NEIGHBOURING PROXIMITY—A KEY IMPACT FACTOR OF DEEP MACHINE LEARNING

by

Hongyuan Shi

B.Sc. Software Engineering
Industrial and Commercial College, Hebei University, 2015

THESIS SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
COMPUTER SCIENCE

UNIVERSITY OF NORTHERN BRITISH COLUMBIA

April, 2019

Abstract

Deep Learning has become increasingly popular since 2006. It has an outstanding capability to extract and represent the features of raw data and it has been applied to many domains, such as image processing, pattern recognition, computer vision, machine translation, natural language processing, and autopilot.

While the advantages of deep learning methods are widely accepted, the limitations are not well studied. This thesis studies cases where deep learning methods lose their advantages over traditional methods. Our experiments show that, when the neighbouring proximity disappears, deep learning methods are significantly less powerful than traditional methods. Our work not only clearly indicates that deep structure methods cannot fully replace traditional shallow methods but also shows the potential risks of applying deep learning to autopilot.

Contents

List of Figures

Acknowledgements

I want to specifically thank my supervisor, Dr. Liang Chen. Thank him for giving me great advices, support, supervision, understandment, patience and help in my study and life. He will always be my example in the future. This thesis can not be done without him.

I also want to thank University of Northern British Columbia for the learning and life support. Thanks to Prof. Jernej Polajnar, who introduced me to mutiagent system and Prof. Jueyi Sui who guided me in this thesis. I would also would like to thank Prof. Fan Jiang for his support when I'm doing TA.

To all my colleagues in UNBC, Negar, Meng Xi, Link Li, thank you for the help and support in my research and study.

To all the friends I met in Prince George, George, Johny, Finch, Andrea, thank you for the company, support and help.

To Siyuan Ren, my girlfriend, your coming made me filled with wonder.

I'm very grateful to my parents. Thank you for your guidance and love. I love you forever.

Chapter 1

Introduction

Machine learning is an important branch of artificial intelligence that has been studied for many years. Its algorithms allow computers to learn rules from large amounts of raw data and present representations of them. The development of machine learning has experienced two waves: shallow learning and deep learning.

In the 1990s, various shallow machine learning models were proposed, such as support vector machines, boosting, and maximum entropy methods. These models have achieved great success both in theoretical analysis and in practical applications because of solid theoretical foundation and little amounts of trainable parameters. Especially in small sample tasks, shallow learning models still have great advantages.

However, the traditional shallow models cannot mine the potential internal information from raw data. Therefore, deep learning was pushed onto the center stage of machine learning. It simulates humans brain structure to achieve efficient processing of complex input data, which allows the deep model to intelligently learn different

knowledge and effectively solve many types of complex problems. In recent years, with emergence of efficient learning algorithms, machine learning community has set off a wave of research on deep learning theory and application.

Both shallow and deep machine learning methods have made breakthroughs in tasks such as computer vision and natural language processing.

1.1 Deep Learning

Deep learning (DL) was first proposed by Hinton et al. [34] on 2006. After that, in 2012, Alexnet [44] made researchers feel the power of deep learning on ILSVRC2012 (All of ILSVRC challenges based on Imagenet [22]). Compared to K-nearest neighbours (KNN), support vector machine (SVM), boosting, maximum entropy and other "shallow" learning methods, deep architecture has more levels of non-linear operations [5]. Multilevel non-linear operators enable deep models to acquire more advanced and more abstract data representations by stacking simple non-linear modules.

It has been demonstrated that deep learning methods could discover abstractions of features at different levels and final representations from labeled raw datasets using general-purpose learning algorithms automatically.

In general, the deep learning model we refer to is a multilayer artificial network such as Deep Convolutional Neural Networks (DCNNs), Deep Belief Networks (DBNs) and Recurrent Neural networks (RNNs). In the DCNNs, lower layers detect and extract simple features with convolutional kernels and pass these features to higher layers, which extract more abstract and global features. The features of data are presented by

vector afterward. Unlike the working principle of DCNNs, in RNNs, state of neurons and current input together affect output of the neurons. Each neuron has a memory unit which captures information from beginning until current moment. They are very powerful models for processing sequences and are capable of processing sequences with arbitrary length theoretically. Deep belief networks are probabilistic generative models that are composed of multi-layers hidden units (also called feature detectors) [32] with connections between layers. But units within a single layer are not connected. A typical DBN is stacked by multiple Restricted Boltzmann machines with directed and undirected connections. In an RBM, each hidden layer encodes distribution of previous units and attempts to reconstruct its input. The final representation of input data is produced by layer-wise transformation from lowest to highest layer on DBNs.

1.2 Current Research Status of Deep learning

As a branch of machine learning, deep learning has become one of the most popular research subjects. In recent years, it has continuously attracted people's attention; especially the deep learning methods have achieved state-of-art results in various datasets as shown in Figure 1.1 [45]. In 2012, Alexnet achieved the state of art result by reducing top-5 error from 26% to 15.3% on ILSVRC2012, which is roughly 1.2 million pieces of data with 1000 categories. That opened curtain of deep learning. Compared to previous common neural networks, Alexnet is deeper and has more convolution kernels. "Deeper" has always been the direction of researchers' efforts—the deeper the structure, the better the presenting ability of neural networks.

Neural networks are usually trained through back-propagation using stochastic gradient decent. The gradient decreases exponentially as it is propagated down from

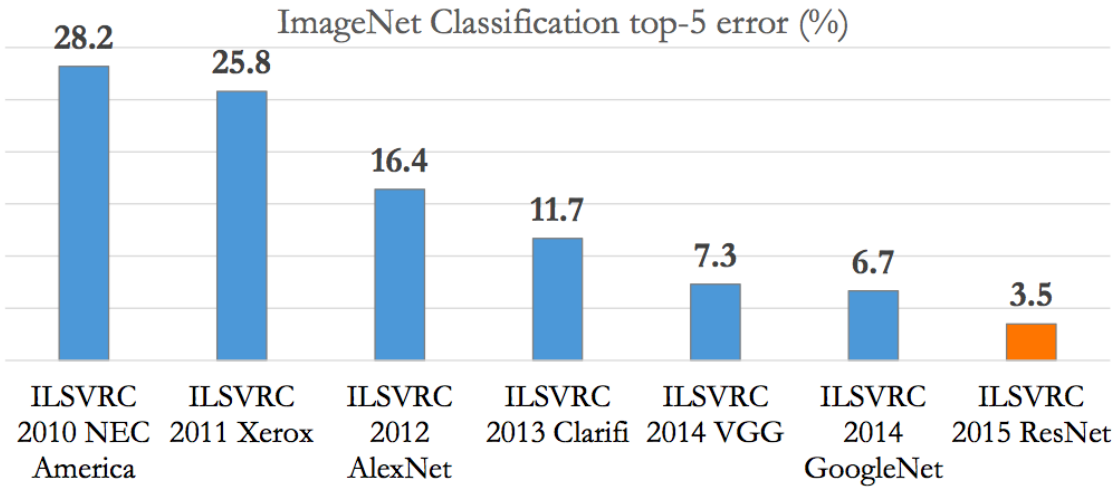


Figure 1.1: Imagenet classification top-5 error [45]. Imagenet is now one of most the popular dataset with more than 1.4 million images and has become one of standard dataset for deep learning algorithms. Since 2011, all the state-of-art results for Imagenet are achieved by deep learning methods

output layer to input layer; it may become vanishingly small, after multiple layers of propagation, that restrain the training from being effective and accurate. This phenomenon is called the vanishing gradient problem in the context of deep learning and it is a major problem that hinders neural networks from going deeper. The problem has been receiving attention again recently due to increasing amounts of applications of deep architectures. And Alexnet suppresses this problem by introducing Rectified Linear Unit and dropout. After that, Szegedy et al. [81] proposed a twenty two layers deep CNN called inception V1 (also called GoogLeNet). It achieved a top-5 error rate of 6.67% on ILSVRC2014. That was very close to human level performance. In Inception V1, two important concepts are employed: batch normalization [40] and small-convolution kernel. Batch normalization allows each layer of a network to normalizes the output of the previous activation layer by subtracting the batch mean and dividing by the batch standard deviation. That effectively suppress the vanishing gradient problem. Besides, small-convolution kernels are capable of reducing the number of parameters. Recently, at ILSVRC2015, He et al. [31] introduced Resid-

ual Neural Network (ResNet) architecture with "skip connection". Such a structure ensures gradients propagate more layers forward. That enables neural networks to be substantially deeper than previous and improves performance of deep architectures. ResNet has achieved a top-5 error rate of 3.57% which beats human expert on ILSVRC2015. Batch normalization, small-convolution kernel and skip connections have been widely adopted by deep learning researchers and have been used to set up the basic framework structures of current deep networks.

Deep RNNs are also rapidly evolving along with Deep Convolutional Neural Networks. Similar to the case of CNNs, vanishing gradient problem also seriously affects developments of RNN. To solve this problem, researchers have proposed many parameters optimization methods such as NAG [65], Adagrad [23], Root Mean Square Prop(RMSProp)[83], Adadelta [96] and so on. Adaptive learning rate algorithm named RMSProp, as a typical representative, can make RNNs' training process smoother. In 2015, Le et al. [46] introduced a novel initialization method with Rectified Linear Units (IRLU) which also obviously boosted performance of deep RNNs. Besides, researchers have improved performance by continuously upgrading and changing structures of RNNs to avoid the problem. For instance, models such as AWD-LSTM [57] and GruRNN [14] with more complex structures significantly improve the accuracy of natural language processing.

Researchers are now trying to combine CNN and RNN to solve more complex problems. Recently, researchers have used CNN-RNN hybrid model for image patching and image Super-Resolution reconstruction [67]. In addition, image-to-text hybrid model [51] is also a hot topic of current research. That is, deep model automatically generates a human-readable description based on a received image(Automatic Image Caption Generation [25]) or automatically generates an image based on some

descriptions.

1.3 Deep Learning Applications

Deep learning is now sweeping across industry and research, as evidenced by the successes of DCNNs and RNNs in fields of computer vision and natural language processing. Although it is a technique facing many challenges and problems, it has achieved a remarkable amount of reliable solutions for today's applications in personal, commercial or governmental environment. Some applications of deep learning are listed below.

A typical example is object recognition. As a subset of object recognition, face recognition/identification has been successfully applied to security department as shown in Figure 1.2 [1]. Australia government has applied facial recognition system as part of self-service border clearance kiosks program in Sydney airport, which improved security reliability and raised efficiency. Besides, Qantas Airways passengers are expected to use facial recognition for check-in, baggage drop, border processing, and airport lounge [24]. Other subsets, such as object tracking, and real-time action detection are also used in video surveillance [42].

Besides, deep learning was also applied to healthcare. It is well known that electrocardiogram is an important measure of cardiac activity and many cardiac abnormalities will be manifested in the electrocardiogram. [99] proposed a deep learning model that can serve as a tool for screening of electrocardiogram to quickly identify different types and frequencies of arrhythmic heartbeats.

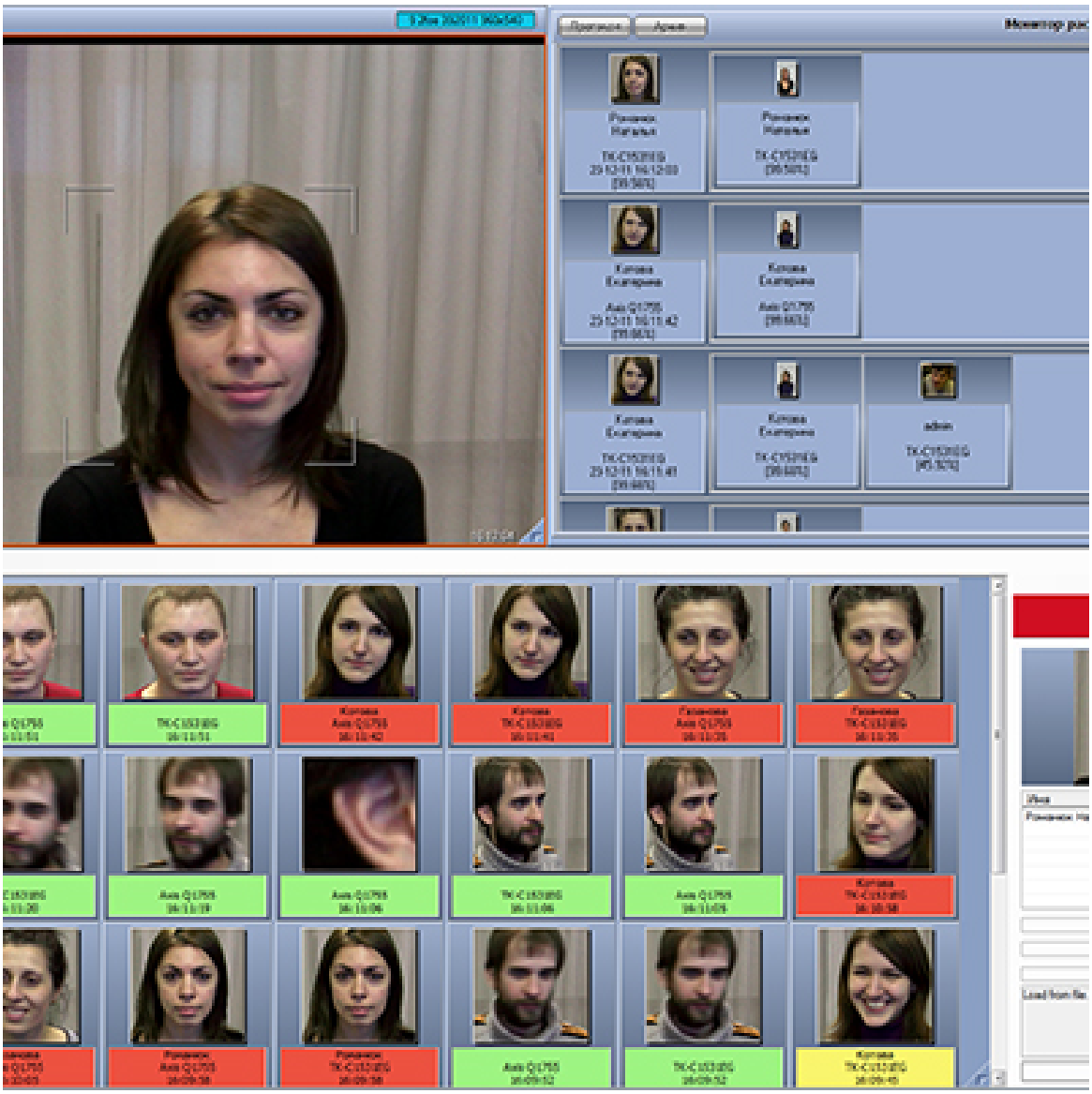


Figure 1.2: Face recognition based on deep learning has been widely used in CCTV systems [1] for verifying and recognizing faces.

For RNNs, one of the most popular usage areas is voice-assistant. The voice-assistant, which is capable of "understanding" your instructions, can even convert sounds into text and search for relevant content in search engines. Typical voice assistants include Apple Siri, Google now, and Microsoft Cortana [58]. Now, they can be found on nearly every smartphone. Deep RNN is also used in automatic text generation and automatic machine translation [60].

In addition, as a typical combination of deep learning with other complex technologies, self-driving vehicles have been developed by many technology companies and entered the road test phase.

1.4 Potential Problems of Deep Learning

Since the release of LeNet-5 [49], researchers have high expectations for deep networks and so they continue to explore advantages of them and try to introduce them into more fields. In the past two decades, researchers have proposed many ways to improve deep neural networks' performance. For example, some studies have made networks more complicated by enriching structures, that is, constantly modifying and introducing additional functional structures [55] [40] [53]. Other studies have initialized network parameters by certain methods to stabilize training process of neural networks. There are also some studies that avoid problems like vanishing gradient by modifying activation functions in networks. However, there remains a potential problem.

As in most of machine learning and pattern recognition applications, a dataset used for training and testing deep networks can be seen as a set of samples in the form of $(X; Y)$, where X is a vector, 2-d matrix or m-d matrix ($m \geq 2$) of elements, and Y is a class label or a class label vector. Image sets, natural language sequences, and text sequences are commonly used datasets for the experiments validating (evaluating) the advances of deep networks. An in-depth examination can easily reveal that there is a strong proximity relationship between neighbouring elements in X in any samples on these applications. In the applications related to images, any neighbouring pairs have statistical dependencies and proximity. Similarly, the words in a text or natural

language sequence keep dependencies between neighbouring words. In general, such datasets are defined as neighbouring proximity preserved datasets. Similarly, when elements in the samples do not keep neighbouring proximity, the datasets are called neighbouring proximity unpreserved datasets.

Based on the aforementioned background, it is natural to ask: Can deep networks work on neighbouring proximity unpreserved dataset as good as these work on neighbouring proximity preserved dataset? Furthermore, how well can deep networks perform on neighbouring proximity unpreserved dataset? Le et al. [46] and Cooijmans et al. [18] have demonstrated that IRNN and RBN-LSTM can work on neighbouring proximity unpreserved datasets. But, comparing with traditional shallow methods, how well can these models work? Especially in the case where the deep learning method has been applied to autonomous driving, will the neighbouring proximity seriously affect the performance of deep learning? And if so, will this potential problem affect people's safety? Besides, from the perspective of the development of machine learning, if these models are always guaranteed to be better than traditional shallow methods, then can we safely dump all the traditional methods? However, if it is not the case, under what situations should the traditional shallow methods be chosen over these models?

