Knowledge Transfer via Dense Cross-Layer Mutual-Distillation

Anbang Yao*⊠ and Dawei Sun*

Intel Labs China
{anbang.yao,dawei.sun}@intel.com

Abstract. Knowledge Distillation (KD) based methods adopt the one-way Knowledge Transfer (KT) scheme in which training a lower-capacity student network is guided by a pre-trained high-capacity teacher network. Recently, Deep Mutual Learning (DML) presented a two-way KT strategy, showing that the student network can be also helpful to improve the teacher network. In this paper, we propose Dense Cross-layer Mutual-distillation (DCM), an improved two-way KT method in which the teacher and student networks are trained collaboratively from scratch. To augment knowledge representation learning, well-designed auxiliary classifiers are added to certain hidden layers of both teacher and student networks. To boost KT performance, we introduce dense bidirectional KD operations between the layers appended with classifiers. After training, all auxiliary classifiers are discarded, and thus there are no extra parameters introduced to final models. We test our method on a variety of KT tasks, showing its superiorities over related methods. Code is available at https://github.com/sundw2014/DCM.

Keywords: Knowledge Distillation, Deep Supervision, Convolutional Neural Network, Image Classification

1 Introduction

In recent years, deep Convolutional Neural Networks (CNNs) have achieved remarkable success in many computer vision tasks such as image classification [24], object detection [9] and semantic segmentation [32]. However, along with the rapid advances on CNN architecture design, top-performing models [40,44,13,52,19,48,17] also pose intensive memory, compute and power costs, which limits their use in real applications, especially on resource-constrained devices.

To address this dilemma, Knowledge Transfer (KT) attracts great attentions among existing research efforts [43]. KT is typically treated as a problem of transferring learnt information from one neural network model to another. The first attempt of using KT to cope with model compression was made in [5]

* Equal contribution. ⊠ Corresponding author. Experiments were mostly done by Dawei Sun when he was an intern at Intel Labs China, supervised by Anbang Yao.

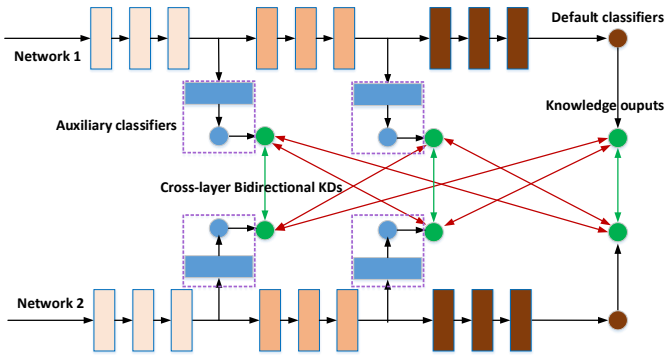

Fig. 1. Structure overview of the proposed method. For illustration, auxiliary classifiers are added to two hidden layers of each network, which are removed after training. Green/red arrows denote bidirectional knowledge distillation operations between the same-staged/different-staged layers of two networks. Best viewed in color.

where Bucilă et al. used an ensemble of neural networks trained on a small annotated dataset to label a much larger unlabelled dataset. By doing this, a smaller and faster neural network can be trained using many more labeled samples, and thus the final model can have much better performance than that is trained solely on the original small dataset. [2] extended this idea to train a shallower and wider neural network. Hinton et al. advanced knowledge transfer research by introducing the well-known Knowledge Distillation (KD) [14] method adopting a teacher-student framework. In KD, a lower-capacity student network is enforced to mimic the probabilistic outputs by a pre-trained high-capacity teacher network as well as the one-hot ground-truth labels. Naturally, the teacher can also be an ensemble of multiple models. Since then, numerous KD variants [37,51,28,53,21,42,45] have been proposed, mostly focusing on using either feature maps or attention maps at the intermediate layers of the teacher network as the extra hints for improving KD designs. Following [14], these methods adopted the same teacher-student framework in which the teacher network is trained beforehand and fixed under the assumption that it always learns a better representation than the student network. Consequently, they all used the one-way KT strategy, where knowledge can only be transferred from a teacher network to a student network. Recently, Deep Mutual Learning (DML) [57] was proposed, which achieved superior performance by a powerful two-way KT design, showing that the probabilistic outputs from the last layer of both teacher and student networks can be beneficial to each other.

In this paper, we restrict our focus to advance two-way KT research in the perspective of promoting knowledge representation learning and transfer design. Dense Cross-layer Mutual-distillation (DCM), an improved two-way KT method which is capable of collaboratively training the teacher and student networks from scratch, is the main contribution of this paper. Fig. 1 shows the structure overview of DCM. Following the deep supervision methodology [27,44,18,42], we first add well-designed auxiliary classifiers to certain hidden layers of both teach-

er and student networks, allowing DCM to capture probabilistic predications not only from the last layer but also from the hidden layers of each network. To the best of our knowledge, deep supervision design is overlooked in the knowledge transfer field. Furthermore, we present dense cross-layer bidirectional KD operations to regularize the joint training of the teacher and student networks. On the one hand, knowledge is mutually transferred between the same-staged supervised layers. On the other hand, we find the bidirectional KD operations between the different-staged supervised layers can further improve KT performance, thanks to the well-designed auxiliary classifiers which alleviate semantic gaps of the knowledge learnt at different-staged layers of two networks. Note that there are no extra parameters added to final models as all auxiliary classifiers are discarded after training. Experiments are performed on image classification datasets with a variety of KT settings. Results show that our method outperforms related methods by noticeable margins, validating the importance of connecting knowledge representation learning with bidirectional KT design.

2 Related Work

In this section, we briefly summarize existing works related to our method.

