A Review of Deep Learning Research

Ruihui Mu[1,2*], Xiaoqin Zeng[1]
[1]College of Computer and Information, Hohai University
Nanjing 210098, P.R.China
[e-mail: muruihui@126.com, xzeng@hhu.edu.cn]
[2]College of Computer and Information Engineering, Xinxiang University
Xinxiang 453000, P.R.China
*Corresponding author: Ruihui Mu

Received July 5, 2018; revised September 15, 2018; accepted October 19, 2018; published April 30, 2019

Abstract

With the advent of big data, deep learning technology has become an important research direction in the field of machine learning, which has been widely applied in the image processing, natural language processing, speech recognition and online advertising and so on. This paper introduces deep learning techniques from various aspects, including common models of deep learning and their optimization methods, commonly used open source frameworks, existing problems and future research directions. Firstly, we introduce the applications of deep learning; Secondly, we introduce several common models of deep learning and optimization methods; Thirdly, we describe several common frameworks and platforms of deep learning; Finally, we introduce the latest acceleration technology of deep learning and highlight the future work of deep learning.

Keywords: Deep learning, machine learning, artificial intelligence, learning model , neural network, optimization method

1. Introduction

With the rapid development of technologies such as cloud computing, Big data, and Internet of things, the emergence of various applications in the Internet space has led to the explosive growth of data scale [1]. According to the report by the International Data Corporation (IDC) in 2012, global total data is expected to be 22 times 2011, reaching 35.2ZB by 2020 [2]. Big data contains rich value and great potential, it will bring about transformation and development for human society, but it also brings serious problems of "information overload". How to quickly and efficiently obtain valuable information from various complex data has become a key challenge. In recent years, deep learning has made breakthroughs in image processing, natural language understanding and speech recognition [3].

In view of the rapid development of deep learning technology, deep learning can map different data to the same hidden space by performing automatic feature learning from multi-source heterogeneous data, then obtains unified data representation [4]. This paper reviews deep learning techniques from various aspects, including common models of deep learning and their optimization methods, commonly used frameworks, existing problems and future research directions.

The remainder of this paper is organized as follows. Section 2 reviews the related applications of deep learning. In Section 3, we address the common models of deep learning. In Section 4, we describe the optimization methods. In Section 5, we introduce commonly used open source frameworks and platforms of deep learning. Section 6 presents acceleration technology of deep learning. Section 7 describes the problems and prospects of deep learning. Section 8 concludes our work and outlines potential future work.

2. Related Applications

2.1. Image Processing

Image recognition is the earliest application of deep learning. In 2014, Dong et al. [5] first proposed the use of deep convolutional neural networks to learn the end-to-end mapping relationship between low-resolution images and high-resolution images for image recognition. DenKer [6] used convolutional neural network to recognize handwritten digital and achieved the best results at the time. Ren et al. [7] proposed the Faster R-CNN object detection method by using a deep convolutional neural network. In 2015, Wang et al. [8] introduced sparse priors into deep convolutional neural networks for image recognition. Yu et al. [9] used Autoencoder to classify images and trained support vector machines for image classification. The accuracy of deep learning exceeded 97% in the 2016 ImageNet

Competition [10]. Xia et al. [11] proposed a supervised depth hashing algorithm called CNNH (Convolutional Neural Network Hashing) for image recognition.

2.2. Speech Recognition

In recent years, deep learning technology has applied in the fields of speech recognition. In the field of speech recognition, Baidu, HKUST, and SOGOU all announced their accuracy of Chinese speech recognition based on deep learning exceeded 97% at the end of 2016. Microsoft's research on speech recognition based on deep neural network has completely changed the original technical framework of speech recognition, deep neural network model has brought about tremendous innovations to improve the accuracy of speech recognition. At present, the speech recognition algorithms used by well-known Internet companies (Baidu, HKUST, and SOGOU) adopt deep neural network model. In [12], they applied convolutional neural network (CNN) to speech feature extraction. H.Zen et al. [13] proposed a speech synthesis model based on multilayer perceptron for speech recognition. In [14], they used the LSTM method to extract speech features, which greatly improves the efficiency of features. In 2012, G. E. Hinton replaced the Gaussian mixture model(GMM) in the traditional model with DBN and the results showed that the error rate dropped to 20.7% on the TIMIT core test set, which was significantly improved [15].

Recently, the speech recognition system of the feedforward sequential memory network proposed by Google, which uses a large number of convolutional layers to directly model the entire sentence speech signal and better express the long-term relevance of speech. Baidu applied deep convolutional neural network to speech recognition research by using visual geometry group network in combination with deep convolutional neural networks, the recognition error rate has dropped greatly.

2.3. Natural Language Processing

Natural language processing is also the application of deep learning. K. Cho *et al.* [16] proposed a vector constant length representation model based on recurrent neural network (RNN) for machine translation. Artificial neural networks have been attracted more attention in natural language processing. Rousseau *et al.* [17] used similar models in statistical machine translation tasks and evaluated them with the bilingual evaluation understudy rating mechanism. Karlen [18] have adopted embedding and multi-layered one-dimensional convolutional structures for typical natural language processing issues such as semantic role labeling. Deoras *et al.* [19] further studied the neural network model. and found that the performance can be improved by adding multiple recursive layers. Vincent *et al.* [20] proposed to use embedding methods to map words into a vector representation space, and then use nonlinear neural networks to represent the language model. Bahdanau *et al.* [21] put forward a RNN search model for Machine Translation.

Deep learning techniques have been widely used in various tasks in the field of natural language processing, including part-of-speech tagging [22], dependency grammar analysis [23], naming body recognition [24], semantic role tagging [25], and Twitter sentiment analysis [26]. Chinese microblog sentiment analysis [27], machine translation [28], Question Answering [29], Dialogue System [30], etc.

3. Common Models of Deep Learning

3.1. Autoencoder

Autoencoder is an unsupervised learning algorithm that applies backpropagation, setting the target values to be equal to the inputs. In 1986, Rumelhart et al. [31] proposed the concept of autoencoder (AE), and used it for high-dimensional complex data processing. Autoencoder can extract the latent feature by reconstructing the input data to form the output data. The basic structure of autoencoder can be viewed as three-layer neural network: input layer *x*, hidden layer *h*, and output layer *y*, where the output layer and input layer has the same scale, and **Fig. 1** is the schematic diagram of autoencoder.

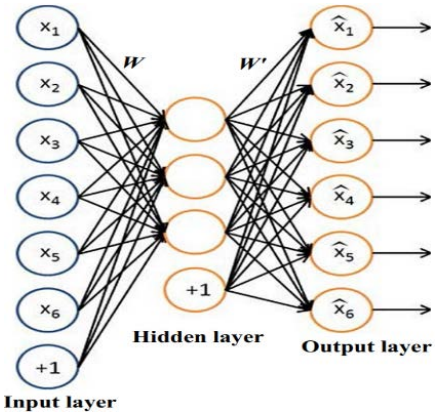

Fig. 1. Schematic diagram of autoencoder model

Autoencoders have three layers, the input layer and the output layer have the same neurons, and more than the middle layer. The output layer reconstructs the input data through training the network, then the similarity of the input data and output data as same as possible. The training error represents the similarity. Assuming that the input signal is *x*, after the input layer reaches the hidden layer, the signal changes to *z*, which can be expressed by the following formula:

$$z = g(Wx + b) \tag{2}$$

Where $g(\cdot)$ is a nonlinear function, commonly used are sigmoid function and the rectified linear unit function, they are also called active function. W is the link weight of the input layer to the hidden layer, and b is the bias of the hidden layer. The signal y is decoded by the

decoding layer and output to the output layer, and the signal becomes *z*, as follows:

$$y = f(W'z + b') \quad (2)$$

Where *f(·)* is a nonlinear function, such as sigmoid function, W' is the link weight of the hidden layer to the output layer, and b' is the bias of the output layer. y is treated as a prediction of *x*. In general, the weight matrix W' is limited to the transpose of the weight matrix *W*: $W' = W^T$. According to the different forms of data, the reconstruction error is usually defined as mean square error or cross entropy.

