An Image Classification Method Based on Deep Neural Network with Energy Model

Yang Yang[1, *], Jinbao Duan[1], Haitao Yu[1], Zhipeng Gao[1] and Xuesong Qiu[1]

Abstract: The development of deep learning has revolutionized image recognition technology. How to design faster and more accurate image classification algorithms has become our research interests. In this paper, we propose a new algorithm called stochastic depth networks with deep energy model (SADIE), and the model improves stochastic depth neural network with deep energy model to provide attributes of images and analysis their characteristics. First, the Bernoulli distribution probability is used to select the current layer of the neural network to prevent gradient dispersion during training. Then in the backpropagation process, the energy function is designed to optimize the target loss function of the neural network. We also explored the possibility of using Adam and SGD combination optimization in deep neural networks. Finally, we use training data to train our network based on deep energy model and testing data to verify the performance of the model. The results we finally obtained in this research include the Classified labels of images. The impacts of our obtained results show that our model has high accuracy and performance.

Keywords: Image classification, deep energy model, deep neural network, stochastic depth, deep learning.

1 Introduction

With the development of artificial intelligence, especially deep learning, humans have made rapid progress in image recognition tasks. In 2006, Geoffrey Hinton designed a structure of deep neural networks called deep belief networks (DBNs) [Hinton and Osindero (2006)] that marked the era of artificial intelligence in deep learning. In 2010, convolutional neural networks became the most commonly used network model in image recognition applications. In 2012, Alex Krizhevsky and Hinton et al. published AlexNet [Krizhevsky, Sutskever and Hinton (2012)], which has a deeper network hierarchy and uses ReLU activation functions instead of the traditional Sigmoid functions. Dropout techniques are used to avoid overfitting the model. And the introduction of max-pooling technology and GPU-assisted training, these new technologies have become an important part of the current convolutional neural networks.

[1] State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, 100876, China.

[*] Corresponding Author: Yang Yang. Email: yyang@bupt.edu.cn.

Convolutional neural networks have obtained a large number of applications at present, such as image recognition, image target detection, image segmentation and other fields. Especially in the ImageNet competition after 2012 (this competition is mainly to identify a data set with 1000 categories), basically every contingent team uses the convolutional neural network extensively. At the beginning of the game, the team that used the convolutional structure of the neural network received the absolute leading result of the first place. The error rate of this team was half that of other teams such as the SVM algorithm. Therefore, most image processing tasks today use convolutional neural networks.

However, these methods exist problems of insufficient image recognition accuracy and scalability. Although Lv et al. [Lv, Yu, Tian et al. (2014)] have studied networks using DBN (Deep Belief Network) neural network, the problem of training difficulty still exists in spite of the fact that many works show network depth is very influential [Szegedy, Liu, Jia et al. (2015)]. Besides, many scholars have carried on continuous research on this problem before [Ioffe and Szegedy (2015); Jaderberg, Czarnecki, Osindero et al. (2016)], as we stack more layers, such as over a hundred layers, the optimization of deep neural networks has been approved to be more difficult, the obstacle problem is vanishing gradients [Bengio, Simard and Frasconi (1994); Glorot and Bengio (2010); He, Zhang, Ren et al. (2015)], which hamper the increase of neural layers [Saxe, McClelland and Ganguli (2013); Ioffe and Szegedy (2015); He and Sun (2015)]. Stochastic depth network can successfully address these problems [Huang, Sun, Liu et al. (2016)], it can greatly increase the nets depth and bring substantial performance than former networks. In addition, the image processing tasks based on deep neural networks have always been a difficult problem. Sometimes a complex model is applied to a large data volume task, often taking days or even weeks to train. In response to these problems, the researchers have also developed a variety of optimization algorithms, designed to improve the convergence speed of the model, and reduce the error of the model, common optimization algorithms such as SGD, Adam and so on.

In this paper, we presented a novel architecture that integrating deep energy model into stochastic depth network (SADIE) to build image features. The novel contributions in this paper are as follows: (1) We improved the stochastic depth network by using deep energy model, which can obtain better generative models when training based on the energy model. (2) We applied the Dirichlet distribution related to the dimension of input data to our network when chose whether to skip the layers, since the distribution can improve the performance of the network and make results more accurate.

Since the connection has a certain probability of being randomly ignored in the block of the stochastic depth network, the amount of calculation can be reduced. The authors of stochastic depth networks claim that the model after the above adjustments expects a depth of 3/4 and a 40% increase. This feature was also confirmed in our experiments. This design of introducing random variables effectively overcomes the over-fitting to make the model better generalized. Inactivation of a part of the block actually realizes a hidden model fusion. Since the depth of the model is random during training, the depth of the model is determined at the time of prediction, and the information is filtered as the layer is extracted. When the information reaches the upper level of the network, it is not very informative, and the high-level network is difficult to get effective training. We set that a part of the block is not activated, so that the higher-level block can receive more information from the bottom layer, and more training can be obtained, so the model has better expression ability. We believe that the stochastic depth network can not only guarantee the training time of the nets but also obtain a deeper network structure. In preliminary experiments, our model is surprisingly effective. At the same time, we also explored the possibility of using Adam and SGD combination optimization in deep neural networks, and designed an improved switching mechanism. The Adam algorithm is used to accelerate convergence in the early stage of training, and the SGD algorithm is used to search for the optimal solution slowly in the later stage of training. The experimental results show that the optimized SADIE model can greatly improve the convergence speed without affecting the accuracy.

The rest of the paper is as follows: we describe the background knowledge in the second part. The third part describes our model and methods thoroughly. The fourth part describes the experiment procedure and the analysis. We summarize the experimental results in the last part.

2 Related works

Traditional image classification algorithms mainly include word bag models. Generally, the complete establishment of image recognition models generally includes several stages such as underlying feature extraction, feature coding, spatial constraints, classifier design, and model fusion. With the development of machine learning, classification algorithms such as SVM and KNN are also widely used. However, due to the large dimensionality of the image data and the large amount of data, the effects of these algorithms are very limited.

Deep neural networks can learn more features from input data compared to other machine learning methods and has a strong ability to express data. However, when the hidden layer is very large, the learning and feature extraction of the whole network is very challenging. But deep energy model can transform the input in the feedforward neural network into a new form [Ngiam, Chen, Koh et al. (2011)], which makes the hidden layer improve the performance and greatly enhances the training performance during the training process. At the same time, energy-based probability model can also be transformed into Boltzmann distribution (also called Gibbs distribution) [Kim and Bengio (2016)]. The energy model uses a scalar energy value for each random variable so that the training goal of the energy model is translated into such a way that the energy value of the whole system is minimized. Meanwhile, deep energy model can also be regarded as a probability distribution which is combined feedforward neural network and energy model.

In general neural networks, each layer of network is trained layer by layer, it's based on the reconstruction as the primary goal of learning [Bengio (2009)]. But deep neural network based on energy model can train the nets as a whole and characterize the data features through hidden layers. So, we use the energy model to train network, and the network is trained by optimizing the likelihood function of the input data.

