Hindawi
Scientific Programming
Volume 2023, Article ID 9896261, 1 page
https://doi.org/10.1155/2023/9896261

Retraction

Retracted: Research on Network Layer Recursive Reduction Model Compression for Image Recognition

Scientific Programming

Received 31 October 2023; Accepted 31 October 2023; Published 1 November 2023

This article has been retracted by Hindawi following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of one or more of the following indicators of systematic manipulation of the publication process:

(1) Discrepancies in scope
(2) Discrepancies in the description of the research reported
(3) Discrepancies between the availability of data and the research described
(4) Inappropriate citations
(5) Incoherent, meaningless and/or irrelevant content included in the article
(6) Peer-review manipulation

The presence of these indicators undermines our confidence in the integrity of the article's content and we cannot, therefore, vouch for its reliability. Please note that this notice is intended solely to alert readers that the content of this article is unreliable. We have not investigated whether authors were aware of or involved in the systematic manipulation of the publication process.

Wiley and Hindawi regrets that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our own Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] H. Ling, W. Zhang, Y. Tao, and M. Zhou, "Research on Network Layer Recursive Reduction Model Compression for Image Recognition," *Scientific Programming*, vol. 2021, Article ID 4054435, 11 pages, 2021.

Hindawi
Scientific Programming
Volume 2021, Article ID 4054435, 11 pages
https://doi.org/10.1155/2021/4054435

Research Article

Research on Network Layer Recursive Reduction Model Compression for Image Recognition

Hongfei Ling ,[1,2] Weiwei Zhang ,[1,2] Yingjie Tao,[1,2] and Mi Zhou[1,2]

[1]*College of Engineering, Huaqiao University, Quanzhou 362021, Fujian, China*
[2]*Industrial Intelligence and System Fujian University Engineering Research Center, Huaqiao University, Quanzhou 362021, Fujian, China*

Correspondence should be addressed to Weiwei Zhang; weiweizh@hqu.edu.cn

Received 26 July 2021; Revised 21 August 2021; Accepted 23 August 2021; Published 9 September 2021

Academic Editor: Bai Yuan Ding

ResNet has been widely used in the field of machine learning since it was proposed. This network model is successful in extracting features from input data by superimposing multiple layers of neural networks and thus achieves high accuracy in many applications. However, the superposition of multilayer neural networks increases their computational cost. For this reason, we propose a network model compression technique that removes multiple neural network layers from ResNet without decreasing the accuracy rate. The key idea is to provide a priority term to identify the importance of each neural network layer, and then select the unimportant layers to be removed during the training process based on the priority of the neural network layers. In addition, this paper also retrains the network model to avoid the accuracy degradation caused by the deletion of network layers. Experiments demonstrate that the network size can be reduced by 24.00%–42.86% of the number of layers without reducing the classification accuracy when classification is performed on CIFAR-10/100 and ImageNet.

1. Introduction

Convolutional neural network (CNN) is a commonly used neural network model in the field of computer vision as it can achieve high accuracy in various tasks in the field of image recognition [1, 2]. Network models can deepen the network structure by CNNs and thus improve the accuracy of tasks such as recognition or detection. For example, LeNet-5, proposed by LeCun et al. uses a 5-layer CNN to classify handwritten text. Later, VGGNet-19 utilized 22 layers to further improve the accuracy [2]. The residual network (ResNet) [3] even uses 152 layers of neural networks to achieve the optimal performance for the current task competition. As a result, ResNet is now commonly used as one of the models of standard CNNs in diverse fields, such as medical disease map classification, forestry pest, and disease classification [4].

ResNet is used to solve the problem of performance degradation caused by increasing depth. The biggest difference between DenseNet and ResNet is that in DenseNet we never combine features through summation before they are passed into a layer; instead, we provide them all as separate inputs. ResNeXt proposes aggregated transformations, using a parallel stack of blocks with the same topology to replace the original three-layer convolution block of ResNet, which improves the accuracy of the model without significantly increasing the parameter level. At the same time, due to the same topology, the number of hyperparameters is also reduced, which is convenient for model transplantation. In SE-ResNet and SE-ResNeXt, SENet can be regarded as a channel-wise attention [5, 6]. SENet adds a branch to calculate the channel-wise scale after the normal action and then multiplies the obtained value to the corresponding channel [7].

CNN improves accuracy through deep structure on the one hand, and on the other hand the computational cost required for learning and model inference increases as the number of layers increases. During model training, computational resources are enhanced by adding hardware devices, and computation time can be significantly reduced by distributed algorithms. However, with the advent of the Internet of Things (IoT) era, models often need to be deployed again on end devices with limited computational resources, for example, image classification on embedded systems, text recognition on portable devices, and speech recognition on mobile devices. A higher task level requires a larger amount of hardware computational memory, and thus the significant problem that arises is the high operational and inference cost requirements of IoT end devices and realistic scenarios where end devices often struggle to meet the high demand for computational resources [8]. Therefore, how to effectively reduce the computational cost of CNN training and inference has received significant attention from researchers. For example, for the problem of computational cost of deep neural network models, Denton et al. [7] proposed to try to reduce the computational cost by cutting the number of layers, preserving all network layers of the residual network, and changing the number of network layers executed according to the input data. However, it is also necessary to save all the network layers in the scheme and not just consider cutting the cost of computational resource consumption. On the contrary, deciding which network layer is skipped also increases the memory consumption due to the additional modules required to determine it. For this reason, Chen et al. [8] proposed a method to decrease the number of residual network layers during learning, which can shorten the time of inference computation and reduce the memory consumption at the same time. However, this scheme removes the network layers completely statically, so it is sometimes difficult to maintain a high accuracy rate. Rastegari et al. [9] used the distillation maneuver to learn new models with fewer network layers from learned models with more layers, but their experiments showed a huge decrease in accuracy. Moreover, from the point of view of reducing the computational cost, to reduce the model inference time and computational resource cost, the use of static deletion of layers and distillation can be satisfied.

