BERT Learns to Teach: Knowledge Distillation with Meta Learning

Anonymous ACL submission

Abstract

We present Knowledge Distillation with Meta Learning (MetaDistil), a simple yet effective alternative to traditional knowledge distillation (KD) methods where the teacher model is fixed during training. We show the teacher network can learn to better transfer knowledge to the student network (i.e., *learning to teach*) with the feedback from the performance of the distilled student network in a meta learning framework. Moreover, we introduce a pilot update mechanism to improve the alignment between the inner-learner and meta-learner in meta learning algorithms that focus on an improved inner-learner. Experiments on various benchmarks show that MetaDistil can yield significant improvements compared with traditional KD algorithms and is less sensitive to the choice of different student capacity and hyperparameters, facilitating the use of KD on different tasks and models.[1]

1 Introduction

With the prevalence of large neural networks with millions or billions of parameters, model compression is gaining prominence for facilitating efficient, eco-friendly deployment for machine learning applications. Among techniques for compression, knowledge distillation (KD) (Hinton et al., 2015) has shown effectiveness in both Computer Vision and Natural Language Processing tasks (Hinton et al., 2015; Romero et al., 2015; Zagoruyko & Komodakis, 2017; Tung & Mori, 2019; Peng et al., 2019; Ahn et al., 2019; Park et al., 2019; Passalis & Tefas, 2018; Heo et al., 2019; Kim et al., 2018; Shi et al., 2021; Sanh et al., 2019; Jiao et al., 2019; Wang et al., 2020b). Previous works often train a large model as the "teacher"; then they fix the teacher and train a "student" model to mimic the behavior of the teacher, in order to transfer the knowledge from the teacher to the student.

However, this paradigm has the following drawbacks: **(1) The teacher is unaware of the student's capacity.** Recent studies in pedagogy suggest student-centered learning, which considers students' characteristics and learning capability, has shown effectiveness improving students' performance (Cornelius-White, 2007; Wright, 2011). However, in conventional knowledge distillation, the student passively accepts knowledge from the teacher, without regard for the student model's learning capability and performance. Recent works (Park et al., 2021; Shi et al., 2021) introduce student-aware distillation by jointly training the teacher and the student with task-specific objectives. However, there is still space for improvement since: **(2) The teacher is not optimized for distillation.** In previous works, the teacher is often trained to optimize its *own* inference performance. However, the teacher is not aware of the need to transfer its knowledge to a student and thus usually does so suboptimally. A real-world analogy is that a PhD student may have enough knowledge to solve problems themselves, but requires additional teaching training to qualify as a professor.

To address these two drawbacks, we propose Knowledge Distillation with Meta Learning (MetaDistil), a new teacher-student distillation framework using meta learning (Finn et al., 2017) to exploit feedback about the student's learning progress to improve the teacher's knowledge transfer ability throughout the distillation process. On the basis of previous formulations of bi-level optimization based meta learning (Finn et al., 2017), we propose a new mechanism called *pilot update* that aligns the learning of the bi-level learners (i.e., the teacher and the student).We illustrate the workflow of MetaDistil in Figure 1. The teacher in MetaDistil is trainable, which enables the teacher to adjust to its student network and also improves its "teaching skills." Motivated by the idea of student-centered learning, we allow the teacher to adjust

[1] The code will be released upon acceptance.

