Few Sample Knowledge Distillation for Efficient Network Compression

Tianhong Li
MIT
tianhong@mit.edu

Jianguo Li
Intel Lab
jianguo.li@intel.com

Zhuang Liu
UC Berkeley
zhuangl@berkeley.edu

Changshui Zhang
Tsinghua University
zcs@tsinghua.edu.cn

Abstract

Deep neural network compression techniques such as pruning and weight tensor decomposition usually require fine-tuning to recover the prediction accuracy when the compression ratio is high. However, conventional fine-tuning suffers from the requirement of full training data and the time-consuming training procedure. This paper proposes a novel solution for knowledge distillation from few unlabeled samples to ***realize both data efficiency and training/processing efficiency****. We treat the original network as "teacher-net" and the compressed network as "student-net". A 1×1 convolution layer is added at the end of each layer block of the student-net, and we fit the block-level outputs of the student-net to the teacher-net by estimating the parameters of the added layers. We prove that the added layer does not introduce extra parameters and computation cost during inference. Experiments on multiple datasets and network architectures verify the method's effectiveness on student-nets obtained by various network pruning and weight decomposition methods. Our method can recover student-net's accuracy to the same level as conventional fine-tuning methods in minutes using only 1% of the full training data.* [1]

1. Introduction

Deep neural networks have demonstrated extraordinary success in a variety of fields such as computer vision [19, 11], speech recognition [12], and natural language processing [25]. However, their resource-hungry nature greatly hinders their wide deployment in some resource-limited scenarios. To address this limitation, many works have been done to accelerate and/or compress neural networks, among which network pruning [9, 21] and network weight decomposition [5, 16] are particularly popular due to their competitive performance and compatibility.

Network pruning [21, 22] and weight decomposition [34, 17] methods can produce extremely compressed networks, but they usually suffer from significant accuracy drops so that fine-tuning is required for possible accuracy recovery. However, current fine-tuning practices are not only time-consuming, but also require the fully annotated large-scale training dataset, which may be unavailable in many cases due to privacy or confidential issues, e.g. for medical data. As a result, when the compression ratio is high, current methods may not recover the dropped accuracy if there are very few annotated training examples or strict training time-budget.

To solve this problem, one may ask if it is possible to utilize the knowledge from the original large network to the compressed compact network, since the latter has a similar block-level structure as the former, and already inherits some the feature representation power from it. A natural solution is to use knowledge distillation (KD) [3, 1, 13], a method for transferring the knowledge from a large "teacher" model to a compact yet efficient "student" model by matching certain statistics between them. Further research introduced various kinds of matching mechanisms [28, 30, 15]. The distillation procedure typically designs a loss function based on the matching mechanisms and optimizes the loss during a full training process. As a result, these methods still require time-consuming training procedure along with fully annotated large-scale training dataset, thus fail to meet our goal of training/processing efficiency and high sample-efficiency.

This paper addresses this issue with a simple and novel method, namely few-sample knowledge distillation (FSKD), for efficient network compression, where *"efficient" here means both training/processing efficiency and data-sample efficiency*. As shown in Figure 1, FSKD contains three steps: compressing teacher-net to obtain student-net, aligning student-teacher with the added layers in student-net, and absorbing the added layers. We first obtain the student-net by pruning or decomposing the teacher-net using existing methods, and ensure that both teacher-net and student-net have the same feature map sizes at each corresponding layer block. Second, we add a 1×1 conv-layer at the end of each block of the student-net. We then forward the few samples of unlabeled data to both the teacher and student, and align the block-level outputs of the student with the teacher by estimating the parameters of the added layer, using least

[1] Work was done when Tianhong Li was an intern at Intel Labs. Source codes will be open to the public.

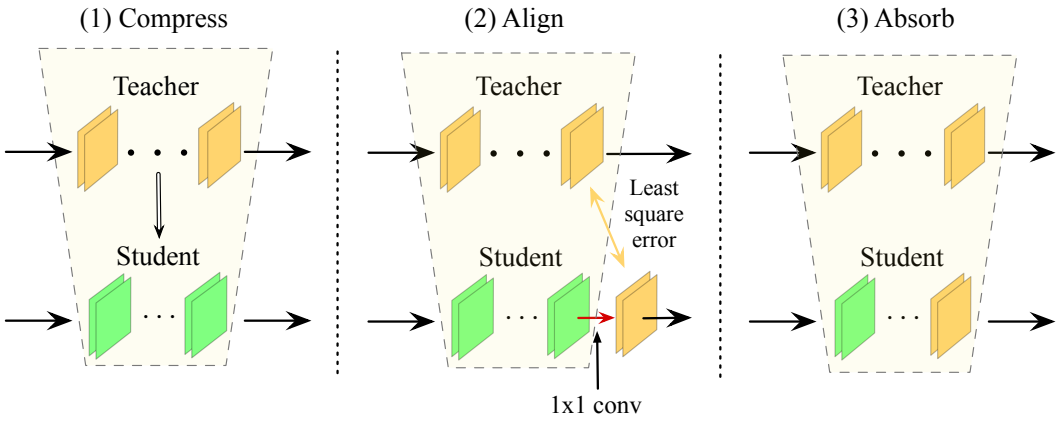

Figure 1: Three-step of few-sample knowledge distillation. (1) obtain student-net by compressing teacher-net; (2) add 1×1 conv-layer at the end of each block of student-net (before ReLU), and align teacher and student by estimating the parameter using least-squared regression; (3) absorb/merge the added $1\times$ 1 conv-layer into the previous conv-layer to obtain final student-net.

square regressions. Because there are very few parameters to estimate in the added conv-layers, we could obtain a good estimation with a very small amount of unlabeled samples. The aligned student-net has the same number of parameters and computation cost as the original one, since the added 1×1 convolution can be absorbed into the previous convolution layer, as we will show later.

