Hindawi
Security and Communication Networks
Volume 2020, Article ID 5218612, 8 pages
https://doi.org/10.1155/2020/5218612

Research Article

Exploiting the Relationship between Pruning Ratio and Compression Effect for Neural Network Model Based on TensorFlow

Bo Liu,[1] **Qilin Wu**,[1] **Yiwen Zhang**,[2] **and Qian Cao**[1]

[1]*College of Information Engineering, Chaohu College, Chaohu 238000, China*
[2]*School of Computer Science and Technology, Anhui University, Hefei 230000, China*

Correspondence should be addressed to Qian Cao; 19875069@qq.com

Received 12 December 2019; Revised 4 February 2020; Accepted 10 February 2020; Published 30 April 2020

Guest Editor: Xiaolong Xu

Pruning is a method of compressing the size of a neural network model, which affects the accuracy and computing time when the model makes a prediction. In this paper, the hypothesis that the pruning proportion is positively correlated with the compression scale of the model but not with the prediction accuracy and calculation time is put forward. For testing the hypothesis, a group of experiments are designed, and MNIST is used as the data set to train a neural network model based on TensorFlow. Based on this model, pruning experiments are carried out to investigate the relationship between pruning proportion and compression effect. For comparison, six different pruning proportions are set, and the experimental results confirm the above hypothesis.

1. Introduction

Model compression is a common method to transplant artificial intelligence from the cloud to the embedded terminal. Network pruning is a particularly effective compression solution for models [1, 2]. In [1, 3], Han et al. proposed a method of compression based on pruning but did not investigate the relationship between pruning proportion and compression effect. At the same time, He et al. [2] studied channel pruning for accelerating very deep neural networks, yet the pruning rate on the prediction effect is not stated. In fact, some studies of pruning methods have been carried out in recent years. However, to the best of our knowledge, there are very few studies on the relationship between the pruning proportion and the size, accuracy, and computing time which is used to make predictions. It is also the motivation of our research.

In a trained neural network model, pruning sets all parameters with values less than a specific threshold to zero. After pruning, retraining and sparsification are normally conducted, where sparsification can delete connections with the zero values to compress the size of the model [4, 5]. As an example, the two figures show the comparison before and after pruning, where Figure 1 shows the original structural diagram, and Figure 2 shows the structural diagram after pruning.

Here, based on TensorFlow, we will use MNIST as the data set to train a neural network model. TensorFlow is an open-source machine learning framework. Specifically, it is software, and users need to build mathematical models by programming in Python and other languages. These models are used in the application of artificial intelligence. MNIST data set is a handwritten data set with 60,000 handwritten digital images in the training library and 10,000 in the test library. It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting.

In this paper, we make the hypothesis that the pruning proportion is positively correlated with the compression scale of the model but not with the prediction accuracy and calculation time. So, our research object is the preliminary relationship between pruning proportion and compression effect in the neural network model. Specifically, this paper studies the relationship from three aspects: first, the

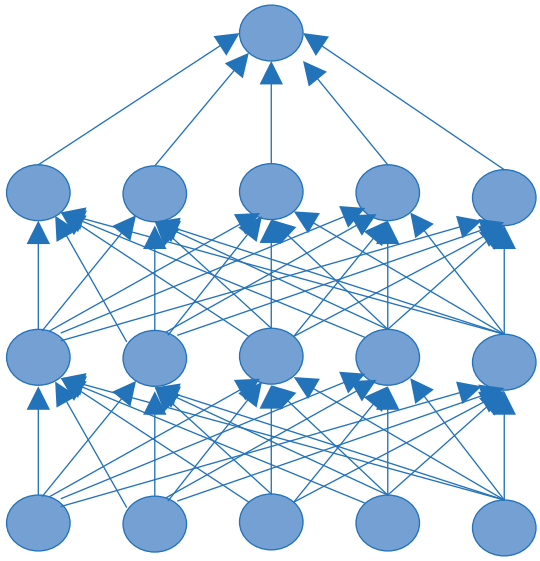

Figure 1: Original structural diagram.

