IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 2, 2018, accepted July 8, 2018, date of publication July 11, 2018, date of current version August 7, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2854918

Beyond Knowledge Distillation: Collaborative Learning for Bidirectional Model Assistance

JINZHUO WANG[1], WENMIN WANG[1], (Member, IEEE), AND WEN GAO[2], (Fellow, IEEE)
[1]School of Electronic and Computer Engineering, Peking University, Shenzhen 518055, China
[2]School of Electronic Engineering and Computer Science, Peking University, Beijing 100086, China

Corresponding author: Wenmin Wang (wangwm@ece.pku.edu.cn)

This work was supported in part by the Shenzhen Peacock Plan under Grant 20130408-183003656, in part by the Shenzhen Key Laboratory for Intelligent Multimedia and Virtual Reality under Grant ZDSYS201703031405467, and in part by the National Natural Science Foundation of China under Grant U1613209.

ABSTRACT Knowledge distillation (KD) is a powerful technique that enables a well-trained large model to assist a small model. However, KD is constrained in a teacher-student manner. Thus, this method may not be appropriate in general situations, where the learning abilities of two models are uncertain or not significantly different. In this paper, we propose a collaborative learning (CL) method, which is a flexible strategy to achieve bidirectional model assistance for two models using a mutual knowledge base (MKB). The MKB is used to collect mutual information and provide assistance, and it is updated along with the learning process of the two models and separately deployed when converged. We show that CL can be applied to any two deep neural networks and is easily extended to multiple networks. Compared with the teacher-student framework, CL can achieve bidirectional assistance and does not impose specific requirements on the involved models, such as pretraining and different abilities. The experimental results demonstrate that CL can efficiently improve the learning ability and convergence speed of the two models, with superior performance to a series of relevant methods, such as ensemble learning and a series of KD-based methods. More importantly, we show that the state-of-the-art models, such as DenseNet, can be greatly improved using CL along with other popular models.

INDEX TERMS Bidirectional model assistance, collaborative learning, deep neural networks, mutual knowledge base.

I. INTRODUCTION

We are in an era of deep neural networks being designed constantly. In particular, a typical evolution of successful deep networks is the streamline of LeNet [1], AlexNet [2], VGG [3], ResNet [4] and DenseNet [5]. Today, we are easy to access these models to produce promising results on various tasks such as computer vision, natural language processing and speech recognition [5]–[8]. Most researchers are devoted to hand-crafted engineering of single networks by borrowing advanced techniques from previously designed networks or making new attempts, including architecture design [9], parameter tuning [10], and optimization [11]. However, few studies have focused on using these deep neural networks in an interactive manner. Although existing model integration methods such as voting, fusion and ensemble have made some progress [12]–[15], they have major limitations such as expensive training and deployment, lack of knowledge sharing in the learning process, and difficulty adapt to deep neural networks [16]–[18].

Knowledge distillation (KD) [17] is a recently proposed method that enables a well-trained large model to help a small model. KD views the behavior of the large model such as the prediction as ''soft target'' and uses this ''soft target'' as additional supervision to assist the small model. KD has been shown to facilitate performance improvements in a small model by providing new knowledge in addition to the standard supervision [19]. Many subsequent works derived from KD have achieved promising results in various scenarios, such as network training [20], embedding learning [21], transfer learning [22], semantic segmentation [23] and object detection [24]. Although these different approaches vary in terms of distilled knowledge expression and application scenarios, they all share a key characteristic: they are performed in a teacher-student framework. To be more precise, these approaches all require a well-trained teacher model to provide the ''soft target'', and this teacher model has to be more powerful than the student model. More importantly, the teacher model cannot be improved with KD. This indicates that KD performs a unidirectional model assistance procedure.

We attempt to break the aforementioned limitations and seek a more general solution to implementing collaboration of the existing models, and enabling them to assist each other in a bidirectional way. Our motivation also derives from the perspectives of feature extraction and data representation. We observe that various models of different network structures [3]–[5], [25]–[27] can produce similar and promising results on the same dataset. This suggest that the same data can be described from different aspects with different kinds of abstraction, which encourages us to pursue a more comprehensive investigation of data by learning different feature hierarchies. From the perspective of game theory, if we have many agents capable of achieving the same goal, then they should be able to collaborate to achieve a better performance by sharing information in the learning process.

In this paper, we present collaborative learning, a flexible method to achieve bidirectional model assistance. Collaborative learning can be applied to any existing feed-forward computation models, especially deep neural networks, to enhance their learning ability in a win-win process, by sharing mutual information in the learning process. In particular, collaborative learning is performed around a mutual knowledge base (MKB), which is used to collect different abstractions of data in the learning process of each model to assist the other. MKB contains an auto-encoder structure, a metric learning module and a verification component. The encoder is in charge of transferring intermediate feature maps of involved models to unified embeddings in a mutual space, where the embeddings are then sent to three trunks. The first trunk is metric learning where additional supervision can be derived to assist each other. The second trunk is a decoder which sends the embeddings back to the original networks to keep them end-to-end trainable. The last trunk is the verification part which links the embeddings directly to the semantic label to ensure the quality of embeddings as well as the mutual space. We provide good practices to insert MKB in the proper position with careful configurations to achieve the best performance. In addition, using multiple MKBs can further boost the performance of collaborative learning. Furthermore, with adaptions to MKB implementations, collaborative learning can be easily extended to multiple networks. Compared with a series of relevant methods in teacher-student fashion, collaborative learning enjoys bidirectional assistance and needs no specific requirements of the involved models such as pretraining and ability differences. Experimental results show that collaborative learning has three major advantages:

- It can facilitate fast convergence among the associated models, which suggests that useful mutual knowledge is efficiently leveraged in the CL process. Each of the two networks trained with CL can reach the saturation region at least two times faster than individually trained networks.
- It is able to improve the learning ability of the two involved models when converged. This shows that collaborative learning achieves efficient bidirectional model assistance. We compare the proposed strategy with a series of relevant methods and show that it presents promising performance improvements.
- It is able to improve state-of-the-art models with other advanced models. This indicates that collaborative learning is general and powerful solution. We use DenseNet [5], a state-of-the-art deep network, and show that it can be considerably improved with other deep networks.

The remainder of this paper is organized as follows. Section II reviews the relevant works from three perspectives. Section III describes the details of collaborative learning, including its key modules, working principles, training manners and extensions. Section IV evaluates the performance of collaborative learning on five popular large-scale benchmarks and demonstrates its advantages on three aspects. Section V concludes this paper.

