Optimize the Performance of the Neural Network by Using a Mini Dataset Processing Method

Jingliang Chen[1], Chenchen Wu[1], Shuisheng Chen[1], Yi Zhu[2], Bin Li[3]*

[1] School of Computer Science, Hubei University of Technology, China
[2] School of Transportation & Information, Hubei Communications Technical College, China
[3] China Railway Fifth Survey and Design Institute Group Co., Ltd, China
chen@hbut.edu.cn, wuhbut@foxmail.com,
chenshuisheng@hbut.edu.com, zhuyi22250@163.com, mountjac@foxmail.com

Abstract

In the case of traditional methods such as network models and algorithms are highly open source and highly bound to hardware, data processing has become an important method to optimize the performance of neural networks. In this paper, we combine traditional data processing methods and propose a method based on the mini dataset which is strictly randomly divided within the training process; and takes the calculation results of the cross-entropy loss function as the measurement standard, by comparing the mini dataset, screening, and processing to optimize the deep neural network. Using this method, each iteration training can obtain a relatively optimal result, and the optimization effects of each time are integrated to optimize the results of each epoch. Finally, in order to verify the effectiveness and applicability of this data processing method, experiments are carried out on MNIST, HAGRID, and CIFAR-10 datasets to compare the effects of using this method and not using this method under different hyper-parameters, and finally, the effectiveness of this data processing method is verified. Finally, we summarize the advantages and limitations of this method and look forward to the future improvement direction of this method.

Keywords: Neural networks, Data processing, Mini dataset, Cross-entropy loss function

1 Introduction

In simple terms, deep learning is the imitation of neurons in the form of a layer to get the data characteristics and store features in an artificial neural network, and the difference is that the network is a black box [1], just like the human brain. Deep learning models are mainly divided into two neural networks: CNN and RNN. CNN involves convolutional computation and is usually used to process data with grid-like topology like images. On the other hand, the RNN model processes and makes decisions based on the information calculated [2].

Generally, we can improve the performance of the deep neural network from four aspects. The first: improve performance through data. Such as acquiring more data, which is the most direct and efficient way. In addition, we can scale the existing data: transform, feature selection, noise processing, reframe, and so on. The second approach is to improve the structure of the model. Such as improving performance through nested models that combine multiple "good enough" models to achieve excellent predictive power, or replacing simple neuron units with complex LSTM neurons, for example, using LSTM models to exploit the advantages of grammar analysis [3-4]. The third is by adjusting the parameters of performance improvement, such as the initialization [5] of an improved model, to ensure that the early gradient has a large number of sparse, or take advantage of the principle of linear algebra [6], to initialize the learning rate, the size of batch size, regularization coefficient, dropout coefficient. The final method is to choose a more robust learning algorithm, such as a way of updating the logarithm gradient [7] or dividing the previous gradient L2 norm to update all parameters, using a Nonlinear time-delay system [8], or even choosing a second-order algorithm [9] with high computational cost.

However, the neural network will be in the process of learning because of various reasons: noise data [10], data initialization [11], and other problems, the final training results and the predicted results are far from the results, and the loss is too significant. Therefore, we need to optimize the model constantly. For example, combining automatic regression integration [12], and stabilizing fuzzy relations [13]. One common way to optimize is to use optimizers, a common form of optimizing the optimizer. During most of the same training time, the process of training and learning, especially small mill are often unable to get a large amount of data and also does not have a powerful machine, so we need to under the condition of the same data set, further pursue the convergence rate, namely under the same time, relatively higher precision and lower loss.

So far, the existing data processing [14] techniques related to this paper, such as data augmentation, normalization, regularization, whitening, etc., are all the integral operations performed on the data set before the data is put into the training, such as data inversion, [0.1] value range scaling, variance normalization, so this paper proposes a new method: Limit Comparison Training in Literation (LCT), which is rooted in the first method of the above four categories of performance improvement methods. On the basis of different optimizers, the attention of optimization is focused on the

*Corresponding Author: Bin Li; E-mail: mountjac@foxmail.com
DOI: 10.53106/160792642023112406005

results of each round inside each batch and each time. The quality of each result is measured through the loss function, and the bad effects are traced to the training process that brought the result. The main contributions are as follows:

- This paper proposes a framework for optimizing the training performance of deep neural networks.
- In this framework, the LCT method is proposed to select the relatively optimal mini data set in each epoch.
- The loss is selected as the standard to measure the quality of the model, and the loss function is determined as the cross-entropy function of multi-classification.
- The effectiveness of the LCT method for optimizing deep neural networks is verified for three different datasets.

