A Novel Dynamic Network Pruning via Smooth Initialization and Its Potential Applications in Machine Learning Based Security Solutions

LEI WU\textsuperscript{1}, HAN YUE\textsuperscript{1}, PENG CHEN\textsuperscript{2}, DE-AN WU\textsuperscript{1}, AND QIU JIN\textsuperscript{1}

\textsuperscript{1}School of Mathematical Science, University of Electronic Science and Technology of China, Chengdu 611731, China
\textsuperscript{2}School of Computer and Software Engineering, Xihua University, Chengdu 610039, China

Corresponding author: Peng Chen (chenpeng@mail.xhu.edu.cn)

This work was supported by the Science and Technology Program of Sichuan Province under Grant 2016CC0016, Grant 2015GZ0119, and Grant 2012GZX085.

ABSTRACT

With the development of artificial intelligence technology, the demand for new digital security and privacy solutions is growing importantly. Inspired by the synaptic pruning in mammalian brains, we develop a network pruning method, called dynamic network pruning (DNP) method, which reduces the necessary number of free parameters in the convolutional neural networks. The DNP can be perfectly integrated with the gradient descent training and performed at any point, even multiple times, during training. We show that the pruning of connections (filters) is more intrinsic than the pruning of neurons (channels), and we relate the significance of filters to the dispersion of their values. With our proposed weight initialization technique, called smooth initialization (SI), unimportant filters can be easily identified based on the simple thresholds. The DNP method does not need pre-training to learn the connectivity of the network nor requires a very long time of fine-tuning to restore the performance. The DNP can also lead to better data privacy under distributed environment due to improved learning efficiency and convergency. The experiments show that our method outmatches several weight reduction methods in terms of reduction ratio and test accuracy on various models and datasets, and the generalization abilities of the pruned models are not damaged.

INDEX TERMS

Network pruning, weight initialization, neural network, parameter reduction, information security.

I. INTRODUCTION

In the past decades, neuroscientists have discovered that during the development of mammalian brains, mass neurogenesis is followed by regressive events called apoptosis and synaptic pruning [1]–[3]. Apoptosis [4] is the genetically programmed death of neurons that have failed in the competition with other neurons to connect with predetermined targets. Synaptic pruning [3] is to delete synapses that are inappropriately functional, and it eventually leads to the loss of about half of the synapses in human brains. The pruning of synapses is largely determined by environmental influences and is considered to represent the actual process of learning [2]. Although the deep mechanisms of apoptosis and synaptic pruning are being discussed [5] and more secrets about human brains are yet to be discovered, the concept of synaptic pruning is highly valuable for one of the most thriving machine learning algorithms, i.e. neural networks [6].

When neural network models are getting better at addressing dedicated tasks, their sizes increase dramatically thus require much more computational resources. Therefore, the reduction of necessary parameters in deep models has attracted researchers’ attention. In convolutional neural networks (CNNs), the connections correspond to the convolution kernels and the neurons correspond to the feature maps or channels [7]–[9]. The apoptosis and synaptic pruning are both feasible in CNNs. Dropping unimportant neurons (along with all their connections to other neurons) is the simulation of apoptosis, which is often called channel pruning. To simulate synaptic pruning, certain connections need to be removed, while the neurons are kept intact. This process is often referred to as filter pruning. Sometimes the accumulation
of synaptic pruning eventually leads to apoptosis. Because when all input/output connections of a neuron are pruned, the neuron will no longer participate in the forward/backward computation. We will show later that the importance of neuron is not as obvious as that of connection. Therefore, the critical part for network pruning is to decide which connections are unimportant, such that it does not lead to dramatic degradation of performance.

FIGURE 1. DNP significantly reduces the number of connections and neurons in neural networks.

In this paper, we propose the on-the-fly network pruning method called dynamic network pruning (DNP) that reduces the number of parameters in CNNs by removing unimportant connections during a one-time training (as shown in Figure 1). The pruning process can be performed at any point during training and even multiple times. Neither a very long time of pre-training nor retraining (fine-tuning) is needed. We also propose a weight initialization method called smooth initialization (SI) which is extremely helpful in distinguishing the important connections from the unimportant ones.

Experimental results show that SI leads to higher accuracy and allows DNP to outperform several state-of-the-art parameter reduction methods. We will also show that the generalization abilities of pruned models are not damaged by DNP. The novel method can lead to better data privacy under distributed environment due to improved learning efficiency and convergence, which has a great potential prospect in the field of healthcare [10]–[12].

The paper is organized as follows. Section 2 gives a brief survey regarding the parameter reduction methods in neural networks. Section 3 describes DNP method in details. Section 4 presents multiple comparison experiments with other methods. Further discussions regarding the method and the results are given in Section 5 and the conclusion is drawn in Section 6.

II. RELATED WORK

A great amount of efforts has been made from different perspectives to reduce the parameters in CNNs while inflicting the minimal impacts on the performances. Based on the pruning space, current methods can be roughly categorized into three areas.

A. CHANNEL PRUNING

These methods directly prune unimportant neurons from the network. For example, Hu et al. [13] considered that a neuron was redundant if most of its outputs were zeros. He et al. [14] designed a pruning method for pre-trained models and adopted different pruning strategies for branching architectures. Molchanov et al. [15] proposed to iteratively prune neurons on pretrained models based on a Taylor expansion criterion. Yu et al. [16] proposed to prune neurons based on the reconstruction error produced by the flatten layer. Hu et al. [17] approximated the Hessian matrix at each layer to determine the sensitivities of neurons. He et al. [18] proposed a learning strategy based on reinforcement learning to select unimportant neurons. Luo and Wu [19] designed an independent pruning layer that could be integrated into the training process. Game theoretic based random channel pruning for better defense against adversarial examples was studied as well [20].

B. FILTER PRUNING

These methods aim to prune redundant connections (entire convolution kernels) in CNNs. In 2015, Han et al. [21] proposed a network pruning method to pre-training stage to determine which connections were unimportant and required a very long time of retraining. Li et al. [22] proposed to prune those kernels whose element sums were relatively small. Luo et al. [23] considered that the importance of a kernel should be determined by the out-put of the next layer, rather than its own layer. He et al. [24] proposed to allow the pruned kernels to continue to be updated and used the same pruning ratio in all layers.