Some researchers introduced the idea of sparse representation, proposed a sparse Autoencoder, in which the average value of the output signal by l_2 penalty or encouragement with a small mean value which approximated by a Gaussian distribution. In [32], a coarse-to-fine Autoencoder was trained to complete the task of locating key points on the face. In order to enhance the generalization of Autoencoder, Mikolov *et al*. [33] proposed denoising Autoencoder that added artificial noise to the training samples to allow the network to reconstruct the original clean input from the noisy signals. In [34], the deep Autoencoder is used to obtain a compact image high-level description and image retrieval. Many researchers have successfully applied the deep Autoencoder to the image feature representation. Autoencoders can reconstruct the input data through training the network, tuning the parameters, and cascading to form a deep neural network.

3.2. Restricted Boltzmann Machine

Hinton and Sejnowski [35] proposed Boltzmann Machine (BM) in 1986. Boltzmann Machine is a random neural network belonging to the type of feedback neural network. Boltzmann Machine consists of some visible units (visible variables, i.e. data samples) and some hidden units (hidden variables), each visible unit is connected to all the hidden units, the visible variables and hidden variables are binary variables, the state is 0 or 1, 0 represents the neuron is in suppressed state, and 1 represents the neuron is in active state. Sejnowski et al. [36] further proposed restricted Boltzmann machine (RBM). **Fig. 2** is the schematic diagram of RBM, it includes the visible layer V and the hidden layer H. Input the training data to the visible layer, then the hidden layer detects the features of input data, the neurons are disconnected in the same layer but fully connected between two layers. The equation (3) represents the joint probability distribution of two layers.

$$P(v,h) = \frac{1}{Z} e^{-(a^T v + b^T h + h^T w v)} \quad (3)$$

Where *Z* represents the normalized function, *a* and *b* denote the biases of the visible layer and the hidden layer respectively, *W* denotes the connection weights of two layers, the equation (4) is the goal of optimization.

$$\arg max \sum_{v \in V} \log P(v,h) \quad (4)$$

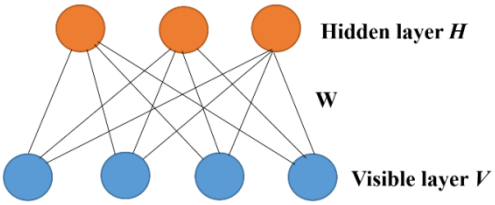

Fig. 2. Restricted Boltzmann machine

The training of Restricted Boltzmann machine is faster than Autoencoder. In [37], proposed a more efficient optimization algorithm based on the stochastic gradient descent method. The traditional training method of RBM requires a large number of sampling steps, which makes the training efficiency of RBM still not high. The contrastive divergence proposed by Hinton solved this problem.

Taylor *et al*. [38] construct a deep deconvolution network by concatenating multiple convolutional sparse Autoencoder and maximum pooling layers to learn the hierarchical structure features from the bottom layer to the top layer directly from the global image. Some researchers have put forward many expansion models based on restricted Boltzmann machine. In [39], proposed that discriminative learning be incorporated into the RBM's generative learning algorithm, so that it can be better applied to discriminative tasks such as classification. In [40], RBM model is directly cascaded into a multi-layer structure, proposed deep Boltzmann machine. Hinton *et al*. [41] proposed a deep sparse Autoencoder model for learning the latent features of image pixel blocks. Restricted Boltzmann machine can be cascaded to form a deep neural network, and use layer-by-layer training method for optimization. Hinton *et al*. [42] extended deep belief network with convolution operations so that the model can learn potential feature representations directly from the original 2D image. Except for the RBM-based deep structure, there are other hierarchical generation models. In [43], a multi-layered directed sigmoid belief network was cascaded with RBM to construct a deep belief network. In [44], by introducing a Gaussian kernel, Restricted Boltzmann machine supports continuous variables as input signals. Restricted Boltzmann machine can be extended to solve more complex problems by modifying the structure of RBM or probability distribution. In these models, usually defined a more complex energy function, and reduced the efficiency of learning and inference.

3.3. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) have convolutional layers, pooling layers, and fully-connected layers except for the input layer and the output layer. CNNs can reduce the complexity and parameters through sharing weights to promoted the generalization ability of neural network, reduce neurons through pooling operation to be more robust. In recent years,

CNNs [45] have been applied in processing multi-dimensional data for image understanding. **Fig. 3** shows the schematic diagram of CNN.

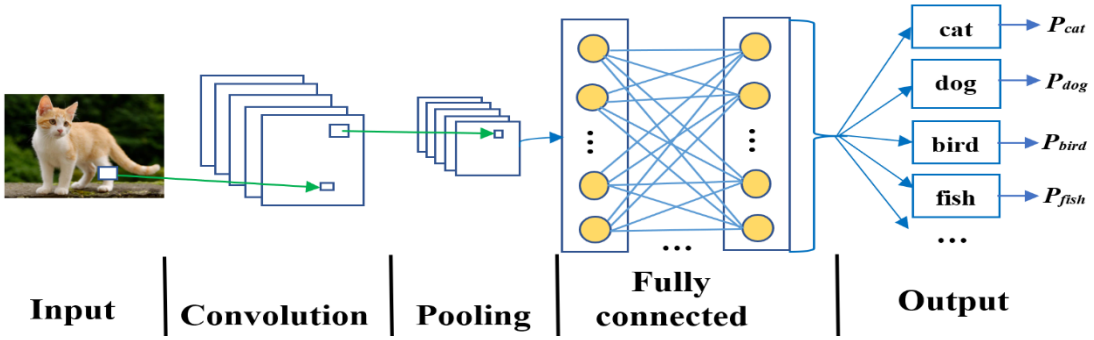

Fig. 3. Convolutional neural network

When inputting a multi-dimensional image to the CNN, the image is directly input to the network, thereby avoiding the complex feature extraction and data reconstruction process in the traditional image processing algorithms. In convolutional neural network, it is necessary to learn a set of two-dimensional filtering templates and convolution operation with the feature map x to obtain two-dimensional feature maps. In the convolutional layer, the feature map of the previous layer is convoluted with convolution kernel, and the output of the convolution result after the activation function forms the neurons of the next layer of the feature map, thereby forming the next layer corresponding to a certain feature map of the features. Each convolutional layer contains three operations: convolution, nonlinear activation function, and maximum pooling. Convolution using different convolution kernels can extract different features of the previous layer of feature maps.

The subsampling layer (pooling layer) usually follows the convolutional layer and down-samples the feature map according to certain subsampling rules. There are two main functions of the down-sampling layer: a) dimensionality reduction of the feature map; b) maintaining the scale invariant characteristics of the feature to a certain extent. In this operation, the feature map is first divided into a number of spatial regions using a uniform grid. These regions may have overlapping portions, and then the average or maximum value of each image region is taken as the output. Existing research work proved that the performance of the maximum pooling operation is better than that of average pooling in image feature extraction.

After multiple convolutional and subsampling layers are alternately transmitted, the convolutional neural network relies on a fully connected network to classify the extracted features to obtain an input-based probability distribution. The training goal of the convolutional neural network is to minimize the loss function of the network. The common loss function has mean squared error (MSE) function, negative log likelihood (NLL) function, etc.

