots g_{\theta_d}^d$ into a very complex function $f_\theta(x) = g_{\theta_d}^d\left(g_{\theta_{d-1}}^{d-1}\left(\cdots g_{\theta_2}^2\left(g_{\theta_1}^1(x)\right)\right)\right)$. Each function

$g^{i}_{\theta_i}$ is a simple function with parameters θ_i. Then the parameters of f is simply the combined parameters of all the layers $\theta=(\theta_1,\cdots,\theta_d)$. Common functions used in deep learning include

1) Matrix multiplication g(x) = Ax + b where the parameters are matrix A and vector b.
2) Rectified Linearity (ReLU) [22]

$$g(x)=\begin{cases}x & x>0\\ o & x<0\end{cases}$$

This function does not contain a parameter.

3) Softmax function: the softmax function "squashes" a n-dimensional vector of arbitrary real values to a n-dimensional vector of real values in the range [0, 1] that add up to 1. The function applied to an n-dimensional input vector z is given by

$$\text{Sigmoid}(z_j)=\frac{e^{z_j}}{\sum_{k=1}^{n}e^{Z_k}}$$

Note that sigmoid naturally produces a distribution because the output sum to 1

$$\sum_j \text{Sigmoid}(z_j)=1$$

4) Convolution [23] [24] [25] [26]: the convolution function takes as input an array z of size M × N × K and output an array o of size M × N × C. The first two dimensions can be interpreted as width and height, while the third is the number of "channels". This function takes the input z, and applies a 2D-convolution operation defined as

$$o_{x,y,c}=\sum_{i=0}^{S}\sum_{i=0}^{S}w^{c}_{i,j,k}*z_{x+i,y+j,k},\quad \forall 1\le x,y,c\le M,N,C$$

where S is called the size of the filter map, and each w^c is an array of size S × S × K. All the w^c combined $\{w^c,c=1,\cdots,C\}$ is the set of parameters of the convolution function.

5) Pooling: Pooling is a process which reduces a M × N × K array into a smaller array, e.g. of size $M/2\times N/2\times C$. Usually we keep the number of "channels" unchanged. For example, in the digit classification task, it can reduce the scale of a clearer picture into a more ambiguous one, making it easier to process in the later steps.

We shall denote the output of $g_{\theta_i}^i\left(\cdots\left(g_{\theta_1}^1\right)\right)$ as h_i. Then $f_\theta(x)=g_{\theta_d}^d\left(h_{d-1}\right), h_{d-1}=g_{\theta_{d-1}}^{d-1}\left(h_{d-2}\right)$ etc. Finally, we have $h_1=g_{\theta_1}^1(x)$. Intuitively the network must map raw input images into highly abstract and meaningful labels, which is a highly complex mapping. The network accomplishes through a sequence of simple mappings composed together. Each function $g_{\theta_i}^i$ can be viewed as one "layer" of a network. This function g_θ^i processes the output of the previous functions h_{i-1} into higher level representations h_i. The network therefore can be viewed as processing the input through a sequence of "layers" whose output become increasingly more high level and abstract, until we finally reach the output layer, which corresponds to the labels. This intuition is illustrated in Figure 3.

Computing Gradients

Now that we have defined our model class, to implement the algorithm in Section 2.1, we must be able to compute the gradient $\nabla_\theta L_\theta$. This is accomplished with the back-propagation algorithm [19] [20] [21].

The back-propagation algorithm sequentially computes

$\nabla_y L_\theta, \nabla_{hd-1} L_\theta, \cdots, \nabla_{h1} L_\theta$ Intuitively, this tells us how each hidden layer must change to minimize loss L_α. When all the $g_{\theta_i}^i$ are simple functions, we can compute $\nabla_{hi-1} L_\theta$ from $\nabla_{hi} L_\theta$ analytically, and this can be computed automatically by software such as Tensorflow [27]. Given gradient $\nabla_{hi} L_\theta$ over each layer h_i, we can correspondingly compute the gradient $\nabla_{\theta i} L_\theta$ over parameters θ_i analytically. This can also be automatically computed by Tensorflow.

Intuitively the computation flows "backward" through the next (hence the name back-propagation). We compute gradient in the following sequence $\nabla_y L_\theta, \nabla_{\theta d} L_\theta, \nabla_{hd-1} L_\theta, \nabla_{\theta d-1} L_\theta, \cdots, \nabla_{h1} L_\theta, \nabla_{\theta 1} L_\theta$

Detecting and Localizing Multiple

Digits In many real-world problems, such as car plate detection [5] [6] [7] [8] or house number recognition [11] there are multiple digits in the same image, and their location is unknown to us. Therefore, not only do we want to classify existing digits, we would also like to locate where the digits are, and how many there are. We show that based on the deep classifier we trained before we can design an algorithm to accomplish this. What we need is that

given an image patch we must identify both whether there is a digit in the image patch, and what digit it is, if there is one. If we can accomplish this, then we may simply apply this method to each patch of our input image, and we will be able to localize and classify all the digits in the image.

Previously we trained a classifier $f_\theta(x)$ that takes as input an image, and outputs a probability distribution over all possible digits. We observe that when the input is an image that do not contain any digit, the output is a distribution with high entropy, that is, the network is not highly confident that any digit has been observed. On the other hand, when presented with an image that contains a digit, the output is a distribution with low entropy, and the network generally outputs the correct digit with very high confidence.

