Journal of Network Intelligence
Taiwan Ubiquitous Information
 ISSN 2414-8105 (Online)
Volume **5**, Number **3**, August **2020**

A Digital Watermarking Scheme for Protecting Weights of Convolutional Neural Networks

Yi-Jia Zhang

School of Information Science and Technology
Zhejiang Sci-Tech University
Hangzhou 310018, P. R. China
waiting@zstu.edu.cn

Hang-Yu Fan

Alibaba Group
Hangzhou 311121, P. R. China
hangyu.fhy@alibaba-inc.com

Zhe-Ming Lu*

School of Aeronautics and Astronautics
Zhejiang University
Hangzhou 310027, P.R. China
*Correponding Author: zheminglu@zju.edu.cn

Received June 2020;Revised July 2020
(Communicated by Zhe-Ming Lu)

Abstract. *Because training a useful Convolutional Neural Network (CNN) needs huge resources, the copyright of the weights of CNNs will definitely be noticed. But there is few related work about embedding watermarks in the weights of CNNs for copyright protection. In this paper, we propose a complete scheme about embedding watermarks in the weights of CNNs. We combined the quantization index modulation scheme and the chaotic Logistic map to embed the watermark. The robustness of the embedding method is proved to be pretty good. If choosing a large quantization step will enhance the robustness but leading to performance degradation. In order to fix the problem of performance degradation, we propose a mask-retraining process to effectively improve the accuracy performance of the embedded weights. According to our experiments, our scheme can embed robust watermarks and preserve the network performance good at the same time.*
Keywords: Weights watermarking, Copyright protection, Deep learning, Convolutional neural network, Robust watermark.

1. **Introduction.** With the development of the Internet, the copyright protection for digital information is becoming more and more important. For digital images and videos, many digital watermarking algorithms have been proposed to protect copyrights. Besides them, many digital intellectual properties need to be protected. Convolutional neural network (CNN) is a new technology which grows rapidly. As a big branch of Artificial Intelligence (AI), the technology of CNN is widely used in many fields. The copyright of weights in CNNs should be protected immediately.

In recent years, CNN becomes the rock star in the field of AI, and various excellent network architectures have emerged one after another. From LeNet [1], a large number of

outstanding networks such as AlexNet [2], VGGNet [3], Xception [4], and ResNet [5] have been born. Thanks to the increase of computing capability of the computer, researchers are able to train deeper networks with more weights. With continuous research, the training processes of CNNs are also getting more complicated, and various training tricks have been proposed, and the training sets are getting larger too. It is very common that you cannot reproduce the state-of-the-art results mentioned in some papers when you do not know the training details. On the other hand, benefiting from the maturity of various machine learning frameworks, such as TensorFlow [6] and Caffe [7], we are able to build a network easily. So, for CNN, it is easy to deploy but hard to train. Thus, the well-trained weights of CNNs are valuable. Like all intellectual properties, the copyright of trained weights should be protected.

To the best of our knowledge, there are few copyright protection methods for the weights of CNNs. Therefore, this paper proposes a complete scheme to protect the copyright of the weights of CNNs. Our method can embed the watermark into the weights of CNNs and has little effect on the network performance, and it can be used in practice.

The rest of this paper is organized as follows. In Section 2 we discuss the related works about the features of CNNs and embedding technologies for digital images. The approaches of our scheme are reported in Section 3. The experimental results and related discussions are showed in Section 4. The conclusions and future work are drawn in Section 5.

2. **Related Works.**

Until now, there are no watermarking schemes for the weights of CNNs. But there have been many related research works [8-13]. Inspired by these research works, we propose a feasible watermarking method for the weights of CNNs.

With the development of CNNs, the number of network weights is increasing. AlexNet [2] is one of the typical representatives. The weights of AlexNet occupy the storage space about 600 megabytes without compression. So some researchers proposed pruning and quantification methods for weights compression [11]. They found part of weights which close to 0 can be removed, and the performance of the network was not affected or slightly affected. This phenomenon indicates that slight changes in weights do not affect the network performance. If the performance is sensitive to slight changes of weights, then according to the machine learning theory, these weights are over-fitted. The over-fitted weights are useless, and they will not be discussed in this paper.