II. RELATED WORKS

A. MODEL ASSISTANCE

In the literature, model assistance is used in various scenarios. One common way to achieve model assistance is serial execution, such as using the output of selective search [28] as the input of CNN to perform R-CNN [29]. This assistance is unidirectional, where the source model assists the target model, while the latter offers no help for the former. An improved version is Faster R-CNN [30], where the RPN and CNN help each other in a bidirectional process, which is also the property of our collaborative learning. Later, model assistance is also exploited with domain adaption and unsupervised learning [31]. In a nutshell, model assistance attempts to extract mutual knowledge for assistance, which can be categorized into knowledge integration and knowledge transfer. Specifically, a typical solution of knowledge integration is ensemble learning [32], where the key is to integrate learned knowledge of all models such as using ensemble predictions, widely adopted in recognition competitions [33]. The key concept underlying knowledge transfer is to extract what one model learns to assist other models. For example, [34] used what a wide network learns to help the fit network for better training. Our proposed collaborative learning can be viewed as a combination of both knowledge integration and transfer, since it is performed by sharing mutual knowledge along with the learning processes of the involved models.

B. KNOWLEDGE DISTILLATION AND EXTENSIONS

A recent successful method for model assistance is knowledge distillation (KD) proposed by Hinton *et al.* [17]. KD is a successful model assistance method that can efficiently leverage what a large model learns as "soft target" to guide the training of a small model. Later, its variants and extensions achieved promising results. In particular, [34] devised a hint-based training approach that uses the pretrained teachers hint layer to assist students guided layer. The trained deep student network showed better accuracy with fewer parameters compared to the original wide teacher network. Reference [19] developed a soft decision tree to exploit the distilled knowledge in the teacher model. Very recently,

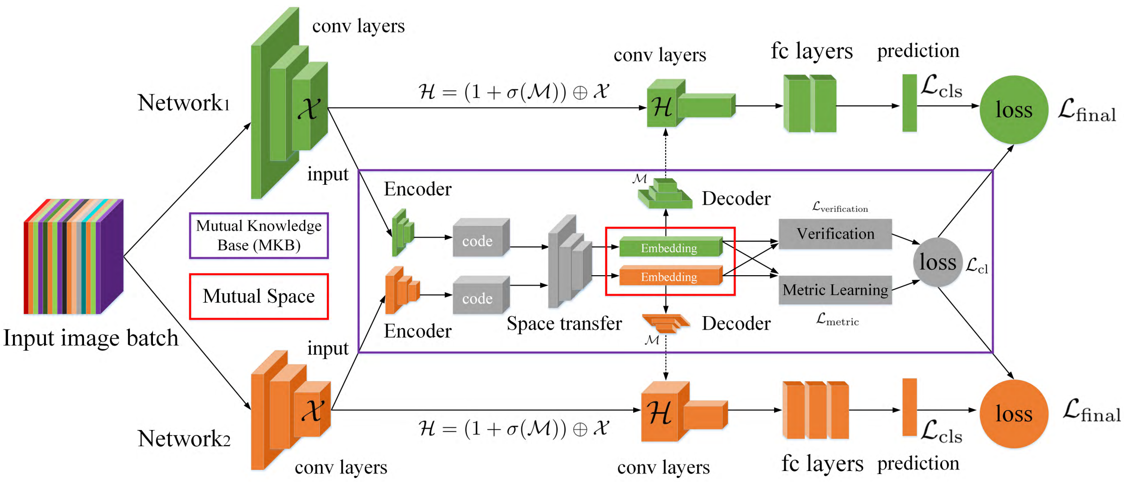

FIGURE 1. Illustration of collaborative learning performed with two standard deep neural networks. More details can be found in the context.

a series of applications using KD are presented such as object detection [24], person-ReID [35], domain adaption [31], transfer learning [22]. However, the above works are all limited in the teacher-student manner. Specifically, the teacher model provides guidance to enhance the learning process of the student model, while the teacher model itself cannot be improved. Our collaborative learning can overcome this limitation, and enables both models to assist each other and achieve bidirectional improvements in terms of fast convergence and classification accuracy. In addition, we show that state-of-the-art models can be further improved by a large margin, using collaborative learning with other popular models, which cannot be achieved by other methods such as ensemble learning and fusion [12], [13], [36].

C. JOINT LEARNING AND ADDITIONAL SUPERVISION

Our method is also related to joint learning technique. In a multiple model system, joint learning is often used by sharing weights or jointly optimizing weights of each involved model. A typical line of works is multi-task learning (MTL) [37], which aims to improve all tasks simultaneously by combining common knowledge from all tasks such as [38] and [39]. There are other joint learning works performed for object detection [24], visual question answering [40], and person re-identification [35]. Compared with MTL, our collaborative learning also aims at sharing common knowledge from each model, but performed within the same task. In addition, we construct a learnable module and a mutual space which are trained along with each model, instead of simply jointly optimizing the weights.

When training a model in the supervised setting, in addition to the true label, one can mine other information for additional supervision. This is also the key idea behind of our collaborative learning. This additional supervision can be derived from various sources and one of the most popular is metric learning [41], widely used in retrieval tasks [21], [35], [42]–[44]. Specifically, [42] learned an energy-based similarity metric that maps input patterns into a target space such that the L_1 norm in the target space approximates the semantic distance in the input space. Reference [43] designed a triplet loss to train three same networks with an additional supervision on its semantic similarity. Reference [44] introduced a deep mutual learning method for simultaneous training of two models by adding an additional mutual loss for each, which forces them to produce similar final predictions. Reference [35] employed this idea for the task of person re-identification. Reference [21] encouraged the model to produce similar feature maps to the semantic label of input data on Wasserstein distance. The additional supervision in our collaborative learning is derived from MKB and can be used by all the involved models.

III. APPROACH

A. OVERVIEW

We use two standard deep neural networks (stacked by a series of convolutional layers followed by fully connected layers) under a classification task scenario to explain the principles of collaborative learning. As shown in Fig. 1, collaborative learning is performed around a mutual knowledge base (MKB), which is in charge of collecting information in the learning process of each involved model to assist each other.

When putting MKB between two models, the feature maps of current layers of both networks are two abstractions of the current input data. MKB uses these abstraction to produce integrated knowledge in a mutual space as additional supervision, and sends it back to both networks to facilitate their learning processes. In this way, the loss function for each deep neural network is formulated as

$$\mathcal{L}_{\text{final}} = \mathcal{L}_{\text{cls}} + \alpha \mathcal{L}_{\text{cl}} \quad (1)$$

$$\mathcal{L}_{\text{cl}} = \mathcal{L}_{\text{metric}} + \mathcal{L}_{\text{verification}} \quad (2)$$

where the first item $\mathcal{L}_{\text{cls}}$ in Equation 1 indicates the standard classification loss, the latter $\mathcal{L}_{\text{cl}}$ is the collaborative loss induced by MKB and collaborative learning. In particular, $\mathcal{L}_{\text{cl}}$ consists of two parts. The first one is $\mathcal{L}_{\text{metric}}$, which is used to perform metric learning to mine mutual knowledge about the data, which is learned from each network. In particular, $\mathcal{L}_{\text{metric}}$ is performed in a constructed mutual space, which can be regarded as a new abstraction level of the input data. Note that this level can not be learned with the original two networks individually. The second one is $\mathcal{L}_{\text{verification}}$, used to ensure the quality of the mutual space. In practice, we directly link this mutual space to the final semantic space, which is the same as the standard supervision $\mathcal{L}_{\text{cls}}$. α is a coefficient that controls the ratio of true label and mutual information to avoid over-regularization.