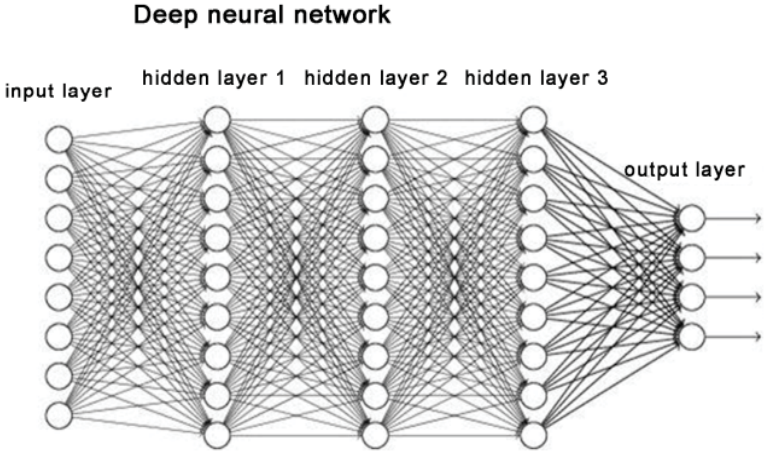


Figure 3. Illustration of a deep network.

We can then take advantage of this property. We measure the difference between the highest probability score and the second highest probability score. If the image contains a digit, the top prediction should have high probability score compared to the second highest. If the image does not contain a digit, all the possible predictions should be assigned similar probability and there should not be a significant difference. We show that this approach works very well in practice and we are able to accurately discover digits in an image in the experiments.

EXPERIMENT

Experiment Setting

We use 50,000 digit figures from the MNSIT training dataset to accomplish our training. Each example is a 28 by 28 single-color image. Our network architecture is as follows

1) A convolution layer with filter map of size 5 that takes as input the 28 × 28 × 1 image and outputs a feature map of shape 28 × 28 × 32
2) A pooling layer that reduces the size from 28 × 28 × 32 to 14 × 14 × 32
3) A ReLU layer
4) A convolution layer with filter map of size 5 and outputs a feature map of shape 14 × 14 × 64
5) A pooling layer that reduces the size from 14 × 14 × 64 to 7 × 7 × 64
6) A matrix multiplication layer that maps a vector of size 7 × 7 × 64 to 1024
7) A ReLU layer
8) A matrix multiplication layer that maps a vector of size 1024 to 10
9) A softmax layer

We train our network with gradient descent with a learning rate of $1e^{-4}$ for 20,000 iterations. We also use a new adaptive gradient descent algorithm known as Adam [28] which has been shown to perform better on a variety of tasks. Because shifting a digit does not change its class, during training we also randomly shift the digit by up to 6 pixels in each direction to augment the dataset. This makes the network more robust to shifting of the digit and improves testing accuracy

For multi-digit classification, we first extract all 28 by 28 image patches with a stride of 2. Then we run our classification network on all the patches, we take the most confident digit prediction in a region as our digit class prediction.

Results

Single Digit Classification

After training our network, we use another 10,000 test data to test the accuracy of our network. We achieved a testing accuracy of 0.993, which indicates that the network only makes a mistake in 7 out of every 1000 digits. We show the training curve in Figure 4. It can be observed that accuracy improves very quickly in the first 5000 iterations, then improves gradually until we reach approximately 99% accuracy on both the training set and testing set. No overfitting is observed.

Multiple Digit Classification

For the multi-digit classification, we show in Figure 5 the response of each digit detector at different locations of the sample input. The redder a region is, the more confident the classifier predicts that digit at that image patch. It can be seen that at correct digit locations, the detector shows consistently confident predictions throughout the region. This can be used to identify a region as containing a digit.

We also show in Figure 6 the confidence score that we computed. Redder color indicates presence of a digit. The location where the confidence score is high corresponds very well to where digits are present.

CONCLUSIONS

This paper applies deep networks to digit classification. Instead of hand designed features, we automatically learn them with a deep network and the back-propagation algorithm. We use a convolutional neural network with ReLU activations. In addition, we use pooling layers to remove unnecessary detail and learn higher level features.

We train our network with stochastic gradient descent. Training progresses quickly, we are able to achieve 90% accuracy with only 1000 iterations. After 100 k iterations, we achieve test performance of 99.3% on the MNIST dataset. We also study multi-digit classification and propose a method to detect digits in an image with multiple digits. We utilize the fact that our classifier produces a probability distribution.

We observe that when the input contains a digit, the classifier produces a distribution with low entropy and high confidence on the correct label. On the other hand, when the input does not contain a digit, the classifier

produces an almost uniform distribution. We use this different to detect whether an image patch contains an image. We experiment on multiple digit detection and our method is able to successfully localize digits and classify them.

