ONE GENERATION KNOWLEDGE DISTILLATION BY UTILIZING PEER SAMPLES

Anonymous authors
Paper under double-blind review

ABSTRACT

Knowledge Distillation is a widely used technique in recent deep learning research to obtain small and simple models whose performance is on a par with their large and complex counterparts. Standard Knowledge Distillation tends to be time-consuming because of the training time spent to obtain a teacher model that would then provide guidance for the student model. It might be possible to cut short the time by training a teacher model on the fly, but it is not trivial to have such a high-capacity teacher that gives quality guidance to student models this way. To improve this, we present a novel framework of Knowledge Distillation exploiting dark knowledge from the whole training set. In this framework, we propose a simple and effective implementation named Distillation by Utilizing Peer Samples (DUPS) in one generation. We verify our algorithm on numerous experiments. Compared with standard training on modern architectures, DUPS achieves an average improvement of 1%-2% on various tasks with nearly zero extra cost. Considering some typical Knowledge Distillation methods which are much more time-consuming, we also get comparable or even better performance using DUPS.

1 INTRODUCTION

Recent years have witnessed continuous development of deep neural network models. A general trend is that improvements in model performance are usually coupled with more complex architecture designs and higher cost of computation. In order to obtain more compact models with higher quality, the idea of Knowledge Distillation (KD) first emerged in the form of knowledge transfer between models (Buciluǎ et al., 2006). KD takes advantage of the "dark knowledge" by transferring it from teacher models to student models so as to facilitate the latter's training process (Hinton et al., 2015). Student models, with the availability of softened output vectors from teacher models in KD, have access to richer information in comparison to directly learning from hard labels provided by training set. KD significantly improves smaller models' performance, and thus it further allows model compression.

Although great progress has been made in this area, much more training cost is incurred due to involved time-consuming mid-output (e.g. feature maps) alignment when training student models, on top of extra training of a huge teacher model. It is ad meaningful objective of finding more efficient KD methods. Recent works by (Furlanello et al., 2018) and (Lan et al., 2018b) show that a stronger teacher model is not the necessary condition for improving the student model. Their research shows that it is possible that the student model's performance can be significantly improved by an identically structured teacher model. Although the techniques remain inefficient due to the cost of multi-generation (at least one extra) training of teacher models, these works give important hints that cheaper teachers with considerable effectiveness may exist. Recently (Yang et al., 2018) extend these works, trying to obtain continuously improved teachers by introducing the cyclic learning rate technique in one-generation training. They propose Snapshot Distillation (SD), which uses models obtained from earlier checkpoints as teachers and skips the process of separately training a teacher model.

Inspired by recent interesting ideas of dataset distillation (Wang et al., 2019) for objectives on other research areas, we propose a novel approach for KD in this paper. Instead of relying on the assitance of a separate teacher model or checkpoint, we exploit hidden knowledge in the dataset to

generate a surrogate teacher. Specifically, we first define a more general framework of knowledge distillation utilizing the whole dataset to generate extra supervision signals, rather than using a single sample alone. Then we propose a very simple yet effective implementation of one-generation KD, called Distillation by Utilizing Peer Samples (DUPS). In DUPS, each sample borrows continuously boosted secondary information from a random subset of peer samples belonging to the same category on the fly.

We perform extensive experiments on CIFAR100 dataset, with various modern architectures such as PreActResNet, WideResNet, and ResNeXt, demonstrating that our proposed DUPS gains significant improvement compared to standard SGD training with nearly zero extra computation cost. DUPS also outperforms recent one-generation KD method SnapShot Distillation (Yang et al., 2018) on most architectures. Moreover, we validate our algorithm on more practical tasks, include ImageNet classification, transfer learning, and language model. Experiments show that DUPS is generally effective across different tasks.

In summary, our main contributions include: 1) To the best of our knowledge, we are the first to propose an extension framework of Knowledge Distillation utilizing the whole training set other than a single sample. 2) Under this framework we implement a general on-the-fly algorithm DUPS which achieves significant improvement at almost no extra cost.

The rest of the paper is organized as follows. Section 2 presents prior works related to this paper. Section 3 introduces our methodology of the general knowledge distillation framework. Section 4 demonstrates our experimental results and provides some discussions. Section 5 concludes the paper.

