Multidimensional Systems and Signal Processing (2020) 31:793–827
https://doi.org/10.1007/s11045-019-00686-z

PAC-Bayesian framework based drop-path method for 2D discriminative convolutional network pruning

Qinghe Zheng[1] · Xinyu Tian[2] · Mingqiang Yang[1] · Yulin Wu[1] · Huake Su[3]

Received: 9 May 2019 / Revised: 25 July 2019 / Accepted: 4 October 2019 / Published online: 19 October 2019

Abstract
Deep convolutional neural networks (CNNs) have demonstrated its extraordinary power on various visual tasks like object detection and classification. However, it is still challenging to deploy state-of-the-art models into real-world applications, such as autonomous vehicles, due to their expensive computation costs. In this paper, to accelerate the network inference, we introduce a novel pruning method named Drop-path to reduce model parameters of 2D deep CNNs. Given a trained deep CNN, pruning paths with different lengths is achieved by ordering the influence of neurons in each layer on the probably approximately correct (PAC) Bayesian boundary of the model. We believe that the invariance of PAC-Bayesian boundary is an important factor to guarantee the generalization ability of deep CNN under the condition of optimizing as much as possible. To the best of our knowledge, this is the first time to reduce model size based on the generalization error boundary. After pruning, we observe that the convolutional kernels themselves become sparse, rather than some being removed directly. In fact, Drop-path is generic and can be well generalized on multi-layer and multi-branch models, since parameter ranking criterion can be applied to any kind of layer and the importance scores can still be propagated. Finally, Drop-path is evaluated on two image classification benchmark datasets (ImageNet and CIFAR-10) with multiple deep CNN models, including AlexNet, VGG-16, GoogLeNet, and ResNet-34/50/56/110. Experimental results demonstrate that Drop-path achieves significant model compression and acceleration with negligible accuracy loss.

Keywords Deep learning · Model compression · Model acceleration · Structure exploration · Generalization error boundary

✉ Mingqiang Yang
yangmq@sdu.edu.cn

1 School of Information Science and Engineering, Shandong University, Qingdao 266237, China

2 College of Mechanical and Electrical Engineering, Shandong Management University, Jinan 250357, China

3 School of Microelectronics, Xidian University, Xi'an 710126, China

1 Introduction

Relying on the ultra-large-scale parameters (Lecun et al. 2015), deep neural networks have become the flexible function approximators and have been very successful in a wide range of tasks, such as industrial automation control (Miao and He 2017), computer aided diagnosis (Shin et al. 2016; Samala et al. 2019) and financial data analysis (Jang and Lee 2018; Jie and Wang 2017). With the development of deep learning, the structure of deep neural networks becomes more and more complex. Inevitably, deep neural networks require high computational costs in both training and testing phases, which is one of the most important reasons that restricts its practical application in consumer electronics. For mobile and embedded devices (Gutierrez-Galan et al. 2018), the inference speed and file size of the model are critical. The depth, size and amount of computation, as well as the memory overhead during the model runtime are rapidly increasing, which make it difficult for deep networks to be applied to mobile terminals or embedded devices with low hardware resources and high real-time requirements, such as autonomous vehicles (Li et al. 2018). Therefore, how to reduce the network size and inference time under the premise of ensuring performance and promote its application in consumer electronics is a hot topic of current research. There have been great interests in reducing the redundancy of deep neural networks to achieve model compression and acceleration (He et al. 2017; Frankle and Carbin 2019; Zoph et al. 2018). The lightening of neural networks (Xu et al. 2019; Kim et al. 2016; Zheng et al. 2017) is the future development direction, and network pruning (Huang and Wang 2018; Yu et al. 2018; Liu et al. 2017) is one of the key technologies.

The pruning of neural networks is meant to reduce or control the number of non-zero parameters or feature maps that need to be used in the model. Pruning can be seen as a structural exploration that finds out how many parameters or feature maps are really needed in each layer to get the best performance. In fact, researchers found that only about 4% of the parameters need to be updated during back-propagation (Molchanov et al. 2017). By sparsifying deep neural networks, we can avoid unnecessary computation and resources, since irrelevant degrees of freedom are pruned away and do not need to be computed. Another benefit is that by reducing the number of parameters (i.e., the redundancy in the parameter space), the generalization ability of the neural network can even be improved. As we have seen in the recent research on the generalization ability of deep neural networks (Painsky and Rosset 2016; Hu et al. 2017; Zheng et al. 2018), the original number of parameters (L0-norm) cannot actually predict its performance. In other words, based on the experimental results, people found that pruning the network in prudent manners even helps to improve generalization. At the same time, new parameter correlations are being developed to predict and describe generalization ability, such as the Fisher-Rao norm (Goh et al. 2014). Interestingly, Fisher pruning algorithm has been proved to have a good correlation with Fisher–Rao norm (Tian et al. 2017), which means that there is a deeper and incomprehensible relationship between network pruning, parameter redundancy and its generalization.

Pruning parameters is simple but challenging because removing parameters in one layer might dramatically impair the input of the following layers. Our goal is to reduce the computational cost of network inference, especially in the transfer-learning environment: when a pre-trained model starts to be fine-tuned, it inherits the large amount of computation that was used to solve the original task, which may be superfluous for solving the target task. On the other hand, small models that are gradually pruned from a dense neural network usually yield much better results than the direct training from scratch of small models, which shows that the value of automatic pruning methods may lie in identifying efficient structures and

performing implicit architecture searches rather than just selecting "important" weights. In practice, the denser network can help to avoid bad local minimums and provide better initializations which are critical for sparser network to learn effective representations (Kim et al. 2016).

In the main part of this paper, to accelerate the network inference, we propose a holistic pruning method named Drop-path to reduce model parameters of 2D deep convolutional neural networks, utilizing redundancy inter parameters per layer under PAC-Bayesian framework. The whole process is followed in two alternative steps: pruning and fine-tuning. Given a trained deep CNN, pruning each path is achieved by ordering the influence of neurons in each layer on the PAC-Bayesian boundary of the model. We believe that the invariance of PAC-Bayesian boundary is an important factor to guarantee the generalization ability of deep CNN under the condition of optimizing as much as possible. To the best of our knowledge, this is the first time to reduce model size based on the generalization error boundary. In the pruning step, parameters in paths with different lengths are removed, as shown in Fig. 1. It is worth noting that most convolutional kernels become sparse, rather than some convolutional kernels being completely removed and acting like Atrous convolution operation (Yu and Koltun 2015). In the fine-tuning step, the new network initialized by the previous model is optimized to retain its classification capability. The two steps are alternated to achieve the trade-off between model size and the performance. The pruning ratio in Drop-path is pre-defined as a hyper-parameter which can be determined according to the actual needs of specific applications, e.g., accuracy, memory, and floating-point operations per second (FLOPs). Drop-path is generic and can be well generalized on multi-layer and multi-branch models, since parameter ranking can be applied to any kind of layer and the importance scores can still be propagated. Compared with layer-by-layer pruning and retraining independently (Yang et al. 2017) or greedily (Li et al. 2017), the reasons for cross-layer removal of connections in different paths are as follows:

- For deep neural networks, the overall pruning and fine-tuning can extremely save the training time.
- Pruning parameters across layers gives a holistic view of the robustness of the model, resulting in a smaller network.
- The overall consideration is necessary, as the pruning of bottom and top layers may affect each other. For example, for the residual network (ResNet), pruning the identity mapping layers or the second layer of each residual block results in additional pruning of subsequent layers.

Finally, we perform various experiments to demonstrate the effectiveness of the proposed Drop-path method. Eight popular deep CNNs [e.g., AlexNet (Krizhevsky et al. 2012), VGG-16 (Simonyan and Zisserman 2014), GoogLeNet (Szegedy et al. 2015) and ResNet (He et al. 2016)] trained on ImageNet (Russakovsky et al. 2015) and CIFAR-10 (Krizhevsky and Hinton 2009) achieve about 2 $\times$ inference speed up along with no more than 1% increase of classification error. In this way, we show that Drop-path can accelerate the inference of network with practical implementations, without seriously hurting the performance of deep CNN.

The rest of this paper is organized as follows. Section 2 gives a brief introduction to the related work of network pruning methods. Section 3 introduces the PAC-Bayesian framework based Drop-path method. Extensive experiments and analyses are presented in Sect. 4. Some properties and conjectures of Drop-path method are discussed in Sect. 5. Finally, we conclude our works and future directions in Sect. 6.

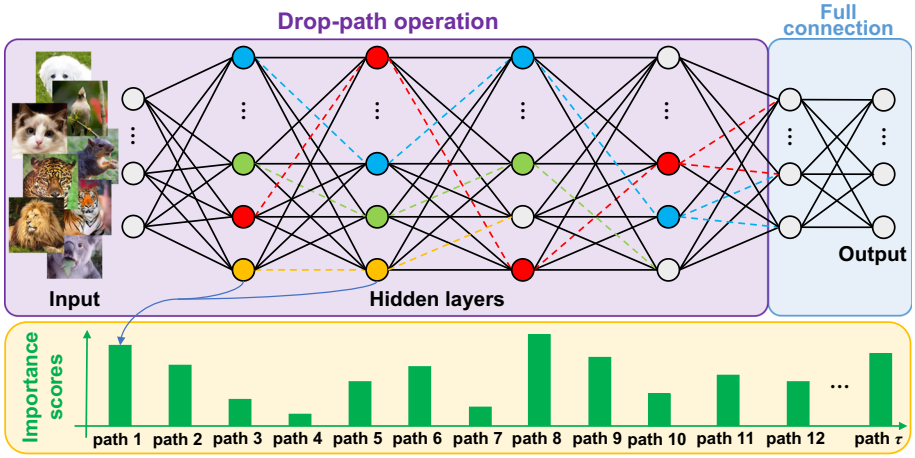


Fig. 1 An illustration of Drop-path method running in a simple neural network model. Paths with different lengths, e.g., green and yellow paths, are gradually pruned during the training process. The dotted lines represent the connections that are removed

2 Related work

Heavy-duty deep neural networks are difficult to be applied to mobile terminals and embedded devices which require great intelligence and real-time performance in real life. Compression and acceleration of deep neural networks have recently drawn much attention in deep learning community. Recent studies (Kim et al. 2016; Srinivas and Babu 2016; Han et al. 2015) have investigated the significant redundancy (the parameters that are not important or meaningless in the reasoning process of deep neural network) in deep neural networks and reduced the number of neurons and filters by pruning the unimportant ones.

In general, there are two common understandings behind the pruning process. First, it is important to train a large and over-parameterized model which can provide strong representation and optimization capabilities, and people can safely remove redundant parameters without significant damage to accuracy. Therefore, it is generally accepted that this method is more effective than directly training a smaller network from scratch. Second, the pruned architecture and the relevant weights are considered to be the key to achieving the efficient performance. Therefore, most of the existing pruning techniques choose to fine-tune the pruned model instead of training it from scratch. The weights that are retained after pruning are often considered critical (Wang et al. 2018) because it is difficult to select an important set of weights accurately from the structural space.

To measure the importance of neurons or connections in a network, the exact solutions are very hard to obtain given the complexity of highly non-linearity. Some previous works (Molchanov et al. 2017; Theis et al. 2018; Luo et al. 2017) approximate it using 2nd-order Taylor expansion. Our work is a different approximation based on the PAC-Bayesian framework. Bolukbasi et al. (2017) regularize runtime in convolutional layers to determine whether some kernels or weights can be bypassed. Sparsity regularization terms (Figurnov et al. 2016) have been used to learn sparse CNN structure. Han et al. (2016) propose to learn important connections and perform network pruning based on the weight of network connection. Denton et al. (2014) apply singular value decomposition to neural network to preserve important connections. The meProp (Sun et al. 2017) uses approximate gradients by keeping only top-k

elements based on the magnitude values. Torfi and Shirvani (2018) propose group sparsity to constrain the effective parameters and set an extra loss term to force some parameters not to be sparse to keep the network performance. The above methods prune the "least important" neurons layer-by-layer either independently or greedily, without considering weights in different layers jointly and the influence of error propagation in the deep network. In fact, one problem with such methods is that neurons deemed unimportant in an early layer can contribute significantly to responses of important neurons in later layers. Therefore, the neurons in the network must be pruned as a whole according to a unified goal.