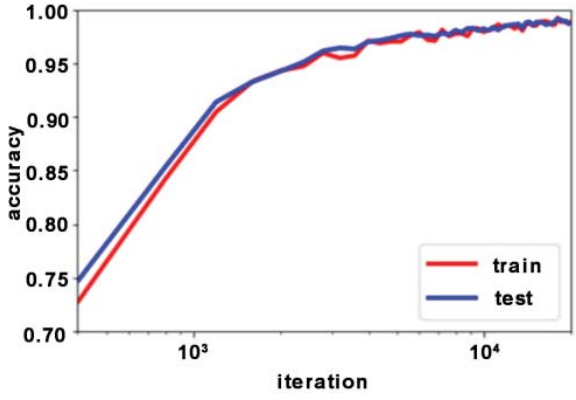


Figure 4. Training Curve. On the x-axis we plot the number of training iterations in log scale. On the y-axis we plot the classification accuracy on the test set. It can be observed that accuracy improves very quickly initially, reaching approximately 90% accuracy with only 1000 iterations. After that accuracy improves slowly. Eventually we reach an accuracy of 99.3% on the test set.

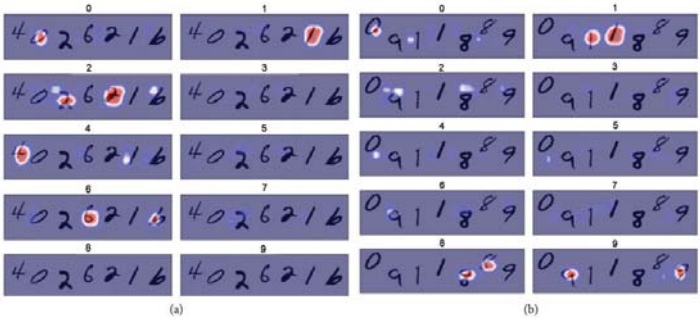


Figure 5. Detection of Multiple Images. Left and right are two examples where our model is able to localize the digits in an image with multiple digits at random positions. We apply our classifier to each patch of the image, and the output of each classification label. Redder color corresponds to higher confidence of the presence of that digit, and blue corresponds to low confidence.

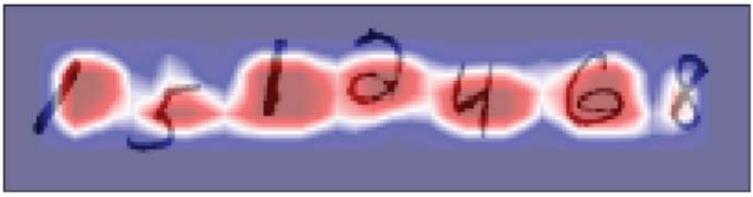


Figure 6. Confidence score indicate the present of a digit. The score is higher (redder) where there is a digit and lower (bluer) when there is not.

Future work should further improve accuracy and handle different size of digits in the multi-digit detection task.

REFERENCES

1. Neumann, L. and Matas, J. (2012) Real-Time Scene Text Localization and Recognition. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Providence, RI, 16-21 June 2012, 3538-3545.
2. Neumann, L. and Matas, J. (2010) A Method for Text Localization and Recognition in Real-World Images. Asian Conference on Computer Vision, Springer, Berlin, Heidelberg, 770-783.
3. Toselli, A.H., Romero, V., Pastor, M. and Vidal, E. (2010) Multimodal Interactive Transcription of Text Images. Pattern Recognition, 43, 1814-1825. https://doi.org/10.1016/j.patcog.2009.11.019
4. Bušta, M., Neumann, L. and Matas, J. (2017) Deep Textspotter: An End-to-End Trainable Scene Text Localization and Recognition Framework. IEEE International Conference on Computer Vision (ICCV), Venice, 22-29 October 2017, 2223-2231.
5. Raus, M. and Kreft, L. (1995) Reading Car License Plates by the Use of Artificial Neural Networks. Proceedings of the 38th Midwest Symposium on Circuits and Systems, 1, 538-541.
6. Barroso, J., Dagless, E.L., Rafael, A. and Bulas-Cruz, J. (1997) Number Plate Reading Using Computer Vision. Proceedings of the IEEE International Symposium on Industrial Electronics, Guimaraes, 7-11 July 1997, 761-766. Lee, S., Son, K., Kim, H. and Park, J. (2017) Car Plate Recognition Based on CNN Using Embedded System with GPU. 10th International Conference on Human System Interactions (HSI), Ulsan, 17-19 July 2017, 239-241.
7. Al-Hmouz, R. and Challa, S. (2010) License Plate Localization Based on a Probabilistic Model. Machine Vision and Applications, 21, 319-330. https://doi.org/10.1007/s00138-008-0164-9
8. Cun, Y.L., Bottou, L. and Bengio, Y. (1997) Reading Checks with Multilayer Graph Transformer Networks. IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP-97, 1, 151–154.
9. Kamal, N., McCallum, A.K., Thrun, S. and Mitchell, T. (2000) Text Classification from Labeled and Unlabeled Documents using EM. Machine Learning, 39, 103-134. https://doi.org/10.1023/A:1007692713085
10. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B. and Ng, A.Y. (2011) Reading Digits in Natural Images with Unsupervised Feature

Learning. NIPS Workshop on Deep Learning and Unsupervised Feature Learning, Granada, 12-17 December 2011, 5.