As shown in Fig. 1, by interacting with all two models, MKB collects different abstractions of data as an integration of feature hierarchies. On the other hand, it uses such knowledge to facilitate the learning processes of each model. Note that collaborative structured losses are shared by all the involved models in the collaborative structure. Compared with previous works that force two deep neural networks to generate similar predictions [44], the loss derived from our collaborative structure is governed by the encoder-decoder structure and updated along with the learning processes of all the involved models.

B. MKB IMPLEMENTATIONS

MKB includes an encoder-decoder structure, a metric learning module and a verification component. As shown in Fig. 1, the encoder receives feature maps of involved networks as input. Since feature maps of each network may have various sizes, this encoder is designed to resize them into unified codes. These two codes are then sent to a space transfer, which is constructed by a series of convolutional layers, to map them to embeddings in a mutual space, where we can leverage their relevance to achieve assistance by metric learning. This mutual space shares the same property with semantic space where two embeddings are reasonable to compare [17].

The output of each network in the mutual space has three branches using the same value as shown in Fig. 2. The first brunch routes back to the original network architecture and the input of the next layer $\mathcal{H}$ is computed by

$$\mathcal{H} = (1 + \sigma(\mathcal{M})) \oplus \mathcal{X} \tag{3}$$

where $\mathcal{X}$ is the feature map of previous layer and also the input of MKB, $\mathcal{M}$ is the embedding in the mutual space, σ is the elu function and $\oplus$ is the element-wise sum operation. This operation firstly ensures that all the modules in our framework are end-to-end trainable. Second, the circuit routed in MKB acts as a selection operation that helps to attend to the important region in the previous feature maps, as shown in Fig. 2, similar to recent proposed residual function [4] and residual attention network [45].

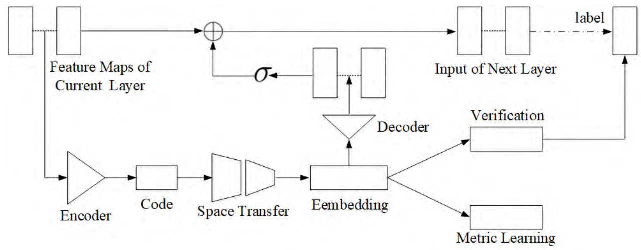

FIGURE 2. Integration between two consecutive layers with one MKB.

The other two branches are sent to metric learning and verification part to act as additional supervision for each other. We expect MKB can help two networks learn from each other and encourage their embeddings in the mutual space mimic each other. Different from the basic KD [17] and its variant [19], where the "soft target" comes from the cumbersome ensemble model and the value is certain for each input, our metric learning is dynamic since it is governed in the mutual space, which is updated along with the training processes of the involved models. The specific implementations of metric learning depend on the input data for each network, as addressed in Section III-D. In particular, we define the metric learning loss in Equation 1 as

$$\mathcal{L}_{\text{metric}} = \begin{cases} \mathcal{D}(\text{embed}_1, \text{embed}_2) \\ \left|\left|\mathcal{D}(\text{embed}_1, \text{embed}_2) - \mathcal{W}_2^2(x_1, x_2)\right|\right|_2^2 \end{cases} \tag{4}$$

where the first line is the distance between two embeddings in the mutual space (note that this distance could in general be replaced by any other similarity metric such as L_n norm. In our experiments, the dot product distance worked best in terms of convergence speed and classification accuracy.), corresponding to the siamese input situation. The second line adds $\mathcal{W}_2^2$ Wasserstein distance for different inputs, and x_1, x_2 are two input images. This is the situation of different input.

The verification loss $\mathcal{L}_{\text{verification}}$ is designed to improve the quality of embeddings, which receives the embedding and maps them to the semantic label with a softmax layer. This is the key that enables our collaborative learning to be different from other KD-based methods, which are performed in the feature maps of the middle layer [22], [24], [34]. In particular, they encourage the middle feature maps of the student network to mimic the output of the teacher network. However, directly performing mimic learning is not reasonable without considering the whole feature map flow of the teacher network. In contrast, our verification loss ensures that the mutual space that we use for metric learning is a high-quality space, by directly linking the embeddings in the mutual space to the semantic label. In practice, we find verification part is essential since the performance of additional supervision depends highly on the quality of mutual space.

DEPLOYMENT LOCATION

Theoretically, MKB can be deployed in any location between two networks as long as it takes as input the feature maps of

TABLE 1. A typical architecture of collaborative learning with one MKB and feature map evolutions, using SmallNet [31] and ResNet-34 [4] on CIFAR-10 dataset.

	SmallNet				ResNet-32		
input layer	$32 \times 32 \times 3$				$32 \times 32 \times 3$		input layer
conv1_x	$\begin{bmatrix} 3 \times 3, & 32 \\ 3 \times 3, & 32 \end{bmatrix} \times 2$	32×32			$3 \times 3, 16$	32×32	conv0
pool1	16×16				$\begin{bmatrix} 3 \times 3, & 16 \\ 3 \times 3, & 16 \end{bmatrix} \times 5$	32×32	conv1_x
						16×16	avg_pool_1
conv2_x	$\begin{bmatrix} 3 \times 3, & 64 \\ 3 \times 3, & 64 \end{bmatrix} \times 2$	16×16			$\begin{bmatrix} 3 \times 3, & 32 \\ 3 \times 3, & 32 \end{bmatrix} \times 5$	16×16	conv2_x
pool1	8×8						
encoder	$3 \times 3, 64$	8×8			$3 \times 3, 64$ stride 2	8×8	encoder
space transfer			$3 \times 3, 64$ $3 \times 3, 64$	8×8 8×8			space transfer
decoder	$3 \times 3, 64$	8×8			$3 \times 3, 64$ stride 2	16×16	decoder
						8×8	avg_pool_2
conv3_x	$\begin{bmatrix} 3 \times 3, & 128 \\ 3 \times 3, & 128 \end{bmatrix} \times 2$	8×8			$\begin{bmatrix} 3 \times 3, & 64 \\ 3 \times 3, & 64 \end{bmatrix} \times 5$	8×8	conv3_x
pool3	4×4				64		global_pool
fc1	128				128		fc1
softmax	10				10		softmax

involved models and sends the outputs of corresponding size back to keep them end-to-end trainable. Experimental results suggest putting MKB in backward layers often produces better performance, which is perhaps because the mutual knowledge in high level is more useful. However, placing the MKB in the fully connected layers often results in worse performance. Fig. 1 shows a typical insertion of MKB in convolutional layer of two deep neural networks. A specific evolution of feature map can be seen in Table 1. In practice, we find that the MKB should be deployed in locations that ensure that the input feature maps have similar sizes.

