Learning Student Networks via Feature Embedding

Hanting Chen, Yunhe Wang, Chang Xu, Chao Xu and Dacheng Tao, *Fellow, IEEE*

Abstract—Deep convolutional neural networks have been widely used in numerous applications, but their demanding storage and computational resource requirements prevent their applications on mobile devices. Knowledge distillation aims to optimize a portable student network by taking the knowledge from a well-trained heavy teacher network. Traditional teacher-student based methods used to rely on additional fully-connected layers to bridge intermediate layers of teacher and student networks, which brings in a large number of auxiliary parameters. In contrast, this paper aims to propagate information from teacher to student without introducing new variables which need to be optimized. We regard the teacher-student paradigm from a new perspective of feature embedding. By introducing the locality preserving loss, the student network is encouraged to generate the low-dimensional features which could inherit intrinsic properties of their corresponding high-dimensional features from teacher network. The resulting portable network thus can naturally maintain the performance as that of the teacher network. Theoretical analysis is provided to justify the lower computation complexity of the proposed method. Experiments on benchmark datasets and well-trained networks suggest that the proposed algorithm is superior to state-of-the-art teacher-student learning methods in terms of computational and storage complexity.

Index Terms—deep learning, teacher-student learning, knowledge distillation.

I. INTRODUCTION

DEEP neural networks (DNNs) have provided state-of-the-art performance in various fields such as image classification [1], [2], semantic modeling [3], [4], visual quality evaluation [5], object detection [6], [7], and segmentation [8]. However, a neural network with considerable parameters requires heavy computation for both training and test, which is difficult to use on edge devices such as mobile phones and smart cameras. For example, a VGGNet [1] consisting of 16 convolutional layers has more than $500MB$ parameters and it requires more than $10^{10} \times$ floating number multiplications, which cannot be tolerated by portable devices. Therefore, how to compress and accelerate existing CNNs has become a research hotspot.

Recently, a variety of CNN compression methods have been proposed to tackle the aforementioned issues such as quantization [9], [10], weight and feature approximation [11],

Hanting Chen and Chao Xu are with the Key Laboratory of Machine Perception (Ministry of Education) and Coopertative Medianet Innovation Center, School of EECS, Peking University, Beijing 100871, P.R. China. E-mail:htchen@pku.edu.cn, xuchao@cis.pku.edu.cn

Hangting Chen and Yunhe Wang are with the Noah's Ark Laboratory, Huawei Technologies Co., Ltd, HuaWei Building, No.3 Xinxi Road, Shang-Di Information Industri Base, Hai-Dian District, Beijing 100085, P.R. China. E-mail:htchen@pku.edu.cn, yunhe.wang@huawei.com

Chang Xu and Dacheng Tao are with the UBTech Sydney Artificial Intelligence Centre and the School of Computer Science in the Faculty of Engineering and Information Technologies at The University of Sydney, J12 Cleveland St, Darlington NSW 2008, Australia. E-mail: c.xu@sydney.edu.au, dacheng.tao@sydney.edu.au.

encoding [12], approximation [13], and pruning [14], [15]. Wherein, weight pruning based methods achieve the highest compression performance since there are considerable subtle weights in most of pre-trained CNNs. In specific, Han *et.al.* [16] showed that over 70% subtle weights in AlexNet [2] can be removed without affecting its original top-5 accuracy. Wang *et.al.* [14] further pointed out that the redundancy can exist in both large and small weights and considerable redundancy also exists in modern CNNs such as ResNet [17].

Although pruning based methods can provide very high compression and speed-up ratios, compressed CNNs by exploiting these approaches cannot be directly used in mainstream platforms (*e.g.* Tensorflow and Caffe) and hardwares (*e.g.* NVIDIA GPU cards) since they require special architectures and implementation tricks (*e.g.* sparse convolution and Huffman encoding). As deeper networks often have higher performance than that of shallow networks [18], the teacher-student learning paradigm [19], [20], [21], [22], [23], [24], [25], [26] has emerged to learn portable networks (student network) of deeper architecture yet fewer parameters and convolution filters from the original network (teacher network). Compared with other approaches, portable networks generated by the teacher-student paradigm are much more flexible since they are exactly regular neural networks which do not need any additional supports for implementing online inference.

However, besides directly making outputs of teacher and student networks similar, most of existing methods cannot directly inherit teacher information in other layers to the student network. Since the student network has a thinner architecture than its teacher's, the feature dimensionality (the number of filters) in the student network is much less than that of its teacher. Therefore, a lot of works [21], [22], [23], [24], [25], [26] proposed to use an intermediate fully-connected layer to connect hint and guided layers to approximately inherit the teacher information. In addition, the fully-connected layer brings in a large number of auxiliary parameters, which have larger space and computational complexities and cannot be applied on large-scale CNNs in practice. Taking Student 4 in Table I as an example, the memory usage for storing the fully-connected layer is about $135MB$, which is much larger than those of the student network ($9MB$) and the teacher network ($35MB$).

Given high-dimensional features from teacher network and low-dimensional features from student networks, it is natural to regard the information propagation between these two networks as a feature embedding task. Therefore, we propose a manifold learning based method for learning portable CNNs. In summary, the proposed approach makes the following contributions:

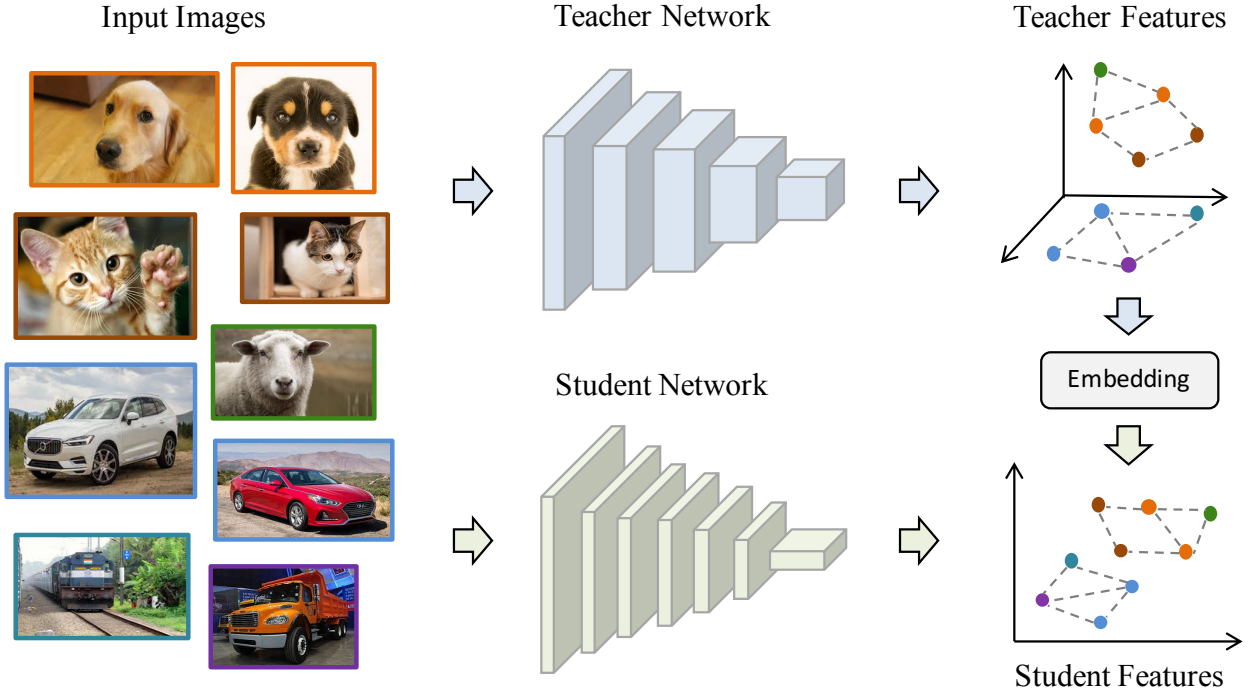- We propose to bridge teacher and student networks via a