Knowledge Distillation Applications. Although KD based methods were primarily proposed for model compression [14,37,51,28,53,21,45,26], there have been many attempts to extend them to other tasks recently. Two representative examples are lifelong learning and multi-modal visual recognition. In lifelong learning task, the combination of KD and other techniques such as fine-tuning and retrospection was applied, targeting to adapt a pre-trained model to new tasks while preserving the knowledge gained on old tasks [30,15,54]. When designing and training multiple-stream networks dedicated to action recognition [8], person re-identification [11], depth estimation and scene parsing [10,50], cross-modal distillation was used to facilitate the knowledge transfer between the network branches trained on different sources of data such as RGB and depth. Other KD extensions include but are not limited to efficient network design [6,47], style transfer [29], machine translation [22,1] and multi-task learning [25]. Our method differs from these approaches in task and formulation.

Co-Training. Blum and Mitchell proposed a pioneering co-training framework [4] in which two models were trained separately on each of two views of labeled data first, and then more unlabelled samples as well as the predictions by each trained model were used to enlarge training data size. Recently, several deep co-training schemes have been proposed, mostly following the semi-supervised learning paradigm. [36] extended the idea of [4] via presenting a deep adversarial co-training method that uses adversarial samples to prevent multiple neural networks trained on different views from collapsing into each other. [3] proposed a cooperative learning mechanism in which two agents handling the same visual recognition task can transfer their current knowledge learnt on different sources of data to each other. [12] addressed multi-task machine translation problem with a dual co-training mechanism. [41] considered the co-training of several

classifier heads of the same network. Unlike these methods, we aim to improve the two-way knowledge transfer design for supervised image classification task.

Deep Supervision. The basic idea of deep supervision is to add extra classifiers to the hidden layers of a deep CNN architecture, which will be removed after training. It was originally proposed in [44,27] to combat convergence issue when designing and training deep CNNs for image recognition task. Since then, it has been widely used in training deep CNN architectures specially designed to handle other visual recognition tasks such as edge detection [49], object detection [31,20], semantic segmentation [59,58], human pose estimation [33,7] and anytime recognition [18,35]. In this paper, we extend the idea of deep supervision to promote the two-way knowledge transfer research.

3 Proposed Method

In this section, we detail the formulation and implementation of our method.

3.1 KD and DML

We first review the formulations of Knowledge Distillation (KD) [14] and Deep Mutual Learning (DML) [57]. For simplicity, we follow the teacher-student framework, and only consider the very basic case where there are one single teacher network and one single student network. Given the training data $X = \{x_n\}_{n=1}^N$ consisting of N samples collected from M image classes, the ground-truth labels are denoted as $Y = \{y_n\}_{n=1}^N$. Let W_t be a teacher network trained beforehand and fixed, and let W_s be a student network. In KD, the student network W_s is trained by minimizing

$$L_s = L_c(W_s, X, Y) + \lambda L_{kd}(\hat{P}_t, \hat{P}_s), \tag{1}$$

where L_c is the classification loss between the predications of the student network and the one-hot ground-truth labels, L_{kd} is the distillation loss, λ is a coefficient balancing these two loss terms. In [14], L_{kd} is defined as

$$L_{kd}(\hat{P}_t, \hat{P}_s) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{m=1}^{M}\hat{P}_t^m(x_n)\log\hat{P}_s^m(x_n). \tag{2}$$

Given a training sample x_n, its probability of image class m is computed as

$$\hat{P}^m(x_n) = \frac{exp(z_n^m/T)}{\sum_{m=1}^{M} exp(z_n^m/T)}, \tag{3}$$

where z_n^m is the output logit for image class m obtained from the last layer (i.e., default classifier) of a neural network, and T is a temperature used to soften the probabilistic outputs. The distillation loss defined by Eq. 2 can be considered as a modified cross-entropy loss using the probabilistic outputs of the pre-trained teacher network as the soft labels instead of the one-hot ground-truth labels.

Now it is clear that KD encourages the student network to match the probabilistic outputs of the pre-trained teacher model via a one-way Knowledge Transfer (KT) scheme. *Two key factors to KD based methods are: the representation of knowledge and the strategy of knowledge transfer*. DML considers the latter one by presenting a two-way KT strategy in which the probabilistic outputs from both teacher and student networks can be used to guide the training of each other. DML can be viewed as a bidirectional KD method that jointly trains the teacher and student networks via interleavingly optimizing two objectives:

$$\begin{aligned} L_s &= L_c(W_s, X, Y) + \lambda L_{dml}(\hat{P}_t, \hat{P}_s) \\ L_t &= L_c(W_t, X, Y) + \lambda L_{dml}(\hat{P}_s, \hat{P}_t). \end{aligned} \tag{4}$$

Here, λ is set to 1 and fixed [57]. As for the definition of the distillation loss L_{dml}, instead of using Eq. 2, DML uses Kullback-Leibler divergence:

$$L_{dml}(\hat{P}_t, \hat{P}_s) = \frac{1}{N}\sum_{n=1}^{N}\sum_{m=1}^{M} \hat{P}_t^m(x_n) \log \frac{\hat{P}_t^m(x_n)}{\hat{P}_s^m(x_n)}. \tag{5}$$

The $\hat{P}^m(x_n)$ is the same as that in KD. KL divergence is equivalent to cross entropy from the perspective of gradients calculation. Unlike KD containing two separate training phases, DML can jointly train the teacher and student networks in an end-to-end manner, and shows much better performance. This is attributed to the two-way KT strategy. However, the information contained in the hidden layers of networks has not been explored by DML. Moreover, the problem of connecting more effective knowledge representation learning with bidirectional KT design has also not been studied by DML.

