Data-Distortion Guided Self-Distillation for Deep Neural Networks

Ting-Bing Xu,[1,2] **Cheng-Lin Liu**[1,2,3]
[1]National Laboratory of Pattern Recognition, Institute of Automation of Chinese Academy of Sciences, Beijing, China
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China
[3]CAS Center for Excellence of Brain Science and Intelligence Technology, Beijing, China
{tingbing.xu, liucl}@nlpr.ia.ac.cn

Abstract

Knowledge distillation is an effective technique that has been widely used for transferring knowledge from a network to another network. Despite its effective improvement of network performance, the dependence of accompanying assistive models complicates the training process of single network in the need of large memory and time cost. In this paper, we design a more elegant *self-distillation* mechanism to transfer knowledge between different distorted versions of same training data without the reliance on accompanying models. Specifically, the potential capacity of single network is excavated by learning consistent global feature distributions and posterior distributions (class probabilities) across these distorted versions of data. Extensive experiments on multiple datasets (i.e., CIFAR-10/100 and ImageNet) demonstrate that the proposed method can effectively improve the generalization performance of various network architectures (such as AlexNet, ResNet, Wide ResNet, and DenseNet), outperform existing distillation methods with little extra training efforts.

Introduction

Recent years have been witnessed significant progress in the performance of various pattern recognition tasks (He et al. 2016a; Huang, Liu, and Weinberger 2017), relying on deep convolutional neural networks. However, state-of-the-art models usually involve very deep networks with tremendous parameters and a large number of floating point operations, which hinders them from real-world applications on low-resource devices, such as smartphones and wearable gadgets. To alleviate this bottleneck, a variety of network compression methods such as low-rank decomposition (Denton et al. 2014), weight pruning (Han, Mao, and Dally 2016) and structured sparsity (Liu et al. 2017), have been exploited to obtain a small model that can work as well as a trained large model while reducing the time and memory space in the inference process effectively.

From a network compression perspective, a large network can be compressed to the corresponding small network of similar accuracy (Denton et al. 2014; Han, Mao, and Dally 2016) by layer-wise decomposition or iteratively pruning. Though small network is capable of preserving complex fitting function with reasonable performance, compared to large network, it is harder to train from scratch and excavate own potential capacity to a more desired solution (robust minima). To better train a small network, model distillation, i.e., *teacher-to-student* mechanism that directly trains a student network to inherit the knowledge (e.g., class probabilities (Hinton, Vinyals, and Dean 2015), logits (Ba and Caruana 2014), intermediate feature maps (Romero et al. 2015), attention map (Zagoruyko and Komodakis 2017)) of a deeper or more complex teacher network, has been introduced. In this way, the student model can approximate the capacity of the powerful pre-trained teacher model by absorbing the extra supervised information. Instead of knowledge transfer from a static teacher to a student, a new *student-to-student* mechanism is designed in (Zhang et al. 2018b) to allow an ensemble of students to teach each other with deep mutual learning throughout the whole training process. This enables each student in such a peer-teaching based scenario to learn a better solution than directly supervised training of student or distilling from a fixed pre-trained teacher model.

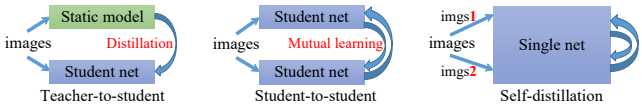

Figure 1: The diagrams of three distillation mechanisms.

Although the above two methods of model distillation can boost the accuracy and generalization ability of student network, they have some obvious drawbacks: 1) The whole training process is expensive due to the involvement of either a cumbersome teacher model or multiple student networks; 2) A teacher network that overfits the training set can be less effective due to the limitation of valuable information beyond the common hard label (Anil et al. 2018); 3) It remains unclear how a teacher model boost another model in principle, considering the facts that slight mistakes made by the teacher model may be helpful for training the student model (Hinton, Vinyals, and Dean 2015), and two entirely same student networks are also able to teach each other (Zhang et al. 2018b). These shortages motivate us to develop a new mechanism that directly optimizes student network to reach a more desired solution from the raw training data without going through another teacher or intermediate network.

In this paper, we propose an elegant training mechanism, namely *self-distillation* (also called self-teaching) in Figure 1, to efficiently optimize single network from the consistent distributions of data representations without the assistance of other models. Usually, the common data-distortion techniques (e.g., random mirror/cropping, rotation, multiple scales) only enlarge additional virtual samples from the neighborhood of existing training samples, which makes network see more distorted training samples to improve model generalization (Simard et al. 1996). In another viewpoint, good generalization requires the model to be capable of keeping the similar or invariant class probability predictions (posterior distributions) and global feature representations (feature distributions) across different distorted versions of same training data. Thus, to further excavate the potential capacity of single network, we design a novel data-distortion guided self-distillation mechanism to minimize the discrepancy of posterior probability and feature distributions for data-to-data knowledge transfer. More specifically, we utilize the Kullback Leibler (KL) divergence constraint to measure the match between the posterior probability distributions of different distorted versions, in a similar way as the knowledge distillation between different models (Hinton, Vinyals, and Dean 2015), and adopt the empirical Maximum Mean Discrepancy (MMD) (Long et al. 2015) as the nonparametric metric for measuring the consistency of feature distributions between these distorted versions. While for training the baseline classification task, we still keep the conventional cross entropy loss term (i.e., softmax loss) with hard labels. The above three loss terms can be integrated together as the optimization objective for implementing self-distillation of single network from scratch in end-to-end manner. This new mechanism can drive single network to automatically learn more inherent representations for generalization, effectively inhibit learned features toward single direction reliance (Morcos et al. 2018).