The remainder of this paper is as follows: In section 2, we discuss the related work of this paper. In Section 3, the background of the proposed LCT method and the principle and framework of the LCT method are described in detail. The fourth part describes the experimental verification of the LCT method, the experimental process, parameter settings, and data results. Finally, in Section V, we discuss and summarize the LCT method and give some prospects.

2 Related Work

This work mainly involves three aspects: (1) The role of gradient optimizer in neural network training; (2) Cross-entropy loss function.

2.1 Deep Learning Optimization

1) SGD

SGD is also called mini-batch Gradient Descent [15]. Generally, under different systems of current deep learning, the random Gradient Descent method is named the mini-batch Gradient Descent method by failure. Mini-batch Gradient Descent means the training dataset is segmented with sizes such as 32, 64, 128. At each literation, a part of the segmented dataset is selected for the Gradient Descent of the model, the update rule is:

$$\theta_i = \theta_i - \alpha \sum_{j=t}^{t+x-1} (y_j - h(x_j)) x_{ji.} \tag{1}$$

In the current development process of deep learning, Hinton [16], Xie [17], and Horvath [18], all verified the effectiveness of Mini-batch Gradient Descent in their papers.

2) Adaptive Moment Estimation (Adam)

Chen proposed the Adaptive Moment Estimation algorithm for nonstationary temperature problem [19], which is a combination of the Moment algorithm and RMSProp algorithm [20].

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}. \tag{2}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}. \tag{3}$$

These are then used to update the parameters, resulting in Adam's update rule:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \varepsilon} \hat{m}_t. \tag{4}$$

In 2017, Richard Socher [21] found that even though Adam has gotten better over countless training runs, researchers found that after testing on the CIFAR-10 dataset. The convergence speed of Adam is faster than that of SGD (Mini-batch GD), but the final convergence result is not as good as SGD, which also leads to the fact that at present, the frequency of neural network models with high requirements for absolute accuracy is higher than that of Adam compared with SGD.

2.2 Cross-entropy Loss Function

The difference between the predicted value of a machine learning model and the actual value for a single sample is called the loss. The smaller the loss, the better the model.

In machine learning, we want the predicted value to be infinitely close to the actual value, so we need to minimize the difference, and in this process, we need to introduce a loss function. In specific projects, some loss functions decrease the gradient of the difference calculated by the gradient fast, while some decrease slowly. Currently, the commonly used loss functions in neural networks include 0-1 loss, Logistics loss, Hinge loss [22], exponential loss, and cross-entropy loss [23].

Cross-entropy loss functions are divided into binary cross-entropy loss functions based on sigmoid and multi-class cross-entropy loss functions based on Softmax. In the case of binary classification, there are only two cases that the model needs to predict. For each class, the probability of our prediction is ρ and 1-ρ. Our function expression is:

$$L = -\frac{1}{N} \sum_i [y^i \log(\rho^i) + (1-y^i) \log(1-\rho^i). \tag{5}$$

In this expression, y^i is the label, and ρ^i is the predicted value.

In the case of multi-class classification, the function expression is:

$$L = \frac{1}{N} \sum_i L_i = -\frac{1}{N} \sum_i \sum_C^m y_{ic} \log(\rho_{ic}). \tag{6}$$

In this expression, M denotes the number of categories, y_{ic} denotes the label of category C, and ρ_{ic} is the predicted value of category C.

3 Methodology

This section describes the research problem, principle, and implementation framework of the LCT method in detail.

3.1 Problem Model

Take the ANN neural network for solving multi-classification problems as an example in Figure 1:

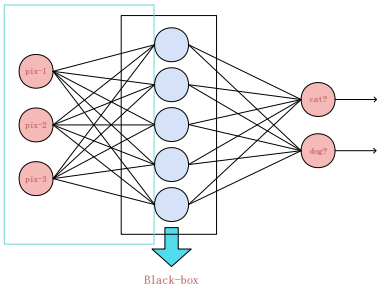


Figure 1. ANN neural network

As we can see from the model graph, we cannot optimize the middle part of the training very well. In the blue wire frame data processing, input, and so on, the function of the black box is actually a portion of the cross. We can see that this part of the analysis ignores the incomprehensible calculation process, starting with the final results.