This feature of CNN weights is used in special networks such as XNOR-Net and Binary-Weight-Networks, which use quantified weights [12]. This feature illustrated that the weights are redundant, and this is the cornerstone of our watermark embedding scheme. In the pruning process of training weights, the pruning ratio needs to be set first. Then all connections with weights below a threshold are removed from the network. Finally, they retrained the network to learn the final weights for the remaining sparse connections. In this process, the pruned weights were not really removed, but set to 0. In the retraining process, the pruned weights were masked and remained 0. The retraining process enlightens us to enhance the performance after the embedding.

Digital image watermarking has been researched and developed for many years, and various embedded technologies have been proposed [8-10]. The pixels of the digital images have similar features to the weights of CNNs. They are redundant, and slight changes have little effect. Therefore, the watermarking scheme for digital images should also work for watermarking the weights of CNNs. Quantization index modulation technology is a data embedding method proposed by Chen and Wornell [13]. This method is widely used in digital watermarking, and this method is convenient and easy to calculate and difficult

to detect. Therefore, we use quantization index modulation technology as the watermark embedding method in our research work.

3. Proposed Approach.

3.1. CNN and Weights.

A classic CNN usually consists of one input layer, some convolution layers, some pooling layers, some fully connected layers, and one output layer. Fig. 1 shows the network architecture of AlexNet [2]. In CNN, the convolutional layers and the fully connected layers need to be trained to obtain weights, which are the watermark carrier. The network weights have an initial value at the beginning of the training process and then the weights converge as the training process goes on.

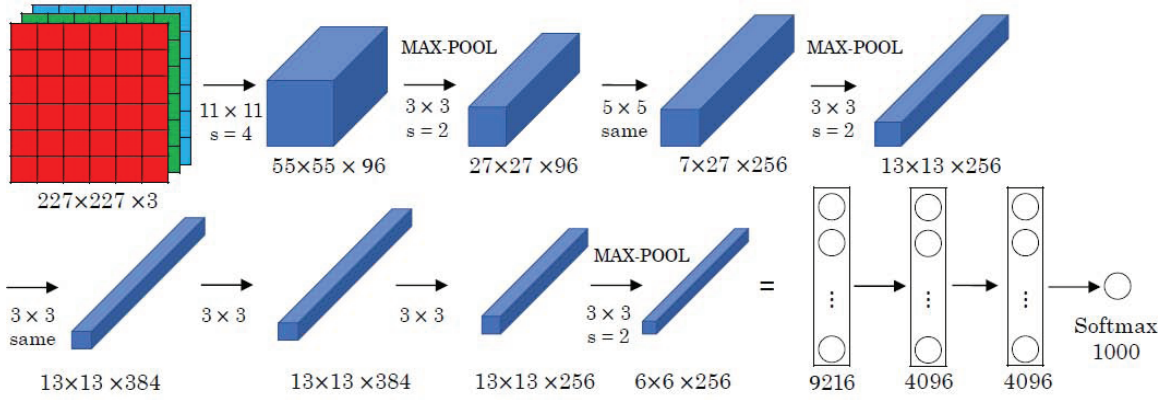


FIGURE 1. The network architecture of AlexNet.

A convolutional layer consists of several feature maps, each of which consists of several neurons. Neurons are connected to local regions of previous feature maps by a convolution kernel. The convolution kernel is a matrix of weights. For a convolution process in the convolutional layer, input and output have different weights. Therefore, the weights of one convolution layer can be abstracted into a four-dimensional tensor, as shown in Eq. (1).

$$\texttt{ConvolutionLayerWeights} = \texttt{tensor}[in, w, h, out] \tag{1}$$

The parameter in represents the number of input channels, out represents the number of output channels. And w and h represent the width and height of the convolution kernel respectively.