2 Related Work

2.1 Knowledge Distillation

Methodologies and Applications In a traditional KD approach, a separate and typically time-consuming phase is unavoidable to first obtain a complex teacher model. The softened outputs of the teacher model are then utilized to train a student model that has a much simpler network structure (Hinton et al., 2015). Other works further develop this idea, such as aligning feature maps, grammian matrix or activation boundaries (Romero et al., 2014; Zagoruyko & Komodakis, 2016; Yim et al., 2017; Heo et al., 2018). KD is typically used to get compact models with high performance or for purely purpose of model quality improvement (Furlanello et al., 2018; Lan et al., 2018b; Yang et al., 2018). In addition to these general works, KD is widely demonstrated to be effective on various modern practical tasks, including transfer learning (Li et al., 2019), object detection (Chen et al., 2017), visual relationship detection (Yu et al., 2017) and so on. Despite its effectiveness, one common limitation is the improved performance of student networks in traditional KD is achieved at the cost of extra training time to obtain the teacher model in the first place.

Efficiency of Knowledge Distillation Recently researchers have begun paying more attention to the inefficiency problem of KD. In the scenario of large scale training, (Anil et al., 2018) proposed online distillation which trains multiple copies of a model in a distributed system, at the expense of involving more activated devices simultaneously. It becomes challenging to distill knowledge in one generation because the model currently being trained lacks the capacity to be a good teacher model for itself. SnapShot Distillation ameliorates this problem by utilizing cyclic learning rate (Yang et al., 2018). They divide the whole training process into a few mini-generations, using cosine annealing learning rate policy (Loshchilov & Hutter, 2016) in each mini-generation so as to ensure the teacher models' quality. Other works tackle the problem by implementing architecture-specific knowledge transfer techniques, e.g. adding new branches or connections to the original model structure (Lan et al., 2018a; Zhang et al., 2019; Hahn & Choi, 2019).

In this paper, we mainly focus on architecture-agnostic KD that aims to use inter-class information to improve general supervision.

2.2 LABEL PROPAGATION

Although rarely mentioned, KD is closely related to the area of label propagation. Some early works exploit language knowledge related to image classification objects to help improve the performance of the latter. For example, (Deng et al., 2010; Verma et al., 2012; Akata et al., 2015) draw on external knowledge such as well-constructed language database WordNet, annotated attributes of each instance, or other dataset with hierarchical semantic labels. Recently (Bagherinezhad et al., 2018) utilize the idea of KD to obtain a refined label of a specific crop by a pre-trained teacher model, offering more accurate targets for cropped input images. Another popular idea is to construct propagated labels for effective regularization. (Szegedy et al., 2016) introduce Label Smoothing (LS) which softens the labels in the dataset: instead of being a one-hot encoding vector, the label after LS becomes smoother in distribution over all entries in the vector. It's a very useful technique in practice till today. Some recent works further analyze the effectiveness of LS or other label-disturb methods (Pereyra et al., 2017; Muller et al., 2019). In this paper We will show that, albeit being similar in appearance, the continuously boosted softened target generated by the whole dataset is more than a data-agnostic regularizer like Label Smoothing.

3 METHODOLOGY

3.1 TYPICAL FORM OF KD

Without loss of generality, we first define a typical formulation of KD based on past works (Hinton et al., 2015; Furlanello et al., 2018; Yang et al., 2018) in the classical classification setting. Let $\mathcal{D} \in \mathcal{X} \times \mathcal{Y}$ be the training set which contains n labelled training samples and C classes. Each sample is denoted by $(\mathbf{x_i}, y_i) \in \mathcal{D}$, where $y_i \in \{1, 2, \ldots, C\}$. We define our objective network $\boldsymbol{f_S}(\mathbf{x}, \boldsymbol{\theta}) : \mathcal{X} \mapsto \mathcal{Y}$ parameterized with $\boldsymbol{\theta}$ as the mapping function. Commonly in practice, we use the cross entropy function to metric the distance between ground-truth one-hot labels and outputs generated by $\boldsymbol{f_S}$. For traditional KD, we need to introduce another optimized solution $\boldsymbol{\theta'}$ of a particular function $\boldsymbol{f_T}$ into the final loss function. We omit normal regularization terms such as L^2 normalization of parameters for the sake of simplicity. Upon some input sample $\mathcal{D}_i = (\mathbf{x_i}, y_i)$, we get

$$\mathcal{L}(\mathcal{D}_i; \boldsymbol{\theta}) = \lambda_{CE} \cdot \mathcal{L}_{CE}[y_i, \boldsymbol{f}_S(\mathbf{x_i}; \boldsymbol{\theta})] + \lambda_{KD} \cdot \mathcal{L}_{KD}[\boldsymbol{f}_S(\mathbf{x_i}; \boldsymbol{\theta}), \boldsymbol{f}_T(\mathbf{x_i}; \boldsymbol{\theta'})] \tag{1}$$