3.2 Principle Frame

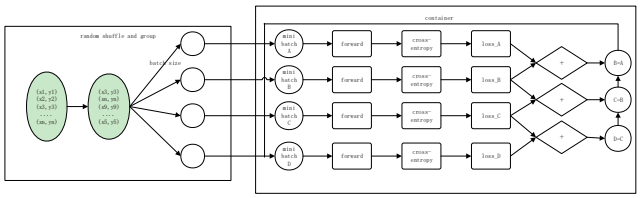


Figure 2. Frame of LCT

As mentioned above in Figure 2, the core idea of the LCT method is: we will focus our attention on dataset optimization on the training data of each literation, actively focus our attention on this "micro-field," and then add a data processing layer to it. The main work of this data processing layer is to train the data of each literation during the training process. The MINI dataset with relatively minor noise is selected, and finally, the selected data is used to retrain and learn.

From the perspective of the internal training process, it is not only feasible, but also does not conflict with previous data processing techniques. Theoretically, it can be applied to any dataset with different optimization algorithms.

To reach this end, we solve the following two problems: 1. How to conduct data processing and select the relatively optimal MINI data set; 2. How to train and learn after the MINI data is processed.

1. How to conduct data processing and select the relatively optimal MINI data set:

The main difficulty lies in getting rid of the process constraints of the framework and finding the MINI dataset and the calculation result of the cross-entropy function originally encapsulated in each literation training process inside the package.

The above model diagram shows that the LCT method's main functional modules are divided into two parts: the random shuffling grouping module and the MINI dataset processing module.

In the random shuffling grouping module, we add random seeds in the processing part of the macro data set and then shuffle our dataset. Informally, an epoch is a process of training all training examples through the model once. The primary purpose of setting epochs is to divide the whole training process of the model into several segments so that we can better observe and adjust the model training.

When the number of samples (i.e., all training samples) for an epoch may be too large (for a computer), we need to split it into smaller chunks, i.e., batches, for training. Each round of network parameter optimization needs to use a batch. Compared with a single sample, a batch of data can better simulate the distribution of the data set. The larger the batch is, the better the simulation of the input data distribution, which is reflected in the network training. It can make the direction of network training "more correct".

$$seed = epoch. \tag{7}$$

$$num\ of\ \ batches = \frac{training\ \ dataset\ \ size}{batch\ \ size}. \tag{8}$$

The MINI dataset processing module is the core module of our LCT method. In this module, we first input the MINI data set into the network and select the cross-entropy loss function to calculate the loss corresponding to each literation. Record the obtained loss with weight parameters and other hyper-parameters. Secondly, the loss value obtained from MINI dataset A is compared with the loss value obtained from Mini dataset B. When the loss value obtained from MINI dataset A is greater than the loss obtained from MINI dataset B, this means that MINI dataset B is better than MINI dataset A. Replace the contents of MINI dataset A with the contents of MINI dataset B. Finally, this operation is repeated until the end of the epoch of this round, when all literation MINI datasets of this round have been processed and transformed. The formula is as follows:

$$\begin{aligned} minibatch = \{ & l_{pre} < \\ & l\,?\,minibatch_{pre} : minibatch\}. \end{aligned} \tag{9}$$

Limit comparison training

Require: Global learning rate r, momentum coefficient α
Require: The exponential decay rates of moment estimation, ρ1, ρ2,
Require: Mini batch size and epoch size. The mini-batch size is usually 32, 64, 128, ... , default values: 64, 10000
Initialize the weight parameters and optimizer
Initialize time step t = 0
If not, proceed to the next step
Step 1: Take samples from mini training set (x1, x2, x3, x4…, xn) and the corresponding tags (y1, y2, y3, y4…, yn),
Step 2: forward propagation and obtain correct loss and accuracy of this time,
Step 3: Compare the loss of this time with the pre-loss of last time.
Step 4: If the current loss is not decreased compared with the previous loss, replace the current data set with the previous data set and jump to step 2: batch pre-batch

Step 5: Record the current loss and accuracy and store them in the array
The end

2. How to instruct and master after the procedure:

Returning to the data information pace of the neural network, the procedure MINI data is re-input into the network, and the previous record nodules are called on at the same time. All the weight and hyper-parameters such as W, B, L, Grads, etc. are restored to the pre-training state. Then, the current network is re-trained.

4 Experiments

4.1 Dataset

This paper uses three open source datasets, which are the MNIST dataset, HaGRID dataset and CIFAR-10 dataset.

1) MNIST Dataset

The MNIST dataset was initiated by NIST. In 1998, Yan LeCun used the MNIST dataset, as shown in Figure 3, which has 60,000 images and labels in the training set and 10,000 in the test set.