Deep neural networks have made a series of breakthrough in image processing and recognition, especially the outstanding performance of deep convolutional neural nets (CNN) in image classification [Simonyan and Zisserman (2014); Szegedy, Liu, Jia et al. (2015)]. The depth of neural networks for the performance of model is a very important factor [He, Zhang, Ren et al. (2015)]. However, when stacking more layers, vanishing gradients occurs in the process of back propagation which leads to a sharp decrease in accuracy and the training time is greatly increased [Huang, Sun, Liu et al. (2016)]. e.g. a 110-layer ResNet requires several weeks for ImageNet dataset even with multiple GPUs [He, Zhang, Ren et al. (2015)].

We also use the stochastic depth nets [Huang, Sun, Liu et al. (2016)], which can shorten the training time of the network while enabling the expression ability of network to be maintained and can also handle the problem of degradation. We apply the Dirichlet distribution related to the dimension of input data to our network. Because the distribution can generate a probability value based on the dimension of the input vector to choose whether to skip the layers.

The main innovation of this paper is to improve the deep neural network with random depth and introduce the depth energy model. The energy model is different from the loss function of the common neural network. When the network forwards, the deep energy model is used to train the network, so that the overall entropy of the network is minimized, which is beneficial to improve the accuracy of the Deep Neural Network. Experiments show that this approach has a good performance.

3 Deep neural network and deep energy model

3.1 Deep neural network model with stochastic depth

Plain neural networks take a long time to train, so they cannot satisfy our requirements. We adopt stochastic depth network in this paper which can be a good balance between training time and accuracy, we also improve it to better match the input data of our model further. Fig. 1 shows the main network structure.

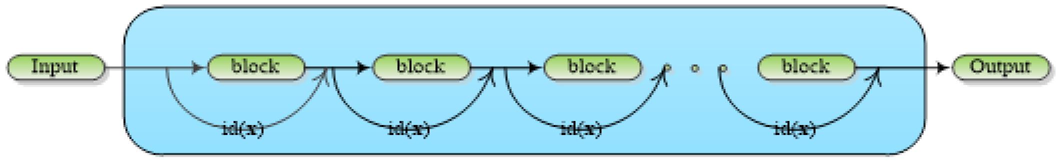


Figure 1: Network structure, $\mathrm{id}(\mathbf{x})$ represents the identity transformation of current layer's input vector

We adopt shortcut connections to every few stacked layers (Fig. 1) to control whether to skip an entire layer block of the network during training. So, the whole network is similar to residual mapping [He, Zhang, Ren et al. (2015)], and the equation is:

$$h_l(\boldsymbol{x}) = F(\boldsymbol{x}, \{\boldsymbol{W}_i\}) + \boldsymbol{x} \tag{3.1}$$

Where $\boldsymbol{x}$ represents the input vector of current layer, $h_l(\boldsymbol{x})$ represents the output vector of l layer. Where $F(\boldsymbol{x}, \{\boldsymbol{W}_i\})$ denotes the residual mapping. The residual mapping is composed of an identity mapping and a shortcut connection, which makes the training of deeper neural networks possible, solves the problem of degraded performance of deep convolutional neural networks under extremely deep conditions, and performs excellent classification performance. The above equation changes to $h_l(\boldsymbol{x}) = \boldsymbol{W}_i\boldsymbol{x} + \boldsymbol{x}$ when the residual mapping has only one layer. To control current layer l skip to next layer block randomly, we set the probability value P_l in the residual mapping to denote the probability that the current layer can propagate directly to the next layer block. The probability that the next layer is ignored is $1 - P_l$.

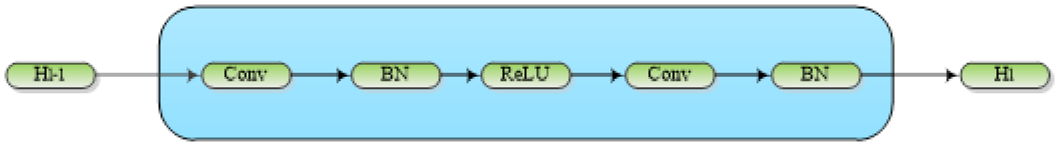


Figure 2: Composition of a layer in the network

In this model, each layer consists of multiple base layers, as shown in Fig. 2, including: the convolutional layer, the Batch Normalization (BN) layer, and the ReLU activation function. Among them, the main effect of BN is to batch normalize the input data and prevent the gradient dispersion problem caused by the uneven distribution of the input data, so that the weights of different levels can be adjusted in unison at the time of updating. The baseline structure of the stochastic depth network we use is based on ResNet. In a block of the network, we use the standard convolution + batch normalization + ReLU structure. Convolutional layer is a simplified form of full connection: incomplete connection and parameter sharing. Inspired by local vision, the weak influence outside the local is directly wiped to zero, while retaining the spatial position information, greatly reducing the parameters and Make training controllable. The BN (batch normalization) layer normalizes the distribution of the input values of any neuron in each layer of the neural network to a non-standard normal distribution, so that the activation input value falls in a region where the nonlinear function is sensitive to the input. It avoids the problem of gradient disappearance. Moreover, the gradient becomes larger, which means that the learning convergence speed is fast, and the training speed can be greatly accelerated. The ReLU layer will make the output of a part of the neurons 0, which causes the sparseness of the network and reduces the interdependence of parameters, which alleviates the problem of over-fitting. After the block layer of the network, we use the fully connected layer to fit the features extracted by the stochastic depth network. Finally, we used the depth energy function to evaluate the output.

In the process of training, the preceding network layer contains a large number of fundamental features of data, and consequently the previous layers should be kept as much as possible to extract more eigenvalues of the input vector, so the probability of the previous network to be ignored should be as small as possible. We can set the probability P_l of the l_{th} layer to be a layer-dependent linear function, and then the resulting feedforward propagation equation is as follows:

$$h_l(\boldsymbol{x}) = \begin{cases} \frac{l}{n_{h+v}}(1-C)F(\boldsymbol{x},\{\boldsymbol{W}_i\}) + x \quad, \ z = 1 \\ \left[1 - \frac{l}{n_{h+v}}(1-C)\right]F(\boldsymbol{x},\{\boldsymbol{W}_i\}) + x, \ z = 0 \end{cases} \tag{3.2}$$

In the above equation, n_{h+v} is the total number of layers in the network, h and v denote hidden layers and explicit layers respectively, while C is a constant value.