3.2 Dense Cross-layer Mutual-distillation

Our DCM promotes DML via jointly addressing two issues discussed above.

Knowledge representation learning with deep supervision. Ideally, the knowledge should contain rich and complementary information learnt by a network and can be easily understood by the other network. Recall that many KD variants [37,51,28,53,21,46] have validated that feature maps or attention maps extracted at the hidden layers of a pre-trained teacher network are beneficial to improve the training of a student network under the premise of using the one-way KT scheme. Being a two-way KT method, instead of using either intermediate feature maps or attention maps extracted in an unsupervised manner as the additional knowledge, our DCM adds relevant auxiliary classifiers to certain hidden layers of both teacher and student networks, aggregating probabilistic knowledge not only from the last layer but also from the hidden layers of each network. This is also inspired by the deep supervision methodology [44,27] which is overlooked in the knowledge transfer research. As showed in our experiments and [18,58,35,42], even adding well-designed auxiliary classifiers to the hidden layers of a modern CNN can only bring marginal or no accuracy improvement. This motivates us to present a more elaborate bidirectional KT strategy.

Cross-layer bidirectional KD. With default and well-designed auxiliary classifiers, rich probabilistic outputs learnt at the last and hidden layers of both teacher and student networks can be aggregated on the fly during the joint training. Moreover, these probabilistic outputs are in the same semantic space, and thus our DCM introduces dense cross-layer bidirectional KD operations to promote the two-way KT process, which are illustrated in Fig. 1.

Formulation. In the following, we detail the formulation of DCM. We follow the notations in the last sub-section. Let $Q = \{(t_k, s_k)\}_{k=1}^{K}$ be a set containing K pairs of the same-staged layer indices of the teacher network W_t and the student network W_s, indicating the locations where auxiliary classifiers are added. Let (t_{K+1}, s_{K+1}) be the last layer indices of W_t and W_s, indicating the locations of default classifier. DCM simultaneously minimizes the following two objectives:

$$\begin{aligned} L_s &= L_c(W_s, X, Y) + \alpha L_{ds}(W_s, X, Y) + \beta L_{dcm_1}(\hat{P}_t, \hat{P}_s) + \gamma L_{dcm_2}(\hat{P}_t, \hat{P}_s) \\ L_t &= L_c(W_t, X, Y) + \alpha L_{ds}(W_t, X, Y) + \beta L_{dcm_1}(\hat{P}_s, \hat{P}_t) + \gamma L_{dcm_2}(\hat{P}_s, \hat{P}_t), \end{aligned} \tag{6}$$

where L_s/L_t is the loss of the student/teacher network. In this paper, we set α, β, γ and T to 1 and keep them fixed owing to easy implementation and satisfied results (*In fact, we tried the tedious manual tuning of these parameters, but just got marginal extra gains compared to this uniform setting*). Note that the teacher and student networks have the same loss definition. For simplicity, we take L_s as the reference and detail its definition in the following description. In L_s, L_c denotes the default loss which is the same as that in KD and DML. L_{ds} denotes the total cross-entropy loss over all auxiliary classifiers added to the different-staged layers of the student network, which is computed as

$$L_{ds}(W_s, X, Y) = \sum_{k=1}^{K} L_c(W_{s_k}, X, Y). \tag{7}$$

L_{dcm_1} denotes the total loss of the same-staged bidirectional KD operations, which is defined as

$$L_{dcm_1}(\hat{P}_t, \hat{P}_s) = \sum_{k=1}^{K+1} L_{kd}(\hat{P}_{t_k}, \hat{P}_{s_k}). \tag{8}$$

L_{dcm_2} denotes the total loss of the different-staged bidirectional KD operations, which is defined as

$$L_{dcm_2}(\hat{P}_t, \hat{P}_s) = \sum_{\{(i,j)|1 \le i,j \le K+1, i \ne j\}} L_{kd}(\hat{P}_{t_i}, \hat{P}_{s_j}). \tag{9}$$

In Eq. 8 and Eq. 9, L_{kd} is computed with the modified cross-entropy loss defined by Eq. 2. It matches the probabilistic outputs from any pair of the supervised layers in the teacher and student networks. According to the above definitions, it can be seen: bidirectional KD operations are performed not only between the same-staged supervised layers but also between the different-staged supervised layers of the teacher and student networks. *Benefiting from*

Algorithm 1: The DCM algorithm

Input : Training data $\{X, Y\}$, two CNN models W_t and W_s, classifier locations $\{(t_k, s_k)\}_{k=1}^{K+1}$, learning rate γ_i

Initialise W_t and W_s, $i = 0$;

repeat

$i \leftarrow i + 1$, update γ_i;

1. Randomly sample a batch of data from $\{X, Y\}$;

2. Compute knowledge set $\{(\hat{P}_{t_k}, \hat{P}_{s_k})\}_{k=1}^{K+1}$ at all supervised layers of two models by Eq. 3;

3. Compute loss L_t and L_s by Eq. 6, Eq. 7, Eq. 8, and Eq. 9 ;

4. Calculate gradients and update parameters: $W_t \leftarrow W_t - \gamma_i \frac{\partial L_t}{\partial W_t}$, $W_s \leftarrow W_s - \gamma_i \frac{\partial L_s}{\partial W_s}$

until *Converge*;

the well-designed auxiliary classifiers, such two types of cross-layer bidirectional KD operations are complimentary to each other as validated in the experiments. Enabling dense cross-layer bidirectional KD operations resembles a dynamic knowledge synergy process between two networks for the same task. The training algorithm of our DCM is summarized in Algorithm 1.