Figure 3. MNIST

2) CIFAR-10 Dataset

CIFAR-10 is a small dataset for identifying pervasive objects curated by Hinton's students Alex Krizhevsky and Ilya Sutskever as shown in Figure 4. There are 50,000 training images and 10,000 test images. Compared with the above two datasets, CIFAR-10 contains natural objects in the real world, which are not only noisy but also have different proportions and features, leading to great difficulties in recognition.

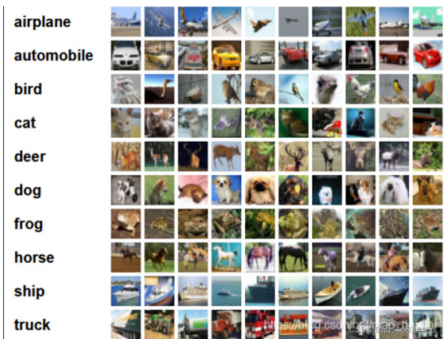


Figure 4. CIFAR-10

3) HaGRID Dataset

The hand Gesture Recognition Image Dataset (HaGRID) dataset size is 716GB and contains 552,992 Full HD (1920 × 1080) RGB images, divided into 18 categories of gestures. The data is split into 92% training set and 8% test set, where 509,323 images are used for training and 43,669 images for testing.

4.2 Performance Comparison

1) LCT + MNIST Dataset

Batch size1=32, batch size2=64, batch size3=100, learning rate: $\eta = 7 * 10^{-4}$ as shown in Figure 5 to Figure 10.

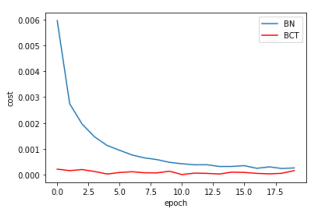


Figure 5. Loss of SGD (size=32)

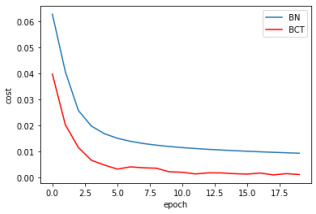


Figure 6. Loss of SGD (size=32)

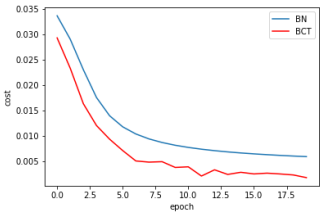


Figure 7. Loss of SGD (size=64)

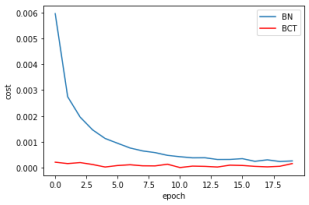


Figure 8. Loss of Adam (size=64)

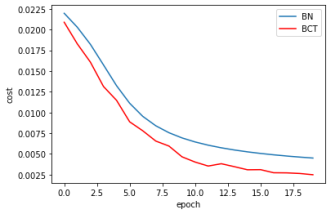


Figure 9. Loss of SGD (size=100)

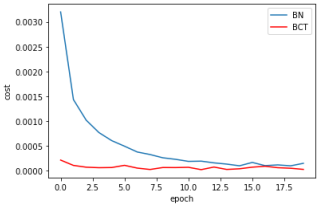


Figure 10. Loss of Adam(size=100)

2) LCT + CIFAR-10 Dataset

Batch size1=32, batch size2=64, batch size3=100, learning rate: $\eta = 7 * 10^{-4}$ as shown in Figure 11 to Figure 16.

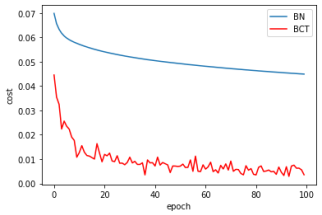


Figure 11. Loss of SGD (size=32)

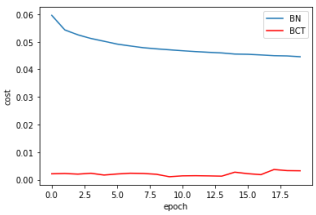


Figure 12. Loss of SGD (size=32)

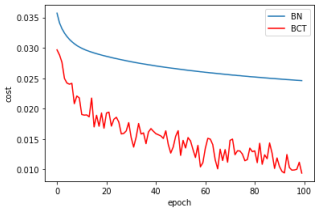


Figure 13. Loss of SGD (size=64)

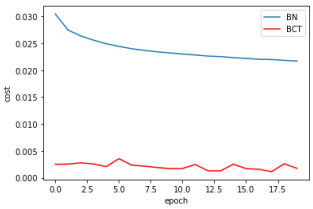


Figure 14. Loss of Adam (size=64)

