KNOWLEDGE DISTILLATION FROM FEW SAMPLES

Anonymous authors
Paper under double-blind review

ABSTRACT

Current knowledge distillation methods require full training data to distill knowledge from a large "teacher" network to a compact "student" network by matching certain statistics between "teacher" and "student" such as softmax outputs and feature responses. This is not only time-consuming but also inconsistent with human cognition in which children can learn knowledge from adults with few examples. This paper proposes a novel and simple method for knowledge distillation from few samples. Taking the assumption that both "teacher" and "student" have the same feature map sizes at each corresponding block, we add a 1×1 conv-layer at the end of each block in the student network, and align the block-level outputs between "teacher" and "student" by estimating the parameters of the added layer with limited samples. We prove that the added layer can be absorbed into the previous conv-layer so that no extra parameters and computation are introduced. Experiments show that the proposed method can recover a student network's top-1 accuracy on ImageNet from 0.2% to 62.7% with just 1000 samples in a few minutes, and is effective on various ways for constructing student networks.

1 INTRODUCTION

Deep neural networks (DNNs) have demonstrated extraordinary success in a variety of fields such as computer vision (Krizhevsky & Hinton, 2012; He et al., 2016), speech recognition (Hinton et al., 2012), and natural language processing (Mikolov et al., 2010). However, DNNs are resource-hungry which hinders their wide deployment to some resource-limited scenarios, especially low-power embedded devices in the emerging Internet-of-Things (IoT) domain. To address this limitation, extensive works have been done to accelerate or compress deep neural networks. Putting those works on designing (Chollet, 2016) or automatically searching efficient network architecture aside (Zoph & Le, 2016), most studies try to optimize DNNs from four perspectives: network pruning (Han et al., 2016; Li et al., 2016), network decomposition (Denton et al., 2014; Jaderberg et al., 2014), network quantization (or low-precision networks) (Gupta et al., 2015; Courbariaux et al., 2016; Rastegari et al., 2016) and knowledge distillation (Hinton et al., 2015; Romero et al., 2015).

Among these method categories, knowledge distillation is somewhat different since it allows designable output network ("student" net). The concept was proposed by (Bucila et al., 2006; Ba & Caruana, 2014; Hinton et al., 2015) for transferring knowledge from a large "teacher" model to a compact yet efficient "student" model by matching certain statistics between "teacher" and "student". Further research introduced various kinds of matching mechanisms in the field of DNN optimization. The distillation procedure designs a loss function based on the matching mechanisms and enforces the loss during a full training process. Hence, all these methods usually require time-consuming training procedure along with fully annotated large-scale training dataset.

Meanwhile, some network pruning (Li et al., 2016; Liu et al., 2017) and decomposition (Zhang et al., 2016; Kim et al., 2016) methods can produce extremely small networks, but with large accuracy drops so that time-consuming fine-tuning is required for possible accuracy recovery. Usually, it may still not be able to recover the accuracy drops with the original cross-entropy loss due to its low representation capacity. Hence, knowledge distillation may be used to alleviate the problem, since the small student's performance can sometimes be trained to match the teacher's. For instance, Crowley et al. (2017) uses cheap group convolutions and pointwise convolutions to build a small student network and adopts knowledge distillation to transfer knowledge from a full-sized "teacher" network to the "student" network. However, it still suffers from high training cost.

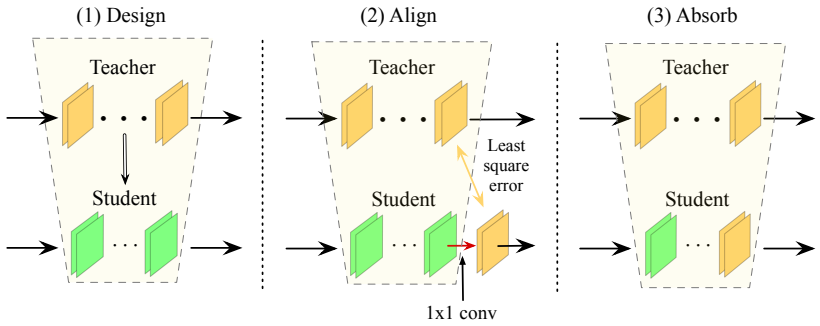


Figure 1: Three steps of our few-sample knowledge distillation. (1) design student network from scratch or by compressing teacher-net; (2) add 1×1 conv-layer at the end of each block of student-net (before ReLU), and align teacher and student by estimating the parameter using least-squared regression; (3) absorb the added 1×1 conv-layer into previous conv-layer to obtain final student-net.

As is known, children can learn knowledge concept from adults with few examples. This cognition phenomenon has motivated the development of the few-shot learning (Fei-Fei et al., 2006; Bart & Ullman, 2005), which aims to learn information about object categories from a few training samples, and focuses more on image classification task. Nevertheless, it inspires people to consider the possibility of knowledge distillation from few samples. Some recent works on knowledge distillation address this problem by constructing "pseudo" training data (Kimura et al., 2018; Lopes et al., 2017) with complicated heuristics and heavy engineering, which are still costly.

