A new learning algorithm based on strengthening boundary samples for convolutional neural networks

Dongning Zhou[1], *Lu* Lu[1], *Junhong* Zhao[1], *Dali* Wang[2], *Wenlian* Lu[3] *and Jie* Yang[1*]

1 School of Mathematical Sciences, Dalian University of Technology, Dalian, Liaoning, China.
2 Central Laboratory, Dalian Children's Hospital of Dalian medical university, China
3 Institute of Science and Technology for Brain-inspired Intelligence, Fudan University

Abstract. CNN is an artificial neural network that can automatically extract features with relatively few parameters, which is the advantage of CNN in image classification tasks. The purpose of this paper is to propose a new algorithm to improve the classification performance of CNN by strengthening boundary samples. The samples with predicted values near the classification boundary are recorded as hard samples. In this algorithm, the errors of hard samples are added as a penalty term of the original loss function. Multi-classification and binary classification experiments were performed using the MNIST data set and three sub-data sets of CIFAR-10, respectively. The experimental results prove that the accuracy of the new algorithm is improved in both binary classification and multi-classification problems.

1 Introduction

Recently, various strategies have been introduced to boost the performance of Convolutional Neural Networks (CNNs). Irsoy et al. [1] proposed introduced dropout to prevent overfitting to increase the generalization. Zoumpourlis et al. [2] presented a second-order convolutional network that combines linear and non-linear filters. Juefei-Xu F et al. [3] introduced the local binary convolution (LBC) to replace the convolutional layer, which improved the computing time of the network. Uchida et al. [4] showed a couple convolutional layer with mutually constrained weights to produce better performance.

Some existing literatures optimize the learning algorithms for CNN. Zhining et al. [5] combined the genetic algorithm and CNN to extract better expression features than CNN dose. Hu et al. [6] adopted Gaussian convolution into CNN to accelerate the training convergence. Kim et al. [7] used the Extreme Learning Machine to calculate the weights between the hidden and output layers to shorten the training time. Yang et al. [8] combined the dropout and stochastic gradient descent optimizer to form a modified CNN algorithm, which improved the accuracy of CNN.

In this paper, a new learning algorithm based on strengthening boundary samples is proposed to improve the classification accuracy of CNN. The importance of each sample is adjusted in each training step according to its predicted value to pay more attention on the hard-classified samples. The rest of this paper is organized as follows. In Sect. 2, the structure of CNN is briefly introduced. In Sect. 3, we provide our new learning algorithm. Sect. 4 discusses the experimental results. Finally, Sect. 5 is the conclusion of this paper.

2 Related works

2.1 Convolutional neural network

Generally, a CNN contains several blocks including the convolution and pooling layers, and a fully connected layer. The convolutional layer has a number of convolution filters to extract the local characteristics of the image

$$x_j^l = f\Big(\sum_{i\in M_j} x_i^{l-1} * W_{ij}^l + b_j^l\Big) \tag{2.1}$$

where x_j^l denotes the j-th feature map in the l layer of the network, M_j is the set of feature maps in the l-1 layer, w_{ij}^l represents the convolution filter, b_j^l is the bias of the j-th feature map in the l layer, * is 2D convolution operation, and $f(\cdot)$ is the activation function. The pooling layer compresses data and parameters

$$y_j^l = \beta_j^l p(x_j^l) \tag{2.2}$$

where β_j^l is the weight and $p(\cdot)$ is the pooling function.

2.2 The new learning algorithm

The back propagation (BP) algorithm [9] is a popular algorithm to train CNN. Generally, the mean square error and cross-entropy loss are used as the loss function for two and multi classification problems, respectively.

* Corresponding author: yangjiee@dlut.edu.cn

The expressions of the two evaluation methods are as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(t_i - y_i)^2, \tag{3.1}$$

$$CE = -\frac{1}{n}\sum_{i=1}^{n} t_i \cdot \log y_i, \tag{3.2}$$

where n is the batch size, t_i and y_i represent the target value and predicted value of the i-th sample, respectively.

