Self-Distillation: Towards Efficient and Compact Neural Networks

Linfeng Zhang, Chenglong Bao, *Member, IEEE*, and Kaisheng Ma

Abstract—Remarkable achievements have been obtained by deep neural networks in the last several years. However, the breakthrough in neural networks accuracy is always accompanied by explosive growth of computation and parameters, which leads to a severe limitation of model deployment. In this paper, we propose a novel knowledge distillation technique named self-distillation to address this problem. Self-distillation attaches several attention modules and shallow classifiers at different depths of neural networks and distills knowledge from the deepest classifier to the shallower classifiers. Different from the conventional knowledge distillation methods where the knowledge of the teacher model is transferred to another student model, self-distillation can be considered as knowledge transfer in the same model - from the deeper layers to the shallow layers. Moreover, the additional classifiers in self-distillation allow the neural network to work in a dynamic manner, which leads to a much higher acceleration. Experiments demonstrate that self-distillation has consistent and significant effectiveness on various neural networks and datasets. On average, 3.49 and 2.32 percent accuracy boost are observed on CIFAR100 and ImageNet. Besides, experiments show that self-distillation can be combined with other model compression methods, including knowledge distillation, pruning and lightweight model design.

Index Terms—Knowledge distillation, model acceleration, model compression, dynamic neural networks, multi-exit neural networks, attention, image classification

1 Introduction

Deep convolutional neural networks have shown promising results in many applications such as image classification [1], [2], [3], [4], object detection [5], [6], [7], [8] and segmentation [9], [10], [11]. However, to achieve a good performance, modern convolutional neural networks always require a tremendous amount of computation and storage, which has severely limited their deployments on resource-limited devices and real-time applications.

In recent years, this problem has been extensively explored and numerous model compression and acceleration methods have been proposed to address this issue. Typical approaches include pruning [12], [13], [14], [15], [16], quantization [17], [18], [19], [20], light-weight neural network design [21], [22], [23], [24], [25], low rank factorization [26], [27] and knowledge distillation [28], [29], [30], [31]. Among them, knowledge distillation is one of the most effective approaches, which first trains an over-parametrized neural network as a teacher and then trains a small student network to mimic the output of the teacher network. Since the student model has inherited the knowledge of teachers, it can replace the over-parameterized teacher model to achieve model compression and fast inference. However, the conventional knowledge distillation has been suffering from two problems - the choice of teacher models and the efficiency of knowledge transferring. Recently, researchers find that the choice of teacher models has a great impact on the accuracy of student models and the teacher with the highest accuracy is not the best teacher for distillation [32], [33], [34]. As a result, substantial experiments are needed to search for the most proper teacher model for distillation, which can be very time-consuming. The second problem of knowledge distillation is that the student models can not always achieve as high accuracy as teacher models do, which may lead to an unacceptable accuracy degradation in the inference period. In other words, it's still hard to get an accurate, efficient and compact student model at the same time.

To address these problems, a novel knowledge distillation method named self-distillation is proposed in this paper. Self-distillation first attaches several attention-based shallow classifiers after the intermediate layers of neural networks at different depths. Then, in the training period, the deeper classifiers are regarded as the teacher models and they are utilized to guide the training of student models by a KL divergence loss on the outputs and L2 loss on the feature maps. In the inference period, all of the additional shallow classifiers can be dropped so they don't bring additional parameters and computation.

Self-distillation reduces the training overhead compared with conventional knowledge distillation. Since both teacher models and student models in the proposed self-distillation are the classifiers in the same neural networks, substantial experiments for searching the teacher model in conventional knowledge distillation can be avoided. Moreover, conventional knowledge distillation is a two-stage training method where we have to train a teacher first and then use the

- *Linfeng Zhang and Kaisheng Ma are with the Institute of Interdisciplinary Information Sciences, Tsinghua University, Beijing 100084, China. E-mail: zhang-lf19@mails.tsinghua.edu.cn, merrydoudou@gmail.com.*
- *Chenglong Bao is with the Yau Mathematical Sciences Center, Yanqi Lake Beijing Institute of Mathematical Sciences and Applications, Tsinghua University, Beijing 100084, China. E-mail: clbao@mail.tsinghua.edu.cn.*

Manuscript received 19 May 2020; revised 11 Mar. 2021; accepted 13 Mar. 2021. Date of publication 18 Mar. 2021; date of current version 1 July 2022.
(Corresponding authors: Chenglong Bao and Kaisheng Ma.)
Recommended for acceptance by T. Liu.
Digital Object Identifier no. 10.1109/TPAMI.2021.3067100

teacher to train the student. In contrast, self-distillation is a one-stage training method where the teacher model and student models can be trained together. The one-stage property of self-distillation further reduces training overhead.

Self-distillation achieves higher accuracy, acceleration and compression compared with conventional knowledge distillation. Different from the conventional knowledge distillation which focuses on knowledge transfer among different models, the proposed self-distillation tries to transfer the knowledge within one model. Experiments show that self-distillation outperforms other knowledge distillation methods by a large margin. Moreover, we also find that the self-distillation and conventional knowledge distillation methods can be utilized together to achieve better results.

Self-distillation allows neural networks to perform dynamic inference according to the input image, which leads to higher acceleration. In the multi-classifier neural networks trained by self-distillation, the deep classifiers can produce more accurate classification results while the shallow classifiers can give quick classification results with slightly lower accuracy. Based on these observations, we further present a dynamic inference mechanism which allows the shallow classifiers to give prediction to easy images and deep classifiers to predict images which are more difficult to be classified. For example, more than 95 percent images in CIFAR100 dataset can be classified by the shallowest classifier of ResNet18 with 3 percent higher accuracy and 3 times acceleration than the baseline model.

Abundant experiments show that the proposed self-distillation achieves a significant and consistent accuracy boost in various datasets and neural networks, including the lightweight models such as MobileNetV2 and ShuffleNetV2. Moreover, we also prove that self-distillation can be utilized to improve the results of neural networks pruning.

To sum up, the main contribution of this paper can be summarized as follows.

- A novel distillation method named self-distillation is proposed to achieve accurate, efficient and compact neural networks with much less training overhead than the conventional knowledge distillation methods. Moreover, we prove that self-distillation and other distillation methods can be utilized together to achieve higher accuracy.
- Based on self-distillation, we proposed a threshold-controlled dynamic inference mechanism which allows easy images to be classified by shallow classifiers and hard images by deeper classifiers. Experiments show that dynamic inference provides higher acceleration with no accuracy loss.
- We evaluate self-distillation in various datasets and neural networks and compare it with other state-of-the-art knowledge distillation methods. Sufficient experiments have been conducted to study how different factors influence self-distillation and how self-distillation influences different model compression methods.

The rest of this paper is organized as follows. Section 2 introduces related work of self-distillation. Section 3 demonstrates the formulation and details of self-distillation. Section 4 shows experiment results in different convolutional networks and datasets. Section 5 explains the reason why self-distillation works. Finally, a conclusion is brought forth in Section 6.

2 Related Work

2.1 Knowledge Distillation

As one of the most utilized techniques in deep learning, knowledge distillation has attracted more and more attention to its methodology, application and rationale. The idea that employing larger models to guide the training of smaller models was first proposed by Buciluǎ *et al.* for the compression of ensemble models [28]. Hinton *et al.* then extended this idea to neural networks and proposed “distillation” concept first [29]. Then, fruitful distillation methods have been proposed to transfer the knowledge of teacher models to student models based on feature maps [35], attention [36], the flow of solution procedure [37], graph [38] and GAN-based methods [39], [40], [41]. Recently, Park *et al.* and Tung *et al.* have proposed a relation-based method, which has built a connection between teachers and students on not only the individual sample but also the relation between samples in a training batch [42], [43]. How to choose the best teacher model for a student model is an unsolved problem in knowledge distillation. Substantial researches demonstrate that the gap of accuracy between teacher and student model can harm the efficiency of knowledge distillation [32], [33]. To address this problem, Mirzadeh *et al.* propose the TAKD method, which employs several teacher assistant models as the bridges of distillation [32]. Jin *et al.* present the RCO distillation to employ various teacher models with different accuracy rates for a single student model [44]. During the training period, the choice of teacher models will be changed according to the accuracy of student models. Moreover, Jang *et al.* propose L2T, which exploits a meta neural network to learn the best layers and weights for knowledge transferring [45]. Reinforcement learning based technique is also presented to search for the best teacher model for knowledge distillation [46].

Besides neural network compression and acceleration, knowledge distillation has also been available in other circumstances. BAN [47] improves model accuracy by sequentially training several student models. Bagherinezhad *et al.* exploit knowledge distillation to refine the quality of labels, achieving significant accuracy gain in classification [48]. Wang *et al.* and Liu *et al.* apply knowledge distillation on objection detection, segmentation, depth prediction and other vision tasks [39], [49]. Gupta *et al.* propose the cross modal knowledge distillation, which guides neural networks training on unlabelled depth prediction and optical flow images with a teacher trained on labelled RGB images [50]. Ge *et al.* regard neural networks trained on high resolution images as the teachers of neural networks trained on low resolution, achieving both accurate and rapid face recognition [51]. Moreover, knowledge distillation is also utilized in the neural network architecture searching [34], semi-supervised learning [52] and distributed neural network training [34]. The success of knowledge distillation also promotes the theoretical study on its rationale. David *et al.* unify the knowledge distillation with privileged information [53], [54]. Tian has proven that the student node in distillation specializes to teacher nodes in the lowest layer under mild conditions and over-parameterization is a necessary condition for specialization [55].

2.2 Other Model Compression Techniques

2.2.1 Neural Network Pruning

Neural networks pruning is one of the most effective techniques in model compression which aims at deleting the unnecessary weights of over-parameterized models. LeCun *et al.* and Hassibi *et al.* prune the weights of neural networks based on the Hessian of the loss function [56], [57]. Han *et al.* propose the deep compression pipeline, which prunes the low magnitude weights in convolution neural networks [12]. Zhang *et al.* apply ADMM [14] to achieve both structure and irregular pruning. Louizos *et al.* train neural networks with the L_0 loss to improve the sparsity of weights [58]. Meta pruning is proposed to prune the channels of neural networks by a meta model automatically [59]. Liu *et al.* point out that neural network pruning may be a possible way searching for the optimal neural network architectures [15]. Frankle *et al.* propose the lottery ticket hypothesis, which reveals that the dense, randomly-initialized and feed-forward networks containing subnetworks can reach test accuracy that is comparable to the original network in a similar number of iterations [60].