Each neuron in the fully connected layer is fully connected to all neurons in the previous layer. The fully connected layer can synthesize all the information from the previous layer. If the previous layer contains d_1 neurons and the fully connected layer outputs d_2 neurons, then the calculation of the fully connected layer can be expressed as Eq. (2):

$$\begin{bmatrix} a_1 \\ . \\ . \\ . \\ a_{d_2} \end{bmatrix} = \begin{bmatrix} W_{1,1} & . & . & . & W_{1,d_1} \\ . & . & & & . \\ . & & . & & . \\ . & & & . & . \\ W_{d_2,1} & . & . & . & W_{d_2,d_1} \end{bmatrix} \begin{bmatrix} x_1 \\ . \\ . \\ . \\ X_{d_1} \end{bmatrix} + \begin{bmatrix} b_1 \\ . \\ . \\ . \\ b_{d_2} \end{bmatrix} \tag{2}$$

The variables $\boldsymbol{x}$ and $\boldsymbol{a}$ denote input neurons and output neurons respectively, and $\mathbf{W}$ and $\boldsymbol{b}$ are weights of the fully connected layer.

3.2. **Quantization Index Modulation.** Quantization Index Modulation (QIM) technology [13] has been widely used in the field of digital image watermarking. QIM is a typical application of quantization embedding technology. The main idea of QIM is to quantize the original data to different quantization intervals according to the different watermark information, and to identify the watermark information according to the associated quantization interval. A typical application example of the QIM algorithm is Dither Modulation (DM) [14]. The basic principle of DM is using the watermark information to dither the carrier data. We use Δ to represent the quantization step size. For the original signal c, the embedded data c_0 and c_1 are shown in Eq. (3):

$$\begin{cases} c_0 = c \pm \mathtt{mod}(c, 2) \times \triangle \\ c_1 = c \pm \mathtt{mod}(c + \triangle, 2) \times \triangle \end{cases} \tag{3}$$

Where c_0 indicates that the embedded information is 0, and c_1 indicates that the embedded information is 1. c_0 and c_1 both have two values to choose. In general, we choose the one which is closer to the original signal.

3.3. **Embedding Positions.** We use the chaotic Logistic map to generate the embedding positions [15]. The chaotic map is generated by the chaotic equation $x_{n+1} = f(x_n)$. The chaotic Logistics map is sensitive to the initial value. It can generate a series of irregular values when provided the initial value x_0. The orders of x_1 can be used as embedding positions. Using chaotic maps to select the embedding positions can increase the security of the watermark. The chaotic equation in this paper is defined in Eq. (4).

$$x_{n+1} = 1 - \mu x_n^2 \tag{4}$$

where μ is an adjustable parameter, and for any n, the obtained $x_n \in [0, 1]$ is guaranteed. According to the initial value x_0, a chaotic map sequence $\{x_i\}, i = 1, 2, \ldots$ can be generated, and the sequence $\{x_i\}$ can be sorted to obtain an ordered sequence $\{k_i\}, i = 1, 2, \ldots,$ which can be used as the watermark embedding positions.

3.4. **Mask and Retraining.** In this study, the performance of the network will decrease due to the changes of weights. If there is too much embedded content, or if the modification is too large (Δ is too large), the degradation of the network performance will become more obvious. Inspired by the retraining process in [11], we propose a training method for improving the performance after the embedding. The method is loading the weights of the embedded watermark, and putting a mask on the embedded positions, then retraining with a low learning rate. The effect of the mask is to ensure the embedded weights will not be modified during the training process. After the mask-retraining operation, the performance of the embedded network will be better.

It is worth to mention that when the appropriate Δ and embedding capacity are selected, the embedded network performance can be almost the same as the original one. In this case, there is no need to perform the mask-retraining process.

3.5. **Processes of Our Scheme.** Based on above description, we propose a copyright protection scheme for the weights of CNNs. The overall embedding process can be illustrated as follows:

Step 0: Prepare the weights of CNNs and the watermark for embedding. Set the performance degradation tolerance δ. The watermark can be encrypted when necessary. Set the quantization step Δ and the initial value x0 for the chaotic Logistic map, then save them as the extraction key.

Step 1: Determine the embedding positions using the chaotic Logistic map.

Step 2: Embed the watermark in selected positions using the dither modulation scheme.

Step 3: Test the network performance after embedding. If the performance degradation is greater than δ, execute Step 4, or exit the embedding process.

Step 4: Load the embedded weights, and then execute the mask-retraining process.

The extraction process is the reverse process of the embedding stage, which can be described as follows:

Step 0: Prepare the possibly-watermarked weights and the watermark extraction key.