It can be seen from equations (3.1) and (3.2) that all samples are equally important during the training process. In fact, some samples are classified with higher confidence, while others are classified with lower confidence. The network does not make reasonable adjustments to complex and simple samples during the training process, resulting in low classification performance. In this paper, we solve this problem by constantly changing the importance of training samples during the training process to appropriately strengthen the training for complex samples.

Firstly, we divide samples into hard samples and easy samples. Easy samples are higher or lower predicted values, because the effect of these samples and the corresponding loss are in line with our expectations. For example, the corresponding loss of a sample with a small value is large, therefore, it has a poor effect and a large impact in training, which is in line with our expectations. However, there are some samples whose prediction value are near the classification boundary value, and the effect on classification are very poor, but the corresponding loss values are relatively small, which will lead to lower classification accuracy. We call this part of the samples hard samples.

The specific distinguishing method is to take an interval near the classification boundary value, the samples with predicted values within the interval are recorded as hard samples, and the samples with predicted values outside the interval are recorded as easy samples. We use the average of the predicted values of the same batch of samples to select the interval. Take the binary classification as an example, let the classification boundary value be *0.5*, the average value be *a*, let the distance from *a* to *0.5* be *d*, take *0.5* as the center value of the interval, and *2*d* as the interval length to obtain the interval [*0.5-d, 0.5 + d*]. Figure 1 shows the sample division in the binary classifications.

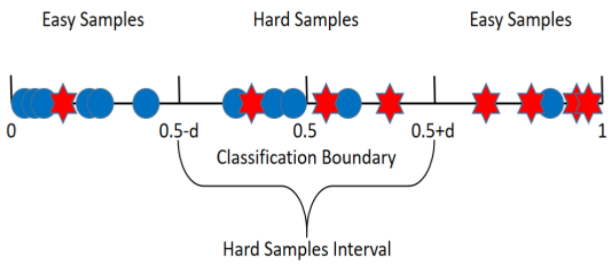

Fig. 1. The Classification of sample

Then we use a new algorithm, PCNN, to strength learning hard samples by adding a penalty term to the loss function. The penalty term extracts the information of the degree to which the sample is classified difficultly. The loss function for PCNN is expressed as:

$$NMSE = \frac{1}{n}\sum_{i=1}^{n}(t_i - y_i)^2 + \lambda\left[\frac{1}{m}\sum_{k=1}^{m}(y_k^{'} - (0.5 - d_k))^2 + \frac{1}{t}\sum_{l=1}^{t}((0.5 + d_l) - y_l^{''})^2\right] \tag{3.3}$$

$$NCE = -\frac{1}{n}\sum_{i=1}^{n} t_i \cdot log\, y_i - \lambda\left[\frac{1}{m}\sum_{k=1}^{m}(log\, y_k^{'} - log(\, 0.5 - d_k)) + \frac{1}{t}\sum_{l=1}^{t}(log(\, 0.5 + d_l) - log\, y_l^{''})\right] \tag{3.4}$$

where Equation (3.3) is the mean square error of the PCNN, Equation (3.4) is the cross-entropy loss function of the PCNN, λ is the proportion of the penalty term in the loss function, and the specific value of λ is determined by subsequent experiments, m and t are the number of hard samples on the left and right of *0.5* in the batch, respectively, $y_k^{'}$ is the value of the k-th hard samples to the left of *0.5*, $y_l^{''}$ is the value of the l-th hard samples to the right of *0.5*, and d_k, d_l are the average of the predicted values of the k-th and the l-th hard sample categories in the batch, respectively.

When $\lambda = 0$, the new loss function is equal to the original loss function. Therefore, the modified loss function can be regarded as a generalization of the original loss function. In addition, the effect of the penalty term changes as the value changes. By choosing the value of λ, the CNN can be better trained and avoid overfitting. In the experiment, since the accuracy is highest when the value of the penalty term $\lambda = 0.7$, $\lambda = 0.7$is selected for subsequent experiments. Figure 2 is the corresponding change of the classification accuracy with respect to the value of λ on the MNIST data set.