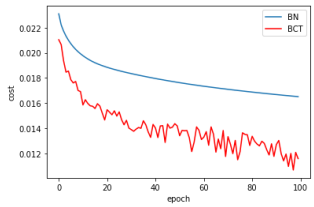


Figure 15. Loss of SGD (size=100)

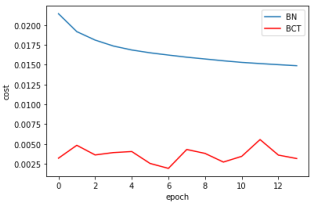


Figure 16. Loss of Adam (size=100)

3) LCT + HaGRID Dataset

The HaGRID dataset has more samples and richer classes than other datasets. In this section, the deep learning model is finally trained and validated using the HaGRID dataset, while using more training batches and strictly controlling variables. In this case, the feasibility of the LCT method in HaGRID is verified, and the credibility of relevant experiments is finally verified.

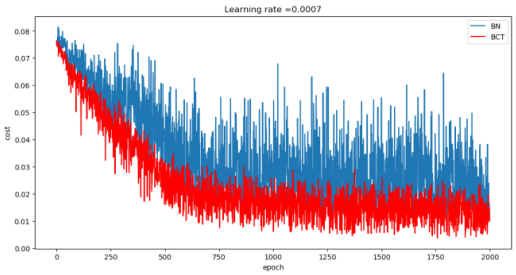


Figure 17. Loss of SGD (size=32)

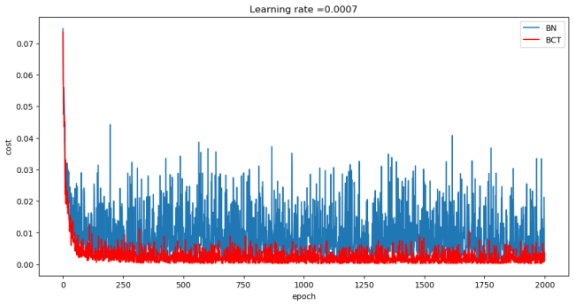


Figure 18. Loss of Adam (size=32)

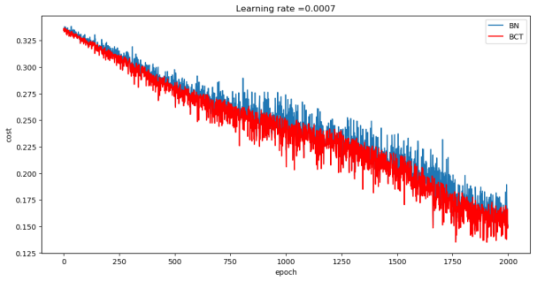


Figure 19. Loss of SGD (size=100)

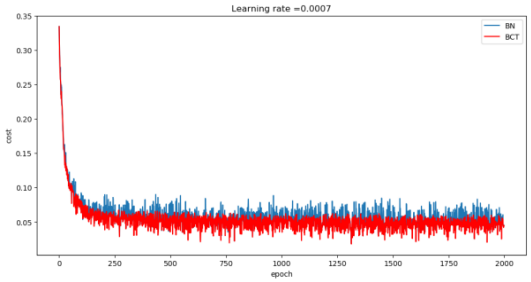


Figure 20. Loss of Adam (size=100)

Under two different optimization algorithms, the SGD optimization algorithm and the Adam optimization algorithm, it can be found that the training effect of the neural network model under the HaGRID dataset is common as shown in Figure 17. The results of using only the normalization method and using the LCT method from the beginning are basically the same. At the same time, when the batch size=32, it can be found that the data fluctuation of the model using the LCT method in the training process is much smaller than that of the model using normalization only as shown in Figure 18. When the batch size=100, the data fluctuations of both are basically the same. This speed is lower than the Adam optimizer model as shown in Figure 19 and Figure 20.

4.3 Experimental Results

After the completion of the experiment, we counted the loss results of all comparative tests and calculated the prediction accuracy of the model. The precision is calculated as follows:

$$accuracy = \frac{\sum_i^n \{y^i = \rho^i \,?1:0\}}{n}. \tag{10}$$

1) Performance of MNIST

Firstly, the MNIST dataset was used for the experiment, and the test was carried out under the learning rate of 0.0007. At the same time, the batch sizes of 32, 64, and 100 training corresponded, and the epoch was fixed 10 times. After the training, the final results were compared through the graphical structure as shown in Table 1 to Table 3.