FSKD has many potential usages, especially when full fine-tuning or re-training is not feasible in practice, or the data at hand is only very limited. We name a few concrete cases below. *First*, edge devices have limited computing resources, while FSKD offers the possibility of on-device learning to compress deep models. *Second*, FSKD may help cloud services obtain a compact model when only a few data is uploaded by the customer due to privacy or confidential issues. *Third*, FSKD enables fast model convergence if there is a strict time budget for training/fine-tuning.

Our major contributions can be summarized as follows:

- To the best of our knowledge, we are the first to show that accuracy recovery from compressed network can be done with few unlabeled samples within minutes using knowledge distillation on desktop PC.
- The proposed FSKD method is widely applicable to compressing deep neural networks by pruning or decomposition based methods.
- We demonstrate significant accuracy improvement from FSKD over existing distillation techniques, as well as compression-ratio and speedup gain over traditional pruning/decomposition based methods on various datasets and network architectures.

2. Related Work

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. [10] proposes to prune the individual weights that are near zero. Recently, filter 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 filters to be pruned, including norm of weights [21], scales of multiplicative coefficients [22], statistics of next layer [24], etc. These pruning based methods usually require iterative loop between pruning and fine-tuning for achieving better pruning ratio and speedup. Meanwhile, **Network Decomposition** methods try to factorize parameter-heavy layers into multiple lightweight ones. For instance, it may adopt low-rank decomposition to fully-connection layers [5], and different kinds of weight decomposition to conv-layers [16, 17, 34]. However, aggressive network pruning or network decomposition usually lead to large accuracy drops, thus fine-tuning is a must to alleviate those drops [21, 22, 34].

Knowledge Distillation (KD) transfers knowledge from a pre-trained large teacher-net (or even an ensemble of networks) to a small student-net, for facilitating the deployment at test time. Originally, this is done by regressing the softmax output of the teacher model [13]. 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 [28, 4], gradient [30], distribution [15], Gram matrix [33], etc. More complicatedly, deep mutual learning [35] trains a cohort of student-nets and teaches each other collaboratively with model distillation throughout the training process. All these methods require a large amount of data (known as the "transfer set") to transfer the knowledge. The student-nets in KD are usually designed with random weight initialization. It is of great interest to start the student-nets

with extremely pruned or decomposed networks, and explore a KD solution under the few-sample setting. That is our motivation. We need *emphasize* that FSKD has a quite different philosophy on aligning intermediate responses to the closest knowledge distillation method FitNet [28]. FitNet re-trains the whole student-net with intermediate supervision using a larger amount of data, while FSKD only estimates parameters for the added 1×1 conv-layer with few samples. Experiments verify that FSKD is not only more efficient but also more accurate than FitNet.

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 [6, 20], while most others study the problem under the notion of transfer learning [2, 27]. In the latter category, meta-learning methods [31, 7] 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 still less-explored for knowledge distillation from few samples. Recently, some works tried to address this problem. [18] constructs pseudo-examples using the inducing point method, and develops a complicated algorithm to optimize the model and pseudo-examples alternatively. [23] records per-layer meta-data for the teacher-net in order to reconstruct a training set, and then adopts a standard training procedure to obtain the student-net. Both are very costly due to the complicated and heavy training procedure. On the contrary, we aim for an efficient solution for knowledge distillation from few unlabeled samples.

3. Few Sample Knowledge Distillation

3.1. Overview

Our FSKD method consists of three steps as shown in Figure 1. *First*, we obtain a student-net either by pruning or by decomposing the teacher-net. *Second*, 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 by estimating the parameters of the added layer from few unlabeled samples. *Third*, we absorb/merge the added 1×1 conv-layer into the previous conv-layer so that it will not introducing extra parameters and computation cost into the student-net.

Three reasons make the idea works efficiently. *First*, the compressed student-net inherits partial representation power from the teacher network, so adding 1×1 conv-layer is enough to calibrate the student-net and restore the accuracy. *Second*, the 1×1 conv-layers have relatively fewer parameters, which do not require too many data for the estimation. *Third*, the block-level output from teacher-net provides rich information as shown in FitNet [28]. Below, we will first describe our algorithm for block-level output alignment, and then prove why the added 1×1 conv-layer can be absorbed/merged into the previous conv-layer.

