Deeply-supervised Knowledge Synergy

Dawei Sun[1,2*]   Anbang Yao[1*]   Aojun Zhou[1]   Hao Zhao[1,2]
[1]Intel Labs China   [2]Tsinghua University
{dawei.sun, anbang.yao, aojun.zhou, hao.zhao}@intel.com

Abstract

Convolutional Neural Networks (CNNs) have become deeper and more complicated compared with the pioneering AlexNet. However, current prevailing training scheme follows the previous way of adding supervision to the last layer of the network only and propagating error information up layer-by-layer. In this paper, we propose Deeply-supervised Knowledge Synergy (DKS), a new method aiming to train CNNs with improved generalization ability for image classification tasks without introducing extra computational cost during inference. Inspired by the deeply-supervised learning scheme, we first append auxiliary supervision branches on top of certain intermediate network layers. While properly using auxiliary supervision can improve model accuracy to some degree, we go one step further to explore the possibility of utilizing the probabilistic knowledge dynamically learnt by the classifiers connected to the backbone network as a new regularization to improve the training. A novel synergy loss, which considers pairwise knowledge matching among all supervision branches, is presented. Intriguingly, it enables dense pairwise knowledge matching operations in both top-down and bottom-up directions at each training iteration, resembling a dynamic synergy process for the same task. We evaluate DKS on image classification datasets using state-of-the-art CNN architectures, and show that the models trained with it are consistently better than the corresponding counterparts. For instance, on the ImageNet classification benchmark, our ResNet-152 model outperforms the baseline model with a 1.47% margin in Top-1 accuracy. Code is available at https://github.com/sundw2014/DKS.

1. Introduction

Deep Convolutional Neural Networks (CNNs) have large numbers of learnable parameters, which makes them have

---

*Equal contribution. This work was done when Dawei Sun was an intern at Intel Labs China, supervised by Anbang Yao who is responsible for correspondence. Interns Aojun Zhou and Hao Zhao contributed to early theoretical analysis.
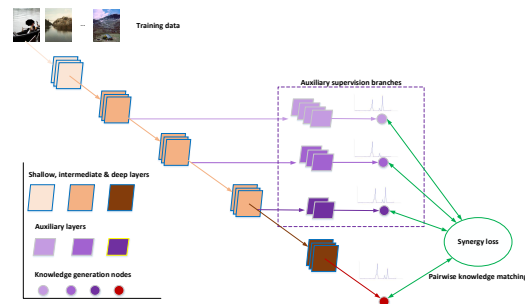

Figure 1: Illustration of the proposed method. In the figure, we add three auxiliary supervision branches on top of some intermediate layers of the backbone network. Every branch will output a class probability distribution conditioned on the training data, which is used as the knowledge. We use circles to indicate the nodes for calculating these knowledge outputs, and propose a synergy loss term to enable the pairwise matching among them. Best viewed electronically.

much better capability in fitting training data than traditional machine learning methods. Along with the growing availability of training resources including large-scale datasets, powerful hardware platforms and effective development tools, CNNs have become the dominant learning models for a variety of visual recognition tasks [21, 26, 7, 42]. In order to get more compelling performance, CNNs [39, 10, 47, 17, 44, 15, 1] are designed to be considerably deeper and more complicated in comparison to the seminal AlexNet [21] which has 8 layers and achieved groundbreaking results in the ImageNet classification competition 2012 [37]. Despite that modern CNNs widely use various engineering techniques such as careful hyper-parameter tuning [39], aggressive data argumentation [44, 49], effective normalization [18, 9] and sophisticated connection path [10, 17, 44, 15, 1] to ease network training, their training remains to be difficult.

We notice that state-of-the-art CNN models such as ResNet [10], WRN [47], DenseNet [17], ResNeXt [44], SENet [15], DPN [1], MobileNet [14, 38] and ShuffleNet [51, 27] adopt the training scheme of AlexNet. More

specifically, during training, the supervision is only added to the last layer of the network and the training error is back propagated from the last layer to earlier layers. Because of the increased complexity in network depth, building blocks and network topologies, this might pose a risk of insufficient representation learning, especially to the layers from which there are long connection paths to the supervision layer. This problem may be alleviated by the deeply-supervised learning scheme proposed in [41] and [22] independently. Szegedy et al. [41] add auxiliary classifiers to two intermediate layers of their proposed GoogLeNet, while Lee et al. [22] propose to add auxiliary classifiers to all hidden layers of the network. During network training, although different types of auxiliary classifiers are used in these two methods, they adopt the same optimization strategy in which the training loss is the weighted sum of the losses of all auxiliary classifiers and the loss of the classifier connected to the last layer. Such methodology has proven to be notably effective in combating the vanishing gradient problem and overcoming the convergence issue for training some old deep classification networks. However, modern CNN backbones usually have no convergence issue, and rarely use auxiliary classifiers. Recently, Huang et al. [16] present a two-dimensional multi-scale CNN architecture using early-exit classifiers for cost-aware image classification. In [16], empirical results show that naively attaching simple auxiliary classifiers to the early layers of a state-of-the-art CNN such as ResNet or DenseNet leads to decreased performance, but this issue can be alleviated with a combination of multi-scale features and dense connections from the architecture design perspective.