These problems are all needed to be studied in order to reach certain conclusions. Unfortunately, these potential problems and the limitations of deep learning are not well researched at this point in time.

1.5 Research Contributions

The main contributions of this thesis are summarized as follows:

This thesis presents the study and analysis of neighbouring proximity that seriously affects the performance of deep learning models. We conducted extensive experiments on neighbouring proximity preserved/unpreserved datasets using both deep learning and shallow methods. The results demonstrate that when the neighbouring proximity is lost, the accuracy of deep learning methods is at most as good as, if not worse than, that of advanced traditional shallow methods. In terms of accuracy, deep learning has no advantage over shallow learning on neighbouring proximity unpreserved datasets.

As the resources that traditional shallow methods needed are always much less than deep learning methods, we conclude that, in situations where neighbouring elements of input samples do not have proximity, deep learning methods are significantly less powerful than traditional methods. Furthermore, this clearly indicates that current deep structure methods cannot fully replace traditional shallow methods. In addition, this thesis shows that the length of the dependencies can lead to performance drop in RNN.

1.6 Organization of this thesis

The rest of this document is organized as follows:

Chapter 2 introduces developments of related learning methods in various periods, including reasoning-based algorithm, knowledge-based algorithm, and sample-based

shallow as well as deep machine learning methods.

In Chapter 3, several deep and shallow learning method frameworks related to this thesis are introduced, including convolutional neural network, recurrent neural network, deep belief network, sparse coding, and support vector machine. The first part of each section introduces the basic components of the framework, and the rest describes the learning algorithms and processes of the framework.

Chapter 4 includes an introduction of the detail designs of deep and shallow architectures we implemented or employed, including Standard CNN, MIN, DBN, Sparse coding and some SVM algorithms.

In Chapter 5, original and neighbouring proximity unpreserved dataset (i.e. MNIST and CIFAR-10), as well as parameters of the models for each dataset are introduced. Experimental results of each model are also presented in this chapter. The results clearly show that the shallow methods on neighbouring proximity unpreserved MNIST dataset and neighbouring proximity unpreserved CIFAR-10 dataset are superior to the deep learning methods in terms of accuracy and efficiency.

In the end, Chapter 6 summarizes this thesis and suggests some future directions.

Chapter 2

Background and History

Learning is a way for a computer to improve its performance after observing the world. Machines with learning capability are products using concepts of artificial intelligence which are developed to a certain advanced stage. In general, machine learning models are always used to learn information and knowledge autonomously from data. The first artificial intelligence program is the shopping program developed by Oettinger [90]. Shoppers simulated a mall with eight stores. When it receives an order to purchase an item, it will randomly enter a store and searches for the desired goods one by one until it finds the specified item or searches through eight stores. It will go directly to the store to look for the specified item when it receives a request to purchase a specified item again if it happens to remember the store where the object located. If not, it will repeat the original process. The learning algorithm of Shoppers is called rote learning, which is the easiest and earliest approach for learning. In general, learning is divided into four categories: rote learning, learning by being told (advice-taking), learning by deduction, learning from examples [21].

The development of machine learning has gone through three periods: inference period, knowledge period and sample learning period. Different research ideas and techniques play different roles, the most important of which are symbolism and connectionism. Symbolism and connectionism have driven machine learning forward in different periods. From the 1950s to the 1970s, many researchers believed that machines are intelligent as long as they have logical reasoning capabilities. This period is also known as reasoning period. As the research progresses, people no longer satisfied with machines that only had reasoning capabilities. In the mid-1970s, researchers began to combine logical reasoning with knowledge and established many expert systems. This period is called knowledge period. But expert systems are not perfect as well, because they are always difficult to teach people the knowledge they have learned. Researchers are eager for the machine to learn from samples itself.

From the 1980s to the present, the focus of machine learning research has shifted to sample-based learning, a period known as the "real" learning period. Due to the excellent classification prediction ability of statistical learning under limited samples, this stage of learning machine was quickly occupied by statistical learning in the mid-1990s. Another mainstream sample-based learning before the mid-1990s was the connectionism represented by neural networks. Because of the large gaps in the artificial neural network research field, neural networks were more like a Blackbox. However, with the deepening of research and the explosion of the hardware revolution, multi-layered neural networks have demonstrated excellent feature extraction and representation capabilities, which laid the foundation for deep learning in the field of machine learning.

Different learning methods during different periods will be introduced in the following sections.

2.1 Reasoning Period

Symbolism is the most popular faction in reasoning and knowledge period. Symbolists thought that basic units of human cognition and thinking are symbols. Besides, they also thought that every person is a physical symbol system and computers is a physical symbol system as well. In addition, a cognitive process was also considered as a symbolic representation of operations. They tried to use computers to simulate human intelligence behaviour, that is, to use computer symbol operations to simulate human cognitive processes.

Based on logical reasoning and symbolic operations, researchers got some achievements. The first and vital "intelligent" system with capable of logical reasoning is the mathematical theorem proving program called "Logic Theorist" (LT). It is also the first program designed to mimic human skill in problem-solving. LT not only successfully proved 38 of the first 52 theorems in the Principia Mathematica, but also found new and more elegant proofs for some [93].

In this program, symbols carry information about the outside world, and thinking is a process of symbol transformation. The goal of theorem proof is achieved by continuous reasoning. In detailed, LT was a search tree model, in which, the root was an initial hypothesis, and each branch was a deduction based on rules of logic [7]. Somewhere in the tree is the goal: a proposition that the program intends to prove. The pathway is along branches that result in a proof. Then, to solve the problem of excessive pathway space, [93] introduced Heuristic strategy that is capable of determining which pathways were unlikely to result in a solution. The heuristic is a mental rule-of-thumb strategy that does not guarantee a solution. In other words, it allows programs to solve complex problems more efficiently and reduce the total number of

possible solutions by discarding impossible and unrelated solutions to a more manageable set. LT opened prelude reasoning period. Based on previous works, A. Newell et al. [92] developed General Problem Solver (GPS). It intends to work as a universal problem solver machine that could be used to solve a variety of different types of problems. GPS is the first computer program to separate knowledge of problem from strategies of how to solve it, which defines problem space based on different goals and transformation rules to be implemented. And it divides overall goal into sub-goals and attempts to solve each of those using means-end-analysis approach. The principles and processes of LT and GPS fully demonstrate the core idea of symbolism: logical reasoning.

However, LT and GPS are still far from the smart machines that people expect. Because neither LT nor GPS is directly in contact with the real world, their input information is second-hand information that has been sorted by researchers. Besides, as the complexity of problems increases, search space becomes extremely large, which leads search quickly lost in a combinatorial explosion. In addition, Logic Theorist and General Problem Solver work only for very constrained toy domains. In other words, they can not solve any real-world problems.

2.2 Knowledge Period

The failure of LT and GPS demonstrates that machines with only logical reasoning capabilities can't solve complex problems of the real world. After intensive research, Feigenbaum et al. [98] thought that we could not expect learning systems to learn advanced concepts without knowledge. Symbolists believed that the core problems of machine learning research are knowledge representation and knowledge reasoning

rather than focus on heuristic computational methods. This period is also known as knowledge period. At this stage, knowledge was explicitly presented in terms of symbols, and machine performance was enhanced by continually expanding knowledge. Machine simulated human cognition and thinking behavior with reasoning to solve complex problems. Based on the above objective, symbolism researcher represented natural world information by symbolic knowledge in a form, combined with deductive reasoning to build a knowledgeable decision-making system. This resulted in the expert system.

The first expert system Dendral [54] was introduced around 1965. It was capable of identifying domains where expertise was highly valued and complex. Dendral is composed of a knowledge database and an inference engine. Besides problem-solving strategy, "heuristic" was also used in Dendral to replicate the process which human experts infer solutions to problems using rules of thumb [95].

In a typical expert system, knowledge database includes facts and rules that are representing human knowledge and experiences. It is not only used to search for possible results that match input data, but also to learn new "general rule" that help prune searches. The inference engine applies the rules to the known facts to determine (deduce) how inferences are being made. General processing pipeline of the expert system has been shown in Figure 2.1 [71].

Dendral laid the foundation for development of expert systems. The structure of knowledge database and inference engine has successfully spawned off many expert systems like XCON [3] (computer hardware configuration system), MYCIN [76] (medical diagnosis system), and ACE [87] (AT&T's cable maintenance system), PROLOG [16] (natural language process program).

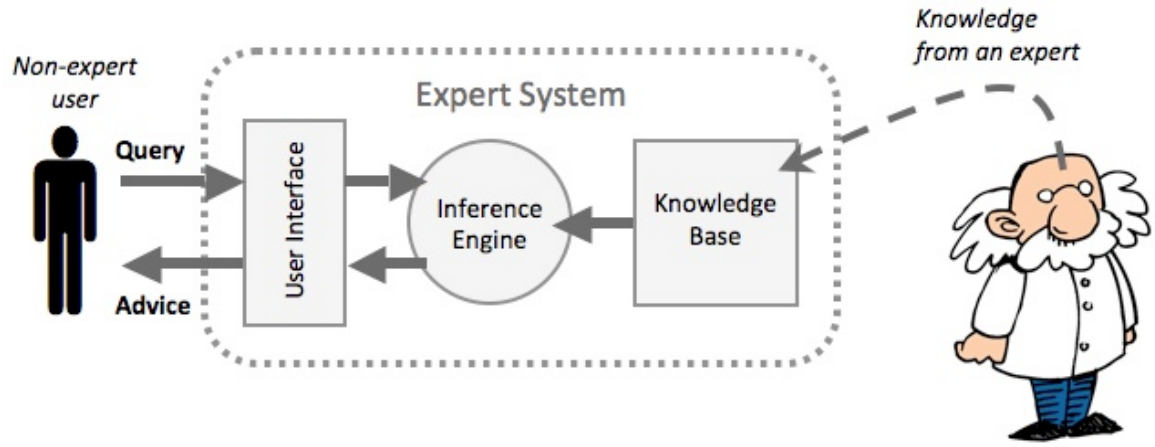


Figure 2.1: In a general expert model [71], users first type satisfactory information into the system. And then, the inference engine deduces the advice based on the information and the knowledge acquired from human experts.

But the expert systems are not perfect intelligence system. The major issue for expert systems is knowledge acquisition. During developments of expert systems, human programmer firstly collects knowledge and then encodes the learned experience. Knowledge acquisition is often seen as a way of discovering static facts of world and relationships of the various events that human uses to solve real-life problems. However, obtaining the knowledge of domain experts for any system is always tricky [38].

For example, we can imagine how humans learn to walk through practice and experience in childhood. Such tests and errors cannot be obtained in form of facts and rules. If people are asked to elaborate a set of rules based on their expertise, they often do not accurately reflect their skills. More importantly, knowledge systems don't learn from their experiences. So sometimes it is difficult for human programmers to summarize and hand over knowledge to the machine.

Another limitation of expert systems is learning capability. As we all know, expert systems require humans to summarize accurate information and experience in a given field as its knowledge. And results are obtained by constantly matching the knowledge

through chains of reasoning. Although expert systems are capable of applying rules to known facts to deduct new facts, they still cannot learn knowledge from real-world data and summarize rules themselves. When people cannot summarize rules from a certain field, expert systems cannot be applied in the field.

2.3 Inductive Learning Period

As intelligent creatures, induction is the most basic ability of human. Contrary to deduction, induction is a process of observing specialization examples to summarize generalization rules. Inductive learning is a process of discovering knowledge, correcting knowledge, and creating new knowledge from given positive and negative samples. Inductive learning also can be described as a search problem for rules space: finding rules that are consistent with samples description in rules space. Modern symbol learning is to learn the rules representation of examples by summarizing examples.

Due to the limitations of expert systems, knowledge engineering had entered a bottleneck period. In the 1980s, researchers attempted to create a machine that could directly extract knowledge and summarize rules from samples. Since symbolism made great achievements in artificial intelligence from the 1950s to 1980s, it was still mainstream in the early stages of learning phase. Symbolists proposed two kinds of models: decision tree learning and inductive logic programming.

The essence of decision tree learning is to generalize a set of classification rules from a training set. Its goal is to build a decision tree that fits well with the training set and has satisfied generalization capabilities. Moreover, it could be visualized through graphics, which makes it easy to interpret and understand. Therefore even

if people do not have the same knowledge as an expert, they can understand the solutions of decision trees after a brief explanation. The decision tree models have better adaptability that allows them to process numeric type data and as well as classification type data. Not only that, but decision tree models are also reliable because they are easily verified by statistical tests. In addition, decision trees are capable of processing datasets with missing or erroneous data.

In 1979, Quinlan proposed the decision tree-based algorithm ID3. After that, Classification and Regression Trees (CART) was proposed by [8] based on previous work. CART not only expands the function of decision trees, but also greatly promotes the decision trees algorithm. Now CART has been widely used in the industrial field. For example, [88] introduces a decision tree to implanted devices to identify features. Moreover, decision trees are also used for target recognition, control systems, financial analysis, physics remote, sensing pharmacology and so on [77].

Unfortunately, there are also some limitations on the decision tree algorithm. In general, the performance of decision trees cannot be as accurate as other models. And the robustness of decision trees is poor: a small change in training data may cause a huge change in entire model, and it will also cause the biggest change in final prediction result. In addition, trees produced by decision tree learning models are still prone to overfitting despite with previous pruning and post-pruning methods [91]. Decision trees are quite difficult to represent complex concepts. In addition, the optimal tree in decision tree learning process is an NP problem.

Inductive logic programming (ILP) is another representative of symbolism based on clause logic framework. It combines first-order representations with inductive methods and completes inductions of examples by modifying and expanding logical expressions iteratively. First ILP implementation was built by Shapiro and Ehud in

1981 [75]. After that, Stephen et al. [64] proposed the GEOLM, in which, bottom-up ILP inductive learning framework was introduced. In 1991, the term "Inductive Logic Programming" was proposed by [62].

The typical ILP learning process can be divided into three steps. First, directly select some specific facts corresponding to one or more positive examples as initial rules and adopt a bottom-up generation strategy to gradually generalized rules. After that, delete examples that already covered by learned rules and repeat step 1 so that the learned rules can cover as many positive examples as possible while excluding as many negative examples as possible. At last, adopt inverse resolution algorithm to learn more to obtain new rules that do not exist in background knowledge.

ILP has been successfully applied in many fields, especially in bioinformatics [84]. In the process of learning protein folding structure, it is necessary to find a language that can clearly describe structural information. Traditional attribute-value methods cannot describe relationships between objects, therefore unable to reasonably represent three-dimensional structure of protein molecules. First-order logic tool of ILP: clause logic, is a suitable language for describing this relationship. It plays a significant role in predicting protein structure information and generation of protein substructures. Moreover, ILP with knowledge engineering significantly improves the performance of inference engine of expert systems. In addition, ILP is also applied to many fields like natural language processing, software analysis and data mining [63].

ILP models learn general theories from examples and combine inductive methods with first-order language to obtain powerful example expression capabilities. Unlike Blackbox models such as neural networks, ILP systems are white-box models, they take background knowledge and structural data into account to learn some concepts that people are able to understand. However, ILP models have high requirements for

time and space, which makes them difficult to deal with larger datasets [63].

2.4 Machine Learning Period

2.4.1 Shallow Learning

Shallow learning, has become increasingly popular since the 1990s. Their algorithms are capable of extracting features, abstracting presentation, discovering knowledge from datasets and constructing a probabilistic model to predict and to analyze future inputs.

It can be divided into two categories: supervised learning and unsupervised learning. In training stage, supervised learning algorithms accept labeled data pair as training set. Each supervised learning algorithm attempts to learn mapping functions from the training set and uses the learned functions to predict future inputs. Unsupervised learning such as Clustering analysis or Sparse Coding is to find some functions to describe native data (i.e. unlabeled data).

Statistical learning is one of the most famous representatives of supervised learning. The earliest and simplest statistical learning model is linear regression, which can help people predict price of houses, rise and fall of stocks, trends in human epidemics and so on. Until the 1970s, Vapnik and A. Chervonenkis [85] proposed VC dimension that is the size of the largest finite subset of instance space X shattered by hypothesis space H. Subsequently, [30] proposed Structural Risk Minimization to reduce VC dimension of learning systems while ensuring classification accuracy (experience risk), and control expected risk of learning systems over entire sample set. In 1995, Cortes

and Vapnik proposed [19] support vector machine based on VC dimension theory and structural risk minimization. SVM models have many unique advantages in solving small sample, nonlinear and high-dimensional pattern recognition. So they are applied to many machine learning problems such as function fitting, text and hypertext categorization, and pattern recognition [61].

In general, statistical learning theory assumes that data is Independent Identically Distributed (I.I.D.), that is, the same class of data has certain statistical rules. Supervised statistical learning systems learn a mapping function from some given input-out pairs (i.e. training set). In other words, learning systems attempt to find an optimal decision function from hypothesis space to describe relationships between input variables and output variables. In prediction phase, the decision function outputs the most likely result based on inputs.