2.2.2 Neural Network Quantization

Neural network quantization targets to quantize the weights and activation in neural networks to achieve model compression and acceleration by low bits. Matthieu *et al.* propose the binary connected neural networks which quantize the weights by 1 bit [17]. XNOR-Net computes the convolution operation by XNOR and bit counting, which further improves the efficiency and accuracy of binary neural networks [18]. Li *et al.* propose the ternary weights neural networks, which constrains the weights to 0, +1 and -1 [61] Zhou *et al* propose the INQ method, targeting to convert convolutional neural networks into a low-precision model whose weights are constrained to be the powers of two [62]. Leng *et al.* formulate the weights quantization as a discretely constrained optimization problem and solve it with ADMM [14]. Markus *et al.* quantize neural networks without touching the training data through weight equalization and bias correction [63].

2.2.3 Lightweight Model Design

Recently, more and more attention has been paid to design models that are not only accurate but also compact and efficient. SqueezeNet [25] compresses the convolutional neural networks with a novel “fire” module, which reduces and expands the number of channels with 1x1 convolution layers. Depthwise and pointwise convolutional layers, inverted residuals and linear bottleneck layers are proposed in MobileNetV1 [21] and MobileNetV2 [22] respectively to achieve model compression and acceleration while maintaining its accuracy. Ma *et al.* propose ShuffleNet which replaces convolutional layers with group convolution and improves its accuracy by channels shuffling. Qin *et al.* present the ThunderNet which can achieve real-time object detection on edge devices. Chen *et al.* study this problem from the perspective of frequency and propose the Octave convolution, which can be utilized as a direct replacement of conventional convolutions. Recently, AutoML methods have also been applied to obtain lightweight models such as MobileNetV3 [64], AMC [65] and so on.

2.3 Adaptive Computation

Adaptive computation, which is based on the principle that neural networks should classify the images correctly at the least cost of computation and energy, is a rising star in the study of neural network acceleration [66], [67]. Since the recognition difficulty of images in the real world has an overlarge difference, adaptive computation methods can earn substantial acceleration on the images which are simple to be classified. SkipNet [68] targets to skip the unnecessary layers in deep convolutional neural networks dynamically. A hybrid reinforcement learning method is utilized to train gating units which decide whether a layer should be skipped or not. BlockDrop [69] learns a policy to select the minimal number of blocks which are needed to correctly classify the given image by Markov decision process. MSDNet [70] employs DenseNet [71] with several intermediate classifiers to achieve anytime prediction and budged computation. Slimmable neural networks [72], [73] propose the switchable batch normalization, which permits models running with different channels. D^2NN employs a specific controlling layer which is trained by Q-learning to decide the routing path of neural networks computation.

The previous study on adaptive computation mainly focuses on how to decide which layer or block should be skipped for specific sample. To solve this problem, complex reinforcement learning control units are introduced, which has brought much complexity in both training and inference period. In contrast, most of attention has been paid on how to improve the accuracy of neural networks, instead of the controlling units. With the proposed self-distillation, most of the shallow classifiers are able to achieve a much higher accuracy than the baseline models. So even with the simple threshold-based dynamic inference method, we still achieve significant acceleration with no accuracy loss.

3 Methodology

Fig. 1 shows the details of the proposed self-distillation on a ResNet18 model. Compared with the original model, self-distillation does not change the architecture of backbone layers yet adds several early-exit branches after the intermediate layers of neural networks. Each early-exit branch is composed of an attention module and a shallow classifier. In the training period, all the classifiers are trained by the proposed self-distillation, which regards the deeper classifiers as teacher models and the shallower classifiers as student models. In the inference period, all the additional attention modules and shallow classifiers are dropped so there are no additional parameters or computation penalty for the deployed model. The left part of this section gives a detailed introduction to these contents.

3.1 Formulation

Let $c : \mathbb{R}^N \mapsto \mathbb{R}^M$ be a given backbone classifier such as ResNets, VGG or MobileNets, we introduce several shallow classifiers by using the intermediate information in the

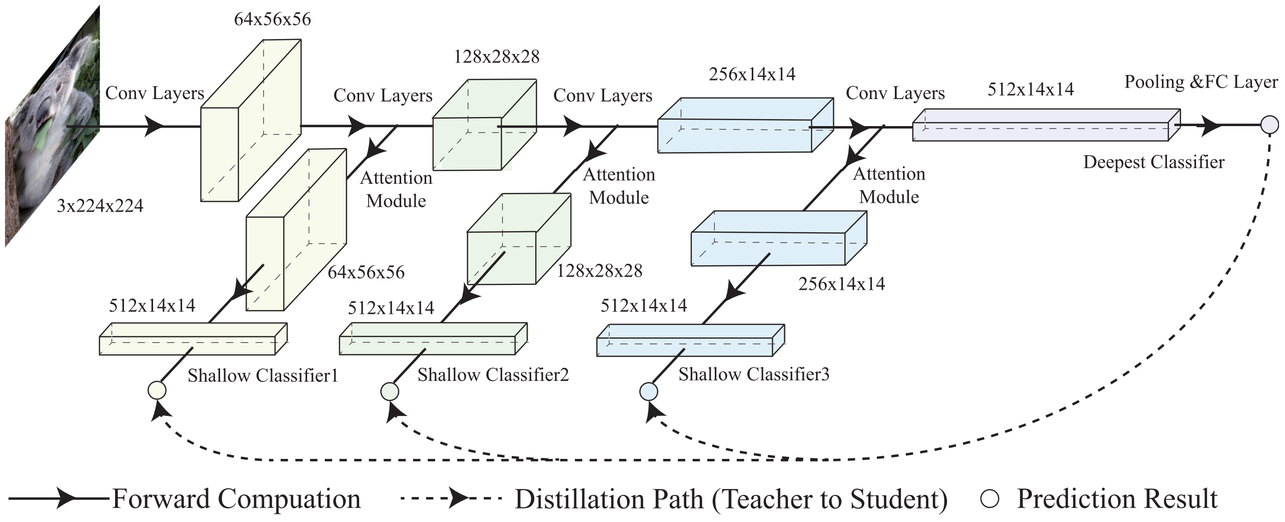


Fig. 1. The architecture of ResNet18 trained by self-distillation. (i) The whole neural networks can be divided into three sections: backbone, attention modules and shallow classifiers. (ii) The backbone section is just identical to the original model. (iii) Additional attention modules are attached after the intermediate features of the backbone. (iv) Features refined by attention modules will be fed into the shallow classifiers, which consists of a bottleneck layer and a fully connected layer. (v) All the attention modules and shallow classifiers are dropped in the inference period, indicating that there is no additional parameters and computation penalty in self-distillation.

backbone classifier F. More specifically, assume $F = g \circ f$ where g is the final classier and f is feature extractor operator and $f = f_K \circ f_{K-1} \circ \cdots \circ f_1$ where K denotes the number of stages in f. At each feature extraction stage i, we attach a classifier g_i for early prediction. Thus, we have K classifiers in total which has the form

$$\begin{aligned} c_1(x) &= g_1 \circ f_1(x), \\ c_2(x) &= g_2 \circ f_2 \circ f_1(x), \\ &\dots, \\ c_K(x) &= g_K \circ f_K \circ f_{K-1} \circ \cdots \circ f_1(x). \end{aligned} \tag{1}$$

Define $h_i = f_i \circ f_{i-1} \circ \cdots \circ f_1$, we have $c_i = g_i \circ h_i$ for all $i = 1, \dots, K$ and $c_K = c$ if g_K is chosen as $g_K = g$. Here, we call $g_1, \dots, g_{K-1}$ as shallow classifiers, and g_K as the final classifier. Each shallow classifier contains two components: feature alignment layer and softmax layer. The feature alignment layer is to guarantee that the feature dimension in the shallow layer is equal to the feature dimension of last layer, and the softmax layer is to smooth the label distribution with temperature T. In other words, for each $i = 1, 2, \dots, K-1$, g_i is represented as $g_i = q_T \circ F_i$ where F_i is the feature alignment layer and q_T is defined as

$$q_T(x) = \frac{\exp(x/T)}{\sum \exp(x/T)}. \tag{2}$$

The detailed architecture of F_i is given in Section 3.3.

Given n training samples $\mathcal{X} = \{(x_j, y_j)\}_{j=1}^n$ where $x_j \in \mathbb{R}^N$ is the input image and $y_j \in \mathbb{R}^M$ is the corresponding label. Define $c_i^j = c_i(x_j)$ to be the predicted label of sample x_j by the ith classifier and $F_i^j = F_i \circ h_i(x_j)$ be the feature vector of the jth sample given by ith classifier, we construct the distillation loss of jth sample

$$L_{\text{dis}}^j = \frac{1}{K} \sum_{i=1}^K \Big((1-\alpha_i) L_{\text{CE}}(c_i^j, y_j) + \alpha_i L_{\text{KL}}(c_i^j, c_{\text{re},i}^j) \Big), \tag{3}$$

where L_{CE} is the cross entropy loss, L_{KL} is the Kullback-Leibler (KL) divergence, $\alpha_i \in [0,1], \forall i = 1, \dots, K$ are the imitation parameters and $c_{\text{re},i}^j$ is the reference label of the jth sample for ith classifer. Besides, motivated by the hint loss in FitNet [35], we add a penalty for the shallow features

$$L_{\text{pen}}^j = \frac{1}{K} \sum_{i=1}^K \lambda_i L_2(F_i^j, F_{\text{re},i}^j), \tag{4}$$

where L_2 is the squared ℓ_2-norm loss, $\lambda_i > 0, i = 1, \dots, K$ are trade-off parameters and $F_{\text{re},i}^j$ is reference feature of jth sample for ith classifier. It is noted that the final classifier g_K is only trained by the L_{CE} loss, i.e., $\alpha_K = \lambda_K = 0$. In summary, the total loss of the self-distillation is

$$L = \frac{1}{n} \sum_{j=1}^n L_{\text{dist}}^j + \lambda L_{\text{pen}}^j. \tag{5}$$

Since there are several classifiers in the self-distillation framework, there are multiple choices of the reference labels c_{re} and the reference features F_{re}. as shown in Fig. 2.

1) *Deepest Teacher Distillation.* The deepest classifier is used as the reference labels and the reference features for all shallow classifiers, i.e., $c_{\text{re},i} = c_K$ and $F_{\text{re},i} = F_K \circ h_K$ for all $i = 1, 2, \dots, K-1$.
2) *Ensemble Teacher Distillation.* An ensemble of labels and features produced by all classifiers are used as the reference labels and the reference features, i.e., $c_{\text{re},i} = \mathcal{S}(\{c_i\})$ and $F_{\text{re},i} = \mathcal{S}(\{F_i \circ h_i\})$ for all $i = 1, 2, \dots, K-1$, where S denotes some ensemble operation.
3) *Transitive Teacher Distillation.* This distillation chooses the reference label and the reference feature transitively, i.e., $c_{\text{re},i} = c_{i+1}$ and $F_{\text{re},i} = F_{i+1} \circ h_{i+1}$ for all $i = 1, 2, \dots, K-1$. This distillation approach is similar to the distillation by teacher assistant [74].