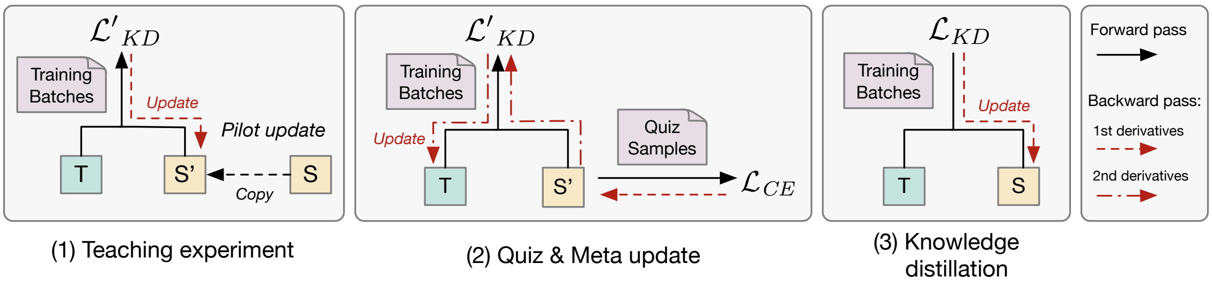

Figure 1: The workflow of MetaDistil. (1) We perform experimental knowledge distillation on a selection of training batches. Instead of updating the student S, we make a temporary copy S' and update S'. (2) We calculate a Cross-Entropy loss $\mathcal{L}_{CE}$ of S' on samples from a separate quiz set. We calculate the gradients of $\mathcal{L}_{CE}$ with respect to the parameters of T and update T by gradient descent. (3) We discard S' and use the updated T to perform actual knowledge distillation and update S.

its output based on the performance of the student model on a "quiz set," which is a separate reserved data split from the original training set. For each training step, we first copy the student S to S' and update S' by a common knowledge distillation loss. We call this process a "teaching experiment." In this way, we can obtain an experimental student S' that can be quizzed. Then, we sample from the quiz set, and calculate the loss of S' on these samples. We use this loss as a feedback signal to meta-update the teacher by calculating second derivatives and performing gradient descent (Finn et al., 2017). Finally, we discard the experimental subject S' and use the updated teacher to distill into the student S on the same training batches. The use of meta learning allows the teacher model to receive feedback from the student in a completely differentiable way. We provide a simple and intuitive approach to explicitly optimize the teacher using the student's quiz performance as a proxy.

To test the effectiveness of MetaDistil, we conduct extensive experiments on text and image classification tasks. MetaDistil outperforms knowledge distillation by a large margin, verifying the effectiveness and versatility of our method. Also, our method achieves state-of-the-art performance compressing BERT (Devlin et al., 2019) on the GLUE benchmark (Wang et al., 2019) and shows competitive results compressing ResNet (He et al., 2016) and VGG (Simonyan & Zisserman, 2015) on CIFAR-100 (Krizhevsky et al., 2009). Additionally, we design experiments to analyze and explain the improvement. Ablation studies show the effectiveness of our proposed pilot update and dynamic distillation. Also, compared to conventional KD, MetaDistil is more robust to different student capacity and hyperparameters, which is probably because of its ability to adjust its parameters.

2 Related Work

Knowledge Distillation Recently, many attempts have been made to accelerate large neural networks (Xu et al., 2020; Zhou et al., 2020, 2021). Knowledge distillation is a prominent method for training compact networks to achieve comparable performance to a deep network. Hinton et al. (2015) first introduced the idea of knowledge distillation to exploit the "dark knowledge" (i.e., soft label distribution) from a large teacher model as additional supervision for training a smaller student model. Since its introduction, several works (Romero et al., 2015; Zagoruyko & Komodakis, 2017; Tung & Mori, 2019; Park et al., 2019; Sun et al., 2019; Jiao et al., 2019) have investigated methods that align different latent representations between the student and teacher models for better knowledge transfer. In the context of knowledge distillation, MetaDistil shares some common ideas with the line of work that utilizes a sequence of intermediate teacher models to make the teacher network better adapt to the capacity of the student model throughout the training process, including teacher assistant knowledge distillation (TAKD) (Mirzadeh et al., 2020) and route constraint optimization (RCO) (Jin et al., 2019). However, the intermediate teachers are heuristically selected independently of the training process and the evolution of the teacher network is discrete. In contrast, MetaDistil employs meta learning to make the teacher model adapt to the current state of the student model and provide a continuously evolving meta-teacher that can better teach the student. Concurrently, Park et al. (2021) and Shi et al. (2021) propose to update the teacher model jointly with the student model with task specific objectives (e.g., cross-entropy loss) during the KD process and add constraints to keep student and teacher similar to each other. Their approaches makes the teacher model aware of the student model by constraining the teacher model's capacity. However, the teacher models in their methods are still not optimized for knowledge transfer. In addition, Zhang et al. (2018) introduced deep mutual learning where multiple models learn collaboratively and teach each other throughout the training process. While it is focused on a different setting where different models have approximately the same capacity and are learned from scratch, it also encourages the teacher model to behave similarly to the student model. Different from all aforementioned methods, MetaDistil employs meta learning to explicitly optimize the teacher model for better knowledge transfer ability, and leads to improved performance of the resulting student model.