A typical supervised statistical learning model could be formally described as follow: X, Y, and F are represented as input space, output space, and hypothesis space, respectively. Therefore, the learning problem is to find a decision function f from F such that loss function $L(Y, f(x))$ is as small as possible. L is the difference between predicted value $f(x)$ and actual value y. Then the goal of learning is to choose a model with the minimum expected risk:

$$R_{exp}(f) = E_p[L(Y, f(x))] = \int_{x \times y} L(y, f(x)) P(x, y) \mathrm{d}x \mathrm{d}y \tag{2.1}$$

Since the joint distribution $P(X, Y)$ is unknown, R cannot be directly calculated. According to the law of large numbers, when N tends to infinity, empirical risk tends to expected risk, so empirical risk can be used to estimate expected risk. Finally, the goal of supervised statistical learning became the problem of minimizing the empirical

risk:

$$\min_{f \in F} \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)) \tag{2.2}$$

The goal of unsupervised learning is to find some function to model hidden structure information from unlabeled data. Unlike supervised learning, there is no explicit way to evaluate the learned structures [20]. The most studied and widely applied unsupervised learning method is Sparse Coding.

Sparse coding is inspired by a study of the human nervous system. [4] believes that an important function of biological vision systems in the initial stage of recognition is to remove statistical redundancy from inputs stimulus as much as possible. This study indicates that the response of optic nerve cells of the primary visual cortex to external environment satisfies characteristics of sparse coding. The so-called sparse coding refers to making the number of activated nerve cells as small as possible while representing input as completely as possible. Only a small number of components are simultaneously in an apparent activation state after data is sparsely encoded. This approach is capable of capturing high-level semantic features from original data. Sequently, Olshausen and Field [66] proposed a well-known sparse coding model, which expresses images by linear superposition of basis function. That makes the reconstructed image as close as possible to the original image in the sense of minimum mean square error, and the "response" is as sparse as possible. After training, the model has the properties of simple cells in the visual cortex, which is in good agreement with the results of neurophysiological experiments.

In short, Sparse Coding uses information theory to establish a quantitative connection between statistical properties of natural environment and biological (or machine) visual system functions (responses). Moreover, it provides a new method for designing

machine vision system by mimicking coding properties and biological vision system.

Supervised and unsupervised learning and other learning methods(for example, Clustering and PCA) together constitute shallow learning methods. The shallow models were the most popular learning methods before deep learning was introduced. Especially in learning tasks with small samples, statistical learning algorithms showed excellent classification and regression performance. Shallow learning methods have been widely applied in computer vision, natural language processing, pattern recognition, and even some commercial activities [41].

2.4.2 Deep Learning

As a branch of machine learning, deep learning differs in structure from traditional shallow machine learning. A deep learning model is composed of multiple non-linear layers. It implements complex function approximation and distributed representation of input data by training a deep nonlinear network.

In general, the deep structures we refer to are multilayer artificial neural networks, such as Deep Convolutional Neural Networks (DCNNs), Deep Belief Networks (DBNs) and Recurrent Neural Networks (RNNs). Deep learning methods succeed in many fields such as object recognition and detection, pattern recognition, natural language processing.

Deep learning methods allow systems to discover representations of potential factors from native data automatically. Deep models are capable of learning representation of data by stacking some multi-layer nonlinear modules. Each module converts a lower representation (starting from the original input) to a higher and slightly ab-

stract representation. Animal's nervous system inspires this multi-level, step-by-step abstract representation. Medical research shows that information processing of the human visual system is hierarchical. The human vision system firstly extracts edge features from lower-level regions and then extracts shapes or portions of targets in higher-level areas. That is to say, the high-level features are a complex combination of low-level features, and they are capable of expressing more abstract semantics or intentions.

Although deep learning is a relatively new term that was proposed in 2006, it actually can be traced back to the late 1940s when it was part of cybernetics. The first artificial neural point McCulloch-Pitts(McP) model [97] was proposed in 1943, which contain a single neuron node. The inputs of McP are 1 or 0 corresponding to excitatory or inhibitory. McP sums the inputs with corresponding weight and compares them to threshold T. If the sum is greater than the threshold, then output is 1: the neural point is excitated. Otherwise, the neutral point is inhibited. McP is a straightforward linear model with some fixed weights that require operator settings. As the first artificial neuron, it was not accepted by a wide range of researchers.

In the 1950s, Rosenblatt [73] proposed a purest form of feedforward neural network and named it "perceptron". Perceptron model is a binary linear classifier composed of single-layer neurons. It learns a function that is mapping from the input vector to output value. Mathematically, it was defined as,

$$f(x) = \begin{cases} 1 & if\ w \cdot x + b > 0 \\ 0 & otherwise \end{cases} \tag{2.3}$$

where x is an input vector, w is a vector of weights obtain by training and b is bias. The learning process of the perceptron can be seen as finding a boundary that is capable

of dividing hyperplane into two parts accurately. It is the first pattern recognition model with the capability of learning weights from given examples. Not only that, but the introduction of perceptron has laid foundations for multilevel feedforward neural networks.

During the same period, single linear neural network "Adaptive Linear Neuron"(ADALINE) with multiple nodes based on McP (Figure 2.2 shows the difference in structure of McP and ADALINE.) was proposed by Widrow [89]. In ADALINE, weighted sum of inputs will adjust weights along with a target output. The weights are updated as follows:

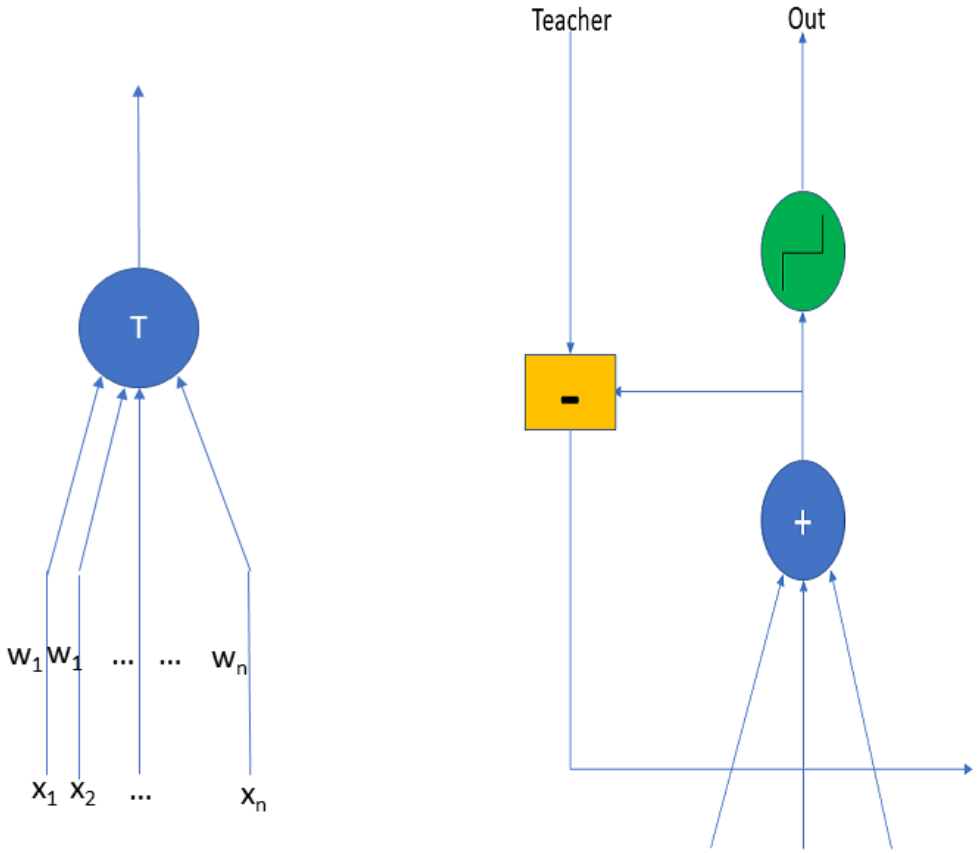
$$y = \sum_{j=1}^{n} x_j w_j \tag{2.4}$$

$$w \leftarrow w + \eta(o - y)x \tag{2.5}$$

where o is target output, y is output of model, η is learning rate, x is input vector and n is the number of the inputs. Least squares error $E = (o-y)^2$ was introduced in ADALINE as learning rule. In fact, this learning rule is the simplest form of random gradient descent algorithm.

But then, Minsky et al. [59] proved that models such as perceptrons based on linear models could not solve linear indivisible problems. For example, perceptrons don't fit exclusive disjunction (XOR) function. That is, perceptrons are unable to learn a function where $f([0,1],w) = 1$ and $f([1,0],w) = 1$ but $f([1,1],w) = 0$ and $f([0,0],w) = 0$.This conclusion makes research of artificial neural network fall into the first "winter".

In 1962, Hubel et al. [39] discovered a series of complex constructed cells in visual cortex through a study of cat's brain visual cortex. These cells are sensitive to local



(a) The structure of McP

(b) The structure of ADALINE

Figure 2.2: By comparing structure of the two models, it clearly shows that ADALINE is capable of automatically adjusting the parameters according to marked information.

areas of visual input space and are referred to as "receptive fields". The receptive field covers the entire visual domain and it plays a local role in the input space, in such a way to better extract the strong local spatial correlation existing in the natural images. Based on Hubel's model and some related neural network concepts, Fukushima et al. [85] proposed Neocognitron. Neocognitron is an iterative stack of simple cell layers and complex cell layers that accurately recognize input patterns with displacement and slight deformation as Figure 2.3 [26]. It has been demonstrated to have the ability to learn how to identify visual patterns.

The release of Neocognitron enables researchers to find potentials of multilevel

feedforward neural networks. Many of concepts in Neocognitron continue to be used to this day. In particular, the way of top-down connections guides establishment of many neural networks. Some years later, Rumelhart et al. [74] applied Back Propagation (BP) algorithm to neural network. The BP algorithm takes derivative (gradient) of loss function of neural networks and iteratively propagates gradients of each layer back through chain rules. Until today, BP remains dominant training algorithm for deep learning models. Moreover, Kurt et al. [37] demonstrate a potential of feedforward neural networks as a universal approximator. In other words, they show that multilayer perceptron overcomes the linear indivisibility problem. Because of the unremitting efforts, a cascade of researchers in artificial neural network made significant progress from the 1980s to 1990s.

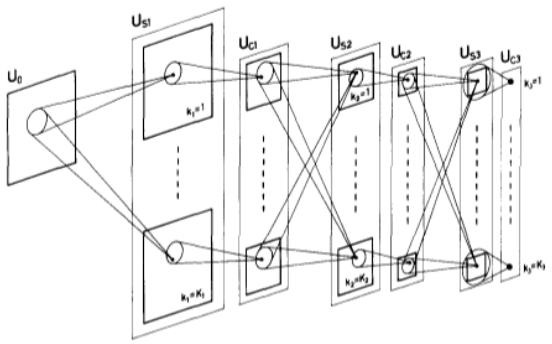


Figure 2.3: Neocognitron [26] extract representation hierarchically from image by "receptive field". Local features and these features' deformation, such as local shifts, are tolerated and integrated gradually in higher layers.

The expert systems and logistic regression models mentioned earlier require hard-coded manual features and engineering designed features. For example, Cycs is a famous expert system, but it can't understand that a person called Fred shaving in the morning. In Cycs's knowledgebase, people do not have electronic modules, so when it receives "Fred while shaving", its reasoning inference will fall into chaos and then naturally ask "whether Fred was still a person while he was shaving" [27]. A typical

application of logistic regression model is cancer prediction system, which receives some pieces of information that are defined by people and extracted manually. But it does not know how people define these features. If a logistic regression algorithm is able to receive medical images and automatically discover potential factors, then it will help doctors make decisions more effectively.

Another idea proposed by Hinton et al. [33] is Distributed representation. It emphasizes that machines learn factor directly from samples, rather than looking for relevance from features defined by human experts. In a nutshell, distributed representation means that every neuron in a neural network does not represent a feature independently. Each concept is represented by multiple activated neurons, and each neuron is also involved in many conceptual expressions. Based on the idea and BP algorithm, Lecun et al. [49] proposed a modern convolutional neural network framework: LeNet-5.

Although feedforward neural networks represented by MLP and CNN have been successful in many fields, they are unable to process information sequences because information sequences are rich in a large amount of content and every single information has a complex time correlation with each other. To solve sequential tasks, Hopfield [36] proposed a kind of temporal multi-layer neural network: Recurrent Neural Network. The biggest difference between RNN and CNN is that the connections between internal neurons in one layer are also established in RNNs. Unlike traditional deep neural networks that use different parameters at each level, RNNs shares the same parameters in all iteration steps.

That structure allows RNN models to remember the state of input information so far. Some years later, a more complete enhanced RNN model Long Short-Term Memory (LSTM) was proposed by Hochreiter and Schmidhuber [35]. Due to the

addition of the Long Short-Term Memory unit, the LSTM model has a longer memory capacity and thus repress long-term dependency problem. LSTM greatly improved accuracy and practicability of the RNN model. Initially, RNN models were applied to handwriting recognition tasks. Later, scientists found recurrent structure is skillful for timing inputs, and now they are mostly used in natural language processing.

Deep learning has entered the second winter because traditional back propagation algorithm is inefficient and tend to fall into the local minima and vanishing gradient problem. After a long period of lack of progress, introductions of the layer-by-layer pre-training algorithm and some better optimization algorithms alleviate these problems. In the last decade, developments of deep learning algorithms and hardware technology (GPU) allow training convolutional neural networks to remove the need of layer-by-layer pre-training algorithm. In 2015, Szegedy et al. [82]. introduced a convolutional neural network model with 48 layers called Inception v3 and it achieved state of art result with ILSVRC 2012. The advantages of deep networks on structures are widely studied, which significantly improve the performance of deep models in computer vision and natural language processing.

In summary, deep learning is a branch of machine learning that relies to a large extent on the human brain, statistics and applied mathematics that scientists have developed over the past few decades. In recent years, deep learning has made tremendous progress in terms of popularity and usability, mainly because of more powerful computers, larger datasets, and techniques for training deeper networks. [48]

Chapter 3

Previous Work

In this chapter, we will introduce several deep and shallow learning method frameworks which inspired our research. The foundational framework of extracting and representing features for visual task named "Convolutional Neural Network" is introduced first, followed by a discussion of its working. The next section will focus on discussing the Recurrent Neural Network (RNN). In the concluding two sections, we will introduce two outstanding contrast shallow methods: sparse coding and support vector machines.

3.1 Convolutional Neural Network

A modern convolutional neural network is a multi-layer artificial neural network. Unlike ordinary neural networks, CNN contains some feature extractors consisting of a convolutional layer and a sub-sampling layer. In the convolutional layer, one neuron

is only connected to a portion of its previous layer. A convolutional layer has several feature maps where each feature map consists of several rectangularly arranged neurons, which share same weights (The shared weights also called as convolution kernels). That weight sharing structure is used to reduce computational complexity of training process. CNN also has certain invariance to translation, scaling and other forms of deformation. Its capacity can be adjusted by changing the depth and breadth of the network.

The structure and training process of CNN will be explained in the following subsections.

3.1.1 The Structure of CNN

Figure 3.1 [70] shows a whole convolutional neural network architecture. When a colourful image (three channels input volume) is inputted to this model, the first layer of convolution kernels is convolved from upper left to lower right of the input image in fixed step size. After obtaining features by convolution, layer 1 will produce some feature maps. And the feature maps will be passed to the first pooling layer and be divided into non-overlap $m \times n$ regions. The mean (or maximum) over these regions will be obtained to pool the features. After several iterations, feature vectors will enter the fully connected layer and will be combined into larger combinations. Finally, a highly abstract representation will be classified by SoftMax classifier.

In general, modern Convolutional Neural Network structure consists of input layers, convolution layers, sampling layers, fully connected layers, output layer and so on. Convolution layers convolve their input and output feature maps to next layer. Pooling layers receive the previous output and down sample them to further abstract

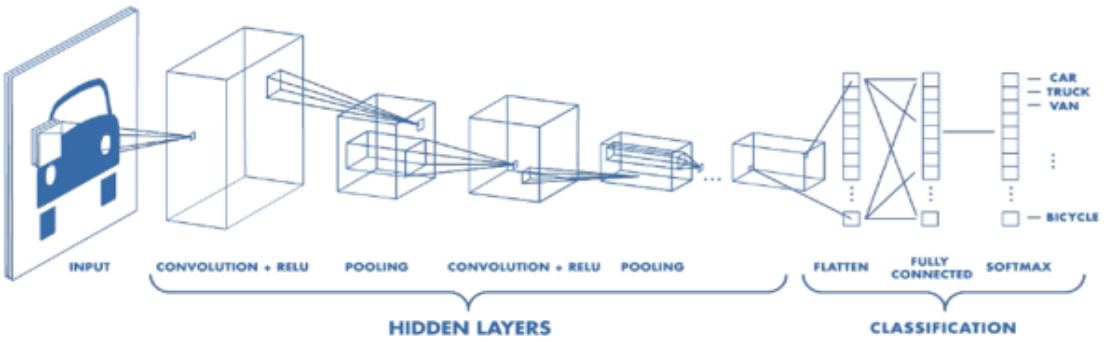


Figure 3.1: A modern convolutional neural network [70] typically consists of hidden layers and classifier except for input layer. In hidden layers, convolution and pooling operation iterative extract features and position information to produce new representation.

information. Iteratively stacked convolutional and sampling layer are followed by fully connected layers, which accept and further combine features and output final representation.

Convolutional Layer: The convolutional layer consists of a number of convolution kernels and produces feature maps. Essentially, kernel is a linear filter with a set of weights. And every kernel corresponds to a feature map. Each feature map's neuron in the convolutional layer is defined as:

$$z^l = g(w * z^{l-1} + b) \tag{3.1}$$

Where z^{l-1} is the input of current layer l, w is the kernel weights and b is bias. $g(\cdot)$ is the pointed activation function.