However, the probability P_l of l_{th} layer that can be reserved in the above equation cannot accurately represent the relationship between the probability of current layer being preserved and the input vector dimension. Since the performance of the network has certain relation with the dimension of the input data. Therefore, we can set a Dirichlet distribution which is a prior distribution associated with the input vector. Because the Dirichlet distribution is the conjugate prior distribution of multinomial distribution, the advantage of taking the conjugate prior distribution of prior probabilities is that whenever a new observation data is available, we can use the previous calculation of the posteriori probability as the priori probability and then multiplied by the likelihood of the new data to obtain a new a posteriori probability, so that it can bypass the step of obtaining a posteriori probability which is the priori probability multiplied by the likelihood of all data. Therefore, using Dirichlet distribution can improve the performance of the network and make results more accurate. The probability that the l_{th} layer can be preserved as:

$$P_l = 1 - \frac{l}{n_{h+v}}[1 - Dri(\boldsymbol{x} \parallel \boldsymbol{\alpha})] \tag{3.3}$$

Where Dri denotes a priori distribution related to the dimension of the input vector, since it is a conjugate prior of the multinomial distribution and can represent a multidimensional input vector whose distribution probability density is:

$$p(\boldsymbol{x} \mid \boldsymbol{\alpha}) = \frac{\Gamma(\sum_{i=1}^{K} \alpha_i)}{\Gamma(\alpha_1)\Gamma(\alpha_1)\ldots\Gamma(\alpha_K)} \prod_{i=1}^{K} x_i{}^{\alpha_{i-1}} \tag{3.4}$$

Where f_l denotes the convolution unit and F is the ReLU constraint we used in the neuron's output in the above equation. So, the whole network training process is based on the probability of P_l to control whether to skip the l_{th} layer block, while we keep the output of all neurons is active in the process of testing so that neurons of entire nets are involved in working.

Based on ResNet, StochasticNet introduces the design of random variables to effectively overcome the over-fitting and make the model better generalized. Simultaneously, we set a Dirichlet distribution to make results more accurate.

3.2 Deep energy model

The idea of the energy function comes from physics. Molecules move vigorously at high temperatures and can overcome local constraints. When gradually descending to low temperatures, the molecules will eventually align with a regular structure, which is also a low-energy state. The low temperature state is a relatively stable state compared to the entire system. It is a state in which many states have a high probability of occurrence. If it is referenced in the objective function of machine learning. The state with the lowest energy is the extreme value of the objective function. One of the main tasks of statistical pattern recognition is to capture the correlation between variables. The same energy model also captures the correlation between variables. The degree of correlation between variables determines the level of energy.

Stochastic neural networks are rooted in statistical mechanics. Inspired by the energy functionals in statistical mechanics, an energy function is introduced. The energy function is a measure that describes the state of the entire system. The more ordered the system or the more concentrated the probability distribution, the smaller the energy of the system.

Conversely, the more disordered the system or the more uniform the probability distribution, the greater the energy of the system. The minimum value of the energy function corresponds to the most stable state of the system.

The energy-based model is a model framework with universal meaning, including the traditional discriminant model and the generated model under its framework. The energy model captures the dependencies between variables by applying a range-limited energy to each configuration of the variables. Its main purpose is to find an appropriate energy function so that the correct input and output energy in the sample is lower than the energy of the wrong input and output. The energy model assigns a scalar energy value to each random variable value, and its training target is transformed to minimize the energy of the entire system. At the same time, the depth energy model can also be regarded as a probability distribution that is defined by the combination of feedforward depth neural network and energy model, and the energy-based probability model can also be transformed into a Boltzmann distribution. For the input vector of the model, the energy function is used to describe the degree of information correlation between the network layers and the joint probability density function of the hidden layer and the explicit layer. That is, the joint probability density is maximized by training, and the energy function is as small as possible. The meaning is similar to thermodynamics, and the smaller the value of the energy function finally obtained, the more stable the system is.

The depth energy model is a way to combine deep neural networks with energy functions. The energy model can be viewed as the prediction of data's distribution, besides, the energy value we derive from the energy equation is a scalar energy value, so we can think of it as a depth classification model that is calculated by energy values and has the capability of feature extraction and discrimination. In the energy model, the energy can be expressed as:

$$E_\theta(\boldsymbol{x},\boldsymbol{y},\boldsymbol{h}) = \sum_{i=1}^{n_v}\sum_{j=1}^{n_h} \boldsymbol{h}_j W_{j,i}\boldsymbol{x}_i - \sum_{i=1}^{n_v}\sum_{j=1}^{n_h} \boldsymbol{h}_j U_{j,i}\boldsymbol{y}_i - \sum_{j=1}^{n_h} \boldsymbol{b}_j\boldsymbol{y}_j - \sum_{i=1}^{n_v} \boldsymbol{a}_i\boldsymbol{x}_i \tag{3.5}$$

Where $\boldsymbol{x}$ is the input vector, which is the output of the previous layer and the input of the next layer in the feedforward propagation of the network model. W is the weights of the connection between explicit layers and hidden layers. And U is the weight of the connection between classes layers and hidden layers. While $\boldsymbol{y}_i$ represents the target vector of class i. $\boldsymbol{b}$ and $\boldsymbol{a}$ represent the bias of hidden layers and explicit layers respectively, $\boldsymbol{h}$ denotes neurons of hidden layers. The deep energy model can be regarded as a probability distribution which is redefined by the feedforward deep neural network model and energy model which training goal is to make the energy value of the whole system to attain a minimum value.

For the input vector of the model, the energy function is used to describe the information association degree between network layers and joint probability density function of hidden layers and explicit layers. However, we need to compute the free energy F finally. It can be computed by sum of the expected energy and the negative entropy H. So it is expressed as:

$$F(\boldsymbol{x},\boldsymbol{y}) = E_\theta(\boldsymbol{x},\boldsymbol{y},\boldsymbol{h}) + \big(-H(\boldsymbol{x},\boldsymbol{y})\big) \tag{3.6}$$

While the negative entropy's expectations are taken with respect to the posterior distribution of the hidden values, $P(\boldsymbol{h} \mid \boldsymbol{x}, \boldsymbol{y})$. For example, when the hidden layer is equal to one, it can be expressed as:

$$P(\boldsymbol{h}=1 \mid \boldsymbol{x}, \boldsymbol{y}) = f(\sum \boldsymbol{x}\boldsymbol{w} + \boldsymbol{u}) \tag{3.7}$$

f represents the activation function, and we use the ReLU activation function. Because the local correlation of the input vector can be obtained and the weights can be well shared simultaneously by using the convolution operation, $\boldsymbol{w}$ and $\boldsymbol{u}$ represent hidden layers weights and explicit layers weights respectively. The joint probability density can be maximized by training, that is, the energy function is as small as possible. Its meaning is similar to thermodynamics, the smaller the energy function value is, the more stable the system is. And the energy function vector form of depth energy model is:

$$F(\boldsymbol{x}, \boldsymbol{y}) = E_\theta(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h}) + \log P(\boldsymbol{h} \mid \boldsymbol{x}, \boldsymbol{y}) \tag{3.8}$$

In our task, we need to minimize the energy function during training and update the parameters, where the parameter θ represents the set of parameters of our network model. Assuming that the target tag for the given training data is t_j, then the formula for the gradient of the parameters of the network during training can be written as the following formula as:

$$\boldsymbol{G}_\theta = \sum_{j=1}^{J} \left[t_j - Q(\boldsymbol{x}, \boldsymbol{y}_j)\right] \nabla_\theta Q(\boldsymbol{x}, \boldsymbol{y}_j) \tag{3.9}$$

Where $\boldsymbol{y}_j$ represents the class of the input data is j. So, the network in the parameters of optimization can be written when the following formula, where α represents the learning rate.

$$\theta := \theta - \alpha \boldsymbol{G}_\theta \tag{3.10}$$

So, we can get from the above, for a given input data $\boldsymbol{x}$, and its real category j. Then we can predict its category according to the following formula.

$$j = \arg max(\boldsymbol{x}, \boldsymbol{y}) \tag{3.11}$$

Where $\boldsymbol{y}$ is the output layer in the above formula. So we can use the Markov chain [Tieleman (2008)] to complete the optimization steps. The initial state of each step of the Markov chain is related to the state of the previous model, so the initialization process can be approximated to the distribution of the model.