C. FINE-GRAINED PRUNING

We include two types of techniques into this category because they are often combined to increase the training speed in distributed systems: (1) weight/gradient quantization that converts parameters or their gradients to lower bits (e.g. from floats to integers) [25]–[28] and (2) weight/gradient smartification that approximate the parameters or gradients with sparse tensors [29]. Xu et al. [30] proposed a weight quantization method that took layer-depth information into consideration. Lin et al. [31] proposed Deep Gradient Compression that took advantage of multiple techniques including gradient smartification, local gradient clipping and warm-up training and achieved impressive reduction ratio on popular deep models.

D. OTHER STRATEGIES

There have been many tensor factorization-based methods that provide profound mathematical support for network pruning. For example, Jaderberg et al. [32] proposed to approximate the kernels by a linear com-bination of a smaller basis set of kernels. The approximation was based on an iterative scheme and designed for pre-trained models. Tai et al. [33] proposed to directly solve the global optimum for low-rank tensor decomposition. Yet, low-rank constrained CNNs needed to be trained, which made it hard to directly integrate with common training procedures. Network pruning during the evaluation stage has been studied. Lin et al. [34]
proposed to prune at runtime based on Markov decision process while models were in fact kept intact. Pruning that takes place on higher levels than neurons has also been studied. For example, Liu and Deng [36] proposed to prune modules that consist of one or several layers and solved network pruning problem by architecture search via reinforcement learning.

Li et al. [37] proposed to merge non-tensor and tensor layers together to obtain compact mobile-friendly networks. Other approaches such as teacher-student networks and hardware accelerators have been discussed by Chen et al. [38].

In this work, we aim to develop a workflow that’s capable of removing unimportant connections during the normal training of CNNs. The approach is pruning-based and designed for filter-level pruning. We use the standard deviations independently calculated within each kernel as the criterion to determine the importance of weights. Therefore, the approach has no effect on the architecture or training process and can be performed at any point or even multiple times during training.

III. METHOD

Our approach consists of two steps: (1) smooth initialization and (2) dispersion-based pruning during training. Smooth initialization of the weights is crucial for the successful pruning. The pruning process can be performed at any point during the gradient descent training. The overall procedure of DNP is described in Figure 2.

A. WEIGHT INITIALIZATION

We propose a simple, yet rather practical weights initialization technique called smooth initialization (SI). As indicated by the name, after the weights have been initialized based on certain selected distribution, we apply smooth processing on the initial weights before the training begins. As a result, the weights converge to two subsets after training:

- (1) a small portion of the weights have much varied values (with large dispersion), and (2) most of the weights tend to be flat (with small dispersion). As shown in Figure 3 where $3 \times 3$ Gaussian blur is used for SI, by comparing the 2nd and 4th groups, the number of flat kernels increases (red boxes) and larger dispersion occurs in some kernels (blue boxes) when weights are initialized with SI. To avoid ambiguity in terminology, henceforth kernel refers to the convolution kernel in CNN and filter represents the smooth processing filter used in SI.

The two main factors that affect the final distribution of weights are (1) the selected distribution to generate original weights from and (2) the smooth filter to perform upon the weights. To demonstrate the influence of different distributions and smooth filters on model’s performance, we present a set of experiments in which we train a 3-layer CNN on SVHN [39] dataset. The model has three convolution layers and a flatten layer as hidden layers. We use a convolution layer instead of fully-connected layer as the flatten layer, in which the kernel sizes equal the sizes of input feature maps (as like C5 layer in LeNet-5 [40]). Hence, we can use the same strategies to initialize and prune the flatten layer as previous layers. Detailed network architecture and data preprocessing are described in Appendix A. As shown in Figure 4a, Xavier [41] distribution outperforms He et al. [42] distribution when no smooth processing is performed. However, the performance of his distribution is almost always better than Xavier distribution when smoothed. Amongst all smooth methods, the Gaussian filter with the window size of $3 \times 3$ based on He distribution achieves the best performance.

B. NETWORK PRUNING

It becomes natural to consider that the dispersed kernels may play more important roles in the classification, and the flat ones scarcely contribute to the decision-making when it’s hard for them to extract useful features from the inputs.
We use the standard deviations that are independently calculated within each kernel as the measurement of such dispersion, only to further decide which kernels are unimportant. Specifically, after certain epochs of training, we prune all the kernels with small standard deviations and continue training on the remaining kernels. The pruning may be performed multiple times, and the selection criteria in different layers at different pruning stages may differ.

![Figure 5](image_url) Relations between standard deviation of kernel (α), connection sensitivity (β) and neuron sensitivity (β) in the 3-layer CNN on SVHN. In (a) and (c), the horizontal axes represent that kernels are ordered by α increasingly. In (b) and (d), kernels are ordered by β increasingly, and the curves of β are in fact step functions in which the “width” of each step correspond exactly to its 64 input connections (whose α values are marked by red dots). Better viewed in color.

1) NEURON SENSITIVITY VS CONNECTION SENSITIVITY

Here, we let σ note standard deviation calculated within a single kernel, α denote connection (kernel) sensitivity and β note neuron sensitivity for simplification. α is defined by the accuracy drop inflicted by the pruning of a connection, while β by the pruning of a neuron. When calculating β, we simply remove all its input connections. Therefore, for jth neuron, $\beta_{ij} = \sum_{i=1}^{M} \alpha_{i}^{(ij)}$, where $i$ denotes the ith input neuron, $M$ denotes the total number of input neurons and $\alpha_{i}^{(ij)}$ denotes the connection between input neuron $i$ and output neuron $j$. A larger α or β value indicates higher significance to the decision-making. As pointed out by Morcos et al. [43], the diversity among the importance of neurons was not as evident as traditionally believed. Their experiments suggested that the ablation of highly selective (high-sensitivity) neurons was no more impactful than that of non-selective (low-sensitivity) ones, or put another way, confusing neurons worked almost well with the absence of interpretable neurons. However, the reason behind such observation was not given. In fact, when comparing the difference between α and β in the 3-layer CNN previously trained on SVHN (Figure 5b and Figure 5d, in which input connections are sorted in an increasing order of α), we find that important kernels (higher red curves) tend to have large σ (higher blue dots). There are also a few kernels with relatively large α leads to accuracy increases (negative values). Further correlation analysis (see Appendix B) indicates that kernels with large standard deviations are highly likely to have more influence in the decision-making and the importance of kernel is more intrinsic than the importance of neuron. Therefore, we will use the standard deviation of each kernel as the selection criterion for pruning.