In convolution phase, kernels slide input volume (input image or feature maps) along width and height of input volume with a fixed step as shown in Figure 3.2 [72].

On each step, products between the entries of the kernel and the input at any position will be computed. Then, activation function will receive the results and produce a value. After the whole process is completed, we will get a feature map corresponding to the convolution kernel.

Each feature map generated by a convolutional layer is a set of learned representation of this layer. Typically, first convolutional layer extracts low-level features such as edges, lines, corners. Its feature maps represent presence or absence of edges at specific directions and positions in the input volume. Second convolutional layer typically detects combinations of edges (motif) regardless of positional change of the edges. Subsequent levels will assemble the previous motif into larger combination and abstract these large combinations into more advanced representations [48].

Pooling Layer: In a typical CNN, pooling layers usually follow a convolutional layer and consist of a plurality of feature maps. Each of them uniquely corresponds to a feature map of its previous layer. The pooling layers down samples each input feature map by the following formulas:

$$x_j^l = f(u_j^l), \quad u_j^l = \beta_j^l(x_j^{l-1}) + b_j^l \tag{3.2}$$

where f is the activation function, x_j^l is the activation value of the jth channel on the pooling layer l, x_j^{l-1} is the feature maps of previous layer, b_j^l is the bais. β is the weights of pooling layer.

As shown in Figure 3.3 [72], the feature map of the pooling layer uniquely corresponds to output feature map of the previous layer. Neurons in pooling layers are also connected to local acceptance domain of their previous layer, and accepted domains of different neurons without overlap. Pooling layers aim to obtain spatially invariant

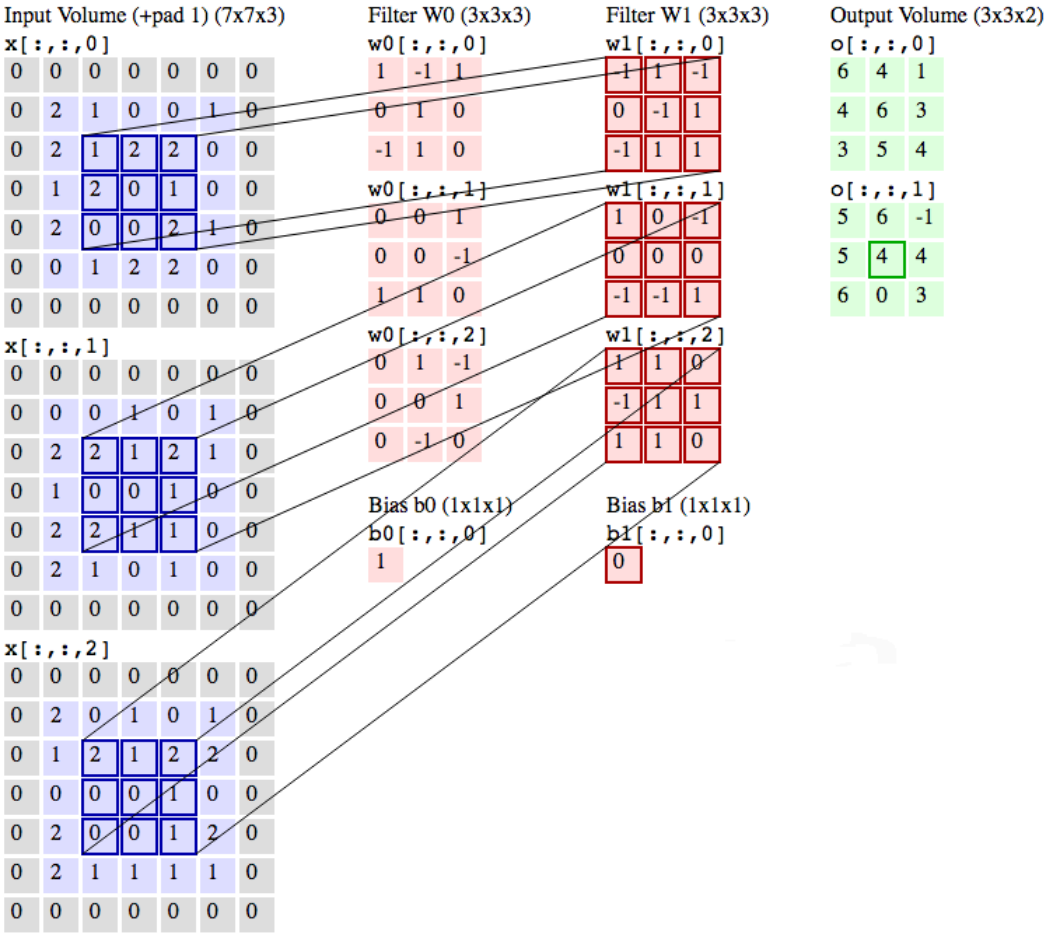


Figure 3.2: The working principle of convolution process, from [72]. The 3 channelled input volume are convoluted by 2 filters with biases. A filter slice corresponds to an input channel, where each local region is mapped to a specified location by the filter on output volume. The number of filters determines the depth of the output volume.

features by reducing dimensions of feature maps. Also, pooling layer can gradually reduce the size of the representation space and thus reduce parameters in the network and control overfitting. The commonly used pooling methods are max-pooling, that is, taking the largest value in a local accepting domain, and mean pooling, that is, averaging all values in a local accepting domain.

Fully Connected Layer: In general, fully connected layers follow iteratively convolutional and pooling layers on the tail of CNN structure. Each neuron is fully

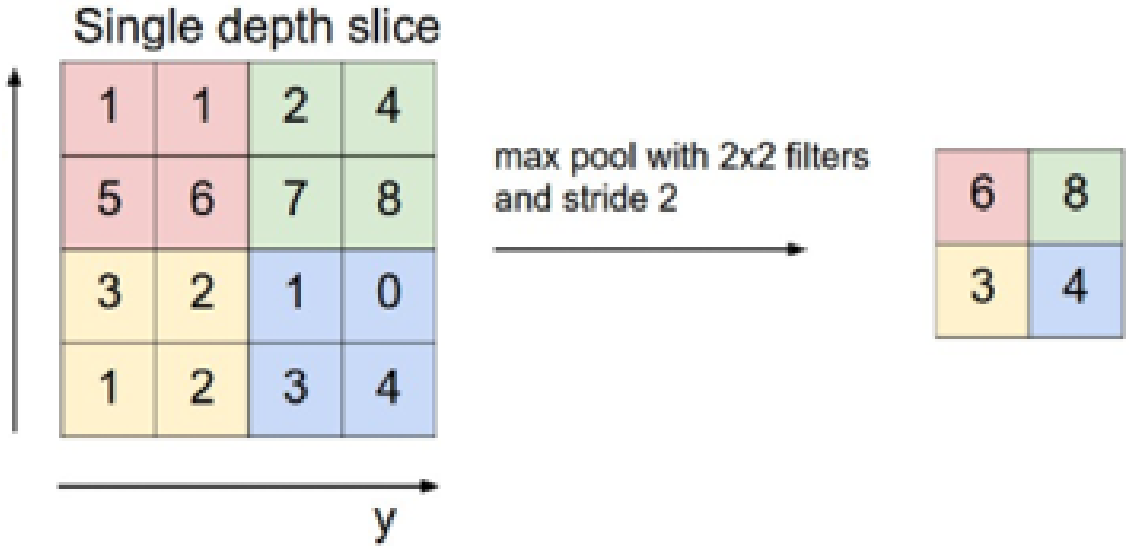


Figure 3.3: The image [72] shows a slice of input volume of pooling layer is pooled with maximum value. In essence, the slice is divided into 4 Non-overlapping local regions with the same size, in where the maximum value will be mapped into the new feature map.

connected to all neurons of previous layers. The fully connected layers integrate representation of input volume from previous layer and produce a highly abstract final representation.

In summary, CNN models are capable of directly receiving raw data and then implicitly learn from training data to avoid accumulation of errors caused by manual extraction of features. Through weight sharing, CNNs reduce the amount of trainable weights and computational complexity of networks. Introduction of pooling operation also makes CNNs have certain invariance to local transformation of inputs, such as translation invariance and scaling invariance, which improves generalization ability of the networks. In the past 30 years, the convolutional neural network model has been continuously researched and it has always surprised us.

3.1.2 The Training of CNN

The training process of convolutional neural networks is mainly to learn kernel weights and network parameters such as inter-layer connection weights. CNN conducts supervised learning through back propagation algorithm. The back propagation algorithm is essentially an iterative learning algorithm that updates any parameter by iterative computation. It adjusts weights along the negative gradient direction of the weights for minimizing target loss function.

BP algorithm could be split into two different parts including error calculation and weight updating. In the error calculation phase, representation of training samples is firstly propagated layer by layer to output layer in forward passage phase. After that, errors between outputs and given labels will be calculated and back propagated from last layer to first layer by the chain rule. Finally, weights are updated along the negative gradient direction with a learning rate.

Assume an L-layer network with Activation σ and a Loss function, in which layer l has inputs a^{l-1} and the Activation function with its input z^l the training process of this network is shown below:

Algorithm 1 Back Propagation Algorithm

Input: Network with L layers, Activation function, Loss function, Learning rate α, Maximum iterations Max, Stop threshold ϵ and Training dataset with m samples :$(x_1, y_1), (x_2, y_2) \cdots, (x_m, y_m)$

Output: Weights Matrix W and Bias b

some description

for iter=1 to Max **do**

  for i=1 to m **do**

    for l=2 to L **do**

      forward propagation to calculate $a^{i,l} = \sigma(z^{i,l}) = \sigma(W^l a^{i,l-1} + b^l)$

      calculate gradient of output layer by Cost function: $\delta^{i,L}$

    for l=L-1 to 2 **do**

      Back propagation to calculate: $\delta^{i,l} = (W_{l+1})^T \delta^{i,l+1} \bigodot \sigma'(z^{i,l})$

  for l=2 to L **do**

    update layer l parameters W^l, b^l:

$$W^l = W^l - \alpha \sum_{i=1}^{m} \delta^{i,l} (a^{i,l-1})^T \tag{3.3}$$

$$b^l = b^l - \alpha \sum_{i=1}^{m} \delta^{i,l} \tag{3.4}$$

  if the changing of W and b less than ϵ **then**

    return W and b

return W and b

3.2 Recurrent Neural Network

Although feedforward neural networks represented by MLPs and CNNs have been successful in many fields, they are burdensome to process information sequences. Information sequences are rich in a large amount of content, every single information has a complicated time correlation with each other, and the length of single information varies. To solve sequential tasks, recurrent neural networks were proposed on the 1980s, which are a kind of temporal neural network. The most significant difference between RNNs and convolutional neural networks is that connections between internal neurons in one layer are also established in RNNs. Unlike traditional deep neural networks that use different parameters at each level, RNNs share the same parameters in every iterative step. This property reflects a fact that RNNs model performs same task at each step with different inputs. The particular structure reduces the total amount of learnable parameters that make them applicable to tasks such as speech recognition, natural language processing connected handwriting.

3.2.1 The Structure of RNN

An RNN is unfolded into a full network based on time. The numbers of layers depend on the length of the input sequence. In RNN, the states of neurons and the current inputs together affect the output of neurons; each neuron seems to have a memory unit which captures information from the beginning until present moment. Compared to traditional feedforward neural network, RNN has better the capability to capture more extended historical information.

Figure 3.4 [78] shows an architecture of typical RNN, where x is a sequential input,

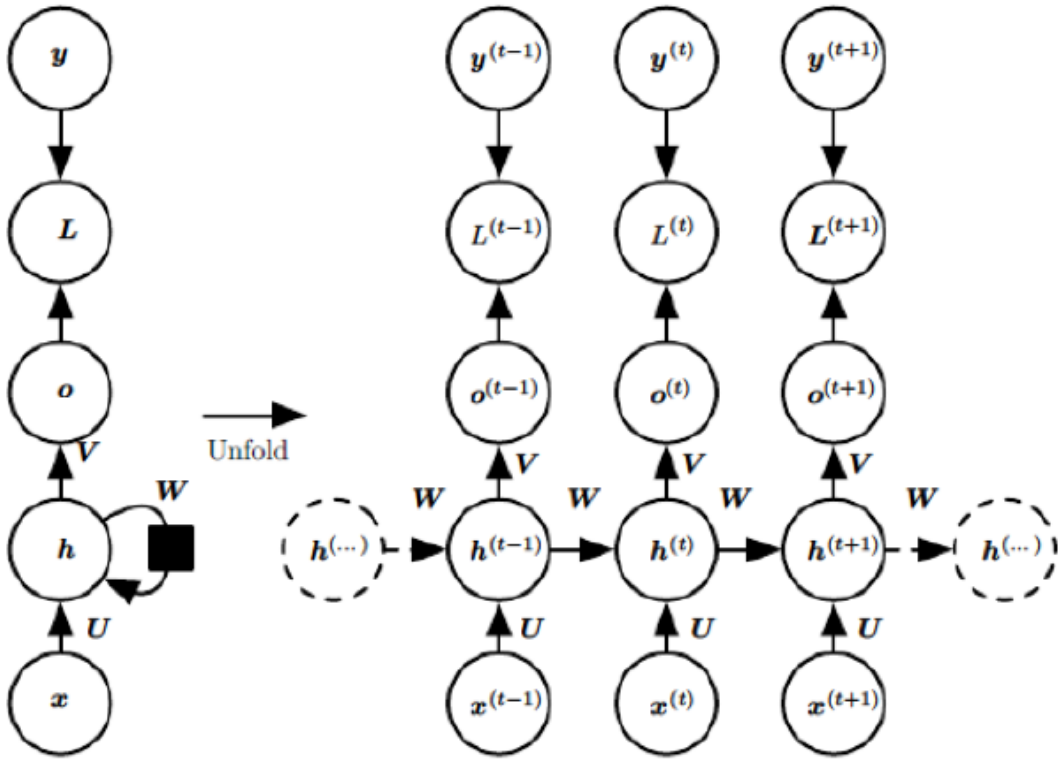


Figure 3.4: The depth of multilayer recurrent neural network [78] is determined by length of input sequence x. Current input$(x^{(t)})$ and state of neurons at time t_1 together affect output at time t.

h is a hidden unit, o is output of the network. Here L also called loss function which is pointed. And y is a label set of training samples. V, W, and U are weights, and same type of connection weight is same. The entire network is unfolded based on times. It is clearly shown that the output of unit h at time t is not only determined by the input of this moment, but also by time before t.

In forward propagation phase, the state of unit h at time t is,

$$h^t = \phi(Ux^{(t)} + Wh^{(t-1)} + b), \tag{3.5}$$

where $\phi(\cdot)$ is an activation function of unit h and b is bias. The output of the unit in

t is,

$$o^{(t)} = Vh^{(t)} + c, \tag{3.6}$$

and the final output of RNN model is,

$$\hat{y}^{(t)} = \sigma(o^{(t)}), \tag{3.7}$$

where $\sigma()$ is the activation of output layer.

3.2.2 The Training Process of RNN

Training RNN is similar to training convolutional neural network. The Back Propagation Through Time (BPTT) algorithm is a commonly used method for training RNN. It is essentially also a gradient descent method. The core of the algorithm is to continuously search for better results along the negative gradient direction of parameters until the change in parameters is sufficiently small. Since every time step share same parameters, current gradient of parameters is not only caused by current time step calculation, but also by past errors. For example, in order to calculate gradient at time $t3$, gradients of all previous time steps (step 1 and step 2) are calculated by back propagation.

Formally speaking, since the loss occurred in each time step of sequence, the final loss L could be defined as,

$$L = \sum_{t=1}^{n} L^{(t)}, \tag{3.8}$$

where n is the total numbers of steps. So that the gradient of V at time t is:

$$\frac{\partial L}{\partial V} = \sum_{t=1}^{n} \frac{\partial L^{(t)}}{\partial o^{(t)}} \cdot \frac{\partial o^{(t)}}{\partial V} \tag{3.9}$$

Since the loss is cumulative, the partial derivative of the entire loss for W and U are:

$$\frac{\partial L^{(t)}}{\partial W} = \sum_{k=0}^{t} \frac{\partial L^{(t)}}{\partial o^{(t)}} \frac{\partial o^{(t)}}{\partial h^{(t)}} \Big(\prod_{j=k+1}^{t} \frac{\partial h^{(j)}}{\partial h^{(j-1)}} \Big) \frac{\partial h^{(k)}}{\partial W} \tag{3.10}$$

$$\frac{\partial L^{(t)}}{\partial U} = \sum_{k=0}^{t} \frac{\partial L^{(t)}}{\partial o^{(t)}} \frac{\partial o^{(t)}}{\partial h^{(t)}} \Big(\prod_{j=k+1}^{t} \frac{\partial h^{(j)}}{\partial h^{(j-1)}} \Big) \frac{\partial h^{(k)}}{\partial U} \tag{3.11}$$

Finally, parameters are updated along negative gradient direction by back propagation algorithm.

Since Vanishing Gradient Problem often occurs when training RNN, researchers always use four effective ways to learn RNN including long-short term memory, hessian free optimization, echo state networks, and proper initialization with momentum [94].

3.2.3 Deep Belief Network

Deep Belief Network is a probabilistic graphical model that learn to extract deep hierarchical representation from raw data [47]. It is composed of multiple layers of hidden units with directed and undirected edges. DBNs are always used for unsupervised learning which is similar to encoders or used for supervised learning as classifiers. From the perspective of unsupervised learning, DBNs are to preserve features of original data as much as possible while reducing dimensions of the features. From the perspective of supervised learning, DBNs aim to make classification error

rate as small as possible. In either case, DBNs training process is essentially a feature learning process, which leads to get better feature representation.