11. Maldonado-Bascón, S., Lafuente-Arroyo, S., Gil-Jimenez, P., Gómez-Moreno, H. and López-Ferreras, F. (2007) Road-Sign Detection and Recognition Based on Support Vector Machines. IEEE Transactions on Intelligent Transportation Systems, 8, 264-278. https://doi.org/10.1109/TITS.2007.895311
12. Forsyth, D.A. and Ponce, J. (2002) Computer Vision: A Modern Approach. Prentice Hall Professional Technical Reference.
13. Mundy, J.L. and Zisserman, A. (1992) Geometric Invariance in Computer Vision. Vol. 92, MIT Press, Cambridge.
14. Ke, Y. and Sukthankar, R. (2004) PCA-SIFT: A More Distinctive Representation for Local Image Descriptors. Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 27 June-2 July 2004, Vol. 2, 2.
15. Rublee, E., Rabaud, V., Konolige, K. and Bradski, G. (2011) ORB: An Efficient Alternative to SIFT or SURF. IEEE International Conference on Computer Vision, Barcelona, 6-13 November 2011, 2564-2571.
16. Liu, C., Yuen, J. and Torralba, A. (2016) Sift Flow: Dense Correspondence across Scenes and Its Applications. In: Dense Image Correspondences for Computer Vision, Springer, Cham, 15-49. https://doi.org/10.1007/978-3-319-23048-1_2
17. Yann, L., Cortes, C. and Burges, C.J.C. (2010) Mnist Handwritten Digit Database. http://yann.lecun.com/exdb/mnist
18. Rumelhart, D.E., Hinton, G.E., Williams, R.J., et al. (1988) Learning Representations by Back-Propagating Errors. Cognitive Modeling, 5, 1.
19. Yann, L., Bengio, Y. and Hinton, G. (2015) Deep Learning. Nature, 521, 436-444. https://doi.org/10.1038/nature14539
20. Schmidhuber, J. (2015) Deep Learning in Neural Networks: An Overview. Neural Networks, 61, 85-117. https://doi.org/10.1016/j.neunet.2014.09.003
21. Nair, V. and Hinton, G.E. (2010) Rectified Linear Units Improve Restricted Boltzmann Machines. Proceedings of the 27th International Conference on Machine Learning, Haifa, 21 June 2010, 807-814.
22. Yann, L., Boser, B.E., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W.E. and Jackel, L.D. (1990) Handwritten Digit Recognition

with a Back-Propagation Network. Advances in Neural Information Processing Systems, 2, 396-404.

23. Krizhevsky, A., Sutskever, I. and Hinton, G.E. (2012) ImageNet Classification with Deep Convolutional Neural Networks. Proceedings of the 25th International Conference on Neural Information Processing Systems, Lake Tahoe, 3-6 December 2012, 1097-1105.
24. Lawrence, S., Lee Giles, C., Tsoi, A.C. and Back, A.D. (1997) Face Recognition: A Convolutional Neural-Network Approach. IEEE Transactions on Neural Networks, 8, 98-113. https://doi.org/10.1109/72.554195
25. LeCun, Y., Jackel, L.D., Bottou, L., Cortes, C., Denker, J.S., Drucker, H., Guyon, I., et al. (1995) Learning Algorithms for Classification: A Comparison on Handwritten Digit Recognition. In: Oh, J.H., Kwon, C. and Cho, S., Eds., Neural Networks: The Statistical Mechanics Perspective, World Scientific, Singapore, 261-276.
26. Martín, A., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., et al. (2016) Tensorflow: Large-Scale Machine Learning on Heterogeneous Distributed Systems.
27. Kingma, D.P. and Ba, J. (2014) Adam: A Method for Stochastic Optimization.

CHAPTER 18

BITCOIN PRICE PREDICTION BASED ON DEEP LEARNING METHODS

Xiangxi Jiang
Barstow School of Ningbo, Ningbo, China

ABSTRACT

Bitcoin is a current popular cryptocurrency with a promising future. It's like a stock market with time series, the series of indexed data points. We looked at different deep learning networks and methods of improving the accuracy, including min-max normalization, Adam optimizer and windows min-max normalization. We gathered data on the Bitcoin price per minute, and we rearranged them to reflect Bitcoin price in hours, a total of 56,832 points. We took 24 hours of data as input and output the Bitcoin price of the next hour. We compared the different models and found that the lack of memory means that Multi-Layer Perceptron (MLP) is ill-suited for the case of predicting price based on current trend. Long Short-Term Memory (LSTM) provides relatively the best prediction when past memory and Gated Recurrent Network (GRU) is included in the model.

Citation: Jiang, X. (2020), Bitcoin Price Prediction Based on Deep Learning Methods. Journal of Mathematical Finance, 10, 132-139. doi: 10.4236/jmf.2020.101009.