2) PRUNING STRATEGY

Two pruning strategies are applied in DNP in different scenarios:

- **Std threshold.** By defining the standard deviation thresholds, we prune all the kernels whose standard deviations are smaller than the thresholds.
- **Target percentage.** By defining the target percent-ages, we first sort all the kernels according to their standard deviations then remove those with the smallest standard deviations to meet the pruning ratios defined by the target percentages.

Different criteria are allowed in different layers and different pruning stages. The main difference between two criteria is that the std threshold has a more delicate control over which weights are pruned and the reduction ratio may vary in different runs (depending on the actual values of trained weights). However, if a high reduction ratio is more of a concern than the accuracy, the target percentage will be come handier and the reduction ratio maintains the same in different runs.

3) NEURON DROPPING

After pruning the unimportant connections, it becomes reasonable to further drop the neurons that have no input or output connection. Specifically, for any neuron in the hidden layers, if the number of its input/output connections is zero, the neuron is dropped permanently from the network. These neurons are often referred to as dead neurons. The dropping of these neurons further simplifies the connectivity of the network and has no effect on the performance. There exists a rare situation, however, where a stranded neuron would be left out, which is discussed in detail in Appendix C. Moreover, as pointed out by Han et al. [21], the continued training after pruning also guarantees that most of the remaining connections of these dead neurons get automatically disconnected due to the back-propagation training. However, our experiments show that sometimes a few dead neurons may “survive” if the continued training is relatively short. In order to reduce computational costs, the neuron dropping is not performed until the training is finished.

IV. EXPERIMENTS

We evaluate our DNP method on different neural network models (e.g. LeNet [40], ResNets [42], [44] and VG-Gnets [45] et al.) that are trained on different datasets...
Table 1. Comparison of test accuracy and weight reduction ratio with other methods on different models. The abbreviation “c/o” denotes Cutout. The reduction ratio of the low-rank regularization method is approximated based on the $K_1$, $K_2$, $K_3$ values provided by [33]. Top-1 accuracy is used for ImageNet. Note that the letters behind “DNP” are unrelated and only represent different pruning settings. The $T$ times(X) of reduction ratio means that $1 - 1/ T$ of the parameters are removed.

| Model | Method | Accuracy gain | Reduction ratio |
|-------|--------|---------------|-----------------|
| LeNet-5 on MNIST (99.20% [40]) | Network Pruning [21] | +0.03 | 12X |
| | Network Trimming [13] | +0.06 | 4X |
| | DNP | +0.08 | 27X |
| 3-layer CNN on Cifar-10 (87.38% [33]) | Low-rank regularization [33] | -0.48 | 3X |
| | DNP | +0.03 | 8X |
| ResNet-56 on Cifar-10 (93.04% [22]) | Filter Pruning [22] | +0.02 | 1.2X |
| | Channel Pruning [14] | -1.24 | 2.0X |
| | NISP [16] | -0.03 | 1.7X |
| | SFP [24] | +0.31 | 2.1X |
| | AMC [18] | -1.14 | 2.0X |
| | DNP-A | +0.43 | 1.9X |
| | DNP+c/o-C | +0.49 | 3.3X |
| VGG-16 on Cifar-10 (93.25% [22]) | Filter Pruning [22] | +0.15 | 2.8X |
| | Genetic channel pruning [17] | -0.51 | 6.3X |
| | Try-and-learn pruning [47] | -1.08 | 6.0X |
| | DNP+c/o-A | +0.52 | 3.7X |
| | DNP+c/o-C | -0.43 | 7.3X |
| ResNet-34 on ImageNet (73.23% [22]) | Filter Pruning [22] | -1.06 | 1.1X |
| | NISP [16] | -0.92 | 1.8X |
| | SFP [24] | -1.40 | 1.7X |
| | DNP-A | -0.19 | 1.4X |
| | DNP-B | -1.32 | 1.8X |

In comparison with several parameter reduction methods, including network pruning method by Han et al. [21], low-rank regularization [33], Network Trimming [13], Filter Pruning [22], Channel Pruning [14], Genetic channel pruning [17], NISP [16], Soft Filter Pruning (SFP) [24], AMC [18] and Try-and-learn pruning [47]. The impacts on the generalization abilities of pruned models are discussed in Section 5.

We use PyTorch (http://pytorch.org) for software implementation. All models are trained on a single NVIDIA RTX 1080Ti video card. We use the fast implementation of the Gaussian and median filters provided by OpenCV (https://opencv.org) for SI. The for Gaussian in both axes is set to 0.8 (for $3 \times 3$) and 1.1 (for $5 \times 5$), and filter values are regularized to ensure the sum of all elements to be 1.

A. MNIST

We employ DNP on LeNet-5 which is trained on MNIST dataset. The original LeNet-5 model achieved 99.20% test accuracy on MNIST with data distortion. We use the same network architecture used by Han et al. [21], except that we replace the first fully-connected layer (flatten layer) with a convolution layer, while the number of parameters is not varied. SI is performed based on truncated He normal distribution in which values with magnitudes more than twice the standard deviation are dropped and re-generated and $3 \times 3$ Gaussian blur is used. We use Momentum [48] as the gradient descent method with a momentum term of 0.9. The learning rate is set to 0.01. Data distortion is not used in our experiments. The DNP is performed on a 30-epoch training on LeNet-5, in which two pruning stages are carried out after the 10th and 20th epochs, respectively (DNP-A). In the 1st pruning stage, 47.7% of the parameters are pruned and only 1% of accuracy drop is inflicted (test accuracy before pruning is 99.40% and after, 98.55%). After another 10 epochs of training, the test accuracy of the pruned LeNet-5 is restored back to 99.40%. We use higher thresholds to prune 95.0% parameters in the 2nd stage of DNP-A. The final compression ratio is 20X, which is higher than 12X of the pruning method by Han et al. [21]. Accuracy drops to 79.56% immediately after the 2nd pruning, and arrives at 99.29% when the training is finished. The pruning ratio in each layer is shown in Table 2. Weights in the output layer are not pruned, considering they only take up around 1% of all parameters and any modification on them will lead to dramatic impacts on the performance.