Fig. 1: The diagram of the proposed method for learning portable deep neural networks. The top line is the teacher network and the bottom line is the student network. By applying the proposed method with a locality preserving loss, the student network embeds features in a low-dimensional space and maintains the relationship between samples in the high-dimensional space.

feature embedding task, so that student networks with fewer filters can generate low-dimensional features to preserve relationships between examples.

- We introduce the locality preserving loss into the teacher-student learning paradigm and provide theoretical analysis to justify the lower computation complexity of the proposed algorithm.

- Experiments on benchmarks demonstrate that the proposed method can efficiently learn portable networks with state-of-the-art performance.

This paper is organized as follows. Section II investigates related works on network compression algorithms. Section III proposes the student network learning method by feature embedding. Section IV analyzes the computational and space complexity of the proposed method. Section V shows the experimental results of the proposed method on several benchmark datasets and Section VI concludes this paper.

II. RELATED WORKS

Since CNNs require heavy computation and storage, it is difficult to adapt a neural network to real-world applications directly. Recently, various related works have been proposed to reduce the complexity of CNNs. Compression methods can be grouped into network trimming, layer decomposition and knowledge distillation based on their techniques.

A. Network Trimming

Network Trimming aims to remove redundant neurons in CNNs to accelerate and compress the original network.

Gong *et.al.* [27] suggested vector quantization to represent similar connections using a cluster center. Denton *et.al.* [11] exploited the singular value decomposition approach and decomposed the weight matrices of fully connect layers. Considering that 32-bit floating numbers are overfined for parameters of CNNs, Courbariaux *et.al.* [28] and Rastegari *et.al.* [29] explored binarized neural networks, whose weights are -1/1 or -1/0/1. Han *et.al.* [15] utilized pruning, quantization and Huffman coding to achieve a higher compression ratio. In addition, Wang *et.al.* [14] introduced the discrete cosine transform (DCT) bases and converted convolution filters into the frequency domain, thereby producing a much higher compression ratio and speed improvement. Subsequently, Wang *et.al.* [30] compressed feature map of CNN in the frequency domain and accelerated the calculation of convolution directly. Sun *et.al.* [31] introduced least absolute shrinkage and selection operator to design a selection method for efficient networks. Wang *et.al.* [32] proposed a novel pruning algorithm with better generalization and pruning efficiency by utilizing Group Lasso. Huang and Yu [33] compressed the weight matrices during training by reshaping them into a high-dimensional tensor with a low-rank approximation.

Although aforementioned algorithms achieve satisfactory results in CNN compression, architectures of networks compressed by these methods would be significantly different from the that of ordinary networks, which means that special implementations are required for high-speed inference and development costs are increased.

B. Layer Decomposition

Traditional layers in DNNs (*e.g.* convolutional layers, fully-connect layers) often result in huge computational cost. Therefore, a number of works aim to design lightweight layers to obtain efficient networks. Jin *et.al.* [34] presented fully factorized convolutions to accelerate the feedforward of deep networks. Wang *et.al.* [35] factorized the convolutional layer by considering spatial convolution. SqueezeNet [36] achieved an accuracy similar with AlexNet yet with $50\times$ fewer parameters by utilizing a bottleneck architecture. Pang *et.al.* [37] proposed sparse shallow MLP to construct a deep network with few parameters and high accuracy. Since the convolution calculation in CNNs is time-consuming, a various of algorithms focused on redesigning the convolutional layers. MobileNets [38] introduced depth-wise separable convolutions that largely reduced the computation cost of convolutional layers. ShuffleNet [39] combined pointwise group convolution and channel shuffle to decrease the computational complexity while maintaining accuracy. Wu *et.al.* [40] presented a parameter-free "shift" operation to alternate vanilla convolutions. Moreover, Sandler *et.al.* [41] improved MobileNets by introducing an inverted residual structure, which consists of thin and linear bottleneck layers. Ma *et.al.* [42] proposed a new metric beyond FLOPs to evaluate the speed of CNNs, which leads to ShuffleNet V2. Unlike the methods to design a low-cost convolutional layer, Wang *et.al.* [43] reused the filters in CNNs by exploiting versatile filters and achieved a better performance.

C. Knowledge distillation

Different from directly compressing the heavy networks, some works investigate the intrinsic information of original networks to learn smaller networks. Knowledge Transfer, first pioneered by Hinton *et.al.* [19], aims to improve the training of a student network by borrowing knowledge from another powerful teacher network, while the student network has fewer parameters. It uses a softened version of the final output of a teacher network called softened target to instruct the student network. Besides the outputs, features of intermediate layers in teacher networks also contain useful information which can guide the learning of student networks. Therefore, Romero *et.al.* [21] minimized the difference between features of a hint layer in the student network and a guide layer in its teacher network, which enables the student network to receive sufficient information from teacher network. McClure and Kriegeskorte [20] proposed the pairwise distance of samples as a useful knowledge to transfer, which increases the robustness of student network. Motivated by ensemble learning methods, You *et.al.* [22] simultaneously utilized multiple teacher networks to learn a better student network. Moreover, several algorithms have been developed to investigate the restriction between teacher and student. Zagoruyk and Komodakis [24] exploited attention mechanism and proposed to transfer attention maps that are the summaries of full activations. Huang and Wang [25] treat the knowledge transfer as a distribution matching problem and used the Maximum Mean Discrepancy (MMD) metric to minimize the difference between features

from teacher and student networks. Wang *et.al.* [26] exploited generative adversarial network to make feature distribution of teacher and student networks similar.