In general, a DBN model is a deep network composed of several stacked Restricted Boltzmann Machines (RBM) and a cascade layer. RBM is a generated network with a visible layer and a hidden layer. Figure 3.5 [80] shows a typical RBM that consists of visible units (corresponding to visible variables, i.e., data samples) and hidden units (corresponding to hidden variables). The entire RBM is a bipartite graph with some internal connections between visible and hidden cells.

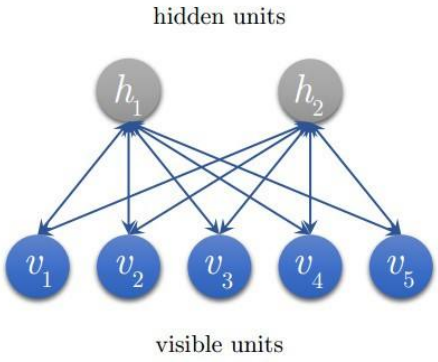


Figure 3.5: In Restricted Boltzmann Machine, visible units is connected with hidden units by a directional way.

RBM model was explained as an energy-based probabilistic model with an energy function. The energy function is a measurement that describes states of the whole system. The more ordered the system or, the more concentrated the probability distribution is, the smaller the energy of the system is. Conversely, the more disordered the system or, the more uniform the probability distribution is, the higher the energy of the system is. Energy function was defined as:

$$E(v, h) = -h'Wv - b'v - c'h \tag{3.12}$$

where W represents weights of connection between visible units and hidden units and b, c is offset of visible and hidden units, respectively. Based on the energy function, probability of visible-hidden units was obtained as:

$$P(v,h) = \frac{\exp^{-E(v,h)}}{Z} \tag{3.13}$$

where

$$Z = \sum_{v,h} \exp^{-E(v,h)} \tag{3.14}$$

Z is normalization factor, also known as a partition function. Assuming the visible and hidden units are conditionally independent given one-another, hence:

$$P(h|v) = \prod_i p(h_i|v), \qquad P(v|h) = \prod_j p(v_j|h) \tag{3.15}$$

Both visible and hidden units are with binary states (active or not), i.e. their states h_i and v_j are taken to be either $0, 1$. Based on previous equations((3.12) to (3.15)) probabilistic activation functions of neuron were obtained as:

$$P(h_i = 1|v) = sigm(c_i + W_i v) \qquad P(v_j = 1|h) = sigm(b_j + W'_j h) \tag{3.16}$$

where $sigm()$ is the sigmoid function.

The goal of training RBM is to minimize system energy, this is, to maximize log-likelihood function of the RBM on the training set. Hence, parameters of RBM were updated along the negative gradient direction of the log-likelihood function like other networks. In general, Contrastive Divergence algorithm (CD) [34] is the most popular

approach to train DBN. In other words, training process of RBM is actually to find a probability distribution that is able to produce training samples. Since decisive factor of this distribution is the weight W, the goal of training RBM is to find the best weight.

DBNs training process is mainly divided into two steps. At first, we should separately train the RBMs by a layer-wise approach to ensure that feature information is preserved as much as possible when feature vectors are mapped to a different feature space. As shown in Figure 3.6 [12], the first layer of RBM is trained with the original input data and the input feature vectors are represented by V_0, which is mapped to another feature space H_0. The extracted features H_0 are then trained as input V_1 of the second layer RBM to obtain the second layer reconstructed features H_1. After some iterations, the topmost cascading layer receives the output feature vectors of RBMs as its input feature vector, and then, it will be trained by the supervised back propagation algorithm. Each layer of RBMs only ensures that its weights are optimal for its feature vector mapping. Therefore, when the cascade layer is training, errors will be propagated from the top to bottom to each RBM layer, and the whole network will be fine-tuned. The training process of DBN models can be regarded as the initialization of the back propagation networks' weights, which makes DBN overcome shortcomings of the back propagation networks which tend to fall into local optimums.

3.3 Sparse Coding

Sparse coding is a typical unsupervised dictionary learning method designed to learn a dictionary consist of overcomplete bases to represent data. As a particular edge

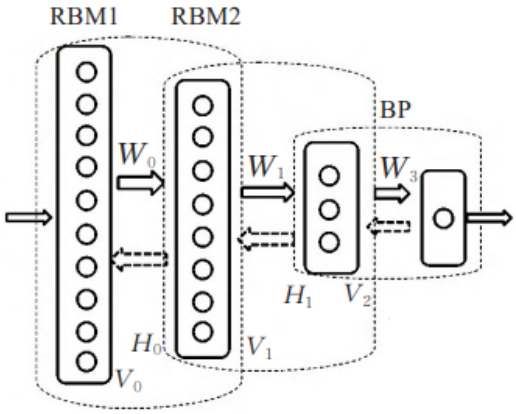


Figure 3.6: Through layer-by-layer pre-training algorithm, the DBN [12] achieves learning goal, that is, hidden units of RBM1 are first trained, and then they are used as visible units to train RBM2 iteratively.

detector, it is capable of extracting boundary features from natural images effectively. In simple terms, sparse coding models reconstruct input data into a linear combination of a set of over complete basis vectors with coefficients, and the coefficients as new features will represent the input data. Sparse coding is an effective method of representation learning.

Although Principal Component Analysis (PCA) enables us to easily find a set of "complete" bases vectors to describe input data, it uses second-order statistical properties between data. This is a global transformation method that is unable to include phase information. In sparse coding approach, overcomplete bases are capable of finding structural and potential patterns in input data more effectively. Formally speaking, sparse coding is to represent an input signal x as a linear combination of a set of over complete bases $dx = Da$ where a is a sparse representation of x. Sparse

encoding cost function with m input data is defined as:

$$\begin{aligned} \underset{a_i^{(j)},\phi_i}{\text{minimize}} \quad & \sum_{j=1}^{m} ||x^{(j)} - \sum_{i=1}^{k} a_i^j \phi_i|| + \lambda \sum_{i=1}^{k} |a_i^j| \\ \text{subject to} \quad & ||\phi_i||^2 \leq C, \forall i = 1, \cdots, k \end{aligned} \tag{3.17}$$

where $\sum_{j=1}^{m} ||x^{(j)} - \sum_{i=1}^{k} a_i^j \phi_i||$ is reconstructed term, which guarantees that error between the new representation and the original input is as small as possible. $|a_i^j|$ is a penalty term, which is used to penalize sparse coefficient that far from zero.

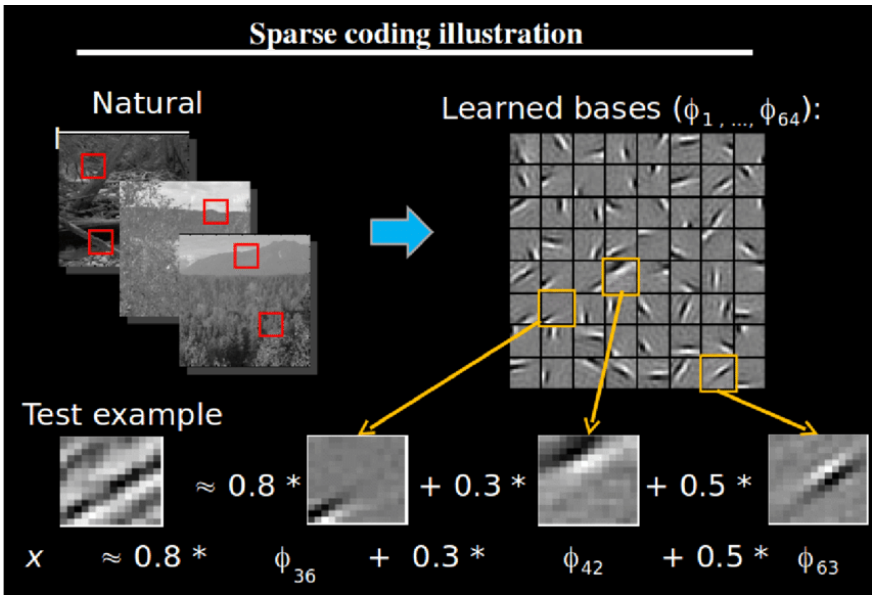


Figure 3.7: Sparse Coding illustration, from [86]. The original local region of the image will be approximated as much as possible through combinations of the learned bases($\phi_1, \cdots, \phi_n$).

In general, sparse coding approach includes two phases: training and coding. The goal of training is learning a set of over complete bases Φ (dictionary) through minimizing the objective function 3.17 by two iteratively independent optimization processes. At first, coefficient a is optimized by the fixed dictionary Φ base on training set. And then, the learned coefficient vector a will optimize the dictionary Φ on the

same training set. By iterating through until the error between the reconstructed representation and the original data smaller than a specified threshold, the model will obtains a set of over complete bases (dictionary), that able to well represent the training set x. In the representation phase (as shown on Figure 3.7 [86]): Given a new data x, the coefficient a is obtained by minimizing the objective function 3.18 based on the dictionary D. a will as the sparse representation of x.

$$\underset{a}{\text{minimize}} \quad ||x - \sum_{i=1}^{k} a_i \phi_i|| + \lambda \sum_{i=1}^{k} |a_i| \tag{3.18}$$

3.4 Support Vector Machine

Support vector machine (SVM) is a supervised machine learning method based on VC dimension theory and Structural Risk Minimization. In general, the SVM model attempts to find the best hyperplane on feature space to classify samples. Compared to neural networks, SVM has a solid theoretical foundation and a simple straightforward mathematical model, which makes it highly respected in the field of statistical learning.

Structural risk minimization and VC dimension are theoretical cornerstones of SVM. The VC dimension of hypothesis space H is the cardinality (size) of the largest set of points that H can shatter on instance space X. For example, a line is capable of breaking three points into 2^3 kinds of results but can not break four points into 2^4 kinds of results, so the VC dimension of the line is three.

The error accumulation between predictions and real solutions is called risk. When a classifier is selected, the real error is not known, but it can be approximated by some known amount. The most intuitive way is employing the difference between results from classifier and actual labels. The difference is also called Empirical Risk. Empirical Risk with a penalty term composes a basic Structural Risk function.

The goal of SVM models is to find an optimal classification hyperplane so that the hyperplane is capable of maximizing Margin on both sides of the hyperplane while ensuring classification accuracy. In theory, support vector machines are capable of achieving the optimal classification for linearly separable data. Suppose the given SVM training dataset is $(x_1, y_1), \cdots (x_1, y_n)$, with $x \in R^m$ and $y \in \{-1, 1\}$. Then the task is a typical binary-class problem. Suppose the data is divided by a hyperplane:

$$w^T x + b = 0 \tag{3.19}$$

where w is a unit vector and b is the bias. Therefore the distance from a point x to the hyperplane is:

$$r = \frac{|w^T x + b|}{||w||} \tag{3.20}$$

Assuming the hyperplane can correctly classify the training samples, for any x_i:

$$\begin{cases} w^T x_i + b \geq +1, & y_i = +1 \\ w^T x_i + b \leq -1, & y_i = -1 \end{cases} \tag{3.21}$$

As shown in Figure 3.8 [98], the training sample points closest to the hyperplane that makes 3.21 hold are called support vector. The separation between the hyperplane and the closest data point for a weight vector w and bias b is called as Margin:

$$\gamma = \frac{2}{||w||} \tag{3.22}$$

The task of support vector machine model is to find w and b to maximize the margin:

$$\begin{aligned} &\max_{w,b} \quad \frac{2}{||w||} \\ &\text{subject to} \quad y_i(w^T x_i + b) \geq 1, \quad i = \forall i = 1, 2, \cdots, m. \end{aligned} \tag{3.23}$$

Maximizing the γ is equivalent to minimizing $||w||^2$ so that the learning problem becomes to the optimization problem. Through Lagrange function, the original problem can be transformed into dual problem of convex quadratic programming:

$$\begin{aligned} &\text{minimize} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j \\ &\text{subject to} \quad \alpha_i \geq 0, i = 1, \cdots, n \\ &\qquad\qquad \sum_{i=1}^{n} a_i y_i = 0 \end{aligned} \tag{3.24}$$

where α is Lagrange multipliers. And there is unique solution α^*, w^* and b^* for this formula. Based on α^*, SVM will learn an optimal classification function:

$$f(x) = sgn((w*)^T x + b^*) = sgn(\sum_{i=1}^{n} \alpha_i^* y_i x_i^* x + b^*) \tag{3.25}$$

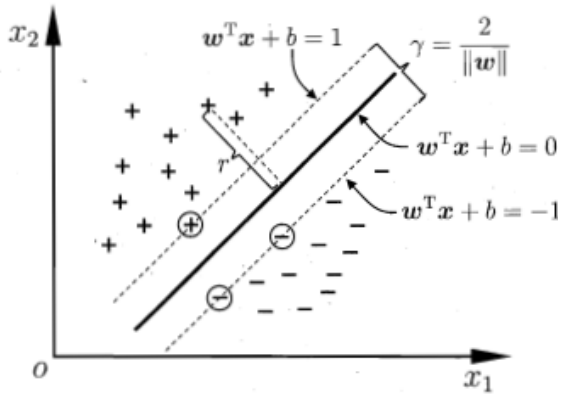


Figure 3.8: $w^T x + b = 0$ is the hyperplane [98]. The distance between the positive and negative classes (distance of two dashed lines to the solid line) is the Margin.

Chapter 4

Learning Models

This chapter introduces four outstanding deep architectures and one shallow architecture model that we have used in our experiments. In section 4.1 and 4.2, two deep models based on Convolutional Neural Network are introduced. The third section will introduces a Recurrent neural network model with initialized parameters. Then, a network based on deep belief network model we implemented will be described. The last section introduces a shallow sparse coding model and a special SVM model to compare with deep learning models.

4.1 Standard CNN

We implemented a complex neural network based on LetNet-5 and named Standard Convolutional Neural Network, which also demonstrates satisfied capabilities on the CIFAR-10 dataset. It consists of two convolutional layers, each followed by a pooling

layer. The average pooling or maximum pooling will be adjusted according to different dataset.

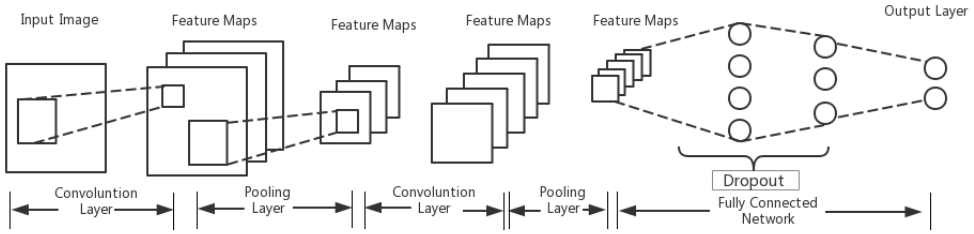


Figure 4.1: This image shows the structure of the Standard CNN in which two convolution layers with Relu, and max-pooling, dropout tricks were gathered to improve performance of the network.

The feature map output from the second pooling layer will be flattened and fully connected to the next layer. Other fully connected layers will be further combined with abstract feature vectors. A method called dropout is applied between the fully connected layer to improve performance of the neural network.

Figure 4.1 shows overall structure and work process of Standard CNN. As shown, input data will be convolved by some kernel of size $5 * 5$ to detect and extract basic features. To reduce effects of the displacement on feature extraction, feature maps will then be pooled in to the second layer.

Feature maps, produced from the second layer, will then again convolved and pooled to get more abstract features.After flattening, features of input data are presented by a highly abstract vector, which will be passed to the fully connected layers. During training process, Dropout is used to shut down each node with a certain probability.

In other words, a certain percentage of fully connected neurons are turned off during each training session. But, all fully connected neurons are turned on during recognition (classification) process, which guarantees performance of the entire neural network.

4.2 MIN

MIN, proposed by Chang and Chen [10], is a deep convolutional network based on the framework of Network In Network structure. The MIN model has been shown to have outstanding capability of feature extraction performance on MNIST [50] and CIFAR-10[43] datasets, with the state-of-art result on MNIST dataset without data augmentation.

MIN is composed of stacking three feed forward MIN blocks and a SoftMax classifier as Figure 4.2 [10] shown. On each MIN block, multilayer perceptrons (MLP) are used as universal approximators to extract features and obtain abstract representations from receptive fields with rectifier units. The Maxout [28] units are capable of approximating arbitrary convex functions and mediating vanishing gradients problem and are used as rectifier units to approximate piecewise linear activation functions in MIN block. To reduce saturation of the Maxout units, data batches are normalized before passing to the hidden layer and after processing from the hidden layer.

In the MIN block, inputs data are convolved by convolution layer first. Then, feature maps will be cross-channel pooled by the Maxout units. In each channel, batch normalization layer will normalize the feature maps to zero mean with unit variance. After repeating above operations twice, feature maps will be normalized

and output to the pooling layer. MIN block-wise extract and combine features and output highly abstract representation to the classifier.