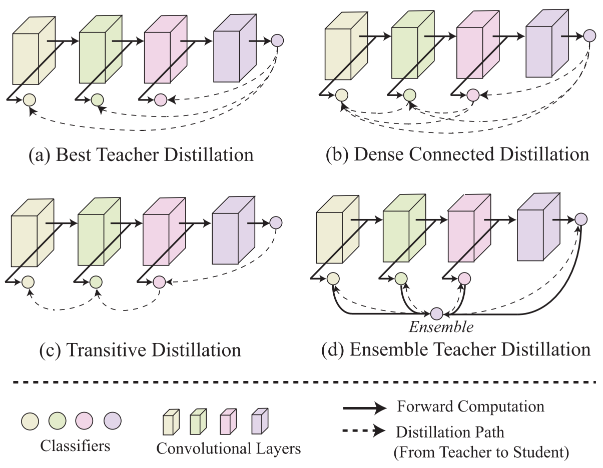

(a) Best Teacher Distillation
(b) Dense Connected Distillation
(c) Transitive Distillation
Ensemble
(d) Ensemble Teacher Distillation
Classifiers
Convolutional Layers
Forward Computation
Distillation Path (From Teacher to Student)

Fig. 2. Four kinds of distillation path in self-distillation. Each distillation path indicates a scheme on how to choose the teacher and student classifiers. A→B indicates classifier A is the teacher of classifier B.

(4) *Dense Teacher Distillation.* The dense distillation connects all the label and feature information among all classifiers, i.e., $c_{\mathrm{re},i} = \{c_k | k \neq i\}$ and $F_{\mathrm{re},i} = \{F_k \circ h_k | k \neq i\}$ for all $i = 1, 2, \ldots, K-1$. Thus, the KL loss in (3) is

$$L_{\mathrm{KL}}(c_i^j, c_{\mathrm{re},i}^j) = \sum_{k=i+1}^{K} L_{\mathrm{KL}}(c_i^j, c_k^j). \tag{6}$$

In this paper, we choose the deepest teacher distillation throughout all experiments and the comparison study of four distillation methods is presented in Table 7.

Remark 1. Compared with the conventional distillation methods where the teacher and student are two or several individual neural networks, the proposed self-distillation method simultaneously trains all classifiers $c_i, i = 1, 2, \ldots, K$, and enables us to perform knowledge distillation within one neural network.

3.2 Shallow Classifiers

As mentioned earlier, each shallow classifier g_i is the composition of the softmax layer q_T and the feature alignment layer F_i. Since the shallow classifiers only use the intermediate information of the backbone network at different depths, the feature alignment layer F_i first extracts the useful intermediate features by the attention module, followed by an alignment net. The alignment net is to adjust the feature size so that the squared ℓ_2-norm loss between shallow features and reference feature can be used for improving the accuracy of shallow classifiers.

The proposed attention module is proposed in Fig. 3a which is composed of a depthwise-pointwise layer and bilinear interpolation. The features from the shallow convolution layers are fed into the attention module to obtain the attention mask. Then, a dot product is performed between the attention mask and the input features.

The architecture of align nets is shown in Fig. 3b. The depthwise and pointwise convolution layers proposed in MobileNet are utilized to replace the vanilla convolution layers to reduce the training overhead. Each depthwise-pointwise convolutional layer performs two times downsample on feature maps.

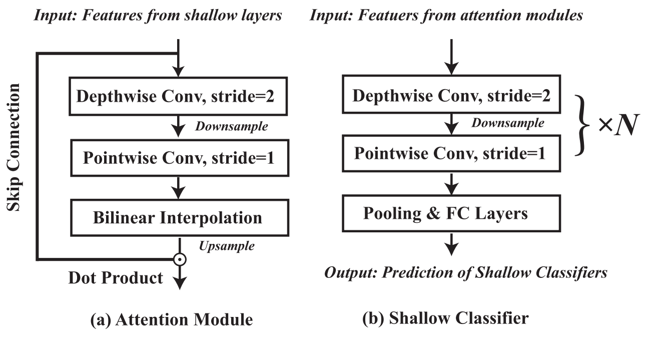


Fig. 3. The architecture of attention modules and shallow classifiers. (i) The attention module consists of a convolution layer for downsampling and a bilinear interpolation layer for upsampling. The attention mask learned by these two layers is utilized to enhance the original features by a dot product operation. (ii) The shallow classifier is composed of several pairs of depthwise and pointwise layers which are designed to downsample the feature with few parameters and computation. N in the figure is decided by the depth of the shallow classifiers.

The number of depthwise-pointwise convolutional layers for the shallow classifiers is set as N_i. Taking $K = 4$ as an example, $N_i = 4 - i, i = 1, 2, 3$ for the three shallow classifiers.

3.3 Dynamic Inference

It is generally acknowledged that the prediction of neural networks with a higher confidence (softmax value) is more likely to be right. In this paper, we exploit this observation to determine whether a classifier gives a right or wrong prediction. As described in Algorithm 1, we set different thresholds for shallow classifiers. If the maximal output of softmax in the shallow classifier is larger than the corresponding threshold, its results will be adopted as the final prediction so much computation can be reduced. Otherwise, the neural networks will employ a deeper classifier to predict. If there is no shallow classifier which can provide confident prediction, the ensemble of all the classifiers is regarded as the final result. Since all the classifiers in self-distillation share the same backbone layers, there is not much extra computation introduced in the dynamic inference. Moreover, experiments show that most of images can be classified by the shallowest classifier correctly, which reduces the computation cost significantly.

Moreover, the threshold-based dynamic inference provides more flexibility in the real-world application. It permits models to dynamically adjust its trade-offs between response time and accuracy on the fly by adjusting the thresholds of shallow classifiers.

Algorithm 1. Scalable Inference

```
Input: Images $x$, Thresholds $\{\sigma_i\}^N$, Classifiers $\{c_i\}^N$
Output: Prediction Result $\hat{y}$
1: for i from 1 to $N$ do
2:     $\hat{y}_i = c_i(x)$
3:     if max($\hat{y}$) $> \sigma_i$ then
4:         return $\hat{y}_i$
5: return ensemble($y_1, \ldots, y_N$)
```

Since the value of thresholds determines which classifier is utilized as the final prediction, it has a direct influence on the accuracy and acceleration results of dynamic inference. A lower threshold for shallow classifiers results in the fact that most samples will be predicted by shallow classifiers, indicating more rapid response yet lower accuracy. Similarly, a higher threshold leads to a phenomenon that most samples will be determined by deeper classifiers, indicating precise prediction yet longer response time. In the proposed self-distillation, we have applied genetic algorithm to find the best thresholds as shown in Algorithm 2. The value of thresholds in different shallow classifiers are encoded into binary codes as the genes. The accuracy and acceleration results are utilized to measure the fitness of the threshold.

$$fitness = acceleration\ ratio + \beta \cdot (accuracy - baseline), \tag{7}$$

where β is a hyper-parameter to balance the impact of these two elements. Adjustment of β leads to trade-offs between accuracy and acceleration.

Algorithm 2. Threshold Searching

```
Input: Images $x$, Classifiers $\{c_i\}^N$, Max Generations $g$
Output: Optimal Thresholds $\Sigma = \{\sigma_i\}^N$
1: Randomly Initialize $G$ as the set of genes
2: for i from 1 to $g$ do
3:     Get fitness of $G$ according to Equation (7).
4:     Sample genes with high fitness from $G$ as $\hat{G}$
5:     $G$ := crossover and mutate $\hat{G}$
6: $\Sigma$ := decode $G$ from binary codes to numbers
7: return $\Sigma$
```

3.4 Rationality of Self Distillation

In this subsection, we interpret our model from the perspective of the generalized distillation [75], which unifies knowledge distillation and privileged information. In generalized distillation, each training sample x_j associates with additional information x_j^* provided by “intelligent teachers”. In the context of traditional distillation framework, the additional information is given by a teacher network.

Given a function $g \in \mathcal{G}$, define $|\mathcal{G}|_C$ to be an appropriate function capacity measure, e.g., VC-dimension and $R(g)$ to be the generalization error. Let $f \in \mathcal{F}$ be the target function, the generalized distillation framework [75] assumes the student network $f_s \in \mathcal{F}_s$ learn the oracle classifier at a slow rate and the teacher network $f_t \in \mathcal{F}_t$ learns it at fast rate, i.e.,

$$R(f_s) - R(f) \leq O(|\mathcal{F}_s|_C/\sqrt{N}) + \epsilon_s, \tag{8}$$

$$R(f_t) - R(f) \leq O(|\mathcal{F}_s|_C/N) + \epsilon_t, \tag{9}$$

where ϵ_s and ϵ_t denote the approximation error of the function class $\mathcal{F}_s$ and $\mathcal{F}_t$ to f, respectively. Furthermore, it assumes that the student learns from the the teacher at a medium rate, i.e.

$$R(f_s) - R(f_t) \leq O(|\mathcal{F}_s|_C/N^\alpha) + \epsilon_l, \tag{10}$$

where $\alpha \in (1/2, 1]$ and ϵ_l is the approximation error of the function class $\mathcal{F}_s$ to f_t. Combining (9) with (10), we arrive at an estimation error of the student network given by

$$R(f_s) - R(f) \leq O\left(\frac{|\mathcal{F}_s|_C + |\mathcal{F}_t|_C}{N^\alpha}\right) + \epsilon_l + \epsilon_t. \tag{11}$$

Therefore, we have to analyze the if the inequality

$$O\left(\frac{|\mathcal{F}_s|_C + |\mathcal{F}_t|_C}{N^\alpha}\right) + \epsilon_l + \epsilon_t \leq O\left(\frac{|\mathcal{F}_s|_C}{\sqrt{N}}\right) + \epsilon_s, \tag{12}$$

holds. Consider $K = 2$ in the proposed self-distillation framework, i.e., there are one shallow classifier c_1 and a final classifier c_2. When the capacity of classifier c_2 is much larger than c_1, we can treat $c_1 = f_s$ and $c_2 = f_t$, which implies the (12). This is also the main assumption in classical knowledge distillation [29]. When the capacity of the shallow classifier c_1 is large enough to approximate the target function, we can treat $f_s = c_2$ and $f_t = c_1$. In this case, $\epsilon_l = 0$ and $\epsilon_t \approx \epsilon_s$ but $|\mathcal{F}_s|_C \gg |\mathcal{F}_t|_C$, which also implies (12).Thus, the features provided by the shallow classifier are the so-called privileged information. In this sense, instead of the teacher-student structure, the proposed self-distillation network is more likely to be the “student-student” which may accelerate the knowledge communications.