Keywords:- Deep Learning Model, Multi-Layer Perceptron, Gated Recurrent Network, Long Short-Term Memory, Cross-Validation, Normalization

INTRODUCTION

Bitcoin is a cryptocurrency and a form of electronic cash. It is a digital currency that can be sent from user to user on the peer-to-peer Bitcoin network without intermediaries. It keeps a record of trading among peers and every record is encrypted. Each new record created contains the cryptographic hash of a previous block. Each record contains a timestamp and the data of the sender, the receiver, and the amount. Given Bitcoin is an emerged technology, few predictions is made on Bitcoin future value. Greaves and Au used linear regression, logistic regression and support vector machine to predict Bitcoin future price with low performance [1]. Indira et al. proposed a Multi-layer Perceptron based non-linear autoregressive with External Inputs (NARX) model to predict Bitcoin price of the next day [2]. Jakob Aungiers proposed a long-short term memory deep neural networks to predict S & P 500 stock price [3]. His research sheds light on Bitcoin prediction which is similar to stock price. Madan et al. used more machine learning approaches like generalized linear models and random forest to address Bitcoin prediction problem [4].

Researches mentioned above focuses on predicting the Bitcoin price of the next day. However, Bitcoin is traded frequently in a much smaller interval. In this research, we try to use historical data to predict next hour's price instead of next day's price which may have better application in real world. First we implemented data normalization like min-max normalization and normalization with window [5] where the data is normalized based on the window's initial value and the percentage of change. Multiple Layer Perceptron (MLP), Long-ShortTerm-Memory (LSTM) and Gated recurrent units (GRU) models are compared on the test dataset with cross-validation.

DATASET EXPLORATION

Data used in this research is collected from Kaggle [6]. Bitcoin data from Jan 2012 to July 2018 is collected. It has a timestamp, the value at Open, High, Low, Close, the volume traded in Bitcoin and USD, the weighted price and the date. This research focuses on predicting Bitcoin price in the future hour by using the price of past 24 hours, so only the timestamp and the weighted price are used in the model.

PRE-PROCESSING

As shown in Figure 1, the dataset is by minute, and contains around 3,409,920 points. Since we predicted the price by hours, we have had 1,409,920/60 which is 56,832 datapoints. The dataset is further split into training, validating and testing sets. As shown in Figure 2, training data takes up to 80% of the entire dataset, and validating and testing 10% respectively. As the time series data, samples are not randomized. We used the first 24 hours' Bitcoin price as input to predict the next hours' Bitcoin price. Several other pre-processing methods are implemented to improve data processing and model convergency efficiency. Minibatch is used to split large data into small batches, which improves memory efficiency. Minimum-Maximum normalization and window-based normalization is used to set the whole training dataset to (−1, 1) scale. Window normalization is based on the reference of stock market. The normalization methods will take each sized window and normalize each one to reflect percentage changes from the start hour of the window [3].

MODELS

Deep learning network is a type of computer modeling that finds the pattern within the given datasets and categorize the input accordingly. There are many different structures for deep learning network, including Multiple Layer Perceptron (MLP) that has a linear activation function, Recurrent Neural Network (RNN) that records a separate hidden unit to influence the next calculation. Extensions of RNN include Long Short-Term Memory (LSTM) and Gated Recurrent Model (GRU).

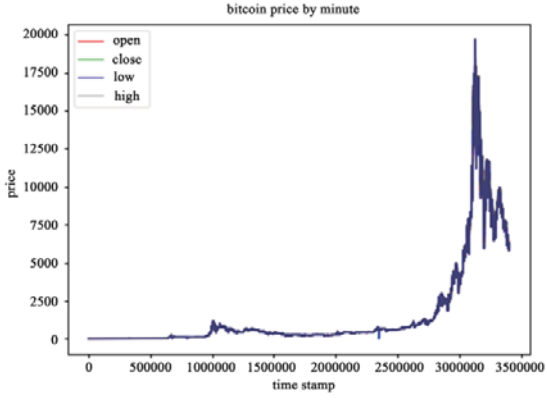


Figure 1. The overview of data listed by minutes.

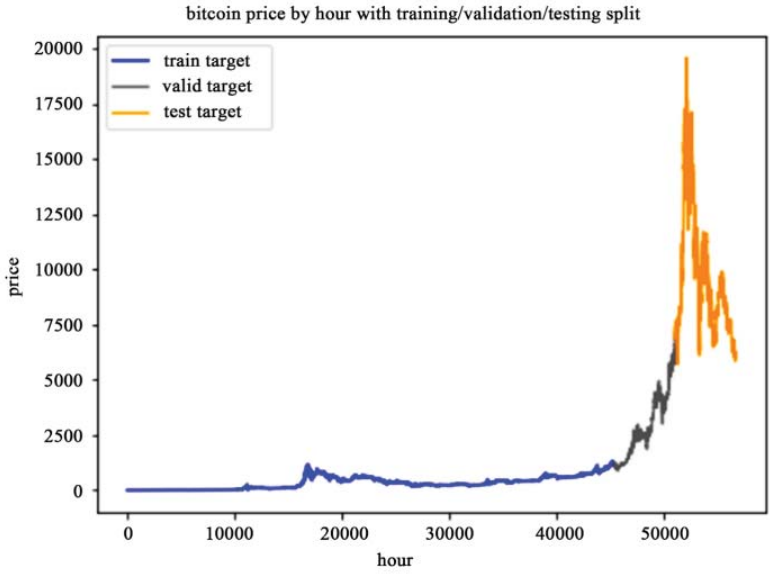


Figure 2. Training, validating and testing dataset.