Algorithm 1 Block-coordinate descent algorithm for FSKD

input Student-net $\boldsymbol{s}$, teacher-net $\boldsymbol{t}$, input data $\{\boldsymbol{X}_i\}_{i=1}^N$, number of aligned blocks M, number of iterations T

for $k = 1 : T$ **do**
  Random flip input dataset to obtain $\{\boldsymbol{X}'_i\}_{i=1}^N$;
  for $j = 1 : M$ **do**
    Feed $\{\boldsymbol{X}'_i\}_{i=1}^N$ to the end of the j-th block of $\boldsymbol{t}$, obtain response $\{\boldsymbol{X}^t_{ij}\}$;
    Feed $\{\boldsymbol{X}'_i\}_{i=1}^N$ to the end of the j-th block of $\boldsymbol{s}$, obtain response $\{\boldsymbol{X}^s_{ij}\}$;
    Add 1×1 conv-layer with tensor $\mathbf{Q}_j$ to the end of j-th block of $\boldsymbol{s}$;
    Solve $\mathbf{Q}_j$ with least-square regression based on Equation 1;
    Merge $\mathbf{Q}_j$ into previous conv-layer L_j with tensor $\mathbf{W}_j$ to obtain new tensor $\mathbf{W}'_j$ based on Theorem 1 for student-net;
    Update the j-th block of $\boldsymbol{s}$, to obtain $\boldsymbol{s}'$;
    $\boldsymbol{s} = \boldsymbol{s}'$;
  end for
end for

output absorbed conv-layers $\{\mathbf{W}'_j\}_{j=1}^M$, updated student-net $\boldsymbol{s}'$;

3.2. Block-level Alignment

In this section, we provide details of our block-level output alignment algorithm. Suppose $\boldsymbol{X}^s, \boldsymbol{X}^t \in \mathbb{R}^{n_o \times d}$ are the block-level output in matrix form for the student-net and teacher-net respectively, where d is the per-channel feature map resolution size. We add a 1×1 conv-layer $\mathbf{Q}$ at the end of each block of student-net before non-linear activation. As $\mathbf{Q}$ is degraded to the matrix form, it can be estimated with least squared regression as

$$\mathbf{Q}^* = \arg\min_{\mathbf{Q}} \sum_{i=1}^{N} \|\boldsymbol{Q} * \mathbf{X}^s_i - \boldsymbol{X}^t_i\|, \tag{1}$$

where N is the number of samples used, and "*" here means matrix product. 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-net and the student-net required to align, to achieve our goal, we need minimize the following loss function

$$\mathcal{L}(\mathbf{Q}_j) = \sum_{j=1}^{M} \sum_{i=1}^{N} \|\mathbf{Q}_j * \boldsymbol{X}^s_{ij} - \boldsymbol{X}^t_{ij}\|_F, \tag{2}$$

where $\mathbf{Q}_j$ is the tensor for the added 1×1 conv-layer of the j-th block. In practice, we optimize this loss with a block-coordinate descent (BCD) algorithm [32], which greedily handles each of the M blocks in the student-net sequentially as shown in Algorithm-1 (FSKD-BCD). We can also

minimize this loss using standard SGD on all added parameters (FSKD-SGD). However, FSKD-BCD has the following advantages over FSKD-SGD:

(1) The BCD algorithm processes each block greedily with a sequential update rule, and each $\mathbf{Q}$ can be solved with the same set of small number of samples by aligning the block-level responses between teacher-net and student-net, while standard SGD considers $\{\mathbf{Q}_j\}$ all together which theoretically requires more data.

(2) The BCD algorithm is much more efficient, which can be usually done in less than a minute.

Unless otherwise noted, we use the FSKD-BCD algorithm in our experiments.

3.3. Absorbable 1×1 conv-layer

Now we prove that the added 1x1 conv can be absorbed into the previous convolutional layer without introducing additional parameters and computation cost during inference.

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* $\mathbf{W} \in \mathbb{R}^{n_o \times n_i \times k \times k}$ *to obtain the* ***absorbed tensor*** $\mathbf{W}' = \mathbf{Q} \circ \mathbf{W}$*, where* $\circ$ *is absorbing operator and* $\mathbf{W}' \in \mathbb{R}^{n'_o \times n_i \times k \times k}$ *if the following conditions are satisfied.*

c1. The output channel number of $\mathbf{W}$ *equals to the input channel number of* $\mathbf{Q}$*, i.e.,* $n_o = n'_i$.

c2. No non-linear activation layer like ReLU [26] between $\mathbf{W}$ *and* $\mathbf{Q}$.

The pointwise convolution can be viewed as a linear combination of the kernels in the previous convolution layer. Due to the space limitation, we put the formal proof and the detailed form of the absorbing operator in Appendix-A. The number of output channels of $\mathbf{W}'$ is n'_o, which is different from that of $\mathbf{W}$ (i.e., n_o). It is easy to have the following corollary.

Corollary 1. *When the following condition is satisfied,*

c3. 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 tensor $\mathbf{W}'$ *has the same parameters and computation cost as* $\mathbf{W}$*, i.e. both* $\mathbf{W}', \mathbf{W} \in \mathbb{R}^{n_o \times n_i \times k \times k}$.

This condition is required not only for ensuring the same parameter size and computing cost, but also for ensuring the output-size of current layer matching to the input-size of next layer so that these two layers are connectable.