, where λ_{CE} and λ_{KD} are hyperparameters to balance the relative contribution of the cross entropy term and knowledge distillation term. $\mathcal{L}_{KD}$ is typically computed by cross entropy (Furlanello et al., 2018; Hahn & Choi, 2019) or Kullback Leibler divergence (Lan et al., 2018a; Yang et al., 2018) between the logits produced by the teacher model and the student model. Note that $\boldsymbol{f_T}$ can be identical in structure as $\boldsymbol{f_S}$, as demonstrated by the successful implementations of Born-Again Network (Furlanello et al., 2018), Self-Referenced Network (Lan et al., 2018a) and Snapshot Distillation (Yang et al., 2018). For one generation distillation, since there isn't an external training process of a teacher model, the teacher signal $\boldsymbol{f}_T(\mathbf{x_i}; \boldsymbol{\theta'})$ for sample $\mathcal{D}_i$ is generated from the internal of training procedure or dataset. For example, Snapshot Distillation uses a sequence of checkpoints at the end of each training cycle $\boldsymbol{\theta'} \in \{\boldsymbol{\theta^{c_1}}, \boldsymbol{\theta^{c_2}}, \ldots\}$ as teacher solutions.

3.2 EXTENDED FORM OF KD

We will extend the formulation of KD, making it more compatible with commonly used neural network architectures and algorithms. In the above typical framework, KD loss term is with respect to some sample $\mathcal{D}_i = (\mathbf{x_i}, y_i)$. In contrast, our proposed KD framework has its KD term as a function of the whole dataset $\mathcal{D}$. Thus with the help of dataset $\mathcal{D}$ and some internal or external obtained supplementary teacher solution(s) $\boldsymbol{\theta'}$, we have

$$\mathcal{L}(\mathcal{D}_i; \boldsymbol{\theta}) = \lambda_{CE} \cdot \mathcal{L}_{CE}[y_i, \boldsymbol{f}_S(\mathbf{x_i}; \boldsymbol{\theta})] + \lambda_{KD} \cdot \mathcal{L}_{KD}[\boldsymbol{f}_S(\mathbf{x_i}; \boldsymbol{\theta}), \boldsymbol{f}_T(\mathcal{D}_i; \boldsymbol{\theta'}, \mathcal{D})] \tag{2}$$

. The notation for teacher signal changes from $\boldsymbol{f}_T(\mathbf{x_i}; \boldsymbol{\theta'})$ to $\boldsymbol{f}_T(\mathcal{D}_i; \boldsymbol{\theta'}, \mathcal{D})$ because the teacher model requires the label y_i and at least part of the dataset to effectively utilize dataset information in our KD. We note that implementations of traditional KD by (Hinton et al., 2015), Born-Again Network (Furlanello et al., 2018) and Snapshot Distillation (Yang et al., 2018) can be seen as special cases of the above general formulation whose distilled information only depends on the input sample while ignoring others.

3.3 TEACHER SIGNAL UTILIZING PEER SAMPLES

The most straightforward idea of leveraging other samples to help is to utilize similar samples. Given a training sample $\mathcal{D}_i = (\mathbf{x_i}, y_i)$, we random select n peer samples in the same category of training set, noted by

$$Y_{y_i} = \{(\mathbf{x_{k_1}}, y_i), (\mathbf{x_{k_2}}, y_i), \ldots, (\mathbf{x_{k_n}}, y_i)\} \subset \mathcal{D}$$

. With the help of these peer samples, we get teaching signal with respect to D_i as below

$$\boldsymbol{f}_T(\mathcal{D}_i; \boldsymbol{\theta'}, \mathcal{D}) = \boldsymbol{f}_T(\mathcal{D}_i; \boldsymbol{\theta'}, Y_{y_i}) \tag{3}$$

. As shown above, we do not focus on how one single instance could provide the student model with secondary information that is not directly available from the dataset labels (Hinton et al., 2015; Yang et al., 2019); rather, we aim to find a general trend that reveals the extent of some statistical characteristics, e.g. inter-class similarity, by utilizing peer samples. There are two benefits. First, since such information is relatively static or stable for a period of training procedure and can be shared among samples of the same category, it is cheap to obtain and store in memory. Second, teaching signals voted by a cluster of samples may offer more reliable secondary information comparing with that produced by a sub-optimized checkpoint found before training completes.