In this paper, we present a model compression method that uses layer deletion and retraining and can suppress accuracy degradation. The proposed method imports judgment values that determine the importance of each layer of the residual network. The judgment values are used to determine or remove unimportant layers after learning and preventing the accuracy degradation. This paper is to retrain the residual network after removing the layers. The experimental results show that layer deletion and retraining in such a way are applicable for overall model compression and reduce the cost of computational resources. Maintaining accuracy in the deleted layers requires retraining by removing individual parameters and then retraining with different hyperparameter settings. Experiments using the CIFAR-10/100 image dataset for the image classification task cut the number of network layers by 24.00% to 42.86%. Accordingly, the computation time for model inference decreased by 60.23% to 76.69%, and the number of parameters of the model was reduced to 69.82% to 93.15%.

1.1. Related Work. Present-day model compression schemes for ResNet fall into three broad categories.

In the first category, Jaderberg et al. [10] dynamically decide whether to execute or skip the next layer in the middle of the inference calculation of the residual network. Han et al. [11] decide whether or not to execute a layer by adding a gate function to each layer. The signal from the previous layer is input to the gate function; if it outputs 1, the next layer is executed, and if it outputs 0, it is not executed. Liu et al. [12] take action based on reinforcement learning to decide which layer is executed by the residual network, which attempts to reduce the number of layers executed by rewarding the accuracy of the actual execution while suppressing the reduction in accuracy. However, while these approaches reduce the average inference time, they require additional gate functions and neural networks and thus suffer from increased memory consumption.

In the second category, Huang et al. [13] used a model that multiplies the reasonable judgment values by the output of the network layers. et al. [14] set many judgment values to 0 in learning by adding L1 regularization on this judgment value, while being able to remove such layers completely since the scalar is the same as the layer corresponding to the value 0 for the unexecuted state. Wu et al. [15] set a threshold value in the output of each layer and removes the layers below the threshold value. These methods differ from the methods in the first category in that the time and memory consumption of the model inference computation can be reduced simultaneously, except that the layers can be cut completely. However, it is difficult to maintain the accuracy rate in order to completely remove layers. Although experiments show that these methods can actually maintain accuracy in data such as CIFAR-10, it is difficult to maintain accuracy in actual data such as specific real-time images. In addition, these methods require adjustment of hyperparameters of continuous values such as regularization strength and threshold. That is, our method is used to obtain the Resnet network with the required number of layers by continuously adjusting the hyperparameters, which does not cause degradation in the accuracy of the image data, and additional cost is spent.

In the third category, Wen et al. [16] used the technique of distillation, which is a framework for efficiently training another neural network (student) using information from an already trained neural network (teacher). Specifically, it aids student learning by imposing a constraint that the probability distribution of the output of the teacher model and the output of the student model should be parsimonious. In addition, it is easy to use as the number of layers required can be set directly for teacher-student learning in the case of hardware memory constraints. Since then, [17–19] distillation methods have been applied to the compression of various models. These schemes are different from the layer parameter removal involved in this study.

2. ResNet

ResNet is a CNN neural network model widely used in the field of image recognition [3]. ResNet achieves the deep structure of the model by stacking multiple residual blocks, and each residual block is composed of residual units constructed from multiple convolutional layers. Residual units in the residual network model are calculated as follows:

$$x_{i+1} = x_i + F(x_i), \tag{1}$$

where x is the input signal to the residual unit and $F(x_i)$ is a module consisting of a convolution layer, batch normalization, and ReLU activation function [20]. Thus, the residual unit takes the input signal through constant mapping and nonlinear mapping and adds its result as a new kind of computational unit.

Multiple residual units of the same size in different dimensions are superimposed in each residual block. When changing the residual block, downsampling, or increasing the number of channels is performed, as shown in Figure 1, the dimensionality of the residual unit is changed.

3. The Proposed Method

The proposed method statically eliminates the number of ResNet layers while minimizing the loss of its accuracy. This method gradually reduces the number of layers of the residual network by iterative layer deletion and retraining.

3.1. Deleting Residual Blocks. Previous studies have shown that schemes that directly remove the residual blocks significantly reduce the model accuracy [14, 21, 22]. However, if these residual blocks are removed, it is difficult to recover the accuracy by retraining. Therefore, it is necessary to remove the residual blocks that have little impact on accuracy by model compression techniques.

For this problem, our approach introduces a variable that represents the importance of the residual units. This variable is a judgment value that can be learned from the training data as well as from the model parameters, introducing a variable for each residual unit. Specifically, the importance of $F(x_i)$ in equation (1) is learned. Identifying the unimportant $F(x_i)$ removes it, and the input of the next residual block becomes $x_i + 1 = x_i$, which has the same effect as removing the residual block itself. By introducing the variables indicating the importance into the residual unit in formula (1), we can obtain

$$x_{i+1} = x_i + \omega_i F(x_i), \tag{2}$$

where ω_i is a judgment-valued variable that can be learned by error backpropagation. ω_i can be viewed as a judgment-valued layer overlaid on top of $F(x_i)$, so calculations such as error backpropagation can be easily implemented using a deep learning framework. When the absolute value $|\omega_i|$ of ω_i becomes small, the output size of $F(x_i)$ is considered small, such that $F(x_i)$ is considered to have little effect on the output. Therefore, in this method, ω_i is used as the judgment

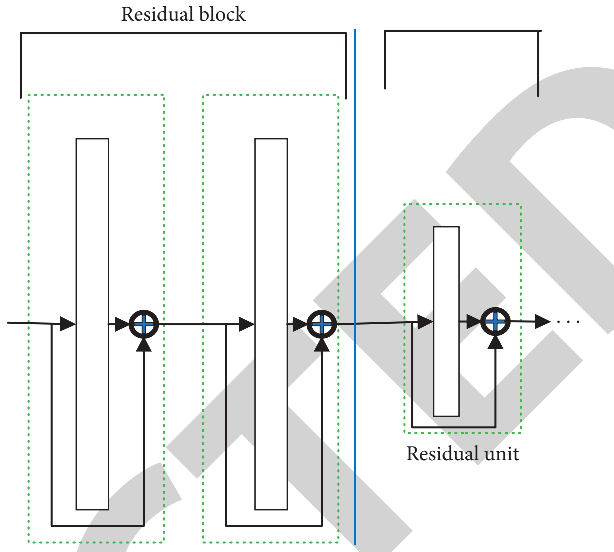


Figure 1: The change of residual block and residual unit in ResNet.

value for the importance of the residual block, and $|\omega_i|$ is used as the target to remove the residual block.