In Algorithm 1 and Algorithm 2, we give the pseudo code description of the SADIE model in the forward propagation and back propagation respectively.

In summary, the deep neural network image classification model of this paper is a random depth network based on the improved depth energy model, which makes the network solve the problem of gradient disappearance well during training, and can greatly speed up the training speed of the network. In addition, when the network forwards, the deep energy model is used to train the network, so that the overall entropy of the network is minimized, which is beneficial to improve the accuracy of the network.

Algorithm 1: Forward propagation of SADIE (one epoch)

```
Parameters: $W_l$ (Weights of the $l$ th layer), $b_l$ (Biases of the $l$ th layer),
            $P_l$ (The skip probability of the $l$ th layer)
```

```
Inputs: The image inputs of the whole dataset, $X$
        The image labels of the whole dataset, $Y$
BEGIN
Initialize $W_l$ and $b_l$ , Divide $X$ and $Y$ into mini-batches
For $\boldsymbol{x}$ and $\boldsymbol{y}$ in mini-batches do
   For $l$ in (1 to L)
      Calculate the skip probability $P_l$ with (3.3)
      Perform the forward calculation with (3.2) to get the output $h_l(\boldsymbol{x})$ of the layer
   end for
   Calculate the output $\boldsymbol{h}$ of the model
   Get the Energy Function $E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h})$ according to (3.5)
end for
Return Energy functions $E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h})$ in each mini-batch
END
```

Algorithm 2: Backward propagation of SADIE (one epoch)

```
Parameters: $W_l$ (Weights of the $l$ th layer)
            $b_l$ (Biases of the $l$ th layer)
            $\alpha$ (Learning rate)
Inputs: The energy functions $E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h})$ in mini-batches
BEGIN
Define free energy function $F(\boldsymbol{x}, \boldsymbol{y})$ according to (3.8)
For $\boldsymbol{x}$ and $\boldsymbol{y}$ in mini-batches do
   Calculate the gradients $\boldsymbol{G}$ to $W_l$ and $b_l$ of $F(\boldsymbol{x}, \boldsymbol{y})$ with (3.9)
   Choose a right optimization method
   Update all the parameters of weights and biases $\theta := \theta - \alpha \boldsymbol{G}_\theta$
end for
Return the optimized SADIE model
END
```

3.3 Exploring the optimization of deep neural networks

In general, the more complex a deep network, the more accurate its ability to express data, but the training complexity of the network increases nonlinearly with the complexity of the network. We need to quickly converge the network to the optimal value when we are trying to find the optimal solution during network training. Therefore, it is very urgent to propose a general optimization algorithm suitable for different network structures. The most common optimization method in neural networks is the gradient descent algorithm. In Eq. (3.10) we refer to the stochastic gradient descent (SGD) method, however it is prone to fall into the local optimal solution. For the large amount of data and the complexity of the model for deep learning, if you use the batch gradient descent algorithm, the training will become slower and take up more memory. Therefore, we use the mini-batch method to divide the training set and use the optimization algorithm on the small batch data set. We used the mini-batch gradient descent method to train neural networks in experiments in Sections 4.1 and 4.2.

However, the traditional gradient descent algorithm is still slow in tasks such as image recognition. The Adam [Kingma and Ba (2014)] (Adaptive Moment Estimation) algorithm is an algorithm that combines the Momentum algorithm with the RMSProp algorithm. The Adam optimization algorithm is an extension of the stochastic gradient descent algorithm.

The Adam algorithm is different from the traditional random gradient drop. The stochastic gradient descent method maintains a single learning rate to update all weights, and the learning rate does not change during the training process. Adam calculates independent adaptive learning rates for different parameters by calculating the first moment estimation and second moment estimation of the gradient. Therefore, in the process of parameter optimization using the Adam algorithm, the parameter update process in Eq. (3.10) is changed to Eq. (3.12)-Eq. (3.14). Where $\boldsymbol{m}_t$ represents the first-order momentum, $\boldsymbol{v}_t$ represents the second-order momentum, t represents the training round, and ϵ represents the correction term. β_1 and β_2 are the parameters for the moving average, which are generally 0.9 and 0.999 respectively. These features make the Adam algorithm easy to use, do not need to manually control the change of learning rate, and the convergence speed of the neural network is faster.

$$\boldsymbol{m}_t \leftarrow \beta_1 \cdot \boldsymbol{m}_{t-1} + (1-\beta_1) \cdot \boldsymbol{G}_\theta^{(t)} \tag{3.12}$$

$$\boldsymbol{v}_t \leftarrow \beta_2 \cdot \boldsymbol{v}_{t-1} + (1-\beta_2) \cdot \left(\boldsymbol{G}_\theta^{(t)}\right)^2 \tag{3.13}$$

$$\theta_t := \theta_{t-1} - \alpha \cdot \widehat{\boldsymbol{m}_t} / \left(\sqrt{\widehat{\boldsymbol{v}_t}} + \epsilon\right) \tag{3.14}$$

However, some of the shortcomings of the Adam algorithm are not negligible. Reddi et al. [Reddi, Kale and Kumar (2018)] explored the convergence of the Adam algorithm in the article, and through the counter-examples, Adam may not converge in some cases. Moreover, since deep neural networks often contain a large number of parameters, in such a very high dimensional space, non-convex objective functions tend to fluctuate, with countless highlands and depressions. Some are peaks, and it is easy to cross them by introducing momentum; but some are plateaus, and they may not be able to find them many times, so they stopped training. This will cause the model to miss the global optimal solution.