Compared to network trimming approaches, teacher-student learning paradigms can be applied to mainstream hardware without special requirements, and can be combined with layer decomposition methods easily. Existing knowledge distillation algorithms can learn efficient networks under the guidance of the teacher networks. However, these methods usually exploited a fully-connect layer to bridge the gap between the high-dimensional features from teacher network and low-dimensional features from student network, which introduced a large number of additional parameters. For example, a $19GB$ fully-connect layer is required to connect $7 \times 7 \times 2048$ dimensional intermediate features in the teacher network (*e.g.* ResNet-101) and $7 \times 7 \times 1024$ dimensional intermediate features in the student network (*e.g.* Inception-BN). Given these huge space and computational complexities, a flexible and effective algorithm to transfer knowledge between features of the teacher and student is urgently required.

III. STUDENT NETWORK EMBEDDING

This section first reviews related works on the teacher-student learning paradigm, and then presents the student network embedding approach by introducing a locality preserving loss.

A. Teacher-Student Interactions

To learning student networks with portable architectures, Hinton *et.al.* [19] first proposed the Knowledge Distillation (KD) approach, which utilizes a softened output of a teacher network to transform information to a smaller network. McClure and Kriegeskorte [20] further proposed to minimize the pairwise distance of samples after employing the student network and the teacher network.

Let $\mathcal{N}_T$ and $\mathcal{N}_S$ denote the original pre-trained convolutional neural network (teacher network) and the desired portable network (student network), respectively. The goal of knowledge distillation is utilizing $\mathcal{N}_T$ to enhance the performance of $\mathcal{N}_S$. Denote the softmax output of the teacher network $\mathcal{N}_T$ as $P_T = \mathrm{softmax}(a_T)$, where a_T denotes activations input into the softmax layer. Similarly, $P_S = \mathrm{softmax}(a_S)$ and a_S are softmax output and activations of the student network $\mathcal{N}_S$, respectively. Hinton *et.al.* [19] introduced the soften softmax output $\tau(P_T)$ and $\tau(P_S)$, which can be calculated as:

$$\tau(P_T) = \mathrm{softmax}(\frac{a_T}{\tau}), \quad \tau(P_S) = \mathrm{softmax}(\frac{a_S}{\tau}). \quad (1)$$

Comparing with original one-hot like output, the soften output can transfer more information since it contains relationship between different classes. By matching the soften outputs of $\mathcal{N}_T$ and $\mathcal{N}_S$, student networks could inherits useful information from the teacher network. The student network is then learned using the following loss function:

$$\mathcal{L}_{KD}(\mathcal{N}_S) = \mathcal{H}(\mathrm{y}, P_S) + \lambda \mathcal{H}(\tau(P_T), \tau(P_S)), \quad (2)$$

where $\mathcal{H}$ is the cross-entropy loss, y is the ground-truth label and λ is a Trade-off parameter. The first term denotes the classical classification objective

However, since architectures of teacher and student networks are significantly different, the constraint on the final output layer cannot be easily achieved. In addition, given the fact that features in the hidden layer also contain useful information, Romero *et.al.* [21] presented a more flexible FitNet approach by introducing an intermediate hidden layer to connect teacher and student networks, which achieves higher performance than that of KD method. FitNet is trained in a two-stage fashion following the student-teacher paradigm. Specifically, a fully-connected layer is first added after the guided layer in the student network. Let f_S and f_T denote the features generated by guided layer and hint layer in the teacher network. The loss function used in the first stage can be formulated as:

$$\mathcal{L}_{HT}(\mathcal{N}_S) = \frac{1}{2}\|r(f_S) - f_T\|^2, \tag{3}$$

where r is the fully connect layer added to match f_S and f_T. Then, the student network $\mathcal{N}_S$ is further tuned using knowledge distillation as illustrated in Eq. 2 in the second stage. Since the feature dimensionality of the intermediate hint layer is much higher than that of the softmax layer, FitNet can transfer more useful information thus yield a student network with higher performance.

In addition, lots of works have been proposed to further enhance the accuracy of the student network by introducing different assumptions. For example, Yim *et.al.* [23] introduced the FSP (Flow of Solution Procedure) matrix to transfer the relationship between convolutional layers. You *et.al.* [22] simultaneously utilized multiple teacher networks for learning a more accurate student network. Zagoruyko *et.al.* [24] transferred useful information from the teacher network using the attention maps. Huang *et.al.* [25] minimized the Maximum Mean Discrepancy (MMD) loss between feature maps from teacher and student networks. Wang *et.al.* [26] utilized the generative adversarial network to make feature distributions of both teacher and student networks similar. However, there are still two important issues to be addressed: 1) Eq 3 independently considers each individual data point without investigating their connections; 2) the introduced fully-connected layer r increases the cost for training the student network.

B. Locality Preserving Loss

As mentioned above, the feature dimensionality of the student network is lower than that of the teacher network (*e.g.* from 6912 to 5120), thus we propose to regard the portable network learning as a low-dimensional embedding procedure, which aims to learn effective low-dimensional features. Considering that input images with similar content should lie on the neighbor area in both high-dimensional and low-dimensional spaces, we propose to exploit the manifold learning approach to address the teacher-student learning paradigm.

Lots of nonlinear manifold learning methods have been proposed for obtaining accurate low-dimensional representations.

Algorithm 1 Learning student network by exploiting the proposed locality preserving loss.

Input: A given teacher network $\mathcal{N}_T$ and its training set $\mathcal{X}$ with n instances, and the corresponding k-label set $\mathcal{Y}$, parameters: λ, γ, and τ.
1: Initialize the student network $\mathcal{N}_S$, whose number of parameters is significantly fewer than that in $\mathcal{N}_T$;
2: **repeat**
3:     Randomly select a batch $\{(\mathrm{x}^i, \mathrm{y}^i)\}_{i=1}^m$;
4:     Employ the teacher network on the mini-batch:
5:         $[\tau(P_T), P_T, f_T] \leftarrow \mathcal{N}_T(\mathrm{x})$;
6:     Employ the student network on the mini-batch:
7:         $[\tau(P_S), P_S, f_S] \leftarrow \mathcal{N}_S(\mathrm{x})$;
8:     Calculate the LP loss $\mathcal{L}_{LP} \leftarrow \frac{1}{m}\sum_{i,j}\alpha_{ij}\|f_S^i - f_S^j\|_2^2$;
9:     Calculate the loss function $\mathcal{L}_{Total}$ (Fcn.8);
10:     Update weights in $\mathcal{N}_S$ using gradient descent;
11: **until** convergence
Output: The student network $\mathcal{N}_S$.

For instance, locally linear embedding (LLE [44]) attempts to represent the manifold locally by reconstructing each input point as a weighted combination of its neighbors. Isomap [45] preserves geometric distances by returning an embedding where the distances between points is approximately equal to the shortest path distance, and laplacian eigenmaps (LE [46]) builds a graph incorporating neighborhood information of the dataset to compute a low-dimensional representation of the data set that optimally preserves local neighborhood information in a certain sense. However, these nonlinear methods are not applicable for large scale problems, due to their enormous computation and storage resource costs. In contrast, locality preserving projections (LPP [47]) is a linear alternative to those nonlinear methods and can be easily embedded into the learning of convolutional neural networks.