Table 1. The effect of batch size = 32

	SGD	SGD (LCT)	Adam	Adam (LCT)
Loss	13.65	3.71	4.52	0. 79
Accuracy	73.28%	84.29%	87.70%	95.90%

Loss optimization effect:

SGD: 9.94——72.82% Adam: 3.73——82.52%

Accuracy optimization effect:

SGD: 11.01% Adam: 8.2%

Table 2. The effect of batch size = 64

	SGD	SGD (LCT)	Adam	Adam (LCT)
Loss	10.13	5.92	2.51	0. 30
Accuracy	68.51%	74.88%	90.1%	93%

Loss optimization effect:

SGD: 4.21——41.56% Adam: 2.21——88.05%

Accuracy optimization effect:

SGD: 6.37% Adam: 2.9%

Table 3. The effect of batch size = 100

	SGD	SGD (LCT)	Adam	Adam (LCT)
Loss	4.49	2.48	1.75	0. 66
Accuracy	76.06%	82.49%	92.93%	97.79%

Loss optimization effect:
SGD: 2.01——47.77% Adam: 1.09——62.28%
Accuracy optimization effect:
SGD: 6.43% Adam: 4.86%

2) Performance of CIFAR-10

Second, CIFAR-10 was used for experiments, and the learning rates of 0.001 and 0.0007 were tested. At the same time, it corresponded to different batch sizes of 32, 64, and 100, and the epoch was fixed 20 times as shown in Table 4 to Table 6.

Table 4. The effect of batch size = 32

	SGD	SGD (LCT)	Adam	Adam (LCT)
Loss	5.43	0.89	4.46	0. 32
Accuracy	36.75%	58.49%	45.56%	71.46%

Loss optimization effect:
SGD: 4.54——83.61% Adam: 4.14——92.83%
Accuracy optimization effect:
SGD: 21.74% Adam: 25.9%

Table 5. The effect of batch size = 64

	SGD	SGD (LCT)	Adam	Adam (LCT)
Loss	2.87	1.67	2.17	0. 17
Accuracy	34.4%	56.7%	47.52%	62.28%

Loss optimization effect:
SGD: 1.2——41.81% Adam: 2——92.17%
Accuracy optimization effect:
SGD: 22.3% Adam: 14.76%

Table 6. The effect of batch size = 100

	SGD	SGD (LCT)	Adam	Adam (LCT)
Loss	1.84	1.34	1.53	0. 29
Accuracy	34.03%	44.18%	43.44%	50.98%

Loss optimization effect:
SGD: 0.5——27.17% Adam: 1.24——81.05%
Accuracy optimization effect:
SGD: 10.15% Adam: 7.54%

3) Performance of HaGRID

Finally, the experiment was carried out using a gesture dataset (HaGRID), and the test was carried out at a learning rate of 0.0007 and 0.001. At the same time, it corresponded to two different batch sizes of 32 and 100, and the epoch was fixed 20,000 times as shown in Table 7 and Table 8.

Table 7. The effect of batch size = 32

	SGD	SGD (LCT)	Adam	Adam (LCT)
Loss	1.53	0.73	1.51	0.31
Accuracy	86.67%	91.67%	94%	96.67%

Loss optimization effect:
SGD: 0. 8——52.28% Adam: 1.2——79.47%
Accuracy optimization effect:
SGD: 5% Adam: 2.67%

Table 8. The effect of batch size = 100

	SGD	SGD (LCT)	Adam	Adam (LCT)
Loss	1.83	1.56	0.75	0.56
Accuracy	83.56%	86.73%	92.1%	94.33%

Loss optimization effect:
SGD: 0.27——14.75% Adam: 0.19——25.33%
Accuracy optimization effect:
SGD: 3.17% Adam: 2.23%

5 Discussion and Conclusion

From the above training images and validation charts, we will summarize all the results of using LCT in Table 9 and Table 10:

Table 9. The loss of LCT

	MNIST%	CIFAR-10%	HaGRID%
SGD (32)	72.85	83.61	52.28
SGD (64)	41.56	41.81	14.75
SGD (100)	47.77	27.17	
Adam (32)	82.52	92.83	79.47
Adam (64)	88.05	92.17	25.33
Adam (100)	62.28	81.05	

Table 10. The accuracy of LCT

	MNIST%	CIFAR-10%	HaGRID%
SGD (32)	11.01	21.74	5
SGD (64)	6.37	22.3	3.17
SGD (100)	6.43	10.45	
Adam (32)	8.2	25.9	2.67
Adam (64)	2.9	14.76	2.23
Adam (100)	4.86	7.54	