If we omit the 1st pruning stage and perform only one pruning stage after the 20th epoch with the exact same std thresholds as in the 2nd stage, more parameters are removed with a reduction ratio of 27X (DNP-B). The test accuracy
TABLE 2. Comparison of pruning ratio with network pruning method by Han et al. [21] in LeNet-5 on MNIST data. Threshold denotes the std thresholds used for pruning and pruned denotes the ratio of pruned parameters. Baseline accuracy of 99.20% [40] is used.

| Method    | Layer  | conv1 | conv2 | flatten | output | Total   | Accuracy gain |
|-----------|--------|-------|-------|---------|--------|---------|---------------|
| DNP-A     | Parameter | 0.5K  | 25K   | 400K    | 5K     | 431K    | +0.20         |
| (stage 1/2) | Threshold | 0.1   | 0.03  | 0.015   | -      | -       | +0.20         |
| DNP-A     | Pruned  | 15%   | 56%   | 47%     | -      | 47.40%  | +0.09         |
| (stage 2/2) | Threshold | 0.15  | 0.05  | 0.03    | -      | -       | +0.09         |
| DNP-B     | Pruned  | 30%   | 85%   | 97%     | -      | 95.00%  | +0.09         |
| DNP-C     | Pruned  | 25%   | 90%   | 98%     | -      | 96.30%  | +0.08         |
| Network Pruning | Pruned  | 33%   | 88%   | 92%     | 81%    | 92%     | +0.03         |

TABLE 3. Accuracy gains and compression ratios of different pruning settings in ResNet-56 trained on Cifar-10. Note that “c/o” stands for Cutout and “d/o” stands for Dropout, in which the dropout rate is 0.5. Baseline accuracy of 93.04% [35] is used. The accuracy gains of +0.56 on the base model benefits from SI.

| Pruning setting | None | A | B | C |
|-----------------|------|---|---|---|
| Target percentage | 0%   | 25% | 50% | 75% |
| ResNet-56       | +0.56 | +0.82 | +0.43 | +0.04 |
| ResNet-56+c/o   | +1.61 | +1.59 | +1.49 | +0.49 |
| ResNet-56+c/o+d/o | +1.47 | +0.39 | +0.25 | +0.44 |
| Reduction ratio | 1X   | 1.3X | 1.9X | 3.3X |

Three different pruning settings are applied to ResNet-56. The performances and reduction ratios when different target percentages are used for ResNet-56 are listed in Table 3. The accuracy of the base model, 93.60%, is higher than the baseline accuracy (93.04%), which is caused by SI. The target percentages for all layers in the residual blocks that are responsible for doubling the channel depths are all set to 25% and parameters in the first and last convolution layers in the network are not pruned, because we find that these layers are more sensitive to pruning. Pruning ratios in other layers are altered simultaneously. Apparently, the test accuracy tends to decrease as more parameters are pruned. We also find that Dropout [52], in this case, is not helpful in increasing the generalization ability of the network. We believe that it is because as more channels are pruned, the sub models selected by Dropout becomes even weaker. However, Cutout [53], on the other hand, is extremely helpful in increasing the generalization ability on Cifar-10. The ResNet-56 model achieves 93.47% test accuracy when nearly half (47.4%) of the parameters are pruned. When trained with Cutout, the accuracy increases to 93.53% when 70.0% of the parameters are pruned, with the reduction ratio of 3.3X.

We also apply DNP to VGG-16 model. The results on Cifar-10 are shown in Table 4. We use the same network architecture used by [22] and use the same training schedule.
as in ResNet-56 above, except that the learning rate is always further scaled by 0.1. Three different pruning settings are applied to VGG-16. Parameters in the first layer are always pruned by half, and the last 6 layers have more redundant weights than others. Specific target percentages for each layer in VGG-16 are presented in Appendix D. Even with only one pass of pruning, the performance degradation is slightly more than 1% at worst. The final test accuracy decreases when more parameters are pruned. When trained with Cutout, the network’s accuracy reaches 93.77% with the reduction ratio of 3.7X (denoted by DNP+c/o-A). When even more parameters are pruned (5.9X, denoted by DNP+c/o-B), the accuracy (93.32%) is still higher than the unpruned model. Detailed analyses regarding the impacts of pruning on ResNet-56 and VGG-16 are given in Section 5.

### C. ImageNet

We perform our DNP method on the ResNet-34 model which is trained on the ILSVRC-2012 training set. Unlike in the Cifar-10 experiments, the average pooling layer is not replaced. Momentum is used, and the momentum term is set to 0.9. The network is trained for a total of 120 epochs. The initial learning rate is 0.1 and scaled by a factor of 0.1 after every 30 epochs and keeps the same since the 61st epoch. No warm restarts are used, because we find that their effects are limited. Two different pruning settings are applied to ResNet-34. A single stage of pruning is performed after the 90th epoch. As pointed out by Li et al. [22], the layers in every residual block respond differently to the pruning. We find that all the residual blocks for mapping the channel dimensions to twice depths and the first layer in the network are the most sensitive to pruning. Therefore, the target percentages in these layers are set to zeros in DNP-A (in Table 1), and no more than 25% in DNP-B. The second layer in each residual block is more sensitive than the first layer. Therefore, in DNP-A, the target percentages for the second layers in rest of the residual blocks are set to 20% and for the first layers, 50%. As for DNP-B, both layers are pruned by 50%. The down-sampling layers, skip connections and the output layer are not directly pruned. Detailed target percentages for ResNet-34 are listed in Appendix E. DNP-A achieves 73.04% top-1 accuracy and 91.31% top-5 accuracy on the ILSVRC-2012 validation set, which are higher than Filter Pruning [22], NISP [16] and SFP [24]. 27.2% parameters are pruned in DNP-A with the reduction ratio of 1.4X. DNP-B achieves 71.91% top-1 accuracy and 90.80% top-5 accuracy with the reduction ratio of 1.8X (44.5% parameters are pruned, which is among the highest in Table 1). The impacts of DNP on ImageNet classification are discussed in Section 5.