In this paper, we revisit the deeply-supervised learning methodology for image classification tasks, and present a new method called Deeply-supervised Knowledge Synergy (DKS) targeting to train state-of-the-art CNNs with improved accuracy and without introducing extra computational cost during inference. Inspired by the aforementioned works [41, 22, 16], we first append auxiliary supervision branches on top of certain intermediate layers during network training as illustrated in Fig. 1. We show that using carefully designed auxiliary classifiers can improve the accuracy of state-of-the-art CNNs to a certain extent. This empirically indicates that the information from the auxiliary supervision is beneficial in regularizing the training of modern CNNs. We conjecture there may still exist room for performance improvement by enabling explicit information interactions among all supervision branches connected to the backbone network, thus we go one step further to explore the possibility of utilizing the knowledge (namely the class probability outputs evaluated on the training data) dynamically learnt by the auxiliary classifiers and the classifier added to the last network layer as a new regularization to improve the training. In the optimization, a novel

synergy loss, which considers pairwise knowledge matching among all supervision branches, is added to the training loss. This loss enables dense pairwise knowledge matching operations in both top-down and bottom-up directions at each training step, resembling a dynamic synergy process for the same task. We evaluate the proposed method on two well-known image classification datasets using the most prevalent CNN architectures including ResNet [10], WRN [47], DenseNet [17] and MobileNet [14]. We show that the models trained with our method have impressive accuracy improvements compared with their respective baseline models. For example, on the challenging ImageNet classification dataset, even to very deep ResNet-152 architecture, there is a 1.47% improvement in Top-1 accuracy.

2. Related Work

Here, we summarize related approaches in the literature, and analyze their relations and differences with our method.

Deeply-Supervised Learning. The deeply-supervised learning methodology [41, 22] was released in 2014. It uses auxiliary classifiers connected to the hidden layers of the network to address the convergence problem when training some old deep CNNs for image classification tasks. Recently, it has also been used in other visual recognition tasks such as edge detection [45], human pose estimation [31], scene parsing [54], semantic segmentation [53], keypoint localization [23], automatic delineation [29] and travel time estimation [50]. Despite these recent advances in its new applications, modern CNN classification models rarely use auxiliary classifiers. As reported in [16], directly appending simple auxiliary classifiers on top of the early layers of a state-of-the-art network such as ResNet or DenseNet hurts its performance. In this paper, we present DKS, a new deeply-supervised learning method for image classification tasks, which shows impressive accuracy improvements when training state-of-the-art CNNs.

Knowledge Transfer. In the recent years, Knowledge Transfer (KT) research has been attracting increasing interest. A pioneering work is Knowledge Distillation (KD) [11] in which the soft outputs from a large teacher model or an ensemble of teacher models are used to regularize the training of a smaller student network. [36], [46] and [48] further show that intermediate feature representations can also be used as hints to enhance knowledge distillation process. KD techniques have also been used in other tasks, for instance, improving the performance of low-precision CNNs for image classification [28] and designing multiple-stream CNNs for video action recognition [5]. Unlike KD and its variants in which knowledge is only transferred from teacher models to a student model, [52] extends KD by presenting a mutual learning strategy, showing that the knowledge of the student model is also helpful to improve the accuracy of the teacher model. Later, this idea was used in person

re-identification [55] and joint human parsing and pose estimation [32]. Li and Hoiem [24] address the problem of adapting a trained neural network model to handle new vision tasks while preserving the old knowledge through a combination of KD and fine-tuning. An improved method is proposed in [12]. Qiao et al. [35] propose a deep co-training method for semi-supervised image classification. In their method, all models are considered as students and trained with different data views containing adversarial samples. In this paper, the proposed deeply-supervised knowledge synergy method is a new form of knowledge transfer within one single neural network, which differs from the aforementioned methods both in focus and formulation.

CNN Regularization. ReLU [30], Dropout [40] and BN [18] are proven to be the keys for modern CNNs to combat over-fitting or accelerate convergence. Because of this, many improved variants [9, 43, 4, 8, 6] have been proposed recently. Over-fitting can also be reduced by synthetically increasing the size of existing training data via augment transformations such as random cropping, flipping, scaling, color manipulation and linear interpolation [21, 13, 41, 49]. In addition, pre-training [39] can assist the early stages of the neural network training. These methods are widely used in modern CNN architecture design and training. Our method is compatible with them. As can be seen in Fig. 3, the model trained with DKS has the highest training error but the lowest test error, showing that our method behaves like a regularizer and reduces over-fitting for ResNet-18.

3. The Proposed Method

In this section, we present the formulation of our method, highlight its insight, and detail its implementation.

3.1. Deeply-Supervised Learning

We begin with the formulation of the deeply-supervised learning scheme as our method is based on it. Let W_c be the parameters of a L-layer CNN model that needs to be learnt. Let $D = \{(x_i, y_i) | 1 \leq i \leq N\}$ be an annotated data set having N training samples collected from K image classes. Here, x_i is the ith training sample and y_i is the corresponding ground truth label (a one-hot vector with K dimensions). Let $f(W_c, x_i)$ be the K-dimensional output vector of the CNN model for a training sample x_i. For the standard training scheme, the supervision is only added to the last layer of the network, and the optimization objective can be defined as

$$\underset{W_c}{\operatorname{argmin}} \quad L_c(W_c, D) + \lambda R(W_c), \tag{1}$$

where L_c is the default loss, R is the regularization term, and λ is a positive coefficient. Here, L_c is defined as