One can set multiple MKBs with this criterion and we report such experimental result in Fig. 6. MKB enjoys characteristics of flexible deployment like recent proposed spatial transformers [46], which means it can be inserted to any two or multiple deep neural networks with the aforementioned deployment practices.

C. TRAINING STRATEGY

1) FEEDING DATA MANNERS

As discussed before, the metric learning loss depends on the way of preparing feed data, which can be implemented with two choices. One is using siamese input, where we send the same input to both networks and the first line in Equation 4 is utilized. The other is using different inputs, as it is also reasonable to feed different inputs to promote collaborative learning to mine mutual information about their relationship, as used in [21] and [42]. In this way, MKB is responsible to preserve the distance in the mutual space to mimic the distance in the original space, and the second line in Equation 4 is activated to calculate the metric loss.

2) TRAINING MANNERS

For the training procedure of our collaborative learning, we introduce two manner, i.e. simultaneous training and iterative training. Both are performed for each mini-batch throughout the whole training process. Specifically, in each training step, simultaneous training requires that the parameters of all the involved models be updated simultaneously, whereas iterative training iteratively updates the parameters of one model while keeping the parameters of the other fixed.

Fig. 3 shows a typical collaborative learning procedure schematically. In the training phase, the MKB is updated along with the two models. While in the testing phase, the MKB is separately used in each model.

D. EXTENSION TO MULTIPLE MODELS

Collaborative learning can be easily extended from two models to multiple models, by adjusting the metric learning part of MKB while keeping other structures unchanged. In particular, we use the metric loss for siamese input and define the improved version for multiple models as

$$\mathcal{L}^{i}_{\text{metric}} = \frac{1}{n-1} \sum_{i=1, i \neq j}^{n} \mathcal{D}(\text{embed}_i, \text{embed}_j) \tag{5}$$

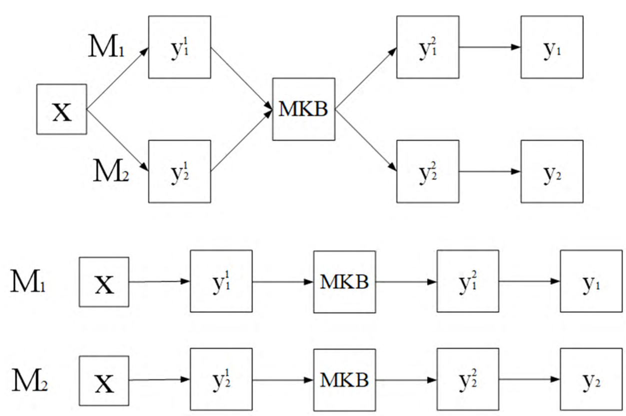

FIGURE 3. Training (up) and testing (down) stages in collaborative learning.

where n is the total number of the involved models in collaborative learning. The training procedure with multiple models is a straightforward extension of that for two models. In addition, it can be distributed by learning each network on one device and passing the small probability vectors between devices.

Instead of Equation 5 where all the involved models learn from each other, we can also make each one learn from all the others by regarding them a universal model. In this way, Equation 5 becomes

$$\mathcal{L}^{i}_{\text{metric}} = \mathcal{D}(\text{embed}_i, \text{embed}_{\text{others}}) \tag{6}$$

where $\text{embed}_{\text{others}} = \frac{1}{n-1}\sum_{j=1, j\neq i}^{n}(\text{embed}_j)$ is the averaged sum of all the other embeddings.

IV. EXPERIMENTS

A. DATASETS AND IMPLEMENTATION DETAILS

We consider three aspects to demonstrate the effectiveness of our collaborative learning. Section IV-B shows the two networks can obtain a fast optimization with collaborative learning. In addition, the two networks can be both improved, with superior performance to a series of relevant methods, including KD and its variants. as described in Section IV-C. Finally, Section IV-D demonstrates that state-of-the-art network can be improved, by collaborative learning with other popular networks. We evaluate our method on four popular datasets, with widely used settings on image classification task as follows.

- The MNIST dataset [47] consists of 60, 000 training images and 10, 000 test images, and each image is 28 × 28 belonging to a digit from 0 to 9. The pixel values are normalized to lie in the [0, 1] range. No other form of data pre-processing or augmentation is performed. The weights in each layer are initialized by sampling random values from $\mathcal{N}(0, 0.01)$, whereas the bias vectors are initialized to 0.
- The CIFAR-10 dataset [48] contains 50, 000 training images with 5, 000 images per class and 10, 000 test images with 1, 000 images per class. The CIFAR-10 dataset comprises 32 × 32 pixel RGB images with 10 classes. However, we padded 4 pixels on each side to make the image size 40 × 40 pixels. Randomly cropped 32 × 32 pixel images were used for training, and the original 32 × 32 pixel images were used for testing.
- The CIFAR-100 dataset [48] uses 50, 000 training images with 500 images per class and 10, 000 test images with 100 images per class. The CIFAR-100 dataset contains 32 × 32 pixel RGB images with 100 classes. We did not use augmentation methods unlike the CIFAR10 case to make the various experiment settings.
- The STL-10 dataset [49] contains 10 object classes with 5, 000 training and 8, 000 test images. There are 10 predefined folds of training images, with 500 images in each fold. In each fold, a classifier is trained on a specific set of 500 training images, and tested on all 8, 000 testing images. Similar to prior work, the evaluation metric we report is average accuracy across 10 folds. Due to its relatively large image size (96 × 96), we follow previous researches to downsample the images to 32 × 32.
- The ILSVRC 2012 classification dataset [50] consists 1.2 million images for training, and 50, 000 for validation, from 1, 000 classes. We adopt the same data augmentation scheme for training images as in [4] and [51] and apply a single-crop with size 224 × 224 at test time. Following [4], [5], [51], and [52], we report the top-1 and top-5 classification accuracy on the validation set.

The encoder of MKB is a series of convolutional layers that map two feature maps into mutual codes of unified size. The space transfer is stacked with two convolutional layers followed by a flatten layer, which outputs an embedding in the mutual space whose size is set as 1, 024. The decoder is a deconvolutional layer that maps the embedding back to the original network with Equation 3. Table 1 shows a typical situation of employing collaborative learning with one MKB, and the evolutions of feature maps with two deep neural networks. In metric learning, dot product distance and Wasserstein distance are implemented with recommended parameter choices [21], [31]. The coefficient α in Equation 1 is set 0.2 to encourage the learning process is dominated by the true label. As for training, results in Fig. 4 suggest that the optimal suite of collaborative learning is using siamese input and simultaneously training, where the learning rate and the weight decay are kept same for two networks. We use these common settings to implement collaborative learning to compared with other competitive works. Other hyperparameter choices are provided in the context of the following experiments.