MLP is a basic method in prediction. It reads all input with no ordering and then determine the relationship between the independent variables and the de- pendent variables. Hidden layers can be added between the input layer and the output layer together with the activation function, to better describe the non-linear relationship.

RNN is a group of method to calculate products from previous result of the model and new input data. In fact, it is better to MLP that it has "experience" from last calculations that will influence its calculations. The "experience" is gained from the model, is kept privately but is allowed to pass onto the next model. This private variable is called the hidden state and is passed on from the current calculation to the future calculation. It determines independently the output of the model, apart from the algorithm itself. However, RNN model depends on the continuous flow, which is sequential like the time series, in order to input data for the training. If the pattern repeats only over the long term, the previous repetition may be not influential enough to affect it at the next repetition. It also requires the data to be in order of time. Therefore determines, unlike MLP, RNN cannot be given random samples.

Long Short-Term Memory solves the issue that the diminished influence of distant events on the RNN network. It has a switch that can choose certain events to remember. It also is not long-term dependent and doesn't require as much training. It has four layers to determine the output, then passes the hidden state with the result to the next cycle. "Forgetting gates" exists in addition to four layers to determine if the experience should not be counted. Four layers and forgetting gates can be given different information to focus on either short-term or long-term memory.

GRU or Gated Recurrent Model is considered as one of the simpler model compared to the LSTM model, combination of the "forget" step with the "input" step into one, and as a result, requires only one hidden unit.

Among the three methods, MLP is mostly credited with its simplicity and the need for less computational power. They have the same amount of information as input. However, the number of hidden layers and the hidden units are more magic numbers. Some number turns out to work well especially, while some may turn out to be just the opposite. RNN accounts on the previous model through the hidden unit. The value uses in the calculation but does not need intervention. It can be very accurate, given the fact that the model has a large training set. However, long term patterns cannot be memorized and this may result in inaccuracy, especially when rapid changes take place in recent years. LSTM can choose whether it should "forget" previous states. Therefore, it is better capable of dealing with data that has repetitive trend over a long time. GRU model is also able to choose whether it should recall previous experience, but it is capable of learning more rapidly and need a bit less resource.

Six models are compared in this research. The model setups are listed in the following Table 1 and training results will be discussed in the next part.

RESULTS

As shown in Figure 3, in MLP and RNN frameworks, we find the similar conclusion that window-based normalization is much better than whole-dataset based normalization. Because of time-series data feature, the RNN frameworks converge faster than MLP methods. Model performance in this research is evaluated by Root Mean Square Error (RMSE) of the predicted price and the true price of the dataset. The results are listed in the following table. As shown in Table 2, normalization by window method performs much better.

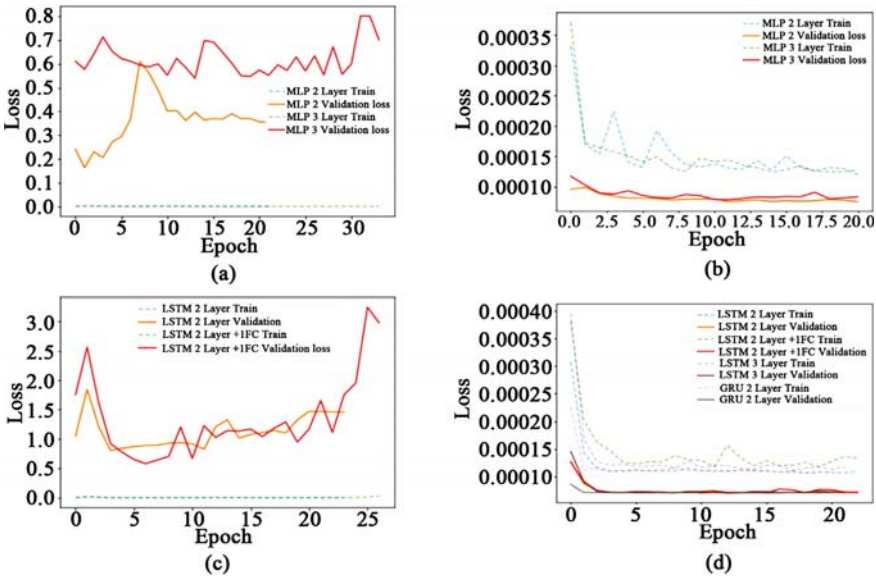


Figure 3. Training performance of models. (a) MLP with whole-dataset-normalization; (b) MLP with window-normalization; (c) RNN with whole-dataset-based normalization; (d) RNN with window-normalization.

Table 1. Font sizes of headings. Table captions should always be positioned above the tables

Model	Layer Information
2 Layer MLP	[256, 256]
3 Layer MLP	[256, 128, 64]
2 Layer LSTM	[256, 256]
2 Layer LSTM + 1Fully Connected layer	[256, 256] + [128]
3 Layer LSTM	[256, 256, 128]
2 Layer GRU	[256, 256]

We visualize the predicted price in the test dataset against true values in Figure 4 and zoom in to have a closer look at the predicted price in Figure 5. We can find that LSTM with normalization by window is the best combination.