$$L_c(W_c, D) = \frac{1}{N} \sum_{i=1}^{N} H(y_i, f(W_c, x_i)),$$

where H is a cross-entropy cost function

$$H(y_i, f(W_c, x_i)) = -\sum_{k=1}^{K} y_i^k \log f^k(W_c, x_i).$$

As λR is a default term and has no relation with our method, we omit this term in the following description for simplicity. Now, the objective function (1) can be reduced into

$$\underset{W_c}{\operatorname{argmin}} \quad L_c(W_c, D). \tag{2}$$

This optimization problem can be readily solved by SGD and its variants [3, 19, 2]. To the best of our knowledge, most of the well-known CNNs [21, 39, 10, 47, 17, 44, 14, 38, 15, 1, 51, 27, 56, 34, 25] adopt this optimization scheme in the model training. By contrast, the deeply-supervised learning scheme explicitly proposed in [22] adds auxiliary classifiers to all hidden layers of the network during training. Let $W_a = \{w_a^l | 1 \leq l \leq L - 1\}$ be a set of auxiliary classifiers attached on the top of every hidden layer of the network. Here, w_a^l denotes the parameters of the auxiliary classifier added to the lth hidden layer. Let $f(w_a^l, W_c, x_i)$ be the K-dimensional output vector of the lth auxiliary classifier. Without loss of generality, the optimization objective of the deeply-supervised learning scheme can be defined as

$$\underset{W_c, W_a}{\operatorname{argmin}} \quad L_c(W_c, D) + L_a(W_a, W_c, D), \tag{3}$$

where

$$L_a(W_a, W_c, D) = \frac{1}{N} \sum_{i=1}^{N} \sum_{l=1}^{L-1} \alpha_l H(y_i, f(w_a^l, W_c, x_i)).$$

The auxiliary loss L_a is the weighted sum of the losses of all auxiliary classifiers evaluated on the training set and α_l weights the loss of the lth auxiliary classifier. By introducing auxiliary loss L_a, the deeply-supervised learning scheme allows the network to gather gradients not only from the last layer supervision but also from the hidden layer supervision during training. This is thought to combat the vanishing gradient problem and enhance convergence [22, 41].

As for the contemporary work [41], its optimization objective can be thought as a special case of (3) as it only adds auxiliary classifiers to two intermediate layers of the proposed GoogLeNet. The other difference lies in the structure of auxiliary classifiers. In the experiments, [22] used simple classifiers with a zero-ing strategy to dynamically control the value of α_l during training, while [41] used more complex classifiers with a fixed value of α_l. We find that setting a fixed value for α_l gives similar performance to the zero-ing strategy when training state-of-the-art CNNs, thus we use fixed values for α_l in our implementation.

3.2. Deeply-supervised Knowledge Synergy

Now, we present the formulation of our DKS which further develops the deeply-supervised learning methodology from a new perspective. DKS also uses auxiliary classifiers connected to some hidden layers of the network, but unlike existing methods, it introduces explicit information interactions among all supervision branches. Specifically, DKS uses the knowledge (i.e., the class probability outputs evaluated on the training data) dynamically learnt by all classifiers to regularize network training. Its core contribution is a novel synergy loss which enables dense pairwise knowledge matching among all classifiers connected to the backbone network, making optimization more effective.

In this section, we follow the notations in the last section. We only add auxiliary classifiers to certain hidden layers. Let $A \subseteq \{1, 2, \cdots, L-1\}$ be a pre-defined set with $|A|$ layer indices, indicating where auxiliary classifiers are added. Let $\hat{A} = A \cup \{L\}$, where L is the index of the last layer of the network, so that $\hat{A}$ indicates the locations of all classifiers connected to the network including both the auxiliary ones and the original one. Let $B \subseteq \hat{A} \times \hat{A}$ be another pre-defined set with $|B|$ pairs of layer indices, indicating where pairwise knowledge matching operations are activated.

Now, following the definition of (3), the optimization objective of our DKS is defined as

$$\underset{W_c, W_a}{\operatorname{argmin}} \quad L_c(W_c, D) + L_a(W_a, W_c, D) + L_s(W_a, W_c, D). \quad (4)$$

Here, the default loss L_c is the same as in (3), the auxiliary loss L_a is defined as

$$L_a(W_a, W_c, D) = \frac{1}{N} \sum_{i=1}^{N} \sum_{l \in A} \alpha_l H(y_i, f(w_a^l, W_c, x_i)),$$

and the proposed synergy loss L_s is defined as

$$L_s(W_a, W_c, D) = \frac{1}{N} \sum_{i=1}^{N} \sum_{(m,n) \in B} H(f_m, f_n).$$

The pairwise knowledge matching from the classifier m to n is evaluated with $H(f_m, f_n)$ which is defined as

$$H(f_m, f_n) = -\beta_{mn} \sum_{k=1}^{K} f_m^k \log f_n^k,$$

where f_m and f_n are the class probability outputs of the classifier m and n evaluated on the training sample x_i respectively, and β_{mn} weights the loss of the pairwise knowledge matching from the classifier m to n. We use a Softmax function to compute class probability. In the experiments, we set $\alpha_l = 1$, $\beta_{mn} = 1$ and keep them fixed, which means there is no extra hyper-parameter in the optimization of our method compared with the optimization (2) and (3). For
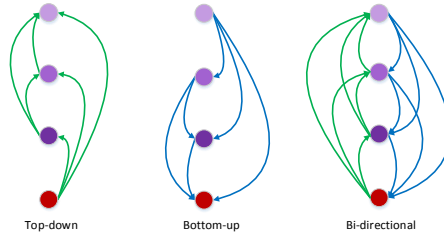