Researchers have proposed many some simple and effective techniques, such as momentum method, weight decay, and data enhancement. Hinton et al. [46] further extended this idea. In the training of full connection layer, each subset is set to 0 at each iteration, which selected randomly from the connection weights of the network, so that each network update different network structures, and this further enhances the generalization of the model. Due to the large number of parameters of the convolutional neural network, it is easy to overfit and affect the final test performance. Glorot *et al.* [47] proposed dropout optimization technique to prevent overfitting by randomly ignoring half of the feature points in each training iteration. However, Dropout's performance improvement for convolutional neural networks is not obvious. The main reason is that convolutional neural networks greatly reduce the number of training parameters compared to fully connected networks due to the weight sharing characteristics of convolution kernels avoid more serious over-fitting. Therefore, the dropout method acting on the fully connected layer is not ideal for the over-fitting effect of the convolutional neural network as a whole.

3.4. Recurrent Neural Networks

The neurons of a fully connected network or a CNN are disconnected in the same layer but fully connected in the different layers, each layer processed signals independent and propagated to the next layer, this type of network cannot work well for the sequential data. RNNs can produced an output by comprised the new input with the latent vector.

RNNs have three layers: the input layer, the hidden layer, the output layer. In theory, RNNs can work well for the sequential data, and complexity of the network is simple, because the current data is only dependent on the previous data. **Fig. 4** shows the schematic diagram of RNN. The neurons are connected in the hidden layer, and RNNs can remain the former information.

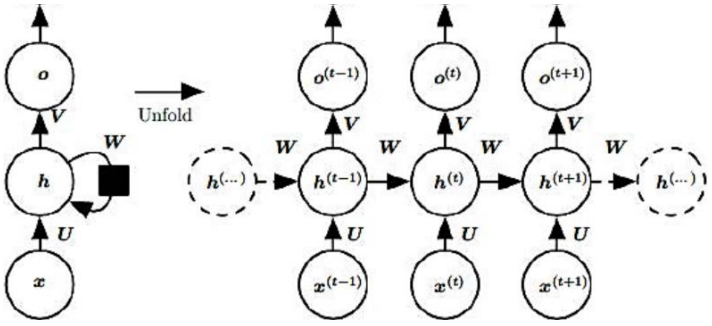

Fig. 4. RNN expands over time

Where *o* and *x* represent the output information and input information respectively, *h* denotes the hidden unit, *W, U, V* represent weights. *t* denotes time, the input at time *t* and the previous state at time *t* are both determine the output of hidden unit at time *t*.

According to the different domains, RNNs have different variants, LSTM(long short-term

memory network) is an example. Bidirectional RNNs, Echo State Networks and so on.

(1) Long Short-Term Memory models(LSTM) [48]. LSTM network is a variant of RNN. LSTM can work well for the time-sequential data. RNNs can only have short-term memory due to the disappearance of gradient. RNN is designed to process the entire time series information, the most deeply remembered signal is the last input signal, but the intensity of the signal affected by the previous signal is getting lower and lower. LSTM can avoid the disappearance of gradient at some extent by controlling the gate through the long memory and short memory. LSTM is different from RNN, because LSTM can determine which information are useful through the cell, the cell includes forget gate except for input gate and output gate. Input a message to LSTM, then determine the information to retain or forget according to whether or not match to the certification of algorithm.

As shown in **Fig. 5**, in the network structure of the LSTM, the input of the previous layer acts on the output through more paths, and the introduction of the gate makes the network have a focusing effect. LSTM generally included the input gate, forget gate, and output gate. The input gate is to supplement the latest input from the current input after the state of the "forgotten" part. The output gate will be based on the latest state C^t, the previous moment output and the current input x^t determine the output h^t at this moment. The forget gate is to make the recurrent neural network "forget" information that was not used before. LSTM can more naturally remember the input long before a long time. The storage unit Cell is a special unit that acts like an accumulator or a "gated leaky neuron": this unit has a direct connection from the previous state to the next state, so it can replicate its current state and accumulate all external signals, and due to the presence of the forget gate, the LSTM can learn to decide when to clear the contents of the memory unit.

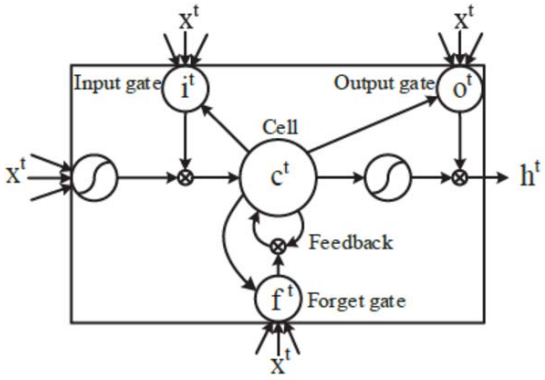

Fig. 5. Schematic diagram of the LSTM cell

(2) Bidirectional RNNs. Bidirectional RNNs [49] are relatively simple RNNs that are composed of two RNNs superimposed on top of each other. The state of two hidden layers determine the output of BRNNs. The structure as shown in **Fig. 6** In the RNN, the current time t usually takes into account the information of the previous moment without

considering the following information. Bidirectional RNNs overcome this shortcoming and introduces the following considerations, i.e., both the previous state and the following state determine the current output.

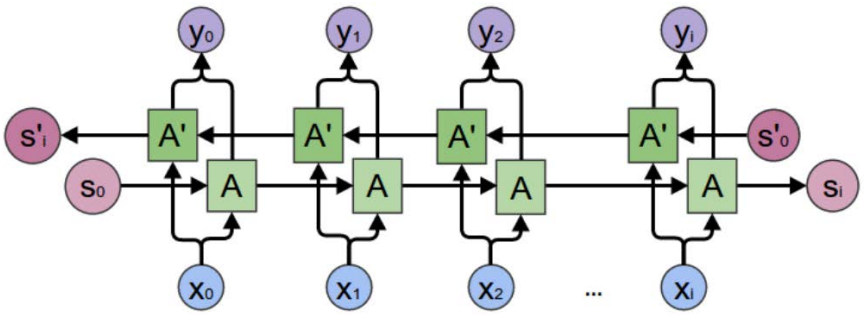

Fig. 6. Bidirectional RNN structure

(3) Echo State Networks(ESNs). ESNs are also a type of RNNs, but they are very different from traditional RNNs. Its core structure makes a reserve pool that is generated at any time and has constant control. The reserve pool is a large-scale, randomly generated, sparsely connected loop structure, and its reserve pool to output layer weight matrix is the only part that needs to be adjusted. The core of algorithm is using a recurrent network of large-scale random connections to replace the hidden layers, thus simplifying the process of training.

3.6. Deep Neural Network

Deep neural networks(DNN) include the input and output layer, multiple hidden layers. DNNs can deal with linear or non-linear problems by computing the probability of each output layer by layer through appropriate activation function. DNNs are usually applied in image understanding and speech recognition, and so on. DNNs essentially are full-connected neural networks. Deep neural network is sometimes called multi-layer perceptron(MLP). The input feature vectors transformed by the hidden layer, then reach to the output layer, finally get the classification result. It was a linear classification model of two categories, mainly used for linear classification and its classification ability was limited. The early discrete transfer functions have some shackles for multiplayer perceptron, so we can use some continuous functions to avoid this problem, e.g. tanh function or sigmoid function. DNNs can be constructed through adding the number of neurons and hidden layers.

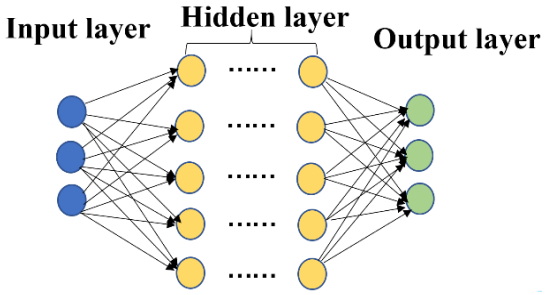

Fig. 7. Deep neural network structure

Fig. 7 shows the structure of a fully connected deep neural network. Deep neural network can be divided into three layers, input layer, hidden layer and output layer. The first layer is the input layer, the last layer is the output layer, and the middle layer is the hidden layer. In the process of deep neural network training: firstly, according to requirements initialization is performed, and set the structure of DNN; then forward is performed, and the layer is transmitted between the layers to obtain an error; then back propagation is performed. According to the principle of minimization of error, the stochastic gradient descent method is used to deduct each parameter, determine the direction of descent, and update each parameter. DNNs have many variants, and have applied in different fields successfully, because the data sets are usually different, the performance of different DNNs usually can't be compared.