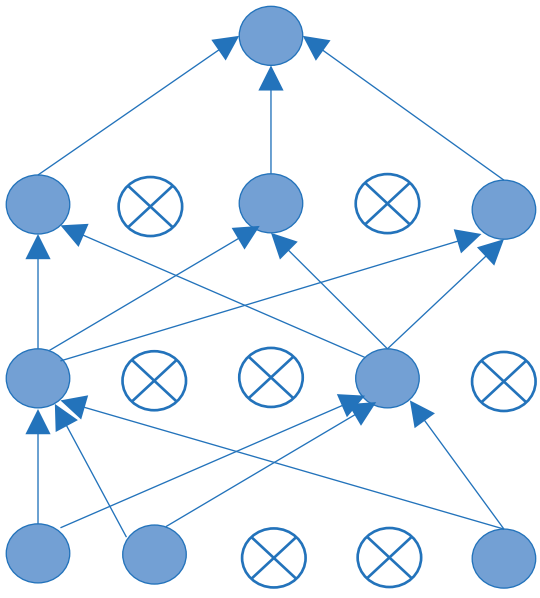

Figure 2: Structural diagram after pruning.

relationship between pruning proportion and model size; second, the relationship between pruning proportion and model prediction accuracy; lastly, the relationship between pruning proportion and computing time for model predictions. For the above objective, a great number of experiments are carried out to investigate the relationship between pruning proportion and compression effect, and the above hypothesis is confirmed, which is our main contribution in this paper.

The rest of this paper is organized as follows. In Section 2, the neural network model is proposed first. To test the hypothesis, an original model and an experimental plan are introduced in Section 3. Section 4 gives the experimental procedures, and Section 5 gives the experimental results and analysis. Finally, Section 6 concludes this paper.

2. Neural Network Model

A neural network is constituted by one input layer, one or several hidden layers, and one output layer, and every layer is constituted by a certain number of neurons. These neurons are interconnected, just like the nerve cells of humans. Figure 3 shows the structure of the neural network.

We assume that $X_i(t) = [x_{i,1}(t), x_{i,2}(t), \ldots, x_{i,D}(t)]$ is the ith individual (solution) in the population. The mutation operator aims to generate mutant solutions. For each solution X_i, a mutant solution V_i is created by the corresponding mutation scheme. There are some classical mutation schemes listed as follows:

(1) DE/rand/1:

$$v_{i,j}(t) = x_{i1,j}(t) + F \cdot \left(x_{i2,j}(t) - x_{i3,j}(t)\right). \tag{1}$$

(2) DE/rand/2:

$$v_{i,j}(t) = x_{i1,j}(t) + F \cdot \left(x_{i2,j}(t) - x_{i3,j}(t)\right) + F \cdot \left(x_{i4,j}(t) - x_{i5,j}(t)\right). \tag{2}$$

(3) DE/best/1:

$$v_{i,j}(t) = x_{\text{best},j}(t) + F \cdot \left(x_{i1,j}(t) - x_{i2,j}(t)\right). \tag{3}$$

(4) DE/best/2:

$$v_{i,j}(t) = x_{\text{best},j}(t) + F \cdot \left(x_{i1,j}(t) - x_{i2,j}(t)\right) + F \cdot \left(x_{i3,j}(t) - x_{i4,j}(t)\right), \tag{4}$$

where $i1$, $i2$, $i3$, $i4$, and $i5$ are five randomly selected individual indices between 1 and N, and $i1 \neq i2 \neq i3 \neq i4 \neq i5$. $F \in [0, 1]$ is usually used. X_{best} is the global best individual (solution).

The crossover operator focuses on recombining two different individuals and creates a new one. In DE, a trial solution U_i is created based on the following crossover operation:

$$\text{swap}(X_{i1}, X_{i2}), \quad \text{if } f(X_{i2}) < f(X_{i1}), \tag{5}$$

where CR is called the crossover rate, the random value rand_j is in the range [0, 1], and j_r is a randomly selected dimension index. As seen, U_i inherits from V_i and X_i based on the value of CR. For a large CR, most dimensions of U_i are taken from V_i. For a small CR, most dimensions of U_i are taken from X_i. For the latter case, U_i is similar to its parent X_i.