Specifically, given a labeled training set with n samples, $\{(\mathrm{x}^1, \mathrm{y}^1), (\mathrm{x}^2, \mathrm{y}^2), \cdots, (\mathrm{x}^n, \mathrm{y}^n)\}$, we denote features of x^i extracted by the teacher and student networks as f_T^i and f_S^i, respectively. Therefore, we propose to preserve the local relationship of the features generated by the student network like that of its teacher, which can be formulated as:

$$\min_{W_S} \sum_{i,j} \alpha_{ij}\|f_S^i - f_S^j\|_2^2, \tag{4}$$

where W_S is the parameters of the student network before the guided layer and $\alpha_{i,j}$ describes the local relationship between the features generated by the selected hint layer of the teacher network. Specifically, $\alpha_{i,j}$ is defined as follows:

$$\alpha_{i,j} = \begin{cases} \exp\left(-\dfrac{\|f_T^i - f_T^j\|_2^2}{\sigma^2}\right) & if \ j \in N(i), \\ 0 & otherwise, \end{cases} \tag{5}$$

where $N(i)$ denotes the k nearest neighbor of the feature f_T^i of the i-th image x^i generated by teacher network, and σ is a normalized constant.

We can obtain a student network $\mathcal{N}_S$ preserving relationship between samples from a high-dimensional space to the target

low-dimensional space by optimizing Eq. 4. However, to compute the k nearest neighbors, we need to take the entire training set in each iteration, which is inefficient. Therefore, we use the mini-batch strategy to learn the student network, and the k-nearest neighbors will only be discovered within the mini-batch. The locality preserving loss function is therefore reformulated as:

$$\mathcal{L}_{LP} = \frac{1}{2m} \sum_{i,j} \alpha_{ij} \|f_S^i - f_S^j\|_2^2, \qquad (6)$$

where m is the batch size of the student network.

In addition, the ground-truth label data is also used for helping the training process of the student network. The entire objective function of the proposed network is then formulated as:

$$\mathcal{L}_{Total} = \frac{1}{2m} \left[\sum_i \mathcal{H}(y^i, P_S^i) + \gamma \sum_{i,j} \alpha_{ij} \|f_S^i - f_S^j\|_2^2 \right], \quad (7)$$

where γ is the weight parameter for seeking the trade-off of two different terms and P_S^i is the output of the classifier in the student network for the i-th sample x^i.

The first term in Fcn. 7 minimizes the cross entropy loss of classifier outputs to maintain the performance of the student network, and the second term embeds samples from the high-dimensional space to the low-dimensional space in the portable student network $\mathcal{N}_S$. Nevertheless, the knowledge distillation approach as discussed in Eq. 2 can be incorporated to further inherit more useful information from the teacher network. Therefore, we reformulate Eq. 7 as

$$\mathcal{L}_{Total} = \frac{1}{m} \sum_i \left[\mathcal{H}(y^i, P_T^i) + \lambda \mathcal{H}(\tau(P_T^i), \tau(P_T^i)) \right] + \gamma \frac{1}{m} \sum_{i,j} \alpha_{ij} \|f_S^i - f_S^j\|_2^2. \qquad (8)$$

Then, we use stochastic gradient descent (SGD) approach to optimize the student network. Since the proposed LP loss is exactly a linear operation, and the gradient of $\mathcal{L}_{LP}$ with respect to f_S^i can be easily calculated as:

$$\frac{\mathcal{L}_{LP}}{\partial f_S^i} = \frac{1}{m} \sum_{j:j \neq i} \alpha_{ij} (f_S^i - f_S^j). \qquad (9)$$

The first term in Eq. 8, *i.e.* the classification loss, will affect all parameters in the student network, and parameters in $\mathcal{N}_S$ before the guided layer will be additionally updated by:

$$\frac{\partial \mathcal{L}_{LP}}{\partial W_S} = \sum_{i=1}^m \frac{\partial \mathcal{L}_{LP}}{\partial f_S^i} \cdot \frac{\partial f_S^i}{\partial W_S}, \qquad (10)$$

where $\frac{\partial f_S^i}{\partial W_S}$ is the gradient of the feature f_S^i. Alg.1 summarizes the detailed procedure of the proposed approach for learning student networks.

IV. ANALYSIS ON THE COMPLEXITY

As mentioned above, traditional FitNet based methods introduce an additional fully-connected layer, which makes the training procedure of the student network very slow. In contrast, the proposed method does not need the fully-connected

layer for connecting teacher and student networks. Thus, the space complexity and the computational complexity are much lower than those of FitNet based methods, as analyzed in Proposition 1.

Proposition 1. *Denote dimensionalities of features generated by the teacher network and the student network as d_T and d_S, respectively. For a mini-batch with m samples, the computational complexity of the proposed scheme for inheriting information from the teacher network to the student network is $\mathcal{O}(m^2(d_S + d_T))$.*

Proof. The proposed method inherits information from the teacher network to the student network by calculating the locality preserving loss $\mathcal{L}_{LP}$, whose computational complexity is calculated through three steps.

The first step is to calculate distances between features generated by the teacher network, *i.e.* $\|f_T^i - f_T^j\|_2^2$, whose computational complexity is $\mathcal{O}(m^2 d_T)$. The second step is to find the k nearest neighbors of each feature f_T^i to calculate α_{ij} in Eq. 6, whose computational complexity is $\mathcal{O}(km^2)$. Then, distances between features generated by the student network, *i.e.* $\|f_S^i - f_S^j\|_2^2$ will be calculated to obtain $\mathcal{L}_{LP}$, with the computational complexity $\mathcal{O}(m^2 d_S)$. Thus, the computational complexity of the proposed method is $\mathcal{O}(m^2(d_S + d_T + k))$.

Considering that, the dimensionality of features in a CNN is usually much larger than k and m, *e.g.* $k = 5$, $m = 128$, $d_T = 6 \times 6 \times 192 = 6912$, and $d_S = 8 \times 8 \times 80 = 5120$ in the Student 4 network as illustrated in Table I, the complexity $\mathcal{O}(km^2)$ could be ignored. Therefore, the computational complexity of the proposed method is $\mathcal{O}(m^2(d_S + d_T))$. $\square$

In contrast, traditional methods [21], [26], [22], [24], [25] use a fully-connected layer to build the connection between teacher and student network have a $\mathcal{O}(md_S d_T)$ computational complexity. According to Proposition 1, the proposed teacher-student learning paradigm has a much smaller computational complexity, which will accelerate the learning process for portable student networks. Taken the Student 4 network in Table I as an example, $md_S d_T / m^2(d_S + d_T) \approx 23$, since m is much smaller than either d_S or d_T. In addition, considering $k = 5$, $m = 256$, $d_T = 7 \times 7 \times 1024 = 50176$, and $d_S = 7 \times 7 \times 2048 = 100352$ in Table VI, $md_S d_T / m^2(d_S + d_T) \approx 131$. If the teacher network becomes more complex, d_T and $\mathcal{O}(md_S d_T)$ will be increased significantly. In addition, the fully-connected layer introduces considerable parameters with a $\mathcal{O}(d_S d_T)$ space complexity, to store such a fully-connected layer would require more than $100MB$ memory usage in practice. While the memory usage for storing parameter in the student network is only about $9MB$. In contrast, the proposed method does not have any additional parameters. We will further illustrate this superiority in the experiment part.