For both individual training and collaborative learning, we used the same experimental settings to examine the effect of collaborative learning. All the networks are trained using stochastic gradient descent (SGD). We adopt batch normalization (BN) [10] immediately after each convolution and before activation. We initialize the weights as in [53] and train both networks from scratch. On MNIST, CIFAR-10/100 and STL-10, we train using mini-batch size 128 for 200 epochs, which contain approximately 85, 000 iterations. The learning rate starts from 0.1 and is divided by 10 when the error plateaus, following [4] and [53]. On ImageNet, we train using mini-batch size 256 for 300 epochs, which contain around 1, 400, 000 iterations. The learning rate starts from 0.1 and is divided by 10 when the error plateaus, following [4] and [5]. We use a weight decay of 0.0001 and a momentum of 0.9. Except the ImageNet dataset, we do not use dropout, following the practice in [10].

B. FAST OPTIMIZATION

Compared with individual training of each network, collaborative learning can assist the involved networks to obtain a fast convergence. In this section, we verify this advantage using ResNet-34 and MobileNet, and conduct experiments on CIFAR-10 dataset.

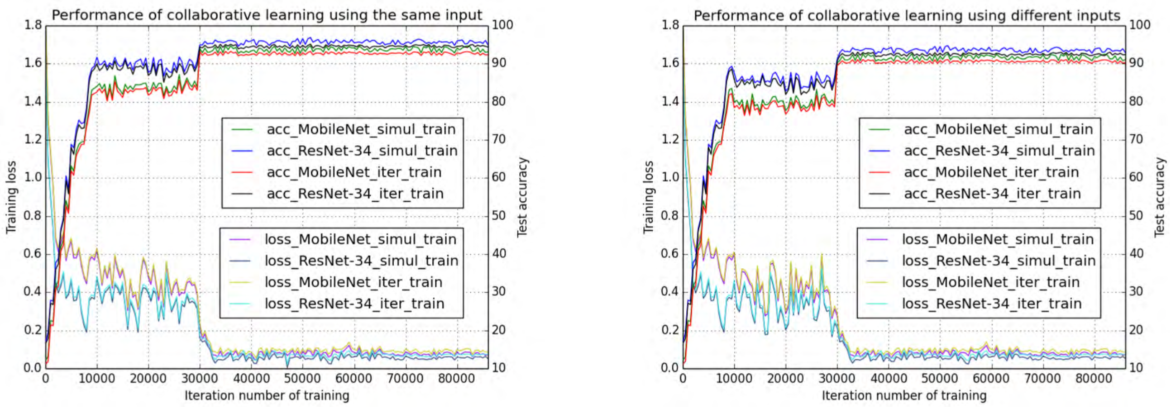

FIGURE 4. Effect of the input manners (left:S; right:D) and training manners using ResNet-34 and MobileNet on CIFAR-10 dataset.

Fig. 5 represents the change of training loss and test accuracy over time. From the convergence procedure, we can observe that the two network can be faster converged (reaching the saturation region) via collaborative learning. Specifically, using collaborative learning, ResNet-20 and MobileNet reach the saturation region at around 30, 000 and 35, 000 iterations. When training these two networks individually, the convergence for both networks is equivalent to approximately 60,000 iterations. This indicates that collaborative learning provides useful mutual knowledge that can assist each other for fast convergence. On the other hand, it can be also noticed that the test accuracy of the two networks is also improved with collaborative learning, by 5.5% and 6.7%, respectively. This part is further examined in the next section with more experiments and comparisons.

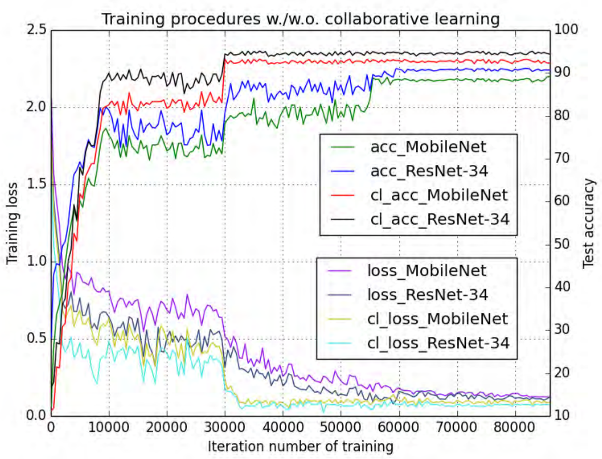

FIGURE 5. Convergence procedure with/without collaborative learning using ResNet-34 and MobileNet on CIFAR-10 dataset.

We also notice that there are several works that used a distilled knowledge from a large network to assist the training procedure of a small network, to obtain a faster convergence, such as [22] and [34]. However, the teacher network cannot be improved in terms of both optimization speed and classification accuracy. In our method, both the involved networks can enjoy fast optimization via collaborative learning, and this assistance is bidirectional.

C. BIDIRECTIONAL PERFORMANCE IMPROVEMENT

In this section, we examine the performance improvement of the two networks via collaborative learning, compared with other methods. We use popular deep neural networks, including SmallNet [31], ResNet-34 [4] and MobielNet [54] for two model evaluation. When using collaborative learning for multiple networks, we also add GoogLeNet [55] and VGG-16 [3]. For comprehensive comparison, we consider the following methods:

- Individual learning (IL): Training each network individually
- Ensemble learning (EL): Combining the final predictions of each networks, followed by a trainable fully connected layer linked to semantic label
- Distillation learning (DL $1 \rightarrow 2$) [17]: Using well trained network 1 to help network 2 with KL divergence as "soft target" in the final predictions
- Hint learning (HL $1 \rightarrow 2$) [34]: Using l_2 loss to match the intermediate feature maps of two networks in the same position as MKB
- Deep mutual learning (DML) [44]: Forcing two networks to produce similar final predictions in the semantic space in terms of KL divergence
- Wasserstein embedding learning (WEL) [21]: Forcing two networks to produce similar embeddings with Wasserstein distance in the original space
- MKB Structure without CL (MKB w/o CL): Each half of MKB is combined with each network and updated individually without metric learning and verification

1) TWO NETWORKS

Tables 2 3 4 report the performance of model assistance using two deep neural networks. The results show that simply using ensemble model of two deep neural networks (EL)

TABLE 2. Test accuracy (%) of ResNet-34 and MobileNet on four datasets.

Method	MNIST	CIFAR-10	CIFAR-100	STL-10
IL	99.60 / 99.58	91.82 / 92.47	68.99 / 73.65	70.42 / 27.61
EL	99.54	91.98	71.45	70.94
DL(1 → 2)	99.60 / 99.55	91.82 / 92.54	68.99 / 74.43	70.42 / 73.35
HL(1 → 2)	99.60 / 99.51	91.82 / 92.37	68.99 / 72.74	70.42 / 73.06
DML	99.55 / 99.57	91.98 / 92.59	71.10 / **76.13**	71.53 / 73.77
WEL	99.48 / 99.39	91.87 / 92.31	69.48 / 72.75	69.74 / 72.47
MKB (w/o CL)	99.59 / 99.57	91.95 / 92.54	69.16 / 73.52	71.43 / 72.11
CL (Ours)	**99.74 / 99.71**	**94.79 / 93.57**	**74.83** / 74.75	**74.84 / 74.76**

TABLE 3. Test accuracy (%) of ResNet-34 and SmallNet on four datasets.