### D. Text Classification

We use the CNN model proposed by Kim [54] to perform text classification on the Movie Review Data1 [55], which contains 5331 positive and 5331 negatives processed sentences. The feature extraction layer in the model is the concatenation of three convolution layers with kernel sizes of $3 \times 3$, $4 \times 4$ and $5 \times 5$. All convolution kernels are initialized based on truncated normal distribution with a standard deviation of 0.1. Adam [56] is used as the training method, in which the learning rate is 0.001, and the two moment terms are 0.9 and 0.999. The model is trained for 20 epochs and achieves test accuracy of 74.11%, which is higher than 73.36% reported by Kim [54]. Mass redundancy in the parameters of the trained model is observed. We find that when even all of the $3 \times 3$ and $4 \times 4$ kernels are removed and most of the $5 \times 5$ kernels are pruned, the overall accuracy is only dropped by 1.12%, and fully restored after a very short time of training. It indicates that the classification falls mainly on the $5 \times 5$ kernels. Finally, we prune a total of 91% parameters with the reduction ratio of 11X.

### V. Discussion

Luo and Wu [19] designed an independent pruning layer that could be integrated into the training process. Yet, additional parameters needed to be trained to label the importance of each neuron. Liu et al. [57] also proposed a pruning scheme based on the scaling factors in batch normalization [58] layers and had no effect to the training. How-ever, the method was designed for channel pruning thus did not address the redundancy in connections. Lin et al. [31] proposed Deep Gradient Compression to reduce the band-width exchange in distributed systems and managed to reduce the bandwidth consumption by 277X in ResNet-50. Despite the different goal from our approach, which aims to reduce the number of free parameters in networks them-selves, their method provides an exciting inspiration. It is completely feasible that we first prune the unimportant kernels then perform gradient smartification and binary coding on the remaining parameters to dramatically improve the training speed of pruned models, and we look forward to exploring the potential of such approach in the future.

### A. Importance of Smooth Initialization

We have previously shown that smooth initialization is extremely helpful in identifying the unimportant kernels during training (Figure 3). We also find that in most cases it also improves the performances of CNNs. For example, unpruned LeNet-5 with SI outperforms the baseline by 0.20% and ResNet-56 by 0.56%. When comparing the dispersion-based pruning and random pruning, as shown in Appendix F, we find that SI not only helps the training converge faster.

---

**TABLE 4.** Accuracy gains and compression ratios of different target percentages in VGG-16 trained on Cifar-10. Note that “c/o” stands for Cutout. Baseline accuracy of 93.25% [22] is used.

| Pruning setting | None | A   | B   | C  |
|-----------------|------|-----|-----|----|
| VGG-16          | -0.25| -0.72| -0.95| -1.19|
| VGG-16+c/o      | +0.77| +0.52| -0.07| -0.43|
| Reduction ratio | 1X   | 3.7X| 5.9X| 7.3X|
during the early iterations, but also improves the accuracies of pruned models.

**B. OTHER PRUNING CRITERIA**

As pointed out by Luo *et al.* [23], two common pruning criteria for weights pruning are (1) weight sum [22] and (2) APoZ (average percentage of zeros) [13]. Both are designed to select weights that have many close-to-zero elements. Weight sum is similar to our std threshold, which is meant to examine each kernel independently; and APoZ, as similar to our target percentage, is selecting weights based on their relative amplitudes in the entire layer. Other forms of norm have been used as pruning criteria as well [24]. In the meantime, Ye *et al.* [59] suggested that other criteria rather than weight norms could work better and proposed an ISTA-based method that took advantage of the scaling factor in batch normalization. Essentially, their method, however, was to select neurons with nearly constant outputs during pruning, which resembles our criteria. We use standard deviations independently calculated within kernels as the selection criteria so that weights to be pruned do not necessarily need to be very close to zeros. In fact, weights with constant high values can also be pruned by DNP, which is explained by the comparison between std threshold and weight sum in Appendix G. It is a reasonable strategy since a flat non-zero kernel is yet incapable of extracting useful information from its inputs.

**C. IMPACT ON THE GENERALIZATION**

During our experiments, we observe the following intriguing phenomena. (1) Models trained with Cutout are more prone to be compromised by pruning but have higher accuracies after training. For example, the test accuracy of ResNet-56-A is dropped by 12.79% after pruning while ResNet-56+c/o-A has degraded by as high as 65.53%. However, ResNet-56-A can be easily outmatched by the latter after a few epochs of training. More statistics on ResNets and VGGNets are listed in Appendix D. (2) When we prune 44.5% parameters in ResNet-34 trained on ImageNet and continue training for only 1 additional epoch, we find that the model’s accuracy on the evaluation set surpasses the accuracy obtained from the training set by roughly 3% (69.66%/89.40% on evaluation and 66.68%/86.24% on training). These observations suggest that by pruning low-dispersion weights, the memory of details might be damaged, but the abstract high-level features learned in the latent space are still reserved by the highly dispersed kernels. Therefore, the generalization ability of the pruned model is not compromised by DNP.

**VI. CONCLUSION**

We propose the dynamic network pruning (DNP) method to reduce the redundant weights in convolutional neural networks. The pruning approach does not affect the training process and can be performed at any point during training. With our proposed smooth initialization technique, unimportant kernels can be easily isolated based on simple criterion, i.e. the standard deviation of each kernel, which allows the pruning to be performed multiple times. Experimental results show that DNP outmatches several weight pruning/reduction methods in reduction ratio and test accuracy on various models and datasets.

**APPENDIX**

**A. DETAILS ON SVHN EXPERIMENTS**

Cropped digits are used in our experiments, which are character level $32 \times 32$ RGB images. 531131 additional samples are also used as training data. The images are converted into 1-channel grayscale before feeding into the network. There are 64, 96 and 128 neurons in the three convolution layers, respectively, and 2048 neurons in the flatten layer, which means that there are 6144 kernels in the 2nd layer and 12288 kernels in the 3rd layer. All convolution kernels are $5 \times 5$.