Huang and Wang (2018) add an item to structural blocks or neural nodes to regularize the network structure and control the output. It adjusts the unit while processing the whole structure, making the final neural network model look cleaner. Yu et al. (2018) determine the importance of each channel in the back-propagation process, and then selects the important parts based on Inf-FS method. Sau and Balasubramanian (2016) use a noisy network to guide the training of another small network. Liu et al. (2017) add a scale quantization factor to each channel of the convolution kernel, which is regularized during training. Controlling each channel is equivalent to controlling the convolution kernels of the corresponding cube. Sun et al. (2016) propose an iterative learning sparse ConvNets, and retrains the entire model with initial weights learned in previous iterations. Targeted Dropout (Gomez et al. 2018) combines the post hoc pruning strategy into the training process and has no significant influence on the potential performance of the particular architecture. Since the estimation of the importance of neurons (i.e., weights) is based on the situation before Dropout, such estimation errors may accumulate during the optimization, and eventually leading to divergence results. In addition, there are special designs that enable small-scale networks to directly train for outstanding performance, such as ShuffleNet (Zhang et al. 2018), Xception (Chollet 2017), Squeezenet (Iandola et al. 2017), and MobileNet (Howard et al. 2017). The introduction of additional prior knowledge (or manual removal of redundancy) makes these frameworks not applicable to a variety of visual tasks, so it is difficult to be integrated into the mobile-end visual processing system.

3 Drop-path method based on PAC-Bayesian framework

In this section, we present the Drop-path method under PAC-Bayesian framework for network pruning and demonstrate the advantages of this approach in achieving more efficient computation and storage capacity savings. It consists of the following steps:

1. Pretrain the baseline deep CNN until convergence on the image classification task;
2. Prune and fine-tune the network alternately;
3. Output the sparse deep CNN model after achieving the target trade-off between classification accuracy and pruning objective, e.g., FLOPs or memory footprint.

The overall pruning and fine-tuning procedure using Drop-path is shown in Fig. 2.

3.1 Network pruning and fine-tuning

The goal of network pruning is to remove redundant parameters or connections while minimizing accuracy loss. Our starting point is a well-trained high-performance baseline deep CNN model $\mathbb{N}_0 : f(\mathbf{x}; \boldsymbol{\theta}_0)$, in which $\mathbf{x}$ and $\boldsymbol{\theta}_0$ represent the inputs (i.e., sample images) and initialized network weights, respectively. Then the paths in deep CNN are defined as connections formed by parameters in different layers, which do not necessarily span the entire model

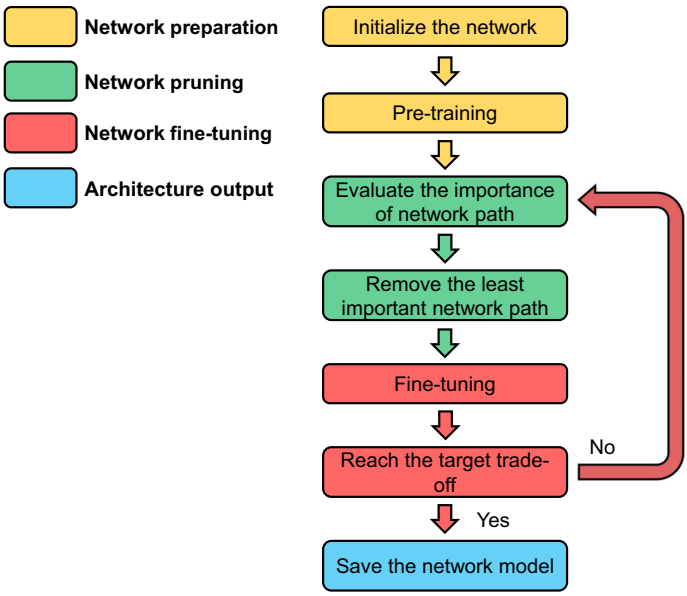


Fig. 2 Network pruning and fine-tuning procedure using Drop-path method

and therefore have varying lengths. Then we remove redundant paths from path set $P_t = \{p_t^1, p_t^2, \ldots, p_t^\tau\}$ in the deep CNN model in a global manner, including all convolutional and fully or locally connected layers. In the path set, $p_t^i = \{\theta_j, \theta_{j+1}, \ldots, \theta_k\}_{1\le j<k\le L}$ represents the i-th path from j-th layer to k-th layer and L is a hyper-parameter representing the maximum length of the path, that is, a path contains at most L ($\ge$2) parameters, and each of which comes from a different layer. Suppose a deep CNN model consists of s layers with r (>1) parameters per layer, then the total number of paths τ can be formulated by

$$\begin{aligned}\tau &= (S-1)r^2 + (S-2)r^3 + \cdots + (S-L+1)r^L \\ &= \sum_{i=1}^{L-1}(S-i)r^{i+1} = \sum_{i=1}^{L-1} Sr^{i+1} - \sum_{i=1}^{L-1} ir^{i+1} \\ &= Sr^2\frac{1-r^{L-1}}{1-r} - r^2\frac{1-r^{L-1}}{(1-r)^2} + r\frac{(L-1)r^L}{1-r} \\ &= \frac{(S-1)^2 - Sr^3 + (L-S)r^{L+1} + (S-L+1)r^{L+2}}{(1-r)^2} \\ &= o(r^{L-1}),\ r \neq 1 \end{aligned} \tag{1}$$

The above equation indicates that the total number of paths is the L-order infinitesimal of r. The number of parameters per layer is determined by the structure of the baseline model, so we can constrain the maximum length of paths to limit their number. Generally, the maximum length of paths in deep CNN is set to less than 4 to prevent operation overflow. Given the initial pruning rate ξ_0 ($0 \le \xi_0 \le 1$), a total of $\xi_0\ \tau$ paths which includes $\xi_0\ \tau\ L$ parameters are pruned from the total number of parameters $|\boldsymbol{\theta}|$ after Drop-path pruning. In the pruning

process, the number of removed parameters at each iteration is controlled by step ζ, i.e., the pruning rate at t-th pruning is updated according to

$$\xi_t = \xi_{t-1} - \zeta \tag{2}$$

This means that a total number of $(1 - \xi_0 + \zeta)\tau$ paths are removed at each iteration.

When the connections are sparsified, a new network $\mathbb{N}_t$: $f(\mathbf{x}; \boldsymbol{\theta}_t)$ initialized by the previous model is trained by using mini-batch stochastic gradient descent (SGD) method, as given by

$$\boldsymbol{\theta}_t^i = \boldsymbol{\theta}_t^{i-1} - \alpha \cdot \frac{1}{M} \sum_{i=1}^{M} \nabla_{\boldsymbol{\theta}_t^{i-1}} \left[\mathcal{L}(\mathbf{x}_i, \mathbf{y}_i) \right] \tag{3}$$

Algorithm 1 Pruning and fine-tuning of deep CNN by Drop-path.

Input: training set $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$.

Initialization: baseline network model $\mathbb{N}_0$, batch size M, learning rate α, Dropout rate, weight decay λ, momentum, pruning rate ξ_0, step size ζ, maximum length L.

1: **for** t from 1 to T **do**
2: transfer the weights of $\mathbb{N}_{t-1}$ to $\mathbb{N}_t$ for initialization: $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1}$;
3: calculate the importance of each path in the model $\mathbb{N}_t$ to obtain
4: the inhibition mask $\boldsymbol{I}_t$;
5: update $\boldsymbol{\theta}_t$ by dot-multiplying them with inhibition mask $\boldsymbol{I}_t$;
6: **while** $\mathbb{N}_t$ not converge **do**
7: **forward-propagation:**
8: compute the loss $\mathcal{L}$ of network $\mathbb{N}_t$;
9: **back-propagation:**
10: update $\boldsymbol{\theta}_t$ by mini-batch SGD;
11: **end while**
12: $\xi_t \leftarrow \xi_{t-1} - \zeta$;
13: **end for**

Output: network sequence $\{\mathbb{N}_1, \mathbb{N}_2, ..., \mathbb{N}_T\}$ with sparse connections specified by inhibition masks $\{\boldsymbol{I}_1, \boldsymbol{I}_2, ..., \boldsymbol{I}_T\}$.

where $\boldsymbol{\theta}_t^i$ represents the updated weights at the i-th iteration of mini-batch SGD in t-th fine-tuning process. $\boldsymbol{\theta}_t^0$ is the weights after t-th pruning. α and M denotes the learning rate and batch size, respectively. $\mathcal{L}$ is a loss function to measure the gap between model output f ($\mathbf{x}$) and its corresponding ground-truth label $\mathbf{y}$, such as the negative log-likelihood function:

$$\mathcal{L}(\mathbf{x}_i, \mathbf{y}_i) = -\frac{1}{C} \sum_{j=1}^{C} \left[y_i^j \log f(\mathbf{x}_i; \boldsymbol{\theta})^j + (1 - y_i^j) \log(1 - f(\mathbf{x}_i; \boldsymbol{\theta})^j) \right] + \lambda ||\boldsymbol{\theta}||_2 \tag{4}$$

where C denotes the total number of dataset categories, that is, the dimension of the output vector. y_i^j and $f(\mathbf{x}_i)^j$ represent the j-th component of ground-truth label and model prediction, respectively. λ is weight decay parameter that controls the L2-regularization intensity. The choice of loss function is independent of pruning, but only depends on the task to be solved by the original baseline model. The parameters transferred from the denser network $\mathbb{N}_{t-1}$ are good initialization of the sparser network $\mathbb{N}_t$ to be further fine-tuned. Finally, a sequence

of network models $\{\mathbb{N}_0, \mathbb{N}_1, \ldots, \mathbb{N}_T\}$ with fewer and fewer parameters are fine-tuned and $\mathbb{N}_T$ is the final sparse deep CNN model obtained. During the whole pruning and fine-tuning process, the previous well-trained network is used to calculate the importance of paths in current version of deep CNN model and guide the next parameter removing procedure.

During alternate pruning and fine-tuning process, we use a binary matrix (referred to as inhibition mask $\boldsymbol{I}$) with the same size as the weight matrix of the whole CNN model to specify the reserved or removed parameters in each path. Before the next fine-tuning of the pruned network, the weight matrix is first updated by dot-multiplying with the inhibition mask:

$$\hat{\boldsymbol{\theta}}_t = \boldsymbol{\theta}_t \circ \boldsymbol{I}_t \tag{5}$$

where $\circ$ represents the element-wise operation for computing the Hadamard product. By doing this, the redundant parameters are removed by multiplying with 0, while the remaining parameters can be preserved by multiplying with 1. Then the following training procedure can be performed in the same way as the baseline network, and the model gradually becomes sparse. All the removed parameters being updated through fine-tuning would be truncated to zero again before next pruning. When the pruned model is deployed for testing on mobile devices or embedded systems, the storage space and computation requirements have been greatly reduced, since the memory footprint of inhibition mask (~10% at 30% pruning rate) is much smaller than that of removed real-valued parameters. Based on this, the sparse convolutional kernels of the pruned network can be transformed into a structured matrix for storage and computation. In other words, a matrix of $m \times n$ size only needs less than $m \times n$ parameters to describe.

The network pruning and fin-tuning process with Drop-path is summarized in Algorithm 1, in which the parameter ranking criterion will be introduced in Sect. 3.2.

3.2 Path ranking criterion

In this part, we introduce the path ranking criteria used to determine the importance of parameters in each path based on the PAC-Bayesian framework. Our intuition is that the invariance of generalization error boundary of deep CNN model should play a key role in the model pruning since the remaining parameters ensure that the potential of the pruned model can be stimulated by fine-tuning. In other words, parameters that have the least impact on the generalization error boundary are the least important. For simpler deep CNN models such as AlexNet and VGG-16, we can easily prune any of the parameters in any trainable layers. However, for more complex network model like ResNet, pruning filters directly is usually difficult. The structure of residual block imposes restrictions, which leads to the interaction of pruning between layers. Therefore, we do not remove the entire convolutional kernels, but prune the parameters on each path to form sparse filters.

Suppose the input space is represented by $\mathcal{X}$ and the classifier $f(\cdot)$ satisfying the distribution $\mathcal{F}$ is used to predict the labels of samples. In order to introduce the criterion for judging the influence of parameters on generalization error boundary, we first give the following relevant definitions (Langford and Schapire 2015) of PAC-Bayesian boundary:

Definition 1 The expected error of a classifier f $(\cdot)$ is defined as the probability Pr of misclassifying the randomly sampled datum $(\mathbf{x}, \mathbf{y})$, as given by

$$C_{\mathcal{X}} \equiv \Pr_{(\mathbf{x},\mathbf{y})\in\mathcal{X}}(f(\mathbf{x}) \neq \mathbf{y}) \tag{6}$$

Definition 2 The empirical error of a classifier f (·) is defined as the average probability of misclassifying dataset D with a total number N of samples, as given by

$$\hat{C}_D \equiv \Pr_{(\mathbf{x},\mathbf{y})\in D}(f(\mathbf{x}) \neq \mathbf{y}) = \frac{1}{N}\sum_{i=1}^{N} P(f(\mathbf{x}_i) \neq \mathbf{y}_i) \tag{7}$$

Definition 3 The expected distribution error of the classifier $f \in \mathcal{F}$ is defined as the probability of misclassifying the sample $(\mathbf{x}, \mathbf{y}) \in \mathcal{X}$, as given by

$$\mathcal{F}_{\mathcal{X}} \equiv \mathbb{E}_{f\in\mathcal{F}} C_{\mathcal{X}} \tag{8}$$

Algorithm 2 Path ranking procedure under PAC-Bayesian framework.