Figure 2: Illustration of three pairwise knowledge matching strategies. In each strategy, the red circle denotes the classifier connected to the last layer of the network, and the purple circles denote three auxiliary classifiers connected to certain intermediate layers, and the curved arrows represent the pairwise knowledge matching directions.

the synergy loss, the knowledge matching between any two classifiers is a modified cross-entropy loss function with a soft target. In principle, taking the current class probability outputs from the classifier m as soft labels (which are considered as constant values and the gradients w.r.t. them will not be calculated in the back-propagation), it forces the classifier n to mimic the classifier m. In this way, the knowledge currently learnt by the classifier m can be transferred to the classifier n. We call this term a directional supervision. Intriguingly, enabling dense pairwise knowledge matching operations among all supervision branches connected to the backbone network resembles a dynamic synergy process for the same task.

Pairwise Knowledge Matching. For DKS, a critical question is how to configure the knowledge matching pairs (i.e., set B). We provide three options including the top-down, bottom-up and bi-directional strategies, as illustrated in Fig. 2. With the top-down strategy, only the knowledge of the classifiers connected to the deep layers of a backbone network are used to guide the training of the classifiers added to the earlier layers. The bottom-up strategy reverses this setting and the bi-directional strategy includes both of them. A comparison study (see experiments section) shows that the bi-directional strategy has the best performance, thus we adopt it in the final implementation.

Auxiliary Classifiers. Another basic question for DKS is how to design the structure of auxiliary classifiers. Although the deeply-supervised learning scheme has proven to be effective in addressing the convergence issue when training some old deep networks for image classification tasks [22], state-of-the-art CNNs such as ResNet and DenseNet are known to be free of convergence issue, even for models having hundreds of layers. In view of this, directly adding simple auxiliary classifiers to the hidden layers of the network might not be helpful, which has been empirically verified by [16] and [53]. From the CNN architecture design perspective, [41] and [16] propose to add
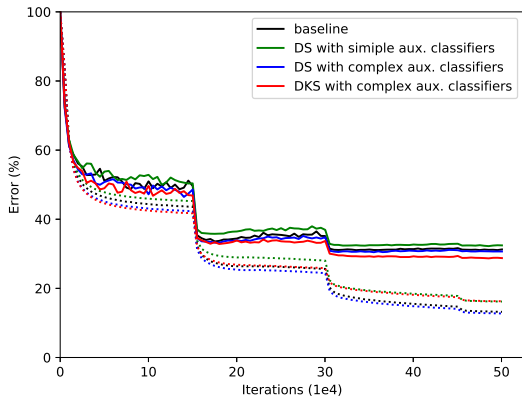
Figure 3: Curves of Top-1 training error (dashed line) and test error (solid line) of the ResNet-18 models trained on the ImageNet classification dataset. Compared with the baseline model, simple auxiliary classifiers (added after the block Conv3_x and Conv4_x) lead to 1.17% drop in Top-1 accuracy and complex designs bring a 0.60% improvement, while our method achieves 2.38% gain. Remarkably, our method converges with the lowest accuracy on training set but achieves the best accuracy on test set, showing better capability in suppressing over-fitting.

complex auxiliary classifiers to some intermediate layers of the network to alleviate this problem. Following them, in the experiments, we append relatively complex auxiliary supervision branches on top of certain intermediate layers during network training. Specifically, every auxiliary branch is composed of the same building block (e.g., residual block in ResNet) as in the backbone network. As empirically verified in [16], early layers lack coarse-level features which are helpful for image-level classification. In order to address this problem, we use a heuristic principle making the paths from the input to every classifier have the same number of down-sampling layers. Comparative experiments show that these carefully designed auxiliary supervision branches can improve final model performance to some extent but the gain is relatively minor. By enabling dense pairwise knowledge matching via the proposed synergy loss, we achieve much better results. Fig. 3 shows some illustrative results, and more results can be found in experiments section.

Comparison with Knowledge Distillation. In the DKS, the pairwise knowledge matching is inspired by the knowledge distillation idea popularly used in knowledge transfer [11, 48, 36, 46, 28, 52, 24, 12, 35]. Here, we clarify their differences. First, our method differs with them in focus. This line of research mainly addresses the network compression problem following a student-teacher framework, but our method focuses on advancing the training of state-of-the-art CNNs by further developing the deeply-

supervised learning methodology. Second, our method differs with them in formulation. Under the student-teacher framework, large teacher models are usually supposed to be available beforehand, and the optimization is defined to use the soft outputs from teacher models to guide the training of smaller student networks. That is, teacher models and student models are separately optimized, and there is no direct relation between them. In our method, auxiliary classifiers share different-level feature layers of the backbone network, and they are jointly optimized with the classifier connected to the last layer. In this paper, we also conduct experiments to compare their performance.

To the best of our knowledge, DKS is the first work that makes a compact association of deeply-supervised learning and knowledge distillation methodologies, enabling the transfer of currently learned knowledge between different layers in a deep CNN model. In the supplemental materials, we provide some theoretical analysis attempting to better understand DKS.