3. Design of the Experiment

3.1. Structure of the Original Model. The basic neural network structure consists of the following layers in sequence: convolutional layer, pooling layer, convolutional layer, pooling layer, and two fully connected layers [6, 7], which is shown in Figure 4. In the experiment plan, pruning is performed by default on the weight parameters w of the two fully connected layers. Alternative pruning is performed on

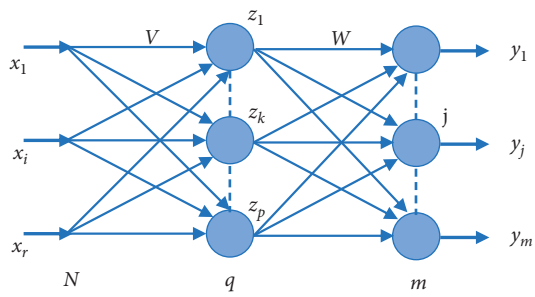

Figure 3: Structure of the neural network model.

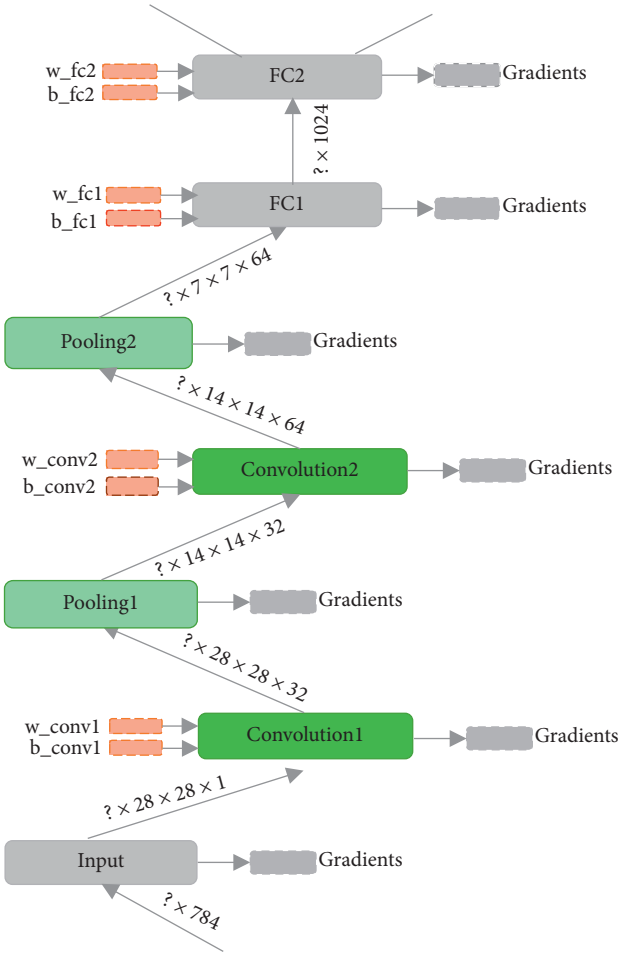

Figure 4: Basic structure of the neural network of the experiment.

all network parameters, and the specific operations are executed by changing the command line parameters [8, 9].

3.2. Experiment Plan. The experiment is based on the TensorFlow framework and used MNIST as the dataset. An original model is trained in the beginning, and then six pruning practices with different pruning proportions are employed [10, 11]. For each pruning, retraining and sparsification are subsequently performed. When all three operations are completed on the original model, the task of pruning compression is also finished [12, 13]. Then, the data are collected and analysed for comparison (size, accuracy, and computing time for making predictions).

4. Experimental Procedures

4.1. Run Command of the Pruning Experiment. Model pruning is executed by the following command: python train.py −1 −2 −3 --train_data_dir /tmp/mnist_data --train_dir /tmp/mnist_train --variables_dir /tmp/mnist_-variables --max_steps 10000 --batch_size 32 --sparse_ratio 0.9 --pruning_variable_names w_fc1, w_fc2. Table 1 specifies the parameters in this command [14–16].

4.2. Pruning Effect View Command. The effects of the −1 or −2 parameters can be viewed through eval_-predict_with_dense_network.py. The specific command is python eval_predict_with_dense_network.py --test_data_-dir /tmp/mnist_data --checkpoint_dir /tmp/mnist_train/step_2_2 --batch_size 32 --max_steps 10. Table 2 specifies the parameters in this command [17–19].

The effect of −3 sparsification can be viewed through eval_predict_with_sparse_network.py. The specific command is python eval_predict_with_sparse_network. py --test_data_dir /tmp/mnist_data --checkpoint_dir /tmp/mnist_train/step_3 --batch_size 32 --max_steps 10. Table 3 specifies the parameters in the command.