To verify the effectiveness of self-distillation, we perform extensive experiments on current state-of-the-art networks ranging from AlexNet (Krizhevsky, Sutskever, and Hinton 2012) to ResNet (He et al. 2016a), WideResNet (Zagoruyko and Komodakis 2016), DenseNet (Huang, Liu, and Weinberger 2017). Experimental results demonstrate that our self-distillation mechanism substantially improves the generalization performance on any of the network architecture. Compared to other mechanisms in Figure 1, we achieve higher accuracy with several times fewer parameters.

Related Works

Model Compression. Deep network compression is aimed to remove redundant parameters and computations of a complex model while preserving original accuracy. Denil et al. (Denil et al. 2013) show that huge redundancies exist in weight parameters of deep networks and are possible to be removed. A set of tangible methods have been designed, such as the low-rank decomposition of kernel tensor (Denton et al. 2014), low-threshold pruning of non-structured sparsity (Han, Mao, and Dally 2016) and parameter sharing of hash trick (Chen et al. 2015). To achieve a larger speedup of network, some structured pruning methods (Liu et al. 2017; He, Zhang, and Sun 2017; Li et al. 2017) have been developed to remove a group of weights or a whole feature map at a time without the requirement of specially designed software/hardware accelerators. However, these methods involve complicated training processes such as layer-wise decomposition, iteratively pruning, and repetitiously fine-tuned procedures after each decomposition or pruning step.

Knowledge Distillation. Knowledge distillation is to train student network to reach a better solution via extra supervised information from pre-trained teacher model. The basic idea was proposed over a decade ago in (Bucila, Caruana, and Niculescu-Mizil 2006) but was refocused recently in (Hinton, Vinyals, and Dean 2015). Its main goal is to transfer dark knowledge from a powerful teacher network such as deep (Urban et al. 2017; Chen et al. 2017), wide (Romero et al. 2015), or an ensemble of models (Hinton, Vinyals, and Dean 2015) to a student network (shallow or thin). Trained in this way, the student network inherits the teacher's similar properties such as class probability distribution, the logits (the input to softmax), and intermediate representations to achieve better performance than the network independently trained with hard labels. In a similar manner, multiple student networks (Furlanello et al. 2018; Zhang et al. 2018b) can also teach each other via mutual learning. By knowledge distillation, either teacher-to-student or student-to-student, the training process is expensive due to the reliance on auxiliary models.

Data Distortion. Data distortion can be taken as a tactic to boost the generalization ability of model (Simard et al. 1996) by alleviating data over-fitting, which is a critical influencing factor of deep networks. The common techniques such as random mirror/cropping (Krizhevsky, Sutskever, and Hinton 2012), scale jittering or color altering (He et al. 2016a), and random erasing (Zhong et al. 2017), are mainly aimed to achieve invariant class for different distorted instances of same training data. Unfortunately, the deep networks may be prone to memorize more training data (Morcos et al. 2018) and result in worse generalization on new test data. To learn more discriminative features from data, some new loss terms have been designed to enforce the learned features toward intra-class compactness and inter-class separability, such as contrastive loss (Hadsell, Chopra, and LeCun 2006), triplet loss (Schroff, Kalenichenko, and Philbin 2015) and center loss (Wen et al. 2016). However, their training processes are very cumbersome and unstable because they need a careful selection of training pairs/triplets from tremendous training samples, while the center loss may become ineffective for many tasks because deep features of each class are not necessarily clustered in single center. Recently, mixup (Zhang et al. 2018a) is proposed to construct the inter-class relation via different combinations of training samples. This is complementary to our data-distortion guided self-distillation and can be combined for further improvement.

Self-distillation

Self-distillation is different from *self-training* (Rosenberg, Hebert, and Schneiderman 2005; Radosavovic et al. 2018), which has been widely used in semi-supervised problem to exploit large-scale unlabeled data in supervised learning. In

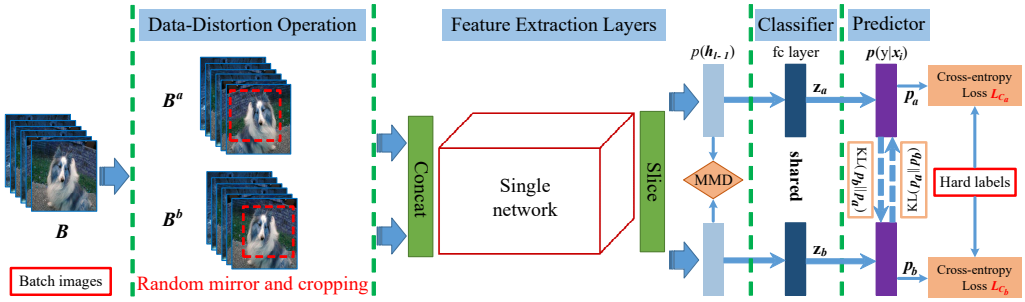

Figure 2: The proposed *self-distillation* mechanism for the training of single network, consisting of data-distortion operation, feature extraction layers, classifier of fully-connected (fc) layer, and predictor of softmax. Three loss terms (MMD loss with global features, KL divergence loss with predictive probabilities, and cross entropy loss with hard labels) are integrated for training a single network. The fc layer is shared by two-branch streams, and does not incur extra parameters.

contrast, self-distillation is a fully-supervised manner to exploit the potential capacity of single network only from labeled data without the need of assistive models.