In order to make the deep neural network model can both converge quickly and avoid the local optimal solution. We consider combining the Adam algorithm with the SGD algorithm. We use the Adam algorithm in the early stage of training to enjoy the advantage of Adam's fast convergence. In the later stage of training, we switch the algorithm to SGD and slowly find the optimal solution. Nitish Shirish Keskar et al. [Keskar and Socher (2017)] have a conversion mechanism that attempts to make the model automatically turn to SGD after a certain round of Adam. They monitored the projection of the Adam algorithm step on the gradient subspace and used its exponential average as an estimate of the SGD learning rate after switching. In addition, when it is detected that the monitoring amount has not changed, the switching is triggered.

$$\lambda_t = \beta_2 \cdot \lambda_{t-1} + (1-\beta_2) \cdot \gamma_t \tag{3.15}$$

$$\left|\frac{\lambda_t}{(1-\beta_2^t)} - \gamma_t\right| < \epsilon \tag{3.16}$$

Eq. (3.15) shows the noise estimate in the switching algorithm. Eq. (3.16) shows the conditions for switching from Adam to SGD. γ_t is a noisy estimate of the scaling needed, λ_t is the exponential average. When the optimization algorithm goes to SGD, the learning rate changes to $\Lambda = \lambda_t/(1-\beta_2^t)$. In the specific experiment, we found that if the optimization algorithm is directly converted from Adam to SGD when the conversion conditions are met, the loss of the model will be unstable. Therefore, we have devised a technique to add a buffering process to the conversion algorithm. In the training process of the model, when the conversion condition of Eq. (3.16) is satisfied, we adopt a buffer mechanism to combine the parameter variation calculated by Adam and SGD algorithm. At the same time, the proportion of Adam's algorithm is decreasing, and the proportion of SGD algorithm is increasing until the SGD algorithm is fully adopted.

$$\theta_t := \theta_{t-1} - \alpha \cdot \left[\frac{i}{k} \cdot \boldsymbol{G}_\theta^{(t)} + \frac{k-i}{k} \cdot \widehat{\boldsymbol{m}}_t / \left(\sqrt{\widehat{\boldsymbol{v}}_t} + \epsilon\right)\right] \tag{3.17}$$

As shown in Eq. (3.17), if the transition phase undergoes *k* training, then in the *i*-th training of the transition phase, the assignment of Adam and SGD is as shown in the formula.

4 Experiments in image classification

In this section, we conduct a series of experiments. We mainly compare the performance of ResNet [He, Zhang, Ren et al. (2015)], Stochastic Depth Network [Huang, Sun, Liu et al. (2016)] and the improved stochastic depth network with deep energy model (SADIE) in different situations, and compare the performance of the model when the amount of data and the layers are different. Then, we explored the optimization algorithm problem of deep neural networks. We added an optimization mechanism combining the Adam and SGD algorithms to SADIE and compared it to SADIE using only the mini-batch gradient descent method. The basic convolutional neural network model used in this algorithm uses residual blocks, the number of layers in the network can be deep. The 110 and 152 layers are already published baseline structures.

Define the following performance indicators:

$$P = \frac{CRAB}{CRAB+FLAB} \tag{4.1}$$

$$R = \frac{CRAB}{CRAB+NOBEL} \tag{4.2}$$

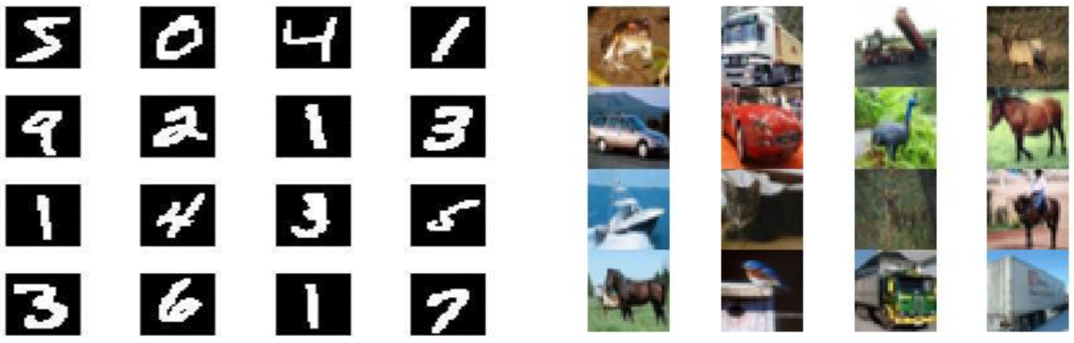
$$F1 = \frac{2PR}{P+R} \tag{4.3}$$

We use precision, recall and *F1-Measure* evaluation indicators, where *P* denotes the precision that is the proportion of correctly predicted image tags to actual output tags,

$CRAB$ indicates the correct output labels, and $FLAB$ indicates mismatched labels with the image which are output to the correct labels. R (4.2) represents the recall rate and indicates the ratio of the correct output labels to all correct labels, and $NOBEL$ indicates the image labels that should be output without output. *F1-Measure* (4.3) is based on the harmonic mean of precision and recall. Through the accuracy and recall rate we can generally know the performance of the model.

During the training in deep learning, most engineers use the GPU to accelerate without using the CPU. The main reason is that the CPU is the central processing task. The GPU is a graphics processing unit, which is most suitable for floating-point operations, which is in line with the large number of fractional operations that exist in network training. In order to make good use of the GPU, we use small batches for training, which can avoid the problem of GPU memory explosion caused by the network optimization process, and also make full use of the GPU. The way we use it is to use the GPU to perform a large number of convolution operations, but use the CPU to perform a large number of logical judgments, which fully exerts the role of the CPU in the logic processing. This separated design can be efficiently implemented using the device interface provided by TensorFlow.

We trained 350 epochs for each model in training. The hardware environment of the experiment is the Titan Xp GPU and the software environment is Python and TensorFlow. The spatial occupancy of the three models is not much different, and the model parameters of the deep convolutional neural network occupy most of the memory space. For the 110-layer network, it takes about 500 MB of memory space; for the 152-layer network, it takes about 650 MB of memory space.



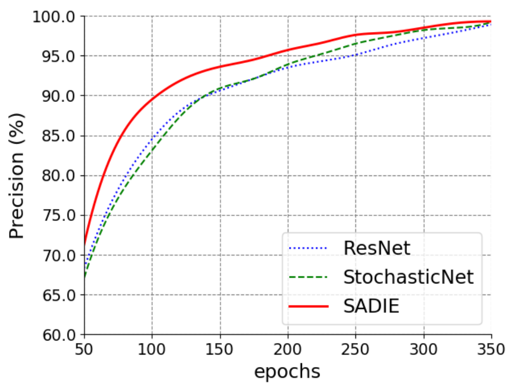
(a) MNIST (b) CIFAR-10

Figure 3: The two data sets used in this article are MNIST and CIFAR-10. Among them, MNIST is obtained from http://yann.lecun.com/exdb/mnist/, and CIFAR-10 is obtained from https://www.cs.toronto.edu/~kriz/cifar.html

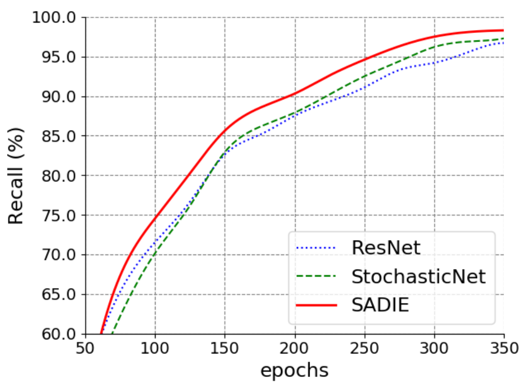
4.1 Experiments on MNIST

The hand-written digital picture database MNIST consists of a training data set and a test data set. The training sample set includes 60,000 samples and the test sample set includes 10,000 samples. These training samples have been normalized to a fixed size. The picture size of the MNIST data set is 28×28, and we perform random image enhancement and other operations in the image preprocessing work. In the training process, we set the mini-batch to 128, the learning rate is 0.01, and the learning rate attenuation mechanism is adopted. We choose the mini-batch SGD algorithm as the optimization algorithm to optimize the energy function. We added a dropout layer after the convolutional layer of the network model to reduce the risk of overfitting. Finally, we use the ten-classified SoftMax layer as the output of the model. Examples of MNIST data are shown in Fig. 3.