4.3. Hardware and Software Configuration of the Experiment. Pruning experiments are based on the following hardware and software parameters and versions [20, 21]:

Operating system: Windows 10

GPU: NVIDIA GeForce GTX 1080 Ti 11.0 G

CPU: Intel(R) Core(TM) i3-4160 CPU @ 3.60 GHz

Memory: 16.0 GB DDR3

Disk: Lenovo SSD SL700 240G

Software: TensorFlow-GPU 1.5.0, Python 3.6

4.4. Construction of Experimental Environment. The experiment is based on the MNIST dataset and the TensorFlow framework. The experimental environment was constructed by the following three steps [22–24]:

Step 1: constructing the Python environment. Directly following Anaconda and then directly adding and running Anaconda3-4.3.1-Windows-x86_64.exe.

Step 2: constructing the plug-ins of NVIDIA GPU. Directly running the cuda_9.0.103_win10.exe for installation. Unzipping cudnn-9.0-windows10-x64-v7.zip and copying its contents to the folder C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.0.

Step 3: constructing the TensorFlow environment. Executing the installation command on the CMD commands: pip install tensorflow-gpu = = 1.5.0.

Table 1: Descriptions of parameters of the pruning command.

Parameter	Description	Default value
−1	Most basic model training	
−2	Pruning is performed on the most basic model	
−3	Pruning is performed, and network parameters of sparsification are stored	
--train_data_dir	Folder path containing training data	/tmp/mnist_data
--train_dir	Log data during training and folder path containing model data	/tmp/mnist_train
--variables_dir	Folder containing data of the last trained model	/tmp/ mnist_variables
--max_steps	Number of iterations of the basic model	10,000
--batch_size	Sample data size of each batch during model training	32
--sparse_ratio	Percentage of compression parameter of pruning/volume of parameters set to 0	0.9
--pruning_variable_names	Parameters on which pruning could be performed. Optional parameters include w_conv1, w_conv2, w_fc1, and w_fc2	w_fc1, w_fc2

Table 2: Descriptions of parameters of the pruning effect view command (view effects of −1 or −2 training).

Parameter	Description	Default value
--test_data_dir	Folder path containing test data	/tmp/mnist_data
--checkpoint_dir	Folder path of the trained model	/tmp/mnist_train/step_2_2
--variables_dir	Folder containing data of the last trained model	/tmp/mnist_variables
--max_steps	Number of iterations during model testing	10
--batch_size	Size of sample data of each batch used for calculation during model testing	32

Table 3: Descriptions of parameters of the pruning effect view command (view effects of −3 sparsification).

Parameter	Description	Default value
--test_data_dir	Folder path containing test data	/tmp/mnist_data
--checkpoint_dir	Folder path of the trained model	/tmp/mnist_train/step_3
--variables_dir	Folder containing data of the last trained model	/tmp/mnist_variables
--max_steps	Number of iterations during model testing	10
--batch_size	Size of sample data of each batch used for calculation during model testing	32

5. Experimental Results and Analysis

In this section, six different pruning proportions are employed in this experiment. The six groups of tables show the specific data of pruning proportion, model size, accuracy, and computing time for predictions.

5.1. 10% Pruning Proportion. First, the pruning proportion is set to 10%; Table 4 shows the parameters of pruning effect in the first scene. In this group of experiments, the parameter threshold values of the two fully connected layers are set to 0.012034996 and 0.013038448. In this way, the valid parameter numbers are reduced from 3,211,264 and 10,240 to 2,890,137 and 9,215, respectively, making exactly 10% of the parameter values of the two fully connected layers equal to 0. However, the model size after the pruning, retraining, and sparsification is 66.5 M, which is larger than the size (37.5 M) of the original model. Hence, no compression effect is achieved. In addition, compared with the original model, the accuracy does not change, and the computing time for predictions slightly increases [25, 26].

Table 4: Descriptions of parameters of the pruning effect in the first scene.