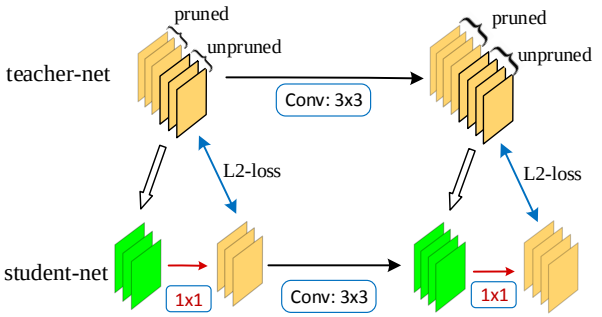

Figure 2: Illustration of FSKD on filter pruning and network slimming. At each block, we copy weights of the unpruned part in teacher-net to student-net, and align the feature maps of student-net to those unpruned feature maps of teacher-net by adding a 1×1 conv-layer (red-color) with L2-loss. The added 1×1 layer can be merged into the previous conv-layer in student-net.

4. Experiments

We perform extensive experiments on different image classification datasets to verify the effectiveness of FSKD on various student-net construction methods. Student-nets can be obtained either by pruning based methods such as filter pruning [21] and network slimming [22], or by decomposition based methods such as network decoupling [8]. We implement the code with PyTorch, and conduct experiments on a desktop PC with Intel i7-7700K CPU and one NVidia 1080TI GPU. The code will be made public avaiable.

4.1. Student-net from Pruning teacher-net

Filter Pruning

We first obtain the student-nets using the filter pruning method [21], which prunes out conv-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 conv-layer. Figure 2 illustrates how FSKD works for block-level alignment on this case. Note that the number of channels in teacher-net may be different from that in student-net. However, we only match the *un-pruned part* of feature-maps in teacher-net to the feature maps in the student-net so that FSKD is applicable in this case.

We make a comprehensive study of VGG-16 [29] on CIFAR-10 dataset to evaluate the performance of FSKD along with three different pruning settings. *First*, we follow the original pruning scheme of [21] and obtain **Prune-A**. *Second*, we propose another more aggressive pruning scheme named **Prune-B**, which prunes 10% more filters in the aforementioned layers, and also pruned 20% filters for the remaining layers. *Third*, since previous works show that one time extremely pruning may yield the pruned network unable to recovery from fine-tuning, while the iteratively pruning and fine-tuning procedure is observed effective to

	Acc.(%)	FLOPs($\times 10^8$)	Speedup	#Param($\times 10^6$)	Pruned
VGG-16	92.66	3.11	1.00×	15	-
Prune-A	85.42	2.06	1.51×	5.3	64%
Prune-B	47.90	1.33	2.34×	3.4	77%
Prune-C	13.05	1.09	2.85×	1.8	88%

Table 1: Prune-A/B/C for **filter pruning** of VGG-16 on CIFAR-10 and their accuracy, FLOPs, #parameters, etc.

	Acc. (%)	#Samples	Time (sec)
VGG-16	92.66	50000	
Prune-A + FSKD	92.37	100	4.8
Prune-A + FitNet	91.23	100	48.5
Prune-A + FSKD	92.46	500	25.5
Prune-A + FitNet	92.13	500	139.2
Prune-A + Fine-tuning	90.25	500	40.4
Prune-A + Full fine-tuning	92.54	50000	1059.6
Prune-B + FSKD	90.17	100	3.7
Prune-B + FitNet	88.76	100	60.1
Prune-B + FSKD	91.21	500	19.3
Prune-B + FitNet	90.68	500	157.1
Prune-B + Fine-tuning	83.36	500	50.3
Prune-B + Full fine-tuning	91.53	50000	1753.4
Prune-C + FSKD	89.55	100	7.4
Prune-C + FitNet	85.09	100	71.3
Prune-C + FSKD	90.41	500	33.5
Prune-C + FitNet	88.31	500	180.3
Prune-C + Fine-tuning	78.13	500	58.7
Prune-C + Full fine-tuning	90.77	50000	2592.3

Table 2: Performance comparison between FitNet, fine-tuning, FSKD by student-nets from **filter pruning** [21] of VGG-16 with pruning scheme A/B/C on CIFAR-10. "Full fine-tuning" uses full training data.

obtain extreme model compression [9, 21, 22], we propose **Prune-C** which iteratively runs the pruning and FSKD procedure as described in Appendix-B for 2 iterations to achieve higher compression rate. Table 1 lists the accuracy, FLOPs and #parameter information for three student-nets obtained by these pruning schemes.

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 2 lists the results of different methods of recovering a pruned network, including FitNet [28], fine-tuning with limited data and full training data [21].

As shown in Table 2, our method is much more efficient and provides better accuracy recovery than both FitNet and the fine-tuning procedure adopted in [21]. For instance, for Prune-B with only 500 samples, our method can recover the accuracy from 47.9% to 91.2% in 19.3s, while FitNet has to take 157.1s to recover the accuracy to 90.7%, and few-sample fine-tuning can only recover the accuracy to 83.4%. When full training set available, it takes about 30 minutes for full fine-tuning to reach similar accuracy as FSKD. This

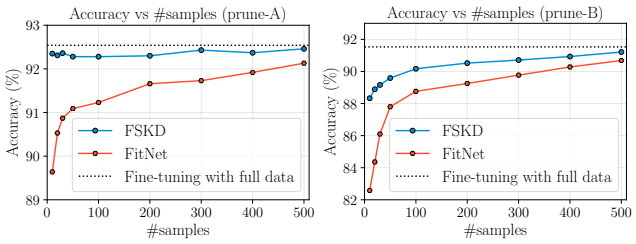

Figure 3: Accuracy vs #samples on CIFAR-10. Student-net Prune-A (left) Prune-B (right) by **filter pruning** [21].