3.4 DUPS IMPLEMENTATION

Similar to SnapShot Distillation proposed by (Yang et al., 2018), our method achieves on-the-fly distillation within one generation of training by dividing the training process into m stages, with each stage consisting of t epochs. However, an important difference is that our method do not strongly depend on cyclic learning rate for obtaining reliable teaching signal. Let M be the set of such stages and T be the set of epochs. Then $M = \{M_1, M_2, \ldots, M_m\}$, and for each i such that $1 \leq i \leq m$, $M_i = \{T_{(i-1)\times t+1}, T_{(i-1)\times t+2}, \ldots, T_{(i-1)\times t+(t-1)}, T_{i\times t}\}$. We follow the algorithm in Alg. 1 to implement DUPS. For a sample $\mathcal{D}_i = (\mathbf{x_i}, y_i)$, its peer samples, denoted by Y_{y_i}, are a random subset of the all samples found in the same category. Hence,

$$Y_{y_i} \subset \{(\mathbf{x}, y) \in \mathcal{D} \mid y = y_i\}$$

. For specific implementation, we share teacher signal among samples of the same category. Thus, Eq. 3 is further simplified to

$$\forall \mathcal{D}_j \in \{(\mathbf{x}, y) \in \mathcal{D} \mid y = y_i\}, \boldsymbol{f}_T(\mathcal{D}_j; \boldsymbol{\theta'}, \mathcal{D}) = \boldsymbol{f}_T(Y_{y_i}; \boldsymbol{\theta'}) \tag{4}$$

, which means the teacher signal $\boldsymbol{f}_T$ for sample $D_j = (\mathbf{x_j}, y_j)$ only depends on the sample's label y_j but not its data input $\mathbf{x_j}$. $\mathcal{D}$ is omitted in $\boldsymbol{f}_T(Y_{y_i}; \boldsymbol{\theta'})$ as the last parameter because only $Y_{y_j} \subset \mathcal{D}$ is utilized.

Formally, let the teacher signal for each category with label y_i at stage M_n be $\boldsymbol{f}_T^n(Y_{y_i}; \boldsymbol{\theta'})$, where n is the index of the current training stage. We denote the checkpoint of student model $\boldsymbol{f}_S$ at the end of n^{th} stage be $\boldsymbol{\theta}^n$. We define $\boldsymbol{f}_T^1(Y_{y_i}; \boldsymbol{\theta'})$ to be zero. The rest of teacher signals are computed based on

$$\boldsymbol{f}_T^n(Y_{y_i}; \boldsymbol{\theta'}) = \frac{1}{\|Y_{y_i}\|} \sum_{(\mathbf{x}, y) \in Y_{y_i}} \boldsymbol{f}_S(\mathbf{x}; \boldsymbol{\theta}^{n-1}) \tag{5}$$

. The average softened logit of each category obtained at the last epoch of each stage is used as the "teacher model" for training of student model at next stage. In total, there are $m - 1$ iterations

of knowledge distillation between "teacher models" and "student models" in the whole training process.

Algorithm 1: DUPS implementation

Input: Current epoch $T_c \in M_n$, the set of training samples $\mathcal{D}$
Output: Improved neural network model parameters $\boldsymbol{\theta}$