4 Experiment

4.1 Experiment Settings

On the image classification task, the proposed self-distillation is evaluated in two benchmark datasets: CIFAR100 [76] and ImageNet (ILSVRC2012) [77] and seven kinds of neural networks: ResNet [3], WideResNet [78], ResNeXt [4], SENet [79], ResNeSt [80] MobileNetV2 [22] and ShuffleNet [24]. On the point cloud classification task, we conduct experiments on ModelNet10 and ModelNet40 [81] on residual graph convolutional neural networks [82].

SGD with learning rate decay and momentum is utilized to optimize the neural networks. On CIFAR100 dataset, neural networks are trained by 300 epochs, with learning rate divided by 10 at the 90-th, 180-th and 270-th epoch. Batchsize is 128 and the initial learning rate is 0.1. On ImageNet dataset and ModelNet, neural networks are trained by 100 epochs, with learning rate divided by 10 at the 30-th, 60-th, and 90-th epoch. Batchsize is 256 and the initial learning rate is 0.1. AutoAugment [83] is utilized across all the experiments on RGB images.

Since the images in CIFAR100 dataset are much smaller than images in ImageNet, the architecture of neural networks on CIFAR100 is slightly modified by removing the first pooling layer and reducing the stride and kernel size of convolutional layers. The reported ImageNet (ILSVRC2012) accuracy is evaluated on the validation set. All the experiments are conducted by PyTorch1.3.0 on GPU devices. The recommended value for α, λ is 0.6 and 4×10^{-2}. Moreover, in the last 5 epoches of the training period, both α and λ are set to 0 to facilitate model convergence.

4.2 Results on CIFAR100

Table 1 shows the experiment results of self-distillation in CIFAR100 dataset. It is observed that (i) On average, the proposed self-distillation leads to 3.47 percent accuracy boost. (ii) In 5 of 14 neural networks, the shallowest classifier, which has much fewer parameters and computation, achieves higher accuracy than their baseline models. (iii) In all the neural networks, the second shallowest classifier achieves higher accuracy than their baseline models. (iv)

TABLE 1
Experiment Results of Accuracy (%) on CIFAR100

Models	Baseline	Classifier1	Classifier2	Classifier3	Classifier4	Ensemble
ResNet18	79.01	76.31	79.32	81.24	81.76	82.64
ResNet50	80.88	82.60	83.55	84.42	83.66	85.42
ResNet101	82.37	81.64	82.73	84.12	84.03	85.48
ResNet152	82.92	81.25	82.94	84.37	84.52	85.41
WRN50-2	81.26	82.85	84.02	84.91	84.33	85.78
WRN101-2	82.37	82.56	83.79	84.87	84.33	86.03
SENet18	79.53	75.60	79.81	81.77	81.84	83.10
SENet50	81.01	81.80	82.93	83.91	83.51	85.21
SENet101	82.75	82.20	82.69	83.17	82.97	84.82
ResNeXt50-4	82.65	82.03	83.50	83.78	83.42	85.12
ResNeXt101-8	82.96	82.84	83.70	84.70	84.31	85.81
ResNeSt50	83.12	83.09	84.01	84.98	85.19	86.40
MobileNetV2	65.49	62.93	66.03	67.95	67.17	68.87
ShuffleNetV2	71.61	73.20	73.87	75.66	/	76.45

"Baseine" in the table indicates a model trained without knowledge distillation. "Ensemble" indicates the prediction ensemble of "Ensemble" indicates the prediction ensemble of all the classifiers.

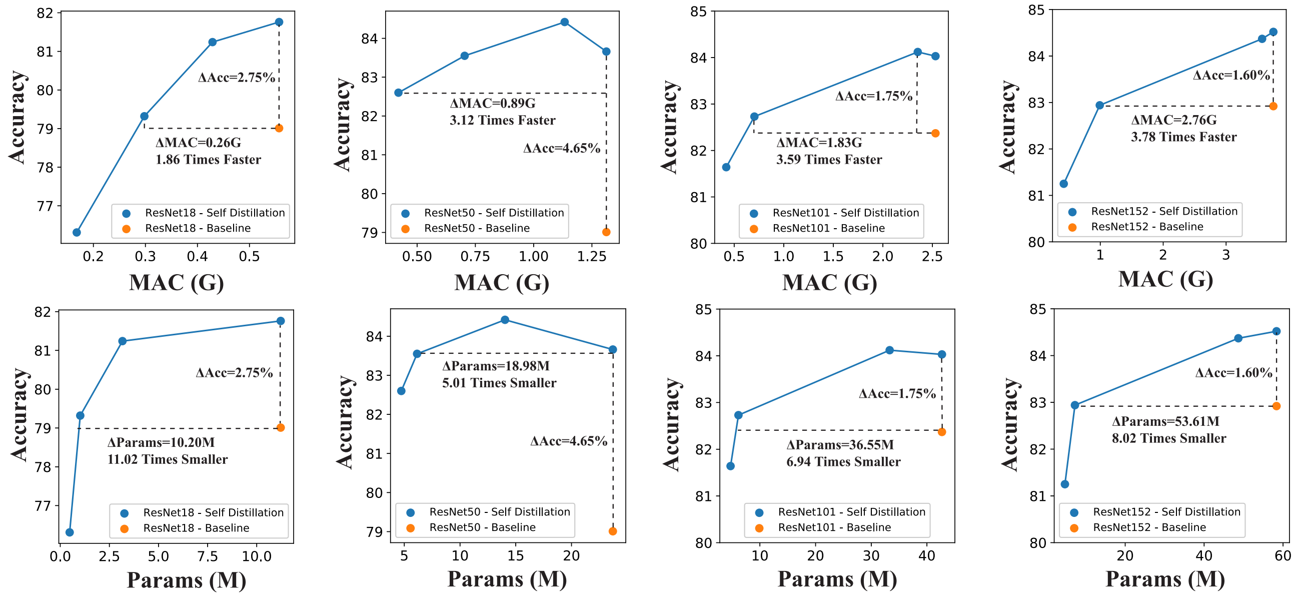


Fig. 4. Experiment results of self-distillation on CIFAR100. MAC indicates the multiply–accumulate operation of neural networks.

There is 3.38 and 4.84 percent accuracy improvements on MobileNetV2 and ShuffleNetV2, indicating that the proposed self-distillation also works on lightweight neural networks.

Fig. 4 shows the comparison on computation, parameters and accuracy of four ResNet models on CIFAR100. The blue dots indicate the three shallow classifiers and the deepest classifier in neural networks and the orange dot indicates the baseline model. It is observed that significant accuracy improvements and acceleration can be achieved simultaneously by replacing the baseline model with shallow classifiers.

4.3 Results on ImageNet

Table 2 shows the experiment results on ImageNet and Fig. 5 shows the parameters, computation and accuracy of ResNet models. It is observed that (i) On average, 2.84

TABLE 2
Experiment Results of Accuracy (%) on ImageNet

Models	Baseline	Classifier1	Classifier2	Classifier3	Classifier4	Ensemble
ResNet18	69.21	55.03	60.94	64.70	70.51	70.63
ResNet50	76.30	71.72	74.58	77.45	77.89	78.28
ResNet101	77.03	71.75	74.39	79.47	79.70	78.87
ResNet152	77.62	71.50	75.36	80.22	80.32	80.56
ResNeXt50-4	77.29	71.95	75.76	79.02	79.96	80.32
WideResNet50	77.46	72.37	75.99	79.22	79.87	80.17

"Baseine" in the table indicates a model trained without knowledge distillation. "Ensemble" indicates the prediction ensemble of all the classifiers.

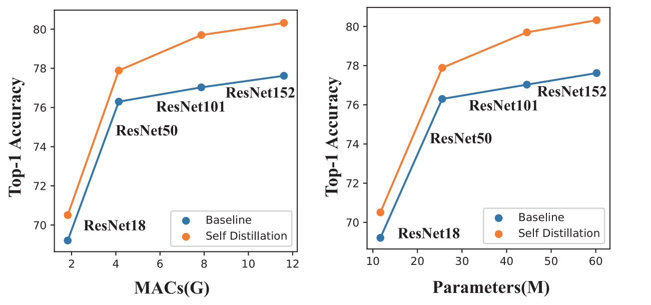


Fig. 5. Experiment results of self-distillation on ImageNet. MAC indicates the multiply–accumulate operation of neural networks.

percent accuracy boost can be observed in neural networks trained by self-distillation on ImageNet, ranging from 2.47 percent on ResNet18 as the minimum to 4.24 percent on ResNet50 as the maximum. (ii) More benefits can be found in neural networks with more layers. (iii) Significant acceleration and compression can be achieved by replacing the deep resnet with shallow resnet trained by self-distillation.

4.4 Results on Point Cloud Classification

Table 4 shows the experiment results on two point cloud datasets - ModelNet10 and ModelNet40. It is observed that (i) On ModelNet10, there is 0.98 and 1.36 percent average accuracy improvements on the 3_{rd} classifier and the ensemble classifier, respectively. (ii) On ModelNet40, there is 0.53 and 1.01 percent average accuracy improvements on the 3_{rd} classifier and the ensemble classifier, respectively. These results demonstrate the effectiveness of self-distillation on point cloud data and graph convolution neural networks.

4.5 Comparison With the Other Distillation Methods

The comparison between the proposed self-distillation and the other three kinds of knowledge distillation methods on CIFAR100 is shown in Table 3. It is observed that (i) Self-distillation achieves the highest accuracy compared with the other knowledge distillation methods. On average, 1.61 percent accuracy boost can be found in the comparison between self-distillation and the second best knowledge distillation method (Feat). (ii) Self-distillation and the other knowledge distillation methods can be utilized together to attain more improvements on accuracy. On average, 3.80, 3.76, 3.74 percent accuracy boost can be achieved by combine KD, AT and DML with self-distillation.

TABLE 3
Comparison With Other Knowledge Distillation Methods on CIFAR100

Teacher Model Student Model	ResNet50 ResNet18	ResNet101 ResNet18	ResNet101 ResNe50
Teacher Accuracy	80.88	82.37	82.37
Student Accuracy	79.01	79.01	80.88
KD [29]	80.49	80.31	82.09
FitNet [35]	80.67	80.54	82.14
AT [36]	80.43	80.39	81.92
DML [84]	80.52	80.57	82.37
RKD [42]	80.69	80.67	82.29
SPKD [43]	80.57	80.45	82.16
Feat [85]	80.91	80.80	82.40
Ours	81.76	81.76	85.42
KD + Ours	82.23	82.17	85.92
AT + Ours	82.34	82.21	85.65
DML + Ours	82.09	82.14	85.91

The numbers are reported by the accuracy of the deepest classifier.