The overall training process is shown in Figure 2, which is composed of four parts: 1) A data-distortion operation generates two-branch distorted instances from the same training data, 2) a global feature extraction through convolutional layers obtains their representation vectors as feature distributions, then the MMD metric reduces the discrepancy of them, 3) a fc layer classifier returns the logits ($\boldsymbol{z}$), and 4) a softmax predictor yields their class probabilities as posterior distributions, then the KL divergence measures the consistency of them. Finally, we can combine the above two constraints (MMD and KL) with the conventional cross entropy loss computed by hard labels, to implement the whole training optimization of single network in end-to-end manner.

Compared to the training of a standard baseline network, the self-distillation training does not add the new data-distortion manner and the extra network parameters. Everytime it uses the standard single network to tackle two distorted instances ($\boldsymbol{x}_{ai}$ and $\boldsymbol{x}_{bi}$) produced by the common random mirror and cropping for each training sample ($\boldsymbol{x}_i$). Due to the finite statuses of mirror and slightly cropping, all the distorted instances of each training sample repetitiously appear in random order for the training processes of baseline and self-distillation. Note that the data-distortion and two-branch operations are not be used during the test stage.

Problem Formulation

We define the D^s=$\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$ as a labeled source dataset from K classes, where the N is the total number of training samples, and y_i is the corresponding hard label of input sample $\boldsymbol{x}_i$. For a mini-batch input B=$\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$ in Figure 2 during training stage, we utilize the same data-distortion $\phi(B)$ manner (i.e., random mirror and cropping) two twice to obtain two-branch distorted samples B^a=$\{(\boldsymbol{x}_{a1}, y_1), \ \ldots, (\boldsymbol{x}_{an}, y_n)\}$ and B^b=$\{(\boldsymbol{x}_{b1}, y_1), \ldots, (\boldsymbol{x}_{bn}, y_n)\}$, where the hard labels $\{y_1, \ldots, y_n\}$ are shared for the two-branch data. For a network with l layers, the first $l-1$ layers are all feature extraction layers, and the $\mathbf{h}(l-1) = f(\mathbf{W}\boldsymbol{x})$ will be global feature vector of input data $\boldsymbol{x}$, where $\mathbf{W}$ is weight projection matrix of all $l-1$ layers. In the following of this paper, $p(\mathbf{h}(\boldsymbol{x}))$ denotes the feature distribution of one sample, while $p(y|\boldsymbol{x})$ represents the posterior distribution. Except for predictions of hard labels, we hope to minimize the discrepancy of feature/posterior distributions across two distorted versions.

The MMD metric for global feature distributions

To match the feature distribution, a metric between representation vectors of two-branch distorted versions need to be defined. Here, we adopt the empirical MMD as a nonparametric metric that has been widely used in domain adaptation (Long et al. 2015) to measure the discrepancy of distributions. Previous researches demonstrate that MMD possesses better robustness and efficiency in computation and optimization (Quadrianto, Petterson, and Smola 2009). Our goal is to make $p(\mathbf{h}(\boldsymbol{x}_{ai}))$ and $p(\mathbf{h}(\boldsymbol{x}_{bi}))$ to be close, and the empirical MMD in Figure 2 can be constructed as follows:

$$\text{MMD}_{emp} = \| \frac{1}{n}\sum_{i=1}^{n}\mathbf{h}(\boldsymbol{x}_{ai}) - \frac{1}{n}\sum_{i=1}^{n}\mathbf{h}(\boldsymbol{x}_{bi}) \|_2^2, \quad (1)$$

where n is the number of samples in a mini-batch. It is equivalent to a l_2 loss on mean vector of global features, which effectively reduces the churn caused by certain outlier sample. In backpropagation algorithm, its gradient need to be transmitted layer by layer and can be calculated as:

$$\nabla_{\mathbf{h}}\text{MMD}_{emp} = \begin{cases} \frac{2}{n^2}(\Sigma_{i=1}^{n}\mathbf{h}(\boldsymbol{x}_{ai}) - \Sigma_{i=1}^{n}\mathbf{h}(\boldsymbol{x}_{bi})), \boldsymbol{x} \in B^a \\ -\frac{2}{n^2}(\Sigma_{i=1}^{n}\mathbf{h}(\boldsymbol{x}_{ai}) - \Sigma_{i=1}^{n}\mathbf{h}(\boldsymbol{x}_{bi})), \boldsymbol{x} \in B^b \end{cases}$$

Thus, this term can be easily incorporated into any existing objective function to jointly optimize network parameters.

The KL divergence for posterior distributions

Usually, we need to add the softmax layer after the last fc-layer classifier to produce the posterior probability $p(y|\boldsymbol{x})$ for an arbitrary input $\boldsymbol{x}$, and its formulation is defined as:

$$p(y=k|\boldsymbol{x}) = softmax(z_k) = \frac{exp(z_k)}{\Sigma_{i=1}^{K} exp(z_i)}, \quad (2)$$

where the z_k is the predicted value of class k from fc layer classifier, and as logit to be fed into subsequent softmax

layer. Then, the posterior distribution vector $\boldsymbol{p}(\boldsymbol{x}) = \{p(y = 1|\boldsymbol{x}), \ldots, p(y = K|\boldsymbol{x})\}$ can be written for all K categories of this sample, which is a K-dimensional vector and each dimension represents a posterior probability of certain category. To reach the consistent posterior distribution between the two-branch distorted versions, we adopt the KL divergence in Figure 2 as the metric to measure the match as follows:

$$\begin{aligned} D_{KL}(\boldsymbol{p}_b||\boldsymbol{p}_a) &= -\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{p}(\boldsymbol{x}_{bi})\log\frac{\boldsymbol{p}(\boldsymbol{x}_{ai})}{\boldsymbol{p}(\boldsymbol{x}_{bi})} \\ &= H(\boldsymbol{p}_b, \boldsymbol{p}_a) - H(\boldsymbol{p}_b), \end{aligned} \tag{3}$$

where it denotes the KL distance from $\boldsymbol{p}_a$ to $\boldsymbol{p}_b$, and $\boldsymbol{p}_b$ is the target probabilities as soft labels to supervise the learning of the predicted probabilities $\boldsymbol{p}_a$. Thus, the gradient of KL distance can be calculated as follows:

$$\partial D_{KL}(\boldsymbol{p}_b||\boldsymbol{p}_a)/\partial z_k = \partial H(\boldsymbol{p}_b, \boldsymbol{p}_a)/\partial z_k - \partial H(\boldsymbol{p}_b)/\partial z_k,$$

where $H(\boldsymbol{p}_b, \boldsymbol{p}_a)$ denotes the cross entropy and $H(\boldsymbol{p}_b)$ is the empirical entropy (it is constant value, and so the gradient is zero). Meanwhile, we construct another KL distance $D_{KL}(\boldsymbol{p}_a||\boldsymbol{p}_b)$ to supervise the learning of predicted probabilities $\boldsymbol{p}_b$. The above two KL terms can be integrated together to update network parameters with gradient descent optimization, and the gradient is written as follows:

$$\partial D_{KL}/\partial z_k = \begin{cases} \partial D_{KL}(\boldsymbol{p}_b||\boldsymbol{p}_a)/\partial z_k = \\ p(y = k|\boldsymbol{x}_{ai}) - p(y = k|\boldsymbol{x}_{bi}), \boldsymbol{x} \in B^a \\ \\ \partial D_{KL}(\boldsymbol{p}_a||\boldsymbol{p}_b)/\partial z_k = \\ p(y = k|\boldsymbol{x}_{bi}) - p(y = k|\boldsymbol{x}_{ai}), \boldsymbol{x} \in B^b \end{cases}$$

This process can be taken as the mutual learning (Zhang et al. 2018b) across two-branch distorted versions.

To better learn the inter-class relation, the temperature parameter T in (Hinton, Vinyals, and Dean 2015) is introduced to soften the output probability $p(y = k|\boldsymbol{x}) = softmax(z_k/T)$. The higher temperature will returns softer labels so that some categories with near-zero probabilities will not be ignored by the above KL term. Trained in this way, it will tell the network extra inter-class information, for instance, 'car' should share more approximated category probability with 'truck' than with 'cat'. Thus, the single network not only learns the consistent posterior distribution, but also constructs the vicinity relation of different categories as the same efficacy in knowledge distillation of different models (Hinton, Vinyals, and Dean 2015).

Optimisation

Our data-distortion guided self-distillation mainly consists of three types of loss terms to update parameters of single network throughout the whole training process. Firstly, we maintain the cross entropy softmax loss for predictions of hard labels, and this term can be defined as follows:

$$L_{C_a} = -\frac{1}{n}\sum_{i=1}^{n}\log p(y = y_i|\boldsymbol{x}_{ai}), \tag{4}$$

where $\boldsymbol{x}_{ai}$ and y_i represent a input sample and the hard label of current sample, respectively. The $p(y = y_i|\boldsymbol{x}_{ai})$ is the posterior probability of predicting the hard label. In Figure 2, L_{C_a} and L_{C_b} denote the respective softmax loss from distorted versions (B^a and B^b). Secondly, we inject the temperature parameter T into the KL loss term in Eq. 3 to learn the consistent posterior distribution and inherent inter-class relation. By integrating Eq. 1, Eq. 3 and Eq. 4, we can construct the final optimization objective of single network:

$$\begin{aligned} L_{net} =& (L_{C_a} + L_{C_b}) + \lambda(D_{KL}(\boldsymbol{p}_b||\boldsymbol{p}_a) + D_{KL}(\boldsymbol{p}_a||\boldsymbol{p}_b)) \\ &+ \mu\text{MMD}_{emp}, \end{aligned} \tag{5}$$

where λ and μ are posterior and feature distributions regulation parameters. We usually set λ=1 as referenced in (Zhang et al. 2018b), which means the equivalent importance between the prediction of hard labels and the learning of consistent class probabilities across the two-branch distorted versions of same input. However, the temperature parameter T in KL term need to be further tuned for various tasks, and the higher T denotes the more prone to emphasize the related attributes of different categories. For hard tasks (i.e., large-scale classification (Hinton, Vinyals, and Dean 2015) and object detection (Chen et al. 2017)), the large T may bring more noises which injure the right updated directions of parameters due to the highly prediction errors on these tasks. When W_{ij} is denoted as one element in the network weight matrix $\mathbf{W}$, we can combine the above gradient of each loss term to calculate the final partial derivative as:

$$\begin{aligned} \frac{\partial L_{net}}{\partial W_{ij}} =& \frac{(L_{C_a} + L_{C_b})}{\partial W_{ij}} + \lambda\frac{D_{KL}}{\partial z_k}\cdot\frac{z_k}{\partial W_{ij}} \\ &+ \mu(\nabla_{\mathbf{h}}\text{MMD}_{emp})^T\frac{\partial \mathbf{h}}{\partial W_{ij}}, \end{aligned} \tag{6}$$

where $\partial\mathbf{h}/\partial W_{ij}$ is a vector consisting of partial derivatives of each element in $\mathbf{h}$ with respect to W_{ij}. For any given network architecture, we can easily compute each weight gradient via Eq. 6 and optimize the whole network with the regular stochastic gradient descent (SGD).