In the residual network, there are residual blocks that cannot be removed from the beginning to the end. According to [23], removing residual blocks immediately after a residual block change significantly reduces the model accuracy because a new intermediate representation is obtained by downsampling and increasing the number of channels when the residual block changes (Figure 1). Since such a residual block is important to maintain accuracy, instead of removing the residual block immediately after a residual block change, the usual formula (1) is used in this paper. This solution requires an additional parameter (importance), which is not a vector or tensor, but a scalar, so it has little impact on the size of the model [24].

3.2. Retraining. The authors in [14, 23] and others showed that the accuracy has a tendency to decrease if multiple residual blocks are removed from the residual network at a time. Based on the present method of alternating the removal and retraining of residual blocks, instead of removing multiple residual blocks at a time, the residual blocks are removed in a gradual manner. Baker et al. [25] categorized the method of removing parameters from the model as elemental variational kernel/group filtering level, and this method removes parameters of the layer level, which is not included in the above category. Therefore, the amount of model modification by retraining is also considered to be large. In retraining, this paper sets the learning rate initial value larger in the stochastic gradient descent optimizer to significantly update the parameters [11, 26]. Specifically, this paper applies the same learning rate for retraining as in the first learning. Such a learning rate setting contrasts with the small learning rate used in [24, 27] for retraining in single parameter removal.

3.3. Overall Algorithm. Algorithm 1 is the pseudocode of our method, where the learning rate of an optimization algorithm such as stochastic gradient descent is denoted as η, the total number of residual blocks in the residual network is denoted as L, the number of residual blocks deleted at one time is denoted as K, the total number of target residual blocks is denoted as L', and the number of retraining iterations is n.

In Algorithm 1, indicating the importance of residual blocks is first initialized for all residual blocks (line 1). The network model weights are initialized with uniform random numbers, and the residual network is trained using stochastic gradient descent optimization (line 2). This algorithm gradually removes the residual blocks in the future and imports L (line 3) indicating the current number of residual blocks. The residual blocks are retrained in a loop (lines 4 to 10). Residual blocks are removed during the algorithm loop with a target of the preset number of residual blocks, but residual block removal is no longer performed when the accuracy of the model decreases. The set of I that indicates the index of the residual unit that becomes the object of deletion is set within the loop (line 5). The indexes of the residual blocks judged as unimportant according to the weights are appended to I (line 6). $\omega_i F(x_i)$ corresponding to the index contained therein is deleted from (2), and the deletion of the layer is performed (line 7). Along with the deletion of residual blocks, the current number of residual blocks $L\prime$ is updated (line 8). For the residual network after the deletion of residual blocks, n retraining iterations are performed (line 9). At this point, the initial value of the learning rate η for retraining is the same as the initial learning rate used in line 2.

The algorithm in this paper is shown in the flowchart in Figure 2. Firstly, the variable of residual block importance is introduced to identify whether it is an inherent residual block or an unimportant residual block to delete it, to update the number of residual blocks. If the set number of residual blocks is not reached, then return to the residual block importance judgment for layer deletion. If it is reached, then retraining is performed. If the accuracy rate does not drop, then return to the residual block importance judgment for layer delete. If the accuracy rate decreases, then the whole layer deletion process is ended.

4. Experimental Results and Analysis

4.1. Experimental Settings

4.1.1. Dataset. The CIFAR-10/100 [28] and ImageNet datasets [29] are used as the experimental validation datasets. CIFAR-10/100 consists of image with 10/100 classifications per dataset. The image size is $32 \times 32 \times 3$. ImageNet is a dataset with 1000 classifications, and the image size is $224 \times 224 \times 3$. In this experiment, a $224 \times 224 \times 3$ single-center crop is applied to the images during training and testing with reference to the literature [30]. Furthermore, as in the literature [31, 32], in this paper, color and proportional aspect ratio enhancements are applied as data enhancements during the training process.

4.1.2. Model. In this evaluation, the residual network model experimented refers to [30] with three convolutional layers for each residual block and combines batch normalization and activation function ReLU. The number of residual blocks is assumed to be 3 for CIFAR-10/100 and 4 for ImageNet. Thus, as proposed by He [3], the number of layers is 56 for CIFAR-10/100 and 50 for ImageNet, and the dimensionality of the residual units changes using a projection scheme [32, 33] at the time of the change of residual blocks.

4.1.3. Hyperparameters. For hyperparameters, according to the settings used in the standard image classification [22], the optimizer uses SGD with modulus of 0.9, with an initial learning rate of 0.1 and 200 iterations. The learning rate in CIFAR-10/100 is 0.81. The batch setting is 128 in CIEAR10/100 and 512 in ImageNet.

4.1.4. Baseline. The baseline solution is model compression using residual networks without model compression and distillation [14, 22]. As described in a related study, model distillation dynamically skips layers compared to the static removal of layers to reduce computational cost while maintaining accuracy. In addition, model distillation, unlike other methods, can directly specify the number of layers and is therefore easy to use in situations such as hardware memory limitations, and its model use is consistent with the structure of the residual network model used in this paper.

4.1.5. Implementation Platform. In this paper, the Keras and TensorFlow [13] are used to implement the designed model and baseline. In addition, CUDA and CUDNN libraries are used for GPU accelerated training to implement the stacked capsule network model. The platform was trained using an Intel i7-10500U Processor, 2.7 GHz, 3M processor speed, 8 GB RAM, 1 TB hard disk, Nvidia GeForce GPU for the system.