Step 1: Generate embedding positions by the chaotic Logistic map.

Step 2: Extract the watermark from the embedding positions by the dither modulation scheme. If the watermark information is encrypted, decryption should be performed.

4. **Experimental Results.** In this section, the weights of AlexNet and VGG16 are the pre-trained weights of ImageNet-1000 [16], and the performances are shown in Table 1. AlexNet and VGG16 are used in Experiments 1 and 2. Experiment 3 uses a shallow CNN, and the training data is Cifar-10 [17], and the shallow CNN architecture is given in Table 2.

Table 1. The accuracy of AlexNet and VGG16.

	Top-1	Top-5
AlexNet	0.484	0.730
VGG16	0.700	0.892

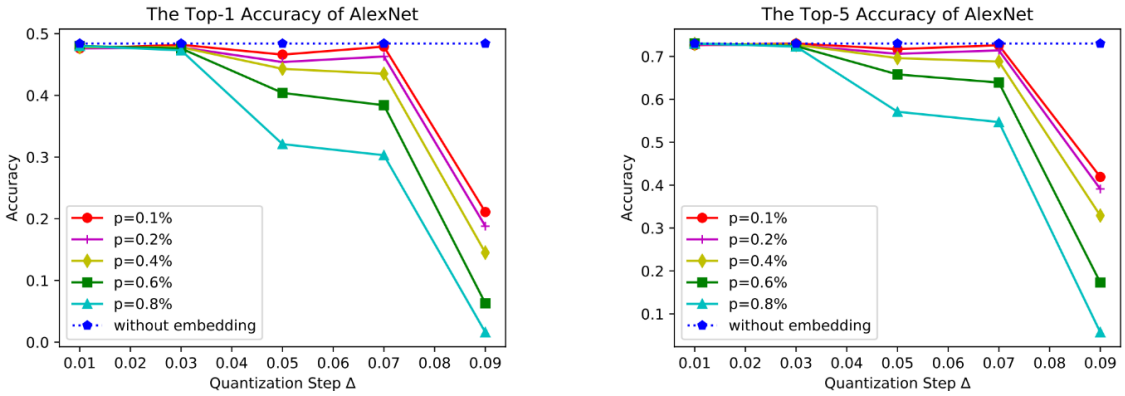
Table 2. The network architecture for Cifar-10 in Experiment 3.

Layer type	Layer size
Input layer	[1,24,24,3]
Conv layer	[3,5,5,64]
Max pooling layer	[1,3,3,1]
Conv layer	[64,5,5,64]
Max pooling layer	[1,3,3,1]
Fully connected layer	[*,384]
Fully connected layer	[384,192]
Fully connected layer	[192,10]

4.1. **Experiment 1.** In this experiment, we test the performance after watermark embedding. $\triangle$ represents the quantization step, and p represents the proportion of embedded weights. This experiment uses AlexNet and VGG16 architectures and uses ImageNet-1000 pre-trained weights as watermark carrier. The watermark is the binary information as shown in Fig. 2. The binary information will be embedded multiple times when the embedding space is larger than the number of watermark bits. The experimental results are shown in Fig. 3 and Fig. 4.