V. EXPERIMENTS

In this section, we implement experiments to validate the effectiveness of the method on three benchmark datasets, including MNIST, CIFAR-10, and CIFAR-100. Experimental results are further analyzed and discussed to investigate the benefits of the proposed method.

(a) accuracy = 99.3%

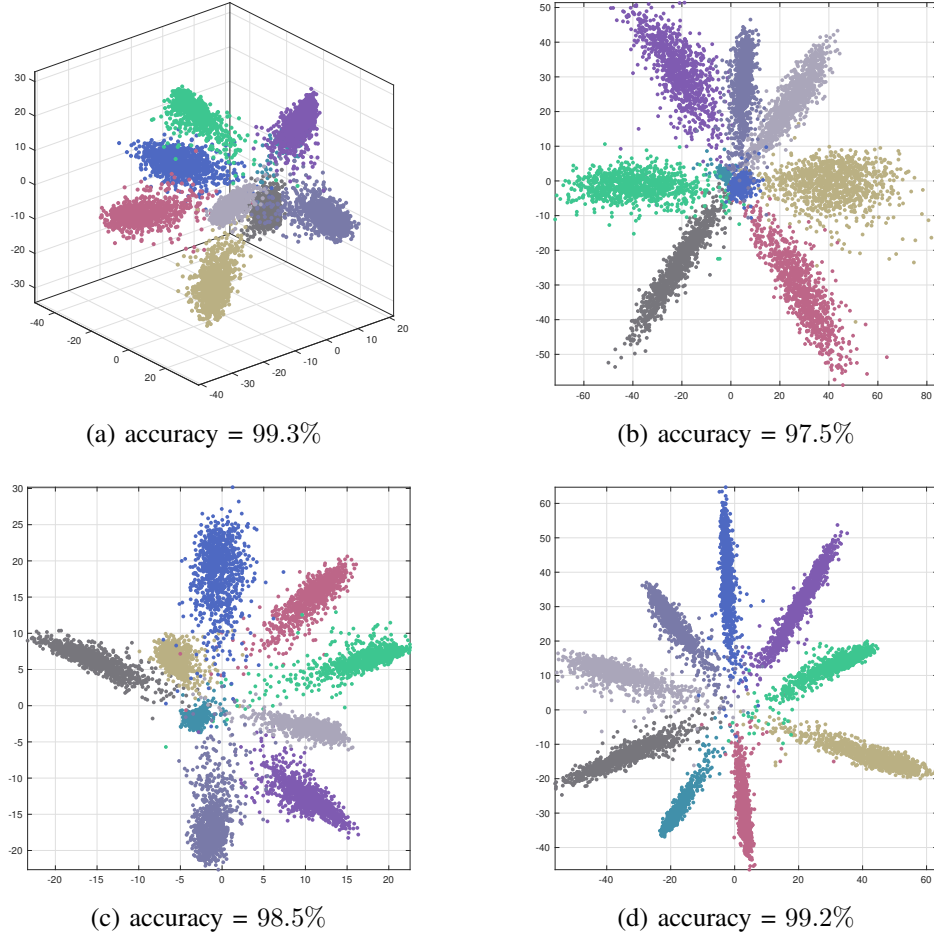(b) accuracy = 97.5%

(c) accuracy = 98.5%

(d) accuracy = 99.2%

Fig. 2: Visualization of features generated by different networks on the MNIST test set. (a) features of the original teacher network; (b) features of the student network learned using the standard back-propagation strategy; (c) features of the student network learned using FitNet; (d) features of the student network learned using the proposed method with locality preserving loss. Note that features of the same category in each figures are marked with the same color.

A. Validations on MNIST

Visualization of Features. The locality preserving loss was introduced in Fcn. 8, which aims to learn the effective low-dimensional features from the pre-trained teacher network. In order to illustrate the superiority of the proposed method, we first trained a LeNet++ [48] as the teacher network, which has six convolutional layers and a fully-connected layers for extracting powerful 3D features. Numbers of neurons in each convolutional layer are 32, 32, 64, 64, 128, 128, and 3, respectively. This teacher network achieved a 99.3% test accuracy on the MNIST dataset, and the memory usage for storing all convolution filters of this teacher network is about 2,982KB.

Then, a thinner student network with also six convolutional layers but half convolution filters per layer was initialized. As for the fully-connected layer, the number of neurons was set as 2 for seeking a low-dimensional embedding. Then, we trained this network using conventional back-propagation scheme, and the FitNet method on the MNIST dataset, respectively. Since the student network has fewer parameters, the accuracy of the network using conventional BP is only 97.5%, and the accuracy of the network using FitNet is about 98.5%.

We then trained a new student network with the same architecture by exploiting the proposed method as described in Fcn. 8. λ and τ were equal to 2 and 0.5, respectively, which refer to those in Hinton *et.al.* [19]. The learning rate η was set to be 0.01, empirically. The hyper-parameter k for searching neighbors was set as 5, and γ was set to be 1. The accuracy of the resulting student network is 99.2% which is slightly lower than that of its teacher network but much higher than that of the student network straightforwardly learned using the conventional back-propagation method. In addition, the memory usage for convolution filters of this student network is about 734KB, which only accounts for $\frac{1}{4}$ of that of the original teacher network.

Moreover, features (*i.e.* input data of the softmax layer) of the above three networks were visualized in Fig. 2. Fig. 2 (a) shows that features of different categories extracted using the original teacher network are separated from each other in the 3-dimensional space, and can be easily distinguished by the following softmax layer. Since the student network has fewer parameters, features of the network trained using the conventional back-propagation are distorted as illustrated in Fig. 2 (b). Therefore, the performance of this network is lower

TABLE I: The performance of the proposed method on student networks with various architectures.

Network	#layers	#params	#mult	speed-up	compression	Training time (min)		Additional params		Accuracy	
						FitNet	Ours	FitNet	Ours	FitNet	Ours
Teacher	5	∼9M	∼725M	×1	×1	150		-		90.21%	
Student 1	11	∼250K	∼30M	×**13.17**	×**36**	146	93	∼14M	0	89.03%	89.56%
Student 2	11	∼862K	∼108M	×4.56	×10.44	221	147	∼35M	0	91.01%	91.33%
Student 3	13	∼1.6M	∼392M	×1.40	×5.62	317	200	∼35M	0	91.14%	91.57%
Student 4	19	∼2.5M	∼382M	×1.58	×3.60	400	265	∼35M	0	91.55%	**91.91%**

deeper is better

TABLE II: Different architectures on the CIFAR-10 dataset.