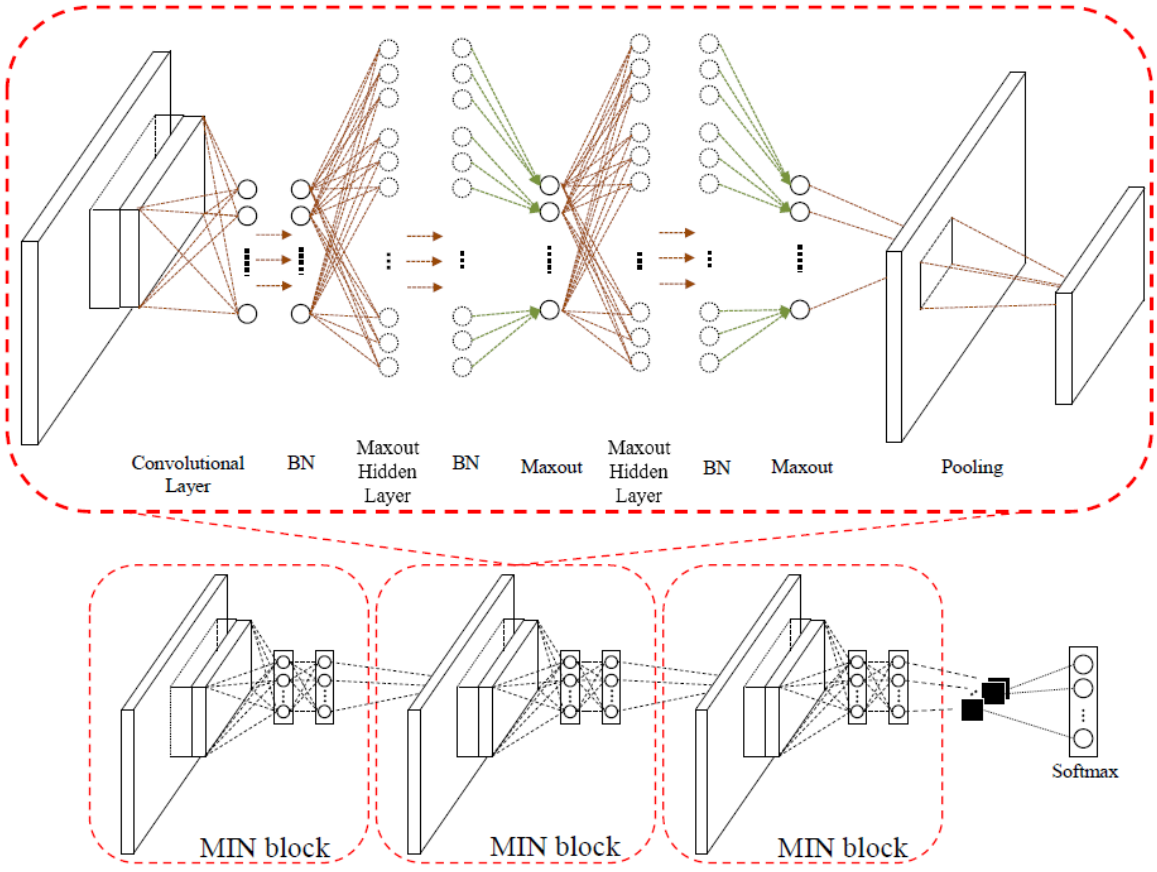


Figure 4.2: Based on the Network IN Network structure, MIN [10] includes 3 blocks with batch normalization and Maxout tricks. In MIN, MLP as universal approximator with rectifier units to extract features. The feature maps will be pooled and pasted to SoftMax classier.

4.3 IRNN

IRNN was proposed by Le et al. [46], which aims to overcome some problems like Vanishing gradient to learn long-range dependencies. The model introduced rectified linear units as activation functions on recurrent neural network. Furthermore, a new way aims initialize recurrent weight matrix by identity matrix with biases 0, were proposed in the model.

In the RNN model, current hidden state vector is obtained by simply copying previously hidden vector then adding on effects of current inputs and replacing all negative states by zero [46]. In other words, the weights of hidden units remain constant when no extra error-derivatives are added on the stage of error derivatives backpropagation through time. IRNN has shown an awesome ability to learn local dependence on MNIST and one billion words language modeling dataset [11].

4.4 DBN

A DBN model was implemented and conducted on experiments based on the Deep Belief Network framework by Theano [2]. It is a probabilistic generative model consisting of two Restricted Boltzmann Machines (RBM) with associative memory. The topmost associative memory consists of a multilayer perception machine.

Except for the top layer (associative memory), each layer encodes the distribution of previous units and could reconstruct its input. The final representation of the input vector is produced by layer-wise transformation from the lowest to the highest layer. As shown in Figures 4.3, each RBM layer contains 1000 units without connection with

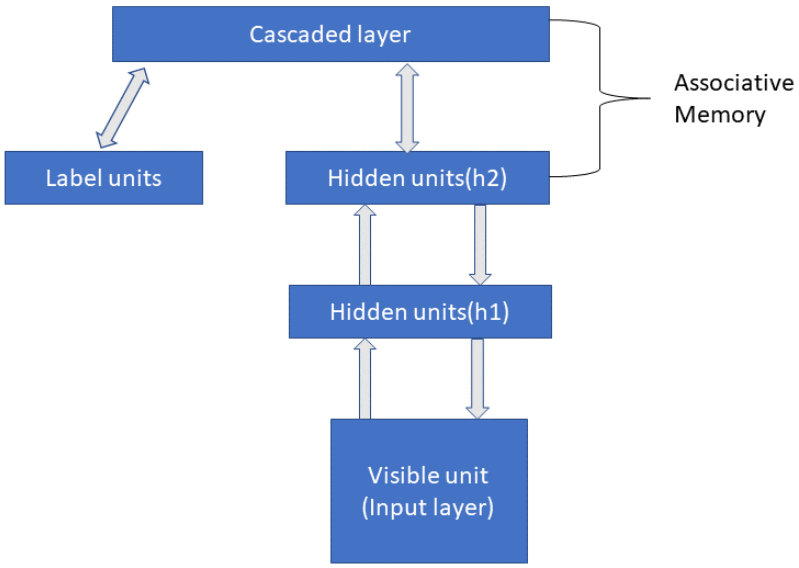


Figure 4.3: The image shows our DBN structure, in where 1000 units of each hidden layer ensure that the original data can be reconstructed as much as possible.

each other. An unsupervised greedy layer-wise algorithm (CD) and BP algorithm are employed to train this model. In training process, raw data are first used to train the hidden layer of first RBM to reconstruct them. After that, the hidden layer of the first RBM is used as the visible layer of the second RBM to train it continuously. The two trained RBMs model connected to a cascade layer with 2000 units. The cascade layer connected to the second RBM and label units is used to obtain the gradients of the cost function. Finally, the BP algorithm also propagates error information from top to bottom to adjust the DBN.

4.5 Shallow Learning Models

4.5.1 Sparse Coding

The work of any sparse coding method can be reformatted into a two-stage process: training stage and mapping stage. In the training stage, a Dictionary(D) is learned from a training set with some training algorithms; in the mapping stage, features are extracted from inputs based on the trained dictionary and then mapped into a new representation space with a mapping function. We employ the coding model proposed by Coates and Ng [17] as a comparative framework. In the framework, a Dictionary is obtained by unsupervised learning algorithm that alternately optimizing the objective function,

$$\begin{aligned} \underset{D,s(i)}{\text{minimize}} \quad & \sum_i ||Ds^{(i)} - x^{(i)}||_2^2 \\ \text{subject to} \quad & ||D^j||_2^2 = 1, \forall j; \\ & ||s^{(i)}||_0 \leq k, \forall i \end{aligned} \tag{4.1}$$

where $x^{(i)}$ is input vector, $s^{(i)}$ is new representation code of $x^{(i)}$. $||s^{(i)}||_0 \leq k$ means that each representation code has at most k non-zero elements and $||D^j||_2^2$ means that the dictionary elements D^j all have unit length. The unsupervised orthogonal matching pursuit algorithm (OMP) [69] will then be used to approximate S and D. After optimizing S and D alternately, we can get a complete dictionary D. Given dictionary D, final representations of the input x are obtained by a fixed threshold α,

$$\begin{cases} f_j = \max\{0, {D^{(j)}}^\top x - \alpha\} \\ f_{j+d} = \max\{0, -{D^{(j)}}^\top x - \alpha\} \end{cases} \tag{4.2}$$

wheref_j and f_{j+d} are final representations of the input x.

Finally, the representation of features will be passed to L2_SVM classifier to produces category of each sample.

4.5.2 RBF_SVM

In this thesis, Radial Basis Function kernel was implemented to increase performance of the SVM model, and named as RBF_SVM. RBF_SVM maps/transits sample space into a high-dimensional feature space. The nonlinear transformation is defined as:

$$\begin{aligned} K(x^{(i)}, x^{(j)}) =& \phi(x^{(i)})^T \phi(x^{(j)}) \\ =& \exp(-\gamma ||x^{(i)} - x^{(j)}||^2) \end{aligned} \tag{4.3}$$

where $\phi()$ is radial basis function and γ is the learning coefficient. Then, the learning problem becomes the optimizing object function:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, k_j) \\ \text{subject to} \quad & \alpha_i \geq 0, i = 1, \cdots, n \\ & \sum_{i=1}^{n} a_i y_i = 0, 0 \leq a_i \leq C \end{aligned} \tag{4.4}$$

Where C is a threshold to be specified. Finally, the learnt classification function of RBF_SVM model is:

$$f(x) = sgn((w*)^T \phi(x) + b^*) = sgn(\sum_{i=1}^{n} \alpha_i^* y_i K(x_i, x) + b^*) \tag{4.5}$$

Chapter 5

Experiments and Results Analysis

We conducted all deep and shallow models on the public MNIST and CIFAR-10 datasets. In this chapter, the experiment setting as well as the result analysis will be discussed.

5.1 Experiments

5.1.1 Data Formation

We expect our experiments will conduct on datasets which have been thoroughly studied by using Deep Learning approaches. Many real data do not have proximity between neighbouring elements (e.g.,[29, 9, 56]), but there are no experiments using Deep Learning approaches on these dataset. To make a fair comparison, we used

image data, CIFAR and MNIST, which have been used for testing DL approaches, but converted into neighbouring proximity unpreserved data as follows: for any data in the dataset, (X, Y), where $X = [x_{ij}] \in \mathbf{R^{m \times n}}$ represent a 2-d image, and Y represents the category. We convert X:

$$X = \begin{pmatrix} x_{11} & \dots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \dots & x_{mn} \end{pmatrix} \tag{5.1}$$

into X^w:

$$X^w = F * X^t = \begin{pmatrix} 0 & \dots & 1 \\ \vdots & \ddots & \vdots \\ 0 & 1 & 0 \end{pmatrix} * \begin{pmatrix} x_{11}, & \dots & x_{1n}, & x_{21}, & \dots & x_{mn} \end{pmatrix}^T \tag{5.2}$$

where F is a randomly generated 0-1 sparse matrix of size $mn \times mn$, with only one 1 item in each row and each column, representing a random permutation of pixels. Finally, the data X^w will be reshaped into a $m \times n$ matrix X^f, which is now neighbouring proximity unpreserved.

5.1.2 Experiments on MNIST

MNIST is a classic handwriting recognition data set. It contains 10 categories of 60,000 training samples and 10,000 test samples without extra illumination and stains. It has been shown by experiments that the Deep learning approaches (e.g., [15, 52, 79]) have outstanding ability to representation and expression the MNIST dataset.

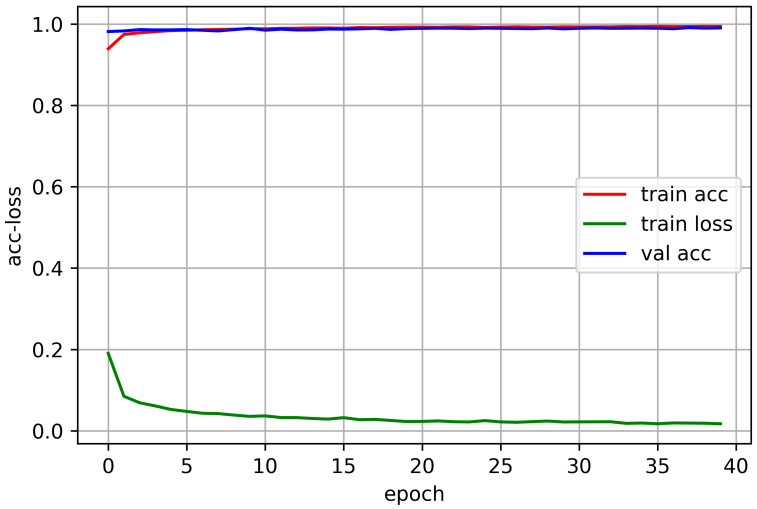
The MINST data (X, Y) was converted into (X^f, Y) to removing neighbouring proximity by above permutation operation before experiment. We called the new dataset as F-MNIST. We conducted all above deep learning methods on F-MNIST. In standard CNN, all convolution kernels are of size is 5×5 and pooling is averaged. And we applied 10^{-2} as the learning rate with a decay rate of 10^{-6}. IRNN only accepts one pixel at each time step, each sample were transferred to a vector of size 784×1. We applied 10^{-6} as the learning rate for achieving faster convergence and performance improvement. At the same time, the original optimizer is replaced by the RMSprop, as recommended by [13], which yields more stable and steady improvement. IRNN stops until convergence or after 1,687,500 iterations. In DBN, we stacked two RBMs with a multilayer perceptron. Every RBM has 1000 units. Since F-MNIST is a lightweight dataset, we took the RBF_SVM model with $C = 5$ and $\gamma = 0.05$ to classify neighbouring proximity unpreserved dataset F-MNIST.

Table 5.1: Test accuracy and decline rate of all methods on F-MNIST

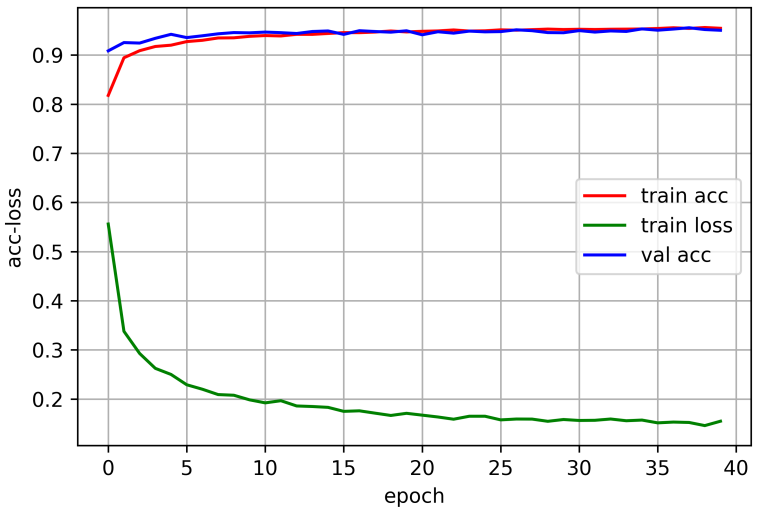
Methods	Test accuracy	Decline rate
Standard CNN	95.43%	3.62%
MIN	97.98%	1.78%
IRNN	72.68%	20.82%
DBN	98.44%	0.31%
RBF-SVM	98.37%	0%
SVM-ploy9	98.60%	0%
RBN-LSTM	95.40%	3.60%

The results are shown in Table 5.1. It is clear that the simple shallow model, RBF-SVM, has the better performance than other DCNN and RNN models. Furthermore, another shallow model SVM with ploy9 [49] achieves the best accuracy. As in the third column, the decline rate is the difference of accuracy between applying the original dataset and the F-MNIST. From Table 5.1, we can find that the accuracies of all deep

structure methods are reduced. However, the shallow methods are not affected.



(a) Standard CNN on MNIST

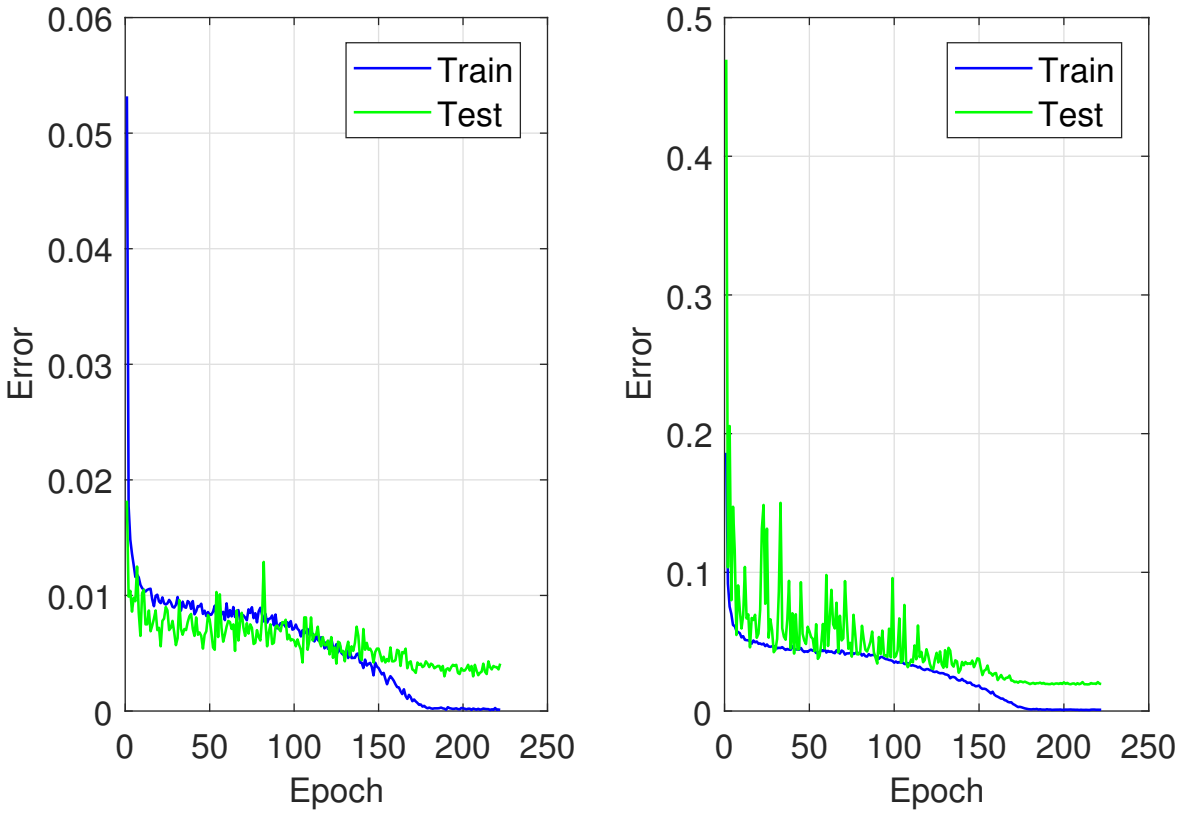


(b) Standard CNN on F-MNIST

Figure 5.1: The training process of Standard CNN on the original MNIST and F-MNIST. The abscissa indicates the number of iterations of whole dataset.