Experiments

To evaluate the performance of the proposed data-distortion guided self-distillation training mechnism, we conducted extensive experiments on the current state-of-the-art networks (i.e., AlexNet, ResNet, Wide ResNet and DenseNet) and several benchmark datasets (i.e., CIFAR-10/100 and ImageNet). And for manifesting the effectiveness of self-distillation mechanism itself, we only adopt the simple and most common data-distortion techniques: random mirror and cropping in all the experiments for various network architectures. Trained in this way, we hope to produce considerable performance gains only from simple data-distortion guidance to display effectiveness and availability of the mechanism itself. All the experiments are performed with the high-efficiency caffe (Jia et al. 2014) platform.

Mechanism	Student net	Test error	Parameters	Comments
Independently training (IT)	ResNet-32	31.01	0.5M	single student (stu.)
Teacher-to-student (TS)		30.52	(0.5+36.5)M	teacher: WRN-28-10
Student-to-student (SS)		29.27	(0.5+36.5)M	another stu.: WRN-28-10
Student-to-student (SS)		28.81	(0.5+0.5)M	another stu.: ResNet-32
Self-distillation (**SD**)		**28.22**	0.5M	single stu.

Table 1: Comparison with model distillation on CIFAR-100 (error rate (%)). WRN denotes the WideResNet.

Datasets and Settings

Datasets. We follow state-of-the-art networks on CIFAR datasets (Krizhevsky and Hinton 2009). The **CIFAR-10** consists of 32×32 colour images in 10 classes including 50000 training samples and 10000 test samples, where each class has 5000 training data. The **CIFAR-100** is a more challenging recognition task, which has more classes with fewer samples on each class. The training and test sets are also 50000 and 10000 colored natural scene images (32×32 pixels each) drawn from 100 classes. For data preprocessing, we find that the previous ZCA whitening has a negative effect on final accuracies of current popular networks, which is also confirmed in (Zagoruyko and Komodakis 2017). Thus, we only use the simpler channel means and standard deviations to normalize data. For data-distortion operation, we follow a standard strategy that is widely used for the two datasets (He et al. 2016a; Zagoruyko and Komodakis 2016): training images are padded 4 pixels with zero-value on each side, and randomly crop a 32×32 region from the padded image or its mirror flip; test images maintain the original shape and size. Additionally, we also follow the classical AlexNet (Krizhevsky, Sutskever, and Hinton 2012) and ResNet-18 (He et al. 2016a) networks on the large-scale **ImageNet-2012** dataset (Russakovsky et al. 2015) that contains about 1.3 million training images and 50000 validation images from 1000 classes.

Implementation Details. We implement training procedures of all the networks according to the specific experimental settings of previous literature, and evaluate the performance improvements between the training processes of self-distillation and original baseline. Specifically, we perform training processes of ResNet-32/110 (He et al. 2016a) or pre-activation ResNet-18/34 (He et al. 2016b) as implemented in (Zhang et al. 2018b), where it sets mini-batch size to 64, weight decay to 10^{-4} and initial learning rate to 0.1 (dropped by 0.1 every 60 epochs and trained for 200 epochs). For WideResNet (Zagoruyko and Komodakis 2016), we follow original configurations that set 128 samples per mini-batch, weight decay to 5×10^{-4} and initial learning rate to 0.1 (dropped by 0.2 after 60, 120 and 180 epochs and trained for 200 epochs). For DenseNet (Huang, Liu, and Weinberger 2017), we adopt the memory-efficient code [1] and follow the original settings that they use weight decay of 10^{-4}, mini-batch size of 64 for 300 epochs and initial learning rate of 0.1 (divided by 10 at 50% and 75% of the total number of training epochs). All the networks are trained using Nesterov momentum (Sutskever et al. 2013) of 0.9 and weight initialization of "msra" in (He et al. 2015).

[1] Code: https://github.com/Tongcheng/caffe/.

Comparison with model distillation mechanisms

As one of our aims of self-distillation mechanism is to overcome the expensive computation in training of model distillation accompanied with assistive models while preserving the generalization performance, we give a detailed comparison to model distillation mechanisms including teacher-to-student and student-to-student. The first mechanism need fixed soft labels produced by a powerful pre-trained teacher model for supervised learning. The second mechanism involves multiple student networks to learn from data and teach each other simultaneously. In contrast, our self-distillation does not require any auxiliary model or network to jointly optimize the single student network.

Table 1 compares our self-distillation (**SD**) with other training mechanisms. As expected the previous model distillation actually improves the student performance compared to independently training the student with hard labels. Furthermore, the deep mutual learning between both of students achieves better accuracy improvement than the teaching from a static teacher, and even two entirely same student architectures have better results. It indicates that the results of model distillation are related to network architectures. However, it is still unclear what the ideal architecture properties of a teacher or another student is. Unlike theirs, our SD learns some more inherent representations only from single student itself by the training of consistent feature and posterior distributions. Finally, the SD achieves a larger performance gain without additional parameters or models. It verifies that the SD indeed excavates the potential capacity of existing parameters to effectively improve the generalization ability of single network, which drives single model to reach a lower generalization error.