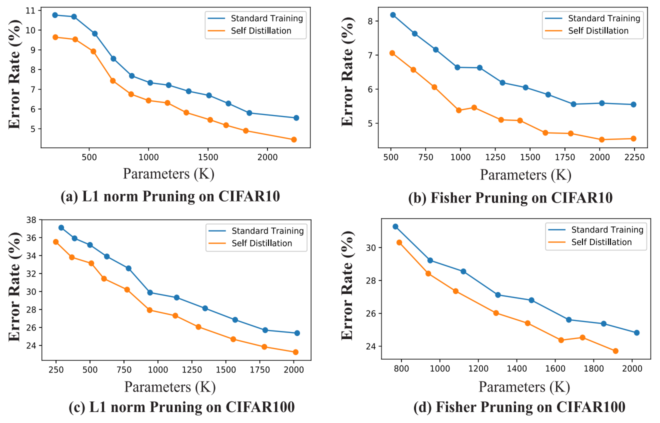


Fig. 6. Comparison experiments with two kinds of pruning methods in WideResNet40-2 on CIFAR10 and CIFAR100.

4.6 Neural Network Pruning With Self Distillation

The proposed self-distillation is orthogonal to other neural network compression methods such as pruning. Fig. 6 shows the pruning results on WideResNet40 trained by standard training methods and the proposed self-distillation on two kinds of pruning methods and two datasets. It is observed that (i) In all kinds of situations, the accuracy improvements from self-distillation can be kept after pruning. (ii) With the same parameters, the self distilled models have higher accuracy. With the same accuracy, the self distilled models have fewer parameters.

4.7 Results of Dynamic Inference

Experiment results of the proposed threshold-controlled dynamic inference method are shown in Tables 5 and 6. It is observed that the proposed dynamic inference leads to benefits on both model accuracy and acceleration. With different thresholds, the trade-offs between them can be adjusted flexibly.

Fig. 7 gives a comparison between the self-distillation and three related dynamic inference methods including ACT, SCAT [86], and BlockDrop [69]. Experiments are conducted on ImageNet with ResNet101 models. It is observed that the dynamic inference in the proposed self-distillation has outperformed other related methods by a large margin.

The thresholds of shallow classifiers have a great impact on the classification results. With a lower threshold, most images can be predicted by shallow classifiers, indicating more rapid responses yet lower accuracy. With a higher threshold, most images can be determined by deeper classifiers, indicating precise prediction yet longer response time. The relation among thresholds, classification accuracy, and the number of images classified are shown in Fig. 8. The X axis is the threshold of the shallowest classifier on ResNet18. The left Y axis indicates its classification accuracy while the right Y axis indicates how many images are classified by the

TABLE 4
Experiments Results of Accuracy (%) in Point Cloud Classification

Dataset	Models	Baseline	Classifier1	Classifier2	Classifier3	Ensemble
ModelNet10	ResGCN8	92.73	89.75	91.42	93.83	94.27
	ResGCN12	93.50	89.51	92.77	94.18	94.42
	ResGCN16	92.40	89.19	93.09	93.57	94.01
ModelNet40	ResGCN8	90.76	85.39	87.81	91.51	91.85
	ResGCN12	90.32	84.28	89.82	91.09	91.72
	ResGCN16	91.33	87.14	88.04	91.41	91.87

"Baseine" in the table indicates a model trained without knowledge distillation. "Ensemble" indicates the prediction ensemble of all the classifiers.

TABLE 5
Experiments Results of Dynamic Inference on CIFAR100 With Different Thresholds

ResNet18		ResNet50		ResNet101		ResNet152	
Accuracy	Acceleration	Accuracy	Acceleration	Accuracy	Acceleration	Accuracy	Acceleration
+3.10	1.4X	+4.04	1.8X	+2.48	+4.0X	+1.69	4.8X
+3.06	1.6X	+4.01	2.0X	+2.47	+4.2X	+1.78	5.4X
+3.06	1.8X	+4.10	2.2X	+2.37	+4.4X	+1.71	5.8X
+2.98	2.0X	+4.10	2.4X	+2.31	+4.6X	+1.82	6.2X
+2.87	2.2X	+3.92	2.6X	+2.05	+4.8X	+1.50	6.6X
+2.82	2.4X	+3.45	2.8X	+2.07	+5.0X	+1.16	7.4X
+2.17	2.6X	+2.68	3.0X	+1.93	+5.2X	+0.37	7.8X

Accuracy and acceleration in this table indicate the accuracy increment and acceleration ratio compared with baseline models.

TABLE 6
Experiments Results of Dynamic Inference on ImageNet With Different Thresholds

ResNet18		ResNet50		ResNet101		ResNet152	
Accuracy	Acceleration	Accuracy	Acceleration	Accuracy	Acceleration	Accuracy	Acceleration
+1.40	1.10X	+0.13	1.74X	+2.32	1.92X	+2.63	2.17X
+1.26	1.15X	+0.87	1.66X	+2.17	2.01X	+2.26	2.59X
+0.88	1.20X	+1.28	1.60X	+1.71	2.15X	+2.03	2.71X
+0.04	1.25X	+1.36	1.56X	+1.42	2.21X	+1.74	2.82X
/	/	+1.54	1.42X	+0.98	2.30X	+0.24	3.17X

Accuracy and acceleration in this table indicate the the accuracy increment and acceleration ratio compared with baseline models.

shallowest classifier. It is observed that (i) The higher the threshold is, the more classification accuracy is achieved, and the fewer images can be classified. (ii) Even with a very high threshold (0.99), there are more than 30 percent images can be classified. (iii) Even with a very low threshold (0.5), the shallowest classifier can achieve higher classification accuracy than the baseline model (79.01 percent).

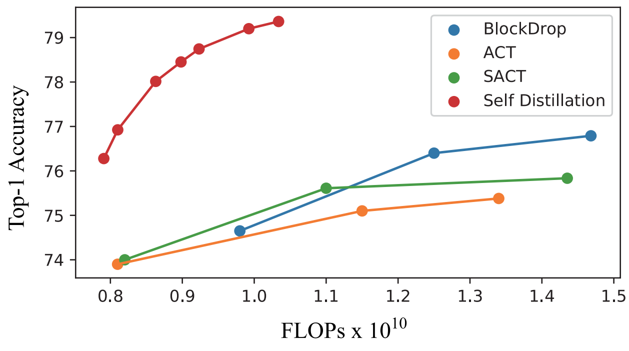


Fig. 7. Comparison between self-distillation with the other dynamic inference methods on ImageNet with ResNet101.

4.8 Influence From the Number of Shallow Classifiers

One of the problems in self-distillation is how many shallow classifiers should be added in self-distillation and how does the number of shallow classifiers influence the accuracy of

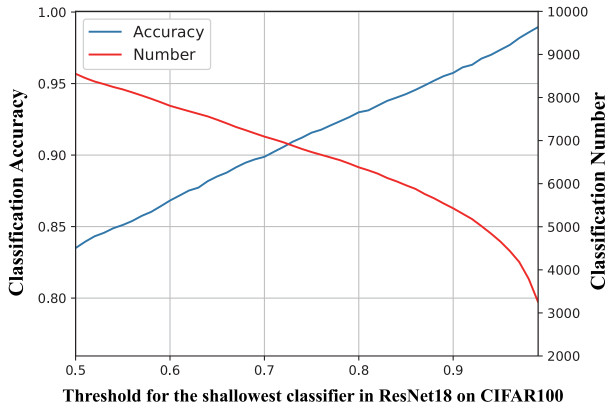


Fig. 8. The relation between the threshold and the number of images classified by this classifier on CIFAR100 with ResNet18. Note that there are 10K images in the testing set totally.

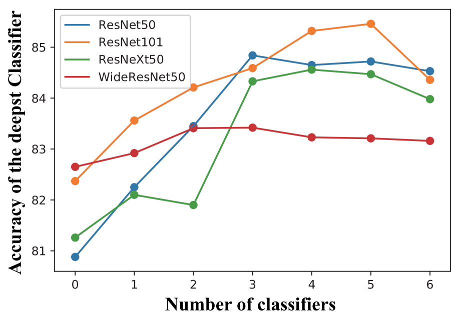


Fig. 9. Experiments on the relation between the accuracy of the deepest classifier and the number of classifiers.

self-distillation. Fig. 9 shows the experiment results on four kinds of neural networks on CIFAR100. The X axis indicates the number of shallow classifiers in self-distillation and Y axis indicates the accuracy of the deepest classifier. It is observed that (i) In all the situations, self-distillation outperforms the baseline models (x = 0) by a large margin. (ii) In all neural networks, self-distillation with four and five classifiers achieves best accuracy.

4.9 Adding Shallow Classifiers to Every Layer

It is generally acknowledged that convolutional neural networks gradually learn the representation of inputs layer by layer. However, self-distillation influences the learning procedure of hidden representation by directly using labels to guide the training of the hidden layers. The above experiments show that when there are a few shallow classifiers, the influence of self-distillation is positive. In this section, we study if self-distillation harms the performance of convolutional neural networks in the extreme situation - adding shallow classifiers after every residual block or every layer.

Taking ResNe18 as an example. A ResNet18 model has 8 residual blocks and each residual block consists of 2 convolutional layers. We add shallow classifiers after each residual block and each convolutional layer, respectively. Experiments show that: (i) When the shallow classifiers are attached after each residual block, self distilled ResNet18 achieves 81.04 percent accuracy in the deepest classifier, which is still 2.03 percent higher than the baseline, but 0.72 percent lower than self-distillation with three shallow classifiers. (ii) When the shallow classifiers are attached after each layer, self distilled ResNet18 achieves 77.54 percent accuracy in the deepest classifier, which is 1.47 percent lower than the baseline. This observation indicates that too many shallow classifiers may harm the performance of self-distillation, and even leads to negative impact. Besides, this conclusion also conforms to our observation in Section 4.8 - more shallow classifiers do not always lead to higher performance.

4.10 Influence From the Position of Shallow Classifiers

In this section, we study how the positions of shallow classifiers influence the performance of self-distillation. The following four kinds of schemes are considered.

(1) *Uniform Scheme.* The shallow classifiers are attached into different depths of a neural network uniformly. For example, on ResNet50 with 16 residual blocks, three shallow classifiers are attached at the 4th, 8th, 12th residual blocks.

(2) *Downsample Scheme.* The shallow classifiers are attached into the layer before downsampling convolutional layers in a neural network. For example, on ResNe50 with 16 residual blocks, three shallow classifiers are attached at the 3rd, 7th, 13th residual blocks.

(3) *Shallow Layers Scheme.* The shallow classifiers are attached at the shallow layers in a neural network. For example, on ResNet50 with 16 residual blocks, three shallow classifiers are attached at the 1st, 2nd, 3rd residual blocks.

(4) *Deep Layers Scheme.* The shallow classifiers are attached at the deep layers in a neural network. For example, on ResNet50 with 16 residual blocks, three shallow classifiers are attached at the 13th, 14th, 15th residual blocks.