Meta Learning The core idea of meta learning is "learning to learn," which means taking the optimization process of a learning algorithm into consideration when optimizing the learning algorithm itself. Meta learning typically involves a bi-level optimization process where the inner-learner provides feedback for optimization of the meta-learner. Successful applications of meta learning include learning better initialization (Finn et al., 2017), architecture search (Liu et al., 2019), learning to optimize the learning rate schedule (Baydin et al., 2018), and learning to optimize (Andrychowicz et al., 2016). These works typically aim to obtain an optimized meta-learner (i.e., the teacher model in MetaDistil), while the optimization of the inner-learner (i.e., the student model in MetaDistil), is mainly used to provide learning signal for the meta optimization process. This is different from the objective of knowledge distillation where an optimized student model is the goal. Recently, there have been a few works investigating using this bi-level optimization framework to obtain a better inner-learner. For example, meta pseudo labels (Pham et al., 2020) uses meta learning to optimize a pseudo label generator for better semi-supervised learning; meta back-translation (Pham et al., 2021) meta-trains a back-translation model to better train a machine translation model. These methods adapt the same bi-level optimization process as previous works where the goal is to obtain an optimized meta-learner. In these approaches, during each iteration, the meta-learner is optimized for the original inner-learner and then applied to the updated inner-learner in the next iteration. This leads to a mismatch between the meta-learner and the inner-learner, and is therefore suboptimal for learning a good inner-learner. In this paper, we introduce a pilot update mechanism, which is a simple and general method for this kind of problem, for the inner-learner to mitigate this issue and make the updated meta-learner better adapted to the inner-learner.

Meta Knowledge Distillation Recently, some works on KD take a meta approach. Pan et al. (2020) proposed a framework to train a meta-teacher across domains that can better fit new domains with meta-learning. Then, traditional KD is performed to transfer the knowledge from the meta-teacher to the student. Liu et al. (2020) proposed a self-distillation network and utilizes meta-learning to train a label-generator, which is a fusion of deep layers in the network, to generate more compatible soft targets for shallow layers. Different from the above, MetaDistil is a general knowledge distillation method that exploits meta-learning to allow the teacher to learn to teach dynamically. Instead of merely training a meta-teacher, our method uses meta-learning throughout the procedure of knowledge transfer, making the teacher model compatible for the student model for every training example during each training stage.

3 Knowledge Distillation with Meta Learning

An overview of MetaDistil is presented in Figure 1. MetaDistil includes two major components. First, the meta update enables the teacher model to receive the student model's feedback on the distillation process, allowing the teacher model to "learn to teach" and provide distillation signals that are more suitable for the student model's current capacity. The pilot update mechanism ensures a finer-grained match between the student model and the meta-updated teacher model.

3.1 Background

3.1.1 Knowledge Distillation

Knowledge distillation algorithms aim to exploit the hidden knowledge from a large teacher network, denoted as T, to guide the training of a shallow student network, denoted as S. To help transfer the knowledge from the teacher to the student, apart from the original task-specific objective (e.g., cross-entropy loss), a knowledge distillation objective which aligns the behavior of the student and the teacher is included to train the student network. Formally, given a labeled dataset $\mathcal{D}$ of N samples $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, we can write the loss function of the student network as follows,

$$\begin{aligned}\mathcal{L}_S(\mathcal{D}; \theta_S; \theta_T) = \frac{1}{N}\sum_{i=1}^{N}[\alpha\mathcal{L}_{\mathcal{T}}(y_i, S(x_i; \theta_S)) \\ + (1-\alpha)\mathcal{L}_{KD}(T(x_i; \theta_T), S(x_i; \theta_S))]\end{aligned} \tag{1}$$

where α is a hyper-parameter to control the relative importance of the two terms; θ_T and θ_S are the parameters of the teacher T and student S, respectively. $\mathcal{L}_{\mathcal{T}}$ refers to the task-specific loss and $\mathcal{L}_{KD}$ refers to the knowledge distillation loss which measures the similarity of the student and the teacher. Some popular similarity measurements include the KL divergence between the output probability distribution, the mean squared error between student and teacher logits, the similarity between the student and the teacher's attention distribution, etc. We do not specify the detailed form of the loss function because MetaDistil is a general framework that can be easily applied to various kinds of KD objectives as long as the objective is differentiable with respect to the teacher parameters.