	Original number of parameters	Threshold value	Valid parameter number
w_conv1	800	0.0	800
w_conv2	51,200	0.0	51,200
w_fc1	3,211,264	0.012034996	2,890,137
w_fc2	10,240	0.013038448	9,215

5.2. 30% Pruning Proportion. Second, the pruning proportion is set to 30%; Table 5 shows the parameters of pruning effect in the first scene. In this group of experiments, the parameter threshold values of the two fully connected layers are set to 0.036936015 and 0.039559085. In this way, the valid parameter numbers are reduced from 3,211,264 and 10,240 to 2,247,884 and 7,167, respectively, making exactly 30% of the parameter values of the two fully connected layers equal to 0. However, the model size after the pruning, retraining, and sparsification is 51.8 M, which is larger than the size (37.5 M) of the original model. Hence, no compression effect was achieved. Again, compared with the

Table 5: Descriptions of parameters of the pruning effect in the second scene.

	Original number of parameters	Threshold value	Valid parameter number
w_conv1	800	0.0	800
w_conv2	51,200	0.0	51,200
w_fc1	3,211,264	0.036936015	2,247,884
w_fc2	10,240	0.039559085	7,167

Table 6: Descriptions of parameters of the pruning effect in the third scene.

	Original number of parameters	Threshold value	Valid parameter number
w_conv1	800	0.0	800
w_conv2	51,200	0.0	51,200
w_fc1	3,211,264	0.06429165	1,605,631
w_fc2	10,240	0.068891354	5,119

original model, the accuracy does not change, and the computing time for predictions slightly increases.

5.3. 50% Pruning Proportion. Third, the pruning proportion is set to 50%; Table 6 shows the parameters of pruning effect in the first scene. In this group of experiments, the parameter threshold values of the two fully connected layers are set to 0.06429165 and 0.068891354. In this way, the valid parameter numbers are reduced from 3,211,264 and 10,240 to 1,605,631 and 5,119, respectively, making exactly 50% of the parameter values of the two fully connected layers equal to 0. The model size after pruning, retraining, and sparsification is 37.1 M, which is slightly smaller than the size (37.5 M) of the original model. Here, compression takes effect. Besides, both accuracy and computing time for predictions slightly decrease as compared with those of the original model.

5.4. 70% Pruning Proportion. Fourth, the pruning proportion is set to 70%; Table 7 shows the parameters of pruning effect in the fourth scene. In this group of experiments, the parameter threshold values of the two fully connected layers are set to 0.09749276 and 0.10360378. In this way, the valid parameter numbers are reduced from 3,211,264 and 10,240 to 963,379 and 3,071, respectively, making exactly 70% of the parameter values of the two fully connected layers equal to 0. The model size after pruning, retraining, and sparsification is 22.3 M, which is smaller than the size (37.5 M) of the original model. The compression effect is obvious. Moreover, both accuracy and computing time for predictions slightly decrease as compared with those of the original model.

5.5. 80% Pruning Proportion. Fifth, the pruning proportion is set to 80%; Table 8 shows the parameters of pruning effect in the fifth scene. In this group of experiments, the parameter threshold values of the two fully connected layers are set to 0.11903707 and 0.12662686. In this way, the valid parameter numbers are reduced from 3,211,264 and 10,240 to 642,252 and 2,047, respectively, making exactly 80% of the parameter values of the two fully connected layers equal to 0. The model size after pruning, retraining, and sparsification is 14.9 M, which is smaller than the size (37.5 M) of the original model, and compression is 60%. Additionally, as compared with the original model, the accuracy slightly decreases and the computing time for predictions slightly increases.

Table 7: Descriptions of parameters of the pruning effect in the fourth scene.

	Original number of parameters	Threshold value	Valid parameter number
w_conv1	800	0.0	800
w_conv2	51,200	0.0	51,200
w_fc1	3,211,264	0.09749276	963,379
w_fc2	10,240	0.10360378	3,071

5.6. 90% Pruning Proportion. Lastly, the pruning proportion is set to 90%; Table 9 shows the parameters of pruning effect in the sixth scene. In this group of experiments, the parameter threshold values of the two fully connected layers are set to 0.14814831 and 0.15710811. In this way, the valid parameter numbers are reduced from 3,211,264 and 10,240 to 321,126 and 1,023, respectively, making exactly 90% of the parameter values of the two fully connected layers equal to 0. The model size after pruning, retraining, and sparsification is 7.6 M, which is compressed by 80%. Furthermore, both accuracy and computing time for predictions slightly decreased as compared with those of the original model.