4.2. Experimental Results

4.2.1. Results of the Proposed Method. The residual block of (2) is employed in the residual network of our method; however, the representation of equation (1) is used in the residual unit transformation. The performance comparison between the deleted residual blocks and the original residual network is given in Table 1. A residual block has 3 convolutional layers, and it can be found that even after some residual blocks are deleted, the accuracy on each dataset does not abate but increases; thus, for model compression, the deletion of neural network layers can be performed, which can effectively avoid overfitting. The number of retraining iterations n is set to be the same as the original training iterations, but in this experiment, even if n is set to be smaller than the original learning iterations, the accuracy can be maintained to some extent. Table 1 compares the number of remaining network layers in each residual block before and after removing the network layers, and it can be seen that the first residual block retain the least number of network layers. In turn, this allows the model to be compressed substantially and effectively.

```
Initialization: learning rate $\eta$, number of residual units $L$, number of residual units removed at one time is $k$, number of retraining iterations is $n$.
 (1) ResNet's parameters and weights $\{\omega_i: L \in [L]\}$ are initialized.
 (2) Set the initial learning rate $\eta$ to train ResNet.
 (3) $L' = L$
 (4) while $L' > L$ do
 (5) $I = \theta$
 (6) Select $k$ in ascending order of the absolute value of $\omega_i$ and add that index to $I$
 (7) Delete $\omega_i F(x_i)$ of $i \in I$ from equation (2)
 (8) $L' \leftarrow L' - k$
 (9) Retraining in SGD with initial learning rate for $n$ iterations
(10) End
```

Algorithm 1: Our method.

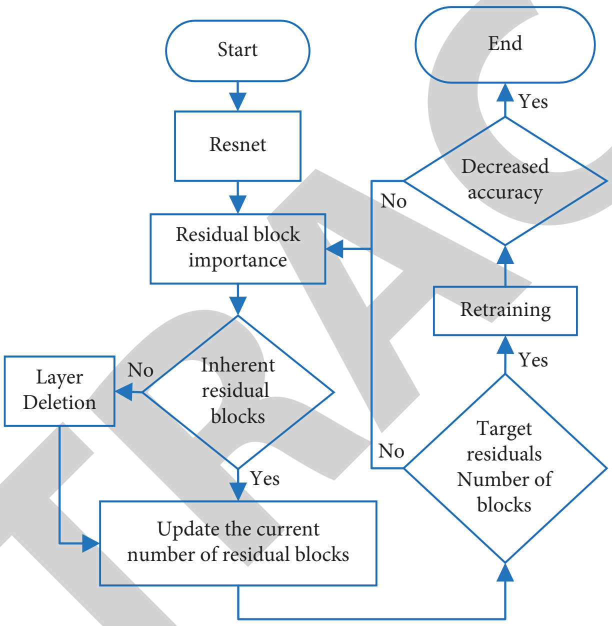

Start
End
Yes
Resnet
Decreased accuracy
No
Residual block importance
Retraining
Yes
Layer Deletion
No
Inherent residual blocks
Target residuals Number of blocks
No
Yes
Update the current number of residual blocks

Figure 2: The flowchart of our algorithm.

Table 1 represents the number of remaining layers in each residual block of the minimum residual network where no accuracy degradation occurs, and according to Table 1, it can be seen that the proposed method can remove different layers of network layers in each residual block. According to the results of this paper, the first residual block, i.e., the residual block closest to the input, has the least number of layers; i.e., the most layers can be removed, and for the latter residuals blocks, some of the network layers can be also removed separately without affecting the final accuracy. Moreover, according to [13, 14], the first residual can be said to have little impact on the accuracy even if the network layers are reduced.

4.2.2. Accuracy. This subsection evaluates the image classification accuracy of the test data when the number of layers of the residual network is changed; however, the accuracy of the validation data is evaluated following the customary evaluation since the ImageNet dataset does not exist for the test data. The comparison method is a residual network varying the number of network layers from 56 to 11 in CIFAR-10/100. Six residual blocks are employed in each dataset, and each residual block has 18, 15, 12, 9, 6, and 3 layers respectively. In ImageNet classification training in this paper, the method is changed from 50 to 17 layers for the residual network and distillation method using 50, 34, and 18 layers of the model as a comparison object.

The experimental results are visualized in Figure 3 with the graphs of the number of network layers and the accuracy of image classification. The blue dashed line indicates the accuracy of the residual network with different number of layers, the green dashed line indicates the accuracy of the residual network with different number of layers learned by distillation, and the red dashed line shows the initial accuracy of the proposed method. The proposed method maintains the initial accuracy indicated by the red dashed line while the layers are deleted. Eventually, the proposed

Table 1: The number of remaining layers of each residual block and accuracy before and after layer deletion.

Dataset	Layer number	Accuracy (%)	1st residual block Layer number	2nd residual block Layer number	3rd residual block Layer number	4th residual block Layer number
CIFAR-10	56	92.88	18	18	18	None
	32	93.05	3	15	12	None
CIFAR-100	56	71.83	18	18	18	None
	35	71.99	3	12	18	None
ImageNet	50	75.89	9	12	18	9
	38	76.12	6	6	15	9