3.5. Deep Belief Network

Deep Belief Network (DBN) was proposed by Geoffrey Hinton in 2006 [50]. It is a generation model. we can make the whole neural network generate training data with a high probability by training the weights between its neurons. Multiple Restricted Boltzmann machine can be stacked to construct Deep Belief Network, the visible layer of Restricted Boltzmann machine can be used as the hidden layer of previous Restricted Boltzmann machine, the training of Restricted Boltzmann machine is simple, because the output of previous Restricted Boltzmann machine can be used for training the next Restricted Boltzmann machine. Its structure is shown in **Fig. 8**. It is a generative model composed of multiple layers of nonlinear variable connections. In deep belief network, the part close to the visible layer is multiple Bayesian belief network, and the part farthest from the visible layer is an RBM. At the same time, through this training method, DBN can also obtain deep feature representation from unlabeled data, identify features, categorize data, and generate data [51].

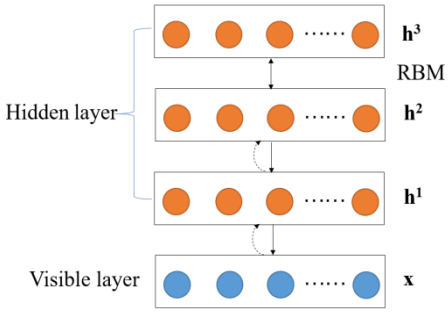

Fig. 8. Deep belief network

DBN consists of multiple layers of neurons, which are subdivided into visible neurons and hidden neurons. The visible element is used to accept input, and the hidden elements are used to extract features. Therefore, hidden elements also have individual names, called feature detectors. The top two layers of DBN are undirected connection and this connection constitutes associative memory. DBN is constructed by RBM[52], one dimension of the data vector is represented by the each neuron of the bottom layer, the bottom layer denotes the data vectors, the layers of DBN are connected layer by layer. The process of training DBN is done layer by layer. In each layer, the data vector is used to infer the hidden layer, and this hidden layer is treated as the data vector of the next layer (higher layer).

3.6. Generative Adversarial Network

Goodfellow *et al*. [53] proposed generative adversarial network in 2014. Its idea comes from the two-person zero-sum game. Different from the previous generation model, it guides the training of the generated model through a discriminant model. The network consists of two parts: generator *G* and discriminator *D*. The discriminator is equivalent to a two-classifier for judging whether the input data is from real data or generated data. The generator is used to capture the distribution of training data. The discriminator is equivalent to a two-classifier for judging whether the input data is from real data or generated data. The result of the network output represents the probability that the input data comes from real data.

The generator can generate an example as similar as the sample in training dataset through a synthetic sample by transforming random noise.

The discriminator can discriminate between data samples and synthetic samples, we can train the discriminator to predicted accurately, and train the generator to produce synthetic samples, which can make the discriminator believe them as real data samples. Its structure is shown in **Fig. 9**.

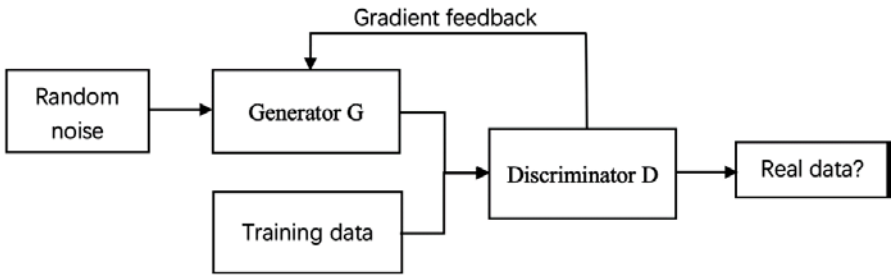

Fig. 9. Generative adversarial network

We use the framework of probability to describe GANs for better understanding, we defined two different marginal distributions, one is $q(\mathrm{x})$, which represents data marginal distribution, the other is $p(\mathrm{x}) = \int p(\mathrm{z})p(\mathrm{x}|\mathrm{z})d\mathrm{z}$, which denotes model marginal distribution.

Equation (5) defines the function of generator G and discriminator D.

$$\min_G \max_D V(D,G) = E_{q(\mathrm{x})}\left[\log\left(D(\mathrm{x})\right)\right] + E_{p(\mathrm{z})}\left[\log\left(1 - D(G(\mathrm{z}))\right)\right]$$

$$= \int q(\mathrm{x}) \log\left(D(\mathrm{x})\right) dx$$

Where x represents the real sample, z represents the noise input to the generator G, G(z) represents the sample generated by generator G, and D(·) represents the probability that the discriminator D determines whether the sample is true.

In the training process, the goal of generator G is to generate a real sample as much as possible to deceive the discriminator D. The goal of discriminator D is to separate the G-generated sample from the real sample. Thus, the generator G and discriminator D constitute a dynamic " adversarial game."

We can minimize $D(\mathrm{x}) = q(\mathrm{x})/(q(\mathrm{x}) + p(\mathrm{x}))$, it equivalent to the divergence between data marginal distribution and model marginal distribution are also minimized, as training continues, the generator G generates samples more and more similar with the real samples.

3.7. Multi-model Fusion Neural Network

Multi-model fusion is to train multiple models and then integrate the models according to certain methods. With the development of deep learning, there will be more and more Researches have shown that a single deep learning model can't always bring the best results. However, the method of increasing the depth of the model often has certain limitations, such as gradient disappearance, computational complexity, and model parallelism. Model fusion based deep learning may bring better results. Excellent neural network models appearing, such as the recent generation of spiking neural network, deep residual network, echo state network (ESN) and attention mechanism-based neural network, etc. Karpathy *et al.* [54] combined CNN with RNN for automatic generation of image descriptions, enabling the combined model to generate textual descriptions based on the features of the image or to

generate corresponding image based on the text. According to the relationship of individual learners, they can be divided into two categories: strong dependencies between individual learners, serialization methods that must be generated serially, representing Boosting methods; no strong dependencies between individual learners, parallelization that can be generated simultaneously Method, the representative is Bagging and random forest.

4. Optimization Methods

For the deep learning model, there are two ways to parallelize deep neural networks: model parallelism and data parallelism. This section mainly introduces the commonly used optimization methods.

4.1. Data Parallelism

Data parallelism refers to the segmentation of training data, and multiple models are used to train multiple pieces of data simultaneously. Data parallelized distributed training stores a backup of the model on each worker node, processing different parts of the data set on each machine. The data parallelization training method needs to combine the results of each working node and synchronize the model parameters between nodes. The basic architecture of data parallelism is shown in **Fig. 10**. The parameter server is responsible for updating the latest model parameter $W' = W - \eta \cdot \Delta W$.