**B. CORRELATION ANALYSIS ON $\sigma - \alpha$ AND $\alpha - \beta$**

If we replott all four subfigures in Figure 5 only in $\alpha - \sigma$ and $\beta - \alpha$ coordinates, we will get Figure 6. In Figure 6a, the Pearson correlation coefficient (or PCC for short) between $\sigma$ and $\alpha$ is 0.792, and the P-value is 0.000 (there are 6144 sample points). In Figure 6c, the PCC between $\sigma$ and $\alpha$ is 0.772 and the P-value is 0.000 (there are 12288 sample points). Therefore, we are confident to say that there is a positive linear correlation between $\sigma$ and $\alpha$ (PCC is close to 1 and P < 0.05). In Figure 6b, the PCC between $\alpha$ and $\beta$ is −0.0264 and the P-value is 0.0385; while in Figure 6d, the PCC between $\alpha$ and $\beta$ is 0.0400 and the P-value is 9.18 × 10$^{-6}$. Therefore, we are confident to say that

**TABLE 5.** Accuracy changes after pruning in ResNet-56 and VGG-16 on Cifar-10. Note that pruning settings (A, B and C) are not shared between ResNet-56 and VGG-16, and “c/o” stands for Cutout.

| Model     | A   | B   | C   |
|-----------|-----|-----|-----|
| ResNet56  | -12.79 | -23.78 | -63.96 |
| ResNet56+c/o | -65.53 | -80.97 | -84.06 |
| Pruning ratio | 1.3X | 1.9X | 3.3X |
| VGG-16    | -23.27 | -30.66 | -59.02 |
| VGG-16+c/o | -37.39 | -56.46 | -76.61 |
| Pruning ratio | 3.7X | 5.9X | 7.3X |

**TABLE 6.** Target percentages in each layer of VGG-16 on Cifar-10.

| conv1 | conv2 | conv3 | conv4 | conv5 | conv6 | conv7 |
|-------|-------|-------|-------|-------|-------|-------|
| A 0% | 50% | 75% | 75% | 25% | 25% | 50% |
| B 0% | 50% | 75% | 75% | 50% | 50% | 75% |
| C 0% | 50% | 75% | 75% | 75% | 75% | 75% |

| conv8 | conv9 | conv10 | conv11 | conv12 | conv13 | conv14 |
|-------|-------|--------|--------|--------|--------|--------|
| A 50% | 50% | 75% | 75% | 85% | 85% | 85% |
| B 75% | 75% | 85% | 85% | 90% | 90% | 90% |
| C 75% | 85% | 90% | 90% | 90% | 90% | 90% |
TABLE 7. Target percentages in each layer of ResNet-34 on ImageNet. The 1st column under each block represents the 1st layer in the residual block, and the 2nd column represents the 2nd layer.

| block1 | block2 | block3 | block4 | block5 | block6 | block7 | block8 |
|--------|--------|--------|--------|--------|--------|--------|--------|
| A      | 0%     | 0%     | 50%    | 20%    | 0%     | 0%     | 50%    | 20%    |
| B      | 10%    | 10%    | 50%    | 50%    | 20%    | 20%    | 50%    | 20%    |

FIGURE 6. Relations between $\sigma$ and $\alpha$, and between $\alpha$ and $\beta$ in the 3-layer CNN on SVHN, in which $\sigma$ is the standard deviation of each kernel, $\alpha$ is kernel sensitivity and $\beta$ is neuron sensitivity.

C. STRANDED NEURONS IN NEURON DROPPING

As shown in Figure 7, after connection pruning, neuron B only has no output connection and is qualified for neuron dropping, while neuron A will survive the neuron dropping. After dropping neuron B, neuron A has no output connection, hence becomes another dead neuron. We refer to neuron A as a stranded neuron, although both neuron A and B have no contribution to the network before/after neuron dropping. To address this situation, we simply need to perform neuron dropping for $k$ times on a $k$-layer network.

D. DETAILS ON Cifar-10 EXPERIMENTS

Accuracy changes inflicted by DNP in ResNet-56 and VGG-16 on Cifar-10 are listed in Table 5. Larger degradation in performance occurs when pruning ratio is higher and networks trained with Cutout are more sensitive to pruning. However, based on Table 3 and Table 4 in the paper, models with Cutout have higher test accuracies when training is finished. It indicates that the memory of details in data might be damaged, but the latent knowledge learned from data are still reserved by high-dispersion kernels.

Target percentages in each layer for different pruning settings for VGG-16 model in Section 4.2 are listed in Table 6. Parameters in the first layer (conv1), flatten layer and output layer are not pruned.

E. DETAILS ON ImageNet EXPERIMENTS

Target percentages in each layer for different pruning settings for ResNet-34 model in Section 4.3 are listed in Table 7. The blocks represent the residual blocks in ResNet-34. The 1st column under each block represents the 1st layer in the residual block, and the 2nd Column represents the 2nd layer. Parameters in the first layer, average pooling layer and output layer are not pruned.

F. IMPORTANCE OF SMOOTH INITIALIZATION

Training loss in the first 500 iterations of ResNet-56 on Cifar-10 is shown in Figure 8. We can see that the loss curves of SI are generally lower in the early training stage, with or without Cutout.
In some literature, random pruning is studied, which does not take the importance of the pruned parameters into account. A randomly pruned model can be considered as a random sub model derived by Dropout. Nevertheless, it is a valid baseline for any well-founded pruning method. By comparing with random pruning, we find that DNP performs better when pruning ratios are higher, and SI improves the test accuracies of pruned models. Again, we use ResNet-56 on Cifar-10 for comparison and the results are shown in Figure 9.

G. STD THRESHOLD VS WEIGHT SUM

Standard deviations and weight sums of the pruned parameters in ResNet-34 trained on ImageNet are illustrated in Figure 10. A clear lower bound can be observed, because the existence of relative higher values is the necessity of larger standard deviations. On one hand, a notable amount of weights with large sums are pruned by our approach; on the other hand, kernels with very low sums cannot have large deviations, which will be left out by our approach. It indicates that by using the std thresholds, more weights are pruned compared to the weight sum criterion.

ACKNOWLEDGMENT

The authors would like to express their gratitude to Dr. Xi-Le Zhao for his support.