5.7. Comparison Results. Figure 5 shows the comparison results for persistence model size of the four networks, with the pruning ratio increases, and the model size represented by the red columns decreases gradually. Apparently, the pruning proportion is positively correlated with the model size.

Figure 6 shows the comparison results for testing accuracy of the four networks, with the pruning ratio increases, and the testing accuracy represented by the red columns has no obvious changes. This means that there is no positive relationship between pruning proportion and accuracy.

Table 8: Descriptions of parameters of the pruning effect in the fifth scene.

	Original number of parameters	Threshold value	Valid parameter number
w_conv1	800	0.0	800
w_conv2	51,200	0.0	51,200
w_fc1	3,211,264	0.11903707	642,252
w_fc2	10,240	0.12662686	2,047

Table 9: Descriptions of parameters of the pruning effect in the sixth scene.

	Original number of parameters	Threshold value	Valid parameter number
w_conv1	800	0.0	800
w_conv2	51,200	0.0	51,200
w_fc1	3,211,264	0.14814831	321,126
w_fc2	10,240	0.15710811	1,023

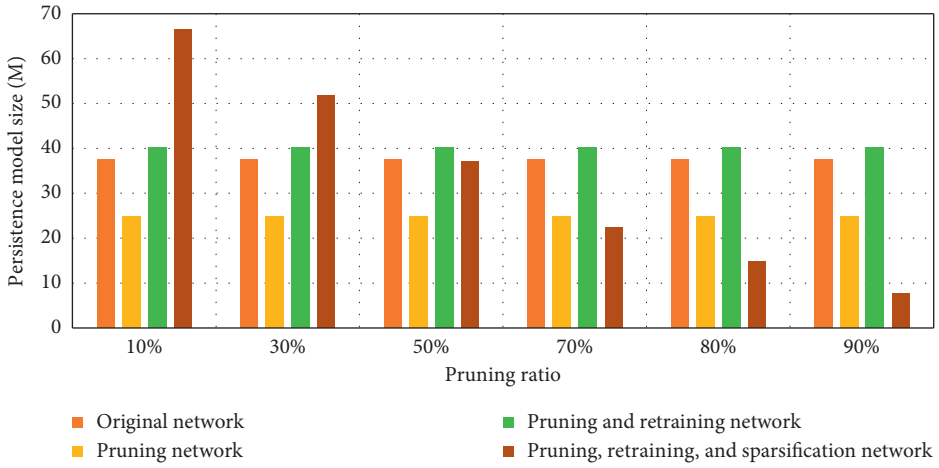

Figure 5: Comparison results for persistence model size of the four networks.

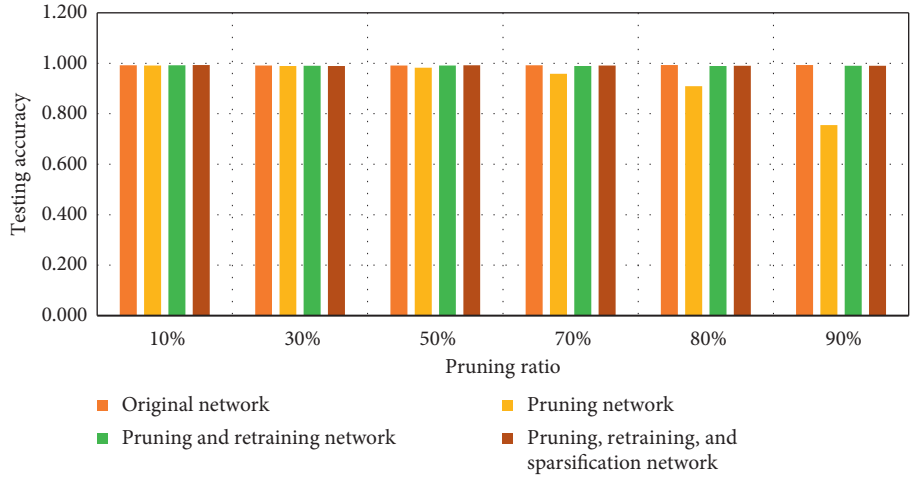

Figure 6: Comparison results for testing accuracy of the four networks.

Figure 7 shows the comparison results for computing time of the four networks. With the pruning ratio increases, the computing time for prediction represented by the red columns changes irregularly. Also, there is no positive relationship between pruning ratio and computing time for predictions.