Connections to DML and KD. Regardless of the selection of measure function (Eq. 2 or Eq. 5) for matching probabilistic outputs, in the case where $Q = \emptyset$ meaning the supervision is only added to the last layer of the teacher and student networks, DCM becomes DML. In the extreme case where $Q = \emptyset$ and L_t is frozen, DCM becomes KD. Therefore, DML and KD are two special cases of DCM. Like KD and DML, DCM can be easily extended to handle more complex training scenarios where there are more than two neural networks. We leave this part as future research once a distributed system is available for training.

Setting of Q. In DCM, forming cross-layer bidirectional KD pairs to be connected depends on how to set Q. Setting Q needs to consider two basic questions: (1) Where to place auxiliary classifiers? (2) How to design their structures? Modern CNNs adopt a similar hierarchical structure consisting of several stages having different numbers of building blocks, where each stage has a down-sampling layer. In light of this, *to the first question, we use a practical principle, adding auxiliary classifiers merely to down-sampling layers of a backbone network* [18,58,35,42]. Existing works [18,42] showed that simple auxiliary classifiers usually worsen the training of modern CNNs as they have no convergence issues. Inspired by them, *to the second question, we use a heuristic principle, making the paths from the input to all auxiliary classifiers have the same number of down-sampling layers as the backbone network, and using backbone's building blocks to construct auxiliary classifiers with different numbers of building blocks and convolutional filters.* Finally in DCM, we enable dense two-way KDs between all layers added with auxiliary classifiers. Although the aforementioned setting may not be the best, it enjoys easy implementation and satisfied results on many CNNs as validated in our experiments.

4 Experiments

In this section, we describe the experiments conducted to evaluate the performance of DCM. We first compare DCM with DML [57] which is closely related to our method. We then provide more comprehensive comparisons for a deep analysis of DCM. All methods are implemented with PyTorch [34]. For fair comparisons, the experiments with all methods are performed under the exactly same settings for the data pre-processing method, the batch size, the number of training epochs, the learning rate schedule, and the other hyper-parameters.

4.1 Experiments on CIFAR-100

First, we perform experiments on the CIFAR-100 dataset with a variety of knowledge transfer settings.

CIFAR-100 dataset. It contains 50000 training samples and 10000 test samples, where samples are 32×32 color images collected from 100 object classes [23]. We use the same data pre-processing method as in [13,57]. For training, images are padded with 4 pixels to both sides, and 32×32 crops are randomly sampled from the padded images and their horizontal flips, which are finally normalized with the per-channel mean and std values. For evaluation, the original-sized test images are used.

Implementation details. We consider 4 state-of-the-art CNN architectures including: (1) ResNets [13] with depth 110/164; (2) DenseNet [19] with depth 40 and growth rate 12; (3) WRNs [52] with depth 28 and widening factor 4/10; (4) MobileNet [16] as used in [57]. We use the code released by the authors to train each CNN backbone. In the experiments, we consider two training scenarios: (1) Two CNNs with the same backbone (e.g., WRN-28-10 & WRN-28-10); (2) Two CNNs with the different backbones (e.g., WRN-28-10 & ResNet-110). In the first training scenario, for ResNets, DenseNet and WRNs, we use the same settings as reported in the original papers [13,19,52]. For MobileNet, we use the same setting as ResNets, following DML [57]. *In the second training scenario, we use the training setting of the network having better capacity to train both networks.* In our method, we append two auxiliary classifiers to the different-staged layers of each CNN backbone. Specifically, we add each auxiliary classifier after the corresponding building block having a down-sampling layer. All auxiliary classifiers have the same building blocks as in the backbone network, a global average pooling layer and a fully connected layer. The differences are the number of building blocks and the number of convolutional filters. *Detailed designs of auxiliary classifiers and training hyper-parameter settings are provided in the supplementary material.* For each joint training case, we run each method 5 times and report "mean(std)" error rates. All models are trained on a server using 1/2 GPUs according to the GPU memory requirement.

First training scenario. Results of training two models with the same backbone are shown in the first part of Table 1 from which we can find: (1) Both DML and DCM obviously improve the model performance compared to the independent training method; (2) Generally, DCM performs better than

Table 1. Result comparison on the CIFAR-100 dataset. WRN-28-10(+) denotes the models trained with dropout. Bolded results show the accuracy margins of DCM compared to DML. *In this paper, for each joint training case on the CIFAR-100 dataset, we run each method 5 times and report "mean(std)" top-1 error rates (%). Results of all methods are obtained with the exactly same training hyper-parameters, and our CNN baselines mostly have better accuracies compared to the numbers reported in their original papers [13,19,52,57].*

Networks		Ind(baseline)		DML		DCM	
Net1	Net2	Net1	Net2	Net1	Net2	Net1|**DCM-DML**	Net2|**DCM-DML**
ResNet-164	ResNet-164	22.56(0.20)	22.56(0.20)	20.69(0.25)	20.72(0.14)	19.57(0.20)|**1.12**	19.59(0.15)|**1.13**
WRN-28-10	WRN-28-10	18.72(0.24)	18.72(0.24)	17.89(0.26)	17.95(0.07)	16.61(0.24)|**1.28**	16.65(0.22)|**1.30**
DenseNet-40-12	DenseNet-40-12	24.91(0.18)	24.91(0.18)	23.18(0.18)	23.15(0.20)	22.35(0.12)|**0.83**	22.41(0.17)|**0.74**
WRN-28-10	ResNet-110	18.72(0.24)	26.55(0.26)	17.99(0.24)	24.42(0.19)	17.82(0.14)|**0.17**	22.99(0.30)|**1.43**
WRN-28-10	WRN-28-4	18.72(0.24)	21.39(0.30)	17.80(0.11)	20.21(0.16)	16.84(0.08)|**0.96**	18.76(0.14)|**1.45**
WRN-28-10	MobileNet	18.72(0.24)	26.30(0.35)	17.24(0.13)	23.91(0.22)	16.83(0.07)|**0.41**	21.43(0.20)|**2.48**
WRN-28-10(+)	WRN-28-10(+)	18.64(0.19)	18.64(0.19)	17.62(0.12)	17.61(0.13)	16.57(0.12)|**1.05**	16.59(0.15)|**1.02**