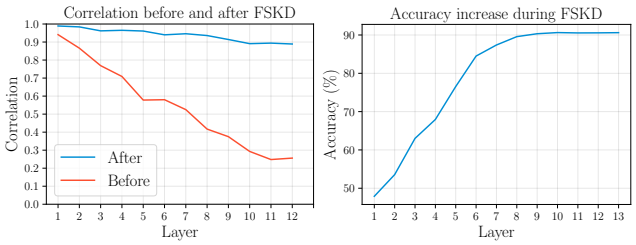

Figure 4: Left: Layer-level output correlation between teacher-net and student-net before and after FSKD on student-nets (Prune-A) by **filter pruning** [21]. Right: Accuracy change during sequentially block-level alignment.

demonstrates the big advantages of FSKD over full fine-tuning based solution.

Figure 3 further studies the performance versus different amount of training samples used. It can be observed that our method keeps outperforming FitNet under the same training samples. In particular, FitNet experience 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.

We further illustrate the per-layer (block) feature responses difference between teacher-net and student-net before and after using FSKD in Figure 4a. Before applying FSKD, the correlation between teacher-net and student-net is broken due to the aggressive compression. However, after FSKD, the per-layer correlations are mostly restored. This verifies the ability of FSKD for recovering lost information. We also show the accuracy change during sequentially block-level alignment in Figure 4b, which clearly demonstrate the effectiveness of our sequentially block-by-block update in the FSKD algorithm.

We use FSKD-BCD in all experiments. Appendix-C evaluates FSKD with more BCD iterations. Appendix-D makes a comparison between FSKD-BCD and FSKD-SGD.

Network Slimming

We then study the student-net from another filter pruning method named network slimming [22], which removes insignificant filter channels and corresponding feature maps using sparsified channel scaling factors. Network slimming consists of three steps: sparse regularized training, pruning

Filter-prune-ratio	Acc. before(%)	Acc. after(%)	FLOPs($\times 10^8$)	Speedup	#Param($\times 10^6$)	Pruned
VGG-19	93.38	-	7.97	1.00×	20	-
70% + FSKD	15.90	93.41	3.91	2.04×	2.2	89%
70% + FitNet	15.90	90.47	3.91	2.04×	2.2	89%
70% + Fine-tuning	15.90	62.86	3.91	2.04×	2.2	89%
ResNet-164	95.07	-	4.99	1.00×	1.7	-
60% + FSKD	54.46	94.19	2.75	1.82×	1.1	37%
60% + FitNet	54.46	88.94	2.75	1.82×	1.1	37%
60% + Fine-tuning	54.46	60.94	2.75	1.82×	1.1	37%
DenseNet-40	94.18	-	5.33	1.00×	1.1	-
60% + FSKD	88.24	93.62	2.89	1.84×	0.5	54%
60% + FitNet	88.24	91.37	2.89	1.84×	0.5	54%
60% + Fine-tuning	88.24	88.98	2.89	1.84×	0.5	54%

Table 3: Performance comparison between FSKD, FitNet and fine-tuning on different network structures obtained by **network slimming** [22] with 100 samples randomly selected from CIFAR-10 training set.

Filter-prune-ratio	Acc. before(%)	Acc. after(%)	FLOPs($\times 10^8$)	Speedup	#Param($\times 10^6$)	Pruned
VGG-19	72.08	-	7.97	1.00×	20	-
50% + FSKD	9.24	71.98	5.01	1.60×	5.0	75%
50% + FitNet	9.24	69.52	5.01	1.60×	5.0	75%
50% + Fine-tuning	9.24	48.75	5.01	1.60×	5.0	75%
ResNet-164	76.56	-	5.00	1.00×	1.7	-
40% + FSKD	46.07	76.11	3.33	1.50×	1.5	14%
40% + FitNet	46.07	73.87	3.33	1.50×	1.5	14%
40% + Fine-tuning	46.07	57.45	3.33	1.50×	1.5	14%
DenseNet-40	73.21	-	5.33	1.00×	1.1	-
40% + FSKD	60.62	73.26	3.71	1.44×	0.71	36%
40% + FitNet	60.62	71.08	3.71	1.44×	0.71	36%
40% + Fine-tuning	60.62	62.36	3.71	1.44×	0.71	36%

Table 4: Performance comparison between FSKD, FitNet and fine-tuning on different network structures obtained by **network slimming** [22] with 500 samples randomly selected from CIFAR-100 training set.

and fine-tuning. Here, we replace the time-consuming fine-tuning step with our FSKD, and follow the original paper [22] to conduct experiments to prune different networks on different datasets. The alignment framework is the same as the filter pruning case as shown in Figure 2.