Input: training set $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$.

Initialization: pruned network model $\mathbb{N}_t$, pruning rate ξ_t, step size ζ, maximum length L, path set $P_t = \left\{p_t^1, p_t^2, ..., p_t^{\tau}\right\}$, predefined pruned path set $\hat{P} = \left\{\hat{p}^1, \hat{p}^2, ..., \hat{p}^{\tau\xi_t}\right\}$;

1: calculate the original loss $\mathcal{L}_D$;
2: **for** i from 1 to τ **do**
3: calculate the loss $\hat{\mathcal{L}}_D^{\,i}$ of network removing the path p_t^i;
4: calculate the KL-divergence ψ_i;
5: **if** $\psi_i < \psi_{\min} = \min\{\psi_1, \psi_2, \psi_{\min}, ..., \psi_{\tau\xi_t}\}$ **do**
6: replace $\psi_{\min}$ in $\{\psi_1, \psi_2, \psi_{\min}, ..., \psi_{\tau\xi_t}\}$ with ψ_i;
7: replace corresponding path with p_t^i in path set $\hat{P}$;
8: **end if**
9: **end for**
10: I_t is created based on $\{p_t^1 \cap p_t^2 \cap \ldots \cap p_t^{\tau\xi_t}\}$;

Output: inhibition mask I_t.

Definition 4 The empirical distribution error of classifier $f \in \mathcal{F}$ is defined as the probability of misclassifying the sample $(\mathbf{x}, \mathbf{y}) \in D$, as given by

$$\hat{\mathcal{F}}_D \equiv \mathbb{E}_{f\in\mathcal{F}} \hat{C}_D \tag{9}$$

Based on the above two error metrics, the property of the PAC-Bayes boundary can be summarized as follows:

Theorem 1 (PAC-Bayesian Boundary) *For the entire input space $\mathcal{X}$, all the prior distributions P(f) of classifier f (·) satisfy the following inequality for any $\varepsilon \in (0,\ 1]$:*

$$\Pr_{D\sim\mathcal{X}^N}\left[\forall\mathcal{F}(f) : KL(\hat{\mathcal{F}}_D||\mathcal{F}_{\mathcal{X}})\right] \leq \frac{KL[\mathcal{F}(f)\ ||\ P(f)] + \ln(\frac{N+1}{\varepsilon})}{N} \geq 1-\varepsilon \tag{10}$$

where

$$KL\left(\hat{\mathcal{F}}_D||\mathcal{F}_{\mathcal{X}}\right) = \mathcal{F}_{\mathcal{X}} \ln\frac{\mathcal{F}_{\mathcal{X}}}{\hat{\mathcal{F}}_D} + (1-\mathcal{F}_{\mathcal{X}}) \ln\frac{1-\mathcal{F}_{\mathcal{X}}}{1-\hat{\mathcal{F}}_D} \tag{11}$$

$$KL[\mathcal{F}(f)\ ||\ P(f)] = \mathbb{E}_{f\in\mathcal{F}} \ln(\frac{\mathcal{F}(f)}{P(f)}) \tag{12}$$

Theorem 1 illustrates that for any classifier f satisfying the distribution $\mathcal{F}$, their expected error and empirical error can be defined by its prior distribution and empirical distribution. Furthermore, if the prior distribution of a classifier is known and the prior distribution and empirical distribution are assumed to be of the same type, the inequality (8) can be further simplified to make the classification boundary more compact, i.e., the loss is smaller. It gives the upper bound of the average empirical error, which can be used as the absolute measure to evaluate the generalization performance of the model. For a given learning algorithm and training set, the empirical error is fixed. And in this case, the empirical error and the relative entropy [i.e., $KL(\mathcal{F}||P)$] are increased. In fact, PAC-Bayesian boundary (Herbrich and Graepel 2002) provides the most compact generalization boundary for the classifier and can therefore be used as a favorable way to evaluate the generalization ability of the learning algorithm.

In practice, the generalized error boundary of deep CNN model is usually difficult to calculate due to large-scale parameters. We propose to use the cross-layer parameters in the path to locally observe their influence on the generalization error boundary, and estimate the change of the generalization error boundary of the whole model. In the end, it is used as ranking criterion for judging the importance of the path. Motivated by this, we tend to remove paths (and the corresponding parameters) where the connected neurons have negligible effects on the generalization error boundary of the model.

The influence of paths on generalization error boundary of deep CNN can be characterized by KL-divergence ψ_i between model losses before and after pruning i-th path, which can be formulated by

$$\psi_i = KL\left(\hat{\mathcal{L}}_D||\mathcal{L}_D\right) = \mathcal{L}_D \ln \frac{\mathcal{L}_D}{\hat{\mathcal{L}}_D} + (1-\mathcal{L}_D)\ln\frac{1-\mathcal{L}_D}{1-\hat{\mathcal{L}}_D} \tag{13}$$

where $\mathcal{L}_D$ and $\hat{\mathcal{L}}_D$ represent the model loss on training set D before and after pruning, respectively. And $\hat{\mathcal{L}}_D$ can be calculated by

$$\hat{\mathcal{L}}(\mathbf{x}_i, \mathbf{y}_i) = -\frac{1}{C}\sum_{j=1}^{C}\left[y_i^j \log f(\mathbf{x}_i;\hat{\boldsymbol{\theta}})^j + (1-y_i^j)\log(1-f(\mathbf{x}_i;\hat{\boldsymbol{\theta}})^j)\right] + \lambda||\hat{\boldsymbol{\theta}}||_2 \tag{14}$$

Then ψ_i is min–max normalized as the criterion to determine the importance of each path (i.e., the corresponding parameters), as calculated by

$$\psi_i \leftarrow \frac{\psi_i - \psi_{\min}}{\psi_{\max} - \psi_{\min}} \tag{15}$$

Finally, we can get a series of importance sequences sorted from small to large as shown below to remove redundant paths:

$$\psi_{\min} \le \psi_i \le \psi_j \le \cdots \le \psi_{\max} \tag{16}$$

In other words, the paths are removed in terms of normalized KL-divergence ψ_i from small to large. In the end, there are $\tau\xi_t$ removed paths $P_t = \{p_t^1, p_t^2, \ldots, p_t^{\tau\xi_t}\}$ at the t-th pruning iteration, and the removed parameters include $\{p^1 \cap p^2 \cap \cdots \cap p_t^{\tau\xi_t}\}$. Finally, the inhibition mask I_t is created for network fine-tuning, in which 1 and 0 denotes reserved and removed

parameters, respectively. The path ranking procedure based on PAC-Bayesian framework is summarized in Algorithm 2.

3.3 Reasons for pruning paths rather than individual parameters

Path is a collection of parameters spanning multiple layers in a model, which plays a role of "receptive field" of convolutional kernels but on model structures, and estimates the importance of local structure of deep CNN in pruning process. Compared with the statistics based on single parameter, the variation of generalized error boundary based on the calculation of multiple parameters in the path is more accurate, which can capture the change of deep CNN in the structure space more comprehensively. On the other hand, the compulsory cross-layer parameter removal mechanism can improve the multi-scale feature extraction ability of the model. In the fine-tuning process after Drop-path, the potential of the model can be fully exploited and the representational ability can be exerted through more sparse convolutional kernels, playing the role of structure distillation like "data distillation" proposed in Radosavovic et al. (2018). Perhaps the redundancy of deep CNN is centralized, which means that the model only needs a small number of parameters to mine the implicit knowledge hidden in the sample set.

We observe the loss gains of two deep CNN models (AlexNet and VGG-16) by suppressing the parameters one by one. Four cases, including the original networks and pruned networks with three different pruning rates, are counted. The results are min-max normalized to [0, 1] and shown in Fig. 3. We observe the effect of the parameters on the total loss of test samples by zeroing the weights in the network one by one. Although the impact of parameters on model classification results cannot directly reflect the importance of the weight, the loss gain can characterize their role in the model. It can be clearly seen that there are sharp rises in the pruned networks, which indicates that the parameters at the bottom play a more important role. On the other hand, the effects of single parameter on the classification results gradually becomes consistent with the pruning process, while at the beginning a large number of parameters hardly affect the results. In other words, in the model structure space, the structure which contains a lot of redundant information is full of the whole space, and a small number of effective sparse structures are clustered together. This is inspiring for model pruning or structure exploration: we can remove model parameters centrally in a small range, i.e., find the appropriate structure along several directions in the structure space, which is exactly what Drop-path implements.

Compared with traditional sorting and pruning algorithms based on single parameter, Drop-path can improve the computational efficiency. The computational complexity of single parameter based ranking method is $L/(L-1)$ times that of path based ranking method. Furthermore, the variation of generalized error boundary caused by a single parameter is similar and indistinguishable. The ranking criterion based on single parameter is unstable, which usually leads to the sharp collapse of deep CNN in the alternative pruning and fine-tuning process. The classification accuracy of the model would be drastically reduced as the pruning proceeds. In Drop-path method, the parameters in one path span at most the L layers at the same time, avoiding excessive removal of a layer of parameters. In fact, the parameters in bottom layers are usually more important, and Drop-path can remove more parameters in top layers as much as possible while retaining more underlying parameters.

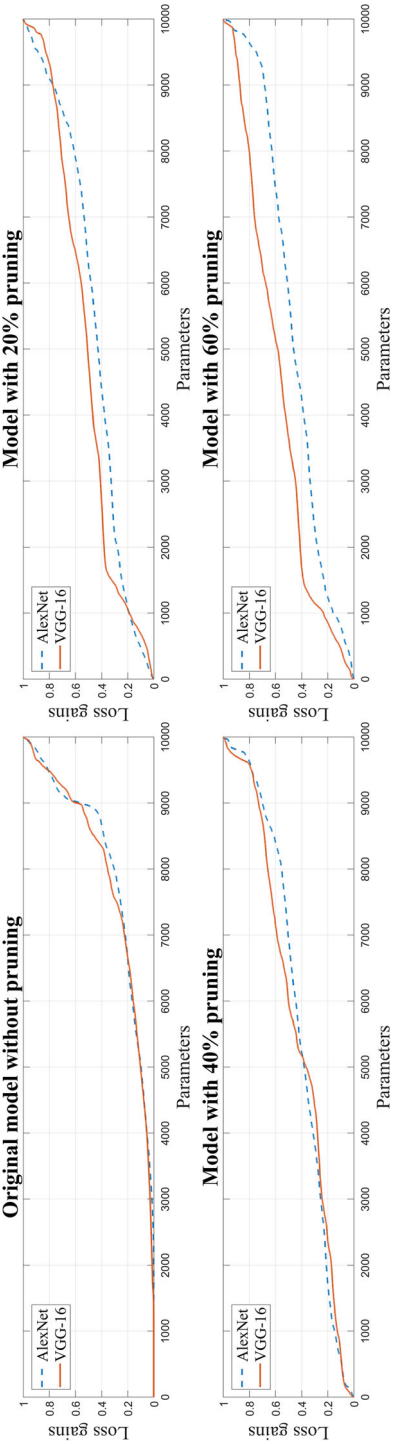


Fig. 3 Loss gains of AlexNet and VGG-16 by suppressing the parameters one by one

4 Experiments

4.1 Experimental setup

4.1.1 Baseline models and benchmark datasets

We conduct experiments on two image classification benchmark datasets, that is, ImageNet (Russakovsky et al. 2015) and CIFAR-10 (Krizhevsky and Hinton 2009). The ImageNet is a large scale dataset, which contains 1000 classes of 1.2 million natural images for training and 50,000 images for testing. CIFAR-10 is a popular benchmark for small-scale natural image classification, which contains 10 classes of 50,000 images for training and 10,000 images for testing.

Then we evaluate the Drop-path method by testing eight popular deep CNN models: AlexNet (Krizhevsky et al. 2012), VGG-16 (Simonyan and Zisserman 2014), GoogLeNet (Szegedy et al. 2015), ResNet-34/50 (He et al. 2016) for ImageNet, and a modified VGG-16, ResNet-56/110 for CIFAR-10. The architectures for ImageNet, including the category of layers, the size and number of convolutional kernels, are shown in Table 1. In the modified VGG-16 for CIFAR-10, the first two fully connected layers fc1 and fc2 are removed, only the last fully connected layer fc3 with 10 neurons is used for classification. ResNet-56/110 have three stages of residual blocks for outputting feature maps with sizes of 32×32, 16×16, and 8×8. For the subsequent comparison, the number of parameters, FLOPs, and classification accuracies of above baseline models are summarized in Table 2.