Method	MNIST	CIFAR-10	CIFAR-100	STL-10
IL	99.60 / 99.39	91.82 / 87.42	68.99 / 64.77	70.42 / 66.48
EL	99.48	88.69	65.26	68.52
DL(1 → 2)	99.60 / 99.52	91.82 / 89.77	68.99 / 67.43	70.42 / 68.98
HL(1 → 2)	99.60 / 99.48	91.82 / 88.86	68.99 / 67.08	70.42 / 67.74
DML	**99.71** / 99.51	91.02 / 89.65	68.95 / 68.41	69.76 / 69.84
WEL	99.63 / 99.44	90.74 / 87.74	69.44 / 65.02	68.75 / 66.73
MKB (w/o CL)	99.62 / 99.59	91.47 / 89.08	69.98 / 65.81	71.45 / 66.53
CL (Ours)	99.69 / **99.69**	**93.16 / 90.92**	**70.14 / 68.53**	**71.87 / 68.98**

TABLE 4. Test accuracy (%) of MobileNet and SmallNet on four datasets.

Method	MNIST	CIFAR-10	CIFAR-100	STL-10
IL	99.58 / 99.39	92.47 / 87.42	73.65 / 64.77	72.39 / 66.48
EL	99.46	88.74	68.71	68.24
DL(1 → 2)	99.58 / 99.32	92.47 / 89.65	73.65 / 66.74	72.39 / 68.95
HL(1 → 2)	99.58 / 99.27	92.47 / 88.35	73.65 / 66.56	72.39 / 67.99
DML	99.58 / 99.44	91.76 / 89.97	72.02 / 67.05	71.94 / 67.99
WEL	99.39 / 99.34	90.45 / 88.79	68.95 / 66.82	70.34 / 65.75
MKB (w/o CL)	99.44 / 99.40	91.05 / 88.75	72.15 / 66.87	71.59 / 67.91
CL (Ours)	**99.67 / 99.65**	**93.79 / 91.38**	**74.31 / 67.78**	**74.75 / 70.65**

often provides limited improvement compared with IL since each network is carefully designed and competitive. In addition, the average of other works including ours outperforms EL, which also demonstrates the limitation of EL. When compared with KD-based methods including DL and HL, we obtain an improvement of approximately 5%. Our method also outperforms two recent works DML and WEL in most cases. Note that the aforementioned four works provide only unidirectional assistance, where the performance of the teacher models cannot be improved. However, with our collaborative learning, both the associated networks can obtain performance improvement. In addition, we replace the collaborative learning process and keep the encoder-decoder structure unchanged (MKB w/o CL) and see a clear performance drop which indicates the importance of collaborative learning.

2) ANALYSIS OF MKB

We examine the position of MKB in both convolutional layers and fully connected layers to see how it affects the performance of collaborative learning. We use the architecture described in Table 1 on CIFAR-10 dataset. The four positions that we adopt include conv1_x, conv2_x, conv3_x and fc1. Table 5 shows the performance comparison. The comparison suggests that the deploying MKB on convolutional layers often provides better results. In particular, conv2_x is the best choice of the architecture in Table 1.

TABLE 5. Performance comparison of different MKB positions in Table 1.

Method	MNIST	CIFAR-10	CIFAR-100	STL-10
conv_1x	99.60 / 99.43	92.67 / 87.41	73.67 / 64.77	70.39 / 66.68
conv_2x	**99.69 / 99.45**	**93.51 / 89.99**	68.71 / **67.97**	**72.74 / 69.88**
conv_3x	99.58 / 99.41	92.98 / 88.25	**74.32** / 64.44	71.07 / 68.25
fc1	99.55 / 99.38	92.59 / 87.55	73.65 / 63.46	69.66 / 67.92

We also examine the evolution of MKB in the learning process to see how it helps collaborative learning. To this end, we stop updating MKB in different stages of all the training process and check the final performance using current learned information in MKB. Fig. 6 (left) shows that with the evolution of updating MKB, the learning ability of each involved model increases, which indicates the effectiveness of collaborative learning.

Previously, we have shown inserting a single MKB with collaborative learning can provide a performance improvement. We further study whether adding more MKBs can provide help for collaborative learning. Results in Fig. 6 (right) indicate there is a positive potential to achieve further improvements with more MKBs. When scaling to multiple models, we compare with methods that can be applied to multiple models, including IL, EL and DML. Fig. 7 summarizes the results on CIFAR-10 dataset, where we can observe that our method is consistently better than other methods with various network numbers. This indicates that the involved networks can learn from each other with collaborative learning. The two yellow lines in both left and right of Fig. 7 are two manners described in Section III-D. The results suggest Equation 6 produces a better performance for multiple model assistance.

D. PERFORMANCE IMPROVEMENT OF THE STATE-OF-THE-ART MODEL

In this section, we demonstrate that state-of-the-art models can be improved via collaborative learning with other models. We choose DenseNet, a popular state-of-the-art architecture, as one model and use several other advanced models to act as the other one in collaborative learning. We choose three popular networks to conduce collaborative learning with DenseNet, including Maxout [25], network in network (NIN) [26] and deeply-supervised net (DSN) [27] for CIFAR-10, and GoogLeNet, VGG-16 and ResNet-34 for ImageNet.

Table 6 shows the performance on CIFAR-10 dataset. As provided in the original paper [5], the DenseNet-40 yields a test accuracy of 93.00 on CIFAR-10. The performance of training the other three networks individually can obtain 90.62, 91.19, and 91.78, respectively. While using collaborative learning, we see a clear improvement of both DenseNet and other three networks in the right column. In particular, the most significant improvement of DenseNet-40 is obtained with DSN-34, of approximately 1.43%. In addition, the other three networks is also able to enjoy performance improvement.

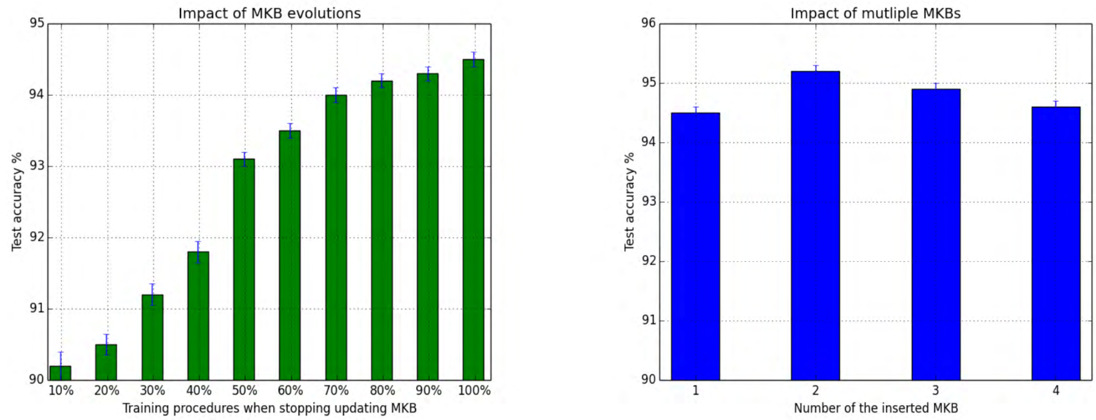

FIGURE 6. MKB analysis on CIFAR-10 dataset. Left: impact of different evolutions of MKB. Right: impact of the number of MKBs.