We also evaluated CNN, MIN and DBN performance on both the original MNIST

dataset and the neighbouring proximity unpreserved dataset F-MNIST. The results are shown in Figure 5.1 to Figure 5.3. From these results, we can find that the performances of CNN, MIN and IRNN converge on F-MNIST dataset are not better than their performance on original MNIST dataset. Some of the methods even become unstable.



(a) MIN on MNIST

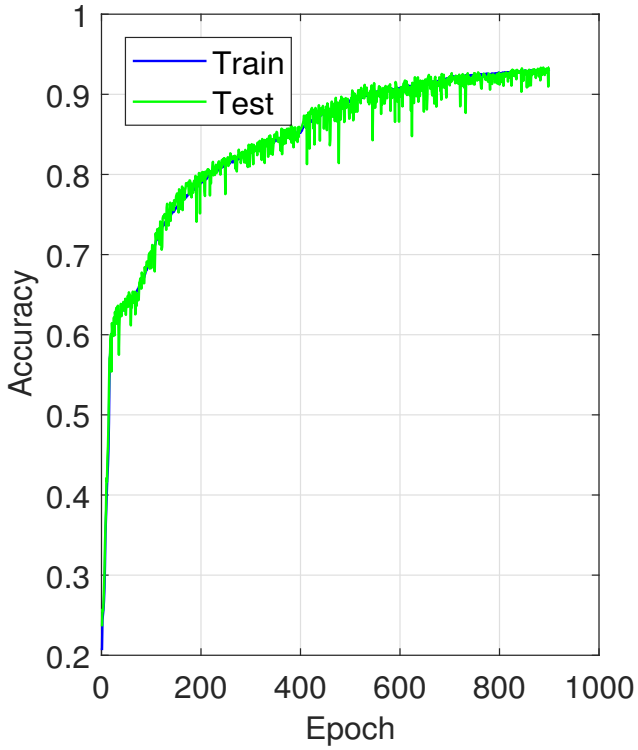
(b) MIN on F-MNIST

Figure 5.2: The results of MIN on original MNIST and F-MNIST.

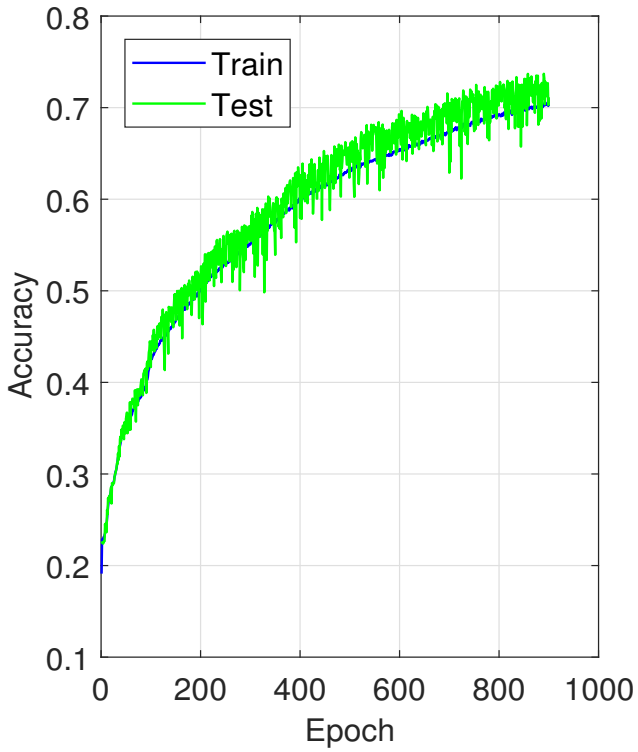
Table 5.2 shows the runtime[1] of deep and shallow methods on the MNIST (F-MNIST) dataset. It is important to note that all deep methods are speed up by a high-performance graphic card[2]. Even so, IRNN took much longer than RBM-SVM (If

[1]The runtime of all methods on this paper is calculated based on the same operating platform: Intel(R) Xeon(R) 2609v3, 80GB RAM.

[2]Nvidia Titan Xp



(a) IRNN on MNIST



(b) IRNN on F-MNIST

Figure 5.3: The results of IRNN on original MNIST and F-MNIST.

Table 5.2: The runtime of deep and shallow methods on F-MNIST

Methods	Runtime(unit:hours)
Standard CNN	0.5
MIN	4.35
IRNN	112.5
DBN	0.75
RBF-SVM	0.27

we remove the graphic card, the runtime of standard CNN with the simplest structure in all deep methods is over five hours). In the sense of efficiency, the shallow methods are much better than deep methods on the MNIST (F-MNIST) dataset.

5.1.3 Experiments on CIFAR-10

The CIFAR-10 dataset consists of 10 categories of natural colour images. It is divided into 50000 training and 10000 test images. It has been removed the neighbouring proximity by the formation operation introduced by Section 5.1.1. we called the new dataset as F-CIFAR-10.

In standard CNN, 10^{-3} was applied as the learning rate with decay rate of 10^{-6}. For the best performance of MIN, we applied the structure and parameters proposed by [10]. For IRNN, each neutron unit accepts three intensity values (i.e. RGB) at each time step. We have tried a variety of different combinations of parameters in order to achieve the best performance. For the shallow approach, we following the coding model explained in Section 4.5.1. The standard process of [17] is applied: extracting 6×6 local patches with stride 1 to yield a bank of feature vectors, then, normalizing and applying Zero-Phase Component Analysis (ZCA) whitening. Dictionary is trained by the feature vectors with OMP-1 algorithm (i.e. $k = 1$); the feature representation

is accepted with a soft threshold $\alpha = 0.25$ for encoding. Finally, global representation is form through average-pooling on a 2×2 grid and passed to SVM for classification.

The experiment results are shown in Table 5.3. We can easily see from the table that all the shallow coding method achieves best results.

Table 5.3: The test accuracy and decline rate of deep and shallow methods on F-CIFAR-10

Methods	Test accuracy	Decline rate
Standard CNN	48.81%	19.07%
MIN	56.68%	35.47%
IRNN	36.67%	9.45%
DBN	49.48%	0.32%
Coding+SVM	58.16%	23.69%

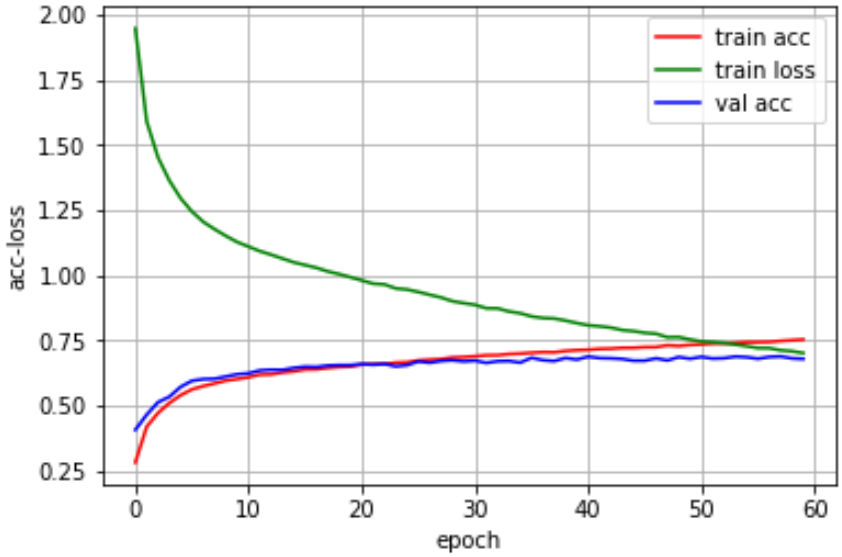
Figure 5.4 to Figure 5.6 show the comparison results of Standard CNN, MIN and IRNN on the original CIFAR-10 and on the neighbouring proximity unpreserved dataset F-CIFAR-10. We get similar results with the experiments with MNIST; the deep methods on neighbouring proximity unpreserved dataset F-CIFAR-10 did not stabilize as it would on the original dataset. Moreover, IRNN needed more iterations to converge. The runtime of all deep and shallow methods on the CIFAR-10 (F-

Table 5.4: The runtime of deep and shallow methods on F-CIFAR-10

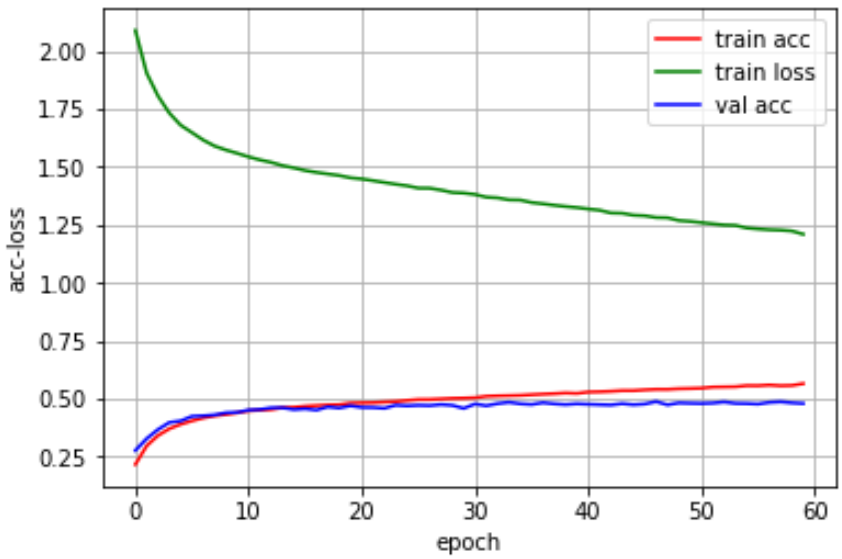
Methods	Runtime(unit:hours)
Standard CNN	0.5
MIN	9.18
IRNN	191.97
DBN	0.77
Coding+SVM	4.62

CIFAR-10) is reported in Table 5.4. We can see that although the runtime of the

shallow method was increased, it was still shorter than the runtime of standard CNN without GPU (over 9 hours).

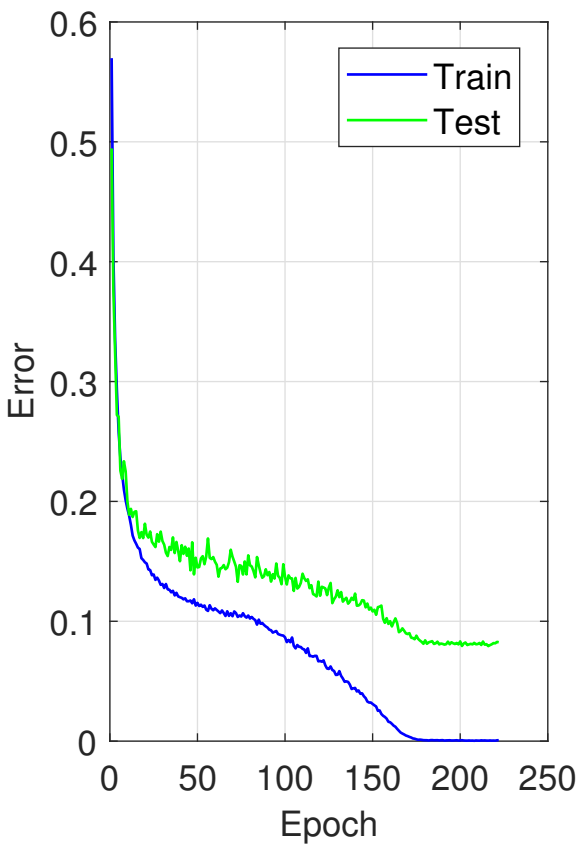


(a) Standard CNN on CIFAR-10

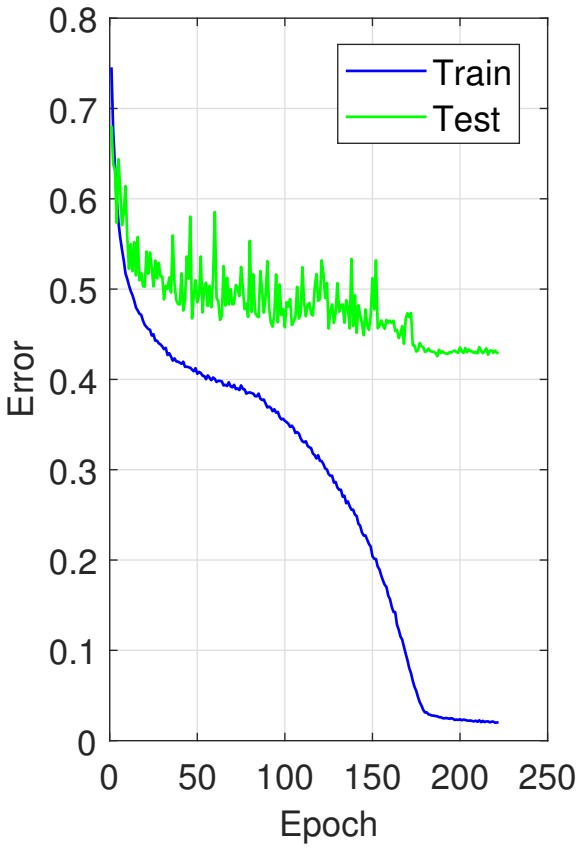


(b) Standard CNN on F-CIFAR-10

Figure 5.4: The train process of Standard CNN on CIFAR-10 and F-CIFAR-10.

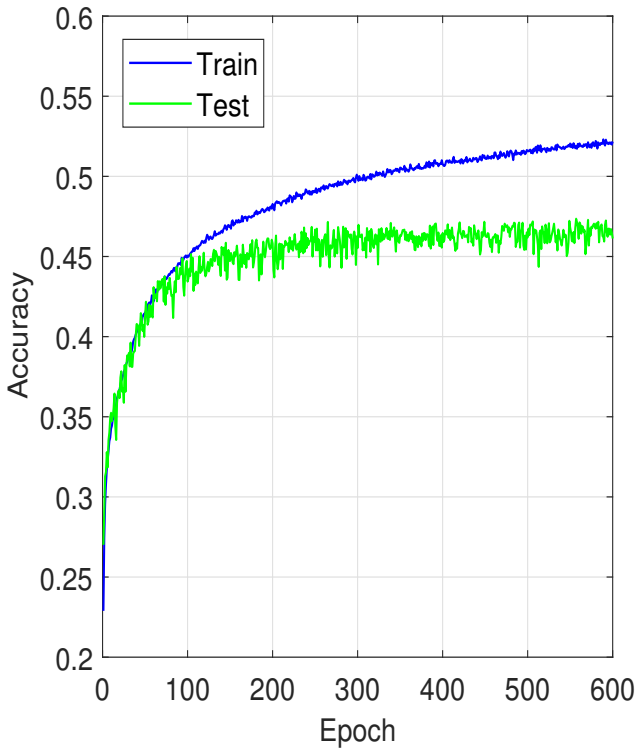


(a) MIN on CIFAR-10

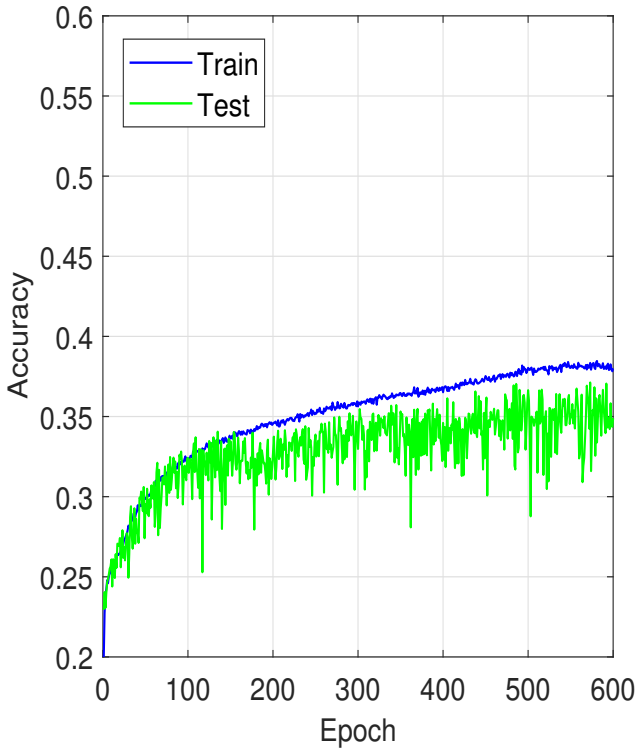


(b) MIN on F-CIFAR-10

Figure 5.5: The results of MIN on original CIFAR-10 and F-CIFAR-10.