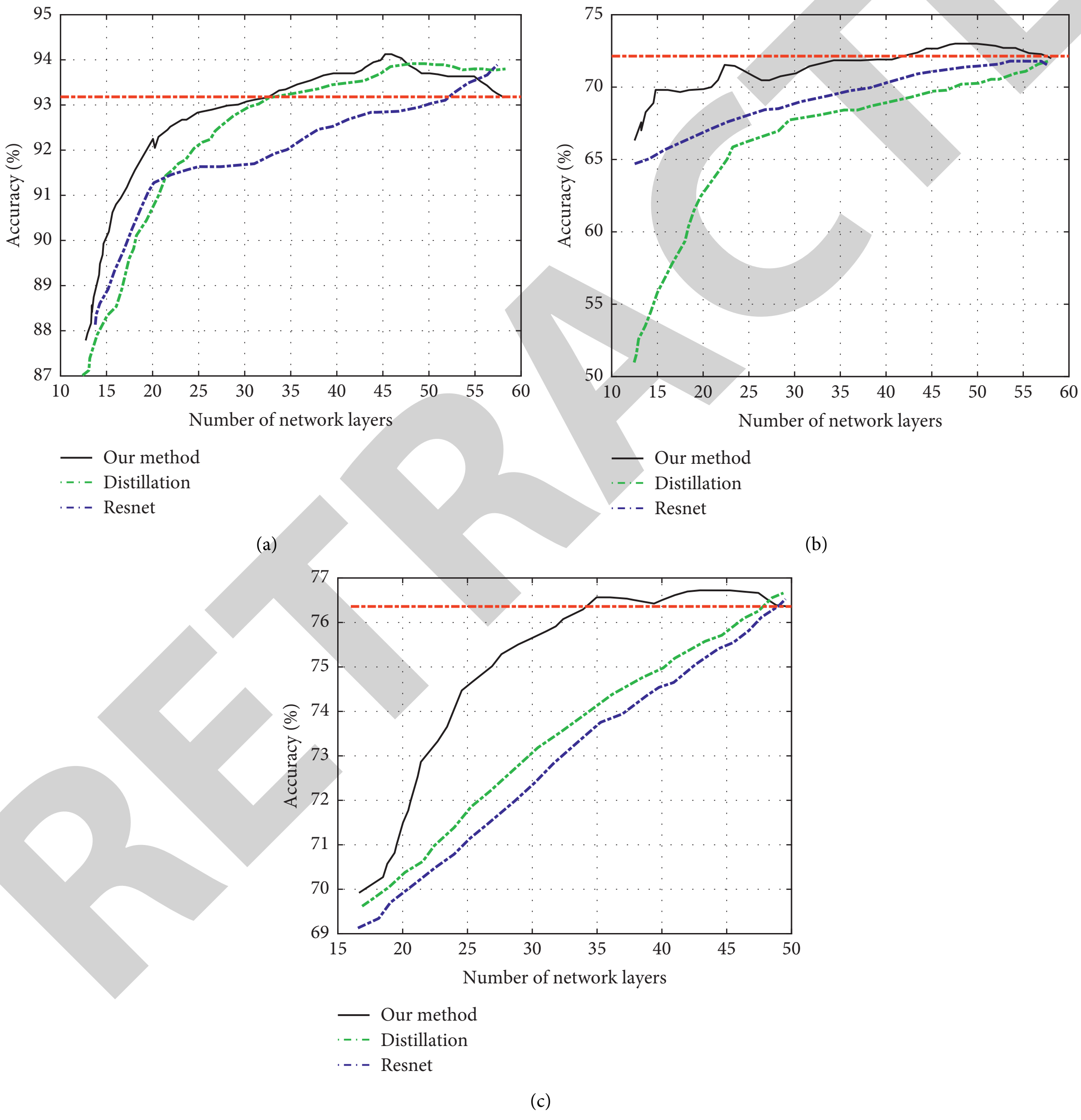


Figure 3: Accuracy of different network layers in different methods: (a) CIFAR-10; (b) CIFAR-100; (c) ImageNet.

method maintains the accuracy of 32, 35, and 38 layers in CIFAR-10/100 and ImageNet, respectively, which are the numbers of layers corresponding to the intersection of the black and red dashed lines. That is, 42.86%, 37.50%, and 24.0% of the network layers were removed in CIFAR-10/100 and ImageNet, respectively. Comparing the blue dashed line and the black line, compared with the usual case of the residual network with layer change learning, the model distillation achieves essentially the same accuracy as the method in this paper in CIFAR-10 but cannot maintain the accuracy in CIFAR-10/images; on the other hand, the method in this paper can also maintain the accuracy while cutting the number of layers in CIFAR-100 and ImageNet. The reason why the method in this paper maintains the original accuracy can be attributed to the fact that a priority term is introduced in this paper to identify the importance of each neural network layer, and then the unimportant layers are selected to be removed during the training process based on the priority of the neural network layers. In addition, the network model is also retrained to avoid accuracy degradation when the network layers are removed.

4.2.3. Calculating Cost. This section evaluates the computational cost of the minimum residual network for which the method in this paper is able to maintain accuracy, as an evaluation metric, the execution time of sequential propagation used in the MAC model [13, 18] inference calculations using the representation of the product and the number of operations, the execution time of back-propagation, and the number of parameters of the model. MAC is calculated using the convolutional layer and the mean of all combined layers. The execution time is evaluated as the average of 100 execution times. The experimental setup in the study is the same as in the previous section. The proposed method can reduce the number of layers while maintaining high accuracy to 32, 35, and 38 layers in CIFAR-10/100 and ImageNet, respectively, and therefore these models are evaluated above utilizing price metrics.

Table 2 indicates the evaluation results. Regarding the number of MACs, it was cut to 60.93%, 62.89%, and 78.59% in CIFAR-10/100 and ImageNet, respectively. The execution time of sequential propagation was cut to 60.23%, 70.13%, and 76.69%, respectively. The execution time of reverse propagation was cut to 59.71%, 60.44%, and 78.9%. In terms of quantity, without accuracy degradation, the number of parameters can be cut to 69.82%, 90.50%, and 93.15%, respectively. These experimental results demonstrate that the proposed method speeds up the model inference computation without increasing the memory consumption and without accuracy degradation. In Section 1.1, it is elaborated that the dynamic layer skipping maneuver tends to maintain accuracy but increases memory consumption; conversely, the static layer deletion maneuver cuts memory consumption but decreases accuracy. When compared in a minimal residual network that does not cause the accuracy degradation caused by the proposed method, the proposed method improves all computational cost metrics without accuracy degradation.

4.2.4. Hyperparameter Dependence. This section explores the relationship between the number of iterations of retraining and the accuracy of the number of residual blocks removed for the hyperparameters of the methods in this paper.

For 56-layer residual network learned in CIFAR-10, the accuracy versus the number of network layers when the number of iterations to be retrained is 30, 60, and 120 is illustrated in Figure 4. It can be known that the accuracy is easily maintained when the number of zones is high, but it can also be maintained to some extent when the number of iterations is low. In this method, even in the case of 30 iterations of retraining, the accuracy can be maintained to some extent while deleting layers in order to be used as the initial value for retraining, and retraining can be started from the model with a certain degree of high accuracy from the beginning. Figure 4 shows that the accuracy is still good when the number of iterations of retraining is fixed at 60 and the number of deleted residual blocks can be 1, 2, or 4. Notice that when $n = 4$, 44, 32, 20, the smaller the $n = 4$, the greater the improvement in accuracy through retraining. As shown in [3], the smaller the number of residual blocks removed, the smaller the decrease in accuracy, which can be considered as the reason why retraining can be started from the beginning with a certain degree of high accuracy. In addition, the minimum residual network that was able to maintain the initial accuracy was 32 layers with $n = 2$. The residual network with 44 layers with $n = 4$ had an accuracy of 92.84%, achieving essentially the same accuracy as the preliminary accuracy of 92.88%.