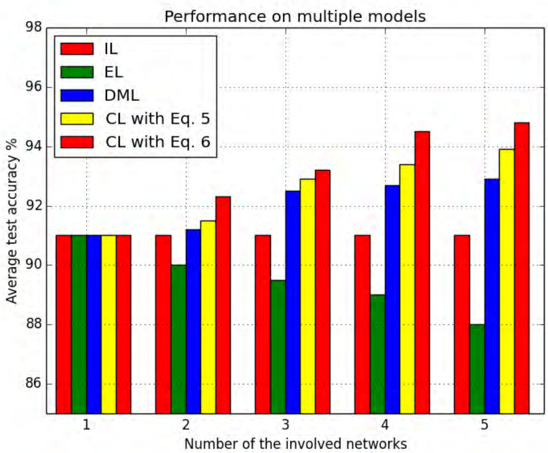

FIGURE 7. Collaborative learning for multiple models on CIFAR-10 dataset.

TABLE 6. Performance improvement (test accuracy) using the state-of-the-art model DenseNet-40 with other models on CIFAR-10 dataset.

DenseNet-40 (S) 93.00	IL	CL		Improvement	
		S	A/B/C	ΔS	Δ A/B/C
Maxout (A)	90.62	94.11	92.07	**+1.19%**	1.60%
NIN (B)	91.19	94.18	92.75	**+1.27%**	1.71%
DSN (C)	91.78	94.33	92.44	**+1.43%**	0.72%

TABLE 7. Performance improvement (top-1 and top-5 accuracy; single-model and 10-crop) using the state-of-the-art model (DenseNet-121) with other models on ImageNet validation set.

DenseNet-121 (S) 74.98 (top-1) 92.29 (top-5)	IL	CL		Improvement	
		S	A/B/C	ΔS	Δ A/B/C
GoogLeNet (A)	70.84 91.85	76.20 94.66	72.49 91.96	**+1.63%** **+2.57%**	2.33% 0.12%
VGG-16 (B)	71.93 90.67	77.27 94.18	74.93 91.49	**+3.05%** **+2.05%**	4.17% 0.90%
ResNet-34 (C)	71.46 89.98	77.82 95.44	73.74 91.93	**+3.79%** **+3.41%**	3.19% 2.17%

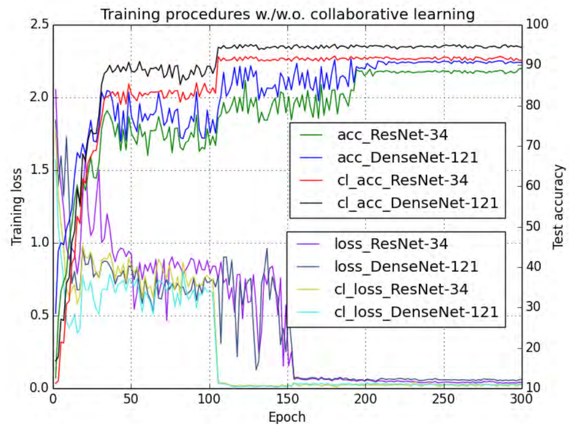

FIGURE 8. Analysis of the optimization speed and the test accuracy (top-5) using DensNet-121 and ResNet-34 on ImageNet validation set.

Table 7 shows the performance on ImageNet validation set. The results are similar to that of CIFAR-10. The average improvement of DenseNet-121 can obtain around 2.82% and 2.67%, in terms of top-1 and top-5 accuracy, respectively. Considering that DenseNet is currently the reported state-of-the-art network architecture, these results show that collaborative learning is a efficient and powerful solution to collaborate advanced deep networks to achieve bidirectional assistance.

Fig. 8 gives an example of the convergence procedure and the test accuracy change on ImageNet validation set, using DenseNet-121 and ResNet-34. The green and blue lines indicate the individually training procedures of the two networks, the red and black lines are the cases of collaborative learning. We can observe that, in the situation of the state-of-the-art model, the conclusions are similar to that of previous experiments in Section IV-B, i.e., collaborative learning can provide fast optimization and bidirectional assistance.

V. CONCLUSION

In this paper, we have present a collaborative learning method to achieve bidirectional model assistance. We show how to construct the mutual space to mine the additional supervision with MKB to assist each other. Compared with a series of KD-based methods, collaborative learning is performed in a peer to peer manner and enjoys bidirectional assistance.

Besides, it is more flexible and needs no specific requirements about the involved models such as model ability difference and pre-training. Experimental results demonstrate that collaborative learning can efficiently improve the learning ability and convergence speed of the two models, with superior performance to a series of associated works. More importantly, we show that state-of-the-art models such as DenseNet can be greatly improved using CL with other popular models.