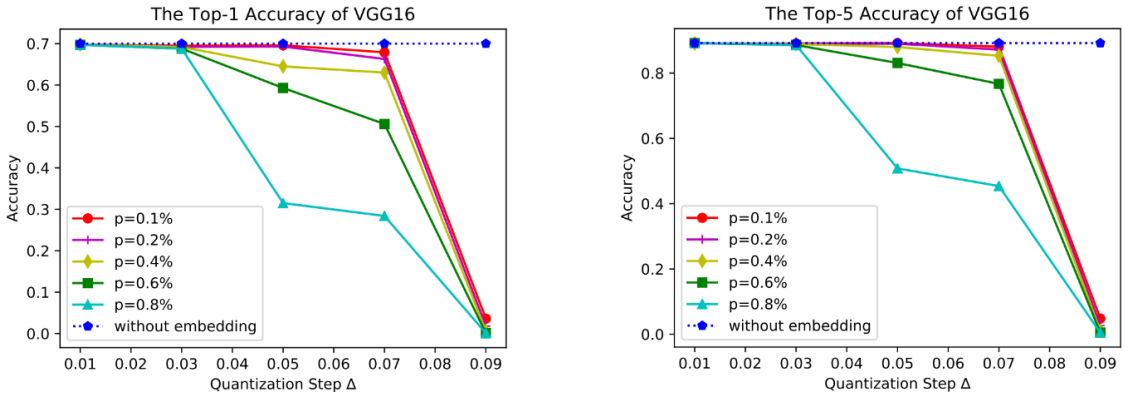
Through this experiment, it can be found that the performance usually gets worse when choosing larger $\triangle$ or p. Therefore, if you want to keep the network performance, try to choose a small $\triangle$ or embed less information. When using appropriate $\triangle$ and p (such as $\triangle = 0.01$ and $p = 0.6\%$ for AlexNet), we found that the degradation of network top-5 accuracy is 0. Using large $\triangle$ will reduce the performance, but it will enhance the robustness of watermark. In Experiment 2, the details of robustness will be shown. In Experiment 3, the method of improving the performance after embedding will be shown.

FIGURE 2. The binary image for embedding.



(a) (b)

FIGURE 3. The Top-1 and Top-5 accuracy of AlexNet after embedding.



(a) (b)

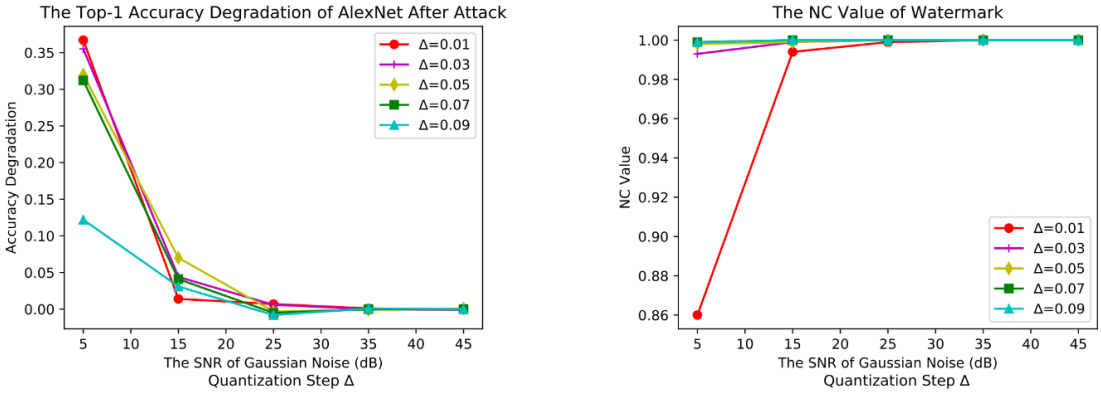
FIGURE 4. The Top-1 and Top-5 accuracy of VGG16 after embedding.

4.2. **Experiment 2.** Experiment 1 introduces the impact of watermark embedding on the network performance. Since the watermark is not attacked, the extracted watermark is the same as the embedded one. In this experiment, we use the additive white Gaussian noise to attack the embedded weights, and observe the impact of the attack. The noises are added to all weights, and we use Signal-to-Noise Ratio (SNR) as the indicator of the strength of noise. Smaller SNR denotes heavier attack. In this experiment, the weights embedding capacity is 0.4%. In order to illustrate the robustness of our method, we use

the normalized correlation (NC) value as the evaluating indicator. The NC value can be computed as follows.