Self-distillation for network compression

Model distillation can result in compressed network architecture, by training small compact network with the assistance of powerful pre-trained teacher network. However, if we independently train this compact network with an off-the-shelf optimization scheme, it will yield a considerable gap between accuracies of small and large networks. To mitigate this issue, our self-distillation mechanism directly optimizes the single compact network with the aid of data-to-data knowledge transfer. Table 2 shows evident performance improvements of compact networks from ResNet, WideResNet and DenseNet by self-distillation. These popu-

Dataset	Compact networks				Complex networks			
	network	depth	params	test error ('n/y')	network	depth	params	test error ('n')
CIFAR-10	ResNet	32	0.5M	7.22 / **6.32**	ResNet	110	1.7M	6.00
	DenseNet	40	1.0M	5.47 / **5.20**	DenseNet	100	7.0M	4.45
	WRN	16-8	11M	4.27 / **3.92**	WRN	28-10	36.5M	4.00
CIFAR-100	ResNet	32	0.5M	31.01 / **28.22**	ResNet	110	1.7M	26.79
	DenseNet	40	1.0M	25.20 / **23.68**	DenseNet	100	7.0M	21.58
	WRN	16-8	11M	20.43 / **19.33**	WRN	28-10	36.5M	19.25

Table 2: The self-distillation for compact networks on CIFAR datasets (error rate (%)). 'n/y' denotes without/with SD.

lar network architectures have both of compact and complex networks. Compared to the self-distilled compact networks, these complex networks yield small accuracy gains with the cost of tremendous computing resource and memory overhead caused by the increasing of network depth or width. Fortunately, it is shown that our SD mechanism can obviously boost the accuracy of compact networks to a similar level of corresponding complex networks.

Overall performances of various networks

Our SD can be viewed as a generic supervised learning mechanism for improving the generalization performance of any single network without the need of teacher models or assistive networks. Thus, we also compare a variety of network

network	depth	params	CIFAR-10		CIFAR-100	
			bas.	SD	bas.	SD
ResNet	32	0.5M	7.22	**6.32**	31.01	**28.22**
	110	1.7M	6.00	**5.57**	26.79	**25.04**
ResNet (pre-act.)	18	11.2M	5.15	**4.83**	23.45	**21.47**
	34	21.3M	4.90	**4.42**	22.68	**20.75**
WRN	40-4	8.9M	4.53	**4.30**	21.18	**19.93**
	16-8	11.0M	4.27	**3.92**	20.43	**19.33**
	28-10	36.5M	4.00	**3.78**	19.25	**18.54**
DenseNet	40	1.0M	5.47	**5.20**	25.20	**23.68**
	100	7.0M	4.45	**4.34**	21.58	**20.50**

Table 3: The performance improvement of self-distillation compared to baseline (bas.) networks. Note all the experiments are implemented with caffe platform, which may be slightly different from published results of original papers.

architectures with and without SD. Table 3 shows its effectiveness for these networks from shallow/thin to deep/wide. From these results, we are able to confirm that our SD mechanism can reliably and steadily reduce the generalization error of single model compared to independently training only with hard labels. Compared to a small network, a large network requires several times of parameters and computations to obtain a slight accuracy gain on CIFAR-10. In contrast, our SD can easily attain a single model of high accuracy without the dependence of extra parameters and models. Furthermore, our SD also produces remarkable accuracy gains ($> 1\%$) on CIFAR-100 for various network architectures. It implies that the SD is more effective for the challenging case (each class only has limited samples).

Extension to large-scale classification task

Since SD has displayed its effectiveness for training various networks on these small datasets, we further evaluate its performance on the large-scale ImageNet-2012 classification task (Russakovsky et al. 2015). In the implementation, we resize all the raw images to single-scale 256×256, and then randomly crop a 227×227 or 224×224 region from a full image (256×256) or its mirror flip during the training stage. Except for random mirror and cropping, we do not use any other data-distortion techniques. Meanwhile, test image still keeps the single view of central region from a full image. Here, the standard AlexNet (Krizhevsky, Sutskever, and Hinton 2012) and ResNet-18 (He et al. 2016a) with shortcut type B are addressed, where are from caffe platform. And the other detailed configurations are the same as the original papers. The experimental results are given in Table 4. For the AlexNet architecture, the final validation error

network	depth	params	baseline	SD
AlexNet	8	61M	42.80 / 19.98	**42.04 / 19.05**
ResNet	18	11.7M	32.27 / 12.13	**31.34 / 11.30**

Table 4: The validation errors (single crop) on ImageNet.

rates can be reduced from $42.8\%/19.98\%$ (top-1/top-5) to **42.04**$\%$/**19.05**$\%$. Furthermore, we can also reduce validation error rates of ResNet-18 by **0.9**$\%$ top-1 and **0.8**$\%$ top-5. It implies that SD is also effective on large-scale classification task. Although AlexNet architecture uses the strong regularization of dropout, our SD still can effectively improve the final performance. It further demonstrates that the proposed mechanism dose indeed drive single network to learn some inherent properties for generalization beyond the common regularization terms. In addition, we also do not find the related literatures to report such large improvements of other model distillation mechanisms (Hinton, Vinyals, and Dean 2015; Ba and Caruana 2014; Romero et al. 2015; Urban et al. 2017) on large-scale dataset. Thus, compared to the model distillation methods, our SD has better extensibility and generality for any network architecture and dataset.