4.2.5. Weighted Regularization Effect. In the proposed method, the absolute value of ω_i is used as an indicator of the importance of the residual blocks; i.e., if the absolute value of ω_i is small, the output of $F(x_i)$ is reduced and the result $\omega_i F(x_i)$ becomes smaller and has a smaller impact on the overall output. The average relationship between the absolute value of ω_i and the size (L2) on the validation data for the 56-layer residual network learned in CIFAR-10 can be visualized in each residual block, as shown in Figure 5. The smaller the importance in Figure 5, the smaller the weight of the output. It can be known that the smaller the absolute value of ω_i, the smaller the L2 parametric number of $\omega_i F(x_i)$, so it can be said that the absolute value of $\omega_i F(x_i)$ can be used as an indicator of the importance of the residual unit according to ω_i reduction $F(x_i)$.

4.2.6. Weight Distribution. In this paper, the residual network is trained on the CIFAR-10 dataset, and the weight distribution of the residual blocks is visualized to investigate the nature of the "importance" introduced in the proposed method. Figure 6 demonstrates the importance histogram of the residual blocks for the residual network trained with CIFAR-10. Initialized with uniform random numbers, the trained shape becomes a mixture of two Gaussian distributions. In element-level parameter removal, the histogram of known learning parameters is monotonously normally distributed with a mean around 0 [12]. Since the absolute value of the parameter is used as importance in element-level parameter deletion, the most frequent values are around 0. Many parameters are judged to be unimportant, and parameters close to 0 are considered to have little impact on the output even if they are deleted, and can be retrained with

Table 2: Calculation cost in model reasoning.

Dataset	Layers number	Accuracy (%)	MAC	Sequential propagation (msec)	Reverse propagation (msec)	Parameters
CIFAR-10	56	92.88	8.19 × 107	6.584	12.93	585.9K
	32	93.05	4.99 × 107	3.970	7.721	409.1K
CIFAR-100	56	71.83	8.65 × 107	6.203	13.36	613.6K
	35	71.99	5.44 × 107	4.350	8.075	555.3K
ImageNet	50	75.89	4.11 × 109	29.95	59.51	25.55M
	38	76.12	3.23 × 109	22.97	46.53	23.80M

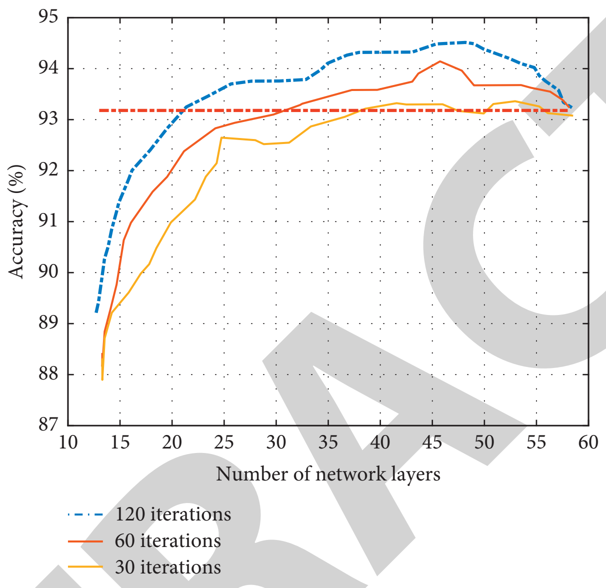
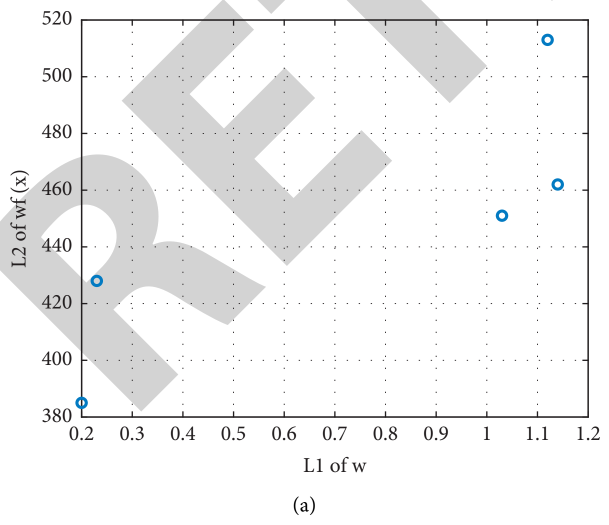


Figure 4: The change of the accuracy of deletion and retraining of ResNet layer in CIFAR-10 training.



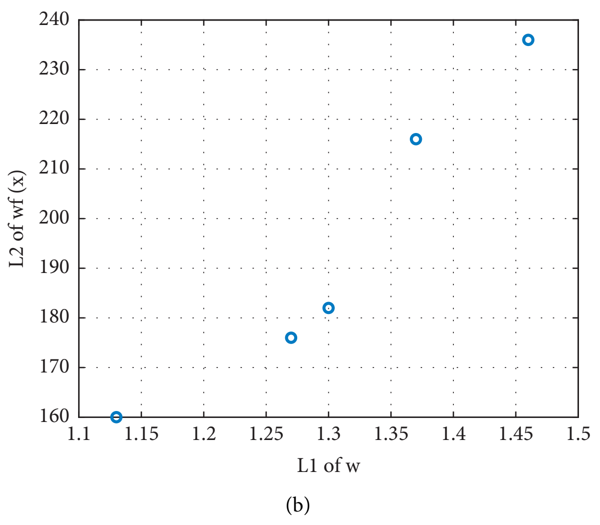
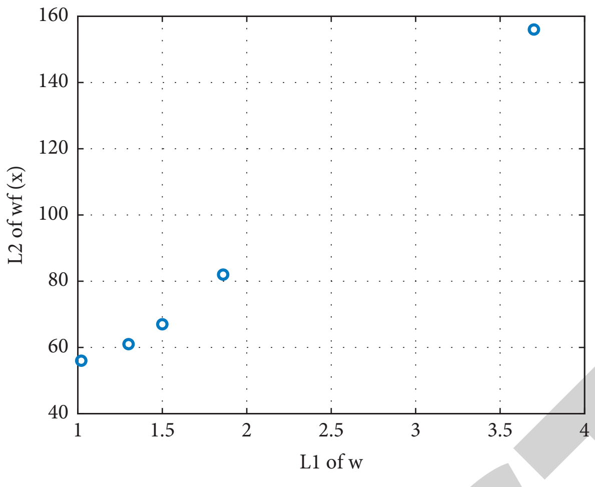


Figure 5: Continued.