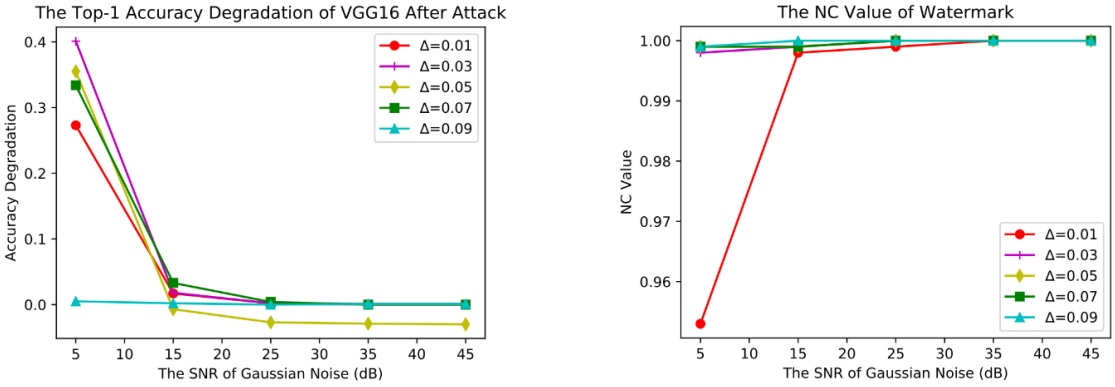
$$NC(w, w^*) = \frac{\sum_{i=0}^{M-1} w(i) * w^*(i)}{\sum_{i=0}^{M-1} w^2(i)} \tag{5}$$

The experimental results are shown in Fig. 5 and Fig. 6. According to the experimental results, it can be seen that the robustness of our method is very good, and using larger $\triangle$ can resist more critical attack. Heavier attack leads to the worse performance, so good performance is a protection to the watermark in the weights. This is the biggest difference between our watermark carrier (i.e., the weights of CNNs) and other watermark carriers (such as images or videos). The attacker must bear the network performance degradation, and a poor performance weights do not need protection.



(a) (b)

FIGURE 5. The accuracy degradation of AlexNet, and the NC values after attack. The accuracy degradation is evaluated by comparing the performance between the embedded network and the embedded-then-attacked one.



(a) (b)

FIGURE 6. The accuracy degradation of VGG16, and the NC values after attack. The accuracy degradation is evaluated by comparing the performance between the embedded network and the embedded-then-attacked one.

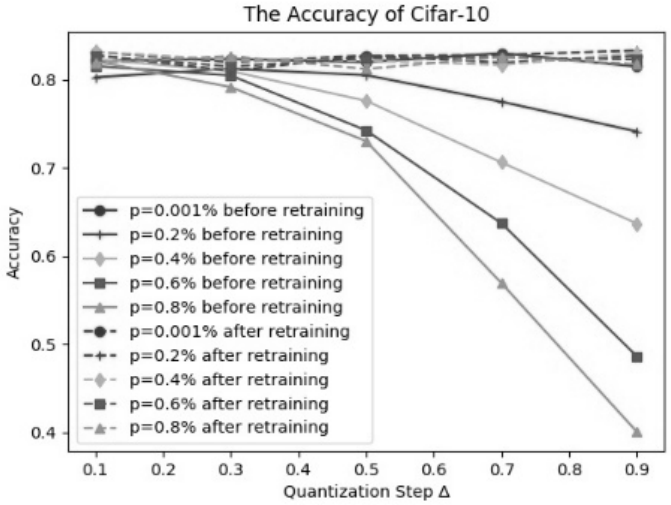


FIGURE 7. The network performance before and after mask-retraining process.

4.3. **Experiment 3.** From Experiments 1 and 2, we know that choosing larger can bear heavier attack, but it will also lead to network performance degradation. In order to solve the performance degradation caused by embedding, we propose the mask-retraining process. This experiment is mainly to show the effect of the mask-retraining process. The retraining process reduces the learning rate compared to the original training process. Because the ImageNet-1000 dataset training needs too much time, we use Cifar-10 as the training data in this experiment. The structure of CNN is shown in Table 2. The best accuracy of this network model is 0.864. In this experiment, we select $\triangle$ and p as variables, and then observe the accuracy before and after mask-retraining. The experimental results are shown in Fig. 7. According to the experimental results, the mask-retraining process can effectively improve the performance after the watermark embedding.

5. **Conclusion and Future Work.** In this paper, we propose a watermarking method for the weights of CNNs, which aims to protect the copyright of the trained weights. With the popularity of CNNs, the copyright issue of network weights will inevitably attract attention. However, there is still few related work. We propose a feasible network weights copyright protection scheme to make up for this vacuum. Aiming at the problem of performance degradation caused by embedding, a mask-retraining solution is proposed to solve this problem. According to our experiments, we can find that the choice of quantization step $\triangle$ and the proportion of embedded weights p are very important. Appropriate $\triangle$ and p not only affect the performance after embedding, but also affect the best performance after mask-retraining. The suitable $\triangle$ and p for different networks or weights should be different. Our advice is choosing smaller $\triangle$ and p. Because compared to watermark information, network performance is more sensitive to attack.