3.1.2 Meta Learning

In meta learning algorithms that involve a bi-level optimization problem (Finn et al., 2017), there exists an inner-learner f_i and a meta-learner f_m. The inner-learner is trained to accomplish a task $\mathcal{T}$ or a distribution of tasks with help from the meta-learner. The training process of f_i on $\mathcal{T}$ with the help of f_m is typically called *inner-loop*, and we can denote $f_i'(f_m)$ as the updated inner-learner after the inner-loop. We can express f_i' as a function of f_m because learning f_i depends on f_m. In return, the meta-learner is optimized with a meta objective, which is generally the maximization of expected performance of the inner-learner after the inner-loop, i.e., $f_i'(f_m)$. This learning process is called a *meta-loop* and is often accomplished by gradient descent with derivatives of $\mathcal{L}(f_i'(f_m))$, the loss of updated inner-leaner on some held-out support set (i.e., the quiz set in our paper).

3.2 Methodology

3.2.1 Pilot Update

In the original formulation of meta learning (Finn et al., 2017), the purpose is to learn a good meta-learner f_m that can generalize to different inner-learners f_i for different tasks. In their approach, the meta-learner is optimized for the "original" inner-learner at the beginning of each iteration and the current batch of training data. The updated meta-learner is then applied to the updated inner-learner and a different batch of data in the next iteration. This behavior is reasonable if the purpose is to optimize the meta-learner. However, in MetaDistil, we only care about the performance of the only inner-learner, i.e., the student. In this case, this behavior leads to a mismatch between the meta-learner and the inner-learner, and is therefore suboptimal for learning a good inner-learner. Therefore, we need a way to align and synchronize the learning of the meta- and inner-learner, in order to allow an update step of the meta-learner to have an instant effect on the inner-learner. This instant reflection prevents the meta-learner from catastrophic forgetting (McCloskey & Cohen, 1989). To achieve this, we design a pilot update mechanism. For a batch of training data $\boldsymbol{x}$, we first make a temporary copy of the inner-learner f_i and update both the copy f_i' and the meta learner f_m on $\boldsymbol{x}$. Then, we discard f_i' and update f_i again with the updated f_m on the same data $\boldsymbol{x}$. This mechanism can apply the impact of data $\boldsymbol{x}$ to both f_m and f_i at the same time, thus aligns the training process. Pilot update is a general technique that can potentially be applied to any meta learning application that optimizes the inner-learner performance. We will describe how we apply this mechanism to MetaDistil shortly and empirically verify the effectiveness of pilot update in Section 4.2.

3.2.2 Learning to Teach

In MetaDistil, we would like to optimize the teacher model, which is fixed in traditional KD frameworks. Different from previous deep mutual learning (Zhang et al., 2018) methods that switch the role between the student and teacher network and train the original teacher model with soft labels generated by the student model or recent works (Shi et al., 2021; Park et al., 2021) that update the teacher model with a task-specific loss during the KD process, MetaDistil explicitly optimizes the teacher model in a "learning to teach" fashion, so that it can better transfer its knowledge to the student model. Concretely, the optimization objective of the teacher model in the MetaDistil framework is *the performance of the student model after distilling from the teacher model.* This "learning to teach" paradigm naturally fits the bi-level optimization framework in the meta learning literature.

In the MetaDistil framework, the student network θ_S is the inner-learner and the teacher network θ_T is the meta-learner. For each training step, we first copy the student model θ_S to an "experimental student" θ'_S. Then given a batch of training examples $\boldsymbol{x}$ and the learning rate λ, the experimental student is updated in the same way as conventional KD algorithms:

$$\theta'_S(\theta_T) = \theta_S - \lambda\nabla_{\theta_S}\mathcal{L}_S(x;\theta_S;\theta_T). \quad (2)$$

To simplify notation, we will consider one gradient update for the rest of this section, but using multiple gradient updates is a straightforward extension. We observe that the updated experimental student parameter θ'_S, as well as the student quiz loss $l_q = \mathcal{L}_\mathcal{T}(\boldsymbol{q}, \theta'_S(\theta_T))$ on a batch of quiz samples $\boldsymbol{q}$ sampled from a held-out quiz set $\mathcal{Q}$, is a function of the teacher parameter θ_T. Therefore, we can optimize l_q with respect to θ_T by a learning rate μ:

$$\theta_T \leftarrow \theta_T - \mu\nabla_{\theta_T}\mathcal{L}_\mathcal{T}\left(\boldsymbol{q}, \theta'_S(\theta_T)\right) \quad (3)$$