4.1.2 Pruning and fine-tuning details

In the pruning process, pruning rate is set to set to 1 to ensure that each network can be pruned as much as possible. Step size is set to 0.01 and the intermediate state of deep CNN is preserved to observe the effectiveness of pruning method. In the fine-tuning process, hyper-parameters including batch size, learning rate, weight decay, momentum, and dropout rate for two datasets are set by default, as shown in Table 3. A Large batch size can help optimization, while large momentum can help model avoid local minimum. The learning rates of models trained on ImageNet and CIFAR-10 are reduced by a factor of 10 after 30 and 15 epochs, respectively. The maximum length of paths in AlexNet, VGG-16, GoogLeNet, and ResNet is set to 2, 3, 3, and 3, respectively. All the training and testing process of various deep CNNs are carried out under the Caffe deep learning framework (Jia et al. 2014), based on a workstation consisting of an Intel Core i9-9900k CPU, two NVIDIA GeForce GTX Titan XP GPUs, and 4×16 gigabytes of memory. In order to make a reasonable comparison with baseline and state-of-the-art methods, no data augmentations are used in training process.

4.2 Model fine-tuning and baseline performance comparison

We first plot the learning curves of final fine-tuning process of eight deep CNNs on ImageNet and CIFAR-10 under different pruning rates, including 20%, 40%, 60%, and 80% of parameters being removed, as shown in Fig. 4. For ImageNet, we fine-tune the networks for 150 epochs and report the log-likelihood loss. For CIFAR-10, 70 epochs are sufficient to optimize the network because too much training leads to over-fitting problem. With the increase of pruning rate, i.e., the decrease of model parameters, the final loss of the model increases slightly: from 20 to 80% of pruning rate, the final loss increases by about 0.07

Table 1 Network architectures of baseline models for ImageNet

AlexNet		VGG-16		GoogLeNet		ResNet-34	
Layer	Output	Layer	Output	Layer	Output	Layer	Output
11 × 11 conv, 96	55 × 55	[3 × 3 conv, 64] × 2 2 × 2 max pool	112 × 112	7 × 7 conv, 64	112 × 112	7 × 7 conv, 64	112 × 112
3 × 3 max pool	27 × 27	[3 × 3 conv, 128] × 2 2 × 2 max pool	56 × 56	3 × 3 max pool 3 × 3 conv, 192	56 × 56	3 × 3 max pool [3 × 3 conv, 64] × 6	56 × 56
5 × 5 conv, 256	27 × 27	[3 × 3 conv, 256] × 2 2 × 2 max pool	28 × 28	3 × 3 max pool inception, 256 inception, 480	28 × 28	3 × 3 max pool [3 × 3 conv, 128] × 8	28 × 28
3 × 3 max pool [3 × 3 conv, 384] × 3	13 × 13	[3 × 3 conv, 512] × 3 2 × 2 max pool	14 × 14	3 × 3 max pool [inception, 512] × 3 inception, 528 inception, 832	14 × 14	3 × 3 max pool	14 × 14
3 × 3 max pool	6 × 6	[3 × 3 conv, 512] × 3 2 × 2 max pool	7 × 7	3 × 3 max pool inception, 832 inception, 1024	7 × 7	3 × 3 max pool [3 × 3 conv, 512] × 6	7 × 7
1000 fc1 1000 fc2 1000 fc3, softmax	1 × 1	1000 fc1 1000 fc2 1000 fc3, softmax	1 × 1	Average pool 1000 fc Softmax	1 × 1	Average pool 1000 fc Softmax	1 × 1

Table 2 Parameter size, FLOPs, and classification accuracy of baseline network models

Measurement	Networks for ImageNet					Networks for CIFAR-10		
	AlexNet	VGG-16	GoogLeNet	ResNet-34	ResNet-50	VGG-16	ResNet-56	ResNet-110
Parameters	6.0×10^7	1.4×10^8	7.5×10^6	2.2×10^7	2.6×10^7	1.4×10^7	8.5×10^5	1.7×10^6
FLOPs	7.3×10^7	1.6×10^{10}	1.6×10^9	3.6×10^9	3.8×10^9	3.1×10^8	1.3×10^8	2.5×10^8
Accuracy	68.5%	78.2%	81.3%	82.4%	82.8%	94.7%	95.4%	96.2%

Table 3 Hyper-parameters setting on two benchmark datasets

Hyper-parameters	Batch size	Learning rate	Weight decay	Momentum	Dropout rate
ImageNet	256	0.01	0.0005	0.6	0.5
CIFAR-10	64	0.001	0.0001	0.9	0.5

and 0.015 on ImageNet and CIFAR-10, respectively. Even if most of the parameters of deep CNNs are removed, the optimization of the model is not significantly affected, thanks to the good initialization provided by the previous model.

We now evaluate the full iterative pruning procedure on two image classification tasks. Figure 5 shows the classification results of AlexNet, VGG-16, GoogLeNet, and ResNet-34/56/110 after pruning and fine-tuning on ImageNet and CIFAR-10. Results of ResNet-50 and modified VGG-16 for CIFAR-10 are only used to demonstrate comparisons with some state-of-the-art algorithms, as shown in Sect. 4.3. The figure depicts classification accuracy relative to the pruning rate of model parameters. Baseline methods, including random and minimum weight pruning, minimum activation criterion, L0-regularization and targeted Dropout (Gomez et al. 2018), are compared to illustrate the effectiveness of Drop-path method. In random and minimum weight pruning, the order of parameters to be pruned is randomly permuted and arranged from small to large, respectively. Minimum activation criterion based pruning sorts the activation values of neurons and removes parameters from small to large. The performance of Drop-path in both six models is much higher than that of random and minimum weight pruning, and minimum activation criterion. Even if the parameters are removed more than 60%, the classification accuracy of six models in ImageNet and CIFAR-10 remains above 70% and 80%, respectively. Furthermore, Drop-path method maintains almost the highest classification accuracy on AlexNet, VGG-16, and ResNet-56 under all parameter sizes. L0-regularization based criterion shows the second-best performance which is closely followed by targeted Dropout. The classification accuracy of removing parameters with larger L0-norms decreases quickly as the pruning rate increases, which indicates the importance of parameters with larger L0-norms. Actually, our pruning method can be combined with these baseline techniques. For example, a deep CNN may be first learned with L0-regularization. Then parameters in the small network can be further pruned by using Drop-path.

4.3 Comparison with state-of-the-art methods

In this section, we compare Drop-path with existing state-of-the-art pruning methods on AlexNet, VGG-16, GoogLeNet and ResNet-34/50/56/110, and show results in Table 4 for ImageNet and Table 5 for CIFAR-10. To compare model inference speedup, we report the accuracy loss and the reduction in the number of multiplication and the number of parameters, and denote them as [Accuracy (↓%)], [FLOPs (↓%)], and [Parameters (↓%)] in the Table. FLOPs is a commonly used metric for comparing the computation complexities of various deep CNN models. It is easy to calculate and can be accomplished statically, which is independent of the underlying hardware and software implementations. The trade-off between accuracy loss and the reduction of FLOPs and parameters can be used to illustrate the effectiveness of proposed Drop-path method.

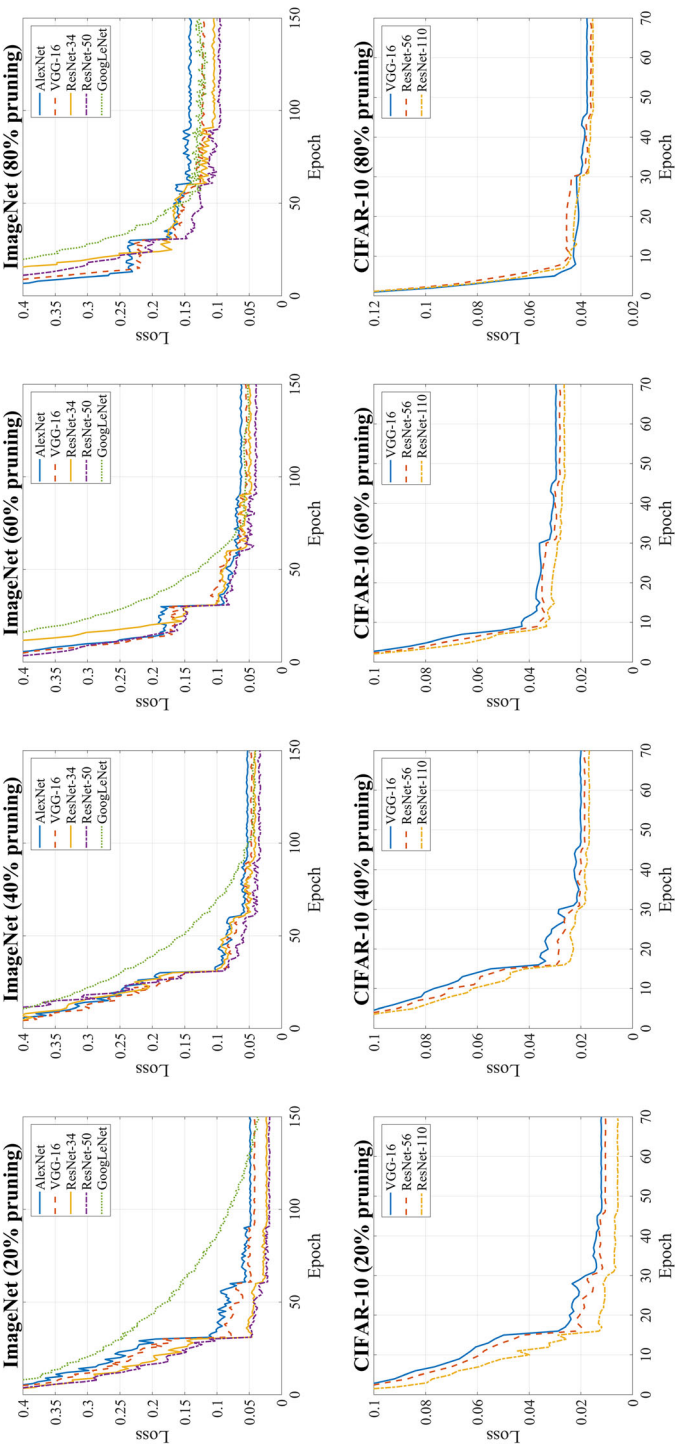


Fig. 4 Learning curves of final fine-tuning process of various network models using Drop-path pruning method (20%, 40%, 60%, and 80% pruning rates) on ImageNet and CIFAR-10

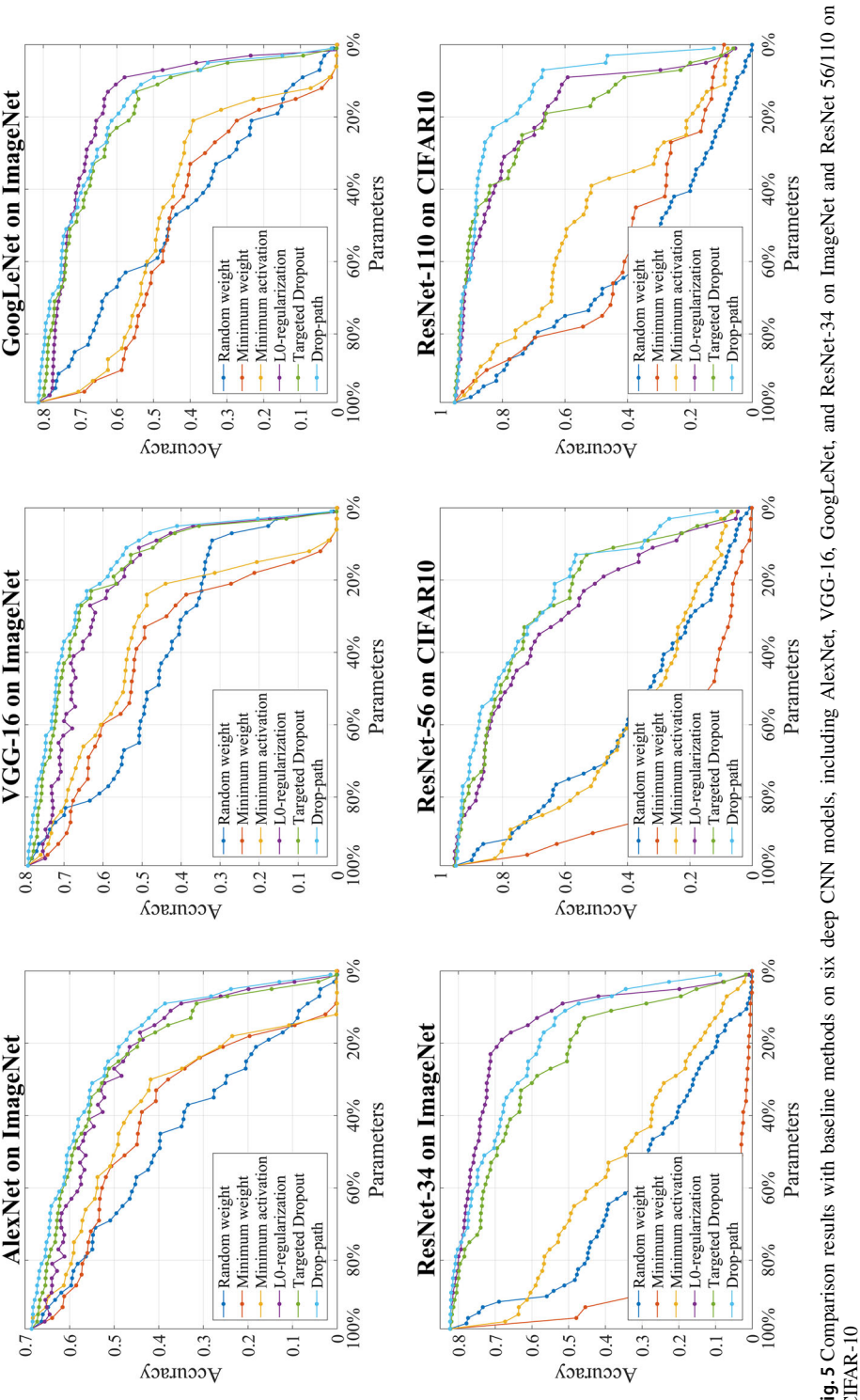


Fig. 5 Comparison results with baseline methods on six deep CNN models, including AlexNet, VGG-16, GoogLeNet, and ResNet-34 on ImageNet and ResNet 56/110 on CIFAR-10