We apply FSKD on networks pruned from VGG-19, ResNet-164, and DenseNet-40 [14], on both CIFAR-10 and CIFAR-100 datasets. Table 3 lists results on CIFAR-10, while Table 4 lists results on CIFAR-100. Note that the filter-prune-ratio (like 70% in Table 3) means the portion of filters that are removed in comparison to the total number of filters in the network.

The results show that that FSKD consistently outperforms FitNet and fine-tuning with a notable margin under the few-sample setting on all evaluated networks and datasets. This study demonstrates that FSKD is universally applicable to various network structure and pruning methods, and can recover the accuracy of the pruned network using few unlabeled samples to the same level of fine-tuning using fully annotated training dataset.

4.2. Student-net from Decomposing teacher-net

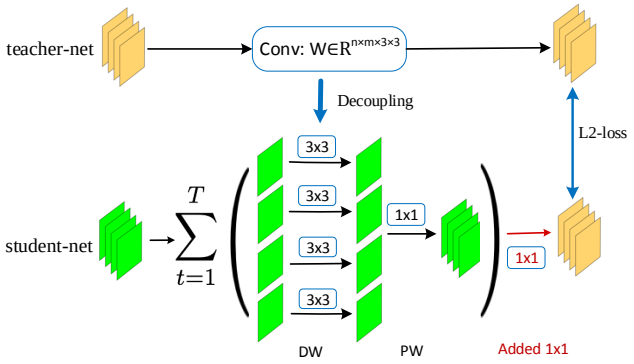

Figure 5: Illustration of FSKD on network decoupling. At each block, we decouple regular-conv in teacher-net into a sum of depth-wise + pointwise conv-layers as the block of student-net, and align the feature maps of student-net to that of teacher-net by adding a 1×1 conv-layer (red-color) with L2-loss. The added 1×1 layer can be merged into previous the pointwise layer in student-net.

	Acc. before(%)	Acc. after (%)	GFLOPs	Speedup	#Param*($\times 10^6$)	Pruned
VGG-16 (teacher)	68.4	-	15.47	1.00×	14.71	-
Decoupled ($T=2$) + FSKD	0.24	62.7	3.76	4.11×	3.35	77.2%
Decoupled ($T=3$) + FSKD	1.57	67.1	5.54	2.79×	5.02	65.8%
Decoupled ($T=4$) + FSKD	54.6	67.6	7.31	2.12×	6.69	54.5%
ResNet-18 (teacher)	67.1	-	1.83	1.00×	11.17	-
Decoupled ($T=2$) + FSKD	0.21	49.5	0.55	3.33×	2.69	75.9%
Decoupled ($T=3$) + FSKD	3.99	61.9	0.75	2.44×	3.95	64.6%
Decoupled ($T=4$) + FSKD	26.5	65.1	0.95	1.92×	5.20	53.4%
Decoupled ($T=5$) + FSKD	53.6	66.3	1.15	1.60×	6.46	42.2%

Table 5: Performance of FSKD on different student-nets obtained by **network decoupling** [8] VGG-16 and ResNet-18 with different parameters T on ImageNet dataset. "$*$" here means that parameters from FC-layer are not counted, only those from conv-layers are counted, since decoupling only handles the conv-layers.

In this section, we apply FSKD on a decomposition based method called network decoupling [8]. Network decoupling decomposes a regular convolution layer into the sum of several *depth-wise separable blocks*, where each such block consists of a depth-wise (DW) conv-layer and a point-wise (PW, 1×1) conv-layer. The compression ratio increases as the number (T) of such 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. Figure 5 illustrates how FSKD works for block-level alignment on this case.

Following [8], we obtain student-nets by decoupling VGG-16 and ResNet-18 pre-trained on ImageNet with different T values. 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 5 shows the top-1 accuracy of student-net before and after applying FSKD on VGG-16 and ResNet-18.

It is quite interesting to see that when T is small, we can recover the accuracy of student-net from nearly random guess (0.24%, 0.21%) to a much higher level (62.7% and 49.5%) with only 1000 samples. One possible explanation is that the highly-compressed networks still inherit some representation power from the teacher-net i.e., the depth-wise 3×3 convolution, while lacking the ability to output meaningful predictions due to the degraded and inaccurate 1×1 convolution. The FSKD calibrates the 1×1 convolution by aligning the block-level responses between teacher-net and student-net so that the lost information in 1×1 convolution is compensated, and reasonable recovery is achieved.

In all the other cases, FSKD can recover the accuracy of a highly-compressed network to be comparable with the original network. This shows that FSKD can be applied to student-net compressed by network decomposition, and that FSKD can achieve great performance on large and difficult dataset such as ImageNet.

	Acc. (%)	#Samples
VGG-16	92.66	50000 (CIFAR-10)
Prune-B + FSKD	90.17	100 (CIFAR-10)
Prune-B + FSKD	90.15	100 (CIFAR-100)
Prune-B + FSKD	91.21	500 (CIFAR-10)
Prune-B + FSKD	91.20	500 (CIFAR-100)

Table 6: Performance comparison between FSKD using samples from CIFAR-10 and CIFAR-100. Student-nets from **filter pruning** [21] of VGG-16 with pruning scheme B on CIFAR-10.