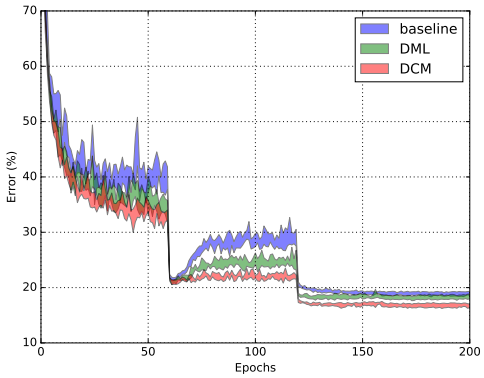

Fig. 2. Comparison of test curves at the different stages of jointly training two WRN-28-10 models. We show the range over 5 runs. Compared to the independent training method (baseline) and DML, DCM shows stably better performance during the whole training, and finally converges with the best accuracy on the test set.

DML. Taking the set of models having better mean accuracy as the example, the ResNet-164, WRN-28-10 and DenseNet-40-12 models trained with DCM show 1.12%, 1.28% and 0.80% average margins to the models trained with DML respectively; (3) The accuracy gain of DCM against DML shows a trend: the higher the network capacity, the larger the accuracy gain.

Second training scenario. The second part of Table 1 provides the results of training two models with the different backbones, from which we can make similar observations as in the first training scenario. Besides, we can find another critical observation: Two networks with different capacities have different accuracy improvements. Comparatively, the lower-capacity ResNet-110/MobileNet/WRN-28-4 can benefit more from the high-capacity WRN-28-10 for both DML and DCM, and the corresponding accuracy improvement becomes much more large with DCM. For example, the WRN-28-4/MobileNet model

trained with DCM shows 18.76%/21.43% mean error rate, outperforming the DML counterpart by 1.45%/2.48% margin.

The aforementioned experiments clearly validate the effectiveness of our method. Fig. 2 shows an illustrative comparison of test curves at the different stages of training two WRN-28-10 jointly with three different methods.

4.2 Experiments on ImageNet

Next, we perform experiments to validate the generalization ability of our method to a much larger dataset.

ImageNet classification dataset. It has about 1.2 million training images and 50 thousand validation images including 1000 object classes [38]. For training, images are resized to 256×256, and 224×224 crops are randomly sampled from the resized images or their horizontal flips normalized with the per-channel mean and std values. For evaluation, we report top-1 and top-5 error rates using the center crops of resized validation data.

Implementation details. On the ImageNet classification dataset, we use popular ResNet-18/50 [13] and MobileNetV2 [39] as the backbone networks, and consider the two same training scenarios as on the CIFAR-100 dataset. For all these CNN backbones, we use the same settings as reported in the original papers. In our method, we add two auxiliary classifiers to the different-staged layers of each CNN backbone. The auxiliary classifiers are constructed with the same building block as in the backbone network. The differences are the number of building blocks and the number of convolutional filters. *Detailed designs of auxiliary classifiers and training hyper-parameter settings are provided in the supplementary material.* For a concise comparison, we use the conventional data augmentation but not aggressive data augmentation methods. All models are trained on a server using 8 GPUs.

Results comparison. The results are summarized in Table 2. It can be seen that both DML and DCM bring noticeable accuracy improvements to the baseline model in the first scenario of training two networks with the same structure jointly, and DCM is better than DML. Comparatively, the better one of two ResNet-18 models trained by DCM shows 28.67%/9.71% top-1/top-5 error rate which outperforms the baseline model with a margin of 2.41%/1.46%. Impressively, our DCM shows at most 1.04%/0.76% accuracy improvement to DML on the MobileNetV2 model. These results are consistent with the results of training two networks with the same backbone on the CIFAR-100 dataset. In the second scenario of jointly training two different CNN backbones, the lower-capacity ResNet-18 benefits more from the high-capacity ResNet-50 than the reverse one for both DML and DCM, and the corresponding accuracy improvement becomes much larger by using DCM. Specifically, the ResNet-18 model trained by DCM can even reach 27.93%/9.19% top-1/top-5 error rate, showing 3.15%/1.98% and 0.72%/0.3% gain to the model trained with the independent training method and DML respectively. Although we use the conventional data augmentation, the best ResNet-18 model trained with our DCM shows 2.5% top-1 accuracy

Table 2. Result comparison on the ImageNet classification dataset. For each network, we report top-1/top-5 error rate (%). Bolded results show the accuracy margins of DCM compared to the independent training method/DML.