Table 4 ImageNet classification results based on various pruning method

Model	Algorithm	Accuracy (↓%)	FLOPs (↓%)	Parameters (↓%)	Time (ms/image)
AlexNet	Perforated (Figurnov et al. 2016)	2.00	50.00	–	2
	Tucker (Kim et al. 2016)	1.70	62.55	–	0.5
	NISP-A (Yu et al. 2018)	1.43	67.85	33.77	–
	Learning (Srinivas and Babu 2016)	1.20	48.19	–	–
	Drop-path-A1	**1.04**	**69.17**	**62.00**	**11**
	NISP-B (Yu et al. 2018)	0.97	62.69	1.96	–
	NISP-C (Yu et al. 2018)	0.54	53.70	2.91	–
	AlexNet Pruned (Han et al. 2015)	0.01	88.89	89.02	4
	Drop-path-A2	**0.01**	**34.80**	**24.00**	**17**
VGG-16	VGG-16-pruned (Huang and Wang 2018)	3.93	75.24	5.64	7
	Taylor criterion (Molchanov et al. 2017)	2.30	54.61	–	20
	VGG-16 (4 ×) (He et al. 2017)	1.00	60.00	8.31	3.3
	ThiNet-GAP (Luo et al. 2017)	1.00	69.79	93.99	–
	Drop-path-V1	**0.98**	**90.32**	**84.00**	**14**
	VGG-16 Pruned (Han et al. 2015)	0.16	92.31	92.54	4
	50% Pruned (Liu et al. 2017)	0.03	30.40	82.50	–
	Drop-path-V2	**0.01**	**35.57**	**28.00**	**34**
GoogLeNet	Tucker (Kim et al. 2016)	0.24	51.50	31.88	1.83
	NISP (Yu et al. 2018)	0.21	58.34	33.76	–
	Drop-path-G	**0.17**	**64.40**	**58.00**	**11**
ResNet-34	Res34-B (Li et al. 2017)	1.06	24.20	10.80	
	Drop-path-34-R1	**1.03**	**57.81**	**44.00**	**13**

Table 4 continued

Model	Algorithm	Accuracy (↓%)	FLOPs (↓%)	Parameters (↓%)	Time (ms/image)
	NISP-34-B (Yu et al. 2018)	0.92	43.76	43.68	–
	Res34-C (Li et al. 2017)	0.75	7.50	7.20	14
	Res34-A (Li et al. 2017)	0.67	15.50	7.60	12
	NISP-34-A (Yu et al. 2018)	0.28	27.32	27.14	–
	Drop-path-34-R2	**0.27**	**35.46**	**28.00**	**17**
ResNet-50	ThiNet-C (Luo et al. 2017)	4.46	71.50	66.12	11
	ResNet-50-pruned (Luo et al. 2017)	3.82	24.22	33.33	9
	ThiNet-B (Luo et al. 2017)	1.87	55.83	51.56	14
	Drop-path-50-R1	**1.74**	**76.80**	**70.00**	**16**
	NISP-50-B (Yu et al. 2018)	0.89	44.01	43.82	–
	ThiNet-A (Luo et al. 2017)	0.84	36.79	33.67	19
	NISP-50-A (Yu et al. 2018)	0.21	27.31	27.12	–
	Drop-path-50-R2	**0.20**	**23.92**	**18.00**	**23**

[Accuracy (↓%)], [FLOPs (↓%)], and [Parameters (↓%)] represent the accuracy loss, the reduction of computations, and the reduction of parameter numbers, respectively. For the sake of comparison, our method is shown in bold

4.3.1 ImageNet classification

In Table 4, on AlexNet, Drop-path-A1 and Drop-path-A2 represent models with 62% and 24% pruning rates, respectively. By achieving smaller accuracy loss (1.04%), Drop-path-A1 reduces significantly more FLOPs (69.17%) than NISP-A (Yu et al. 2018). Drop-path-A2 can achieve lowest accuracy loss (0.01%) with 33.80% and 24% reduction in FLOPs and parameters, respectively. On VGG-16, Drop-path-V1 achieves similar accuracy loss (0.98% vs 1.00%) with larger FLOPs reduction (90.32%), and Drop-path-V2 achieves lowest accuracy loss (0.01%) with 35.57% and 28% reduction in FLOPs and parameters. Comparing with Tucker (Kim et al. 2016) and NISP (Yu et al. 2018), GoogLeNet with Drop-path-G improves accuracy loss by 0.04% and 0.07%, respectively. Using ResNet, our methods (Drop-path-34-R1 and Drop-path-50-R1) reduce more FLOPs and parameters with smaller accuracy loss, which demonstrate the superior performance when compared with the state-of-the-art methods (Frankle and Carbin 2019; Yu et al. 2018; Li et al. 2017). On the other hand, the test time for a single image of pruned AlexNet, VGG-16, GoogLeNet, ResNet-34

Table 5 CIFAR-10 classification results based on various pruning method

Model	Algorithm	Accuracy (↓%)	FLOPs (↓%)	Parameters (↓%)	Time (ms/image)
VGG-16	One-shot pruning (0.01) (Frankle and Carbin 2019)	0.25	51.17	47.44	–
	One-shot pruning (0.1) (Frankle and Carbin 2019)	0.21	33.00	30.27	–
	VGG-16-pruned-A (Li et al. 2017)	0.15	34.20	64.00	5
	VGGNet (70% pruned) (Liu et al. 2017)	0.14	51.00	88.50	5
	Sparse VGG-16 (Frankle and Carbin 2019)	0.09	37.43	60.00	–
	Drop-path-V	**0.05**	**57.92**	**48.00**	**7**
ResNet-56	One-shot pruning (0.01) (Frankle and Carbin 2019)	0.34	40.82	34.40	–
	One-shot pruning (0.1) (Frankle and Carbin 2019)	0.24	37.20	30.57	–
	NISP-56 (Yu et al. 2018)	0.03	43.61	42.60	–
	Drop-path-56-R1	**0.17**	**60.24**	**54.00**	**11**
	ResNet-56-B (Li et al. 2017)	− 0.02	27.60	13.70	12
	ResNet-56-A (Li et al. 2017)	− 0.06	10.40	9.40	17
	Drop-path-56-R2	**− 0.03**	**14.37**	**6.00**	**18**
ResNet-110	ResNet-110-B (Li et al. 2017)	0.23	38.60	32.40	11
	NISP-110 (Yu et al. 2018)	0.18	43.78	43.25	–
	Drop-path-110-R1	**0.15**	**57.84**	**48.00**	**15**
	ResNet-110-A (Li et al. 2017)	0.02	15.90	2.30	14
	Drop-path-110-R2	**− 0.04**	**17.24**	**4.00**	**21**

Our results in bold to facilitate comparison

and ResNet-50 are reduced from 24, 38, 32, 29 and 36 to 11, 14, 11, 13 and 16 ms. It can be seen that the test time of the pruned model is lower than that of the original model, which shows the effectiveness of the pruning method.

4.3.2 CIFAR-10 classification

On VGG-16 for CIFAR-10 classification, Drop-path achieves 0.05% accuracy loss, while the parameter saving can be up to 2 × and the FLOPs reduction is typically around 50%. It achieves the lowest accuracy loss while reducing the maximum number of FLOPs. On ResNet-56 and 110, classification accuracy under Drop-path-56-R2 and Drop-path-110-R2 even rises slightly (0.03%↑ and 0.04%↑) with a small number of parameters (6% and 4%) removed. We conjecture this is due to the removal of some parameters can help improve the generalization of the "bottleneck" structure. For compression and acceleration, Drop-path-56-R1 and Drop-path-110-R1 removes approximately half of the parameters and achieves almost the same accuracy loss as One-shot pruning (0.1) (Frankle and Carbin 2019) and NISP-110 (Yu et al. 2018). We also observe that with the same accuracy loss, the parameter reduction rate on CIFAR-10 is typically slightly lower than ImageNet, which is possibly due to the fact that ImageNet contains more samples and categories, and thus the deep CNN has more redundancy. The test time for a single image of pruned VGG-16, ResNet-56 and ResNet-110 are reduced from 11, 22 and 24 to 7, 11 and 15 ms.

In theory, both the energy consumption and the inference delay of the network will decrease when the parameters and the amount of computation are reduced. But in practice, they are not simply proportional to these complexity measures. Due to the complexity of physical hardware, the inference time of the large network may be shorter than that of the small network in some cases.

4.4 Layer robustness to drop-path pruning

The more parameters removed, the less the classification accuracy, which is consistent with our understanding. Both layers in the model can be pruned but with different sensitivity. To observe the pruning sensitivity for convolutional layers and fully connected layers of Drop-path, we plot the trade-off curves between classification accuracy and the number of parameters in each layer, as shown in Fig. 6. Preliminary results show that the classification accuracy of deep CNN is related to the total number of parameters in each layer. Moreover, the length of a path is positively correlated with the possibility of removal. The longer the path is, the greater the probability of removal will be, which means that the bottom connections are more important. The majority of parameters of baseline networks reside in the fully or locally connected layers. We remove as many parameters as possible at these higher layers, while the underlying connections are kept untouched.

On AlexNet, even the parameters in last three convolutional layers (i.e., conv3 to conv5) and fully connected layers (fc1 to fc3) are reduced to less than 20%, the accuracy loss is no more than 20%. The convolutional layers are more sensitive to pruning than the fully connected layers, especially the first two convolutional layers conv1 and conv2. The first convolutional layer conv1, which interacts with the input image directly, is undoubtedly the most sensitive to pruning. This sensitivity is due to the fact that the input has only 3 channels and therefore it has less redundancy than the other convolutional layers.

On VGG-16 for ImageNet, the underlying convolutional layers become more important and sensitive. The accuracy loss has exceeded 20%, even if the parameters of layer conv1

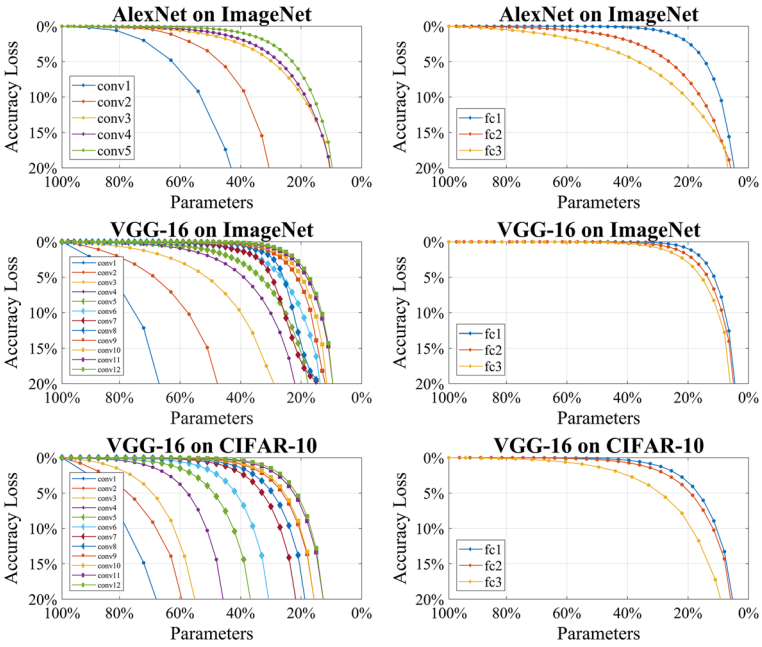


Fig. 6 Pruning sensitivity to convolutional layers (left) and fully connected layers (right) of AlexNet and VGG-16

are still more than 60%. We believe that the deeper the network structure, the greater the importance of underlying building. To prove this empirically, we plot the distribution histogram of removing parameters under 50% pruning rate in Fig. 7. It can be observed that fully connected layers in AlexNet and VGG-16 have a large amount of redundant parameters that can be pruned. Moreover, with the layers going deeper, more and more parameters are removed.