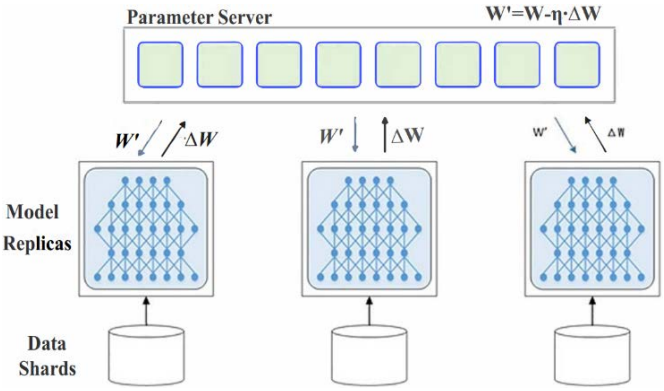

Fig. 10. The basic architecture of data parallelism

Data parallelism is divided into synchronous mode and asynchronous mode. In the synchronous mode, one batch of training data can be trained at the same time by all training programs, after synchronization, and then exchange parameters at the same time. After the parameter exchange is completed, all the training programs have a common new model as a starting point, and then the next batch is trained. Synchronous update is a process of data parallel distributed gradient descent. The advantage is that the convergence of the model will

be relatively stable, because this mode uses a large batch. The larger the batch, the closer the effect of the batch gradient will be to the overall gradient. However, the disadvantage of synchronous mode is that the information transmission overhead is large and has a short board effect.

In the asynchronous mode, the training program completes a batch of training data and immediately exchanges parameters with the parameter server, regardless of the state of other training programs. The latest results of a training program in asynchronous mode are not immediately reflected in other training programs until they perform the next parameter exchange. The advantage of data parallelism is that it is easier to implement and easier to extend; only *W* communication is required, and the data in the model does not need to communicate. The disadvantage is that when the model is large, the GPU memory cannot meet the storage requirements and cannot complete the calculation.

4.2. Model Parallelism

Model parallelism is also a good method for large models that can't be accommodated by stand-alone memory. **Fig. 11** shows the basic framework for model parallelism. Different from data parallelism, Model parallelism can split the model into several shards, which are held by several training units and work together to complete the training. The communication overhead occurs when the input of one neuron comes from the output of a neuron on another training unit. In most cases, the communication overhead and synchronization consumption brought by model parallelism exceeds the data parallelism, so its acceleration is less than data parallelism.

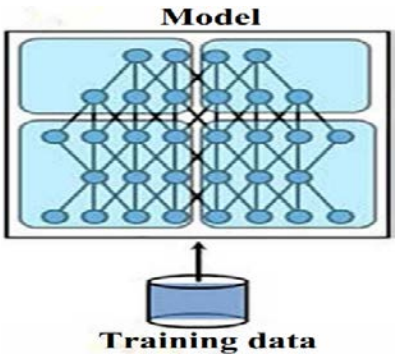

Fig. 11. The basic architecture of the model parallelism

Data parallelism and model parallelism cannot be extended indefinitely. When there are too many data parallel training programs, the learning rate has to be reduced to ensure the smoothness of the training process. When there are too many slices in the parallel mode, the exchange amount of the neuron output values will increase sharply and the efficiency will drop drastically. Therefore, simultaneous model parallelism and data parallelism are also

common solutions.

5. Common Frameworks and Platforms of Deep Learning

5.1. Common Frameworks

Both industry and academia have introduced related open source frameworks that use similar methods for parallel learning of deep learning models. High-performance multi-threaded libraries are used on the CPU to train the network, using data parallelism or model parallelism, and the DNN library is supported on the GPU to speed up the network. In the implementation of distributed parallel deep learning models, data parallelism or model parallelism is used. Commonly used software tools are as follows:

(1) Theano. Theano born in Montreal in 2008, has spawned a number of deep learning Python packages, most notably Blocks. Theano is a Python library for defining, optimizing, and computing mathematical expressions for efficient computation of multidimensional arrays. Its advantages are: integrated NumPy; use GPU to accelerate calculations; efficient symbol differentiation; speed and stability optimization; dynamically generate C code; extensive unit testing and self-verification; Its disadvantages are: the transfer limit of bad parameters in scan, the immutable mechanism causes the function compile to take too long; Theano lacks a flexible polymorphic mechanism when defining function; a difficult debugging method.

(2) Torch. The core computational unit is well optimized using C, and a common model is built using Lua. The core features are: a powerful n-dimensional array; many implementations of indexing, slicing, transposing routines; c-interfaces via Lua; linear algebra routines; neural networks, and energy-based models; numerical optimization routines; efficient GPU support; embeddable, portable to IOS, Android and FPGA backend.

(3) MXNet. MXNet combines the advantages of imperative and declarative programming to optimize the system and facilitate debugging. Resource and computational scheduling, memory allocation resource management, data representation, computational optimization, etc. are all worth learning. Native support for distributed training. MXNet provides imperative tensor calculations to bridge the main language and symbolic expressions. On the other hand, ND Array can seamlessly interface with symbolic expressions. MXNet provides a distributed key-value store for data exchange. It mainly has two functions, push: push the key-value pair from a device into the store, pull: pull the value on a key from the store.

(4) CNTK. CNTK is a unified deep learning tool that describes a neural network as a series of computational steps on a directed graph. In this directed graph, leaf nodes represent input layers or network parameters, and other nodes are represented as matrix operations on the input layer. It is easy to implement and combine today's popular models on CNTK, such as feedforward neural networks DNNs, convolutional neural networks (CNNs), and recurrent neural networks (RNNs/LSTMs). Gradient can be automatically calculated when

implementing stochastic gradient descent learning, and parallel computing can be implemented through multiple GPUs or servers. The biggest advantage of CNTK is that it can parallel multiple GPUs or servers. Another advantage of CNTK is support for Microsoft Windows. But this open source tool is written in C++.

(5) Caffe. Caffe is a learning framework for deep convolutional neural networks. It is convenient to train and test CNN models using Caffe. Caffe is a pure C++/CUDA architecture that supports command line, Python, and MATLAB interfaces; it can be seamlessly switched between CPU and GPU. Caffe is very suitable for rapid development and engineering applications. Caffe officially provides a large number of examples, according to the example, Caffe only asks to write Proto txt, its training process, gradient descent algorithm is packaged, understand the syntax of Proto txt, basically can construct their own neural network.

(6) TensorFlow. TensorFlow is Google's second-generation artificial intelligence learning system based on DistBelief. Tensor means an N-dimensional array, and Flow means a calculation based on a data flow graph. TensorFlow is a process in which tensors flow from one end of the image to the other. TensorFlow is a system that transmits complex data structures to an artificial intelligence neural network for analysis and processing.

Table 1 shows several commonly used open resource frameworks. It can be seen that there are many open resource frameworks based on deep learning, and there are also many corresponding programming languages. **Table 2** shows performance comparison of common open resource frameworks in each dimension. In the future, we believe that more efficient programming languages or platforms may also emerge.

Table 1. Comparison of common frameworks

Framework	Support hardware	Operating language	Parallel mode
Theano	CPU, GPU	Python，C	Data parallelism
Torch	CPU, GPU, PGA	Lua，C++	Data parallelism
MXNet	CPU, GPU	C++，Python	Data parallelism, Model parallelism
CNTK	CPU, GPU	C++，Python	Data parallelism
Caffe	CPU, GPU	C++	Data parallelism
TensorFlow	CPU, GPU	C++，Python	Data parallelism, Model parallelism

Table 2. Performance comparison of common frameworks

Framework	Model design	Interface	Deploy	Performance	Architecture design	Overall rating
TensorFlow	80	80	90	90	100	88
Caffe	60	60	90	80	70	72
Torch	90	70	60	70	90	76
MXNet	70	100	80	80	90	84
CNTK	50	50	70	100	60	66
Theano	80	70	40	50	50	58

5.2. Industry Platforms

(1) Mariana. It’s Tencent's deep learning platform, has evolved into three frameworks, including: DNN GPU data parallel framework, CNN GPU model parallel and data parallel framework, and DNN CPU cluster model parallel and data parallel framework. Through data parallelism and model parallelism, Mariana solves the problem of deep learning time-consuming and becomes an effective help for deep learning research. Mariana effectively supports large models through model parallelism. Mariana has done a lot of work to enhance ease of use, simplify deep learning experiments, and greatly save algorithm development time. Mariana's DNN GPU data parallel framework, for WeChat speech recognition applications, can complete DNN model training of billions of high-dimensional training samples in a few days, while the model error rate is reduced by more than 10%. Mariana's CNN model parallel and data parallel framework, for ImageNet image classification issues, achieved 87% accuracy in the ImageNet 2012 dataset. At present, Tencent deep learning platform Mariana has supported the voice input method of WeChat voice recognition, voice open platform, long press voice message to text and other products, and began to apply in WeChat image recognition. In addition, in the fields of advertising recommendation and personalized recommendation, it is also actively exploring and experimenting.