Teacher	Student 1	Student 2	Student 3	Student 4
conv 3x3x96 pool 4x4	conv 3x3x16 conv 3x3x16 conv 3x3x16 pool 2x2	conv 3x3x16 conv 3x3x32 conv 3x3x32 pool 2x2	conv 3x3x32 conv 3x3x48 conv 3x3x64 conv 3x3x64 pool 2x2	conv 3x3x32 conv 3x3x32 conv 3x3x32 conv 3x3x48 conv 3x3x48 pool 2x2
conv 3x3x96 pool 4x4	conv 3x3x32 conv 3x3x32 conv 3x3x32 pool 2x2	conv 3x3x48 conv 3x3x64 conv 3x3x80 pool 2x2	conv 3x3x80 conv 3x3x80 conv 3x3x80 conv 3x3x80 pool 2x2	conv 3x3x80 conv 3x3x80 conv 3x3x80 conv 3x3x80 conv 3x3x80 conv 3x3x80 pool 2x2
conv 3x3x96 pool 4x4	conv 3x3x48 conv 3x3x48 conv 3x3x64 pool 8x8	conv 3x3x96 conv 3x3x96 conv 3x3x128 pool 8x8	conv 3x3x128 conv 3x3x128 conv 3x3x128 pool 8x8	conv 3x3x128 conv 3x3x128 conv 3x3x128 conv 3x3x128 conv 3x3x128 conv 3x3x128 pool 8x8
fc softmax	fc softmax	fc softmax	fc softmax	fc softmax

TABLE III: Classification error on the MNIST dataset.

Algorithm	#params	Misclass
Teacher	∼361K	0.55%
Standard back-propagation	∼30K	1.90%
Knowledge Distillation [19]	∼30K	0.65%
FitNet [21]	∼30K	0.51%
Student (Ours)	∼30K	**0.48%**

than that of the teacher network. Fig. 2 (d) shows features of the student network learned using the proposed scheme with LP loss. It is clear that, features are separated by supervising of the proposed locality preserving loss, which can be seen as an excellent low-dimensional embedding of features learned by the teacher network.

Compression Results. In order to further illustrate the superiority of the proposed method, we followed the setting in Romero *et.al.* [21] and Wang *et.al.* [26] to conduct the student network learning experiment on the MNIST dataset. The teacher network consists of maxout convolutional layers as reported in Goodfellow *et.al.* [49] and the student network is twice as deep as the teacher network following Romero *et.al.* [21]. The hyper-parameter k of the proposed

method for searching neighbors was set as 5, and γ was set to be 1, which were tuned on the last 5,000 images in the train set. The impact of parameters would be showed in the following experiment. The teacher network contains 3 maxout layers and a fully-connected layer, and the student network has 6 maxout layers and a fully-connected layer. Parameters in the student network is only about 8% of that in the teacher network. The guide layer of the teacher network is the 2nd layer while the hint layer of the student network is the 4th layer, respectively. The parameters of the networks were initialized randomly in U(-0.005,0.005) and the networks were trained using stochastic gradient descent with RMSProp whose learning rate is 0.0005 and weight decay is 0.9.

Table III reports the results of different networks on the MNIST dataset by exploiting the proposed method. In order to illustrate the advantage of the introduced locality preserving loss, the performance of student networks with the same architecture trained by using standard back-propagation, knowledge distillation [19], and FitNets [21] was also reported. It can be found in Table III, the student network trained using the standard back-propagation achieved a 1.90% error rate. The student network learned utilizing the knowledge distillation obtained a 0.65% misclassification error. The error rate of the student network trained by exploiting the FitNet approach is 0.51%, which outperforms both conventional back-propagation

TABLE IV: 10-Class results of different networks on the CIFAR-10 datasets.

Algorithm	plane	car	bird	cat	deer	dog	frog	horse	ship	truck
Teacher	90.1%	93.8%	86.0%	74.6%	93.5%	86.2%	95.2%	92.6%	95.3%	95.2%
FitNet [21]	90.7%	97.6%	91.0%	82.7%	93.8%	86.2%	92.7%	93.6%	94.6%	93.5%
Knowledge Distillation [19]	90.0%	95.2%	83.2%	84.4%	93.2%	87.1%	95.0%	91.6%	97.3%	93.7%
Multiple Teachers [22]	91.0%	96.8%	90.0%	83.1%	92.8%	87.1%	93.3%	94.2%	94.4%	93.8%
FSP Learning [23]	90.9%	96.7%	90.4%	82.9%	93.3%	87.1%	95.6%	92.7%	96.0%	93.1%
Adversarial Learning [26]	91.3%	96.5%	89.8%	84.2%	91.8%	87.5%	93.1%	95.3%	93.4%	93.8%
Student(Ours)	91.1%	97.0%	90.2%	83.6%	92.4%	87.2%	95.3%	93.2%	95.1%	94.1%

and knowledge distillation methods, and is slightly lower than that of the teacher network. In contrast, the student network using the proposed method achieves a 0.48% accuracy.
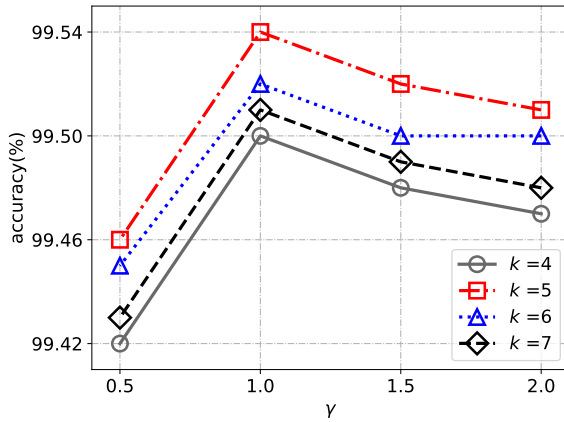

Fig. 3: The performance of the proposed method with different parameters α and β on the MNIST dataset.

Impact of parameters. As discussed above, the proposed method has two hyper-parameters: k and γ. We tested their impact on the accuracy of the student network on the validation set of MNIST dataset. It can be seen from Figure 3 that the student network trained utilizing the proposed method achieved the highest accuracy (99.54%) on the validation set when $k = 5$ and $\gamma = 1$. Therefore, we kept these hyper-parameters for the proposed method on the following experiments.