Networks		Ind(baseline)		DML		DCM					
Net1	Net2	Net1	Net2	Net1	Net2	Net1	**DCM-Ind**	**DCM-DML**	Net2	**DCM-Ind**	**DCM-DML**
ResNet-18	ResNet-18	31.08/11.17	31.08/11.17	29.13/9.89	29.25/10.00	28.67/9.71	**2.41/1.46**	**0.46/0.18**	28.74/9.74	**2.34/1.43**	**0.51/0.26**
MobileNetV2	MobileNetV2	27.80/9.50	27.80/9.50	26.61/8.85	26.78/8.97	25.62/8.16	**2.18/1.34**	**0.99/0.69**	25.74/8.21	**2.06/1.29**	**1.04/0.76**
ResNet-50	ResNet-18	25.47/7.58	31.08/11.17	25.24/7.56	28.65/9.49	24.92/7.42	**0.55/0.16**	**0.32/0.14**	27.93/9.19	**3.15/1.98**	**0.72/0.30**

Table 3. Result comparison of jointly training two WRN-28-10 models on the CIFAR-100 dataset using different layer location settings for placing auxiliary classifiers. C1 denotes the default classifier over the last layer of the network, and C2, C3 and C4 denote 3 auxiliary classifiers with the increased layer distance to C1 (see supplementary material for details). We report "mean(std)" error rates (%) over 5 runs.

Classifier locations	WRN-28-10	
	Net1	Net2
baseline	18.72(0.24)	18.72(0.24)
C1+C4	17.16(0.14)	17.25(0.15)
C1+C3	16.89(0.21)	17.04(0.06)
C1+C2	17.40(0.20)	17.38(0.17)
C1+C2C3(default)	16.61(0.24)	16.65(0.22)
C1+C2C3C4	**16.59**(0.12)	16.73(0.17)

gain against the model (trained with aggressive data augmentation methods) released at the official GitHub page of Facebook [1].

4.3 Deep Analysis of DCM

Finally, we conduct extensive ablative experiments on the CIFAR-100 dataset to better understand our method and show its capability to handle more challenging scenarios.

Setting of Q. Recall that the set Q plays a critical role in DCM. The setting of Q is closely related to two basic questions: (1) Where to place Auxiliary CLassiFiers (ACLFs)? (2) How to design the structure of ACLFs? As we discussed in the method section, we add ACLFs to the down-sampling layers of the network, following the common practices as used in [18,58,35,42]. However, a modern CNN architecture usually has several (e.g., 3/5) down-sampling layers, and thus there exist many layer location combinations for placing ACLFs. To this question, we conduct ablative experiments to jointly train two WRN-28-10 models considering different settings by adding ACLFs to at most three down-sampling layers. The results summarized in Table 3 show that the 2-ACLF model brings relatively large gain compared to the 1-ACLF model, while the 3-ACLF model gives negligible gain compared to the 2-ACLF model. Therefore, we add 2 ACLFs as the default setting of DCM for a good accuracy-efficiency trade-off. To the second question, we evaluate two additional kinds of ACLFs besides the default ACLFs used in DCM. Results are shown in Table 4 where "APFC" refers to a structure that uses an average pooling layer to down-sample input feature

[1] `https://github.com/facebook/fb.resnet.torch`

Table 4. Result comparison of jointly training DenseNet-40-12 and WRN-28-10 on the CIFAR-100 dataset using different types of auxiliary classifiers. We report "mean(std)" error rates (%) over 5 runs.

Net1/Net2	Classifier type	DCM
DenseNet-40-12	APFC	25.10(0.25)
	Narrow	22.45(0.25)
	default	**22.35**(0.12)
WRN-28-10	APFC	18.23(0.10)
	Narrow	16.88(0.17)
	default	**16.61**(0.24)

Table 5. Result comparison of training two DenseNet-40-12 models jointly on the CIFAR-100 dataset using different settings of cross-layer bidirectional KD. DCM-1/DCM-2 performs KD operations between the same-staged/different-staged layers. We report "mean(std)" error rates (%) over 5 runs.

Method	Error (%)	
	Net1	Net2
baseline	24.91(0.18)	24.91(0.18)
DML	23.18(0.18)	23.15(0.20)
DML + DS	23.18(0.33)	23.08(0.28)
DCM-1	22.86(0.16)	22.89(0.14)
DCM-2	22.43(0.25)	22.51(0.18)
DCM	**22.35(0.12)**	**22.41(0.17)**

maps and then uses a fully connected layer to generate logits. "Narrow" refers to a narrower version (smaller width multipliers) compared to the default ACLF design. It can be seen that simple ACLFs may hurt performance sometimes (similar experiments are also provided in [18,58,42]), in which scenario our method has a small gain. Comparatively, large accuracy gains are obtained in the other two cases, therefore relatively strong ACLFs are required to make our method work properly. After training, ACLFs are discarded, and thus there are no extra parameters added to final models.

Analysis of cross-layer bidirectional KDs. Recall that our DCM presents two cross-layer bidirectional KD designs (between either the same-staged or different-staged layers) to mutually transfer knowledge between two networks. In order to study their effects, we conduct two experiments in which we keep either the first or second bidirectional KD design. In the experiments, we consider the case of jointly training two DenseNet-40-12 models. Surprisingly, the results provided in Table 5 show that the second design brings larger performance gain than the first design, which means knowledge transfer between the different-staged layers is more effective. Because of the introduction of well-designed auxiliary classifiers, DCM enables much more diverse and effective bidirectional knowledge transfer which improves joint training performance considerably.

Accuracy gain from deep supervision. There is a critical question to DCM: Is the performance margin between DCM and DML mostly owing to the auxiliary classifiers added to certain hidden layers of two networks as they have additional parameters? To examine this question, we conduct extensive experiments considering two different settings: (1) For DCM, we remove all the bidi-

Table 6. Comparison of Deep Supervision (DS) and DCM on the CIFAR-100 dataset. We report "mean(std)" error rates (%) over 5 runs.