We evaluate the performance of the experimental student on a separate quiz set to prevent overfitting the validation set, which is preserved for model selection. After meta-updating the teacher model, we then update the "real" student model in the same way as described in Equation 2. Intuitively, optimizing the teacher network θ_T with Equation 3 is maximizing the expected performance of the student network after being taught by the teacher with the KD objective in the inner-loop. This meta-objective allows the teacher model to adjust its parameters to better transfer its knowledge to the student model. We apply the pilot update strategy described in Section 3.2.1 to better align the learning of the teacher and student. The complete algorithm is shown in Algorithm 1.

4 Experiments

4.1 Experimental Setup

We evaluate MetaDistil on two commonly used classification benchmarks for knowledge distillation in both Natural Language Processing and Computer Vision (see Appendix A).

Settings For NLP, we evaluate our proposed approach on the GLUE benchmark (Wang et al., 2019). Specifically, we test on MRPC (Dolan & Brockett, 2005), QQP[2] and STS-B (Conneau & Kiela, 2018) for Paraphrase Similarity Matching; SST-2 (Socher et al., 2013) for Sentiment Classification; MNLI (Williams et al., 2018), QNLI (Rajpurkar et al., 2016) and RTE (Wang et al., 2019) for the Natural Language Inference; CoLA (Warstadt et al., 2019) for Linguistic Acceptability. Following previous studies (Sun et al., 2019; Jiao et al., 2019; Xu et al., 2020), our goal is to distill BERT-Base (Devlin et al., 2019) into a 6-layer BERT with the hidden size of 768. The reported results are in the same format as on the GLUE leaderboard. For MNLI, we report the results on MNLI-m and MNLI-mm, respectively. For MRPC and QQP, we report both F1 and accuracy. For STS-B, we report Pearson and Spearman correlation. The metric for CoLA is Matthew's correlation. The other tasks use accuracy as the metric.

Following previous works (Sun et al., 2019; Turc et al., 2019; Xu et al., 2020), we evaluate MetaDistil in a *task-specific* setting where the teacher model is fine-tuned on a downstream task and the student model is trained on the task with the KD loss. We do not choose the pretraining distillation setting since it requires significant computational resources. We implement MetaDistil based on Hugging Face Transformers (Wolf et al., 2020).

Baselines For comparison, we report the results of vanilla KD and patient knowledge distillation (Sun et al., 2019). We also include the results of progressive module replacing (Xu et al., 2020), a state-of-the-art task-specific compression method for BERT which also uses a larger teacher model to improve smaller ones like knowledge distillation. In addition, according to Turc et al. (2019), the reported performance of current task-specific BERT compression methods is underestimated because the student model is not appropriately initialized. To ensure fair comparison, we re-run task-specific baselines with student models initialized by a pretrained 6-layer BERT model and report our results in addition to the official numbers in the original papers. We also com-

[2] https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs

Algorithm 1 Knowledge Distillation with Meta Learning (MetaDistil)

Require: student θ_S, teacher θ_T, train set $\mathcal{D}$, quiz set $\mathcal{Q}$
Require: λ, μ: learning rate for the student and the teacher

while not done **do**
  Sample batch of training data $\boldsymbol{x} \sim \mathcal{D}$
  Copy student parameter θ_S to student θ'_S
  Update θ'_S with $\boldsymbol{x}$ and θ_T: $\theta'_S \leftarrow \theta'_S - \lambda \nabla_{\theta'_S} \mathcal{L}_S(x; \theta_S; \theta_T)$
  Sample a batch of quiz data $\boldsymbol{q} \sim \mathcal{Q}$
  Update θ_T with $\boldsymbol{q}$ and θ'_S: $\theta_T \leftarrow \theta_T - \mu \nabla_{\theta_T} \mathcal{L}_T(\boldsymbol{q}, \theta'_S(\theta_T))$
  Update original θ_S with $\boldsymbol{x}$ and the updated θ_T: $\theta_S \leftarrow \theta_S - \lambda \nabla_{\theta_S} \mathcal{L}_S(x; \theta_S; \theta_T)$
end while