This paper proposes a novel and simple 3-step method for few-sample knowledge distillation (FSKD) as illustrated in Figure 1, including student-net design, teacher-student alignment, and absorbing added conv-layer. We assume that both "teacher" and "student" nets have the same feature map sizes at each corresponding block. However, the relatively small student-net can be obtained in various ways, such as pruning/decomposing the teacher-net, and fully redesigned network with random initialization. We add a 1×1 conv-layer at the end of each block of the student-net and align the block-level outputs between "teacher" and "student", which is done by estimating the parameters of the added layer with few samples using least square regression. Since the added 1×1 conv-layers have relatively few parameters, we can get a good approximation from a small number of samples. We further prove that the added 1×1 conv-layer can be absorbed by the previous conv-layer when certain conditions fulfill, without introducing extra parameters and computation cost to the student-net. Our major contributions can be summarized as follows:

(1) To the best of our knowledge, we are the first to show that knowledge distillation can be done with few samples within minutes on desktop PC.

(2) The proposed few-sample knowledge distillation (FSKD) method is widely applicable not only for fully redesigned student networks but also for compressed networks from pruning and decomposition-based methods.

(3) We demonstrate significant performance improvement of the student network by FSKD. For example, FSKD can recover a student network's top-1 accuracy on ImageNet from only 0.2% to 62.7% with just 1000 samples.

2 Related Work

Knowledge Distillation (KD) transfers knowledge from a pre-trained large "teacher" network (or even an ensemble of networks) to a small "student" network, for facilitating the deployment at test time. Originally, this is done by regressing the softmax output of the teacher model (Hinton et al., 2015). The soft continuous regression loss used here provides richer information than the label based loss, so that the distilled model can be more accurate than training on labeled data with cross-entropy loss. Later, various works have extended this approach by matching other statistics, including intermediate feature responses (Romero et al., 2015; Chen et al., 2016), gradient (Srinivas & Fleuret, 2018), distribution (Huang & Wang, 2017), Gram matrix (Yim et al., 2017), etc. These methods require a large amount of data (known as the "transfer set") to transfer the knowledge, whereas we aim to provide a solution in the case that we only have limited number of samples. We need *emphasize* that our FSKD has a totally different philosophy on aligning intermediate responses to the very close knowledge distillation method FitNet (Romero et al., 2015). FitNet re-trains the whole student-net with intermediate supervision using a larger amount of data, while our FSKD only

estimates parameters for the added 1×1 conv-layer with few samples. We will verify in experiments that our FSKD is not only more efficient but also more accurate than FitNet.

Network Pruning methods obtain a small network by pruning weights from a trained larger network, which can keep the accuracy of the larger model if the prune ratio is set properly. Han et al. (2015) proposes to prune the individual weights that are near zero. Recently, channel pruning has become increasingly popular thanks to its better compatibility with off-the-shelf computing libraries, compared with weights pruning. Different criteria have been proposed to select the channel to be pruned, including norm of weights (Li et al., 2016), scales of multiplicative coefficients (Liu et al., 2017), statistics of next layer (Luo et al., 2017), etc. Meanwhile, **Network Decomposition** methods try to factorize heavy layers in DNNs into multiple lightweight ones. For instance, it may adopt low-rank decomposition to fully-connection layers (Denton et al., 2014), and different kinds of tensor decomposition to conv-layers (Zhang et al., 2016; Kim et al., 2016). However, aggressive network pruning or network decomposition usually lead to large accuracy drops, thus fine-tuning is required to alleviate those drops (Li et al., 2016; Liu et al., 2017; Zhang et al., 2016). As aforementioned, KD is more accurate than directly training on labeled data, it is of great interest to explore KD on extremely pruned or decomposed networks, especially under the few-sample setting.

Learning with few samples has been extensively studied under the concept of one-shot or few-shot learning. One category of methods directly model few-shot samples with generative models (Fei-Fei et al., 2006; Lake et al., 2011), while most other methods study the problem under the notion of transfer learning (Bart & Ullman, 2005; Ravi & Larochelle, 2017). In the latter category, meta-learning methods (Vinyals et al., 2016; Finn et al., 2017) solve the problem in a learning to learn fashion, which has been recently gaining momentum due to their application versatility. Most studies are devoted to the image classification task, while it is a less-explored problem for knowledge distillation from few samples. Recently, some works tried to address this problem. Kimura et al. (2018) constructs pseudo-examples using the inducing point method, and develops a complicated algorithm to optimize the model and pseudo-examples alternatively. Lopes et al. (2017) records per-layer meta-data for the teacher network in order to reconstruct a training set, and then adopts a standard training procedure to obtain the student network. Both are still very costly due to the complicated and heavy training procedure. On the contrary, we aim for a simple solution for knowledge distillation from few samples.

3 Few-Sample Knowledge Distillation (FSKD)

3.1 Overview

Our few-sample knowledge distillation (FSKD) method consists of three steps as shown in Figure 1. *First*, we design a student network either by pruning/decomposing the teacher network, or by fully redesigning a small student network with random initialization. *Second*, we add a 1×1 conv-layer at the end of each block of the student network and align the block-level outputs between "teacher" and "student" by estimating the parameters for the added layer from few samples. *Third*, we absorb the added 1×1 conv-layer into the previous conv-layer, so that no extra parameters and computations are introduced into the student network.