In the experiment of MNIST, we compare the three 110-layer and 152-layer models' performance. From the experimental results in Fig. 4 and Tab. 3, we can learn that the precision of the model SADIE proposed in this paper is 0.38% higher than that of ResNet after MNIST training in the 110-layer network, and the precision rate is 0.14% higher compared with Stochastic-Net. In terms of recall rate, the overall recall rate of the model presented in this paper can reach 98.30%, its recall rate is 1.65% higher than ResNet, and 1.02% higher than Stochastic-Net's. At the same time, it can also be seen that the model proposed in this paper is higher than the above two models in F1-Measure, it is 1.02% higher than ResNet and 0.58% higher than Stochastic-Net. These experimental results show that our model not only have advantages in precision, but also have a good coverage of the images' labels.



(a) Precision rate



(b) Recall rate

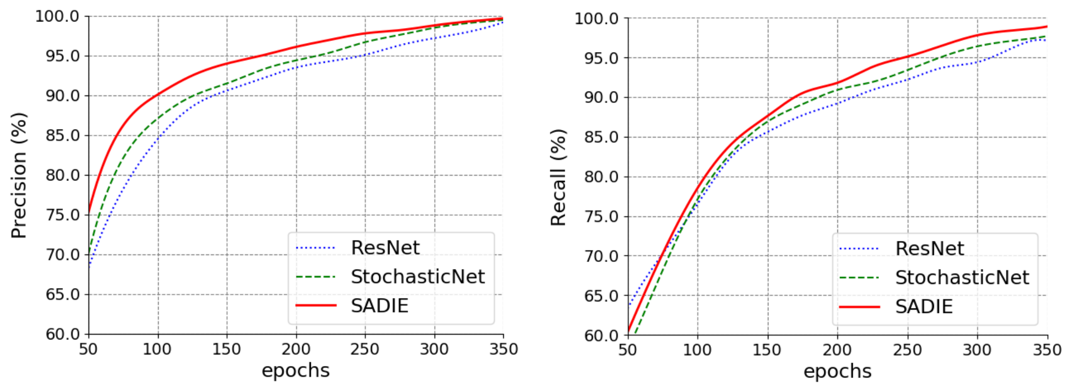
Figure 4: Comparison of the performance of three 110-layer models with MNIST, the blue line represents ResNet, green line represents Stochastic Depth Net and the red line represents our model(SADIE)

Besides that, in the comparison of the average time of training, we can see from Tab. 1 that although our model incorporates the calculation of energy, the training time of network does not increase too much. Our proposed SADIE model is 4.08% faster than ResNet, which is slower than the Stochastic-Net by 2.17% of the average training time of MNIST.

Table 1: Comparison of average training time of three 110-layer models in MNIST

Algorithm	Training time
ResNet	1 *h* 38 *min*
StochasticNet	**1 *h* 32 *min***
SADIE	1 *h* 34 *min*

On the other hand, we set up 152 layers for each of the three networks, using MNIST for training and the training indicators are shown in Fig. 5 and Tab. 3. From the experimental results we can see that when we set the three models to 152-layer, and after we use MNIST training, the precision rate of the model SADIE proposed in this paper is 0.59% higher than that of ResNet, which is 0.39% higher than Stochastic-Net's precision. In terms of recall rate, the overall recall rate of the model presented in this paper can rise to 98.90%, which is 1.85% and 1.23% higher than ResNet and StochasticNet respectively. At the same time, the model proposed in this paper is higher than the above two models in *F1-Measure*. It can be seen from the experimental results that with the increase of the number of layers, the proportion of the model in the precision, recall and *F1-Measure* values increase. It can be shown that with the increase of the network layers, the model proposed in this paper has the advantages of precision and recall rate. In addition, our model in precision, recall and *F1-Measure* aspects increase ratio than the other two models when we changed models from 110 layers to 152 layers.



(a) Precision rate (b) Recall rate

Figure 5: Comparison of the performance of three 152-layer models with MNIST

Furthermore, in contrast to the average time of training, we can see from Tab. 2 that the SADIE model proposed in this paper is 5.31% faster than the ResNet network and 1.90% slower than the Stochastic-Net. Although the increase in network layers improves the accuracy, recall rate and *F1-Measure*, the models' training time is relatively increased. However, the increase in time can be ignored compared with the average training time for large amounts of data.

Table 2: Comparison of average training time of three 152-layer models in MNIST

Algorithm	Training time
ResNet	1 *h* 53 *min*
StochasticNet	**1 *h* 45 *min***
SADIE	1 *h* 47 *min*

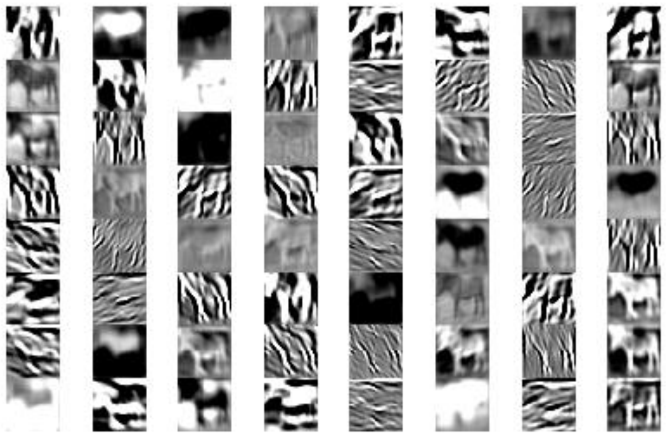
Table 3: Precision, recall rate and *F1-Measure* of the three models in MNIST when networks are 110 layers, the three models in MNIST when networks are 152 layers and the three models in CIFAR-10 when networks are 152 layers

Algorithm	Task	Precision (%)	Recall(%)	F1(%)
ResNet	MNIST/110layers	98.99	96.70	97.83
StochasticNet	MNIST/110layers	99.23	97.30	98.26
SADIE	MNIST/110layers	**99.37**	**98.30**	**98.83**
ResNet	MNIST/152layers	99.20	97.10	98.14
StochasticNet	MNIST/152layers	99.40	97.70	98.54
SADIE	MNIST/152layers	**99.79**	**98.90**	**99.34**
ResNet	CIFAR-10/152layers	93.59	87.1	90.23
StochasticNet	CIFAR-10/152layers	94.77	87.9	91.21
SADIE	CIFAR-10/152layers	**95.10**	**91.3**	**93.16**

4.2 Experiments on CIFAR-10

Next, we set the three models to 152-layer architecture and use CIFAR-10 to train models. The CIFAR-10 dataset contains 60,000 32×32 color images in 10 categories. There are 50,000 training images and 10,000 test images. The data set is divided into 5 training blocks and 1 test block, each block has 10,000 images. The test block contains 1000 images randomly selected from each class. Training blocks contain these images in a random order, but some training blocks may contain more images than other classes. Training blocks contain 5000 images per category. In the training process, we set the mini-batch to 128, the learning rate is 0.01, and the learning rate attenuation mechanism is adopted. We choose the mini-batch SGD algorithm as the optimization algorithm to optimize the energy function. We added a dropout layer after the convolutional layer of the network model to reduce the risk of overfitting. Finally, we use the ten-classified SoftMax layer as the output of the model. Examples of CIFAR-10 data are shown in Fig. 3.