More future works about the weights watermark can be done. How to find the most appropriate weights positions to embed watermark? How to enhance the robustness? Is the vector quantization can be used in CNN weight watermarking for saving storage space and enhance embedding quality? Is watermark can be embedded in the frequency domain? These questions need to be thought and conquered.

Acknowledgment. This work is supported by Science Foundation of Zhejiang Sci-Tech University(ZSTU) under Grant No.19032458-Y.

REFERENCES

[1] S. Lin, L. Cai, X. Lin, R. Ji, Masked face detection via a modified LeNet, *Neurocomputing*, vol.218, pp. 192–202, 2016.

[2] S. Lu, Z. Lu, Y.-D. Zhang, Pathological brain detection based on AlexNet and transfer learning, *Journal of Computational Science*, vol.30, pp.41–47, 2019.

[3] I. Hammad, K. El-Sankary, Impact of approximate multipliers on VGG deep learning network, *IEEE Access*, vol.6, pp. 60438–60444, 2018.

[4] F. Chollet, Xception: Deep learning with depthwise separable convolutions, *Proceedings of the IEEE conference on computer vision and pattern recognition*, vol.1, pp.1800–1807, 2017.

[5] B. Li, Y. He, An improved ResNet based on the adjustable shortcut connections, *IEEE Access*, vol.6, pp. 18967–18974, 2018.

[6] M. Liu, D. Grana, Accelerating geostatistical seismic inversion using TensorFlow: A heterogeneous distributed deep learning framework, *Computers & Geosciences*, vol. 124, pp. 37–45, 2019.

[7] M. Komar, P. Yakobchuk, V. Golovko, V. Dorosh, A. Sachenko, Deep neural network for image recognition based on the Caffe framework, *IEEE Second International Conference on Data Stream Mining & Processing (DSMP)*, 2018.

[8] H.-Y. Fan, Z.-M. Lu, Y. Liu, The digital image watermarking scheme using low frequency construction and histogram, *International Journal of Innovative Computing, Information and Control*, vol.16, no.1, pp.367-384, 2020.

[9] H. Sadreazami, M. Amini, A robust image watermarking scheme using local statistical distribution in the contourlet domain, *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol.66, no.1, pp.151-155, 2019.

[10] H.-Y Fan, and Z.-M. Lu, A BTC-compressed domain information hiding method based on histogram modification and visual cryptography, *International Journal of Innovative Computing, Information and Control*, vol.12, no.2, pp.395-405, 2016.

[11] T.-Y. Hsiao, Y.-C. Chang, H.-H. Chou, C.-T. Chiu, Filter-based deep-compression with global average pooling for convolutional networks, *Journal of Systems Architecture*, vol.95, pp.9-18, 2019.

[12] H. Peng, S. Chen, BDNN: Binary convolution neural networks for fast object detection, *Pattern Recognition Letters*, vol.1251, pp.91–97, 2019.

[13] N. Cai, N. Zhu, S. Weng, B. W.-K. Ling, Difference angle quantization index modulation scheme for image watermarking, *Signal Processing: Image Communication*, vol.34, pp.52–60, 2015.

[14] M. Li, and X. Yuan, Robust feature extraction based watermarking method using spread transform dither modulation, *International Conference on Machine Vision and Information Technology (CMVIT)*, 2017.

[15] C. Han, An image encryption algorithm based on modified logistic chaotic map, *Optik*, vol. 181, pp. 779–785, 2019.

[16] D. Mishkin, N. Sergievskiy, J. Matas. Systematic evaluation of convolution neural network advances on the Imagenet, *Computer Vision and Image Understanding*, vol.161, pp.11–19, 2017.

[17] R. Doon, T. K. Rawat, S. Gautam, Cifar-10 classification using deep convolutional neural network, *IEEE Punecon*, 2018.