Hyper-parameters

For selections of hyper-parameters in Eq. 5, we firstly fix the $\lambda = 1$ as referenced in (Zhang et al. 2018b). For a robust model, it should predict the correct class label while producing approximated inter-class probability distributions

for these different distorted instances of same training input, which is the same as the human vision. Secondly, the temperature parameter T is aimed to soften final predicted probabilities of softmax, and then softer probabilities are incorporated into KL term in Eq. 3 as supervised information to learn feature representations toward the inter-class relations. According to our experiences and suggestions in (Hinton, Vinyals, and Dean 2015; Romero et al. 2015), we find that the setting of fixed values can keep stable and reliable results for all the networks ($T = 2$ for CIFAR-10, $T = 3$ for CIFAR-100 and ImageNet), and the larger T may incur more noises and the worse final results. Finally, the hyper-parameter μ is to emphasize consistent global features. However, we can not control the exactly the same on all the dimensions of global features due to the real presence of different distortions, and thus it is only a regularization term as same as the regular weight decay. We also

μ	0	0.0001	0.0005	0.001	0.005	0.01
Err.	29.06	28.79	28.22	28.81	28.88	28.54

Table 5: Error rate (%) of ResNet-32 for hyper-parameter μ on CIFAR-100. $\mu = 0$ denotes without the MMD.

conduct experiments for different μ in Table 5, and we can see that the final results are insensitive to the values of μ in a wide range. However, overly high μ (e.g., $\mu = 10$) also leads to the final accuracy degeneration. By default we set $\mu = 0.0001 \sim 0.0005$.

Ablation studies

The SD mainly contains two components: MMD metric for feature distribution and KL divergence for posterior distribution. To analyze the respective effectiveness, we conduct ablation studies with results shown in Table 6. The baseline training denotes the regular manner with single mini-batch samples augmented by data-distortion operation and hard labels. In the two-branch case, two-branch samples are generated by the same distorted operation (i.e., random mirror and cropping) on current mini-batch input, and hard labels are shared for them. To evaluate the effects of MMD and KL constraints, we add them individually or jointly to the training optimization objective of the standard baseline network. From Table 6, we can see that two-branch data

network	bas.	two-branch	MMD	KL	SD
ResNet-32	31.01	31.32	30.62	29.06	28.22
WRN-16-8	20.43	20.25	19.98	19.48	19.33

Table 6: The ablation studies of different components in self-distillation on CIFAR-100 (error rate (%)).

streams trained only with hard labels are not able to obviously improve the accuracy of model. When we incorporate the MMD term or KL term, the accuracy will be effectively promoted. It implies that each component of SD is independently effective. Furthermore, the better result can be achieved by integrating the two terms into baseline training, it further displays that the two components are also jointly effective.

Discussions

Although the experimental results demonstrate that self-distillation mechanism can effectively reduce the network generalization error, the theoretical interpretation needs further study in the future. The notation of single direction reliance (Morcos et al. 2018) can be a possible way to quantify the generalization ability of the model trained by the proposed mechanism or regular optimized mechanism. In addition, there maybe several possibilities for further exploitation of the potential capability of single model. Firstly, we can consider more types of data-distortion techniques (i.e., affine, thin plate spline, and perspective transformations (Jaderberg et al. 2015)) to harvest the greater benefits from the self-distillation mechanism. Secondly, deep neural networks trained with hard labels (Szegedy et al. 2014) easily cause the unexpected changing of their predictions when tested on samples beyond the training distribution (known as adversarial samples). Therefore, we can try the adversarial idea (Goodfellow et al. 2014) to produce adversarial samples, then use the self-distillation optimization between adversarial and raw examples. Finally, unlike the category probability distribution, the global feature distribution can not exactly the same on all the dimensions due to the real presence of data-distortion operation. Thus, we can design some more robust metrics instead of MMD, or extract important and key features to match the consistency across different distorted versions of the same training data.

Conclusion

We propose a simple yet effective self-distillation mechanism to optimize single network to a more desired solution by introducing the learning of consistent feature/posterior distributions between different distorted versions of same training input. The proposed mechanism can be used to train a compact network of high accuracy without the dependence of powerful pre-trained teacher model. Compared to the previous knowledge distillation (i.e., teacher-to-student and student-to-student), the proposed method efficiently utilize the data-to-data knowledge distillation to thoroughly discard the dependence of assistive models, which effectively reduce the training cost of obtaining a small network of high accuracy. Extensive experiments show that this method obviously boosts the accuracy and generalization ability of single model whether it is compact or complex network. Future works include the theoretical interpretation of self-distillation, combination with other learning strategies for further improving generalization, and applications to more scenarios such as object detection and image segmentation.

Acknowledgments

This work has been supported by the National Natural Science Foundation of China (NSFC) grants 61721004 and 61633021. We thank Xu-Yao Zhang and Zhiqiang Chen for helpful discussions.

References

Anil, R.; Pereyra, G.; Passos, A.; Ormandi, R.; Dahl, G. E.; and Hinton, G. E. 2018. Large scale distributed neural network training through online distillation. In *ICLR*.

Ba, J., and Caruana, R. 2014. Do Deep Nets Really Need to be Deep? In *NIPS*, 2654–2662.

Bucila, C.; Caruana, R.; and Niculescu-Mizil, A. 2006. Model compression. In *KDD*, 535–541.

Chen, W.; Wilson, J. T.; Tyree, S.; Weinberger, K. Q.; and Chen, Y. 2015. Compressing Neural Networks with the Hashing Trick. In *ICML*, 2285–2294.

Chen, G.; Choi, W.; Yu, X.; Han, T. X.; and Chandraker, M. 2017. Learning efficient object detection models with knowledge distillation. In *NIPS*, 742–751.