5. Analysis and Discussion

FSKD with Arbitrary Data

In this section, we try to answer the following question: is FSKD totally label-free? For example, is FSKD still valid if the available few samples are arbitrary images and the teacher network never sees these images before? To answer this question, we evaluate FSKD's performance on VGG-16 model trained on CIFAR-10 and compressed using filter pruning (prune-B), with the few samples for FSKD are randomly selected from CIFAR-100 instead of CIFAR-10.

As shown in Table 6, there is no difference in accuracy between FSKD using data from CIFAR-10 or CIFAR-100. This shows that FSKD aligns the student-net with the teacher-net without any information about the labels of the data. Even if the input images are of classes it has never seen before (CIFAR-100 does not include classes in CIFAR-10), FSKD can still recover the student-net to the same accuracy level. This further demonstrates FSKD's potential in situations where only few samples of unlabeled data are available.

Filter Visualization

In this section, we try to answer why FSKD works so well that it can provide almost the same results as that of fine-tuning with full training set. We conduct experiments based on VGG-13 on CIFAR-10. For a given VGG-13 network, We first decouple a conv-layer to obtain one DW conv-layer and one PW conv-layer, as is done in network decoupling

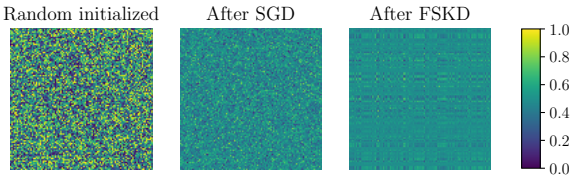

Figure 6: Decouple VGG-13 into DW conv-layer and PW conv-layers, and show one PW conv-layer with random initialization (left), after SGD based fine-tuning (middle), and after FSKD (right). Note values of the PW tensor are scaled into the range (0,1.0) by the min/max values of the tensor for better visualization.

[8]. Then we visualize the PW conv-layer of the decoupled layer. For simplicity, we only visualize the PW conv-layer of the first decoupled layer. We do the visualization on three VGG-13 network with different parameters:

(1) Initialize the VGG-13 network with the MSRA initialization (Figure 6 left).

(2) Run SGD based fine-tuning on 500 samples for VGG-13 with random initialization until convergence (Figure 6 middle).

(3) Run FSKD on 500 samples for VGG-13 with SGD based initialization (Figure 6 right). The teacher-net is also a VGG-13 trained on full CIFAR-10 training set.

It clearly shows that the PW conv-layer before fine-tuning is fairly random on the value range, the one after fine-tuning is less random, while the one after FSKD further starts to show regular patterns, which demonstrates that FSKD can distill the knowledge from the teacher-net to student-net effectively with few samples.

What if student-nets are hand designed?

Our previous experiments construct student-nets by pruning or decomposing teacher-net, and then apply FSKD to boost their performance. People may be interested in the problem "what if student-nets are hand designed with random initialization". In fact, there are two existing works [18, 23] making some pioneer trials on this topic with specific methods for "pseudo" examples generation. Here we conduct experiments to compare our method to these two methods under the same few-sample setting on the same dataset MNIST, for a fair comparison.

Due to different network structures used in these two methods, we make a separate comparison. For [18], the teacher-net has 3 conv-layers followed by 2 fully-connected layers. For [23], the teacher-net is a standard LeNet-5. For both cases, the student-net is the "half-sized" to that of the corresponding teacher-net in terms of the number of feature map channels per conv-layers. As the channel number between student-net and teacher-net is different, we adopt the same strategy as in Figure 2 for filter pruning. That means, the student-net only corresponds to the un-pruned part of the teacher-net, which is obtained the same as [21]. One difference is that we did not copy the weight from un-pruned part of teacher-net to the student-net, while keeping the weight of student-net randomly initialized. For both cases, we compared our FSKD with (1) standard SGD trained on few samples with labeled loss; (2) method from [18] or [23] under the same setting; (3) the FitNet method trained on few samples. In order to better simulate the few-sample setting, we do not apply data augmentation to the training set. We randomly pick 10, 20, 50, 100 and 200 samples from the MNIST training set and keep these few-sample sets fixed across this study. Table 7 lists the comparison results. It shows that SGD with few samples performs the worst, while [18] performs better on the same settings than SGD (still worse on the case of 200 samples). The data-free method [23] performs better than SGD. On both cases, FitNet shows much better performance than SGD and the two compared methods, while our FSKD further outperform FitNet with a noticeable gap, where the gap becomes smaller and smaller when the number of samples increases. This may be due to the following reason. FSKD can be viewed as a special case of FitNet. FitNet optimizes all the weights between teacher-net and student-net using standard SGD algorithm, while FSKD optimizes only the weights from added 1×1 conv-layers in student-net with the BCD algorithm. The BCD algorithm is more sample-efficient than the SGD based algorithm so that FSKD performs both much more efficient and accurate than FitNet on few-sample settings. When training samples used are increased, FSKD will converge to FitNet in the end.

#labeled data	10	20	50	100	200	all-meta-data
SGD	37.91	46.0	66.0	78.3	86.7	-
[18]	44.1	53.9	70.4	80.0	86.6	-
FitNet	86.1	92.3	94.5	96.0	96.5	-
FSKD	**94.4**	**96.5**	**97.0**	**97.5**	**97.8**	-
SGD	57.1	68.3	81.3	85.8	89.7	-
[23]	-	-	-	-	-	92.5
FitNet	90.3	94.2	96.1	96.7	97.3	-
FSKD	**95.5**	**97.2**	**97.6**	**98.0**	**98.1**	-

Table 7: Performance of FSKD on hand designed student-nets with random initialization, compared with previous works [18, 23].