On VGG-16 for CIFAR-10, the pruning sensitivity of convolutional layers is generally higher. There are four convolutional layers (conv1-conv4) with an accuracy loss of more than 20%, even though their parameters are preserved by more than 40%. It can be seen that the sensitivity of fully connected layers and most convolutional layers to Drop-path pruning is acceptable and usually does not produce devastating results.

5 Discussions

5.1 Difficulty in directly using small-sized networks

The classification performance of pruned deep CNNs presented in Sect. 4 are surprising and also raise questions. Since small-sized models have the potential to show good performance, can we directly optimize sparse models without relying on dense models? To answer this question, we report the classification performance of eight pruned sparse networks with random initializations being trained from scratch, as shown in Fig. 8. Networks pruned by Drop-path with a 20% pruning rate are randomly initialized to MSRA Gaussian distribution (i.e., $\boldsymbol{\theta} \sim G\left[0, \sqrt{2/n_{input}}\right]$) (He et al. 2015) and Xavier Uniform distribution (i.e., $\boldsymbol{\theta} \sim U$

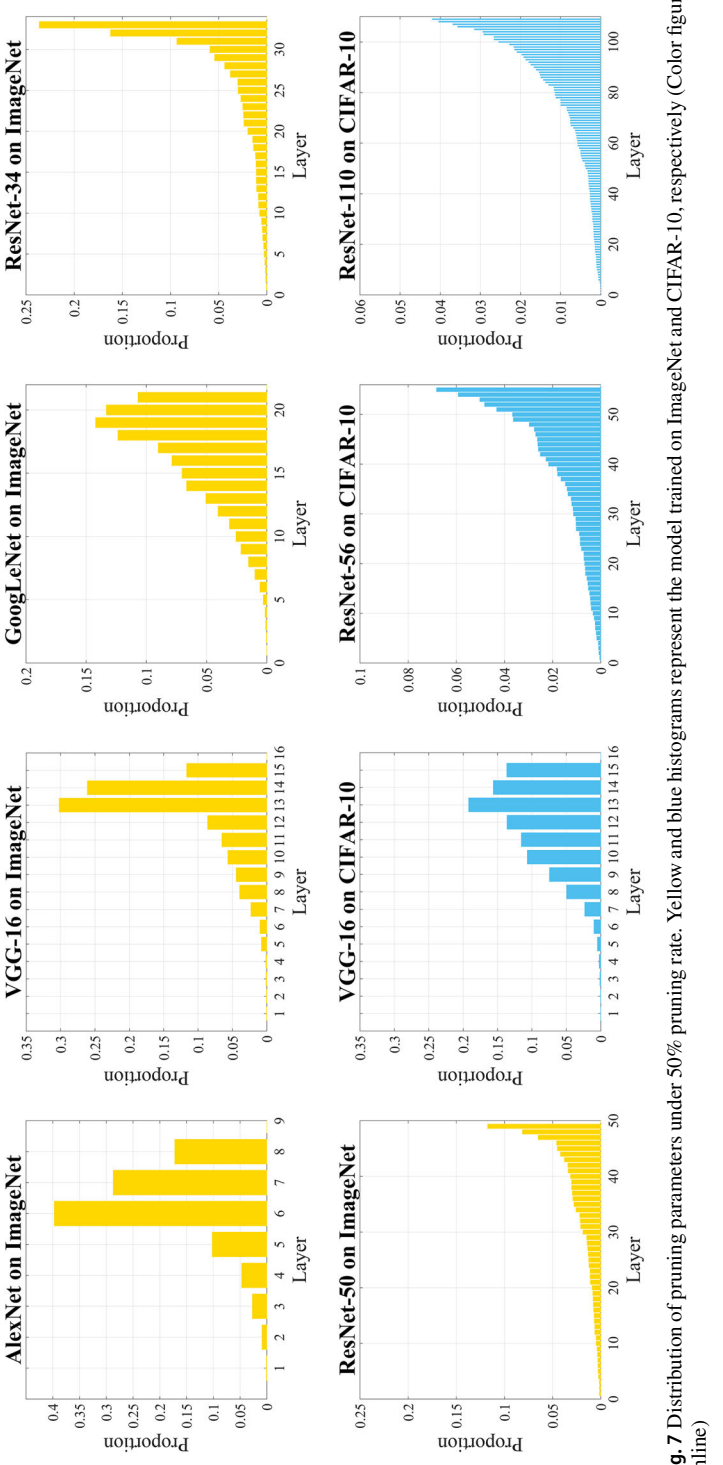


Fig. 7 Distribution of pruning parameters under 50% pruning rate. Yellow and blue histograms represent the model trained on ImageNet and CIFAR-10, respectively (Color figure online)

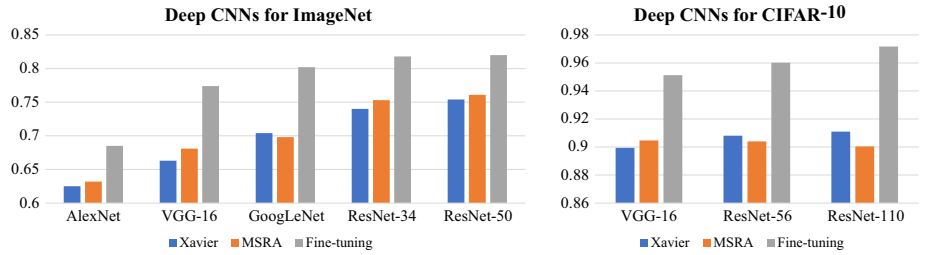


Fig. 8 Classification results of pruned deep CNNs with fine-tuning or reinitialization on two benchmark datasets

$\left[0,\ 2/(n_{input}+n_{output})\right]$) (Glorot and Bengio 2010), where n_{input} and n_{output} represent the number of inputs and outputs in the current layer, respectively. Taking the same pruning rates, networks learned from MSRA or Xavier re-initializations perform significantly worse than the fine-tuned deep CNNs: the accuracy loss on ImageNet and CIFAR-10 has risen by 10% and 5%, respectively. The results demonstrate that the weight initialization learned from the previous network is crucial for continuing to learn sparse networks.

This result is interesting and has enlightened our inference on the optimization and generalization properties of deep CNNs. Although the learning ability of sparse networks is sufficient enough to fit the training images, it is usually easier to fall into local minimum, while the denser network with more parameters can help to explore better initial positions. Once a good initialization position is found by the denser baseline model, fine-tuning the network can improve its classification performance. To see how initialization affects the performance of the network, we plot the loss landscapes of deep CNNs trained on ImageNet in Fig. 9. Noises are added to the parameters along the first two principal component analysis (PCA) directions μ and υ of the weight matrix to observe the loss of the model, which can reflect the local optimization process of deep CNN. If the iterates of SGD are projected onto the plane defined by two random directions, almost none of the motion can be captured. PCA based visualization technique (Li et al. 2018) provides insights into the consequences of a variety of choices facing the neural network practitioner, including network architecture, optimizer selection, and batch size. Then the parameters near the optimal value are calculated as

$$\tilde{\boldsymbol{\theta}}_{(d_1,\,d_2)} = \boldsymbol{\theta}_* + d_1\mu\boldsymbol{E} + d_2\upsilon\boldsymbol{E} \tag{17}$$

where $d_1 \in (-1, 1)$ and $d_2 \in (-1, 1)$ are forward steps along the two directions μ and υ in principal component analysis. $\boldsymbol{E}$ is an all-one matrix which has the same dimensions as weight. Finally, the loss of deep CNN can be updated according to

$$\tilde{\mathcal{L}}(\mathbf{x}_i, \mathbf{y}_i)_{(d_1,\,d_2)} = -\frac{1}{C}\sum_{j=1}^{C}\left[y_i^j \log f(\mathbf{x}_i; \tilde{\boldsymbol{\theta}}_{(d_1,\,d_2)})^j + (1 - y_i^j)\log(1 - f(\mathbf{x}_i; \tilde{\boldsymbol{\theta}}_{(d_1,\,d_2)})^j)\right] \tag{18}$$

It can be seen that model after pruning and fine-tuning has a similar curvature and minimum region as the baseline model, while the re-initialized network is sharper and accompanied by more local minimums, which means that it is difficult to find a good solution for sparse networks without any prior knowledge. Furthermore, the performance of pruned deep CNNs can even be comparable to that of specially designed sparse networks, as shown in Table 6. By fine-tuning, pruned deep CNNs with much fewer parameters outperform the sparse networks, such as SqueezeNet (Iandola et al. 2017) and ShuffleNet (Zhang et al. 2018), which illustrates

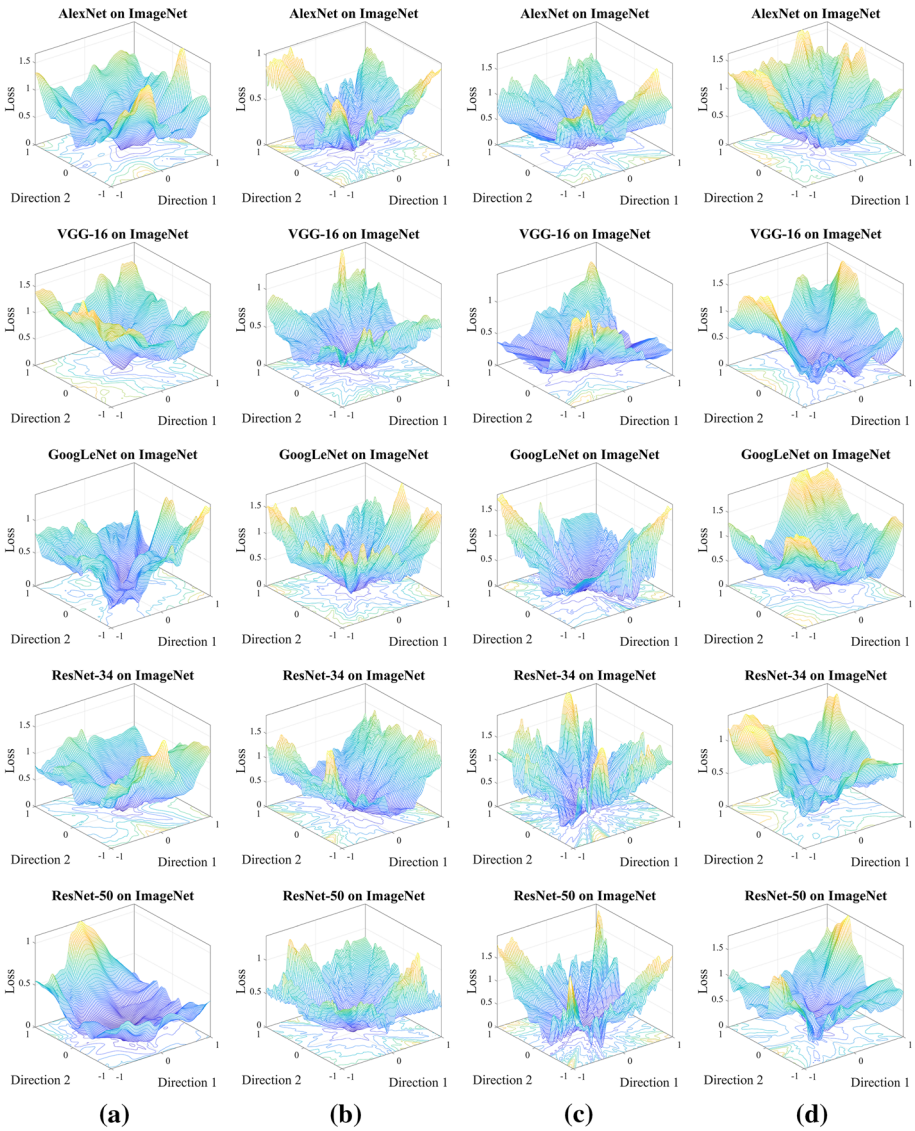


Fig. 9 Loss landscapes of various deep CNN models trained on ImageNet. **a** Original network; **b** MSRA reinitialization, **c** Xavier reinitialization, **d** fine-tuning

that deep CNNs with gradually pruning and fine-tuning are more suitable than directly training from scratch for mobile systems or embedded devices.