Two reasons make this idea work efficiently. *First*, the 1×1 conv-layers have relatively few parameters, which does not require too much data for the estimation. *Second*, the block-level output from teacher network provides rich information as shown in FitNet (Romero et al., 2015). Below, we will first provide the theoretical derivation why the added 1×1 conv-layer could be absorbed into the previous conv-layer. Then we provide details on how we do the block-level output alignment.

3.2 Absorbable 1×1 conv-layer

Let's first give some mathematic notions for different kinds of convolutions before moving to the theoretical derivation. A *regular convolution* consists of multi-channel and multi-kernel filters which build both cross-channel correlations and spatial correlations. Formally, a regular convolution layer can be represented by a 4-dimensional tensor $\mathbf{W} \in \mathbb{R}^{n_o \times n_i \times k_w \times k_h}$, where n_o and n_i are the number of output and input channels respectively, and k_h and k_w are the spatial height and width of the kernel respectively. Usually, squared spatial kernel is used so that $k_w = k_h = k$. The *point-wise (PW) convolution*, also known as 1×1 convolution (Lin et al., 2014) can be represented by a tensor

$\mathbf{P} \in \mathbb{R}^{n_o \times n_i \times 1 \times 1}$, which is actually degraded from a 4-dimensional tensor to a 2-dimensional matrix. The *depth-wise (DW) convolution* (Chollet, 2016) does per-channel 2D convolution for each input channel, so that it can be represented by a tensor $\mathbf{D} \in \mathbb{R}^{1 \times n_i \times k_h \times k_w}$. Due to no-correlation among output channels, it usually follows by a point-wise convolution to model their correlations. This combination (DW + PW) is also named as *depth-wise separable convolution* by Chollet (2016).

Theorem 1. *A pointwise convolution with tensor $\mathbf{Q} \in \mathbb{R}^{n'_o \times n'_i \times 1 \times 1}$ can be absorbed into the previous convolution layer with tensor $\mathbf{W}$ if the following conditions are satisfied.*

- *c1. $\mathbf{W}$ corresponds to a regular convolution, i.e., $\mathbf{W} \in \mathbb{R}^{n_o \times n_i \times k_w \times k_h}$, or a point-wise convolution, i.e., $\mathbf{W} \in \mathbb{R}^{n_o \times n_i \times 1 \times 1}$.*
- *c2. The output channel number of $\mathbf{W}$ equals to the input channel number of $\mathbf{Q}$, i.e., $n_o = n'_i$.*
- *c3. No non-linear activation layer like ReLU (Nair & Hinton, 2010) between $\mathbf{W}$ and $\mathbf{Q}$.*

Due to the space limitation, we put the proof in the appendix.

The absorbed convolution can be represented by a tensor $\mathbf{W}' \in \mathbb{R}^{n'_o \times n_i \times k_w \times k_h}$. It is easy to have the following corollary when no extra parameters and computations are introduced after absorbing.

Corollary 1. *When the following condition is satisfied for $\mathbf{Q}$,*

- *c4. the number of input and output channels of $\mathbf{Q}$ equals to the number of output channel of $\mathbf{W}$, i.e., $n'_i = n'_o = n_o$, $\mathbf{Q} \in \mathbb{R}^{n_o \times n_o \times 1 \times 1}$,*

the absorbed convolution $\mathbf{W}'$ has the same parameters and computation cost as $\mathbf{W}$, i.e. both $\mathbf{W}', \mathbf{W} \in \mathbb{R}^{n_o \times n_i \times k_w \times k_h}$.

3.3 Block-level Alignment and Absorbing

Now we consider the knowledge distillation problem. Suppose $\mathbf{x}^s$ and $\mathbf{x}^t$ are the block-level output vectors for the student network and teacher network respectively, we add a 1×1 conv-layer $\mathbf{Q}$ at the end of each block of student network before non-linear activation, which satisfies condition $c1 \sim c4$. As $\mathbf{Q}$ is degraded to the matrix form, it can be estimated with least squared regression as

$$\mathbf{Q}^* = \arg\min_{\mathbf{Q}} \sum_{i=1}^{N} \|\mathbf{Q} * \mathbf{x}_i^s - \mathbf{x}_i^t\|, \tag{1}$$

where N is the number of samples used. The number of parameters of $\mathbf{Q}$ is $n_o \times n_o$, where n_o is the number of output channels in the block, which is usually not too large so that we can estimate $\mathbf{Q}$ with a limited number of samples.

Suppose there are M corresponding blocks in the teacher and student networks, to achieve our goal, we need minimize the following loss function

$$\mathcal{L}(\mathbf{Q}_j) = \sum_{j=1}^{M} \sum_{i=1}^{N} \|\mathbf{Q}_j * \mathbf{x}_{ij}^s - \mathbf{x}_{ij}^t\|, \tag{2}$$

where $\mathbf{Q}_j$ represents the tensor for the added 1×1 conv-layer of the j-th block. In practice, we do not optimize this loss all together using SGD due to that too much hyper-parameters need tuning in SGD. Instead, we minimize each of the M term for $\mathbf{Q}_j$ in the student network block-by-block sequentially. Once we estimate $\mathbf{Q}_1$ of the first block by Eq.(1), we immediately replace the older block with the newer merged/absorbed block, and continue this procedure until we reach the last block in the student-net. FSKD has the following advantages:

(1) Parameter $\mathbf{Q}$ can be solved with a small number of samples by aligning the block-level responses between teacher and student networks.