(2) XDL. It’s Ali Mama's deep learning framework, based on Alibaba's massive scale of business scenario practice, a new generation of distributed deep learning framework designed and developed. XDL has created four key paradigms (new data paradigm, new model paradigm, new capability paradigm, new architectural paradigm) that make it truly industrial-strength. XDL achieves an approximate linear computational speedup of thousands of nodes concurrently and can accommodate hundreds of billions of sparse parameters training, online streaming training, and the ability of a full-process asynchronous pipeline to maximize hardware saturation. At present, XDL has been widely used in many business scenarios of Ali Mama, creating a revenue of 10 billion scale. At the same time, thanks to XDL's new architectural capabilities, a series of industry-first model algorithms

can be effectively trained on the XDL platform, including any deep learning library-based search model based on tree structure, user behavior images.

(3) Paddle. Paddle is an open source deep learning framework provided by Baidu. It is a new generation deep learning framework based on “deep learning programming language”. It has greatly improved the ability of the framework to express the model and can describe any potential possibility and friendly to large-scale computing: After a lot of large-scale computing business in Baidu, Paddle is excellent in distributed computing. EDL-based technology can save a lot of computing resources and can also support large-scale sparse model training. It provides visual deep learning: visual DL can help developers to easily observe the overall trend of training, data sample quality and intermediate results, parameter distribution and trend, and the structure of the model helping developers to complete the programming process more conveniently.

6. Acceleration Technology of Deep Learning

At present, deep learning acceleration technology is accelerated by a dedicated accelerator or FPGA, which has the advantages of low power consumption and high speed. The main deep learning acceleration techniques are as follows:

6.1. Multi-core CPU Acceleration Technology

By designing a dedicated CPU processor, it is a processor designed for an algorithm or algorithm family. It has the characteristics of low power consumption, high speed and high development cost. In order to design a dedicated processor, we should make analysis of the algorithm, get the features of it, then based on the features of algorithm design a dedicated circuit.

However, due to the large amount of computation and the high degree of parallelism in deep learning, the CPU is increasingly difficult to adapt to the computing needs and can only utilize the multi-core CPU. Researchers optimize the use of registers based on the window movement features of the convolution and pooling algorithm, and finally accelerate the convolutional neural network algorithm by accelerating convolution and pooling calculations. Researchers design hardware neurons for neuron structures, map connections between neurons to latches or other memory, and finally accelerate neural network algorithms by mapping neurons on neural networks to hardware neurons. It designs different storage units according to the features of algorithm memory allocation. The types of data are different, the storage units are different. Finally, the utilization of the computing unit is improved by pipeline, which is lower than the general processor.

6.2. GPU Acceleration Technology

GPU (Graphics Processing Unit) is a concept relative to the CPU, which was proposed with the rapid growth of multimedia computing and eventually became a processor independent

of the traditional CPU. Today, GPUs can provide tens of times or even hundreds of times the performance of CPUs in terms of computations such as floating-point operations and parallel computing. Compared with CPU, GPU has the following advantages: powerful computing power, GPU has more powerful computing power, in the same period, the theoretical peak capacity of GPU floating-point calculation is one order of magnitude higher than CPU; the price is relatively cheap, for individuals For small organization users, to obtain high-performance computing, if the traditional CPU cluster-based computer system is used, it is expensive, and the requirements for site and power supply are also high. The use of GPUs to train deep neural networks can achieve highly efficient parallel computing capabilities. In massive amounts of training data, the time spent is drastically reduced and fewer servers are occupied.

6.3. ASIC Acceleration Technology

ASIC (Application Specific Integrated Circuit), which is a more efficient hardware acceleration method than GPU, on the one hand, its high degree of customization increases its computing power and limits its mobility. On the other hand, ASICs are expensive and difficult for individuals and small organizations. Combining these two aspects, researchers of deep learning currently rarely consider using ASICs for computer acceleration.

ASIC is an integrated circuit designed for a specific purpose. Refers to integrated circuits designed and manufactured to meet the needs of specific users and the needs of specific electronic systems. ASIC is characterized by the needs of specific users, ASIC is divided into full custom and semi-custom. The highlight is dedicated, tailored so that the execution speed is faster, faster than FPGA for the same process FPGA, and can save some unused logic implementation in FPGA, the cost will be lower than FPGA for large-scale production.

The characteristics of ASIC are: for the needs of specific users, ASIC has the advantages of smaller size, lower power consumption, higher reliability, improved performance, enhanced confidentiality and lower cost compared with general-purpose integrated circuits in mass production; ASIC needs Longer development cycles than FPGAs, FPGA solutions have greater flexibility, can shorten development cycles; ASIC's high risk. Because of the re-programmability of FPGAs and the flexibility, systems using programmable devices can easily be upgraded or corrected in the field. Once the ASIC has problems, all the pieces are discarded; the cost of designing and supporting the tools. FPGA solutions also save a lot of money in design and support tools; ASICs are used for large projects, and FPGAs are more suitable for small projects that need to be quickly placed on the market and support remote upgrades.

6.4. FPGA Acceleration Technology

FPGA (field programmable gate array), which is a product of further development based on

programmable devices such as PAL, GAL, CPLD, etc., is one of the fields in the field of application specific integrated circuits (ASIC). FPGA is a semi-custom circuit in the field of ASIC, which not only solves the shortcomings of custom circuits, but also overcomes the original programmable device gates. The highlight is its programmability, which brings great convenience to the design implementation. It also provides a viable solution for reducing design costs, but at a slower rate than an ASIC of the same process. The emergence of semi-custom circuit not only solves the shortcomings of the custom circuit, but also overcomes the shortcomings of the limited number of original programmable device gates. FPGAs have achieved trade-offs between GPUs and ASICs, with a good balance between processing speed and control. On the one hand, FPGA is programmable reconfigurable hardware, so it has more powerful controllability than GPU; on the other hand, it has more design space with increasing gate resources and memory bandwidth. More conveniently, the FPGA also eliminates the tape-out process required in the ASIC solution. One disadvantage of FPGAs is that they require the user to program them in a hardware description language. However, technology companies and research institutes have developed languages that are easier to use, such as the C-to-FPGA compiler developed by Impulse Accelerated Technologies Inc., making the FPGA more user-friendly; Yale's E-Lab developed Lua. Scripting languages, etc., these tools have shortened the user's development time limit to a certain extent, making research easier.

7. Problems and Prospects

Deep learning brings the dawn of the research of artificial intelligence by designing complex multi-layer neural network models, using huge training data, consuming a large amount of computing resources to train them, and finally learning the ability to extract multi-level abstract features of data. Due to the complex process of deep neural network algorithm, the number of iterations and the high computational complexity, there are some challenges and bottlenecks in the parallelization of deep neural networks. Below we propose some directions worth exploring in the future parallel development of deep neural networks.

(1) Algorithm optimization. In the original deep learning algorithm design process, the main focus is on algorithm accuracy without considering parallelism as a key factor. Therefore, current deep learning algorithms cannot fully utilize multi-core computing capabilities. In order to better utilize the potential of multi-core platforms, when designing algorithms, in addition to the training precision of the algorithm, the parallelism of the algorithm should be taken into consideration.