pare against deep mutual learning (DML) (Zhang et al., 2018), teacher assistant knowledge distillation (TAKD) (Mirzadeh et al., 2020), route constraint optimization (RCO) (Jin et al., 2019), and proximal knowledge teaching (ProKT) (Shi et al., 2021), where the teacher network is not fixed. For reference, we also present results of pretraining distilled models including DistilBERT (Sanh et al., 2019), TinyBERT (Jiao et al., 2019), MiniLM v1 and v2 (Wang et al., 2020b,a). Note that among these baselines, PKD (Sun et al., 2019) and Theseus (Xu et al., 2020) exploit intermediate features while TinyBERT and the MiniLM family use both intermediate and Transformer-specific features. In contrast, MetaDistil uses none of these but the vanilla KD loss (Equation 1).

Training Details For training hyperparameters, we fix the maximum sequence length to 128 and the temperature to 2 for all tasks. For our method and all baselines (except those with officially reported numbers), we perform grid search over the sets of the student learning rate λ from {1e-5, 2e-5, 3e-5}, the teacher learning rate μ from {2e-6, 5e-6, 1e-5}, the batch size from {32, 64}, the weight of KD loss from {0.4, 0.5, 0.6}. We randomly split the original training set to a new training set and the quiz set by $9:1$. For RCO, we select four unconverged teacher checkpoints as the intermediate training targets. For TAKD, we use KD to train a teacher assistant model with 10 Transformer layers.

4.2 Experimental Results

We report the experimental results on both the development set and test set of the eight GLUE tasks (Wang et al., 2019) in Table 1. MetaDistil achieves state-of-the-art performance under the task-specific setting and outperforms all KD baselines. Notably, without using any intermediate or model-specific features in the loss function, MetaDistil outperforms methods with carefully designed features, e.g., PKD and TinyBERT (without data augmentation). Compared with other methods with a trainable teacher (Zhang et al., 2018; Mirzadeh et al., 2020; Jin et al., 2019; Shi et al., 2021), our method still demonstrates superior performance. As we analyze, with the help of meta learning, MetaDistil is able to directly optimize the teacher's teaching ability thus yielding a further improvement in terms of student accuracy. Also, we observe a performance drop by replacing pilot update with a normal update. This ablation study verifies the effectiveness of our proposed pilot update mechanism. Moreover, MetaDistil achieves very competitive results on image classification as well, as described in Section A.2.

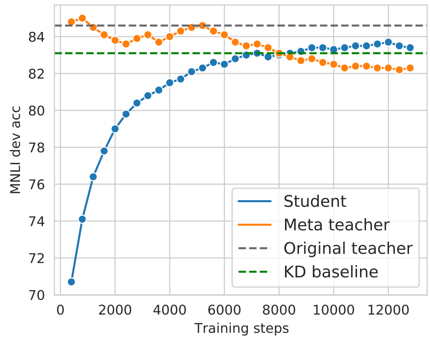

Figure 2: Learning dynamics of the student and teacher in MetaDistil on the development set of MNLI.

5 Analysis

5.1 Why Does MetaDistil Work?

We investigate why MetaDistil works on the development sets of MNLI, SST, and MRPC, which are important tasks in GLUE that have a large, medium, and small training set, respectively.

We illustrate the validation accuracy curves of the meta teacher and student models with training steps in Figure 2, and compare them to the student performance in conventional KD. We see the meta teacher maintains high accuracy in the first 5,000 steps and then begins to slowly degrade. Starting