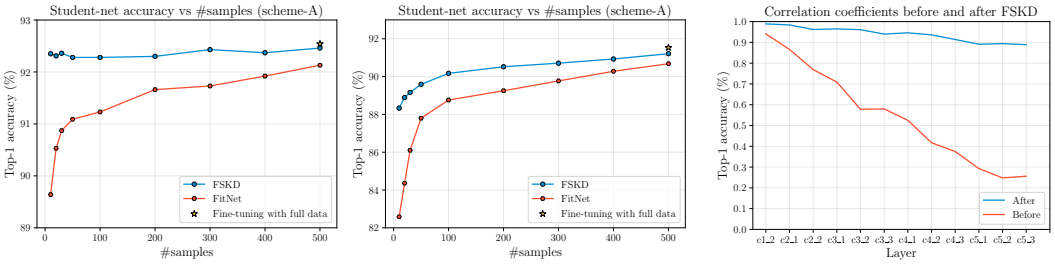
(2) The alignment procedure is very efficient, which can be usually done within several minutes for the entire network.

(3) The alignment procedure itself does not require class labels of input data due to its regression nature. However, if we fully redesign the student network from scratch with random weights, we may leverage SGD on a few labeled samples to initialize the network. Our FSKD can produce significant performance gains over SGD for this kind of student network design.

(4) Our FSKD works extremely well for student network obtained by aggressively pruning/decomposing the teacher network. It beats the standard fine-tuning based solution on the number of data required, processing speed, and accuracy of the output network.

	Top-1 Before (%)	Top-1 After (%)	FLOPs	FLOPs reduced	#Samples
VGG-16	92.66	-	3.13×10^8	-	-
Scheme-A + FSKD	85.42	92.37	2.06×10^8	34%	100
Scheme-A + FitNet	85.42	91.23	2.06×10^8	34%	100
Scheme-A + FSKD	85.42	**92.46**	2.06×10^8	34%	500
Scheme-A + FitNet	85.42	92.13	2.06×10^8	34%	500
Scheme-A + Fine-tuning	85.42	90.25	2.06×10^8	34%	500
Scheme-A + Full fine-tuning	85.42	92.54	2.06×10^8	34%	50000
Scheme-B + FSKD	47.90	90.17	1.33×10^8	58%	100
Scheme-B + FitNet	47.90	88.76	1.33×10^8	58%	100
Scheme-B + FSKD	47.90	**91.21**	1.33×10^8	58%	500
Scheme-B + FitNet	47.90	90.68	1.33×10^8	58%	500
Scheme-B + Fine-tuning	47.90	83.36	1.33×10^8	58%	500
Scheme-B + Full fine-tuning	47.90	91.53	1.33×10^8	58%	50000

Table 1: Performance comparison between FitNet, fine-tuning, FSKD with student-nets from **filter pruning** of VGG-16 with scheme-A/B on CIFAR-10. "Full fine-tuning" means fine-tuning with full training data.



(a) (b)

Figure 2: Accuracy vs #samples on CIFAR-10. Student-net by **filter pruning** of VGG-16 with (a) scheme-A (b) scheme-B.

Figure 3: Layer-level output correlation between "teacher" and "student" before and after FSKD on student-nets (scheme-A) by **filter pruning**.

4 EXPERIMENT

We perform extensive experiments on different image classification datasets to verify the effectiveness of FSKD on various student network construction methods. Student networks can be obtained either from compressing the teacher network or redesigning network structure with random initialization (termed "zero student network"). For the former case, we evaluate FSKD on three previous compression methods, filter pruning (Li et al., 2016), network slimming (Liu et al., 2017), and network decoupling (Guo et al., 2018). We implement the code with PyTorch, and conduct experiments on a desktop with Intel i7-7700K CPU and NVidia 1080TI GPU.

4.1 STUDENT NETWORK FROM COMPRESSING TEACHER NETWORK

FILTER PRUNING

We first obtain the student networks using the filter pruning method proposed by Li et al. (2016), which prunes out convolutional filters according to the L_1 norm of their weights. The L_1 norm of filter weights are sorted and the smallest portion of filters will be pruned to reduce the number of filter-channels in a convolutional layer.

We make a full study of VGG-16 (Simonyan & Zisserman, 2015) on CIFAR-10 dataset to evaluate the performance of FSKD. Following Li et al. (2016), we first prune half of the filters in conv1_1, conv4_1, conv4_2, conv4_3, conv5_1, conv5_2, conv5_3 while keeps the other layer unchanged (*scheme-A*). We also propose another more aggressive pruning scheme named *scheme-B*, which pruned 10% more filters in the aforementioned layers (60% in total), and also pruned 20% filters for the remaining layers. Scheme-A reduces 34% of total FLOPs with 7% of accuracy drop. Scheme-B reduces 58% of FLOPs with almost 50% of accuracy drop. We use those two pruned networks as student networks in this study.