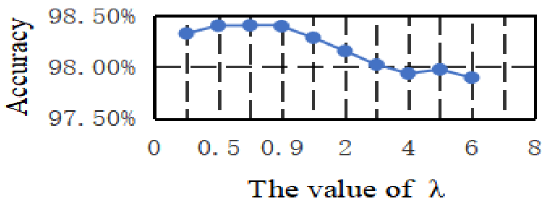

Fig. 2. Classification accuracy varies with λ

4 Experiments

In this section, some experiments are performed to test the effectiveness of the new learning algorithm for CNN on CIFAR-10 and MNIST datasets. The architecture of CNN is designed as shown in table 1. The ReLU function is used as the activation function [13] and Dropout [14] is used to avoid overfitting. In all experiments, the learning rate is set as 0.001 and the batch size choose 50. 10 experiments with different initial weights are performed on each dataset and the average accuracies are recorded.

Table 1. The architecture of CNN.

Layer	CIFAR-10	MNIST
Input layer	32×32×3	28×28×1
Conv layer	3×3×64	3×3×64
Conv layer	3×3×64	3×3×64
Max pooling layer	2×2	2×2
Conv layer	3×3×128	3×3×128
Conv layer	3×3×128	3×3×128
Max pooling layer	2×2	2×2
Conv layer	3×3×256	3×3×256
Conv layer	3×3×256	3×3×256
Conv layer	3×3×256	3×3×256
Fully Connected layer (dropout)	1024	1024
Output layer	1	10

4.1. CIFAR-10

The pictures in the CIFAR-10 are equally divided into ten categories and each category contains 5,000 training samples and 1,000 test samples. We constructed three sub-data sets for the binary classification problem from the CIFAR-10 dataset, of which the first and the second types constitute the first sub-data set, the fifth and the eighth to the second sub-dataset, and ninth and tenth to the third sub-dataset. In these sub-datasets, the loss function is Equation (3.3).

Figure 3 shows the test accuracies of PCNN and CNN for the first sub-dataset in different iterations. As we can see that PCNN perform better than CNN at any time. When iterations=1600, the accuracy of CNN reached 95.495%, while the accuracy of PCNN reached 95.885%. And in the whole process, the performance of PCNN is better than CNN, especially when the number of epochs is less than 800. If the number of iterations is 160, the classification accuracy of PCNN is 3.37% higher than that of CNN.

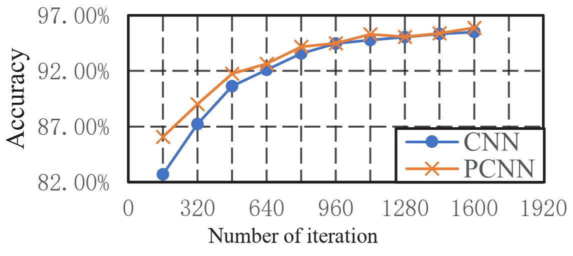

Fig. 3. Comparison on test set between CNN and PCNN on the first sub-dataset.

On the second sub-dataset, Figure 4 shows the classification accuracy curves of PCNN and CNN under different iterations. We can see that the PCNN curve is always higher than the CNN. With iteration=1600, the classification accuracy of traditional CNN is 86.95%, and the classification accuracy of PCNN is 87.306%. If the number of iterations is 640, the classification accuracy of PCNN is 1.188% higher than that of CNN.

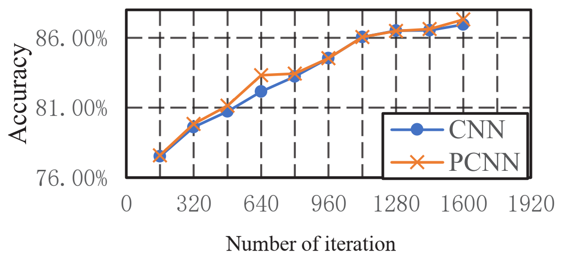

Fig. 4. Comparison on test set between CNN and PCNN on the second sub-dataset.

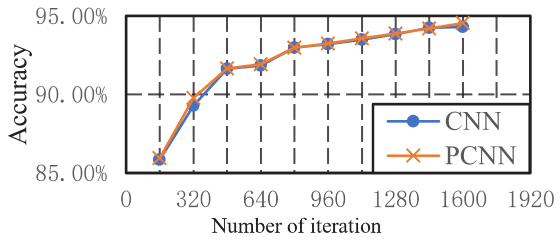

Fig. 5. Comparison on test set between CNN and PCNN on the third sub-dataset.