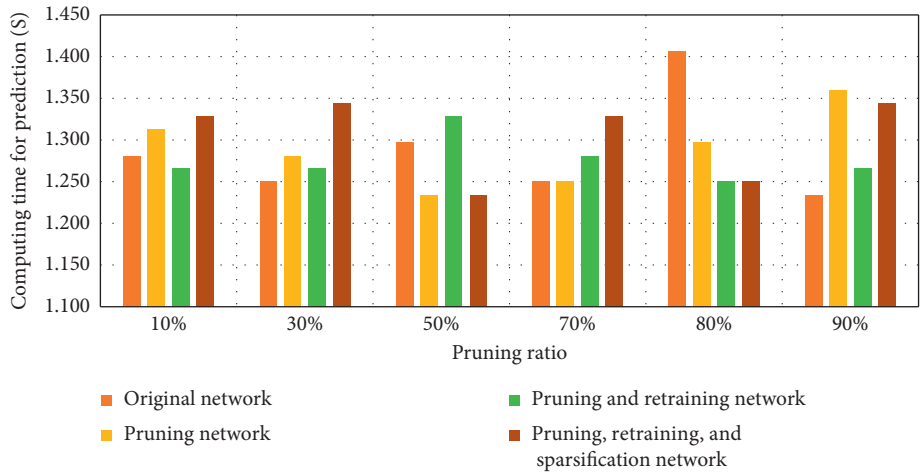

Figure 7: Comparison results for computing time of the four networks.

6. Conclusions

By comparing the experimental data of six different pruning proportions, it is found that pruning does not necessarily compress the size of the model. Compression takes effect only when the pruning proportion reaches 50% or more. Furthermore, we found a positive relationship between the pruning proportion and the model size. However, there was no positive relationship between pruning proportion and accuracy and between pruning proportion and computing time for predictions.

Since there is no specific experimental verification for other models, the conclusion does not apply to other models. Additionally, the experimental is based on the pruning method, pruning is only one of the compression methods of various models; thus, the conclusion of this study is not applicable to other compression methods [27–29].

Data Availability

The data used to support the findings of this study can be accessed publicly in the website http://yann.lecun.com/exdb/mnist/.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the key project of the Natural Science Research of Higher Education Institutions in Anhui Province (grant no. KJ2018A0461); Anhui Province Key Research and Development Program Project (grant no. 201904a05020091); and a provincial quality engineering project from Department of Education Anhui Province (grant no. 2019mooc283).

References

[1] S. Han, J. Pool, S. Narang, H. Mao et al., "DSD: dense-sparse-dense training for deep neural networks," in *Proceedings of the International Conference on Learning Representations (ICLR)*, pp. 1–13, Toulon, France, April 2017.

[2] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.

[3] S. Han, J. Kang, H. Mao et al., "ESE," in *Proceedings of the Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays—FPGA '17*, pp. 75–84, Monterey, California, USA, February 2017.

[4] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," 2016, https://arxiv.org/abs/1608.08710.

[5] Y. Zhang, G. Cui, S. Deng et al., "Efficient query of quality correlation for service composition," *IEEE Transactions on Services Computing*, 2018.

[6] C. Wan, X. Yan, D. Zhang, Z. Qu et al., "An advanced fuzzy Bayesian-based FMEA approach for assessing maritime supply chain risks," *Transportation Research Part E: Logistics and Transportation Review*, vol. 125, pp. 222–240, 2019.

[7] L. Qi, Y. Chen, Y. Yuan, S. Fu et al., "A QoS-aware virtual machine scheduling method for energy conservation in cloud-based cyber-physical systems," *World Wide Web*, vol. 23, no. 2, pp. 1275–1297, 2019.

[8] Z. Huang, G. Shan, J. Cheng, and J. Sun, "TRec: an efficient recommendation system for hunting passengers with deep neural networks," *Neural Computing and Applications*, vol. 31, no. 1, pp. 209–222, 2019.

[9] S. Anwar, K. Hwang, and W. Sung, "Structured pruning of deep convolutional neural networks," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 13, no. 3, p. 32, 2017.

[10] S. Han, H. Mao, and W. J. Dally, "Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015, https://arxiv.org/abs/1510.00149.