Experiment results of the four kinds of schemes on CIFAR100 with ResNet18 and ResNet50 have been given in Table 9. Since the shallow classifiers in different schemes are attached at different depths, we only compare the accuracy of the deepest classifier. It is observed that (1) On average, the uniform scheme achieves the highest accuracy boost (2.79 percent) while the shallow layer scheme leads to the lowest accuracy increment (2.26 percent). (2) Even in the worst situation, self-distillation still outperforms all the other knowledge distillation methods in Table 3. This observation indicates that self-distillation is not sensitive to the position of shallow classifiers.

4.11 Influence From Different Distillation Path

The experiment results of these four kinds of distillation schemes are shown in Table 7. It is observed that (i) All of the four distillation paths have achieved significant accuracy boost than the baselines, especially on the shallow classifiers. (ii) There is no obvious difference among the four kinds of distillation schemes, indicating that the proposed

TABLE 7
Experiments of Different Distillation Path in Self-Distillation

Distillation Path	Classifier1	Classifier2	Classifier3	Classifier4	Ensemble
No Distillation	75.85	78.73	81.27	81.26	82.65
Best Teacher Distillation	76.22	79.81	81.93	81.29	82.32
Densely Connected Distillation	77.52	79.98	81.32	81.37	82.42
Transitive Distillation	76.31	79.32	81.24	81.76	82.64
Ensemble Teacher Distillation	76.50	78.90	82.01	81.86	83.33

TABLE 8
Accuracy of Different Attention Modules in ResNet18 on CIFAR100

Attention Method	Classifier1	Classifier2	Classifier3	Classifier4	Ensemble
No Attention	74.83	78.08	81.04	81.44	82.20
BAM [87]	75.24	78.82	81.07	81.49	82.24
CBAM [88]	74.95	78.91	81.13	81.59	82.17
SENet [79]	75.14	78.76	81.31	81.47	82.46
Our approach	76.31	79.32	81.24	81.76	82.64

self-distillation is not sensitive to the choice of teacher and student models.

4.12 Influence From Different Attention Modules

As shown in Fig. 3, the proposed attention module is composed of a depthwise-pointwise layer and bilinear interpolation. The features from the shallow convolution layers are fed into the attention module to obtain the attention mask. Then, a dot product is performed between the attention mask and the inputted features. Table 8 shows the comparison between the proposed self-distillation with other three kinds of attention modules, it is observed that the proposed attention outperforms other methods by a large margin, especially on the shallow classifiers.

4.13 Ablation Study

Compared with the standard training method, two distillation loss L_{KL} and L_2 are introduced in the proposed self-distillation. As shown in Table 10, an ablation study is conducted to demonstrate their effectiveness. It is observed that (i) Self-distillation with only one of the two losses still outperforms the baseline model by a large margin. (ii) Self-distillation achieves the highest accuracy when the two loss functions are utilized together.

4.14 Hyper-Parameters Sensitivity Study

The proposed self-distillation introduces two additional hyper-parameters α and λ to control the ratio of different losses. A sensitivity study on CIFAR100 with ResNet18 has been given in Fig. 10. It is observed that (i) When α ranges from 0.1 to 0.8, the accuracy of self distilled ResNet18 ranges from 82.48 to 82.80 percent. In the worst situation ($\alpha = 0.8$), self-distillation still leads to a 3.47 percent accuracy boost. (ii) When λ ranges from 0.01 to 0.08, the accuracy of self distilled ResNet18 ranges from 82.52 to 82.95 percent. In the worst situation ($\lambda = 0.02$), self-distillation still leads to 3.51 percent accuracy boost. These observations indicate that self-distillation is not sensitive to the choice of hyper-parameters.

TABLE 9
Experiment Results of Self-Distillation With Shallow Classifiers in Different Position Schemes on CIFAR100

Scheme	ResNet18	ResNet50
Baseline	79.01	80.88
Uniform Scheme	81.76	83.71
Downsample Layer Scheme	81.76	83.66
Shallow Layer Scheme	81.14	83.27
Deep Layer Scheme	81.47	83.67

TABLE 10
Ablation Study on the Loss Function With ResNet18 on CIFAR100

Loss	Baseline		Self Distillation	
L_{KL}	×	✓	×	✓
L_2	×	×	✓	✓
Accuracy	79.01	81.58	80.37	81.76

5 Discussion

5.1 Flat Minimum

It is universally acknowledged that although shallow neural networks (e.g., AlexNet) can also achieve almost zero loss on the training set, their performance on test set or in practical applications is far behind over-parameterized neural networks (e.g., ResNet). Keskar *et al.* proposed explanations that over-parameters models may converge easier to the flat minima, while shallow neural networks are more likely to be caught in the sharp minima, which is sensitive to the bias of data [89]. Fig. 11 gives an intuitive explanation of the difference between flat and sharp minima. The X axis represents the parameters of models in one dimension. The Y axis is the value of loss function. The two curves denote the

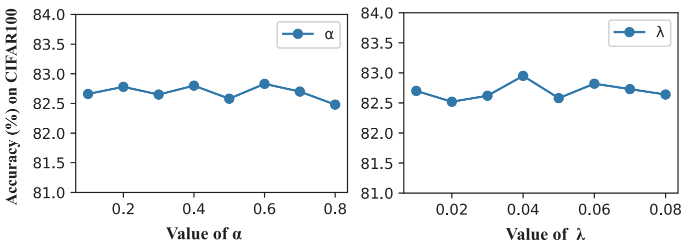


Fig. 10. Hyper-parameters sensitivity study of α and λ on CIFAR100 with ResNet18.

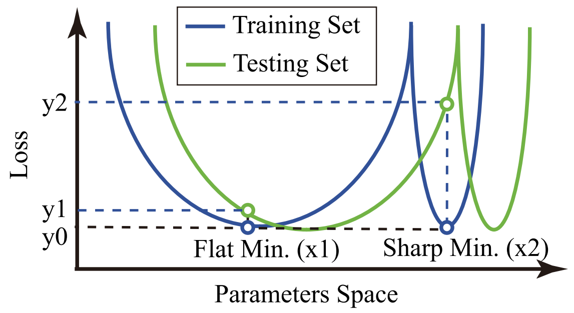


Fig. 11. An intuitive explanation of the difference between the flat and the sharp minimum [89]. y0 indicates the loss of two minima in the training set. y1 and y2 indicates the loss of the flat minimum and the sharp minimum in the testing set.

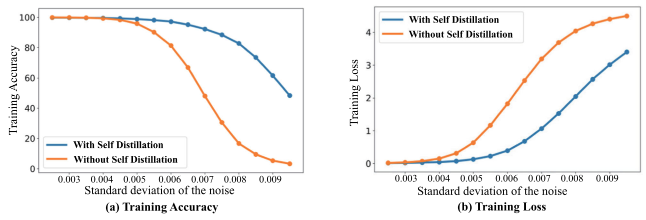


Fig. 12. Comparison of training accuracy and loss with increasing Gaussian noise: models trained with self-distillation are more tolerant to noise - flat minima.

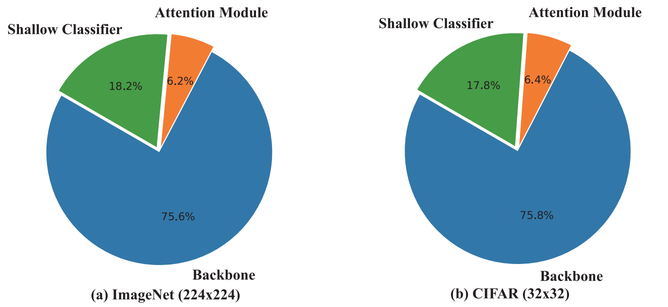


Fig. 13. The computation ratio of different layers in self-distillation on two datasets in the training period of ResNet50. Backbone in the figure indicates the original layers in neural networks while the shallow classifiers and attention modules are additional layers from self-distillation in the training period.

loss curves on the training set and test set. Both two minima (x1 for flat minima and x2 for sharp minima) can achieve extremely small loss on the training set (y0). Unfortunately, the training set and the test set are not independently and identically distributed. While in the test, x1 and x2 are still utilized to find the minima y1 and y2 in the testing curve, which causes severe bias in the sharp minimum curve (y2 - y0 is much larger than y1 - y0).

Inspired by the work of Zhang *et al.* [84], we conduct the following experiments to show that the proposed self-distillation framework can converge to a flat minimum. Two 18-layer ResNets have been trained on CIFAR100 dataset first, one with self-distillation and the other one not. Then Gaussian noise is added to the parameters of the two models and then their entropy loss and predicted accuracy on the training set are obtained and plotted in Fig. 12. As can be seen in Fig. 12a, the training set accuracy in the model trained with self-distillation maintains at a very high level with noise level, presented as standard deviation of the Gaussian noises, keeping increasing, while the training accuracy in the model without self-distillation drops severely, as shown in Fig. 12a. Same observations and conclusions can be obtained in Fig. 12b with training loss as the metric. Based on the above observations, we conclude that the models trained with self-distillation are flatter. According to the conclusion sourced from Fig. 11, the model trained with self-distillation is more robust to perturbation of parameters. Note that the 4/4 classifier is used in self-distillation ResNet for a fair comparison. To sum up, the model trained without self-distillation is much more sensitive to the Gaussian noise. These experiment results support our view that self-distillation helps models find flat minima, permitting better generalization performance.

5.2 Visualization of Attention Map

In the proposed self-distillation, attention modules are introduced to obtain classifier-specific features, leading to a significant performance gain on shallow classifiers. We further visualize the spatial attention maps as depicted in Fig. 14. The heat maps indicate learned attention maps where the value of each pixel is computed as the mean value of pixels in the same position of all channels.

As is depicted in Fig. 14, all the classifiers pay their attention to the same spatial position while ignoring the backgrounds, which indicates that all of the attention modules have learned to find the most informative pixels. The attention maps in shallow classifier 1 seem to concentrate on the details of shark's and cat's features such as their outlines. In contrast, the attention maps in shallow classifier 3 focus more on the texture features, which indicates deep classifiers that have a larger receptive filed are more likely to predict based on global and low frequency information while shallow classifiers incline to be dominated by local and high frequency information.