for *each batch* $\mathcal{B}$ *in* $\mathcal{D}$ **do**
  for *each sample* $(\mathbf{x_j}, y_j)$ *in* $\mathcal{B}$ **do**
    Compute the logits l and prediction $\boldsymbol{f}_S(\mathbf{x_j})$ produced by the current model;
    Compute cross entropy loss $\mathcal{L}_{CE}$ using $\boldsymbol{f}_S(\mathbf{x_j})$ and y_j
    if M_n *is the first stage* M_1 **then**
      Set knowledge distillation loss $\mathcal{L}_{KD}$ to be $\boldsymbol{f}_T^1(Y_{y_j}; \boldsymbol{\theta'}) = 0$;
    else
      Compute knowledge distillation loss $\mathcal{L}_{KD}$ using l and $\boldsymbol{f}_T^n(Y_{y_j}; \boldsymbol{\theta'})$ where $Y_{y_j} = \{(\mathbf{x}, y) \in \mathcal{D} \mid y = y_j\}$;
    Obtain the total loss (Eq 2);
  Calculate the average loss of this batch;
  Update model parameters;
if T_c *is the last epoch in the current stage* M_n **then**
  Update $\boldsymbol{f}_T^{n-1}(Y_{y_p}; \boldsymbol{\theta'})$ to $\boldsymbol{f}_T^n(Y_{y_p}; \boldsymbol{\theta'})$ using the corresponding set of peer samples Y_{y_p} for all $y_p \in \{1, 2, \ldots, C\}$ (Eq 5);

4 EXPERIMENTS

4.1 TASKS AND DATASETS

Image Classification We use CIFAR100 (Krizhevsky, 2009) and ImageNet (Deng et al., 2009) to test the performance of DUPS on image classification task. CIFAR 100 contains RGB images categorized into 100 classes, with each class composing 600 images. There are 500 training images and 100 testing images in each class. ImageNet is a tree-structured image database created according to the WordNet hierarchy. It consists of more than 20K categories and a total of 14 million images. We use the popular subset ILSVRC2012 which containing 1.3M images covering 1K categories. We do not conduct experiments on CIFAR10 because it has been empirically observed that, due to the lack of fine-level categorization in CIFAR10, neural networks do not significantly benefit from distillations between teacher and student models (Yang et al., 2018; Furlanello et al., 2018).

Transfer Learning We use ImageNet as the source dataset and have 4 different datasets as the target dataset in the transfer learning task, covering typical types of plants, animals, objects and texture . The 4 target datasets are (1) Flower102 (Nilsback & Zisserman, 2008) which contains 102 categories of 8189 flower images, (2) Caltech-UCSD Birds-200-2011 (Wah et al., 2011), which has 11,788 images classified into 200 categories, (3) FGVC-Aircraft (Maji et al., 2013) which composes 10,000 images of aircraft across 100 aircraft models, and (4) Describable Textures Dataset (DTD) (Cimpoi et al., 2015) which is a texture database, consisting of 5640 images, organized according to a list of 47 terms (categories).

Natural Language Processing We use Penn Tree Bank (PTB) dataset (Marcus et al., 1993) to evaluate the performance of DUPS on language modeling.

4.2 EXPERIMENT SETTINGS

Image Classification We train all models except DenseNet for image classification in 160 epochs. For DenseNet we train 240 epochs because it converges slower. The initial learning rate is 0.1 for all architectures. Training batch size is 64. We use standard SGD optimizer with momentum 0.9 and weight decay 0.0001. We apply standard data augmentation the same way as the Pytorch official examples on both CIFAR100 and ImageNet classification task. For CIFAR100, we pad the input

Table 1: **Test accuracy on CIFAR100.** SGD refers to models trained with standard Stochastic Gradient Descent optimizer. LS are models improved by using Label Smoothing technique. SD are the models trained with SnapShot Distillation method. Finally, DUPS and DUPS+ are our methods proposed in this paper. The difference between DUPS and DUPS+ is that DUPS+ uses cyclic learning rate while DUPS does not.

Model	SGD	LS	DUPS	SD	DUPS+
PreActResNet18	0.7646	**0.7795**	0.7785	0.7729	**0.7794**
PreActResNet34	0.7701	0.7729	**0.7823**	0.7665	**0.7785**
PreActResNet50	0.7687	0.7798	**0.7847**	0.7809	**0.7877**
PreActResNet101	0.7756	0.7829	**0.7882**	0.7788	**0.7899**
DenseNet40(240)	0.7003	0.7034	**0.7101**	0.7115	**0.7174**
DenseNet100(240)	0.7515	**0.7559**	0.7557	**0.7657**	0.7617
WideResNet28x10	0.7968	0.7996	**0.8028**	0.8013	**0.8062**
ResNeXt29_8x64d	0.8017	0.8103	**0.8184**	-	-

images by 4 pixels, and then randomly crop a sub-region of 32×32 and randomly do a horizontal flip. For ImageNet, we first randomly crop a sub-region of 224×224 and randomly do a horizontal flip. We normalize the input data as done in common practice.