(c)

Figure 5: The average distribution of the weight of each residual block of ResNet in CIFAR-10: (a) CIFAR-10; (b) CIFAR-100; (c) ImageNet.

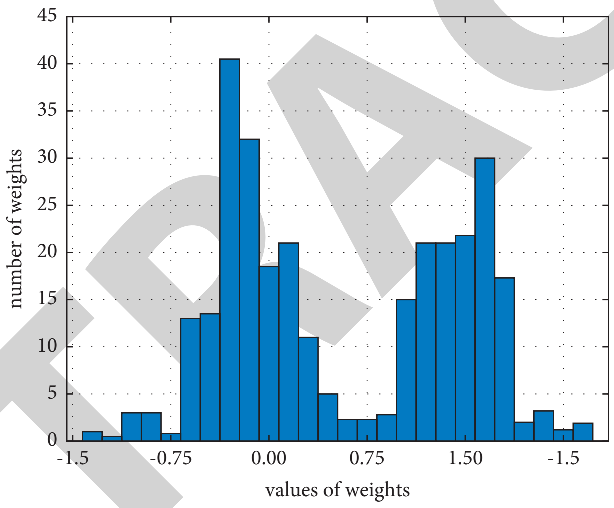


Figure 6: Weight histogram of residual unit in CIFAR-10.

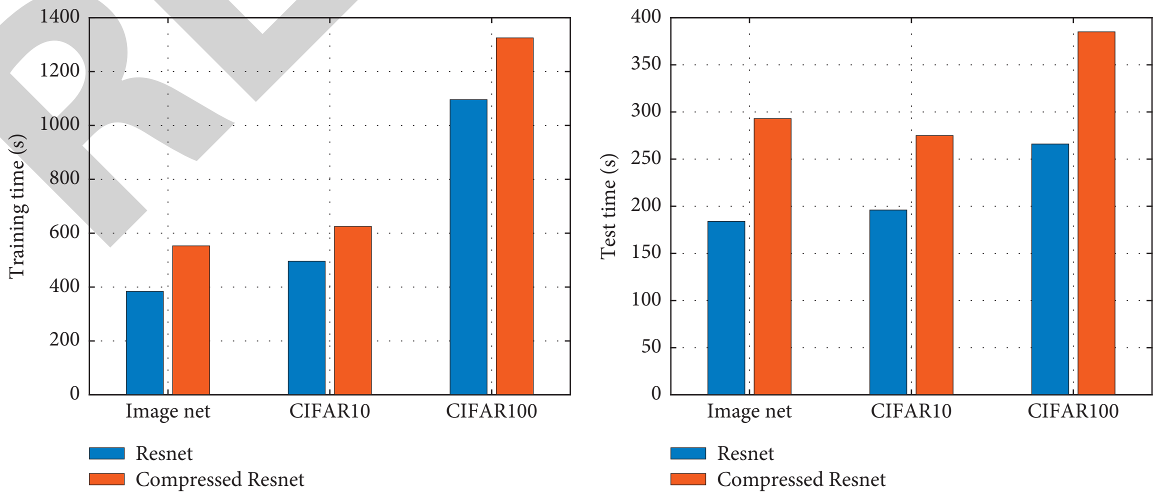


Figure 7: Time spent on (a) training stage and (b) testing stage.

Table 3: The number of model parameters before and after compression.

Models/model parameters	CIFAR-10	CIFAR-100	ImageNet
ResNet	585.9K	613.6K	25.55M
Compressed ResNet	409.1K	555.3K	23.80M

small learning rates to maintain accuracy as in [12]. On the other hand, the histogram of cascading parameter deletion is like two normal distributions symmetrical around 0, as shown in Figure 6. If we take the absolute value as the importance in this paper, the most common value is not 0, but around 0.7. In other words, when layer deletion is performed with the proposed method, there are cases when some network layers with higher importance must be deleted. In this case, the model needs to be modified substantially to maintain the accuracy of the model, so a larger learning rate needs to be set during the retraining process.

4.2.7. Model Complexity Analysis. Figure 7(a) indicates the time spent in the training phase for the original ResNet and the compressed ResNet of this paper. It can be clearly seen that the training time of the compressed ResNet in this paper is the least on all three different datasets, which indicates that the model complexity is lower than that of the original ResNet. In the testing phase indicated in Figure 7(b), the compressed ResNet in this paper takes even less time. These results show that the compressed ResNet in this paper has faster recognition speed for both image samples at the end of the training phase. The numbers of parameters of the original ResNet and the compressed ResNet in this paper are shown in Table 3, and the number of parameters of the compressed ResNet in this paper is very much smaller than that of the original ResNet, which indicates that the model overhead cost of the compressed ResNet in this paper is lower than that of the original ResNet. This indicates that the compressed ResNet in this paper can be flexibly and quickly deployed on existing hardware.

5. Conclusions

In order to reduce the computational cost of inference, this study proposes a method to reduce the number of residual network layers without reducing the accuracy. The proposed method has a scalar parameter to identify the insignificant residual units, based on which the residual units are selected and removed. In the image classification task of CIFAR-10/100 and ImageNet, the number of network layers is effectively removed for model compression while maintaining accuracy compared to existing methods.

Data Availability

The dataset used in this paper are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding this work.

Acknowledgments

This work was supported by the Natural Science Foundation of China (Grant no. 61976098) and Science and Technology Development Foundation of Quanzhou City (Grant no. 2020C067).