4. Experiments

In this section, we first apply DKS to train state-of-the-art CNNs on the CIFAR-100 [20] and ImageNet [37] classification datasets, and compare it with the standard training scheme and the Deeply-Supervised (DS) learning scheme. We then provide experiments for a deep analysis of DKS and more comprehensive comparisons. All algorithms are implemented with PyTorch [33]. For fair comparisons, the experiments of these three methods are conducted with exactly the same settings for data pre-processing, batch size, number of training epochs, learning rate scheduling, etc.

4.1. Experiments on CIFAR-100

CIFAR-100 dataset [20] contains 50000 training images and 10000 test images, where instances are 32×32 color images drawn from 100 object classes. We use the same data pre-processing method as in [10, 22]. For training, images are padded with 4 pixels to both sides first, and then 32×32 crops are randomly sampled from the padded images or their horizontal flips, and are finally normalized with the per-channel mean and std values. For evaluation, we report the error on the original-sized test images.

Backbone Networks and Implementation Details. We consider four state-of-the-art CNN architectures including: (1) ResNets [10] with depth 32 and 110; (2) DenseNets [17] with depth 40/100 and growth rate 12; (3) WRNs [47] with depth 28/28 and widening factor 4/10; (4) MobileNet [14] as used in [52]. We use the released code by the authors and follow the standard settings to train each backbone network. During training, for ResNets and MobileNet, we use SGD with momentum, and we set the batch size as 64, the weight decay as 0.0001, the momentum as 0.9 and the number of training epochs as 200. The initial learning rate is 0.1, and

it is divided by 10 every 60 epochs. For DenseNets, we use SGD with Nesterov momentum, and we set the batch size as 64, the weight decay as 0.0001, the momentum as 0.9 and the number of training epochs as 300. The initial learning rate is set to 0.1, and is divided by 10 at 50% and 75% of the total number of training epochs. For WRNs, we use SGD with momentum, and we set the batch size as 128, the weight decay as 0.0005, the momentum as 0.9 and the number of training epochs as 200. The initial learning rate is set to 0.1, and is divided by 5 at 60, 120 and 160 epochs. Inspired by [41, 16], we append two auxiliary classifiers to certain intermediate layers of these CNN architectures. 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 networks, a global average pooling layer and a fully connected layer. The differences are the number of building blocks and the number of convolutional filters (see supplementary materials for details). All models are trained on a server using 1 GPU. For each network, we run each method 5 times and report 'mean(std)' error rates.

Results Comparison. Results are summarized in Table 1 where baseline denotes the standard training scheme, and DS denotes the deeply-supervised learning scheme [41, 22] using our designed auxiliary classifiers. Generally, with our designed auxiliary classifiers, DS improves model accuracy in all cases compared to the baseline method, and its accuracy gain ranges from 0.08% to 0.92%. Comparatively, our method performs the best on all networks, bringing at least 0.67% and at most 3.08% accuracy gain to DS. As the network goes to much deeper (e.g., ResNet-110 and DenseNet-100)/much wider (e.g., WRN-28-10)/much smaller (e.g., MobileNet), our method also has noticeable accuracy improvements over all counterparts. These experiments clearly validate the effectiveness of the proposed method when training state-of-the-art CNNs.

4.2. Experiments on ImageNet

ImageNet classification dataset [37] is much larger than CIFAR-100 dataset. It has about 1.2 million training images and 50 thousand validation images, consisting of 1000 object classes. For training, images are resized to 256×256 first, and then 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 center crops of the resized validation data.

Backbone Networks and Implementation Details. We use popular ResNets as the backbone networks for evaluation. Specifically, ResNet-18, ResNet-50 and ResNet-152 are considered. All models are trained with SGD for 100 epochs. We set the batch size as 256, the weight decay as 0.0001 and the momentum as 0.9. The learning rate starts

Model	Method	Error(%)	Average gain(%)
ResNet (d=32)	baseline	29.97(0.33)	-
	DS	29.89(0.26)	0.08
	DKS	**26.81(0.36)**	**3.16**
ResNet (d=110)	baseline	27.66(0.60)	-
	DS	26.95(0.51)	0.71
	DKS	**24.98(0.35)**	**2.68**
DenseNet (d=40, k=12)	baseline	24.91(0.18)	-
	DS	24.46(0.22)	0.45
	DKS	**23.61(0.20)**	**1.30**
DenseNet (d=100, k=12)	baseline	20.92(0.31)	-
	DS	20.34(0.23)	0.58
	DKS	**19.67(0.29)**	**1.25**
WRN-28-4	baseline	21.39(0.30)	-
	DS	20.47(0.21)	0.92
	DKS	**18.91(0.08)**	**2.48**
WRN-28-10	baseline	18.72(0.24)	-
	DS	18.32(0.13)	0.40
	DKS	**17.24(0.22)**	**1.48**
WRN-28-10 (0.3 dropout)	baseline	18.64(0.19)	-
	DS	17.80(0.29)	0.84
	DKS	**16.71(0.17)**	**1.93**
MobileNet	baseline	23.60(0.22)	-
	DS	22.98(0.17)	0.62
	DKS	**21.26(0.16)**	**2.34**

Table 1: Accuracy comparison on the CIFAR-100 dataset. For each network, we run each method 5 times and report 'mean(std)' error rates. Our method achieves state-of-the-art accuracy when training each backbone network.

at 0.1, and is divided by 10 every 30 epochs. To show the compatibility of DKS with data augmentation methods, we train ResNet-18 and ResNet-50 with a simple data augmentation method, and train ResNet-152 with a more aggressive data augmentation method as in [41]. For each network, we add two auxiliary classifiers after the block Conv3_x and Conv4_x. The auxiliary classifiers are constructed with the same building block as in the backbone network. The differences are the number of residual blocks and the number of convolutional filters (see supplementary materials for details). All models are trained on a sever using 8 GPUs.