(2) Model compression. The techniques of model compression can be divided into four categories: one is called pruning, and the neural network is mainly connected by nodes of one layer and one layer, and each side has some weight. Pruning means that if we find that the weights on some sides are small, such edges may not matter, and these edges can be

removed. After training the large model, we can see edges who have smaller weights, remove those edges, and then retrain the model on the reserved edges. Another way to compress the model is through weight sharing. Assuming that there are full connections between two adjacent layers, each layer has a thousand nodes, there are one thousand by one thousand between the two layers, that is, one million weights (parameters).

(3) Dual learning. Dual learning attempts to apply the dual attribute of this structure to deep learning. Through this dual process, we obtain feedback information from unlabeled data, know whether our model works well or not, and then train and update according to feedback information. Reverse the model to achieve the goal of learning from unlabeled data.

(4) Broadband limitations and computing power of embedded systems. Deep learning requires a lot of data and uses DDR to transfer between layers. If the weight of the convolution and full connection data comes from DDR, the data transmission is extremely large. In these cases, also use floating point precision. In many cases, the same network is used to process multiple ROIs. While large, high-power electrical machines perform these tasks, the embedded platform sets strict limits. To achieve cost-effectiveness, low power, and minimal scale, embedded solutions use a small amount of data, limiting memory size, usually running at integer precision, which is the opposite of floating point.

(5) Untagged data feature learning. At present, the feature learning of tagged data still dominates, and there are massive unmarked data in the real world. It is obviously unrealistic to add artificial tags to these unmarked data one by one. Therefore, with the development of data sets and storage technologies, more and more attention will be paid to the feature learning of unmarked data and the research on automatic labeling of unmarked data.

(6) Super parameter optimization. Hyper parameters are pre-defined neural network parameters that are pre-defined in the training and learning process. These parameters have a huge impact on the neural network, and the difference is a thousand miles. Hyper parametric tuning is an important task in deep learning. The performance of a network is likely to depend on the nuances of hyper parameters, which is the most in-depth experience in deep learning and neural networks.

(7) The unexplainable nature of deep learning. For deep learning, we know the parameters of the model, the input data, and how to connect these to the network, but we still don't know how it implements this process. Deep learning is like a black box of nerves, it has a very magical function but we don't know how it works. The interpretability of deep learning hinders the abstraction and summarization of this technology and limits the research and development of high-level cognitive intelligence. At the same time, the operation of the neural network is too abstract for humans, and we can't really verify whether its working process is reasonable.

(8) Unsupervised feature learning. A supervised method learns a classification model by training data and applies the classification model to the classification of unknown data (test

data). Therefore, the quality of the classification model is strongly dependent on the quantity and quality of the training data set. A good training data set can obtain a good classification model, so that a better classification result can be obtained for the unknown data. Currently, there are still relatively few trained training data sets available, which may limit the further development of deep learning. Therefore, it is necessary to obtain a larger labeled data set, making it possible to further deepen the deep learning.

8. Conclusion

As an important research direction of machine learning, deep learning brings the dawn of the research of artificial intelligence. In view of the rapid changes of deep learning, we systematically introduced the latest progress of deep learning. Firstly, we introduced several commonly used neural network models of deep learning, analyzed two commonly used parallel training models based on deep learning, compared the advantages and disadvantages of training methods of the two models. Then, we analyzed commonly used deep learning open source frameworks, compared their application features and several industrial research platforms. Finally, we focused on the current research of neural network hardware accelerators. The future development of deep learning technology is still full of different opportunities and challenges, and it is highly promising.

Acknowledgements

This research was partially supported by the National Key Research and Development Plan Key Projects of China under Grant No.2017YFC0405800, the National Natural Science Foundation of China Grant (Nos. 60971088, 60571048, 61432008, and 61375121), the Natural Science Foundation of the Colleges and Universities in Jiangsu Province of China, No.17KJB520022.

We thank the anonymous referees for their helpful comments and suggestions on the initial version of this paper.

References

[1] Marz N, Warren J, “Big Data: Principles and best practices of scalable real time data systems,” Greenwich, USA: Manning Publications Co, 2015.

[2] Gantz J, Reinsel D, “The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east,” *IDC iView: IDC Analyze the future*, pp.1-16, 2012. Article (CrossRef Link).

[3] LeCun Y, Bengio Y, Hinton G, “Deep learning,” *Nature*, vol.521, pp.436-444, 2015. Article (CrossRef Link).

[4] Peng Y, Zhu W, Zhao Y, "Cross-media analysis and reasoning: advances and directions," *Frontiers of Information Technology & Electronic Engineering*, vol.18, no.1, pp.44-57, Feb.2017. Article (CrossRef Link).

[5] Dong C., Loy C. C., He K., Tang X, "Image Super-Resolution Using Deep Convolutional Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vo.38, no.2, pp. 295-307, 2015. Article (CrossRef Link).

[6] Denker J S, Boser B, Lecun Y, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol.1, no.4, pp.541-551, 1989. Article (CrossRef Link).

[7] Ren S., He K., Girshick R., Sun J, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. of the Advances in Neural Information Processing Systems, Montreal, Canada*, pp.91-99, 2015. Article (CrossRef Link).

[8] Wang Z., Liu D., Yang J., Han W., Huang T, "Deep Networks for Image Super-Resolution with Sparse Prior," in *Proc. of the International Conference on Computer Vision, Santiago, Chile*, pp.370-278, 2015. Article (CrossRef Link).

[9] Yu K, Yang J, Gong Y, "Linear spatial pyramid matching using sparse coding for image classification," in *Proc. of 2009 IEEE on Computer Vision and Pattern Recognition(CVPR), Miami, FL, USA*, pp. 1794-1801, 2009. Article (CrossRef Link).

[10] Hinton G E, Salakhutdinov R R, "Reducing the dimensionality of data with neural networks," *Science*, Vol.313(5786), pp.504-507, 2006. Article (CrossRef Link).

[11] Xia R., Pan Y., Lai H., Liu C., Yan S, "Supervised Hashing for Image Retrieval via Image Representation Learning," in *Proc. of the Association for the Advancement of Artificial Intelligence, Québec City, Canada*, pp.2156-2162, 2014. Article (CrossRef Link).

[12] Mao Q, Dong M, Huang Z, "Learning salient features for speech emotion recognition using convolutional neural networks," *IEEE Transactions on Multimedia*, vol.6, no.8, pp.2203- 2213, 2014. Article (CrossRef Link).

[13] ZEN H, SENIOR A, SCHUSTER M, "Statistical parametric speech synthesis using deep neural networks," in *Proc. of 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, NJ: IEEE*, pp.7962-7966, May 2013. Article (CrossRef Link).

[14] Felix W, Jürgen G, Martin W, "Feature enhancement by deep LSTM networks for ASR in reverberant multisource environments," *Computer Speech & Language*, vol.28, no.4, pp.888-902, 2014. Article (CrossRef Link).

[15] MOHAMED A R, DAHL G E, HINTON G, "Acoustic modeling using deep belief networks," *IEEE transactions on audio, speech, and language processing*, vol.20, no.1, pp.14-22, 2012. Article (CrossRef Link).

[16] CHO K, van MERRIENBOER B, GULCEHRE C, "Learning phrase representations using RNN encoder decoder for statistical machine translation," *CORR*, pp.1406-1416, 2014. Article (CrossRef Link).

[17] Rousseau A, Attik M, Schwenk H, "Large pruned or continuous space language models on a GPU for statistical machine translation," in *Proc. of the NAACL-HLT 2012 Workshop: Will We*

Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT, Association for Computational Linguistics, pp.11-19, June 2012. Article (CrossRef Link).

[18] Karlen M, Collobert R, Weston J, "Natural Language Processing (Almost) from Scratch," *Journal of Machine Learning Research*, vol.12, no.1, pp.2493-2537, 2011. Article (CrossRef Link).

[19] Deoras A, Kombrink, SMikolov T, "Empirical evaluation and combination of advanced language modeling techniques," in *Proc. of Conference of the International Speech Communication Association*. pp.605-608, 2011.