5.2 The choice of maximum length of paths in the pruning of deep CNNs

In fact, the choice of maximum length of paths during the pruning process of deep CNNs is critical. Intuitively, the long path containing a set of neurons spanning multiple layers can

Table 6 Comparison of lightweight CNNs and pruned deep CNN models on two benchmark datasets

Dataset	Network model	Accuracy (%)	Storage (MB)
ImageNet	SqueezeNet (Iandola et al. 2017)	57.5	4.8
	Conv MobileNet (Howard et al. 2017)	71.7	29.3
	Xception (Chollet 2017)	79.0	260.6
	ShuffleNet (g=1) (Zhang et al. 2018)	66.4	143.0
	ShuffleNet (g=2) (Zhang et al. 2018)	67.3	140.0
	ShuffleNet (g=3) (Zhang et al. 2018)	67.4	137.0
	ShuffleNet (g=4) (Zhang et al. 2018)	67.2	133.0
	AlexNet-Drop-path		
	(10%)	68.7	206.1
	(20%)	68.5	181.9
	(30%)	67.7	157.3
	VGG-16-Drop-path		
	(10%)	78.0	453.1
	(20%)	77.4	404.6
	(30%)	75.8	352.5
	GoogLeNet-Drop-path		
	(10%)	80.7	26.4
	(20%)	80.2	23.5
	(30%)	79.4	19.7
	ResNet-34-Drop-path		
	(10%)	82.1	74.1
	(20%)	81.8	67.5
	(30%)	80.6	53.9
	ResNet-50-Drop-path		
	(10%)	82.3	91.0
	(20%)	82.0	75.8
	(30%)	81.4	64.6
CIFAR-10	LightweightNet (Xu et al. 2019)	93.53	9.4
	NASNet-A (Zoph et al. 2018)	96.59	3.3
	NASNet-B (Zoph et al. 2018)	97.03	27.6
	VGG-16-Drop-path		
	(10%)	95.22	47.2
	(20%)	95.13	41.1
	(30%)	94.42	33.4
	ResNet-56-Drop-path		
	(10%)	96.14	2.98
	(20%)	96.02	2.51
	(30%)	95.78	1.98
	ResNet-110-Drop-path		
	(10%)	97.11	5.89
	(20%)	97.07	5.21
	(30%)	96.80	3.97

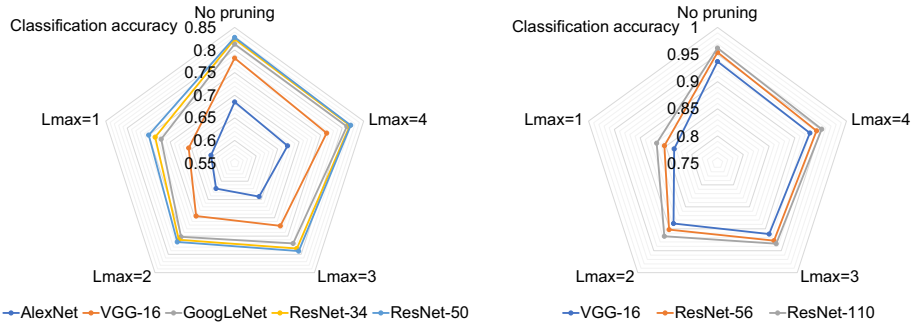


Fig. 10 Classification results of the eight baseline CNNs with various maximum length of paths, including $L_{\max} = 1, 2, 3$ and 4

comprehensively describe the network coding and decoding process and assess the importance of these neurons as a whole. However, this does not mean that each of these neurons has the same level of importance. On the other hand, short paths can locally assess the importance of neurons at other scales and are therefore still necessary. Larger $L_{\max}$ can lead to more paths with different sizes, which are beneficial for assessing the importance of network neurons or connections. Researchers believe that neurons have the ability to encode light, sound, taste, and other information that the body perceives (Carsen et al. 2016). In other words, the neural coding process attempts to establish a mapping from stimulus to response, focusing on understanding how neurons respond to different stimuli and building models to predict the response of neurons to specific stimuli. The corresponding neural decoding process studies the mapping in the opposite direction, that is, from the known responses to reconstruct features and estimate the external stimulus. Context relations play an important role in neural networks (Chen and Jahanshahi 2017; Zheng et al. 2018, 2019), and it is difficult for a single neuron to correctly reflect its importance in the inference process of neural networks.

Then we present the classification performance of the eight baseline CNN models with different maximum lengths of path to illustrate the validity of $L_{\max} = 4$, as shown in Fig. 10. The classification results of the eight networks showed consistency: all the models with $L_{\max} = 4$ achieved the highest classification accuracy under the same pruning rate (~50%). When $L_{\max}$ equals 1, pruning is based on a single neuron and results in a cliff-like decline in the accuracy of fine-tuning networks. During alternate pruning and fine-tuning steps, networks with larger $L_{\max}$ required more local preparation time, which means that more information reflecting the importance of neurons was observed and utilized.

5.3 Sparse convolutional kernels are efficient

Intuitively, directly removing the entire convolutional kernels should be more appropriate than removing the parameters, which guarantees the integrity of the convolution kernel. However, the redundancy of deep CNN model is distributed in each filter, rather than some specific filters are redundant. We present 64 kernels in the first convolutional layer of pruned AlexNet for ImageNet and pruned VGG-16 for CIFAR-10 in Fig. 11. Even when 50% of the parameters are pruned, the number of remaining convolutional kernels is still larger than the number of raw input channels, and only the kernels are split into multiple small sawtooth-like blocks and become sparse. We believe that sparse convolutional kernels play a similar role to Atrous convolution operation (Yu and Koltun 2015), removing redundancy and guaranteeing

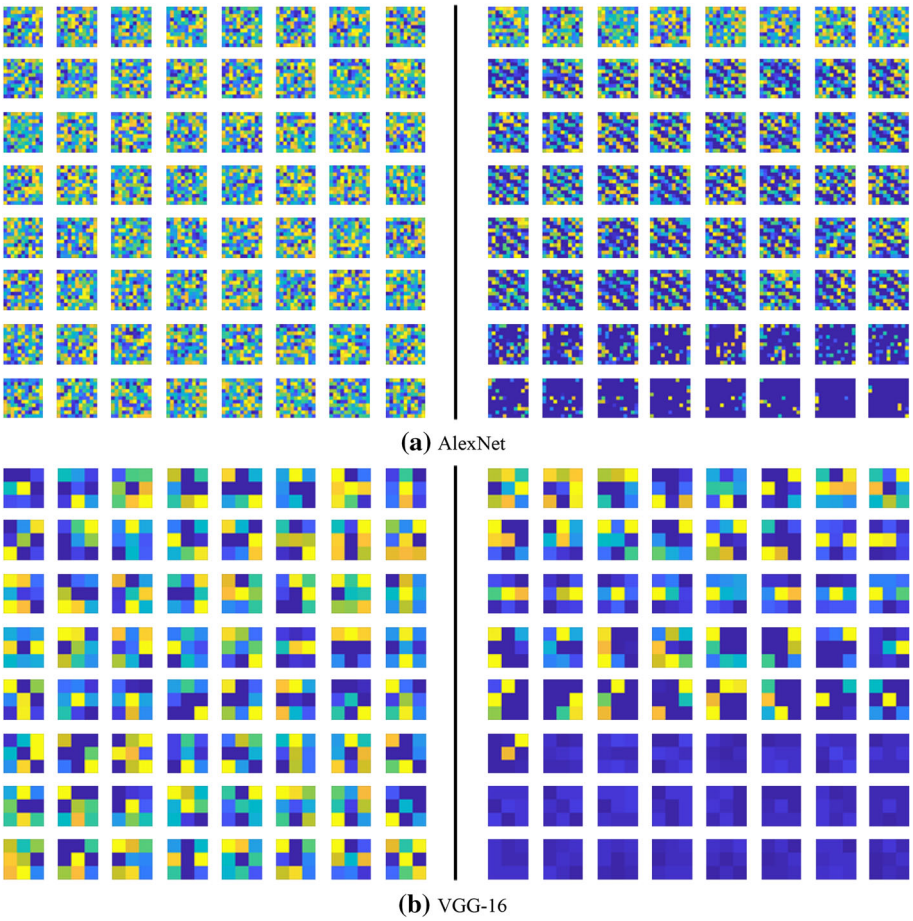


Fig. 11 Visualization of filters in the first convolutional layer of AlexNet and VGG-16 for ImageNet. Pruned convolutional kernels are ranked by L2-norm. The left and right examples represent the original and pruned convolutional kernels, respectively

the mining of potential features. At the same time, this enlarges the field of perception, so that the output of each convolution layer contains a larger range of information. On the other hand, VGG-16 has more inhibited filters than AlexNet as shown in the last two rows in the figure. This is possible for a small-sized dataset such as CIFAR-10, on which the deep CNN does not need to learn as much significative convolutional kernels as on ImageNet.

6 Conclusion

In this paper, we propose a generic approach named Drop-path for network compression and acceleration based on identifying the importance levels of connections in different paths under PAC-Bayesian framework. To the best of our knowledge, this is the first time to reduce model size based on the generalization error boundary. Drop-path is generic and straightforward, which can be easily and suitably applied to any multi-layer and multi-branch

models. Extensive experiments have demonstrated that Drop-path can effectively reduce the redundancy of deep CNN and achieve network compression and acceleration with negligible accuracy loss. Eight popular deep CNNs including AlexNet, VGG-16, GoogLeNet, and ResNet-34/50/56/110 trained on ImageNet and CIFAR-10 achieve the state-of-the-art performance by about 2 $\times$ speed up along with no more than 1% increase of error. This results in smaller memory footprint and computational requirements for real-time image processing, making the deep CNN easier to be deployed on mobile systems or embedded devices. Moreover, the proposed Drop-path method can be viewed as a tool to further explore the dependence of model structure on optimization and generalization of neural networks. In the process of theoretical analysis and experimental verification, we conclude:

1. By iteratively removing the least important parameters in different paths, deep CNNs can be successfully pruned by ensuring the invariance of network generalization error boundary as much as possible.
2. Automatic structured pruning can usually find effective network architecture, which performs better than directly training from scratch of sparse models.
3. Sparse convolutional kernels are more efficient than being removed directly as a whole.

As for future work, we plan to apply our approach to more deep learning based applications, such as real-time object detection and instance segmentation. It is also interesting to use Drop-path method to accelerate more advanced neural networks, such as recurrent neural network (RNN) and 3D CNN. In these cases, the approximation method used to estimate the generalization error boundary of neural networks needs to be redesigned.

Acknowledgements This work was supported by National Key R&D Program of China (Grant No. 2018YFC0831503), National Natural Science Foundation of China (Grant No. 61571275), China Computer Program for Education and Scientific Research (Grant No. NGII20161001), and Fundamental Research Funds of Shandong University (Grant No. 2018JC040).

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

Bolukbasi, T., Wang, J., Dekei, O., & Saligrama, V. (2017). Adaptive neural networks for fast test-time prediction. In *Proceedings of 34th international conference on machine learning (ICML)* (pp. 527–536), Sydney.

Carsen, S., Marius, P., Nicholas, A. S., Michael, O., Peter, B., et al. (2016). Inhibitory control of correlated intrinsic variability in cortical networks. *Elife*. https://doi.org/10.7554/elife.19695.

Chen, F. C., & Jahanshahi, R. J. (2017). NB-CNN: Deep learning-based crack detection using convolutional neural network and naive Bayes data fusion. *IEEE Transactions on Industrial Electronics, 65*(5), 4392–4400.

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 1800–1807), Honolulu.

Denton, E., Zaremba, W., Bruna, J., LeCun, Y., & Fergus, R. (2014). Exploiting linear structure within convolutional networks for efficient evaluation. In *Proceedings of conference and workshop on neural information processing systems (NIPS)*, Montreal, http://papers.nips.cc/paper/5544-exploiting-linear-structure-within-convolutional-networks-for-efficient-evaluation.pdf.

Figurnov, M., Ibraimova, A., & Dmitry, P. V. (2016). PerforatedCNNs: Acceleration through elimination of redundant convolutions. In *Proceedings of conference and workshop on neural information processing systems (NIPS)* (pp. 1–9), Barcelona. https://arxiv.org/abs/1504.08362.

Frankle, J., & Carbin, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *Proceedings of international conference on learning representations (ICLR)*. https://openreview.net/forum?id=rJl-b3RcF7.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of 13th international conference on artificial intelligence and statistics (AISTATS)* (pp. 249–256), Sardinia.

Goh, H., Thome, N., Cord, M., & Lim, J. (2014). Learning deep hierarchical visual feature coding. *IEEE Transactions on Neural Networks and Learning Systems, 25*(12), 2212–2225.

Gomez, N. A., Zhang, I., Swersky, K., Gal, Y., & Hinton, G. E. (2018). Targeted dropout. In *Proceedings of conference and workshop on neural information processing systems (NIPS)*. https://nips.cc/Conferences/2018/Schedule?showEvent=10941.