Results Comparison. Table 2 shows the results. Similar to the results on the CIFAR-100 dataset, on the ImageNet classification dataset, DS also shows minor accuracy improvements over the baseline models, even using our designed auxiliary classifiers. Its gain in Top-1/Top-5 accuracy is 0.60%/0.33%, 0.38%/0.11% and 0.46%/0.25% for ResNet-18, ResNet-50 and ResNet-152, respectively. These results are consistent with the results reported in [41]. Benefiting from the proposed synergy loss, DKS achieves the best performance which outperforms DS with a margin of 1.78%/1.25%, 1.56%/1.07% and 1.01%/0.41% in Top-1/Top-5 accuracy, respectively. Even using simple data augmentation, the ResNet-18/ResNet-50 model trained by our

Model	Method	Top-1/Top-5 Error(%)	Gain(%)
ResNet-18	baseline	31.06 / 11.13	-
	DS	30.46 / 10.80	0.60 / 0.33
	DKS	**28.68 / 9.55**	**2.38 / 1.58**
ResNet-50	baseline	25.47 / 7.58	-
	DS	25.09 / 7.47	0.38 / 0.11
	DKS	**23.53 / 6.40**	**1.94 / 1.18**
ResNet-152	baseline	22.45 / 5.94	-
	DS	21.99 / 5.69	0.46 / 0.25
	DKS	**20.98 / 5.28**	**1.47 / 0.66**

Table 2: Accuracy comparison on the ImageNet dataset.

method shows $1.75\%/0.48\%$ Top-1 accuracy gain against the models released at Facebook github[1], which are trained with much stronger data augmentations. Furthermore, it can be seen that the accuracy improvement from our method decreases slightly as network depth increases. Curves of Top-1 training and test error rates can be found in supplemental materials.

4.3. Ablation Study

Analysis of Auxiliary Classifiers. Given a backbone network, the questions of how to design auxiliary classifiers and where to place them are critically important for the deeply-supervised learning methods [22, 41] and our method. We perform experiments on the ImageNet classification dataset with ResNet-18 to study these two questions. To the first question, we compare our designed auxiliary classifiers and the relatively simple ones suggested in [22]. In the experiments, auxiliary classifiers are added on top of the block Conv3_x and Conv4_x. With simple auxiliary classifiers, DS introduces $1.17\%/0.80\%$ drop in Top-1/Top-5 accuracy. Comparatively, with our designed auxiliary classifiers, DS brings $0.60\%/0.33\%$ increase and DKS achieves $2.38\%/1.58\%$ gain. The training and test curves are shown in Fig. 3. We also perform extensive experiments on the CIFAR-100 dataset using ResNet-32 to analyze the effect of auxiliary classifiers with different levels of complexity to DS and our method. Results are shown in Table 3. With very simple auxiliary classifiers, DS shows accuracy drop and DKS further decreases model accuracy. Along with the increased complexity of auxiliary classifiers, DKS outperforms DS with improved margin. Please see supplementary materials for details. To the second question, we consider different settings by adding our designed auxiliary classifiers to at most three intermediate layer locations (including the block Conv2_x, Conv3_x and Conv4_x) of ResNet-18. Detailed results are shown in Table 4 where C1, C2, C3 and C4 denote the auxiliary classifiers connected on top of the last layer, the block Conv4_x, Conv3_x and Conv2_x, sequentially. From Table 4, we can make following observations: (1) With only one auxiliary classifier, an early location is better than a relatively deep location;

Aux.Classifiers	Error(%) (DS)	Error(%) (DKS)	Avg Gain(%) (DKS to DS)
AP+2FC	31.85(0.42)	35.09(0.54)	-3.24
AP+1Conv+2FC	30.24(0.05)	32.52(0.27)	-2.28
Narrow Blocks	29.52(0.30)	29.18(0.28)	0.34
Shallow Blocks	29.39(0.09)	28.69(0.28)	0.70
Ours	29.89(0.26)	26.81(0.36)	3.08

Table 3: Accuracy comparison of DKS and KD using auxiliary classifiers with different levels of complexity. The baseline ResNet-32 model trained on CIFAR-100 shows $29.97\%(0.33\%)$ 'mean(std)' error rates over 5 runs. In the table, AP denotes average pooling layer, Conv denotes convolutional layer and FC denotes fully connected layer.

Model	Top-1/Top-5 Error(%)	Gain(%)
baseline(C_1)	31.06 / 11.13	-
C_1C_2	29.64 / 10.09	1.42 / 1.04
C_1C_3	29.30 / 9.86	1.76 / 1.27
C_1C_4	29.36 / 9.91	1.70 / 1.22
$C_1C_2C_3$	**28.68 / 9.55**	**2.38 / 1.58**
$C_1C_2C_3C_4$	29.00 / 9.79	2.06 / 1.34

Table 4: Accuracy gains of DKS with auxiliary classifiers connected to different intermediate layers of ResNet-18.