Transfer Learning We use ResNet-101 as the base model to apply DUPS. We train the model with 40 epochs and the batch size for training is 64. SGD optimizer is used with a momentum of 0.9. The initial learning rate is set to 0.01 and the weight decay is set to 0.0001. We use exactly the same data augmentation methods as in ImageNet classification task.

Natural Language Processing We validate the performance of DUPS on three regularized LSTM models with varying depth. For the large model, we use 1500 hidden units and train 55 epochs. The learning rate decays by 1 / 1.5 after 15 epochs. Its dropout rate is set to be 0.65 and dropout is applied to all non-recurrent connections. The medium model consists of 650 hidden units. We train it with 40 epochs. The learning rate decays by 0.8 after 5 epochs. Its dropout rate is 0.5. The small model has 200 hidden units. It is trained in 15 epochs. The learning rate remains as the initial value for 4 epochs and then starts to decay at the rate of 0.5. No dropout is used in the small model. For all three models, the initial learning rate is 20.

While training DUPS to compare with models trained with SGD optimizer or label smoothing technique in the three tasks, we use cosine annealing policy to update learning rates for all these models. In order to make fair comparisons with Snapshot Distillation, we upgrade our DUPS with cyclic learning rate policy, which is named DUPS+. Within each cycle , DUPS+ uses cosine annealing learning rate.

For hyper-parameters specific to DUPS, we also use a common setting for all tasks following instructions from some empirical studies. We set λ_{CE} to 0.8 and λ_{KD} to 0.2. The Kullback Leibler divergence is used to calculate knowledge distillation term. We use temperature of 6 to soften the logits. We divide the training process into 10 stages for image classification tasks and 5 stages for transfer learning and language model tasks.

4.3 Experiment Results

We present our experiment results and give insights into the advantages as well as disadvantages using DUPS. We achieve this by comparing DUPS with models trained with SGD, and sometimes with closely related algorithms, namely Label Smoothing and Snapshot Distillation. Although SnapShot Ensembles (Huang et al., 2017) also offers ideas on which we further develop, we do not carry out duplicate experiments to compare with it because (Yang et al., 2018) have already made comparisons between Snapshot Distillation and SnapShot Ensembles. Therefore, we directly compare our proposed DUPS with SnapShot Distillation.

Image Classification Table. 1 shows that our DUPS implementation consistently and significantly improve baseline models in accuracy for the vast majority of neural network architectures we tested.

Table 2: **Training Time of Different Algorithms on PreActResNet50 and WideResNet28x10.** All abbreviations follow the same rule as in Table. 1. We run experiments on Tesla V100 GPUs and normalize the training time of SGD to 1 for simplified comparison.

Model	SGD	LS	SD	DUPS
PreActResNet50	1	$\sim$1	1.32	1.04
WideResNet28x10	1	$\sim$1	1.27	1.01

Table 3: **Top-1 Test accuracy on ImageNet.** All abbreviations follow the same rule as in Table. 1.

Model	SGD	LS	DUPS
ResNet50	76.58	76.69	**77.58**
ResNet101	77.86	78.41	**78.74**
ResNet152	78.17	78.63	**79.06**

The improvement is generally within 1%-2% in comparison to models trained with the standard SGD optimizer. We notice that more complex architectures do not always perform better than simpler ones, while DUPS achieves stable improvement. We can observe a similar trend in Table. 3 when we apply DUPS to different models on ImageNet.

For the majority of the models we tested, DUPS also outperforms Label Smoothing (LS) in image classification. Although DUPS does not beat LS on PreActResNet18 and DenseNet100, the difference is relatively marginal. When compared with a recently proposed KD method SnapShot Distillation, DUPS shows comparable performance even though the SD uses a more complicated learning rate policy. We also demonstrate that DUPS+ which combines cyclic learning rate outperforms SD in most cases. However we noticed that leveraging cyclic learning rate doesn't always bring benefit. In addition, DUPS obtains the improved results with significantly less training time than that SD needs as showed in Table. 2.