REFERENCES

[1] G. Chechik, I. Meilijson, and E. Ruppin, “Neuronal regulation: A mechanism for synaptic pruning during brain maturation,” Neural Comput., vol. 11, no. 8, pp. 2061–2080, Nov. 1999.
[2] F. I. M. Craik and E. Bialystok, “Cognition through the lifespan: Mechanisms of change,” Trends Cognit. Sci., vol. 10, no. 3, pp. 131–138, Mar. 2006.
[3] R. C. Paolicelli, G. Bolasco, F. Pagani, L. Maigi, M. Scianitti, P. Panzanelli, M. Giustetto, T. A. Ferreira, E. Guiducci, L. Dumas, D. Ragozzino, and C. T. Gross, “Synaptic pruning by microglia is necessary for normal brain development,” Science, vol. 333, no. 6048, pp. 1456–1458, Sep. 2011.
[4] S. Elmore, “Apoptosis: A review of programmed cell death,” Toxicologic Pathol., vol. 35, no. 4, pp. 495–516, Jun. 2007.
[5] P. Vanderhaeghen and H.-J. Cheng, “Guidance molecules in axon pruning and cell death,” Cold Spring Harbor Perspect. Biol., vol. 2, no. 6, Jun. 2010, Art. no. a001859.
[6] J. Han, D. Zhang, G. Cheng, L. Guo, and J. Ren, “Object detection in optical remote sensing images based on weakly supervised learning and high-level feature learning,” IEEE Trans. Geosci. Remote Sens., vol. 53, no. 6, pp. 3325–3337, Jun. 2015.
[7] A. K. Sangaiah, D. V. Medhane, T. Han, and M. S. Hossain, “Hossain and G. Muhammad, “Enforcing position-based confidentiality with machine learning paradigm through mobile edge computing in real-time industrial informatics,” IEEE Trans. Ind. Inform., to be published.
[8] X. Cheng, Y. Wang, J. Zhang, B. Xiao, F. Xiao, and J. Zhang, “Multiple kernel coupled projections for domain adaptive dictionary learning,” IEEE Trans. Multimedia, to be published.
[9] Y. H. Zheng, L. Sun, S. F. Wang, J. Zhang, and J. Ning, “Spatially regularized structural support vector machine for robust visual tracking,” IEEE Trans. Neural Netw. Learn. Syst., to be published.
[10] W. Q. Wu, S. Pirbhalul, H. Zhang, and S. C. Mukhopadhyay, “Quantitative assessment for self-tracking of acute stress based on triangulation principle in a wearable sensor system,” IEEE J. Biomed. Health Inform., vol. 23, no. 2, pp. 703–713, Mar. 2018.
[11] W. Q. Wu, S. Pirbhalul, A. K. Sangaiah, S. C. Mukhopadhyay, and G. L. Li, “Optimization of signal quality over comfortability of textile electrodes for ECG monitoring in fog computing based medical applications,” Future Gener. Comput. Sci., vol. 86, pp. 515–526, Sep. 2018.
[12] S. Pirbhalul, H. Zhang, W. Q. Wu, S. C. Mukhopadhyay, and Y.-T. Zhang, “Heartbeats based biometric random binary sequences generation to secure wireless body sensor networks,” IEEE Trans. Biomed. Eng., vol. 65, no. 12, pp. 2751–2759, Dec. 2018.
[13] H. Hu, R. Peng, Y.-W. Tai, and C. K. Tang, “Network trimming: A data-driven neuron pruning approach towards efficient deep architectures,” 2016, arXiv:1607.03250. [Online]. Available: https://arxiv.org/abs/1607.03250
[14] N. He, X. Zhang, and J. Sun, “Channel pruning for accelerating very deep neural networks,” in Proc. ICCV, Venice, Italy, Oct. 2017, pp. 1398–1406.
[15] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, “Pruning convolutional neural networks for resource efficient inference,” Jun. 2017, arXiv:1611.06440. [Online]. Available: https://arxiv.org/abs/1611.06440
[16] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, “Nisp: Pruning networks using neuron importance score propagation,” in Proc. CVPR, Salt Lake City, UT, USA, Jun. 2018, pp. 9194–9203.
[17] H. Wu, S. Sun, J. Li, X. Wang, and Q. Gu, “A novel channel pruning method for deep neural network compression,” 2018, arXiv:1805.11394. [Online]. Available: https://arxiv.org/abs/1805.11394
[18] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, “AMC: AutoML for model compression and acceleration on mobile devices,” in Proc. ECCV, Munich, Germany, Sep. 2018, pp. 815–832.
[19] J.-H. Luo and J. Wu, “Autopruner: An end-to-end trainable filter pruning method for efficient deep model inference,” 2018, arXiv:1805.08941. [Online]. Available: https://arxiv.org/abs/1805.08941
[20] G. S. Dhillon, K. Azizzadenesheli, Z. C. Lipton, J. Bernstein, J. Kossaifi, A. Khanna, and A. Anandkumar, “Stochastic activation pruning for robust adversarial defense,” May 2019, arXiv:1803.01442. [Online]. Available: https://arxiv.org/abs/1803.01442
[21] S. Han, J. Pool, J. Tran, and W. Dally, “Learning both weights and activations in neural networks through model compression,” in Proc. NIPS, Montreal, QC, Canada, 2015, pp. 1135–1143.
[22] H. Li, A. Kadav, I. Durkhanovic, H. Samet, and H. P. Graf, “Pruning filters for efficient convnets,” Aug. 2016, arXiv:1608.08710. [Online]. Available: https://arxiv.org/abs/1608.08710

FIGURE 9. Training loss in the first 500 iterations of ResNet-56 on Cifar-10. Curves are smoothed over a step size of 51 for better observation. SI helps converge faster during the early stage of training both with and without Cutout.

FIGURE 10. Standard deviations and weight sums of the pruned parameters by DNP in ResNet-34 trained on ImageNet. Parameters from different layers are illustrated in different colors. Better viewed in color.
F. Tung and G. Mori, "CLIP-Q: Deep network compression learning by in-parallel pruning-quantization," in Proc. ICLR, Feb. 2018. [Online]. Available: https://openreview.net/forum?id=rY-TW-WAb

S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in Proc. ICLR, Caribe Hilton, San Juan, Puerto Rico, Oct. 2016.

Y. Wang, H. Liu, W. Zheng, Y. Xia, Y. Li, P. Chen, K. Guo, and H. Xie, "Multi-objective workflow scheduling with deep-q-network-based multi-agent reinforcement learning," IEEE Access, vol. 7, pp. 39974–39982, 2019.

A. F. Aji and K. Heafield, "Sparse communication for distributed gradient descent," in Proc. EMNLP, Copenhagen, Denmark, Sep. 2017, pp. 440–445.