A ten-fold cross-validation is conducted on all the models. As shown in Figure 6, we can see that the error goes down after the training set is

enlarged. At the end of the time-series data, when the fluctuation goes high, the error goes up a little again. Based on the cross-validation results, as summarized in Table 3, 2 layers of GRU is the best, and 2 layers of LSTM are very close to the performance.

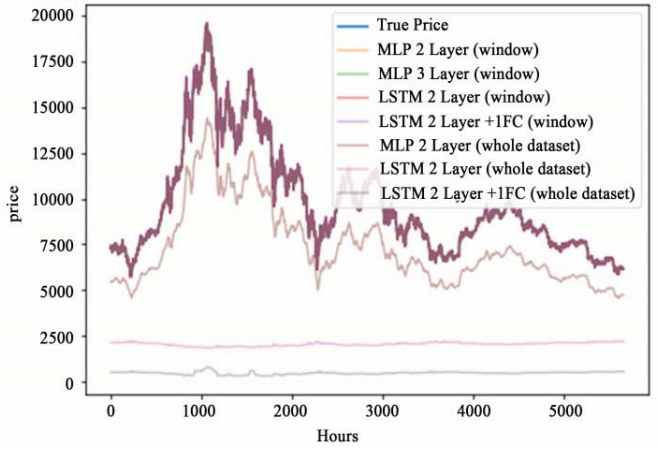


Figure 4. Predicted price on the test dataset.

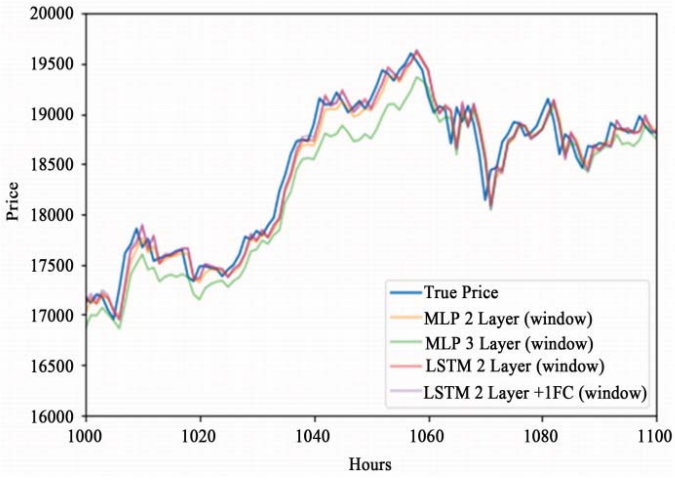


Figure 5. Zooming in.

Table 2. RMES of six models by different normalization methods

	Normalized By Window	Normalized Whole Dataset
2 Layer MLP	131.023	2541.128
3 Layer MLP	156.922	3692.356
2 Layer LSTM	125.387	8312.999
2 Layer LSTM + 1FC	126.016	9187.938
3 Layer LSTM	125.414	Not tried
2 Layer GRU	126.512	Not tried

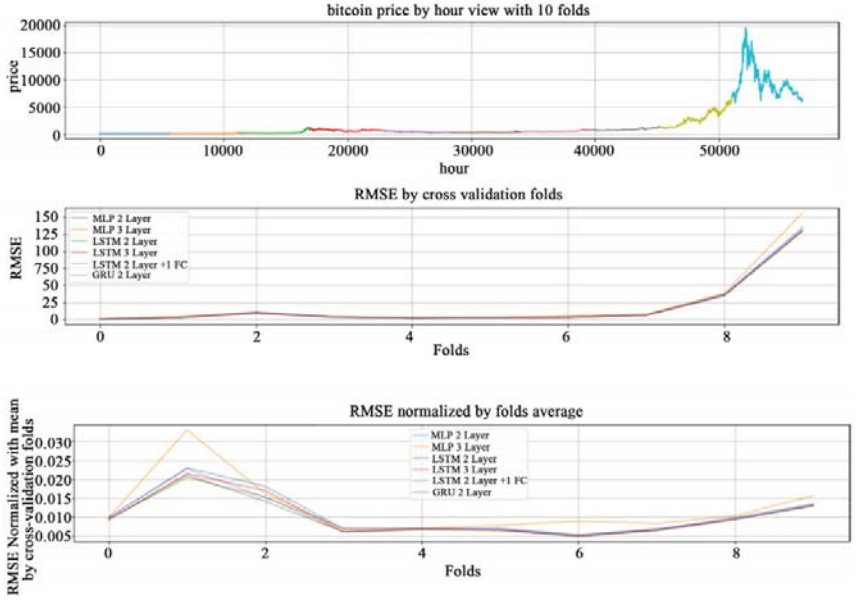


Figure 6. Cross validation results. The top one is the 10-fold split of original data, the middle one is the average RMSE for each fold, the bottom one is the RMSE/average price in that fold.

Table 3. Summarize of cross-validation results

	Mean RMSE	Std RMSE
2 Layer MLP	20.093	39.630
3 Layer MLP	22.736	45.711
2 Layer LSTM	19.121	38.214
2 Layer LSTM + 1FC	19.486	38.808
3 Layer LSTM	19.250	38.177
2 Layer GRU	19.020	38.146

CONCLUSION AND DISCUSSION

According to cross-validation results, 2 layers of LSTM has the best performance on the original test dataset and 2 layers of GRU is the best. All six models have close performance, so different models may be preferred in different scenarios. MLP model requires less computing power while it has slightly lower performance than RNN model. Our study combines several unique features, including the hour-based prediction, the usage of data from the past 24 hours, normalization by window and the comparison of different types of model with different amounts of layers. Based on this research, future work can be done on predicting a sequence of estimation so that it can be applied in more common Bitcoin trading scenarios.