6. Conclusion

We proposed a novel and simple method, namely few-sample knowledge distillation (FSKD) for efficient network compression, while *"efficient" lies in both training/processing efficient and data-sample efficient*. FSKD works for student-nets constructed by either pruning or decomposing teacher-nets with different methods, and demonstrates great efficiency over fine-tuning based solution and advantages over traditional knowledge distillation methods by a large margin in the few-sample setting. This advantage can bring many potential applications for FSKD.

References

[1] J. Ba and R. Caruana. Do deep nets really need to be deep? In *NIPS*, 2014. 1

[2] E. Bart and S. Ullman. Cross-generalization: Learning novel classes from a single example by feature replacement. In *CVPR*. IEEE, 2005. 3

[3] C. Bucila, R. Caruana, A. Niculescu-Mizil, et al. Model compression. In *SIGKDD*. ACM, 2006. 1

[4] T. Chen, I. Goodfellow, J. Shlens, et al. Net2net: Accelerating learning via knowledge transfer. In *ICLR*, 2016. 2

[5] E. Denton, Zaremba, Y. Lecun, et al. Exploiting linear structure within convolutional networks for efficient evaluation. In *NIPS*, 2014. 1, 2

[6] L. Fei-Fei, R. Fergus, P. Perona, et al. One-shot learning of object categories. *IEEE Trans PAMI*, 2006. 3

[7] C. Finn, P. Abbeel, S. Levine, et al. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017. 3

[8] J. Guo, Y. Li, W. Lin, Y. Chen, and J. Li. Network decoupling: From regular to depthwise separable convolutions. In *BMVC*, 2018. 4, 6, 7

[9] S. Han, H. Mao, B. Dally, et al. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *NIPS*, 2016. 1, 4

[10] S. Han, J. Pool, J. Tran, W. Dally, et al. Learning both weights and connections for efficient neural network. In *NIPS*, 2015. 2

[11] K. He, X. Zhang, J. Sun, et al. Deep residual learning for image recognition. In *CVPR*, 2016. 1

[12] G. Hinton, L. Deng, D. 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. 1

[13] G. Hinton, O. Vinyals, J. Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 1, 2

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

[15] Z. Huang and N. Wang. Like what you like: Knowledge distill via neuron selectivity transfer. *arXiv preprint arXiv:1707.01219*, 2017. 1, 2

[16] M. Jaderberg, A. Vedaldi, A. Zisserman, et al. Speeding up convolutional neural networks with low rank expansions. In *BMVC*, 2014. 1, 2

[17] Y. Kim, E. Park, S. Yoo, et al. Compression of deep convolutional neural networks for fast and low power mobile applications. In *ICLR*, 2016. 1, 2

[18] A. Kimura, Z. Ghahramani, K. Takeuchi, et al. Few-shot learning of neural networks from scratch by pseudo example optimization. In *BMVC*, 2018. 3, 8

[19] A. Krizhevsky and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1

[20] B. Lake, R. Salakhutdinov, J. Gross, et al. One shot learning of simple visual concepts. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 33, 2011. 3

[21] H. Li, A. Kadav, D. I, et al. Pruning filters for efficient convnets. *ICLR*, 2017. 1, 2, 4, 5, 7, 8

[22] Z. Liu, J. Li, Z. Shen, et al. Learning efficient convolutional networks through network slimming. In *ICCV*, 2017. 1, 2, 4, 5, 6

[23] R. G. Lopes, S. Fenu, T. Starner, et al. Data-free knowledge distillation for deep neural networks. *arXiv preprint arXiv:1710.07535*, 2017. 3, 8

[24] J. Luo, J. Wu, W. Lin, et al. Thinet: A filter level pruning method for deep neural network compression. In *ICCV*, 2017. 2

[25] T. Mikolov, M. Karafiát, L. Burget, et al. Recurrent neural network based language model. In *INTERSPECH*, 2010. 1

[26] V. Nair and G. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010. 4

[27] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017. 3

[28] A. Romero, N. Ballas, S. E. Kahou, et al. Fitnets: Hints for thin deep nets. In *ICLR*, 2015. 1, 2, 3, 5

[29] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 4

[30] S. Srinivas and F. Fleuret. Knowledge transfer with jacobian matching. *arXiv preprint arXiv:1803.00443*, 2018. 1, 2

[31] O. Vinyals, C. Blundell, T. Lillicrap, et al. Matching networks for one shot learning. In *NIPS*, 2016. 3

[32] Y. Xu and W. Yin. A globally convergent algorithm for nonconvex optimization based on block coordinate update. *Journal of Scientific Computing*, 72(2):700–734, 2017. 3

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

[34] X. Zhang, J. Zou, J. Sun, et al. Accelerating very deep convolutional networks for classification and detection. *IEEE TPAMI*, 38(10), 2016. 1, 2

[35] Y. Zhang, T. Xiang, T. Hospedales, et al. Deep mutual learning. In *CVPR*, 2018. 2