Network	baseline	DS	DCM
ResNet-164	22.56(0.20)	21.38(0.32)	**19.57(0.20)**
DenseNet-40-12	24.91(0.18)	24.46(0.22)	**22.35(0.12)**
WRN-28-10	18.72(0.24)	18.32(0.13)	**16.61(0.24)**
WRN-28-10(+)	18.64(0.19)	17.80(0.29)	**16.57(0.12)**

rectional KD connections except the one between the last layer of two DenseNet-40-12 backbones while retaining auxiliary classifiers. This configuration can be regarded as a straightforward combination of DML and Deep Supervision (DS); (2) Further, we add auxiliary classifiers to individual CNN backbones, and train each of them with DS independently while train two same backbones with DCM simultaneously. The results under the first setting are provided in Table 5, denoted as DML + DS. It can be observed that the combination of DML and DS only brings 0.07% average improvement to DenseNet-40-12, which only occupies 8.75% of the total margin brought by DCM. The results under the second setting are summarized in Table 6. It can be noticed that the average gain of DS to each baseline model is less than 0.85% in the most cases. Comparatively, DCM shows consistently large accuracy improvements over the baseline models, ranging from 2.07% to 2.99%.

Comparison with KD and its variants. *Note that a fair comparison of DCM/DML with KD and its variants is impractical as the training paradigm is quite different.* Here we illustratively study how much KD will work in our case, via using a pre-trained WRN-28-10 to guide the training of a ResNet-110. Surprisingly, KD shows a slightly worse mean error rate than baseline (26.66% vs. 26.55%). We noticed that during training, the soft labels generated by the teacher (WRN-28-10) are not so "soft" and the accuracy of these soft labels is very high ($\sim$ 99% on the CIFAR-100 dataset, meaning the model usually fits training data "perfectly"). These soft labels don't provide any more useful guidance than the hard labels and cause overfitting somehow. Using DML or DCM, the soft labels are generated dynamically as the teacher and student networks are jointly trained from the scratch, so they are comparatively softer and contain more useful guidance at every training iteration. *Besides, we provide horizontal comparisons of DCM with KD variants in the supplementary material.*

With noisy data. We also explore the capability of our method to handle noisy data. Following [56,55], we use CIFAR-10 dataset and jointly train two DenseNet-40-12 models as a test case. Before training, we randomly sample a fixed ratio of training data and replace their ground truth labels with randomly generated wrong labels. After training, we evaluate the models on the raw testing set. The results are summarized in Table 7. Three corruption ratios 0.2, 0.5, and 0.8 are considered. Compared to the case with 0.2 corruption ratio, the margin between DCM and baseline increases as the corruption ratio increases. One possible explanation of this phenomenon is that DCM behaves as a regularizer. When the training labels get corrupted, the baseline model will try to fit the training data and capture the wrong information, which causes severe over-

Table 7. Result comparison on the CIFAR-10 dataset with noisy labels. We jointly train two DenseNet-40-12 models, and report "mean(std)" error rates (%) over 5 runs.

Corruption ratio	Method	Error (%)
0.2	baseline	9.85(0.24)
	DML	8.13(0.14)
	DCM	**7.11(0.11)**
0.5	baseline	17.93(0.39)
	DML	14.31(0.30)
	DCM	**12.08(0.34)**
0.8	baseline	35.32(0.42)
	DML	32.65(0.96)
	DCM	**31.26(0.94)**

fitting. In the DCM configuration, things are different. Beyond the corrupted labels, the classifiers also get supervision from the soft labels generated by other different-staged or same-staged classifiers. These soft labels can prevent the classifiers from fitting the corrupted data and finally improve the generalization to a certain degree. In normal training without corrupted data, this can also happen. For example, if there is an image of a person with his or her dog, the human-annotated ground truth will be a 1-class label, either "dog" or "person", but not both. This kind of images can be seen as "noisy" data, and this is where soft-labels dynamically generated will kick in.

With strong regularization. The aforementioned experiments show DCM behaves as a strong regularizer which can improve the generalization of the models. In order to study the performance of DCM under the existence of other strong regularizations, we follow the dropout experiments in [52]. We add a dropout layer with $p = 0.3$ after the first layer of every building block of WRN-28-10. The results are shown in Table 1 as WRN-28-10(+). It can be seen that combining DCM with dropout achieves better performance than DCM, which means DCM is compatible with traditional regularization techniques like dropout.

Comparison of efficiency. In average, DCM is about 1.5× slower than DML during the training phase due to the use of auxiliary classifiers. However, all auxiliary classifiers are discarded after training, so there is no extra computational cost to the resulting model during the inference phase compared with the independent training method and DML. With DCM, the lower-capacity model has similar accuracy but requires much less computational cost compared to high-capacity model. For example, as shown in Table 1, WRN-28-4 models trained with DCM show a mean error rate of 18.76% which is almost the same to that of WRN-28-10 models trained with the independent training method.

5 Conclusions

In this paper, we present DCM, an effective two-way knowledge transfer method for collaboratively training two networks from scratch. It connects knowledge representation learning with deep supervision methodology and introduces dense cross-layer bidirectional KD designs. Experiments on a variety of knowledge transfer tasks validate the effectiveness of our method.