As can be seen in Table 9 and Table 10, no matter what kind of dataset is used or what optimizer is selected, the neural network model using the LCT method has a positive improvement compared with the model using only data normalization operation. Simarly, the comprehensive convergence efficiency of LCT method is always higher. From the final data, the average optimization effect of loss in the MNIST data set is 65.85%, the average optimization effect of loss in the CIFAR-10 data set is 69.78%, and the average optimization effect of loss in the HaGRID data set is 42.96%. The average improvement for accuracy is MNIST: 6.63%; CIFAR- 10:17.12%; HaGRID: 3.27%. At the same time, it is also demonstrated that this method does not rely on some rare qualification conditions. As long as the model has the universal structure of epoch, batch, and literation, this method can be used and has universal applicability.

However, the LCT method also has some limitations. Firstly, the LCT method relies heavily on the calculation result of the loss function as the measurement standard for the core operation, so the limitations of the loss function become the limitations of the LCT method. Secondly, when the LCT method obtained the relatively highest quality mini dataset, although the quality of the training data in each epoch was improved, part of the sample characteristics were also reduced. Finally, the LCT method belongs to the data

processing method inside the training process. The more original dataset samples there are, the longer the additional data processing time inside the training will be, and the more the total time cost will increase.

In future work, we will actively map out an optimal MINI dataset based on literation and combine it with the hybridised loss function to obtain more accurate loss and achieve better results.

Acknowledgements

This work is supported by the Key Project of Hubei Education Department under Grant No. D20191406 & D20201402; the Teaching Research Project of Hubei Education Department under Grant No. 2021295 & 2022286; the Science Start-up Foundation for High-level Talents of HBUT under Grant No. 430100391 & 337396; the Research and Demonstration of Digital Technology for Urban Renewal and Future Community Development sponsored by China Railway Construction Corporation Limited.

References

[1] O. S. Bayomie, R. Cerqueira, L. Neuendorf, I. Kornijez, S. Kieling, T. H. Sandermann, K. Lammers, N. Kockmann, Detecting flooding state in extraction columns: Convolutional neural networks vs. a white-box approach for image-based soft sensor development, Computers & Chemical Engineering: *An International Journal of Computer Applications in Chemical Engineering,* Vol. 164, Article No. 107904, August, 2022.

[2] S. Aggarwal, S. Sehgal, Text Independent Data-level Fusion Network for Multimodal Sentiment Analysis, *International Journal of Performability Engineering*, Vol. 18, No. 9, pp. 605-612, September, 2022.

[3] R. Socher, C. Y. Lin, A. Y. Ng, C. D. Manning, Parsing Natural Scenes and Natural Language with Recursive Neural Networks, *Proceedings of the 28th International Conference on Machine Learning*, ICML-2011 Bellevue, Washington, USA, 2011, pp. 129-136.

[4] S. Alagarsamy, V. James, RNN LSTM-based Deep Hybrid Learning Model for Text Classification using Machine Learning Variant XGBoost. *International Journal of Performability Engineering*, Vol. 18, No. 8, pp. 545-551, August, 2022.

[5] Q. V. Le, N. Jaitly, G. E. Hinton, *A Simple Way to Initialize Recurrent Networks of Rectified Linear Units*, Computer Science, arXiv preprint arXiv:1504.00941 April, 2015, https://arxiv.org/abs/1504.00941.

[6] A. Takekawa, M. Kajiura, H. Fukuda, Role of Layers and Neurons in Deep Learning With the Rectified Linear Unit, *Cureus*, Article No. e18866, Vol. 13, No. 10, October, 2021

[7] J. C. Duchi, E. Hazan, Y. Singer, Adaptive Subgradient Methods for Online Learning and Stochastic Optimization, *Journal of Machine Learning Research*, Vol. 12, pp. 2121-2159, July, 2011

[8] Y.-X. Duan, Z.-Y. Sun, B.-L. Su, Optimisation control via the distributed model predictive method for nonlinear time-delay systems, *International Journal of Systems Science*, Vol. 51, No. 16, pp. 3339–3346, 2020.

[9] R. Feng, Y. Liu, Y. Hou, H. Li, Z. Fang, Mixed element algorithm based on a second-order time approximation scheme for a two-dimensional nonlinear time fractional coupled sub-diffusion model, *Engineering with Computers*, Vol. 38, pp. 51-68, February, 2022.

[10] S. Herawati, Y. D. P. Negara, M. Latif, Complete ensemble empirical mode decomposition with adaptive noise integrating feedforward neural network for tourist arrival forecasting, *Journal of Physics: Conference Series, IOP Publishing*, Vol. 2193, No. 1, Article No. 012049, 2022.