Denil, M.; Shakibi, B.; Dinh, L.; Ranzato, M.; and de Freitas, N. 2013. Predicting Parameters in Deep Learning. In *NIPS*, 2148–2156.

Denton, E. L.; Zaremba, W.; Bruna, J.; LeCun, Y.; and Fergus, R. 2014. Exploiting Linear Structure Within Convolutional Networks for Efficient Evaluation. In *NIPS*, 1269–1277.

Furlanello, T.; Lipton, Z. C.; Tschannen, M.; Itti, L.; and Anandkumar, A. 2018. Born-Again Neural Networks. In *ICML*, 1602–1611.

Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A. C.; and Bengio, Y. 2014. Generative adversarial nets. In *NIPS*, 2672–2680.

Hadsell, R.; Chopra, S.; and LeCun, Y. 2006. Dimensionality Reduction by Learning an Invariant Mapping. In *CVPR*, 1735–1742.

Han, S.; Mao, H.; and Dally, W. J. 2016. Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding. In *ICLR*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *ICCV*, 1026–1034.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016a. Deep Residual Learning for Image Recognition. In *CVPR*, 770–778.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016b. Identity Mappings in Deep Residual Networks. In *ECCV*, 630–645.

He, Y.; Zhang, X.; and Sun, J. 2017. Channel Pruning for Accelerating Very Deep Neural Networks. In *ICCV*, 1398–1406.

Hinton, G. E.; Vinyals, O.; and Dean, J. 2015. Distilling the Knowledge in a Neural Network. In *arXiv:1503.02531*.

Huang, G.; Liu, Z.; and Weinberger, K. Q. 2017. Densely Connected Convolutional Networks. In *CVPR*.

Jaderberg, M.; Simonyan, K.; Zisserman, A.; and Kavukcuoglu, K. 2015. Spatial Transformer Networks. In *NIPS*, 2017–2025.

Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R. B.; Guadarrama, S.; and Darrell, T. 2014. Caffe: Convolutional Architecture for Fast Feature Embedding. In *ACM Multimedia*, 675–678.

Krizhevsky, A., and Hinton, G. E. 2009. Learning multiple layers of features from tiny images. In *Technical report, University of Toronto*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*, 1106–1114.

Li, H.; Kadav, A.; Durdanovic, I.; Samet, H.; and Graf, H. P. 2017. Pruning Filters for Efficient ConvNets. In *ICLR*.

Liu, Z.; Li, J.; Shen, Z.; Huang, G.; Yan, S.; and Zhang, C. 2017. Learning Efficient Convolutional Networks through Network Slimming. In *ICCV*, 2755–2763.

Long, M.; Cao, Y.; Wang, J.; and Jordan, M. I. 2015. Learning transferable features with deep adaptation networks. In *ICML*, 97–105.

Morcos, A. S.; Barrett, D. G. T.; Rabinowitz, N. C.; and Botvinick, M. 2018. On the importance of single directions for generalization. In *ICLR*.

Quadrianto, N.; Petterson, J.; and Smola, A. J. 2009. Distribution matching for transduction. In *NIPS*, 1500–1508.

Radosavovic, I.; Dollár, P.; Girshick, R. B.; Gkioxari, G.; and He, K. 2018. Data distillation: Towards omni-supervised learning. In *CVPR*.

Romero, A.; Ballas, N.; Kahou, S. E.; Chassang, A.; Gatta, C.; and Bengio, Y. 2015. FitNets: Hints for Thin Deep Nets. In *ICLR*.

Rosenberg, C.; Hebert, M.; and Schneiderman, H. 2005. Semi-supervised self-training of object detection models. In *IEEE Workshop on Applications of Computer Vision*, 29–36.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M. S.; Berg, A. C.; and Fei-Fei, L. 2015. ImageNet Large Scale Visual Recognition Challenge. *IJCV* 115(3):211–252.

Schroff, F.; Kalenichenko, D.; and Philbin, J. 2015. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 815–823.

Simard, P. Y.; LeCun, Y.; Denker, J. S.; and Victorri, B. 1996. Transformation Invariance in Pattern Recognition-Tangent Distance and Tangent Propagation. In *Neural Networks: Tricks of the Trade*. 239–27.

Sutskever, I.; Martens, J.; Dahl, G. E.; and Hinton, G. E. 2013. On the importance of initialization and momentum in deep learning. In *ICML*, 1139–1147.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I. J.; and Fergus, R. 2014. Intriguing properties of neural networks. In *ICLR*.

Urban, G.; Geras, K. J.; Kahou, S. E.; Aslan, Ö.; Wang, S.; Caruana, R.; Mohamed, A.; Philipose, M.; and Richardson, M. 2017. Do Deep Convolutional Nets Really Need to be Deep and Convolutional? In *ICLR*.

Wen, Y.; Zhang, K.; Li, Z.; and Qiao, Y. 2016. A Discriminative Feature Learning Approach for Deep Face Recognition. In *ECCV*, 499–515.

Zagoruyko, S., and Komodakis, N. 2016. Wide Residual Networks. In *BMVC*.

Zagoruyko, S., and Komodakis, N. 2017. Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer. In *ICLR*.

Zhang, H.; Cisse, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2018a. mixup: Beyond Empirical Risk Minimization. In *ICLR*.

Zhang, Y.; Xiang, T.; Hospedales, T. M.; and Lu, H. 2018b. Deep Mutual Learning. In *CVPR*.

Zhong, Z.; Zheng, L.; Kang, G.; Li, S.; and Yang, Y. 2017. Random erasing data augmentation. In *arXiv:1708.04896*.