	Top-1 Before (%)	Top-1 After (%)	FLOPs	FLOPs reduced	
VGG-19	93.38	-	7.97×10^8	-	
VGG-19 (70% Pruned) + FSKD	15.90	**93.41**	3.91×10^8	51%	
VGG-19 (70% Pruned) + FitNet	15.90	90.47	3.91×10^8	51%	
VGG-19 (70% Pruned) + Fine-tuning	15.90	62.86	3.91×10^8	51%	
ResNet-164	95.07	-	4.99×10^8	-	-
ResNet-164 (60% Pruned) + FSKD	54.46	**94.19**	2.75×10^8	45%	
ResNet-164 (60% Pruned) + FitNet	54.46	88.94	2.75×10^8	45%	
ResNet-164 (60% Pruned) + Fine-tuning	54.46	60.94	2.75×10^8	45%	
DenseNet-40	94.18	-	5.33×10^8	-	-
DenseNet-40 (60% Pruned) + FSKD	88.24	**93.62**	2.89×10^8	46%	
DenseNet-40 (60% Pruned) + FitNet	88.24	91.37	2.89×10^8	46%	
DenseNet-40 (60% Pruned) + Fine-tuning	88.24	88.98	2.89×10^8	46%	

Table 2: Performance comparison between FSKD, FitNet and fine-tuning on different network structures obtained by **network slimming** with 100 samples randomly selected from CIFAR-10 training set.

	Top-1 Before (%)	Top-1 After (%)	FLOPs	FLOPs reduced	
VGG-19	72.08	-	7.97×10^8	-	
VGG-19 (50% Pruned) + FSKD	9.24	**71.98**	5.01×10^8	37%	
VGG-19 (50% Pruned) + FitNet	9.24	69.52	5.01×10^8	37%	
VGG-19 (50% Pruned) + Fine-tuning	9.24	48.75	5.01×10^8	37%	
ResNet-164	76.56	-	5.00×10^8	-	-
ResNet-164 (40% Pruned) + FSKD	46.07	**76.11**	3.33×10^8	33%	
ResNet-164 (40% Pruned) + FitNet	46.07	73.87	3.33×10^8	33%	
ResNet-164 (40% Pruned) + Fine-tuning	46.07	57.45	3.33×10^8	33%	
DenseNet-40	73.21	-	5.33×10^8	-	-
DenseNet-40 (40% Pruned) + FSKD	60.62	**73.26**	3.71×10^8	30%	
DenseNet-40 (40% Pruned) + FitNet	60.62	71.08	3.71×10^8	30%	
DenseNet-40 (40% Pruned) + Fine-tuning	60.62	62.36	3.71×10^8	30%	

Table 3: Performance comparison between FSKD, FitNet and fine-tuning on different network structures obtained by **network slimming** with 500 samples randomly selected from CIFAR-100 training set.

For the few-sample setting, we randomly select 100 (10 for each category) and 500 (50 for each category) images from the CIFAR-10 training set, and keep them fixed in all experiments. Table 1 lists the results of different methods of recovering a pruned network, including FitNet (Romero et al., 2015), fine-tuning with limited data and full training data. Regarding the processing speed, FSKD can be done in 19.3 seconds for student-net from scheme-B with 500 samples, while FitNet requires 157.3 seconds when converged, which is about 8.1$\times$ slower. This verifies our previous claim that FSKD is more efficient than FitNet. It can be seen that in the few-sample setting, FSKD provides better accuracy recovery than both FitNet and the fine-tuning procedure adopted in Li et al. (2016). For instance, for scheme-B with only 500 samples, FSKD can recover the accuracy from 47.9% to 91.2%, while few-sample fine-tuning can only recover the accuracy to 83.36%. When full training set available, it will take about 30 minutes for full fine-tuning to reach similar accuracy as FSKD. This demonstrates the big advantages of FSKD over full fine-tuning based solutions.

Figure 2 further studies the performance with different amount of training samples available. It can be observed that our FSKD keep outperforming FitNet under the same training samples. In particular, FitNet experiences a noticeable accuracy drop when the number of samples is less than 100, while FSKD can still recover the accuracy of the pruned network to a high level. It is also interesting to note that fine-tuning experiences even larger accuracy drops than FitNet when the data amount is limited. This verifies that knowledge distillation methods like FitNet provides richer information than fine-tuning/training with label based loss.

We further illustrate the per-layer (block) feature responses difference between teacher and student before and after using FSKD in Figure 3. Before applying FSKD, the correlation between teacher and student is broken due to the aggressive compression. However, after FSKD, the per-layer correlations between teacher and student are restored. This verifies the ability of FSKD for recovering lost information. We do see a decreasing trend with layer depth increased, which is possibly due to error accumulation through multiple convolutional layers.