[11] J. Liu, Y. Liu, Q. Zhang, A weight initialization method based on neural network with asymmetric activation function, *Neurocomputing*, Vol. 483, pp. 171-182, April, 2022.

[12] Z. S. Khozani, F. B. Banadkooki, M. Ehteram, A. N. Ahmed, A. El-Shafie, Combining autoregressive integrated moving average with long short-term memory neural network and optimisation algorithms for predicting ground water level, *Journal of Cleaner Production*, Vol. 348, Article No. 131224, May, 2022.

[13] Z. Wang, S.-K. Oh, W. Pedrycz, E.-H. Kim, Z. Fu, Design of stabilized fuzzy relation-based neural networks driven to ensemble neurons/layers and multi-optimization, *Neurocomputing*, Vol. 486, pp. 27-46, May, 2022.

[14] F. A. Freitas, R. M. Jafelice, J. W. da Silva, D. de S. Rabelo, Q. S. S. Nomelini, J. dos R. V. de Moura, C. A. Gallo, M. J. da Cunha, J. E. Ramos, A new data normalization approach applied to the electromechanical impedance method using adaptive neuro-fuzzy inference system, *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, Vol. 43, No. 11, pp. 1-13, November, 2021.

[15] J. L. Chen, Metrological Software Test for Simulating the Method of Determining the Thermocouple Error in Situ During Operation, *Measurement Science Review*, Vol. 18, No. 2, pp.52-58, 2018

[16] G. E. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, B. Kingsbury, Deep Neural Networks for Acoustic Modeling in Speech Recognition, The Shared Views of Four Research Groups, *IEEE Signal Processing Magazine*, Vol. 29, No. 6, pp. 82-97, November, 2012.

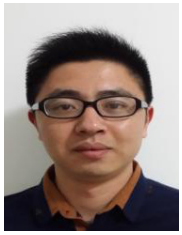
[17] F. Xie, J. Wang, R. Xiong, N. Zhang, Y. Ma, K. He, An Integrated Service Recommendation Approach for Service-based System Development, *Expert Systems with Applications*, Vol. 123, pp. 178-194, June, 2019.

[18] T. Horvath, A. Tomasov, P. Munster, P. Dejdar, V. Oujezsky, Unsupervised Anomaly Detection Using Bidirectional GRU Autoencoder Neural Network for PLOAM Message Sequence Analysis in GPON, *2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, Maldives, Maldives, 2022, pp. 1-5.

[19] J. L. Chen, N. O. Hembara, M. Hvozdyuk,

Nonstationary Temperature Problem for a Cylindrical Shell with Multilayer Thin Coatings, *Materials Science*, Vol. 54, No. 3, pp. 339-348, November, 2018.
[20] D. Xu, S. Zhang, H. Zhang, D. P. Mandic, Convergence of the RMSprop deep learning method with penalty for nonconvex optimization, *Neural Networks*, Vol. 139, pp. 17-23, July, 2021.
[21] N. S. Keskar, R. Socher, *Improving Generalization Performance by Switching from Adam to SGD*, December, 2017, https://arxiv.org/abs/1712.07628.
[22] J. Luo, H. Qiao, B. Zhang, Learning with smooth hinge losses, *Neurocomputing*, Vol. 463, pp. 379-387, November, 2021.
[23] L. Li, T. Zheng, W. Yin, Q. Wu, Novel pythagorean fuzzy entropy and pythagorean fuzzy cross-entropy measures and their applications, *Journal of Intelligent & Fuzzy Systems*, Vol. 41, No. 6, pp. 6527-6546, December, 2021.

Biographies



Jingliang Chen received his Ph.D. degree in Computer Software and theory from Wuhan University in 2016. He is an associate professor in School of Computer Science, Hubei University of Technology. His research interests include software engineering, intelligent computing, and software methodology.

Chenchen Wu was born in Jiangsu, China. His major interests are in the areas of deep learning and training optimization for neural networks.

Shuisheng Chen is a professor in mechanics institute, Hubei University of Technology. His research interests include mechanical manufacturing and automation, mechanical and electronic engineering, industrial robot technology and applications.

Yi Zhu received the PhD degree in computer software from School of Computer Science, Huazhong University of Science and Technology. She is a lecturer in computer network technology, Hubei Communications Technical College. Her research focuses on database security and cloud computing.

Bin Li is currently working as a senior engineer at China Railway Fifth Survey and Design Institute. His research interests include Building Information Modeling, Large Language Model of engineering, and underwater structural defect image recognition.