Figure 5 shows the classification accuracy curves of PCNN and CNN on the third sub-dataset. It is obvious that whether on PCNN or CNN, the classification accuracy improves with the increase of the times. After 1600 iterations, the accuracy of CNN reached 94.3%, and the accuracy of PCNN reached 94.54%. In this dataset, the effect of PCNN is not very obvious. This is because the difference between the hard and simple samples is small and the effect of the penalty term is small.

4.2. MNIST

MNIST is the dataset consists of handwritten digital pictures, which is divided into 10 categories, and the corresponding labels are 0 to 9 respectively. In this experiment, we use Equation (3.4) as a loss function to explore the impact of PCNN on multi-classification problems.

Experiments are performed on PCNN and CNN with different iterations, and the results are shown in Figure 6. It can be seen that the classification accuracy of PCNN can always be higher than CNN. Specifically, when the number of epochs is less than 11,000, the accuracy of PCNN is significantly higher than that of CNN. And when the number of periods is greater than 11,000, the improvement of PCNN becomes smaller. The reason for this phenomenon is that with time, the prediction accuracy of all samples will increase, and the increased error will decrease. After 33,000 iterations, the accuracy of CNN and PCNN is 99.165% and 99.185%, respectively.

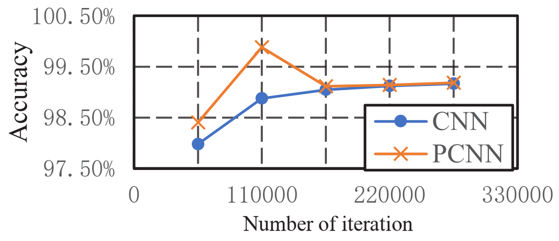

Fig. 6. Comparison on test set between CNN and PCNN on MNIST dataset.

5 Conclusion

In this paper, a new learning algorithm is proposed to enhance the classification performance of CNN. It focuses on dividing samples into easy and hard samples to strength learning the hard samples by adding a penalty term to the loss function of BP. Experimental results on the CIFAR-10 and MNIST datasets show that the new learning algorithm can improve the classification accuracy and speed of CNNs in image classification problems. In future study, we will try to set the weight of the penalty term with a dynamic parameter to make it adaptively to improve the performance of CNN.

The authors would like to thank the editor and reviewers for their constructive comments that greatly improved the quality of this paper. This work was supported by National Key R&D Program of China (No.2018AAA0100300), Shanghai Municipal Science and Technology Major Project (No. 2018SHZDZX01) and ZJLab and the National Natural Science Foundation of China (No. 11201051).

References

1. Irsoy O and Alpaydinodotn E (2018) Dropout Regularization in Hierarchical Mixture of Experts *arXiv* **1812.10158**
2. Zoumpourli G, Doumanoglou A and Vretos N,et al. (2017) Non-linear Convolution Filters for CNN-based Learning *IEEE International Conference on Computer Vision* **4771-4779**
3. Juefei-Xu F, Boddeti V and Savvides M (2017) Local Binary Convolutional Neural Networks *IEEE Conference on Computer Vision and Pattern Recognition* **4284-4293**
4. Uchida K, Tanaka M and Okutomi M (2016) Coupled Convolution Layer for Convolutional Neural Network *International Conference on Pattern Recognition* **3548-3553**
5. Zhining Y and Yunming P (2015) The genetic convolutional neural network model based on random sample Int. Journal of u-and e-Service, *Science and Technology* **8 pp 317-326**
6. Kim J, Kim J and Jang G J (2017) Fast learning method for convolutional neural networks using extreme learning machine and its application to lane detection Neural Networks **87 109-121**
7. LeCun Y, Boser B E and Denker J S (1990) Handwritten digit recognition with a back-propagation network *NIPS (Denver)* **396-404**
8. Nair V and Hinton G E (2010) Rectified linear units improve restricted boltzmann machines *ICML-10 (Haifa)* **807-814**
9. Srivastava N, Hinton G and Krizhevsky A (2014) Dropout: a simple way to prevent neural networks from overfitting *The Journal of Machine Learning Research* **15 1929-1958**