NETWORK SLIMMING

We then study the student network from another channel pruning method named network slimming (Liu et al., 2017), which removes insignificant filter channels and corresponding feature maps using

	Top-1 Before (%)	Top-1 After (%)	GFLOPs	FLOPs Reduced
VGG-16 (teacher)	68.4	-	15.47	-
Decoupled ($T = 1$mix) + FSKD	0.12	51.3	2.73	82.4%
Decoupled ($T = 2$) + FSKD	0.24	62.7	3.76	75.7%
Decoupled ($T = 3$) + FSKD	1.57	67.1	5.54	64.2%
Decoupled ($T = 4$) + FSKD	54.6	67.6	7.31	52.7%
ResNet-18 (teacher)	67.1	-	1.83	-
Decoupled ($T = 2$) + FSKD	0.21	49.5	0.55	70.0%
Decoupled ($T = 3$) + FSKD	3.99	61.9	0.75	59.0%
Decoupled ($T = 4$) + FSKD	26.5	65.1	0.95	48.1%
Decoupled ($T = 5$) + FSKD	53.6	66.3	1.15	37.2%

Table 4: Performance of FSKD on different student nets obtained by **network decoupling** VGG-16 and ResNet-18 with different parameters T on ImageNet dataset.

sparsified channel scaling factors. Network slimming consists of three steps: sparse regularized training, pruning and fine-tuning. Here, we replace the time-consuming fine-tuning step with our FSKD, and follow the original paper (Liu et al., 2017) to conduct experiments to prune different network architectures on different datasets. We apply FSKD on networks pruned from VGG-19, ResNet-164, and DenseNet-40 (Huang et al., 2017), on both CIFAR-10 and CIFAR-100 datasets. Table 2 lists results on CIFAR-10, while Table 3 lists results on CIFAR-100. Note that the pruning-ratio (like 70% in Table 2) means the portion of channels that are removed in comparison to the total number of channels in the network. It shows that our FSKD consistently outperforms FitNet and fine-tuning with a noticeable margin under the few-sample setting on all evaluated networks and both datasets.

NETWORK DECOUPLING

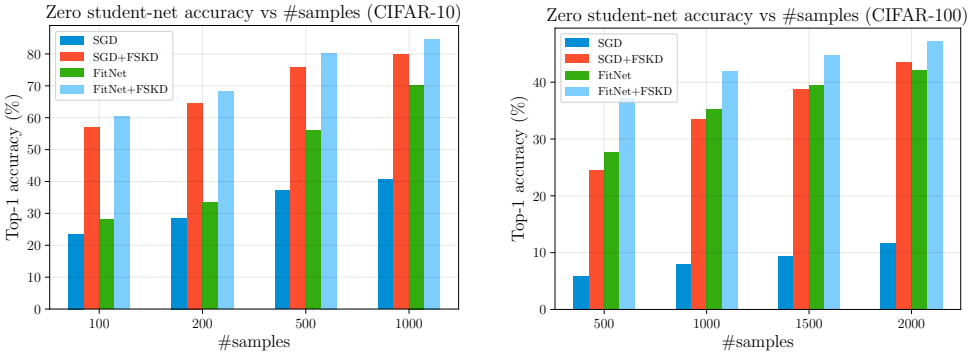
Network decoupling (Guo et al., 2018) decomposes a regular convolution layer into the sum of several blocks, where each block consists of a depth-wise (DW) convolution layer and a point-wise (PW, 1×1) convolution layer. The ratio of compression increases as the number of blocks decreases, but the accuracy of the compressed model will also drop. Since each decoupled block ends with a 1×1 convolution, we can apply FSKD at the end of each decoupled block.

Following (Guo et al., 2018), we obtain student networks by decoupling VGG-16 and ResNet-18 pre-trained on ImageNet with different T values, where T stands for the number of DW + PW blocks that a conv-layer decouples out. For VGG-16, we also decouple half of the convolution layer with $T = 1$ and the other half $T = 2$, and denote the case as "$T = 1$mix". We evaluate the resulted network performance on the validation set of the ImageNet classification task. We randomly select one image from each of the 1000 classes in ImageNet training set to obtain 1000 samples as our FSKD training set. Table 4 shows the top-1 accuracy of student network before and after applying FSKD on VGG-16 and ResNet-18.

It is quite interesting to see that in the case of $T = 1$mix for VGG-16 and $T = 2$ for ResNet-18, we can recover the accuracy of student network from nearly random guess (0.12%, 0.21%) to a much higher level (51.3% and 49.5%) with only 1000 samples. In all the other cases, FSKD can recover the accuracy of a highly-compressed network to be comparable with the original network. One possible explanation is that the highly-compressed networks still inherit some representation power from the teacher network i.e., the depth-wise 3×3 convolution, while lacking the ability to output meaningful predictions due to the inaccurate/degraded 1×1 convolution. The FSKD calibrates the 1×1 convolution by aligning the block-level response between "teacher" and "student" so that the lost information in 1×1 convolution is compensated, and reasonable recovery is achieved.

4.2 ZERO STUDENT NETWORK

Finally, we evaluate FSKD on fully redesigned student network with a different structure from the teacher and random initialized parameters (named as zero student-net). We conduct experiments on CIFAR-10 and CIFAR-100 with VGG-19 as the teacher network and a shallower VGG-13 as the student network. Due to the similar structure between VGG-13 and VGG-19, they can be easily aligned in block-level.



(a) (b)

Figure 4: Accuracy vs #samples for **zero student network** on (a) CIFAR-10, (b) CIFAR-100.