(a) IRNN on CIFAR-10



(b) IRNN on F-CIFAR-10

Figure 5.6: The results of IRNN on original CIFAR-10 and F-CIFAR-10.

5.2 Possible Reasons For The limitations of Deep Methods

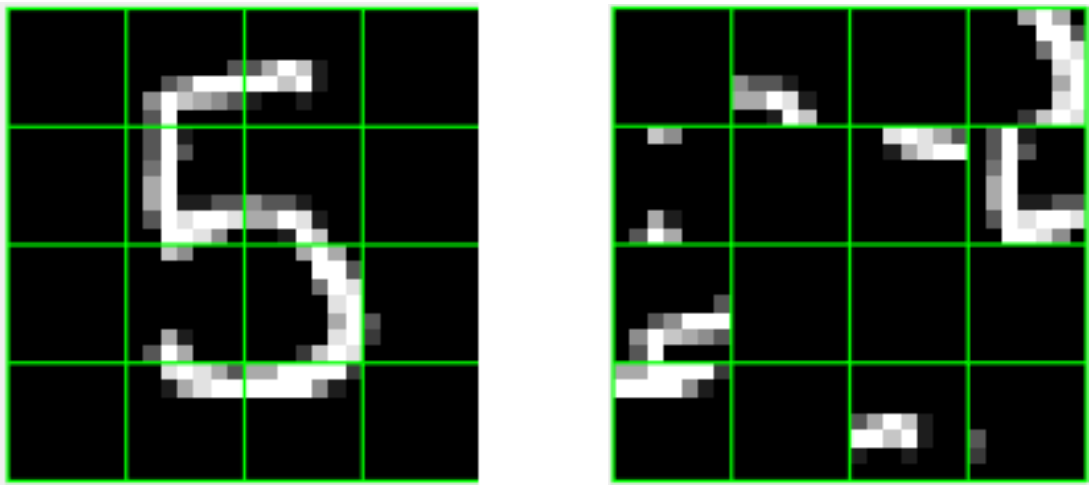
5.2.1 CNN

In the CNN hierarchical structure, the inputs of the present layers' neurons are actually a dense set of outputs of previous layers' local patches with local dependency. Essentially, CNN achieves global dependencies and global representations by iteratively stacking local patches dependencies and representations. The neighbouring proximity enables information extractions from local patches with limited training samples. However, as shown in the experiments of Section 5.1, the local dependencies (i.e., neighbouring proximity) do not exist starting from the first layer (the inputs do not have neighbouring proximity). Therefore, the kernels of local patches lost their power in extracting meaningful information from limited training samples. Based on that, regardless of the number of layers is used, the global dependency cannot be effectively extracted, which affect the capacity of global representations.

5.2.2 IRNN

RNN is capable to "remember" long histories if it is provided enough capacity. However, large capacity means large-scales, and large-scale neural networks always encounter difficulties in optimization ([6, 68]). Therefore, the length of the dependencies caused the performance drop of RNN. In order to verify this hypothesis, we carried an advanced experiment: in data formation phase, we gradually scaled up the operation

units.



(a) Original MNIST data

(b) New generated data

Figure 5.7: Each green block is an atomic operation unit. Original blocks are shuffled with a random permutation to generate new data without neighbouring proximity

Table 5.5: The performance of IRNN conducted on the shuffled MNIST with different operation unit

Unit	Test accuracy
1×1	70.82%
2×2	79.63%
4×4	81.38%
7×7	83.25%
14×14	87.52%

As shown in Figure 5.7, original MNIST data (image matrix) was divided into several blocks (i.e. operating unit). Then, the blocks of every MNIST data are shuffled with a fixed random permutation. In order to retain more neighbours of elements, the size of these blocks was gradually scaled up. The final results are shown in Table 5.5. From Table 5.5, we can see that the performance of the RNN improves gradually as the operating unit increasing.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Deep learning has become one of the most popular methods in many applications (e.g., natural language processing, pattern recognition and generation, computer vision, etc.), especially since it was introduced into self-driving. In the past two decades, the advantages of deep learning have been continuously explored and advanced structural components have been introduced, but its limitations have not been well studied. Leading to the idea that traditional approaches are completely replaced by deep learning approaches.

In this thesis, we have performed some outstanding deep structure methods, such as Standard CNN, MIN, IRNN, and DBN, on datasets where proximity is removed and compared results with such methods running on the original datasets. The comparison clearly shows that these deep structures have significantly reduced performance on

the datasets without neighbouring proximity. Also, the most important conclusion obtained with results produced by shallow method is that the performance of the deep methods is worse than the shallow methods when applied to neighbouring proximity unpreserved datasets.

Even though RNN has the capacity of long distance memory, in practice the "length" of long distance memory is still very limited. This leads to problems with input size 30×30, if neighbouring proximity is removed and the performance of RNNs is far behind traditional shallow approaches. In such situations, deep structures do not have any advantages over traditional shallow methods.

Therefore, based on the series of contradistinctive experiments, we can conclude that proximity among neighbouring elements significantly affects the performance of deep structures. And our experiments show potential risks of applying deep learning to autopilot and also give a warning to deep learning's safety. At last, we should not expect that deep approaches will end traditional AI and machine learning research.

6.2 Future Work

The conception of further improvements for the deep learning approach is listed below:

- Introducing a new operator: Our experiments show that the deep structure model is not sensitive to neighbouring proximity unpreserved dataset. In the future, new mapping/learning operators can be further introduced to improve the performance of deep structure models.
- Combining with shallow structure methods: The dataset could be preprocessed

by the shallow methods before it is applied to the deep structure model.

- Streamlining parameters: Reduce the parameters to improve the training speed of the deep structure model.

Bibliography

[1] Axxonsoft face intellect facial recognition and face search with intellect enterprise. `https://blogs.sap.com/2018/02/08/machine-learning-in-a-box-week-3-algorithms-learning-styles`, 2018. [Online; accessed 05-December-2018].

[2] R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov, A. Belopolsky, et al. Theano: A python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*, 2016.

[3] V. E. Barker, D. E. O'Connor, J. Bachant, and E. Soloway. Expert systems for configuration at digital: Xcon and beyond. *Communications of the ACM*, 32(3):298–318, 1989.

[4] H. B. Barlow et al. Possible principles underlying the transformation of sensory messages. *Sensory communication*, 1:217–234, 1961.

[5] Y. Bengio et al. Learning deep architectures for AI. *Foundations and trends in Machine Learning*, 2(1):1–127, 2009.

[6] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with

gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[7] J. M. Bishop. History and philosophy of neural networks. *Computers & Graphics*, 37(5):348–363, 2013.

[8] L. Breiman. *Classification and regression trees*, pages 1–17. Routledge, 2017.

[9] J. Catlett. Statlog (Shuttle) Dataset. `https://archive.ics.uci.edu/ml/datasets/Statlog+(Shuttle)`.

[10] J.-R. Chang and Y.-S. Chen. Batch-normalized maxout network in network. *arXiv preprint arXiv:1511.02583*, 2015.

[11] C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn, and T. Robinson. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2013.

[12] W. N. L. X. C. Y. Chen Liang, Zhang Junchi. Deep belief networks based popularity prediction for online video services. *Computer Engineering and Applications*, 53(9):162–169, 2017.

[13] F. Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

[14] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[15] D. Ciregan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2012)*, pages 3642–3649. IEEE, 2012.

[16] W. F. Clocksin and C. S. Mellish. *Programming in Prolog: Using the ISO standard*. Springer Science & Business Media, 2012.

[17] A. Coates and A. Y. Ng. The importance of encoding versus training with sparse coding and vector quantization. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 921–928, 2011.

[18] T. Cooijmans, N. Ballas, C. Laurent, Ç. Gülçehre, and A. Courville. Recurrent batch normalization. *arXiv preprint arXiv:1603.09025*, 2016.

[19] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[20] A. Dadouche. Machine learning in a Box (week 3) : Algorithms learning styles. `https://blogs.sap.com/2018/02/08/machine-learning-in-a-box-week-3-algorithms-learning-styles/`, 2018.

[21] M. Dekker. Models of Learning Systems. *Encyclopedia of Computer Science and Technology*, 11:24–51, 1978.

[22] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition(CVPR 2009)*, 2009. `http://www.image-net.org/papers/imagenet_cvpr09.bib`.

[23] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[24] P. Farquhua. Sydney airport is rolling out facial recognition technology even as its use is still being debated in parliament. `https://www.businessinsi`

der.com.au/sydney-airport-is-rolling-out-facial-recognition-technology-even-as-its-use-is-still-being-debated-in-parliament-2018-6, 2018. [Online; accessed 02-December-2018].

[25] Y. Feng and M. Lapata. Automatic caption generation for news images. *IEEE transactions on pattern analysis and machine intelligence*, 35(4):797–812, 2013.

[26] K. Fukushima and S. Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982.

[27] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*, page 2. MIT Press, 2016. http://www.deeplearningbook.org.

[28] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013.

[29] F. Graf, H. P. Kriegel, M. Schubert, S. Poelsterl, and A. Cavallaro. Relative location of CT slices on axial axis Dataset, 2010. http://archive.ics.uci.edu/ml/datasets/Relative+location+of+CT+slices+on+axial+axis.

[30] I. Guyon, V. Vapnik, B. Boser, L. Bottou, and S. A. Solla. Structural risk minimization for character recognition. In *Advances in neural information processing systems*, pages 471–479, 1992.

[31] 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*, pages 770–778, 2016.

[32] G. E. Hinton. Deep belief networks. *Scholarpedia*, 4(5):5947, 2009.

[33] G. E. Hinton, J. L. McClelland, D. E. Rumelhart, et al. *Distributed representations*. Carnegie-Mellon University Pittsburgh, PA, 1984.

[34] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[35] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[36] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[37] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[38] T. huang. Expert systems. In *The History of Artificial Intelligence*, chapter 4, pages 12–16. 2006. `https://courses.cs.washington.edu/courses/csep590/06au/projects/history-ai.pdf`.

[39] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154, 1962.

[40] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[41] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An introduction to statistical learning*, volume 112, pages 1–14. Springer, 2013.

[42] C.-B. Jin, S. Li, and H. Kim. Real-time action detection in video surveillance

using sub-action descriptor with multi-cnn. *arXiv preprint arXiv:1710.03383*, 2017.

[43] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[44] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[45] F. Kung. Tensorflow Image Recognition in ROS. `https://medium.com/@k3083518729/tensorflow-image-recognition-58b0ac77c263`. [Online; accessed 03-December-2018].

[46] Q. V. Le, N. Jaitly, and G. E. Hinton. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015.

[47] D. learning 0.1 documentation. deep belief networks. `http://deeplearning.net/tutorial/DBN.html`, 2018. [Online; accessed 03-December-2018].

[48] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[49] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[50] Y. LeCun, C. Cortes, and C. J. Burges. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.

[51] C. W. Leong and R. Mihalcea. Going beyond text: A hybrid image-text approach for measuring word relatedness. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1403–1407, 2011.

[52] M. Liang and X. Hu. Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3367–3375, 2015.

[53] M. Lin, Q. Chen, and S. Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[54] R. Lindsay, B. Buchanan, E. Feigenbaum, and J. Lederberg. Applications of artificial intelligence for organic chemistry. 1980.

[55] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[56] P. Mahé, M. Arsac, S. Chatellier, V. Monnin, N. Perrot, S. Mailler, V. Girard, M. Ramjeet, J. Surre, B. Lacroix, et al. Automatic identification of mixed bacterial species fingerprints in a maldi-tof mass-spectrum. *Bioinformatics*, 30(9):1280–1286, 2014.

[57] S. Merity, N. S. Keskar, and R. Socher. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*, 2017.

[58] L. C. Michael Martinez. Voice technology: As google duplex wows and scares, a post-screen world emerges with questions that the smart speakers cannot answer, 2018. Online; accessed 02-December-2018.

[59] M. Minsky and S. A. Papert. *Perceptrons: An introduction to computational geometry*. MIT press, 2017.

[60] V. Mittal. Top 15 Deep Learning applications that will rule the world in 2018 and beyond. `https://medium.com/@vratulmittal/top-15-deep-lea`

rning-applications-that-will-rule-the-world-in-2018-and-beyond-7c6130c43b01, 2017. [Online; accessed 02-December-2018].

[61] J. M. Moguerza, A. Muñoz, et al. Support vector machines with applications. *Statistical Science*, 21(3):322–336, 2006.

[62] S. Muggleton. Inductive logic programming. *New generation computing*, 8(4):295–318, 1991.

[63] S. Muggleton. Inductive logic programming: derivations, successes and shortcomings. *ACM SIGART Bulletin*, 5(1):5–11, 1994.

[64] S. Muggleton, C. Feng, et al. Efficient induction of logic programs. In *Proceedings of the First Conference on Algorithmic Learning Theory*, 1990.

[65] Y. Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.

[66] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325, 1997.

[67] S. C. Park, M. K. Park, and M. G. Kang. Super-resolution image reconstruction: a technical overview. *IEEE signal processing magazine*, 20(3):21–36, 2003.

[68] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318, 2013.

[69] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*, pages 40–44. IEEE, 1993.

[70] Prahu. Understanding of convolutional neural network (cnn)deep learning. https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148, 2018.

[71] C. A. Rayed and A. S. Embark. A proposed expert system for evaluating the partnership in banks. *International Journal of Advance Robotics & Expert Systems (JARES)*, 1(2).

[72] C. C. N. N. F. V. Recognition. Convolutional Neural Networks (CNNs / ConvNets). http://cs231n.github.io/convolutional-networks/. [Online; accessed 03-December-2018].

[73] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[74] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533, 1986.

[75] E. Y. Shapiro. The model inference system. In *Proceedings of the 7th international joint conference on Artificial intelligence-Volume 2*, pages 1064–1064. Morgan Kaufmann Publishers Inc., 1981.

[76] E. Shortliffe. *Computer-based medical consultations: MYCIN*, volume 2. Elsevier, 2012.

[77] Y. Y. Song and Y. Lu. Decision tree methods: applications for classification and prediction. *Shanghai Arch Psychiatry*, 27(2):130–135, Apr 2015.

[78] Srikanth. Simple implementation of lstm using tensorflow-stock price dataset. https://learn2codewithmesite.wordpress.com/2018/01/15/stock-price-prediction-using-lstm/, 2018. [Online; accessed 03-December-2018].

[79] R. K. Srivastava, K. Greff, and J. Schmidhuber. Training very deep networks. In *Advances in neural information processing systems*, pages 2377–2385, 2015.

[80] X. Sun, X. Peng, and F. Ren. Detect the emotions of the public based on cascade neural network model. In *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, pages 1–6, 2016.

[81] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[82] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[83] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[84] M. Turcotte, S. H. Muggleton, and M. J. Sternberg. Use of inductive logic programming to learn principles of protein structure. *Computer and Information Science*, 5(39), 2000.

[85] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity*, pages 11–30. Springer, 2015.

[86] D. Vassallo. *BioRFID: A Patient Identification System using Biometrics and RFID*. PhD thesis, The University of Liverpool, 2016.

[87] G. T. Vesonder, S. J. Stolfo, J. E. Zielinski, F. D. Miller, D. H. Copp, et al. Ace: An expert system for telephone cable maintenance. In *IJCAI*, volume 8, page 983, 1983.

[88] Y. Wang, C. P. Rose, A. Ferreira, D. M. McNamara, R. L. Kormos, and J. F. Antaki. A Classification Approach for Risk Prognosis of Patients on Mechanical Ventricular Assistance. *Proc Int Conf Mach Learn Appl*, pages 293–298, Dec 2010.

[89] B. Widrow et al. *Adaptive "adaline" Neuron Using Chemical "memistors"*. Standford University, 1960. `http://www-isl.stanford.edu/~widrow/papers/t1960anadaptive.pdf`.

[90] Wikipedia. Anthony Oettinger — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Anthony%20Oettinger&oldid=838223197`, 2018. [Online; accessed 02-December-2018].

[91] Wikipedia. Decision tree — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Decision%20tree&oldid=869963629`, 2018. [Online; accessed 03-December-2018].

[92] Wikipedia. General Problem Solver — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=General%20Problem%20Solver&oldid=856579631`, 2018. [Online; accessed 03-December-2018].

[93] Wikipedia. Logic Theorist — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Logic%20Theorist&oldid=869549813`, 2018. [Online; accessed 02-December-2018].

[94] Wikipedia. Vanishing gradient problem — Wikipedia, the free encyclope-

dia. http://en.wikipedia.org/w/index.php?title=Vanishing%20gradient%20problem&oldid=884430479, 2019. [Online; accessed 15-April-2019].

[95] S. J. Winter. Methodology development of an engineering design expert system utilizing a modular knowledge-base inference process. *Retrospective Theses and Dissertations*, 1998.

[96] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[97] L. Zhang and B. Zhang. A geometrical representation of mcculloch-pitts neural model and its applications. *IEEE Transactions on Neural Networks*, 10(4):925–929, 1999.

[98] Z. Zhihua. *Machine Learning*, volume 1, page 12. Tsinghua University Press, 2016.

[99] M. Zihlmann, D. Perekrestenko, and M. Tschannen. Convolutional recurrent neural networks for electrocardiogram classification. *Computing*, 44:1, 2017.