(a) An image of horse



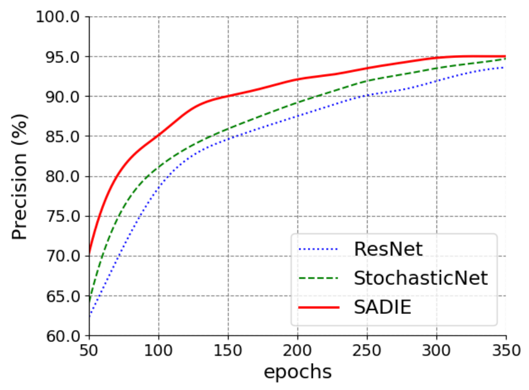
(b) Image after a layer of the SADIE model

Figure 6: (a) Shows an image in CIFAR-10, and (b) Shows a multi-channel grayscale image of the image processed through a layer of the SADIE model

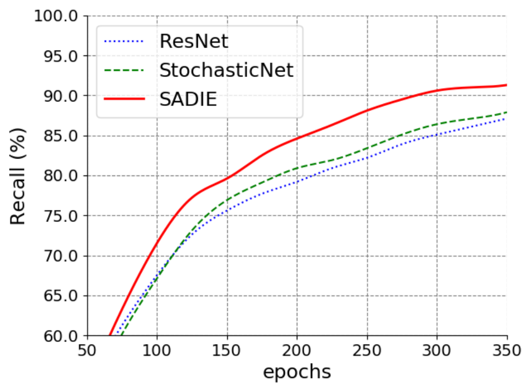
Fig. 6 shows the image of a horse in CIFAR-10, and the 64-channel grayscale image of the image processed by the SADIE model. We can see that the images of different channels retain the different features of the original image, such as contour features and position features. Some of these images show a relatively obvious noise property, which is due to some operations such as dropout.

The training results are shown in Fig. 7 and Tab. 3. It can be seen from the experimental results that when the models' architecture are 152 layers, the model of this paper is 1.6% higher than that of ResNet and 0.4% higher than StochasticNet in terms of precision. In terms of recall, when the data set is CIFAR-10 and the models still remain at 152 layers, our model is 4.8% higher than ResNet and 3.9% higher than StochasticNet in terms of recall rate. In the comparison of *F1-Measure*, our model is 3.2% and 2.1% higher than ResNet and StochasticNet respectively.

In addition, from the point of view of average training time, the average training time for our model is 5.10% faster than ResNet and 2.40% slower than the Stochastic-Net.



(a) Precision rate



(b) Recall rate

Figure 7: Comparison of the performance of three 152-layer models with CIFAR-10, the blue line represents ResNet, green line represents Stochastic Depth Net and the red line represents our model(SADIE)

Table 4: Comparison of average training time of three 152-layer models with CIFAR-10

Algorithm	Training time
ResNet	3 *h* 35 *min*
Stochastic-Net	**3 *h* 20 *min***
SADIE	3 *h* 24 *min*

Through the above experiments, we can see that our model has a good performance compared to the other two models ResNet and StochasticNet.

Besides that, the hyper-parameter λ in the previous Dirichlet distribution related to input vectors also affects the performance of our model, so take the appropriate value to make our model has a good performance is still worthy of our study. We need to choose suitable evaluation criteria to determine whether the λ value makes the overall performance of the model to achieve the best. Therefore, we use BEP (Break-Even Point) to evaluate the performance of the model when the λ takes different values, and BEP refers to the value of the model when the precision and recall rate are equal.

$$BEP = F1 = P = R \tag{4.4}$$

From the experimental result we can conclude that when the hyper-parameter λ is 50, the BEP of the whole model reaches the maximum, so our model has always set λ to the fixed value of 50 in the above experiment, which can make our performance best.

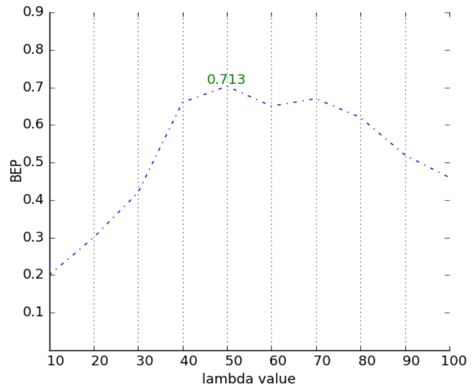


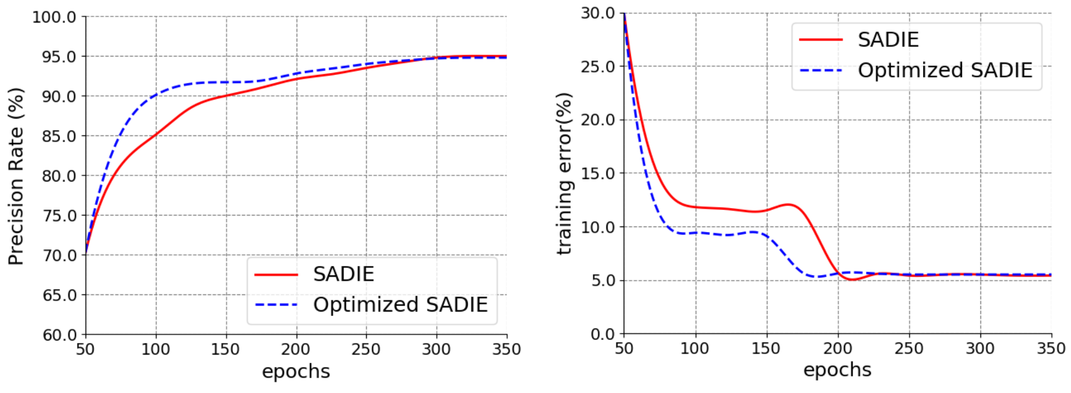
Figure 8: The change curve of BEP when the hyper-parameter λ takes different values

4.3 Further optimization of the performance of deep neural networks

In Section 3.3 we discussed the optimization algorithm for neural networks in the deep learning domain. For tasks such as image classification, the amount of data is large, and the neural network has many adjustable parameters. We need both an optimization method with fast convergence (Adam) and an optimization method (SGD) that can achieve global optimal values. Therefore, this paper designs an improved automatic switching mechanism between Adam algorithm and SGD algorithm. We chose the SADIE model of 152-layer in Section 4.2 and replaced its optimization algorithm with the method of combining Adam with SGD. Similarly, it was used to train the CIFAR-10 data set and get the model's loss curve and the classification results in the test set. Finally, we compare the data metrics of the standard SADIE with the data metrics of SADIE using the improved optimization algorithm.