Y. Xu, Y. Wang, A. Zhou, W. Lin, and H. Xiong, "Deep neural network compression with single and multiple level quantization," in Proc. AAAI, New Orleans, LA, USA, Apr. 2018, pp. 4335–4342.

Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," Dec. 2017, arXiv:1712.01887. [Online]. Available: https://arxiv.org/abs/1712.01887

M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," May. 2014, arXiv:1405.3866. [Online]. Available: https://arxiv.org/abs/1405.3866

C. Tai, T. Xiao, Y. Zhang, X. Wang, "Convolutional neural networks with low-rank regularization," Nov. 2015, arXiv:1511.06067. [Online]. Available: https://arxiv.org/abs/1511.06067

J. Lin, Y. Rao, J. Lu, and J. Zhou, "Runtime neural pruning," in Proc. NIPS, Long beach, CA, USA, 2017, pp. 2181–2191.

Q. Peng, M. Zhou, Q. He, Y. Xia, C. Wu, and S. Deng, "Multi-objective optimization for location prediction of mobile devices in sensor-based applications," IEEE Access, vol. 6, pp. 71123–71132, 2018.

L. Liu and J. Deng, "Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective execution," in Proc. AAAI, New Orleans, LA, USA, Apr. 2018, pp. 3675–3682.

D. Li, X. Wang, and D. Kong, "Deeprerbirth: Accelerating deep neural network execution on mobile devices," in Proc. AAAI, New Orleans, LA, USA, Apr. 2018, pp. 3675–3682.

J. Cheng, P.-S. Wang, G. Li, Q.-H. Hu, and H.-Q. Lu, "Recent advances in efficient computation of deep convolutional neural networks," Frontiers Inf. Technol. Electron. Eng., vol. 19, no. 1, pp. 64–77, Jan. 2018.

Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in Proc. NIPS, Granada, Spain, 2011, p. 5.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in Proc. AISTATS, Sardinia, Italy, 2010, pp. 249–256.

K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in Proc. ICCV, Santiago, Chile, Dec. 2015, pp. 1026–1034.

A. S. Morcos, D. G. Barrett, N. C. Rabinowitz, and M. Botvinick, "On the importance of single directions for generalization," Mar. 2018, arXiv:1803.06959. [Online]. Available: https://arxiv.org/abs/1803.06959

K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in Proc. ECCV, Amsterdam, Netherlands, Sep. 2016, pp. 630–645.

K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," Sep. 2014, arXiv:1409.1556. [Online]. Available: https://arxiv.org/abs/1409.1556

W. Li, Y. Xia, M. Zhou, X. Sun, and Q. Zhu, "Fluctuation-aware and predictive workflow scheduling in cost-effective infrastructure-as-a-service clouds," IEEE Access, vol. 6, pp. 61488–61502, 2018.

Q. Huang, K. Zhou, S. You, and U. Neumann, "Learning to prune filters in convolutional neural networks," in Proc. WACV, Lake Placid, NY, USA, Mar. 2016, pp. 709–718.

I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in Proc. ICMIL, Atlanta, GA, USA, 2013, pp. 1139–1147.

A. Krizhevsky and G. Hinton. Learning Multiple Layers of Features from Tiny Images. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html

L. Kaiser, A. N. Gomez, and F. Chollet, "Depthwise separable convolutions for neural machine translation," Jan. 2017, arXiv:1706.03059. [Online]. Available: https://arxiv.org/abs/1706.03059

I. Loshchilov and F. Hutter, "SGDR: Stochastic Gradient Descent with Warm Restarts," Aug. 2016, arXiv:1608.03983. [Online]. Available: https://arxiv.org/abs/1608.03983

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," J. Mach. Learn. Res., vol. 15, pp. 1929–1958, Jun. 2014.

T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," 2017, arXiv:1708.04552. [Online]. Available: https://arxiv.org/abs/1708.04552

Y. Kim, "Convolutional neural networks for sentence classification," in Proc. EMNLP, Doha, Qatar, Aug. 2014, pp. 1746–1751.

B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," in Proc. ACL, Ann Arbor, MI, USA, Jun. 2005, pp. 115–124.

D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," Dec. 2014, arXiv:1412.6980. [Online]. Available: https://arxiv.org/abs/1412.6980

Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in Proc. ICCV, Venice, Italy, Oct. 2017, pp. 2755–2763.

S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in Proc. IJCAI, Lille, France, Feb. 2015, pp. 448–456.

J. Ye, X. Lu, Z. Lin, and J. Z. Wang, "Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers," Jan. 2018, arXiv:1802.00124. [Online]. Available: https://arxiv.org/abs/1802.00124

LEI WU received the B.S., M.S., and Ph.D. degrees in computer science and technology from the University of Electronic Science and Technology of China, Chengdu, China, in 2001, 2004, and 2010, respectively. From 2004 to 2007, he was a Lecturer with the Department of Applied Mathematics, University of Electronic Science and Technology of China, where he is currently an Associate Professor with the School of Mathematical Sciences. His current research interests include image processing, data mining, and pattern recognition.

HAN YUE received the B.S. and M.S. degrees in computational mathematics from the University of Electronic Science and Technology of China, Chengdu, China, in 2015 and 2018, respectively. His current research interests include image processing, pattern recognition, and machine learning.
PENG CHEN received the B.S. degree in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2001, the M.S. degree in computer science from Peking University, Beijing, China, in 2004, and the Ph.D. degree in computer science and technology from the Sichuan University of China, Chengdu, in 2010. He is currently a Professor with the School of Computing and Software Engineering, Xihua University, Chengdu. His research interests include the areas of machine learning and pattern recognition.

DE-AN WU received the Ph.D. degree in applied mathematics from the University of Tsukuba, Japan, in 2004. From 2005 to 2009, he was a CREST Research Fellow with the Tsukuba Advanced Research Alliance, University of Tsukuba. Since 2010, he has been an Associate Professor with the University of Electronic Science and Technology of China. His research interests include the areas of machine learning and statistical learning theory.

QIU JIN received the B.S. degree in statistics from Northeastern University, in 2017. She is currently pursuing the M.S. degree in statistics with the University of Electronic Science and Technology of China. Her research interests include computer vision and machine learning.