The random initialized network does not contain any information about the training set. Simply training this network with few samples will lead to poor generalization ability, as shown in Figure 4. We propose two schemes to combine FSKD with training: SGD+FSKD and FitNet+FSKD. In the SGD+FSKD case, we first use stochastic gradient descent (SGD) to train the student network on the given number of labeled few samples with 150 epochs (multi-step learning-rate decay at every 50 epochs from 0.01 to 0.0001). Then we apply FSKD to the obtained student network using the same few-sample set. We repeat these two steps until the training loss converges. In the FitNet+FSKD case, we keep the same few-sample set, and simply replace the SGD with FitNet to add supervision on intermediate responses during training.

We compare the results from four different recovery methods: running SGD until convergence, SGD+FSKD, running FitNet until convergence, and FitNet+FSKD. In order to better simulate the few-sample setting, we do not apply data augmentation to the training set. We randomly pick 100, 200, 500, 1000 samples from the 10-category CIFAR-10 training set, and 500, 1000, 1500, 2000 samples from the 100-category CIFAR-100 training set, and keep these few-sample sets fixed in this study.

Figure 4 shows the comparison results on the four recovery methods and four few-sample sets. It shows that FSKD+SGD takes a big jump over pure SGD, and FSKD+FitNet also takes a big jump over pure FitNet. FSKD+SGD performs much better than FitNet on CIFAR-10, while this is not true on CIFAR-100. There are two possible reasons. *First*, we did not enable data augmentation so that few-sample SGD is underfitting, which provides much less information than what the student-net can get from the teacher in FitNet. *Second*, CIFAR-100 is much more difficult than CIFAR-10 so that the performance is more sensitive to the number of samples. However, FSKD+SGD can still achieve accuracy on par with FitNet. We should also note here that due to no data augmentation and few-sample settings, the accuracy of all our evaluated methods has gaps to that of the teacher networks. Nevertheless, it still demonstrates the advantages of our FSKD over FitNet and fine-tuning based methods.

5 Conclusion

We proposed a novel, simple and effective method for knowledge distillation from few samples. The method works for student networks constructed in various ways, including compression from teacher networks and fully redesigned networks with random initialization on various datasets. Notably, we can recover a student network with 0.2% top-1 accuracy to 62.7% out of 68.4% with just 1000 samples on ImageNet dataset. The method outperforms existing knowledge distillation methods by a large margin in the few-sample setting, while requires significantly less computation budget. This advantage will bring many potential applications and extensions for FSKD.

REFERENCES

Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *NIPS*, 2014.

Evgeniy Bart and Shimon Ullman. Cross-generalization: Learning novel classes from a single example by feature replacement. In *CVPR*. IEEE, 2005.

Cristian Bucila, Rich Caruana, Alexandru Niculescu-Mizil, et al. Model compression. In *SIGKDD*. ACM, 2006.

Tianqi Chen, Ian Goodfellow, Jonathon Shlens, et al. Net2net: Accelerating learning via knowledge transfer. In *ICLR*, 2016.

François Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint arXiv:1610.02357*, 2016.

M. Courbariaux, Y. Bengio, Jean-Pierre David, et al. Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. In *ICLR*, 2016.

Elliot J Crowley, Gavin Gray, and Amos Storkey. Moonshine: Distilling with cheap convolutions. *arXiv preprint arXiv:1711.02613*, 2017.

Emily Denton, Zaremba, Yann Lecun, et al. Exploiting linear structure within convolutional networks for efficient evaluation. In *NIPS*, 2014.

Li Fei-Fei, Rob Fergus, Pietro Perona, et al. One-shot learning of object categories. *IEEE Trans PAMI*, 2006.

Chelsea Finn, Pieter Abbeel, Sergey Levine, et al. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

Jianbo Guo, Yuxi Li, Weiyao Lin, Yurong Chen, and Jianguo Li. Network decoupling: From regular to depthwise separable convolutions. In *BMVC*, 2018.

Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, et al. Deep learning with limited numerical precision. In *ICML*, 2015.

Song Han, Jeff Pool, John Tran, William Dally, et al. Learning both weights and connections for efficient neural network. In *NIPS*, 2015.

Song Han, Huizi Mao, Bill Dally, et al. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *NIPS*, 2016.

K. He, X. Zhang, J. Sun, et al. Deep residual learning for image recognition. In *CVPR*, 2016.

G. Hinton, O. Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Geoffrey Hinton, Li Deng, Dong Yu, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6), 2012.

Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *CVPR*, 2017.

Zehao Huang and Naiyan Wang. Like what you like: Knowledge distill via neuron selectivity transfer. *arXiv preprint arXiv:1707.01219*, 2017.

M. Jaderberg, A. Vedaldi, A. Zisserman, et al. Speeding up convolutional neural networks with low rank expansions. In *BMVC*, 2014.

Y. Kim, E. Park, S. Yoo, et al. Compression of deep convolutional neural networks for fast and low power mobile applications. In *ICLR*, 2016.

Akisato Kimura, Zoubin Ghahramani, Koh Takeuchi, et al. Few-shot learning of neural networks from scratch by pseudo example optimization. In *BMVC*, 2018.

A. Krizhevsky and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

Brenden Lake, Ruslan Salakhutdinov, Jason Gross, et al. One shot learning of simple visual concepts. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 33, 2011.