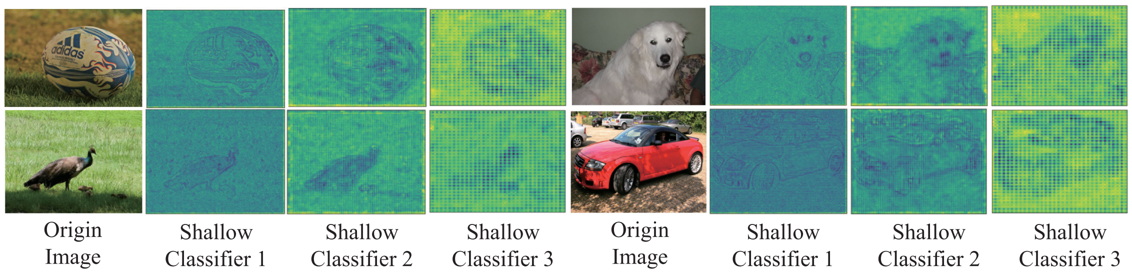


Fig. 14. Visualization of attention maps in shallow classifiers on ImageNet.

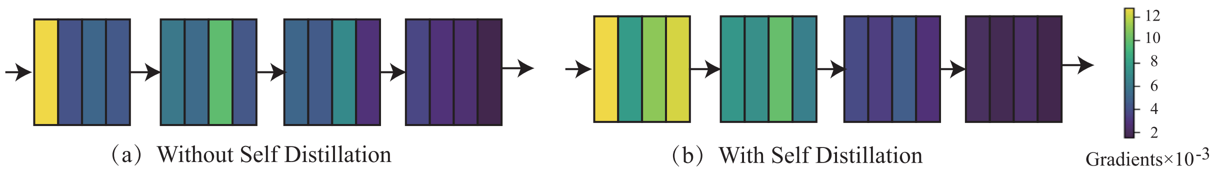


Fig. 15. The visualization of average absolute value of gradients in the training period of ResNet18 models on CIFAR100. The blocks in the figure indicate the convolutional layers.

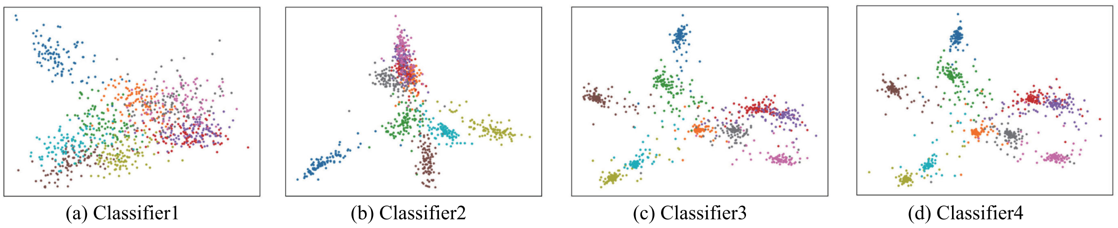


Fig. 16. PCA (principal component analysis) visualization of feature distribution in four classifiers.

5.3 Model Interpretation by Integrated Gradients

As shown in Fig. 17, the output of shallow classifiers is interpreted by the integrated gradients method [90]. The bright pixels in the figure indicate the pixels have a large impact on the prediction result of the image. It is observed that the shallow classifiers and the deepest classifier has similar interpretation results - the pixels containing objects have a high value while the pixels of background have a low value.

5.4 Visualization of Feature Map in Different Classifiers

More discriminating features are extracted with deeper classifiers in self-distillation. Since there are multiple classifiers existing in self-distillation, features of each classifier can be computed and analyzed to demonstrate their discriminating principles. As depicted in Fig. 16, experiments on WideResNet trained on CIFAR100 are conducted to compare features of different classifiers.

Fig. 16 visualizes the distances of features in different classifiers. To begin with, it is obvious that the deeper the classifier is, the more concentrated clusters are observed. In addition, the changes of the distances in shallow classifiers, as shown in Figs. 16a and 16b, are more severe than that in deep classifiers, as demonstrated in Figs. 16c and 16d.

Table 11 further summarizes the sort separability for each classifier. SSE stands for the sum of squares due to error, and SSB is short for the sum of squares between groups. The smaller the SSE is, the denser the clusters are. Also, the clusters become more discriminating with SSB growing. Here we use SSE/SSB to evaluate the distinct capability of the models. The smaller it is, the more clear the classifier is. It can be seen in Table 11 that the SSE/SSB decreases as the classifier goes deeper. In summary, the more discriminating feature maps in the classifier, the higher accuracy the model achieves.

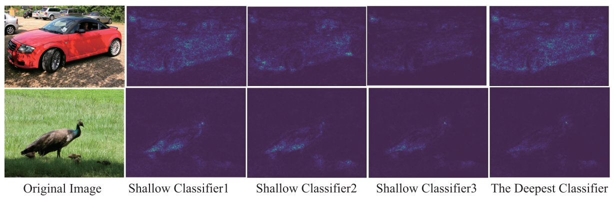


Fig. 17. The interpretation map of classifiers by the intergrated gradients [90]. The brightness of each pixel indicates its influence on the prediction results of the classifier.

TABLE 11
Measurement of Sort Separability and Accuracy (%) for Each Classifier on WideResNet28-12

Classifier	SSE*	SSB**	SSE/SSB	Accuracy
Classifier1/4	20.85	1.08	19.21	76.21
Classifier2/4	8.69	1.15	7.54	80.86
Classifier3/4	11.42	1.87	6.08	81.58
Classifier4/4	11.74	2.05	5.73	81.59

* *SSE: Sum of squares due to error.*
* *SSB: Sum of squares between groups.*

5.5 Vanishing Gradients Problems

Self-distillation can prevent models from vanishing gradient problem. It is generally acknowledged that deep neural networks are hard to train due to the problem of vanishing gradients. In self-distillation, the supervision on the neural networks is injected into different depths. DSN [91] has proved that the multi-exit neural networks can alleviate the vanishing gradients problem mathematically. In this paper, we conduct the following experiments to support this view. Two 18-layer ResNets are trained, one of them is equipped with self-distillation and the other is not. We compute the mean magnitude of gradients in each convolutional layer as shown in Fig. 15. It is observed that the magnitude of gradients of the model with self-distillation (Fig. 15a) is larger than the one without self-distillation (Fig. 15b), especially in the first and second ResBlocks.

5.6 Analysis of Training Overhead

Compared with the traditional two-state knowledge distillation methods where a teacher model and a student model should be trained successively, the proposed self-distillation is a one-stage knowledge distillation method where the teacher and student models are trained together. The additional training overhead of self-distillation is caused by the additional layers - the attention modules and the shallow classifiers. Fig. 13 shows the computation ratio of attention modules, the shallow classifiers and the backbone in the training period. It is observed that on both ImageNet and CIFAR, the additional layers only takes $\frac{1}{4}$ of the total computation, indicating self-distillation doesn't bring much training overhead compared with the traditional training methods.

6 Conclusion

In this paper, we have proposed a novel distillation method named self-distillation, which leads to benefits on model accuracy, model acceleration and model compression simultaneously. In the training period, the original neural network is modified to a multi-exit neural network by introducing additional shallow classifiers at different depths. Knowledge distillation is performed on these classifiers to transfer the knowledge from the deep layers to the shallow layers. Based on the multi-exit neural network in self-distillation, the threshold-controlled dynamic inference can be utilized to achieve higher acceleration. Moreover, we have proved that self-distillation can be utilized with other model compression and acceleration methods such as neural network pruning, lightweight models, and even the other knowledge distillation methods.

There are mainly two insights provided in the proposed self-distillation. First, the knowledge transferring inside one neural network is very promising. Different from conventional knowledge distillation which transfers knowledge among different models, the proposed self-distillation has proved that the knowledge transfer inside one model can be utilized to improve the performance of neural networks.

Second, the success of dynamic inference in this paper shows that the bottleneck of dynamic inference is how to train a high performance early-exit in the neural network, instead of how to control different exiting paths. Previous works on dynamic inference have paid most of attention on how to train a complex controlling unit to decide the computation path of neural networks while ignoring how to improve the performance of early-exit path. In this paper, since self-distillation has significantly improved the accuracy of shallow classifiers, even with the simple threshold method, self-distillation can outperform other dynamic inference methods by a large margin.

ACKNOWLEDGMENTS

This work was supported in part by the IIISCT (Institute for interdisciplinary Information Core Technology), National Natural Sciences Foundation of China under Grants 31970972 and 11901338) , and Tsinghua University Initiative Scientific Research Program.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Very deep convolutional networks for large-scale image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit*, 2016, pp. 770–778.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[4] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5987–5995.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 779–788.

[6] W. Liu *et al.*, "SSD: Single shot MultiBox detector," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 21–37.

[7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 91–99.

[8] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 318–327, Feb. 2020.

[9] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2980–2988.

[10] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assisted Intervention*, 2015, pp. 234–241.

[11] L. Yu, X. Yang, H. Chen, J. Qin, and P.-A. Heng, "Volumetric ConvNets with mixed residual connections for automated prostate segmentation from 3D MR images," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 66–72.

[12] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," in *Proc. 4th Int. Conf. Learn. Representations* 2016, pp. 1–14.

[13] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 1135–1143.

[14] T. Zhang *et al.*, "A systematic DNN weight pruning framework using alternating direction method of multipliers," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 191–207.

[15] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," in *Proc. 7th Int. Conf. Learn. Representations*, 2019, pp. 1–21.

[16] S. Ye *et al.*, "Adversarial robustness vs. model compression, or both?" in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 111–120.

[17] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training deep neural networks with binary weights during propagations," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 3123–3131.

[18] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-net: ImageNet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 525–542.

[19] M. Nagel, M. V. Baalen, T. Blankevoort, and M. Welling, "Data-free quantization through weight equalization and bias correction," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 1325–1334.

[20] K. Lee, K. Lee, J. Shin, and H. Lee, "Overcoming catastrophic forgetting with unlabeled data in the wild," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 312–321.

[21] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, pp. 1–9, 2017.

[22] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetv2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4510–4520.

[23] Z. Qin *et al.*, "ThunderNet: Towards real-time generic object detection on mobile devices," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 6717–6726.

[24] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical guidelines for efficient CNN architecture design," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 116–131.

[25] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5 MB model size," 2016, *arXiv:1602.07360*.

[26] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 1269–1277.

[27] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," 2014, *arXiv:1405.3866*.

[28] C. Buciluă, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2006, pp. 535–541.

[29] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 1–9.

[30] L. Zhang, J. Song, A. Gao, J. Chen, C. Bao, and K. Ma, "Be your own teacher: Improve the performance of convolutional neural networks via self distillation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 3713–3722.

[31] L. Zhang, Z. Tan, J. Song, J. Chen, C. Bao, and K. Ma, "SCAN: A scalable neural networks framework towards compact and efficient models," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2019, pp. 4029–4038.

[32] S. I. Mirzadeh, M. Farajtabar, A. Li, N. Levine, A. Matsukawa, and H. Ghasemzadeh, "Improved knowledge distillation via teacher assistant," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 5191–5198.

[33] J. H. Cho and B. Hariharan, "On the efficacy of knowledge distillation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 4793–4801.