[20] Vincent P, Janvin C,Bengio Y, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol.3, no.6, pp.1137-1155,2003.

[21] BAHDANAU D, CHO K, BENGIO Y, "Neural machine translation by jointly learning to align and translate," *CORR*, pp.473-485, 2014. Article (CrossRef Link).

[22] Kumar A, Irsoy O, Ondruska P, "Ask me anything: Dynamic memory networks for natural language processing," in *Proc. of International Conference on Machine Learning. New York, NJ, USA: ACM*, pp.1378-1387, 2016. Article (CrossRef Link).

[23] Weiss D, Alberti C, Collins M, "Structured training for neural network transition-based parsing," *Eprint Arxiv*, vol.6, no.3, pp.125-133, 2015. Article (CrossRef Link).

[24] Lample G, Ballesteros M, Subramanian S, "Neural architectures for named entity recognition," *Eprint Arxiv*, vol.6, no.4, pp.86-93, 2016. Article (CrossRef Link).

[25] He L, Lee K, Lewis M, "Deep semantic role labeling: What works and what's next," in *Proc. of Annual Meeting of the Association for Computational Linguistics. New York, NJ, USA: ACM*, pp.473-483, 2017. Article (CrossRef Link).

[26] Severyn A, Moschitti A, "Twitter sentiment analysis with deep convolutional neural networks," in *Proc. of Research on Development in Information Retrieval. New York, NJ, USA: ACM*, pp.959-962, 2015. Article (CrossRef Link).

[27] He Y X, Sun S T, Niu F F, "A deep learning model enhanced with emotion semantics for microblog sentiment analysis," *Chinese Journal of Computers*, vol.40, no.4, pp.773-790, 2017.

[28] Gehring J, Auli M, Grangier D, "Convolutional sequence to sequence learning," *Eprint Arxiv*, vol.3, no.2, pp.86-93, 2017. Article (CrossRef Link).

[29] Vinyals O, Fortunato M, Jaitly N, "Pointer networks," in *Proc. of Neural Information Processing Systems. Cambridge, USA: MIT Press*, pp.2692-2700, 2015. Article (CrossRef Link).

[30] Zhou X, Dong D, Wu H, "Multi-view response selection for human-computer conversation," in *Proc. of Empirical Methods in Natural Language Processing. New York, NJ, USA: ACM*, pp.372-381, 2016. Article (CrossRef Link).

[31] Rumelhart D, Williams R, Hinton G, "Learning representations by back-propagating errors," *Nature*, vol.323, pp. 533-538, 1986. Article (CrossRef Link).

[32] Bourlard H, Kamp Y, "Auto-association by multi-layer perceptron and singular value decomposition," *Biological Cybernetics*, vol.59, no.4, pp.291-294, 1988. Article (CrossRef Link).

[33] Mikolov T, Le Q V, "Distributed Representations of Sentences and Documents," *Eprint arXiv:1405.4053*, vol.4, pp.1172-1188, 2014.Article (CrossRef Link).

[34] Kim Y, "Convolutional Neural Network for Sentence Classification," *Eprint arXiv:1408.5882*, vol.6, no.3, pp.105-112, 2015. Article (CrossRef Link).

[35] Vincent P, Larochelle H, Bengio Y, "Extracting and composing robust features with denoising autoencoders," in *Proc. of International Conference*, pp.1096-1103, 2008. Article (CrossRef Link).

[36] Sejnowski T J, Hinton G E, "Learning and relearning in Boltzmann machines," *Parallel distributed processing: explorations in the microstructure of cognition, MIT Press*, vol.1, pp.45-76, 1986. Article (CrossRef Link).

[37] Le Q V, Ngiam J, Coates A, "On optimization methods for deep learning," in *Proc. of International Conference on Machine Learning*, pp.265-272, 2011.

[38] Taylor G W, Fergus R, Zeiler M D, "Adaptive deconvolutional networks for mid and high-level feature learning," in *Proc. of IEEE International Conference on Computer Vision*, pp.2018-2025, 2011. Article (CrossRef Link).

[39] Rifai S, Vincent P, Muller X, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Proc. of the 28th International Conference on Machine Learning*, pp.833-840, 2011. Article (CrossRef Link).

[40] Zhang J, Shan S, Kan M, Chen X, "Coarse-to-fine auto-encoder networks for real-time face alignment," in *Proc. of European Conference on Computer Vision. Springer International Publishing*, pp.1-16, 2014. Article (CrossRef Link).

[41] Hinton G E, "Training products of experts by minimizing contrastive divergence," *Neural computation*, vol.14, no.8, pp.1771-1800, 2002. Article (CrossRef Link).

[42] Hinton G E, "A Practical Guide to Training restricted Boltzmann machines," *Momentum*, vol.9, no.1, pp.599-619, 2012. Article (CrossRef Link).

[43] Krizhevsky A, Hinton G E, "Using very deep autoencoders for content-based image retrieva," in *Proc. of ESANN*, pp.262-270, 2011. Article (CrossRef Link).

[44] Zou W Y, Ng A Y, Yu K, "Unsupervised learning of visual invariance with temporal coherence," in *Proc. of Workshop on Deep Learning and Unsupervised Feature Learning*, pp.362-370, 2011. Article (CrossRef Link).

[45] Krizhevsky A, Sutskever I, Hinton G E, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol.60, no.6, pp.84-90, 2017. Article (CrossRef Link).

[46] Hinton G E, Ranzato M, "Modeling pixel means and covariances using factorized third-order Boltzmann machines," in *Proc. of Computer Vision and Pattern Recognition Conference*. pp.2551-2558, 2010. Article (CrossRef Link).

[47]Glorot X, Bengio Y, "Understanding the difficulty of training deep feedforward neural networks," *Journal of Machine Learning Research*, vol.9, pp.249-256, Jan 2010. Article (CrossRef Link).

[48] Courville A, Bergstra J, Bengio Y, "A spike and slab restricted Boltzmann machine," in *Proc. of 14th International Conference on Artificial Intelligence and Statistics*, pp.233-241, 2011. Article (CrossRef Link).

[49] Memisevic R, Hinton G, "Unsupervised learning of image transformations," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp.1-8, 2007. Article (CrossRef Link).

[50] Hinton G E, Osindero S, Teh Y W, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol.18, no.7, pp.1527-1554, 2006. Article (CrossRef Link).

[51] Wang X, Wang Y, "Improving content-based and hybrid music recommendation using deep learning," in *Proc. of 22nd ACM International Conference on Multimedia, Orlando, USA*, pp.627-636, 2014. Article (CrossRef Link).

[52] Bengio Y, "Learning deep architectures for AI," *Foundations and Trends® in Machine Learning*, vol.2, no.1, pp.1-127, 2009. Article (CrossRef Link).

[53] Goodfellow Ian J, Pouget-Abadie Jean, Mirza Mehdi, "Generative Adversarial Networks," *Eprint arXiv:1406.2661*, vol.6, no.3, pp.125-133, 2014. Article (CrossRef Link).

[54] Karpathy A, Li F F, "Deep visual-semantic alignments for generating image descriptions," *IEEE Trans on Pattern Analysis& Machine Intelligence*, vol.39, no.4, pp.656-664, 2017. Article (CrossRef Link).

Ruihui Mu received his M.S. degree in computer technology from Huazhong University of Science and Technology, He is currently a Ph.D. student at Hohai University. He is an associate professor in Henan Xinxiang University. His research interests include deep learning, neural networks and personalized recommender systems.

Xiaoqin Zeng received his B.S. degree in computer software from Nanjing University, M.S. degree in computer applications from Southeast University in China, and his Ph.D. degree in computer science from the Hong Kong Polytechnic University. He is a professor at Hohai University, a doctoral supervisor, and director of the Institute of Intelligent Science and Technology. His research interests focus on machine learning, computing and intelligence, and neural networks.