References

[1] C. Zhang, T. Xie, and K. Yang, "Positioning optimisation based on particle quality prediction in wireless sensor networks," *Networks, IET*, vol. 8, no. 2, pp. 107–113, 2019.

[2] Y. X. Zou, J. S. Yu, Z. Chen, J. Chen, and Y. Wang, "Convolution neural networks model compression BasedOn feature selection for image classification," *Control Theory & Applications*, vol. 34, no. 6, pp. 746–752, 2017.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, Hilton Head, SC, USA, 15-15 June 2000.

[4] Z. Ye and S. Xiao, "Compression of convolutional neural network applied to image classification," *Journal of Beijing Information Science and TechnologyUniversity*, vol. 33, no. 3, pp. 52–56, 2018.

[5] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural network," in *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, Montreal, Canada, December 2015.

[6] M. Müller, J. Finke, L. Stahl, Y. Tong, H. Fricke, and T. Vallee, "Development and validation of a compression flow model ofnon-Newtonian adhesives," *The Journal of Adhesion*, vol. 2021, no. 4, pp. 1–38, 2021.

[7] E. L. Denton, W. Zaremba, J. Bruna, Y. Lecun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluationp," in *Advances in Neural Information Processing Systems*, Montreal, Canada, December 2014.

[8] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weiberger, and Y. Chen, "Compressing neural networks with the hashing trick,"vol. 37, pp. 2285–2294, in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, vol. 37, pp. 2285–2294, PMLR, Lille, France, July 2015.

[9] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnornet: Imagenet classification using binary convolutional neural networks," in *Proceedings of the European Conference on Computer Vision*, pp. 525–542, (ECCV), Amsterdam, Netherlands, October 2016.

[10] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," in *Proceedings of British Machine Vision Conference (BMVC)*, Nottingham, UK, September 2014.

[11] S. Han, J. Pool, J. Tran, and W. J. Dally, "Mcdnn: an approximation-based execution framework for deep stream processing under resource constraints," in *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 123–136, Association for Computing Machinery, New York, NY, USA, 2016.

[12] Z. Liu, J. Li, and Z. Shen, "3DCNN-DQN-RNN: a deep reinforcement learning framework for semantic parsing of large-scale 3D point clouds," in *Proceedings of the IEEE international conference on computer vision*, pp. 5678–5687, Venice, Italy, 2017.

[13] G. Huang, Z. Liu, K. Q. Weinberger, and L. V. D. Maatren, "Densely connected convolutional networks," in *Proceedingsof the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4700–4708, Honolulu, HI, USA, July 2017.

[14] M. Lotfollahi, M. J. Siavoshani, R. Shirali, H. Zade, and M. Saberian, "Deep packet: a novel approach for encrypted traffic classification using deep learning," *Soft Computing*, vol. 24, no. 3, pp. 1999–2012, 2020.

[15] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, "Quantized convolutional neural networks for mobile devices," in *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, pp. 4820–4828, Las Vegas, NV, USA, June 2016.

[16] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pp. 1094–1103, Barcelona Spain, December 2016.

[17] S. Changpinyo, M. Sandler, and A. Zhmoginov, "The power of sparsity in convolutional neural networks," 2017, https://arxiv.org/abs/1702.06257.

[18] H. Zhou, J. M. Alvarez, and F. Porikli, "Less is more: towards compact cnns," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 662–677, Amsterdam, Netherlands, October 2016.

[19] S. Han, H. Mao, and W. J. Dally, "Deep compression: compressing deep neural network with pruning, trained quantization and huffman coding," in *Proceedings of International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico, May 2016.

[20] M. Courbariaux and Y. Bengio, "Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1," 2016, https://arxiv.org/abs/1602.02830.

[21] Y. Guo, A. Yao, and Y. Chen, "Dynamic network surgery for efficient dnns," in *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pp. 1379–1387, San Francisco, CA, USA, December 2016.

[22] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, May 2015.

[23] J. Jin, Z. Yan, K. Fu, N. Jiang, and C. Zhang, "Neural network architecture optimization through submodularity and super modularity," 2016, https://arxiv.org/abs/1609.00074.

[24] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proceedings of International Conference on Learning Representations (ICLR)*, Toulon, France, April 2017.

[25] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," in *Proceedings of International Conference on Learning Representations (ICLR)*, Toulon, France, April 2017.

[26] R. Tkcvhenko and I. Izonin, "Model and principles for the implementation of neural-like structures based on geometric data transformations," in *International Conference on Computer Science, Engineering and Education Applications (ICCSEEA)*, pp. 578–587, Kiev, Ukraine, January 2018.

[27] I. Izonin, R. Tkachenko, N. Kryvinska, P. Tkachenko, and M. Greguš, "Multiple linear regression based on coefficients identification using non-iterative SGTM neural-like structure," in *International Work-Conference on Artificial Neural Networks (IWANN),2019:Advances in Computational Intelligence*, pp. 467–479, Gran Canaria, Spain, June 2019.

[28] L. Wu, *Numerical Methods for Pregularization Problems*, Ph.D dissertation, College of Mathematics and Econometrics of Hunan University, Changsha, China, 2013.

[29] J. Zhu and T. Hastie, "Classification of gene microarrays by penalized logistic regression," *Biostatistics*, vol. 5, no. 3, pp. 427–443, 2004.

[30] A. Londhe, R. Rastogi, A. Srivastava, K. Kiran, K. M. Sirasala, and K. Komal, "Adaptively accelerating FWM2DA seismic modelling program on multi-core CPU and GPU architectures," *Computers & Geosciences*, vol. 146, Article ID 104637, 2021.

[31] B. Hla, C. Xh, B. Zya, and J. Luo, "Noise-robust image fusion with low-rank sparse decomposition guided by external patch prior - ScienceDirect," *Information Sciences*, vol. 523, pp. 14–37, 2020.

[32] N. Srivastava, G. Hinton, A. Krizhevsky, S. Ilya, and S. Ruslan, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[33] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pp. 1269–1277, Montreal, Canada, January 2014.

[34] G. Huang, D. Chen, T. Li, F. Wu, L. V. D. Maaten, and K. Q. Weinberger, "Multi-scale dense convolutional networks for efficient prediction," 2017, https://arxiv.org/abs/1703.09844.