Model	Top-1/Top-5 Error(%)	Gain(%)
C_1	31.06 / 11.13	2.38 / 1.58
C_2	30.69 / 11.05	3.23 / 2.16
C_3	31.89 / 11.51	2.39 / 1.68

Table 5: Accuracy gains of DKS training against separate training of each individual auxiliary classifier connected to the corresponding intermediate layer of ResNet-18.

(2) Adding two or all of three auxiliary classifiers obtains larger gain than adding only one; (3) Adding C4 connected to an earlier intermediate layer into the combination of C2 and C3 decreases its accuracy. According to these results, we choose to add C2 and C3 for all experiments on the ImageNet classification dataset. In addition, we also analyze whether DKS is beneficial to auxiliary supervision branches or not. To this end, we train each individual auxiliary supervision branch separately, and compare it with the corresponding one trained with DKS. According to the results shown in Table 5, we can see that our method also brings obvious accuracy gain to each auxiliary supervision branch.

Comparison of Knowledge Matching Strategies. We also compare the performance of three pairwise knowledge matching strategies shown in Fig. 2. Experiments are conducted on the ImageNet classification dataset with ResNet-18 using our best auxiliary classifier setting just discussed. Compared with the baseline model, our method obtains $0.50\%/0.45\%$, $2.22\%/1.19\%$ and $2.38\%/1.58\%$ increase in Top-1/Top-5 accuracy by using the top-down, bottom-up and bi-directional pairwise knowledge matching strategies,

respectively. As the bi-directional strategy shows the best results, we adopt it as the default choice for DKS. Another interesting observation is that they all achieve improved results compared with the baseline method, showing that the pairwise knowledge transfer among the supervised classifiers connected to the backbone network is really helpful in regularizing model training.

DKS on Very Deep Network. Next, we conduct a set of experiments to analyze the performance of DKS on very deep CNNs. In the experiments, we consider the training of a ResNet variant with 1202 layers [10] on the CIFAR-100 dataset. Unlike auxiliary classifiers used in the other experiments, we study DKS with shallow but wide auxiliary classifiers in this experiment (see supplementary materials for details). Remarkably, although the network depth is significantly increased, the average accuracy of the models trained with our method is 69.54%, showing a $3.76\%/2.04\%$ margin compared with the baseline/DS method.

DKS with Strong Regularization. In order to explore the compatibility of DKS and strong regularization methods, we conduct the experiments on the CIFAR-100 dataset following [47]. We add a dropout layer with a ratio of 0.3 after the first layer of every building block of WRN-28-10. The results are shown in Table 1. It can be seen that the models trained with DKS show a mean accuracy of 16.71%, bringing 0.53% gain to the DKS case without dropout.

DKS vs. Knowledge Distillation. Further, we compare the performance of DKS, Knowledge Distillation (KD) and its variants. Experiments are conducted on the ImageNet classification dataset using ResNet-18. We use a pre-trained ResNet-50 model as the teacher and consider three different KD settings: (1) KD on C1 (the standard KD as in [11]); (2) KD on C1+DS; (3) KD on C2C3+DS. We evaluate temperature values of [1, 2, 5, 10, 20] and choose the best choice for each KD setting. From the results shown in Table 6, we can make following observations: (1) KD can improve model training in all cases; (2) Distilling learnt knowledge into auxiliary classifiers connected to the earlier layers has small gain to DS, and larger gain can be achieved by applying KD on auxiliary classifiers added to the deep layers; (3) DKS achieves the best performance, showing the effectiveness of the proposed synergy loss.

DKS on Noisy Data. Finally, we explore the capability of our method to handle noisy data. Following [49], we use CIFAR-10 dataset and DenseNet (d=40, k=12) 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. Results show that the average accuracy of the baseline model decreases from 94.62% to 82.07%, while DS further decreases it to 80.47% and ours is 83.73%, when 50% training data are corrupted. As the ratio of the corrupted training data goes to 80%, our model still has 67.19% mean accuracy, outperforming the

Model	Top-1/Top-5 Error(%)	Gain(%)
baseline	31.06 / 11.13	-
DS	30.46 / 10.80	0.60 / 0.33
KD on C1 [11]	29.71 / 10.33	1.35 / 0.80
KD on C1+DS	29.38 / 10.10	1.68 / 1.03
KD on C2C3+DS	30.32 / 10.64	0.74 / 0.49
DKS	**28.68 / 9.55**	**2.38 / 1.58**

Table 6: Accuracy comparison of DKS, KD and its variants on the ImageNet classification dataset using ResNet-18.

baseline/DS with a margin of $2.51\%/2.27\%$. These experiments partially show that our method has good capability to suppress noise disturbance and behaves like a strong regularizer.

4.4. Discussion

Although the CNNs used in our experiments have sophisticated building block designs which increase the flexibility of feature connection path and show stable convergence, our DKS can impressively improve their training in comparison to the standard training scheme and DS. This is first benefitted from adding proper auxiliary classifiers to the intermediate layers of the network, but we believe it is more benefitted from the proposed synergy loss which enables comprehensive pairwise knowledge matching among all supervised classifiers connected to the network, enhancing learnt feature representation. On the other hand, we observe substantial time increase for model training. For instance, a baseline ResNet-18 model is trained for about 20 hours on a server with 8 GPUs (an SSD is used to accelerate data accessing process), while our method needs about 37 hours, nearly doubling the training time. Besides, the training time for DS is almost the same as our method. We believe this mainly correlates with the number of auxiliary classifiers and their complexity. Therefore, there is a trade-off between the required training time and the expected accuracy improvement. Achieving larger accuracy gain needs auxiliary classifiers to be more complex, while simple ones usually worsen model accuracy. Since increasing the number of auxiliary classifiers does not always bring higher accuracy gain, as shown in our ablation study, we think the current increase in training time is reasonable. More importantly, all auxiliary classifiers are discarded at inference phase, thus there is no extra computational cost.