Transfer Learning We fixed our test model to be ResNet101 and perform experiments on the chosen datasets. Results in Table. 4 indicate that, compared with models trained using SGD, DUPS improves the transfer learning outcomes on all four datasets, and the improvements range from 0.46% to 1.78%. These results testify that DUPS can enhance model performance on varying datasets.

In contrast, LS does not always yield positively improved results. Furthermore, the extent of improvement LS brings is considerably less than that of DUPS, as shown by the statistics.

Natural Language Processing Last but not least, we investigate the effectiveness of DUPS on language modeling. As shown in Table 5, the use of DUPS reduces the perplexity value by a proportion of 1.5%$\sim$4.1% on all three models. Note that the model displays better performance when the perplexity value is low in language modeling. Since DUPS significantly lowers the perplexity values to both small and large size of the LSTM models, its effectiveness is not hindered by scaling the size of the model in this task. This flexibility is another advantage of DUPS.

4.4 Discussions

Here we give a short discussion about how and why DUPS brings benefits. We first demonstrate some empirical characteristics of DUPS observed in our experiments. We plot the learning curve of the whole training procedure of PreActResNet18 as Fig. 1. For better demonstration purpose, we divide the training process into only 4 stages for DUPS training, with each stage consisting of 40 epochs. We observe that SGD and DUPS display almost the same standard of performance in the first stage as expected. While at the 41th epoch, both training and test accuracy of DUPS get a sharp rise due to involving teacher signal generated in the 40th epoch. Then both training and test accuracy drop slightly for a few epochs, and then return to the trend of slowly rising for the remaining epochs until next stage. A similar phenomenon also appears at the beginning of next stage, although the magnitude of accuracy improvement becomes much more smaller. As the model reaches the beginning of final stage, we no longer see increases in accuracy since training is nearly

Table 4: **Test accuracy using ResNet101 on various datasets.** All abbreviations follow the same rule as in Table. 1 or explained in Section 4.1.

Dataset	SGD	LS	DUPS
Flower102	0.9179	0.9279	**0.9294**
FGVC_Aircraft	0.7741	0.7675	**0.7787**
DTD	0.6646	0.6705	**0.6824**
CUB_200_2011	0.8172	0.8152	**0.8246**

Table 5: **Perplexity value of the model using LSTM on Penn Treebank.** Lower is better. All abbreviations follow the same rule as in Table. 1 or explained in Section 4.1.

Model Size	SGD	LS	DUPS
Small	129.7	129.1	**125.4**
Medium	95.3	96.7	**93.9**
Large	89.2	88.7	**85.7**

saturated. We notice that since the first sharp rising, DUPS continuously outperforms SGD by a stable gap for the following training epochs until convergence.

We also investigate the influence of different choices of hyper-parameters specific to DUPS. The most important two are the number of stages and number of random peer samples. We run a grid search method to validate different combinations of these two variables. We use update intervals, or equivalently number of epochs per stage, instead of number of stages for clarity in this experiments. In Fig. 2 we see that performance of DUPS does not seem to be very sensitive to most combinations of the hyperparameters. When the number of peer samples increases to 5 or more, model accuracy tends to be over 77.3%. Even when the number of peer samples is low, a good choice of the value of update interval can boost the model performance significantly. For example, when number of peer samples is 1 and update interval is set to be between 20 to 40, DUPS still delivers satisfying results which is comparable to its best performance. Low accuracy of the model only happens consistently when the value of update interval is large. If update interval is set to 80, the model, teacher-student knowledge transfer only takes place once during the whole training process. Consequently, the opportunity to distill the knowledge obtained from the dataset is too rare for the model to benefit from DUPS.

5 Conclusion

In this paper, we have introduced a general framework for one-generation KD: We incorporate the information contained within the dataset into teacher-student optimization. We have also proposed an effective implementation of this general framework named DUPS. With extensive experiments, this simple yet effective algorithm is verified to be effective in improving model performance in tasks like image classification, transfer learning and language modeling with almost no additional cost in training resources. The demonstrated success of DUPS imply that utilizing dataset information during training potentially allow us to gain even more benefits.

References

Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. Label-embedding for image classification. *IEEE transactions on pattern analysis and machine intelligence*, 38(7): 1425–1438, 2015.