[11] B. Wu, X. Yan, Y. Wang, and C. Guedes Soares, "An evidential reasoning-based CREAM to human reliability analysis in maritime accident process," *Risk Analysis*, vol. 37, no. 10, pp. 1936–1957, 2017.

[12] B. Wu, L. Zong, X. Yan, and C. Guedes Soares, "Incorporating evidential reasoning and TOPSIS into group decision-making under uncertainty for handling ship without command," *Ocean Engineering*, vol. 164, pp. 590–603, 2018.

[13] Y. Wang, E. Zio, X. Wei, D. Zhang, and B. Wu, "A resilience perspective on water transport systems: the case of Eastern Star," *International Journal of Disaster Risk Reduction*, vol. 33, pp. 343–354, 2019.

[14] L. Qi, X. Zhang, S. Li, S. Wan et al., "Spatial-temporal data-driven service recommendation with privacy-preservation," *Information Sciences*, vol. 515, pp. 91–102, 2020.

[15] R. Zhu, Z. Sun, T. Ristaniemi, and J. Hu, "Special issue on green telecommunications," *Telecommunication Systems*, vol. 52, no. 2, pp. 1233-1234, 2013.

[16] R. Zhu, W. Shu, T. Mao, and T. Deng, "Enhanced MAC protocol to support multimedia traffic in cognitive wireless mesh networks," *Multimedia Tools and Applications*, vol. 67, no. 1, pp. 269–288, 2013.

[17] D. Zhang, R. Zhu, S. Men, and V. Raychoudhury, "Query representation with global consistency on user click graph," *Journal of Internet Technology*, vol. 14, no. 5, pp. 759–769, 2013.

[18] J. Chang, H. Chao, C. Lai, and R. Zhu, "An efficient geographic routing protocol design in vehicular ad-hoc network," *Computing*, vol. 96, no. 2, pp. 119–131, 2014.

[19] K. Zhu, R. Zhu, H. Nii, H. Samani et al., "PaperIO: a 3D interface towards the internet of embedded paper-craft," *IEICE Transactions on Information and Systems*, vol. E97.D, no. 10, pp. 2597–2605, 2014.

[20] Y. Jalaeian, C. Yin, Q. Wu et al., "Location-aware deep collaborative filtering for service recommendation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems (TSMC)*, pp. 1–12, 2019.

[21] Y. Ma, W. Cho, J. Chen, Y. Huang, and R. Zhu, "RFID-based Mobility for seamless personal communication system in cloud computing," *Telecommunication Systems*, vol. 58, no. 3, pp. 233–241, 2015.

[22] L. Qi, Q. He, F. Chen et al., "Finding all you need: web APIs recommendation in web of Things through keywords search," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 5, pp. 1063–1072, 2019.

[23] X. Xu, Y. Xue, L. Qi, Y. Yuan, X. Zhang et al., "An edge computing-enabled computation offloading method with privacy preservation for internet of connected vehicles," *Future Generation Computer Systems*, vol. 96, pp. 89–100, 2019.

[24] K. Guo, S. Han, S. Yao, Y. Wang et al., "Software-hardware co-design for efficient neural network acceleration," *IEEE Micro*, vol. 37, no. 2, pp. 8–25, 2017.

[25] X. Xu, Y. Li, T. Huang, Y. Xue et al., "An energy-aware computation offloading method for smart edge computing in wireless metropolitan area networks," *Journal of Network and Computer Applications*, vol. 133, pp. 75–85, 2019.

[26] Y. Peng, K. Wang, Q. He et al., "Covering-based web service quality prediction via neighborhood-aware matrix factorization," *IEEE Transactions on Services Computing*, 2019.

[27] X. Xu, Q. Liu, Y. Luo, K. Peng et al., "A computation offloading method over big data for IoT-enabled cloud-edge computing," *Future Generation Computer Systems*, vol. 95, pp. 522–533, 2019.

[28] B. Jalaeian, R. Zhu, H. Samani, and M. Motani, "An optimal cross-layer framework for cognitive radio network under Interference Temperature model," *IEEE Systems Journal*, vol. 10, no. 1, pp. 293–301, 2014.

[29] T. Zhou, C. Wu, J. Zhang, and D. Zhang, "Incorporating CREAM and MCS into fault tree analysis of LNG carrier spill accidents," *Safety Science*, vol. 96, pp. 183–191, 2019.