B. Validations on CIFAR-10

After investigating the performance of the proposed method on the MNIST dataset, we then evaluated our method on the CIFAR-10 dataset, which consists of 32×32 pixel RGB color images with 10 categories. There are 50,000 training images and 10,000 testing images in this dataset. For fair comparison, images in the dataset were first processed using global contrast normalization (GCA) and ZCA whitening as suggested in Goodfellow *et.al.* [49] and Romero *et.al.* [21], and the last 10,000 training images were selected as the validation set for tuning the hyper-parameters. We followed Goodfellow *et.al.* [49] to train a teacher network, which consists of three convolutional layers and a fully-connected layer of 500 units with 5-linear-piece maxout activations. The

teacher network has about 9M parameters with a 90.21% classification accuracy, and trained for 400 epochs using RMSProp with an initial learning rate of 0.005 and a weight decay of 0.9. Note that the intermediate hidden layer of the teacher network and student network used in the proposed methods is same as that of FitNet [21] for a fair comparison.

Trade off between compression/speed-up and accuracy. Since the CIFAR-10 dataset consists of more complex images, its samples cannot be easily distinguished by a casually designed neural network. Therefore, several student networks with different architectures, as reported in Romero *et.al.* [21], were established to further explore the compression performance and accuracy of the proposed method. Table II showed the detailed structures of these student networks. Numbers of parameters in these student networks are 250K, 862K, 1.6M, and 2.5M, respectively. The compression ratio and the speed-up ratio of each student network can be directly calculated by comparing its numbers of parameters and the floating number multiplications to those of the teacher network, respectively. Detailed results were reported in Table I.

It can be found in Table I that, the student network (Student 1) with the highest compression and speed-up ratios has a lower classification accuracy. While, when we appropriately increase the number of parameters, student networks (Student 2-4) can achieve higher performance than that of the teacher network. Moreover, compared with the results of student networks learned using FitNet [21], the proposed method achieves better performance of all the four student networks.

Complexity. As discussed in Proposition 1, the proposed method has significantly lower computational and space complexities. Therefore, we also report training time and additional parameters (*i.e.* weights in the fully-connected layer) of the proposed method and conventional FitNet based method in Table I. It is obvious that, the training time using the proposed method is much less than those of the FitNet and other FitNet based methohds, and we do not need any additional parameters for training student networks. Considering that there are lots of networks with heavy architectures [1], [6], the proposed method is a much more flexible approach with higher performance and efficiency.

Comparison with state-of-the-art methods. To illustrate the superiority of the proposed method, we compared it with other student-teacher learning methods using the same architecture (Student 4) on the CIFAR-10 dataset as summarized in Table V and Table IV. As a result, the student network utilizing the

TABLE V: Classification results of different networks on CIFAR-10 and CIFAR-100 datasets.

Algorithm	#layers	#params	CIFAR-10	CIFAR-100
Teacher	5	∼9M	90.21%	62.78%
Student(Ours)	19	∼2.5M	**91.91%**	**65.13%**
FitNet [21]	19	∼2.5M	91.55%	64.89%
Knowledge Distillation [19]	19	∼2.5M	91.04%	63.07%
Multiple Teachers [22]	19	∼2.5M	91.66%	65.06%
FSP Learning [23]	19	∼2.5M	91.89%	64.65%
Adversarial Learning [26]	19	∼2.5M	91.68%	65.11%

proposed method achieved a 91.91% accuracy. Comparison results show that our student network outperforms networks produced by state-of-the-art approaches, which proves that the proposed LP loss transfers more useful and intrinsic information from the teacher network. In addition, the student network with more portable architecture (Student 3) can achieve a 91.57% accuracy, which is slightly higher than that of the student network (Student 4) trained using FitNet.

C. Validations on CIFAR-100

Moreover, we also conducted the validation on the CIFAR-100 dataset. This dataset has 60,000 RGB color images of pixel 32×32, which is the same as that of CIFAR-10. However, the CIFAR-100 dataset has 100 categories with only 600 images per class, which is a more challenging benchmark dataset for conducting the classification experiment. For example, the teacher network used in Table V on the CIFAR-100 dataset is only about 62%, which is much lower than that on the CIFAR-10 dataset. Similarly, images in this dataset were pre-processed using global contrast normalization and ZCA whitening. Random flipping, random crop and zero padding was also used for data augmentation as suggested in Romero *et.al.* [21].

We used the fourth student network (Student 4 in Table I) to conduct the experiment on the CIFAR-100 dataset. Table V reports the classification results of student networks learned by exploiting different student-teacher learning paradigms with the same architecture. As a result, the student network learned by the proposed method obtained a 65.13% accuracy. It is clear that the student network learned using the proposed LP loss outperforms those of networks generated by the state-of-the-art methods, which demonstrated that the proposed method inherits useful information from the teacher network in a more effective and efficient way.

D. Validations on ImageNet

We next conducted experiments on an extremely large image dataset, namely ImageNet ILSVRC 2012 [2], which has about 1.28M training images and 50K validation images. We followed the settings in Huang and Wang [25] to implement the teacher-student learning paradigm. Wherein, we used ResNet-101 [17] and Inception-BN [50] as the teacher and student networks, respectively. The teacher network has about 128M parameters and student network has only about

32M parameters, which is a relatively portable model. We used the scale and aspect ratio augmentation in Ioffe and Szegedy [50] and color augmentation in Krizhevsky *et.al.* [2] following Huang and Wang [25].

These networks were optimized using Nesterov Accelerated Gradient (NAG), and the weight decay and the momentum were set as 10^{-4} and 0.9, respectively. We trained the networks for 100 epochs, and the initial learning rate was set as 0.1 and divided by 10 at the 30, 60 and 90 epochs, respectively. The batch size was set as 256 and the hyper-parameters of the proposed method are the same as those in CIFAR experiments. In addition, λ and τ in knowledge distillation were equal to 2 and 0.5, respectively [25]. For FitNet, attention transfer and neuron selectivity transfer, the value of λ was set to 10^2, 10^3 and 5, respectively, which refers to the settings in [25].

Table V shows the classification results of student networks on the ImageNet dataset by exploiting the proposed method and state-of-the-art learning methods. The top-1 accuracy and the top-5 accuracy of the teacher network are 77.32% and 93.42%, respectively. The student network without inheriting information (*i.e.* the standard BP) from the teacher achieved a 74.80% top-1 accuracy and a 92.18% top-5 accuracy.

It can be found in Table V that, all the teacher-student learning methods achieve better results than that of the standard back-propagation, which demonstrates that there are abundant information in the teacher network. The student network learned using the proposed method obtained a 93.13% top-5 accuracy, which is about 1% higher than that of the original student network. When compared to other methods, the student network learned through the proposed locality preserving loss achieved the highest accuracy. In addition, as discussed in Proposition 1, the proposed method has significantly lower computational and space complexities. Especially, deep neural networks on the ImageNet dataset have sophisticated architectures with billions of parameters, and the proposed method is more suitable for efficiently learning portable student networks. For example, the number of additional parameters for connecting teacher and student networks needed by FitNet based methods is about 5,035M. In contrast, the proposed method does not need any additional parameters but can obtain better results.

E. Intermediate Layer Selection

Recalling Eq. 3, most of the teacher-student learning algorithms utilize the output features of intermediate layers of

TABLE VI: Classification results of different networks on ImageNet datasets.