Hao Li, Asim Kadav, Durdanovic I, et al. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

Min Lin, Qiang Chen, and Shuicheng nd others Yan. Network in network. In *ICLR*, 2014.

Zhuang Liu, Jianguo Li, Zhiqiang Shen, et al. Learning efficient convolutional networks through network slimming. In *ICCV*, 2017.

Raphael Gontijo Lopes, Stefano Fenu, Thad Starner, et al. Data-free knowledge distillation for deep neural networks. *arXiv preprint arXiv:1710.07535*, 2017.

J. Luo, J. Wu, W. Lin, et al. Thinet: A filter level pruning method for deep neural network compression. In *ICCV*, 2017.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, et al. Recurrent neural network based language model. In *INTERSPECH*, 2010.

Vinod Nair and Geoffrey Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, 2016.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, et al. Fitnets: Hints for thin deep nets. In *ICLR*, 2015.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

Suraj Srinivas and Francois Fleuret. Knowledge transfer with jacobian matching. *arXiv preprint arXiv:1803.00443*, 2018.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, et al. Matching networks for one shot learning. In *NIPS*, 2016.

Junho Yim, Donggyu Joo, Jihoon Bae, et al. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *CVPR*, 2017.

X. Zhang, J. Zou, J. Sun, et al. Accelerating very deep convolutional networks for classification and detection. *IEEE TPAMI*, 38(10), 2016.

B. Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *ICLR*, 2016.

APPENDIX: PROOF OF THEOREM 1

Proof. When $\mathbf{W}$ is a point-wise convolution with tensor $\mathbf{W} \in \mathbb{R}^{n_o \times n_i \times 1 \times 1}$, both $\mathbf{W}$ and $\mathbf{Q}$ are degraded into matrix form. It is obvious that when condition $c1 \sim c3$ satisfied, the theorem holds with $\mathbf{W}' = \mathbf{Q} * \mathbf{W}$ in this case, where $*$ indicates matrix multiplication.

When $\mathbf{W}$ is a regular convolution with tensor $\mathbf{W} \in \mathbb{R}^{n_o \times n_i \times k_w \times k_h}$, the proof is non-trivial. Fortunately, recent work on network decoupling (Guo et al., 2018) presents an important theoretic result as the basis of our derivation.

Lemma 1. *Regular convolution can be exactly expanded to a sum of several depth-wise separable convolutions. Formally,* $\forall\ \mathbf{W} \in \mathbb{R}^{n_o \times n_i \times k_w \times k_h}$, $\exists\ \{\mathbf{P}_k, \mathbf{D}_k\}_{k=1}^{K}$, *where* $\mathbf{P}_k \in \mathbb{R}^{n_o \times n_i \times 1 \times 1}$, $\mathbf{D}_k \in \mathbb{R}^{1 \times n_i \times k_h \times k_w}$,

$$\begin{aligned} s.t.\ \ & (a) K \leq k_h k_w; \\ & (b) \mathbf{W} = \sum_{k=1}^{K} \mathbf{P}_k \circ \mathbf{D}_k, \end{aligned} \tag{3}$$

where $\circ$ *is the compound operation, which means performing* $\mathbf{D}_k$ *before* $\mathbf{P}_k$.

Please refer to Guo et al. (2018) for the details of proof for this Lemma. When $\mathbf{W}$ is applied to an input patch $\mathbf{x} \in \mathbb{R}^{n_i \times k_h \times k_w}$, we obtain a response vector $\mathbf{y} \in \mathbb{R}^{n_o}$ as

$$\mathbf{y} = \mathbf{W} \otimes \mathbf{x}, \tag{4}$$

where $y_o = \sum_{i=1}^{n_i} W_{o,i} \otimes x_i, o \in [n_o], i \in [n_i]$, and $\otimes$ here means convolution operation. $W_{o,i} = \mathbf{W}[o, i, :, :]$ is a tensor slice along the i-th input and o-th output channels, $x_i = \mathbf{x}[i, :, :]$ is a tensor slice along the i-th channel of 3D tensor $\mathbf{x}$.

When point-wise convolution $\mathbf{Q}$ is added after $\mathbf{Q}$ without non-linear activation between them, we have

$$\mathbf{y}' = \mathbf{Q} \circ (\mathbf{W} \otimes \mathbf{x}). \tag{5}$$

With Lemma-1, we have

$$\mathbf{y}' = (\mathbf{Q} \circ \sum_{k=1}^{K} \mathbf{P}_k \circ \mathbf{D}_k) \otimes \mathbf{x} = (\sum_{k=1}^{K} (\mathbf{Q} * \mathbf{P}_k) \circ \mathbf{D}_k) \otimes \mathbf{x} \tag{6}$$

As both $\mathbf{Q}$ and $\mathbf{P}_k$ are degraded into matrix form, denoting $\mathbf{P}_k^{'} = \mathbf{Q} * \mathbf{P}_k$ and $\mathbf{W}' = \sum_{k=1}^{K} \mathbf{P}_k^{'} \circ \mathbf{D}_k$, we have $\mathbf{y}' = \mathbf{W}' \circ \mathbf{x}$. This proves the case when $\mathbf{W}$ is a regular convolution. □