REFERENCES

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012, pp. 1097–1105.
[3] K. Simonyan and A. Zisserman. (2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556
[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, Jun. 2016, pp. 770–778.
[5] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. CVPR*, Jun. 2017, pp. 4700–4708.
[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[7] S. Antol *et al.*, "VQA: Visual question answering," in *Proc. ICCV*, 2015, pp. 2425–2433.
[8] D. Amodei *et al.*, "Deep Speech 2: End-to-end speech recognition in english and mandarin," in *Proc. ICML*, 2016, pp. 1–10.
[9] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
[10] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. ICML*, 2015, pp. 1–9.
[11] D. Kingma and J. Ba. (2014). "Adam: A method for stochastic optimization." [Online]. Available: https://arxiv.org/abs/1412.6980
[12] R. Polikar, "Ensemble learning," in *Ensemble Machine Learning*. Boston, MA, USA: Springer, 2012, pp. 1–34.
[13] J. Cao, Z. Lin, G.-B. Huang, and N. Liu, "Voting based extreme learning machine," *Inf. Sci.*, vol. 185, no. 1, pp. 66–77, 2012.
[14] J. Wang, W. Wang, X. Chen, R. Wang, and W. Gao, "Deep alternative neural network: Exploring contexts as early as possible for action recognition," in *Proc. NIPS*, 2016, pp. 1–9.
[15] H. Song, W. Wang, J. Wang, and R. Wang, "Collaborative deep networks for pedestrian detection," in *Proc. BigMM*, Apr. 2017, pp. 146–153.
[16] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.
[17] G. Hinton, O. Vinyals, and J. Dean. (2015). "Distilling the knowledge in a neural network." [Online]. Available: https://arxiv.org/abs/1503.02531?context=cs
[18] B. Krawczyk, L. L. Minkub, J. Gama, J. Stefanowski, and M. Woźniak, "Ensemble learning for data stream analysis: A survey," *Inf. Fusion*, vol. 37, pp. 132–156, Sep. 2017.
[19] N. Frosst and G. Hinton. (2017). "Distilling a neural network into a soft decision tree." [Online]. Available: https://arxiv.org/abs/1711.09784
[20] W. M. Czarnecki, S. Osindero, M. Jaderberg, G. Swirszcz, and R. Pascanu, "Sobolev training for neural networks," in *Proc. NIPS*, 2017, pp. 4278–4287.
[21] N. Courty, R. Flamary, and M. Ducoffe. (2017). "Learning Wasserstein embeddings." [Online]. Available: https://arxiv.org/abs/1710.07457
[22] J. Yim, D. Joo, J. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *Proc. CVPR*, Jul. 2017, pp. 7130–7138.
[23] P. Luc, N. Neverova, C. Couprie, J. Verbeek, and Y. LeCun, "Predicting deeper into the future of semantic segmentation," in *Proc. ICCV*, 2017, pp. 648–657.
[24] G. Chen, W. Choi, X. Yu, T. Han, and M. Chandraker, "Learning efficient object detection models with knowledge distillation," in *Proc. NIPS*, 2017, pp. 1–10.
[25] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. (2013). "Maxout networks." [Online]. Available: https://arxiv.org/abs/1302.4389
[26] M. Lin, Q. Chen, and S. Yan. (2013). "Network in network." [Online]. Available: https://arxiv.org/abs/1312.4400
[27] C.-Y. Lee, S. Xie, P. W. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *Proc. AISTATS*, 2015, pp. 562–570.
[28] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *Int. J. Comput. Vis.*, vol. 104, no. 2, pp. 154–171, Apr. 2013.
[29] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. CVPR*, Jun. 2014, pp. 580–587.
[30] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," in *Proc. NIPS*, 2015, pp. 91–99.
[31] P. Haeusser, A. Mordvintsev, and D. Cremers, "Learning by association: A versatile semi-supervised training method for neural networks," in *Proc. CVPR*, Jun. 2017, pp. 89–98.
[32] T. G. Dietterich, "Ensemble methods in machine learning," in *Proc. 1st Int. Workshop Multiple Classifier Syst.*, vol. 1, 2000, pp. 1–15.
[33] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.
[34] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. (2014). "FitNets: Hints for thin deep nets." [Online]. Available: https://arxiv.org/abs/1412.6550
[35] X. Zhang *et al.* (2017). "Alignedreid: Surpassing human-level performance in person re-identification." [Online]. Available: https://arxiv.org/abs/1711.08184
[36] A. J. C. Sharkey, *Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems*. New York, NY, USA: Springer, 2012.
[37] R. Caruana, "Multitask learning," in *Learning to Learn*, 1998, pp. 95–133.
[38] W. Ouyang and X. Wang, "Joint deep learning for pedestrian detection," in *Proc. ICCV*, Dec. 2013, pp. 2056–2063.
[39] Y. Bengio, "Learning deep architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.
[40] C. Su, F. Yang, S. Zhang, Q. Tian, L. S. Davis, and W. Gao, "Multi-task learning with low rank attribute embedding for person re-identification," in *Proc. ICCV*, Dec. 2015, pp. 3739–3747.
[41] B. Kulis, "Metric learning: A survey," *Found. Trends Mach. Learn.*, vol. 5, no. 4, pp. 287–364, 2013.
[42] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Proc. CVPR*, Jun. 2005, pp. 539–546.
[43] J. Wang *et al.*, "Learning fine-grained image similarity with deep ranking," in *Proc. CVPR*, Jun. 2014, pp. 1386–1393.
[44] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu. (2017). "Deep mutual learning." [Online]. Available: https://arxiv.org/abs/1706.00384
[45] F. Wang *et al.*, "Residual attention network for image classification," in *Proc. CVPR*, Jun. 2017, pp. 3156–3164.
[46] M. Jaderberg, K. Simonyanb, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," in *Proc. NIPS*, 2015, pp. 1–9.
[47] Y. LeCun, C. Cortes, and C. J. C. Burges. (1998). *The MNIST Database of Handwritten Digits*. [Online]. Available: http://yann.lecun.com/exdb/mnist/
[48] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009, vol. 1, no. 4, p. 7.
[49] A. Coates, A. Y. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proc. AISTATS*, 2011, pp. 215–223.
[50] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. CVPR*, Jun. 2009, pp. 248–255.
[51] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. ECCV*, 2016, pp. 630–645.
[52] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *Proc. ECCV*, 2016, pp. 646–661.
[53] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imageNet classification," in *Proc. ICCV*, 2015, pp. 1026–1034.
[54] A. G. Howard *et al.* (2017). "MobileNets: Efficient convolutional neural networks for mobile vision applications." [Online]. Available: https://arxiv.org/abs/1704.04861
[55] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. CVPR*, Jun. 2015, pp. 1–9.

JINZHUO WANG received the B.S. degree from the School of Electronic Engineering and Computer Science, Peking University, in 2013, where he is currently pursuing the Ph.D. degree. He has published several papers on relevant conferences and journals, such as NIPS, ACM Multimedia, AAAI, ICME, ICIP, and the IEEE TRANSACTIONS ON MULTIMEDIA. His research interests include computer vision and deep learning.

WENMIN WANG (M'17) received the Ph.D. degree in computer architecture from the Harbin Institute of Technology, China, in 1989. He was an Assistant Professor and an Associate Professor with the Harbin University of Science and Technology and the Harbin Institute of Technology. Since 1992, he has gained about 18 years of oversea industrial experiences in Japan and America, where he served as a Staff Engineer, a Chief Engineer, and the General Manager of the Software Division. He came back the Academia of China in 2009, and is currently a Professor with the School of Electronic and Computer Engineering, Peking University, China. His current research interests include video computer vision, multimedia retrieval, artificial intelligence, and machine learning.

WEN GAO (M'92–SM'05–F'09) received the Ph.D. degree in electronics engineering from The University of Tokyo, Japan, in 1991. He was a Professor of computer science with the Harbin Institute of Technology from 1991 to 1995 and a Professor with the Institute of Computing Technology, Chinese Academy of Sciences. He is currently a Professor of computer science with Peking University, China.

He has published extensively including five books and over 600 technical articles in refereed journals and conference proceedings in the areas of image processing, video coding and communication, pattern recognition, multimedia information retrieval, multimodal interface, and bioinformatics. He chaired a number of prestigious international conferences on multimedia and video signal processing, such as the IEEE ICME and ACM Multimedia. He also served on the advisory and technical committees of numerous professional organizations. He served on the Editorial Board for several journals, such as the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, the IEEE TRANSACTIONS ON MULTIMEDIA, the IEEE TRANSACTIONS ON AUTONOMOUS MENTAL DEVELOPMENT, the *EURASIP Journal of Image Communications*, and the *Journal of Visual Communication and Image Representation*.