Gutierrez-Galan, D., Dominguez-Morales, J. P., Cerezuela-Escudero, E., Rios-Navarro, A., Tapiador-Morales, R., Rivas-Perez, M., et al. (2018). Embedded neural network for real-time animal behavior classification. *Neurocomputing, 272,* 17–26.

Han, S., Mao, H., & Dally, W. J. (2016). Deep compression: Compressing DNNs with pruning, trained quantization and huffman coding. In *Proceedings of international conference on learning representations (ICLR)*, San Juan. https://arxiv.org/abs/1510.00149.

Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural networks. In *Proceedings of conference and workshop on neural information processing systems (NIPS)* (pp. 1–9), Montreal, Canada. http://papers.nips.cc/paper/5784-learning-both-weights-and-connections-for-efficient-neural-network.pdf.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of IEEE international conference on computer vision (ICCV)* (pp. 1026–1034), Santiago.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition ResNet. In *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 770–778), Las Vegas.

He, Y., Zhang, X., & Sun, J. (2017). Channel pruning for accelerating very deep neural networks. In *Proceedings of IEEE international conference on computer vision (ICCV)* (pp. 1398–1406), Venice.

Herbrich, R., & Graepel, T. (2002). A PAC-Bayesian margin bound for linear classifiers". *IEEE Transactions on Information Theory, 48*(12), 3140–3150.

Howard, A. G., Zhu, M., Chen, B., & Kalenichenko, D. (2017). *MobileNets: Efficient convolutional neural networks for mobile vision applications*. arXiv preprint, https://arxiv.org/abs/1704.04861.

Hu, Y., Li, C., Meng, K., Qin, J., & Yang, X. (2017). Group sparse optimization via l p, q, regularization. *Journal of Machine Learning Research, 8*(30), 960–1011.

Huang, Z., & Wang, N. (2018). Data-driven sparse structure selection for deep neural networks. In *Proceedings of European conference on computer vision (ECCV)* (pp. 317–334), Munich.

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2017). SqueezeNet: AlexNet-Level accuracy with 50X fewer parameters and < 0.5 MB model size. In *Proceedings of international conference on learning representations (ICLR)*, Toulon. https://openreview.net/pdf?id=S1xh5sYgx.

Jang, H., & Lee, J. (2018). An empirical study on modeling and prediction of bitcoin prices with Bayesian neural networks based on blockchain information. *IEEE Access, 6,* 5427–5437.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., et al. (2014). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of 22nd ACM international conference on multimedia* (pp. 675–678), Florida.

Jie, W., & Wang, J. (2017). Forecasting stochastic neural network based on financial empirical mode decomposition. *Neural Networks, 90,* 8–20.

Kim, Y., Park, E., Yoo, S., Choi, T., Yang, L., & Shi, D. (2016). Compression of deep convolutional neural networks for fast and low power mobile applications. In *Proceedings of international conference on learning representations (ICLR)*, Caribe Hilton. https://arxiv.org/abs/1511.06530.

Krizhevsky, A., & Hinton, G. E. (2009). Learning multiple layers of features from tiny images. *Technical Report*, *1*(4), p. 7, University of Toronto, Toronto, Canada.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Proceedings of conference and workshop on neural information processing systems (NIPS)*. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

Langford, J., & Schapire, R. (2015). Tutorial on practical prediction theory for classification. *Journal of Machine Learning Research, 6*(3), 273–306.

Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436–444.

Li, H., Kadav, A., Durdanovic, I., Samet, H., & Graf, H. P. (2017). Pruning filters for efficient convnets. In *Proceedings of international conference on learning representations (ICLR)*, Toulon. https://openreview.net/pdf?id=rJqFGTslg.

Li, H., Xu, Z., Taylor, G., & Goldstein, T. (2018). Visualizing the loss landscape of neural nets. In *International conference on learning representations workshop (ICLRW)*, Vancouver, BC, Canada (pp. 1–17).

Li, Y., Yin, G., Zhuang, W., Zhang, N., Wang, J., & Geng, K. (2018b). Compensating delays and noises in motion control of autonomous electric vehicles by using deep learning and unscented Kalman predictor. *IEEE Transactions on Systems, Man, and Cybernetics: Systems,*. https://doi.org/10.1109/TSMC.2018.2850367.

Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., & Zhang, C. (2017). Learning efficient convolutional networks through network slimming. In *Proceedings of IEEE international conference on computer vision (ICCV)* (pp. 2755–2763), Venice.

Luo, J., Wu, J., & Lin, W. (2017). Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of IEEE international conference on computer vision (ICCV)* (pp. 5068–5076), Venice.

Miao, H., & He, D. (2017). Deep learning based approach for bearing fault diagnosis. *IEEE Transactions on Industry Applications, 53*(3), 3057–3065.

Molchanov, P., Tyree, S., Karras, T., Aila, T., & Kautz, J. (2017). Pruning convolutional neural networks resource efficient inference. In *Proceedings of international conference on learning representations (ICLR)*, Toulon. https://openreview.net/forum?id=SJGCiw5gl.

Painsky, A., & Rosset, S. (2016). Isotonic modeling with non-differentiable loss functions with application to Lasso regularization. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 38*(2), 308–321.

Radosavovic, L., Dollár, P., Girshick, R., Gkioxari, G., & He, K. (2018). Data distillation: Towards omni-supervised learning. In *Proceedings of IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 4119–4128), Salt Lake City.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., et al. (2015). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision, 115*(3), 211–252.

Samala, R. K., Chan, H., Hadjiiski, L., Helvie, M. A., Richter, C. D., & Cha, K. H. (2019). Breast cancer diagnosis in digital breast Tomosynthesis: effects of training sample size on multi-stage transfer learning using deep neural nets. *IEEE Transactions on Medical Imaging, 38*(3), 686–696.

Sau, B. B., & Balasubramanian, V. N. (2016). *Deep model compression: Distilling knowledge from noisy teachers*. arXiv preprint, https://arxiv.org/abs/1610.09650.

Shin, H. C., Roth, H. R., Gao, M., Lu, L., Xu, Z., Nogues, I., et al. (2016). Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. *IEEE Transactions on Medical Imaging, 35*(5), 1285–1298.

Simonyan, K., & Zisserman, A. (2014). *Very deep convolutional networks for large-scale image recognition*. arXiv preprint, https://arxiv.org/abs/1409.1556.

Srinivas, S., & Babu, R. V. (2016). Learning the architecture of deep neural networks. In *Proceedings of international conference on learning representations (ILCR)*, Caribe Hilton. https://arxiv.org/abs/1511.05497v1.

Sun, X., Ren, X., Ma, S., & Wang, H. (2017). meProp sparsified backpropagation for accelerated deep learning with reduced overfitting. In *Proceedings of 34th international conference on machine learning (ICML)* (pp. 3299–3308), Sydney.

Sun, Y., Wang, X., & Tang, X. (2016). Sparsifying neural network connections for face recognition. In *Proceedings of IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 4856–4864), Las Vegas.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., et al. (2015). Going deeper with convolutions. In *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 1–9), Boston.

Theis, L., Korshunova, I., Tejani, A., & Huszár, F. (2018). *Faster gaze prediction with dense networks and Fisher pruning*. arXiv preprint, https://arxiv.org/abs/1801.05787.

Tian, Q., Arbel, T., & Clark, J. J. (2017). Deep LDA-pruned nets for efficient facial gender classification. In *Proceedings of IEEE conference on computer vision and pattern recognition workshops (CVPRW)* (pp. 512–521), Honolulu.

Torfi, A., & Shirvani, R. A. (2018). Attention-based guided structured sparsity of deep neural networks, In *Proceedings of international conference on learning representations workshops (ICLRW)*, Canada. https://openreview.net/pdf?id=S1dGIXVUz.

Wang, J., Xu, C., Yang, X., & Zurada, J. M. (2018). A novel pruning algorithm for smoothing feedforward neural networks based on group Lasso method. *IEEE Transactions on Neural Networks and Learning Systems, 29*(5), 2012–2024.

Xu, T., Yang, P., Zhang, X., & Liu, C. (2019). LightweightNet: toward fast and lightweight convolutional neural networks via architecture distillation. *Pattern Recognition, 88,* 272–284.

Yang, T. J., Chen, Y. H., & Sze, V. (2017). Designing energy-efficient convolutional neural networks using energy-aware pruning. In *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 6071–6079), Honolulu.

Yu, F., & Koltun, V. (2015). *Multi-scale context aggregation with dilated convolutions*. arXiv preprint, https://arxiv.org/abs/1511.07122v2.

Yu, R., Li, A., Chen, C. F., Lai, J., Morariu, V. I., Han, X., et al. (2018). NISP: Pruning networks using neuron importance score propagation. In *Proceedings of IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 9194–9203), Salt Lake City.

Zhang, X., Zhou, X., Lin, M., & Sun, J. (2018). ShuffleNet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 6848–6856), Salt Lake City.

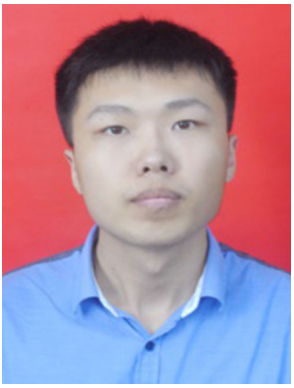
Zheng, Q., Tian, X., Yang, M., & Wang, H. (2019). Differential learning: a powerful tool for interactive content-based Image Retrieval. *Engineering Letters, 27*(1), 202–215.

Zheng, Q., Yang, M., Zhang, Q., & Yang, J. (2018a). A bilinear multi-scale convolutional neural network for fine-grained object classification. *IAENG International Journal of Computer Science, 45*(2), 340–352.

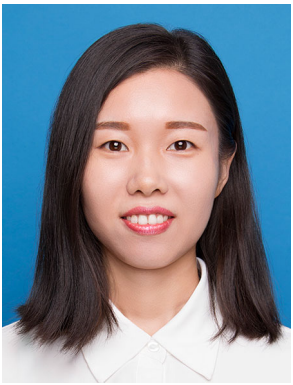
Zheng, Q., Yang, M., Zhang, Q., & Zhang, X. (2018b). Improvement of generalization ability of deep CNN via implicit regularization in two-stage training process. *IEEE Access, 6,* 15844–15869.

Zheng, Q., Yang, M., Zhang, Q., Zhang, X., & Yang, J. (2017). Understanding and boosting of deep convolutional neural network based on sample distribution, In *IEEE Information Technology, Networking, Electronic and Automation Control Conference (ITNEC). Chengdu, China, 2017,* 823–827.

Zoph, B., Vasudevan, V., Shlens, J., & V. Le, Q. (2018). Learning transferable architectures for scalable image recognition. In *Proceedings of IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 8697–8710), Salt Lake City.



Qinghe Zheng was born in Jining, Shandong, China in 1993. He received the B.E. degree in communication engineering from Xian University of Posts and Telecommunications, Xian, China, in 2014, and the M.Eng. degree in information and communication engineering from Shandong University, Jinan, China, in 2018. He is currently pursuing the Ph.D. degree in information and communication engineering at Shandong University, Qingdao, China. He is a member of the Intelligent Visual Laboratory. He has authored more than 10 peer-reviewed journal and conference papers on his research topic. His research interests include non-stationary signal processing, pattern classification, and non-convex optimization.



Xinyu Tian was born in Taian, Shandong, China in 1992. She received the B.E. degree in electronic communication engineering from Shandong Jiaotong University, Jinan, China, in 2014, and the M.Eng. degree in electronic communication engineering from Shandong University, Jinan, China, in 2018. She is currently a Teaching Assistant with Electrical and Mechanical College, Shandong Management University. Her research interests include computer vision, intelligent transportation, path planning, and autonomous vehicles.

Mingqiang Yang was born in Jinan, Shandong, China in 1969. He received the B.E. degree in radio technology from Shandong University of Technology, Jinan, China, in 1992, and the M.Eng. degree in communication and information processing from Shandong University, Jinan, China, in 2000. He received the Ph.D. degree in signal and image processing from INSA-RENNES, Rennes, France, in 2008. From 2001 to 2004, he was a Research Assistant with Shandong University. Since 2009, he is an Associate Professor with the School of Information Science and Engineering, Shandong University. He is the author of two books, more than 60 peer-reviewed journal and conference papers, and more than 10 patents. His research interests include image processing, feature extraction, and pattern recognition.

Yulin Wu received B.E. and M.Eng. degrees in electronics and communication engineering from Shandong University, China, in 2015 and 2018, respectively. He is currently a Ph.D. student at the School of Information Science and Engineering, Shandong University. His research interests include deep learning, image processing and computer vision, such as image classification and detection.

Huake Su was born in in Taining, Fujian, China in 1997. He received B.E. degree in School of Microelectronics from Xidian University, China, in 2019. He is currently pursuing the M. Eng. degree at the School of Microelectronics, Xidian University. His research interests include machine learning, integrated circuit design and semiconductor optoelectronic devices, such as LED.