In Fig. 9, We compared the performance of the standard SADIE and the optimized SADIE on CIFAR-10, using training loss and precision rate as evaluation indicators. From (a) we can see that in the early stage of training (before 100 epochs), the precision rate of the optimized SADIE model using Adam algorithm is significantly faster than the standard SADIE. Starting with the 150th epoch, the precision rate of the SADIE using the Adam algorithm tends to be stable. We can see that using Adam's model is more difficult to achieve convergence and improve accuracy. As the training continues, the optimized SADIE model is automatically converted to the SGD algorithm to continue training. At the end, the two models achieve a considerable precision rate.

From (b) we can see that in the training of the first 100 epoch, the model using the Adam algorithm converges faster than the SGD. But when the optimized SADIE training error reaches 9.5%, it tends to be stable. After the optimization algorithm was converted to SGD, the optimized SADIE eventually converges to a 5.5% training error, which is essentially the same as standard SADIE, but uses fewer epochs. From the experimental results, we can analyze and draw certain conclusions. The model using the transformation optimization algorithm can achieve similar effects to the original model, but reduce the number of training iterations. This method compensates for the shortcomings of Adam and SGD to a certain extent, and has practical significance in dealing with the task of image recognition, such as large amount of data and large amount of calculation.



(a) Precision rate (b) Training error

Figure 9: Comparison of SADIE and Optimized SADIE in Training Error and Precision Rate with CIFAR-10

In summary, it is a big direction to correct Adam by limiting the update step and lowering the bounds in the later stages of training and trying to make the parameters update steps are similar. It is certainly feasible to cut SGD with Adam first, but it is still not elegant enough. If you can use a unified iterative algorithm to take into account Adam's fast convergence ability and SGD's good generalization ability, it will be a big improvement. This is also our next research direction.

5 Conclusions

In this paper, we proposed stochastic depth network based on deep energy model (SADIE) to achieve the image classification process. Then, we apply the Dirichlet distribution when we choose whether to skip the layers in stochastic depth network, making the model more suitable for the dimension of input images and improving the performance of our model. At the same time, we also explored the possibility of using Adam and SGD combination optimization in deep neural networks, and designed an improved switching mechanism. The Adam algorithm is used to accelerate convergence in the early stage of training, and the SGD algorithm is used to search for the optimal solution slowly in the later stage of training.

In experiments, we compared the performance of each model in different network layers and different datasets. We can obtain from the above experimental results that when we increase the number of layers and the size of image datasets, the accuracy and recall rate of our model is higher than that of ResNet and Stochastic-Net. Also, we used the SADIE model using the improved optimization method for the CIFAR-10 experiment and compared it to the standard SADIE model. The optimized SADIE greatly improved the convergence speed while maintaining the accuracy, which confirmed the feasibility of optimizing the combination of Adam and SGD algorithms.

All of the above experimental results show that our model for image classification have great advantages in training time and accuracy. But on the other hand, we only use simpler single scene images for experiments and do not consider the recognition of complex content images. We are planning to provide a more comprehensive image processing model through detailed analyzing and data mining in the future.

Acknowledgement: This work is supported by National Key Research and Development Program of China (2016YFE0204500), and Industrial Internet Innovation and Development Project 2018 of China.

References

Bengio, Y. (2009): Learning deep architectures for AI. *Foundations & Trends in Machine Learning*, vol. 2, no. 5, pp. 1-127.

Bengio, Y.; Simard, P.; Frasconi, P. (1994): Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157-166.

Glorot, X.; Bengio, Y. (2010): Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research*, vol. 9, pp. 249-256.

Gurusamy, R.; Subramaniam, V. (2017): A machine learning approach for MRI brain tumor classification. *Computers, Materials & Continua*, vol. 53, no. 2, pp. 91-108.

He, K. M.; Sun, J. (2015): Convolutional neural networks at constrained time cost. *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5353-5360.

He, K. M.; Zhang, X. Y.; Ren, S. Q.; Sun, J. (2015): Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-12.

Hinton, G.; Osindero, S. (2006): A fast learning algorithm for deep belief nets. *Neural Computation*, vol. 18, no. 7, pp. 1527-1554.

Huang, G.; Sun, Y.; Liu, Z.; Sedra, D.; Weinberger, K. (2016): Deep networks with stochastic depth. *European Conference on Computer Vision*, pp. 1-16.

Ioffe, S.; Szegedy, C. (2015): Batch normalization: accelerating deep network training by reducing internal covariate shift. *International Conference on Machine Learnin*g, pp. 448-456.

Jaderberg, M.; Czarnecki, W. M.; Osindero, S.; Vinyals, O.; Graves, A. et al. (2016): Decoupled neural interfaces using synthetic gradients. *arXiv preprint arXiv:*1608.05343.

Keskar, N. S.; Socher, R. (2017): Improving generalization performance by switching from adam to sgd. *arXiv preprint arXiv*:1712.07628.

Kim, T.; Bengio, Y. (2016): Deep directed generative models with energy-based probability estimation. *International Conference on Learning Representations*, pp. 1-9.

Kingma, D.; Ba, J. (2014): Adam: A method for stochastic optimization. *arXiv preprint arXiv:*1412.6980.

Krizhevsky, A.; Sutskever, I.; Hinton, G. (2012): ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, pp. 1097-1105.

Lv, H. X.; Yu G.; Tian, X. Y.; Wu, G. (2014): Deep learning-based target customer position extraction on social network. *Management Science & Engineering*, pp. 590-595.

Ngiam, J.; Chen, Z. H.; Koh, P. W.; Ng, A. Y.; (2012): Learning deep energy models. *International Conference on Machine Learning*, pp. 1105-1112.

Reddi, S. J.; Kale, S.; Kumar, S. (2018): On the convergence of adam and beyond. *International Conference on Learning Representations*.

Saxe, A. M.; McClelland, J. L.; Ganguli, S. (2013): Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:*1312.6120.

Simonyan, K.; Zisserman, A. (2014): Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:*1409.1556.

Szegedy, C.; Liu, W.; Jia, Y. Q.; Sermanet, P.; Reed, S. et al. (2015): Going deeper with convolutions. *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-9.

Tieleman, T. (2008): Training restricted boltzmann machines using approximations to the likelihood gradient. *International Conference on Machine Learning*, pp. 1064-1071.

Yuan, C. S.; Li, X. T.; Wu, Q. M. J.; Li, J.; Sun, X. M. (2017): Fingerprint Liveness Detection from different fingerprint materials using convolutional neural network and principal component analysis. *Computers, Materials & Continua*, vol. 53, no. 3, pp. 357-371.