REFERENCES

1. Alex, G. and Au. B. (2015) Using the Bitcoin Transaction Graph to Predict the Price of Bitcoin.
2. Indera, N.I., Yassin, I.M., Zabidi, A. and Rizman, Z.I. (2017) Non-Linear Autoregressive with Exogeneous Input (NARX) Bitcoin Price Prediction Model Using PSO-Optimized Parameters and Moving Average Technical Indicators. Journal of Fundamental and Applied Sciences, 9, 791-808. https://doi.org/10.4314/jfas.v9i3s.61
3. Aungiers, J. (2018) Time Series Prediction Using LSTM Deep Neural Networks. https://www.altumintelligence.com/articles/a/Time-Series-Prediction-Using-LSTMDeep-Neural-Networks
4. Isaac, M., Saluja, S. and Zhao. A. (2015) Automated Bitcoin Trading via Machine Learning Algorithms. http://cs229.stanford.edu/proj2014/Isaac%20Madan,%20Shaurya%20Saluja,%20Aoj ia%20Zhao,Automated%20Bitcoin%20Trading%20via%20Machine%20Learning%2 0Algorithms.pdf
5. Pedregosa, F., et al. (2011) Scikit-Learn: Machine Learning in Python. Journal of machine learning research, 12, 2825-2830.
6. Zielak. (2019) Bitcoin Historical Data, Bitcoin Data at 1-Min Intervals from Select Exchanges, Jan 2012 to July 2018, Version 14. https://www.kaggle.com/mczielinski/Bitcoin-historical-data

Index

E

F

G

H

I

J

K

L

M

N

T

U

V

Deep Learning Algorithms

The Deep learning is a branch of machine learning based on data presentation via complex representations with high degree of abstraction - that are obtained by applying learned nonlinear transformations. Deep learning methods find their application in important areas of artificial intelligence, such as: computer vision, natural language processing, speech and sound comprehension, as well as in bioinformatics. Deep learning is a class of machine learning algorithms that:

- uses multilayer nonlinear processor units to extract and transform features. Each subsequent layer takes as input the output elements of the previous layer.
- learns in a supervised and / or unsupervised manner.
- learns a number of levels of representation - corresponding to different degrees of abstraction.
- uses some form of descending gradient algorithm to train through error backpropagation.

The layers used in deep programming include the hidden layers of the artificial neural network and a multitude of statement formulas. This book covers the most important discriminant and generative deep models with special emphasis on practical implementations. We cover the key elements of classical neural networks and provides an overview of the building blocks, regularization techniques, and learning methods that are specific to deep models. Also we consider the deep convolutional models and illustrates their application in image classification and natural language processing. The generative deep models are often used in computer vision applications and natural language processing. Sequence modeling by deep feedback neural networks can be applied in the field of natural language processing. Practical implementations of deep learning are made in modern dynamic languages (Python, Lua or Julia), and also with application frameworks for deep learning (e.g. Theano, TensorFlow, Torch).

This edition covers different topics from deep learning algorithms, including: methods and approaches for deep learning, deep learning applications in biology, deep learning applications in medicine, and deep learning applications in pattern recognition systems.

Section 1 focuses on methods and approaches for deep learning, describing advancements in deep learning theory and applications - perspective in 2020 and beyond; deep ensemble reinforcement learning with multiple deep deterministic policy gradient algorithm; dynamic decision-making for stabilized deep learning software platforms; deep learning for hyperspectral data classification through exponential momentum deep convolution neural networks; and ensemble network architecture for deep reinforcement learning.

Section 2 focuses on deep learning applications in biology, describing fish detection using deep learning; deep learning identification of tomato leaf disease; deep learning for plant identification in natural environment; and applying deep learning models to mouse behavior recognition.

Section 3 focuses on deep learning applications in medicine, describing application of deep learning in neuroradiology: brain hemorrhage classification using transfer learning; a review of the application of deep learning in brachytherapy; exploring deep learning and transfer learning for colonic polyp classification; and deep learning algorithm for brain-computer interface.

Section 4 focuses on deep learning applications in pattern recognition systems, describing application of deep learning in airport visibility forecast; hierarchical representations feature deep learning for face recognition; review of research on text sentiment analysis based on deep learning; classifying hand written digits with deep learning; and bitcoin price prediction based on deep learning methods.

Dr. Zoran Gacovski has earned his PhD degree at Faculty of Electrical engineering, Skopje. His research interests include Intelligent systems and Software engineering, fuzzy systems, graphical models (Petri, Neural and Bayesian networks), and IT security. He has published over 50 journal and conference papers, and he has been reviewer of renowned Journals. Currently, he is a professor in Computer Engineering at European University, Skopje, Macedonia.

ISBN 978-1-77469-183-0

9 781774 691830 00000

ARCLER PRESS