References

1. Aguilar, G., Ling, Y., Zhang, Y., Yao, B., Fan, X., Guo, C.: Knowledge distillation from internal representations. In: AAAI (2020)
2. Ba, L.J., Caruana, R.: Do deep nets really need to be deep? In: NIPS (2014)
3. Batra, T., Parikh, D.: Cooperative learning with visual attributes. arXiv preprint arXiv:1705.05512 (2017)
4. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: COLT (1998)
5. Bucilă, C., Caruana, R., Niculescu-Mizil, A.: Model compression. In: KDD (2006)
6. Chen, T., Goodfellow, I., Shlens, J.: Net2net: Accelerating learning via knowledge transfer. In: ICLR (2016)
7. Chen, Y., Wang, Z., Peng, Y., Zhang, Yu, G., Sun, J.: Cascaded pyramid network for multi-person pose estimation. In: CVPR (2018)
8. Garcia, N.C., Morerio, P., Murino, V.: Modality distillation with multiple stream networks for action recognition. In: ECCV (2018)
9. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR (2014)
10. Guo, X., Li, H., Yi, S., Ren, J., Wang, X.: Learning monocular depth by distilling cross-domain stereo networks. In: ECCV (2018)
11. Hafner, F., Bhuiyan, A., Kooij, J.F.P., Granger, E.: A cross-modal distillation network for person re-identification in rgb-depth. arXiv preprint arXiv:1810.11641 (2018)
12. He, D., Xia, Y., Qin, T., Wang, L., Yu, N., Liu, T.Y., Ma, W.Y.: Dual learning for machine translation. In: NIPS (2017)
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
14. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
15. Hou, S., Pan, X., Loy, C.C., Wang, Z., Lin, D.: Lifelong learning via progressive distillation and retrospection. In: ECCV (2018)
16. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
17. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: CVPR (2018)
18. Huang, G., Chen, D., Li, T., Wu, F., van der Maaten, L., Weinberger, K.Q.: Multi-scale dense networks for resource efficient image classification. In: ICLR (2018)
19. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: CVPR (2017)
20. Jia, S., Bruce, N.D.B.: Richer and deeper supervision network for salient object detection. arXiv preprint arXiv:1901.02425 (2018)
21. Kim, J., Park, S., Kwak, N.: Paraphrasing complex network: Network compression via factor transfer. In: NeurIPS (2018)
22. Kim, Y., Rush, A.M.: Sequence-level knowledge distillation. In: EMNLP (2016)
23. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. In: Tech Report (2009)
24. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS (2012)
25. Kundu, J.N., Lakkakula, N., Babu, R.V.: Um-adapt: Unsupervised multi-task adaptation using adversarial cross-task distillation. In: ICCV (2019)

26. Lan, X., Zhu, X., Gong, S.: Knowledge distillation by on-the-fly native ensemble. In: NeurIPS (2018)
27. Lee, C.Y., Xie, S., Gallagher, P., Zhang, Z., Tu, Z.: Deeply-supervised nets. In: AISTATS (2015)
28. Lee, S.H., Kim, H.D., Song, B.C.: Self-supervised knowledge distillation using singular value decomposition. In: NeurIPS (2018)
29. Li, Y., Wang, N., Liu, J., Hou, X.: Demystifying neural style transfer. In: IJCAI (2016)
30. Li, Z., Hoiem, D.: Learning without forgetting. In: ECCV (2016)
31. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: ECCV (2016)
32. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR (2015)
33. Newell, A., Yang, K., Deng, J.: Stacked hourglass networks for human pose estimation. In: ECCV (2016)
34. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch. In: NIPS Workshops (2017)
35. Phuong, M., Lampert, C.H.: Distillation-based training for multi-exit architectures. In: ICCV (2019)
36. Qiao, S., Shen, W., Zhang, Z., Wang, B., Yuille, A.: Deep co-training for semi-supervised image recognition. In: ECCV (2018)
37. Romero, A., Ballas, N., Kahou, S.E., Chassang, A., Gatta, C., Bengio, Y.: Fitnets: Hints for thin deep nets. In: ICLR (2015)
38. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Li, F.F.: Imagenet large scale visual recognition challenge. International Journal of Computer Vision **115**(3), 211–252 (2015)
39. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: CVPR (2018)
40. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR (2015)
41. Song, G., Chai, W.: Collaborative learning for deep neural networks. In: NeurIPS (2018)
42. Sun, D., Yao, A., Zhou, A., Zhao, H.: Deeply-supervised knowledge synergy. In: CVPR (2019)
43. Sze, V., Chen, Y.H., Yang, T.J., Emer, J.S.: Efficient processing of deep neural networks: A tutorial and survey. Proceedings of the IEEE **105**(12), 2295–2329 (2017)
44. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: CVPR (2015)
45. Tian, y., Krishnan, D., Isola, P.: Contrastive representation distillation. In: ICLR (2020)
46. Tian, Y., Krishnan, D., Isola, P.: Contrastive representation distillation. ICLR (2020)
47. Wang, Z., Deng, Z., Wang, S.: Accelerating convolutional neural networks with dominant convolutional kernel and knowledge pre-regression. In: ECCV (2016)
48. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: CVPR (2017)
49. Xie, S., Tu, Z.: Holistically-nested edge detection. In: ICCV (2015)

50. Xu, D., Ouyang, W., Wang, X., Nicu, S.: Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. In: CVPR (2018)
51. Yim, J., Joo, D., Bae, J., Kim, J.: A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In: CVPR (2017)
52. Zagoruyko, S., Komodakis, N.: Wide residual networks. In: BMVC (2016)
53. Zagoruyko, S., Komodakis, N.: Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In: ICLR (2017)
54. Zhai, M., Chen, L., Tung, F., He, J., Nawhal, M., Mori, G.: Lifelong gan: Continual learning for conditional image generation. In: ICCV (2019)
55. Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning requires rethinking generalization. In: ICLR (2017)
56. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. In: ICLR (2018)
57. Zhang, Y., Xiang, T., Hospedales, T.M., Lu, H.: Deep mutual learning. In: CVPR (2018)
58. Zhang, Z., Zhang, X., Peng, C., Cheng, D., Sun, J.: Exfuse: Enhancing feature fusion for semantic segmentation. In: ECCV (2018)
59. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: CVPR (2017)