[34] M. Kang, J. Mun, and B. Han, "Towards oracle knowledge distillation with neural architecture search," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 4404–4411.

[35] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "FitNets: Hints for thin deep nets," in *Proc. 3rd Int. Conf. Learn. Representations* 2015, pp. 1–13.

[36] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," in *Proc. 5th Int. Conf. Learn. Representations* 2017, pp. 1–13.

[37] J. Yim, D. Joo, J. Bae, and J. Kim, “A gift from knowledge distillation: Fast optimization, network minimization and transfer learning,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4133–4141.

[38] S. Lee and B. C. Song, “Graph-based knowledge distillation by multi-head attention network,” in *Proc. 30th Brit. Mach. Vis. Conf.*, 2019, pp. 141–154.

[39] X. Wang, R. Zhang, Y. Sun, and J. Qi, “KDGAN: Knowledge distillation with generative adversarial networks,” in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 775–786.

[40] P. Liu, W. Liu, H. Ma, Z. Jiang, and M. Seok, “KTAN: Knowledge transfer adversarial network,” in *Proc. Int. Joint Conf. Neural Netw.*, 2020, pp. 1–7.

[41] B. Heo, M. Lee, S. Yun, and J. Y. Choi, “Knowledge distillation with adversarial samples supporting decision boundary,” in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, pp. 3771–3778.

[42] W. Park, D. Kim, Y. Lu, and M. Cho, “Relational knowledge distillation,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3967–3976.

[43] F. Tung and G. Mori, “Similarity-preserving knowledge distillation,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 1365–1374.

[44] X. Jin *et al.*, “Knowledge distillation via route constrained optimization,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 1345–1354.

[45] Y. Jang, H. Lee, S. J. Hwang, and J. Shin, “Learning what and where to transfer,” in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 3030–3039.

[46] Y. Liu, X. Jia, M. Tan, R. Vemulapalli, Y. Zhu, B. Green, and X. Wang, “Search to distill: Pearls are everywhere but not the eyes,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 7539–7548.

[47] T. Furlanello, Z. C. Lipton, M. Tschannen, L. Itti, and A. Anandkumar, “Born again neural networks,” in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 1607–1616.

[48] H. Bagherinezhad, M. Horton, M. Rastegari, and A. Farhadi, “Label refinery: Improving imagenet classification through label progression,” 2018, *arXiv: 1805.02641*.

[49] Y. Liu, K. Chen, C. Liu, Z. Qin, Z. Luo, and J. Wang, “Structured knowledge distillation for semantic segmentation,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2604–2613.

[50] S. Gupta, J. Hoffman, and J. Malik, “Cross modal distillation for supervision transfer,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2827–2836.

[51] S. Ge, S. Zhao, C. Li, and J. Li, “Low-resolution face recognition in the wild via selective knowledge distillation,” *IEEE Trans. Image Process.*, vol. 28, no. 4, pp. 2051–2062, Apr. 2019.

[52] Z. Ke, D. Wang, Q. Yan, J. Ren, and R. W. Lau, “Dual student: Breaking the limits of the teacher in semi-supervised learning,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 6727–6735.

[53] D. Lopez-Paz, L. Bottou, B. Schölkopf, and V. Vapnik, “Unifying distillation and privileged information,” in *Proc. 4th Int. Conf. Learn. Representations*, 2016, pp. 1–10.

[54] V. Vapnik and R. Izmailov, “Learning using privileged information: Similarity control and knowledge transfer,” *J. Mach. Learn. Res.*, vol. 16, no. 2023/2049, 2015, Art. no. 2.

[55] Y. Tian, “Over-parameterization as a catalyst for better generalization of deep ReLU network,“ *CoRR*, vol. abs/1909.13458, pp. 1–23, 2019.

[56] Y. LeCun, J. S. Denker, and S. A. Solla, “Optimal brain damage,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 1990, pp. 598–605.

[57] B. Hassibi and D. G. Stork, “Second order derivatives for network pruning: Optimal brain surgeon,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 1993, pp. 164–171.

[58] C. Louizos, M. Welling, and D. P. Kingma, “Learning sparse neural networks through L_0 regularization,” in *Proc. 6th Int. Conf. Learn. Representations*, 2018, pp. 1–13.

[59] Z. Liu *et al.*, “MetaPruning: Meta learning for automatic neural network channel pruning,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 3295–3304.

[60] J. Frankle and M. Carbin, “The lottery ticket hypothesis: Finding sparse, trainable neural networks,” in *Proc. 7th Int. Conf. Learn. Representations*, 2019, 1–42.

[61] F. Li, B. Zhang, and B. Liu, “Ternary weight networks,” 2016, *arXiv:1605.04711*.

[62] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, “Incremental network quantization: Towards lossless cnns with low-precision weights,” in *Proc. 5th Int. Conf. Learn. Representations* 2017, pp. 1–14.

[63] M. Nagel, M. V. Baalen, T. Blankevoort, and M. Welling, “Data-free quantization through weight equalization and bias correction,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 1325–1334.

[64] A. Howard *et al.*, “Searching for MobileNetV3,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 1314–1324.

[65] 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. Eur. Conf. Comput. Vis.*, 2018, pp. 784–800.

[66] Y. Kaya, S. Hong, and T. Dumitras, “Shallow-deep networks: Understanding and mitigating network overthinking,” in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 3301–3310.

[67] X. Chen, H. Dai, Y. Li, X. Gao, and L. Song, “Learning to stop while learning to predict,” in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 1520–1530.

[68] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. E. Gonzalez, “SkipNet: Learning dynamic routing in convolutional networks,” in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 409–424.

[69] Z. Wu *et al.*, “BlockDrop: Dynamic inference paths in residual networks,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8817–8826.

[70] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Q. Weinberger, “Multi-scale dense networks for resource efficient image classification,” in *Proc. 6th Int. Conf. Learn. Representations*, 2018, pp. 1–14.

[71] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4700–4708.

[72] J. Yu, L. Yang, N. Xu, J. Yang, and T. S. Huang, “Slimmable neural networks,” in *Proc. 7th Int. Conf. Learn. Representations*, 2019, pp. 1–12.

[73] J. Yu and T. S. Huang, “Universally slimmable networks and improved training techniques,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 1803–1811.

[74] S. I. Mirzadeh, M. Farajtabar, A. Li, N. Levine, A. Matsukawa, and H. Ghasemzadeh, “Improved knowledge distillation via teacher assistant,” in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 5191–5198.

[75] D. Lopez-Paz, L. Bottou, B. Schölkopf, and V. Vapnik, “Unifying distillation and privileged information,” in *Proc. 4th Int. Conf. Learn. Representations*, 2016, pp. 1–10.

[76] A. Krizhevsky *et al.*, *Learning Multiple Layers of Features From Tiny Images*, Princeton, NJ, USA: Citeseer, 2009, pp. 1–60.

[77] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.

[78] S. Zagoruyko and N. Komodakis “Wide residual networks,” in *Proc. British Mach. Vis. Conf.*, 2016, pp. 87.1–87.12.

[79] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7132–7141.

[80] H. Zhang *et al.*, “Resnest: Split-attention networks,” 2020, *arXiv:2004.08955*.

[81] Z. Wu *et al.*, “3D ShapeNets: A deep representation for volumetric shapes,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1912–1920.

[82] G. Li, M. Muller, A. Thabet, and B. Ghanem, “DeepGCNs: Can GCNs go as deep as CNNs?,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 9267–9276.

[83] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, “Autoaugment: Learning augmentation policies from data,” 2018, *arXiv:1805.09501*.

[84] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu, “Deep mutual learning,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4320–4328.

[85] B. Heo, J. Kim, S. Yun, H. Park, N. Kwak, and J. Y. Choi, “A comprehensive overhaul of feature distillation,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 1921–1930.

[86] M. Figurnov *et al.*, “Spatially adaptive computation time for residual networks,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1039–1048.

[87] J. Park, S. Woo, J.-Y. Lee, and I. S. Kweon, “BAM: Bottleneck attention module,” in *Proc. British Mach. Vis. Conf.*, 2018, pp. 147–161.

[88] S. Woo, J. Park, J.-Y. Lee, and I. So Kweon, “CBAM: Convolutional block attention module,” in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 3–19.

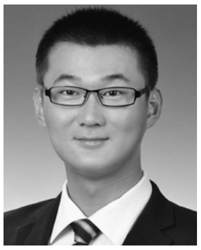
[89] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. Peter Tang, “On large-batch training for deep learning: Generalization gap and sharp minima,” in *Proc. 5th Int. Conf. Learn. Representations* 2017, pp. 1–16.

[90] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 3319–3328.

[91] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *Proc. 18th Int. Conf. Artif. Intell. Statist.*, 2015, pp. 562–570.

Linfeng Zhang is currently working toward the PhD degree with the Institute for Interdisciplinary Information Sciences (IIIS), Tsinghua University, China, supervised by Kaisheng Ma. He publishes papers on conferences including NeurIPS, ICCV, CVPR, ICLR etc. His research interests includes computer vision, deep neural network compression, and acceleration.

Chenglong Bao (Member, IEEE) received the PhD degree from the Department of Mathematics, National University of Singapore, Singapore, in 2014. He is currently an assistant professor at Yau Mathematical Sciences Center, Tsinghua University, China and Yanqi Lake Beijing Institute of Mathematical Sciences and Applications, China. His main research interests include mathematical image processing, large scale optimization, and its applications.



KaiSheng Ma received the PhD degree in computer science and engineering from the Pennsylvania State University, State College, Pennsylvania, following professor Vijaykrishnan Nanrayanan at Penn State, Yuan Xie at UCSB, Jack Sampson at Penn State. He is currently a tenure-track assistant professor of computer science in Institute for Interdisciplinary Information Sciences (IIIS), Tsinghua University, China. He leads the Algorithms, Architecture and Chipset Lab (ArChip). His research interests include the interdisciplinary fields of human-machine interface (HMI), brain-spired AI algorithm design, computer vision for self-driving, compact model design, computer architecture and microelectronics. He believes in vertical integration for systems, with a special emphasis on implanted medical and neural devices, and chip level system solutions for self-driving systems. His previous research on Nonvolatile Processor and Energy Harvesting won 2018 EDAA Best Dissertation Award. He publishes papers on conferences including NeurIPS, ICCV, AAAI, CVPR, ISCA, ASPLOS, MICRO, HPCA, DAC etc. He has won many awards, including 2015 HPCA Best Paper Award, 2016 IEEE MICRO Top Picks, 2017 ASP-DAC Best Paper Award, and 2018 EDAA Best Dissertation Award. He has many honors, including 2016 Penn State CSE Department Best Graduate Research, 2016 Cover Feature of NSF ASSIST Engineering Research Center, and 2011 Yang Fuqing and Wang Yangyuan Academician Scholarship.