5. Conclusion

In this paper, we revisit the deeply-supervised learning research and propose a new optimization scheme called DKS for training deep CNNs. It introduces a novel synergy loss which regularizes the training by considering dense pairwise knowledge matching among all supervised classifiers connected to the network. Extensive experiments on two well-known image classification tasks validate the effectiveness of our method.

References

[1] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng. Dual path networks. In *NIPS*, 2017. 1, 3

[2] T. Dozat. Incorporating nesterov momentum into adam. In *ICLR-W*, 2016. 3

[3] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(7):2121–2159, 2011. 3

[4] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 2016. 3

[5] N. C. Garcia, P. Morerio, and V. Murino. Modality distillation with multiple stream networks for action recognition. In *ECCV*, 2018. 2

[6] G. Ghiasi, T.-Y. Lin, and Q. V Le. Dropblock: A regularization method for convolutional networks. In *NIPS*, 2018. 3

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1

[8] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *ICML*, 2013. 3

[9] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015. 1, 3

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 2, 3, 5, 8

[11] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 2, 5, 8

[12] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin. Lifelong learning via progressive distillation and retrospection. In *ECCV*, 2018. 3, 5

[13] A. G. Howard. Some improvements on deep convolutional neural network based image classification. *arXiv preprint arXiv:1312.5402*, 2013. 3

[14] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 1, 2, 3, 5

[15] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *CVPR*, 2018. 1, 3

[16] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Q. Weinberger. Multi-scale dense networks for resource efficient image classification. In *ICLR*, 2018. 2, 4, 5, 6

[17] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 1, 2, 3, 5

[18] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 1, 3

[19] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 3

[20] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Handbook of Systemic Autoimmune Diseases*, 2009. 5

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 3

[22] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *AISTATS*, 2015. 2, 3, 4, 5, 6, 7

[23] C. Li, M. Z. Zia, Q.-H. Tran, X. Yu, G. D. Hager, and M. Chandraker. Deep supervision with intermediate concepts. *arXiv preprint arXiv:1801.03399*, 2018. 2

[24] Z. Li and D. Hoiem. Learning without forgetting. In *ECCV*, 2016. 3, 5

[25] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy. Progressive neural architecture search. In *ECCV*, 2018. 3

[26] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1

[27] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *ECCV*, 2018. 1, 3

[28] A. Mishra and D. Marr. Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy. In *ICLR*, 2018. 2, 5

[29] A. Mosinska, P. Márquez-Neila, M. Kozinski, and P. Fua. Beyond the pixel-wise loss for topology-aware delineation. In *CVPR*, 2018. 2

[30] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010. 3

[31] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016. 2

[32] X. Nie, J. Feng, and S. Yan. Mutual learning to adapt for joint human parsing and pose estimation. In *ECCV*, 2018. 3

[33] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017. 5

[34] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean. Efficient neural architecture search via parameter sharing. In *ICML*, 2018. 3

[35] S. Qiao, W. Shen, Z. Zhang, B. Wang, and A. Yuille. Deep co-training for semi-supervised image recognition. In *ECCV*, 2018. 3, 5

[36] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. Fitnets: Hints for thin deep nets. In *ICLR*, 2015. 2, 5

[37] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and F.-F. Li. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 1, 5, 6

[38] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018. 1, 3

[39] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1, 3

[40] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 3

[41] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, et al. Going deeper with convolutions. In *CVPR*, 2015. 2, 3, 4, 6, 7

[42] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, 2014. 1

[43] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus. Regularization of neural networks using dropconnect. In *ICML*, 2013. 3

[44] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 1, 3

[45] S. Xie and Z. Tu. Holistically-nested edge detection. In *ICCV*, 2015. 2

[46] J. Yim, D. Joo, J. Bae, and J. Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *CVPR*, 2017. 2, 5

[47] S. Zagoruyko and N. Komodakis. Wide residual networks. In *BMVC*, 2016. 1, 2, 3, 5, 8

[48] S. Zagoruyko and N. Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *ICLR*, 2017. 2, 5

[49] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. 1, 3, 8

[50] H. Zhang, H. Wu, W. Sun, and B. Zheng. Deeptravel: a neural network based travel time estimation model with auxiliary supervision. *arXiv preprint arXiv:1802.02147*, 2018. 2

[51] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 2018. 1, 3

[52] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu. Deep mutual learning. In *CVPR*, 2018. 2, 5

[53] Z. Zhang, X. Zhang, C. Peng, D. Cheng, and J. Sun. Exfuse: Enhancing feature fusion for semantic segmentation. In *ECCV*, 2018. 2, 4

[54] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *CVPR*, 2017. 2

[55] X. Zhuang, H. Luo, X. Fan, W. Xiang, Y. Sun, Q. Xiao, W. Jiang, C. Zhang, and J. Sun. Alignedreid: Surpassing human-level performance in person re-identification. In *CVPR*, 2018. 3

[56] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017. 3