Rohan Anil, Gabriel Pereyra, Alexandre Passos, Róbert Ormándi, George E. Dahl, and Geoffrey E. Hinton. Large scale distributed neural network training through online distillation. *ArXiv*, abs/1804.03235, 2018.

Hessam Bagherinezhad, Maxwell Horton, Mohammad Rastegari, and Ali Farhadi. Label refinery: Improving imagenet classification through label progression. *ArXiv*, abs/1805.02641, 2018.

Figure 1: SGD and DUPS Learning Curves on PreActResNet18

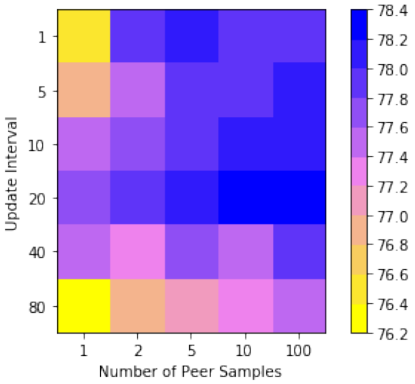

Figure 2: DUPS hyperparameter experiment on PreActResNet18

Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pp. 535–541, New York, NY, USA, 2006. ACM.

Guobin Chen, Wongun Choi, Xiang Yu, Tony X. Han, and Manmohan Krishna Chandraker. Learning efficient object detection models with knowledge distillation. In *NIPS*, 2017.

Mircea Cimpoi, Subhransu Maji, and Andrea Vedaldi. Deep filter banks for texture recognition and segmentation. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3828–3836, 2015.

Jia Deng, Alexander C Berg, Kai Li, and Li Fei-Fei. What does classifying more than 10,000 image categories tell us? In *European conference on computer vision*, pp. 71–84. Springer, 2010.

Jun Deng, Wei Dong, Richard Socher, Li-Jia Li, Kuntai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

Tommaso Furlanello, Zachary C. Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *ICML*, 2018.

Sangchul Hahn and Heeyoul Choi. Self-knowledge distillation in natural language processing. *arXiv preprint arXiv:1908.01851*, 2019.

Byeongho Heo, Minsik Lee, Sangdoo Yun, and Jin Young Choi. Knowledge transfer via distillation of activation boundaries formed by hidden neurons. *ArXiv*, abs/1811.03233, 2018.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531, 2015.

Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. Snapshot ensembles: Train 1, get m for free. *ArXiv*, abs/1704.00109, 2017.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

Xu Lan, Xiatian Zhu, and Shaogang Gong. Knowledge distillation by on-the-fly native ensemble. In *NeurIPS*, 2018a.

Xu Lan, Xiatian Zhu, and Shaogang Gong. Self-referenced deep learning. *ArXiv*, abs/1811.07598, 2018b.

Xingjian Li, Haoyi Xiong, Hanchao Wang, Yuxuan Rao, Liping Liu, and Jun Huan. Delta: Deep learning transfer using feature map with attention for convolutional networks. *ArXiv*, abs/1901.09229, 2019.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2016.

Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew B. Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *ArXiv*, abs/1306.5151, 2013.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19:313–330, 1993.

Rafael J. Muller, Simon Kornblith, and Geoffrey E. Hinton. When does label smoothing help? *ArXiv*, abs/1906.02629, 2019.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pp. 722–729, 2008.

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *CoRR*, abs/1412.6550, 2014.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Nakul Verma, Dhruv Mahajan, Sundararajan Sellamanickam, and Vinod Nair. Learning hierarchical similarity metrics. In *2012 IEEE conference on computer vision and pattern recognition*, pp. 2280–2287. IEEE, 2012.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A. Efros. Dataset distillation. *ArXiv*, abs/1811.10959, 2019.

Chenglin Yang, Lingxi Xie, Chi Su, and Alan L. Yuille. Snapshot distillation: Teacher-student optimization in one generation. *ArXiv*, abs/1812.00123, 2018.

Chenglin Yang, Lingxi Xie, Siyuan Qiao, and Alan L. Yuille. Training deep neural networks in generations: A more tolerant teacher educates better students. In *AAAI*, 2019.

Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2017.

Ruichi Yu, Ang Li, Vlad I. Morariu, and Larry S. Davis. Visual relationship detection with internal and external linguistic knowledge distillation. *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 1068–1076, 2017.

Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*, 2016.

Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. *ArXiv*, abs/1905.08094, 2019.