Algorithm	Model	#params	Top-1	Top-5
Teacher	ResNet-101	~128M	77.32%	93.42%
Student using LP loss (Ours)	Inception-BN	~32M	**75.91%**	**93.13%**
Standard back-propagation	Inception-BN	~32M	74.80%	92.18%
FitNet [21]	Inception-BN	~32M	75.52%	92.73%
Knowledge Distillation [19]	Inception-BN	~32M	75.44%	92.65%
Attention Transfer [24]	Inception-BN	~32M	75.36%	92.74%
Neuron Selectivity Transfer [25]	Inception-BN	~32M	75.66%	92.89%

the teacher and student networks (*i.e.* the guide layer and hint layer). An important issue is how to select the hint layer from teacher network and the guided layer from the student network and how would the selection of such layers affect the performance. Zagoruyko and Komodakis [51] has discussed this problem and the experimental results showed that different selections of intermediate layers could only have a minor influence on the accuracy of student networks (less than 0.2% on the CIFAR-10 dataset). Moreover, using more than 1 guided layer/hint layer results in a slight improvement of performance (less than 0.1% on the CIFAR-10 dataset). Therefore, we do not study the selection of intermediate layers. Instead, we focus on the performance of the student networks under the same experimental settings.

VI. CONCLUSION

Here we examine the deep neural network compression problem for learning portable networks from original teacher models. Besides features generated by the teacher network, the relationship between samples in the feature space is another important information for maintaining the performance of the student network. In this paper, we present a novel teacher-student learning paradigm by introducing the locality preserving loss. The neighbor relationship between samples represented by the teacher network is embedded into the student network. Therefore, the resulting portable network can achieve similar performance as that of the teacher network naturally. Experiments on benchmark datasets show that the proposed method can efficiently produce portable neural networks with higher performance, which is superior to the state-of-the-art approaches.

REFERENCES

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1097–1105.

[3] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2017.

[4] Y. Yuan, L. Mou, and X. Lu, "Scene recognition by manifold regularized deep learning architecture," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 10, pp. 2222–2233, 2015.

[5] W. Hou, X. Gao, D. Tao, and X. Li, "Blind image quality assessment via deep learning," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 6, pp. 1275–1286, 2015.

[6] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *NIPS*, 2015, pp. 91–99.

[7] M. Gong, J. Zhao, J. Liu, Q. Miao, and L. Jiao, "Change detection in synthetic aperture radar images based on deep neural networks," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 1, pp. 125–138, 2016.

[8] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015, pp. 3431–3440.

[9] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," *arXiv preprint arXiv:1412.6115*, 2014.

[10] J. Cheng, J. Wu, C. Leng, Y. Wang, and Q. Hu, "Quantized cnn: a unified approach to accelerate and compress convolutional networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2017.

[11] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *NIPS*, 2014.

[12] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," in *ICML*, 2015.

[13] P. Gysel, J. Pimentel, M. Motamedi, and S. Ghiasi, "Ristretto: A framework for empirical study of resource-efficient inference in convolutional neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2018.

[14] Y. Wang, C. Xu, S. You, D. Tao, and C. Xu, "Cnnpack: packing convolutional neural networks in the frequency domain," in *NIPS*, 2016, pp. 253–261.

[15] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[16] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *NIPS*, 2015.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[18] J. Ba and R. Caruana, "Do deep nets really need to be deep?" in *NIPS*, 2014.

[19] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[20] P. McClure and N. Kriegeskorte, "Representational distance learning for deep neural networks," *Frontiers in computational neuroscience*, vol. 10, 2016.

[21] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," in *ICLR*, 2015.

[22] S. You, C. Xu, C. Xu, and D. Tao, "Learning from multiple teacher networks," in *ACM SIGKDD*, 2017.

[23] J. Yim, D. Joo, J. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *CVPR*, 2017.

[24] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," in *ICLR*, 2017.

[25] Z. Huang and N. Wang, "Like what you like: Knowledge distill via neuron selectivity transfer," *arXiv preprint arXiv:1707.01219*, 2017.

[26] Y. Wang, C. Xu, C. Xu, and D. Tao, "Adversaial learning of portable student networks," in *AAAI*, 2018.

[27] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," *arXiv preprint arXiv:1412.6115*, 2014.

[28] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with

weights and activations constrained to+ 1 or-1," *arXiv preprint arXiv:1602.02830*, 2016.

[29] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *ECCV*. Springer, 2016, pp. 525–542.

[30] Y. Wang, C. Xu, C. Xu, and D. Tao, "Beyond filters: Compact feature map for portable deep model," in *ICML*, 2017, pp. 3703–3711.

[31] K. Sun, S.-H. Huang, D. S.-H. Wong, and S.-S. Jang, "Design and application of a variable selection method for multilayer perceptron neural network with lasso," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 6, pp. 1386–1396, 2017.

[32] J. Wang, C. Xu, X. Yang, and J. M. Zurada, "A novel pruning algorithm for smoothing feedforward neural networks based on group lasso method," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 5, pp. 2012–2024, 2018.

[33] H. Huang and H. Yu, "Ltnn: A layerwise tensorized compression of multilayer neural network," *IEEE transactions on neural networks and learning systems*, 2018.

[34] J. Jin, A. Dundar, and E. Culurciello, "Flattened convolutional neural networks for feedforward acceleration," *arXiv preprint arXiv:1412.5474*, 2014.

[35] M. Wang, B. Liu, and H. Foroosh, "Factorized convolutional neural networks." in *ICCV Workshops*, 2017, pp. 545–553.

[36] 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," *arXiv preprint arXiv:1602.07360*, 2016.

[37] Y. Pang, M. Sun, X. Jiang, and X. Li, "Convolution in convolution for network in network," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 5, pp. 1587–1597, 2018.

[38] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[39] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," *arXiv preprint arXiv:1707.01083*, 2017.

[40] B. Wu, A. Wan, X. Yue, P. Jin, S. Zhao, N. Golmant, A. Gholaminejad, J. Gonzalez, and K. Keutzer, "Shift: A zero flop, zero parameter alternative to spatial convolutions," *arXiv preprint arXiv:1711.08141*, 2017.

[41] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *CVPR*, 2018, pp. 4510–4520.

[42] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," *arXiv preprint arXiv:1807.11164*, 2018.

[43] C. X. C. X. D. T. Yunhe Wang, Chang Xu, "Learning versatile filters for efficient convolutional neural networks," in *NIPS*, 2018.

[44] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[45] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[46] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *NIPS*, 2002, pp. 585–591.

[47] X. He and P. Niyogi, "Locality preserving projections," in *NIPS*, 2004, pp. 153–160.

[48] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *ECCV*, 2016.

[49] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," *arXiv preprint arXiv:1302.4389*, 2013.

[50] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015, pp. 448–456.

[51] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," *arXiv preprint arXiv:1612.03928*, 2016.