Method	#Param.	Speed-up	CoLA (8.5K)	MNLI (393K)	MRPC (3.7K)	QNLI (105K)	QQP (364K)	RTE (2.5K)	SST-2 (67K)	STS-B (5.7K)
Dev Set										
BERT-Base (teacher) (Devlin et al., 2019)	110M	1.00×	58.9	84.6/84.9	91.6/87.6	91.2	88.5/91.4	71.4	93.0	90.2/89.8
BERT-6L (student) (Turc et al., 2019)	66M	1.94×	53.5	81.1/81.7	89.2/84.4	88.6	86.9/90.4	67.9	91.1	88.1/87.9
Pretraining Distillation										
TinyBERT‡ (Jiao et al., 2019)	66M	1.94×	54.0	84.5/84.5	90.6/86.3	91.1	88.0/91.1	73.4	93.0	90.1/89.6
MiniLM (Wang et al., 2020b)	66M	1.94×	49.2	84.0/ -	88.4/ -	91.0	- /91.0	71.5	92.0	-
MiniLM v2 (Wang et al., 2020a)	66M	1.94×	52.5	84.2/ -	88.9/ -	90.8	- /91.1	72.1	92.4	-
Task-specific Distillation										
KD† (Hinton et al., 2015)	66M	1.94×	54.1	82.6/83.2	89.6/85.2	89.2	87.3/90.9	67.7	91.2	88.6/88.2
PKD† (Sun et al., 2019)	66M	1.94×	54.5	82.7/83.3	89.4/84.7	89.5	87.8/90.9	67.6	91.3	88.6/88.1
TinyBERT w/o DA†	66M	1.94×	52.4	83.6/83.8	90.5/86.5	89.8	87.6/90.6	67.7	91.9	89.2/88.7
RCO† (Jin et al., 2019)	66M	1.94×	53.6	82.4/82.9	89.5/85.1	89.7	87.4/90.6	67.6	91.4	88.7/88.3
TAKD† (Mirzadeh et al., 2020)	66M	1.94×	53.8	82.5/83.0	89.6/85.0	89.6	87.5/90.7	68.5	91.4	88.2/88.0
DML† (Zhang et al., 2018)	66M	1.94×	53.7	82.4/82.9	89.6/85.1	89.6	87.4/90.3	68.4	91.5	88.4/88.1
ProKT† (Shi et al., 2021)	66M	1.94×	54.3	82.8/83.2	90.7/86.3	89.7	87.9/90.9	68.4	91.3	88.9/88.6
MetaDistil *(ours)*	66M	1.94×	**58.6**	**83.5/83.8**	**91.1/86.8**	**90.4**	**88.1/91.0**	**69.4**	**92.3**	**89.4/89.1**
w/o pilot update	66M	1.94×	56.3	83.0/83.4	90.6/86.6	89.9	88.0/88.5	67.7	92.0	89.2/89.0
Test Set										
BERT-Base (teacher) (Devlin et al., 2019)	110M	1.00×	52.1	84.6/83.4	88.9/84.8	90.5	71.2/89.2	66.4	93.5	87.1/85.8
Pretraining Distillation										
DistilBERT (Sanh et al., 2019)	66M	1.94×	45.8	81.6/81.3	87.6/83.1	88.8	69.6/88.2	54.1	92.3	71.0/71.0
TinyBERT‡ (Jiao et al., 2019)	66M	1.94×	51.1	84.3/83.4	88.8/84.5	91.6	70.5/88.3	70.4	92.6	86.2/84.8
Task-specific Distillation										
KD (Turc et al., 2019)	66M	1.94×	-	82.8/82.2	86.8/81.7	88.9	70.4/88.9	65.3	91.8	-
PKD (Sun et al., 2019)	66M	1.94×	43.5	81.5/81.0	85.0/79.9	89.0	70.7/88.9	65.5	92.0	83.4/81.6
BERT-of-Theseus (Xu et al., 2020)	66M	1.94×	47.8	82.4/82.1	87.6/83.2	89.6	**71.6/89.3**	66.2	92.2	85.6/84.1
ProKT (Shi et al., 2021)	66M	1.94×	-	82.9/82.2	87.0/82.3	89.7	70.9/88.9	-	93.3	-
DML† (Zhang et al., 2018)	66M	1.94×	48.5	82.6/81.6	86.5/81.2	89.5	70.7/88.7	66.3	92.7	85.5/84.0
RCO† (Jin et al., 2019)	66M	1.94×	48.2	82.3/81.2	86.8/81.4	89.3	70.4/88.7	66.5	92.6	85.3/84.1
TAKD† (Mirzadeh et al., 2020)	66M	1.94×	48.4	82.4/81.7	86.5/81.3	89.4	70.6/88.8	66.8	92.9	85.4/84.1
MetaDistil *(ours)*	66M	1.94×	**50.7**	**83.8/83.2**	**88.7/84.7**	**90.2**	71.1/88.9	**67.2**	**93.5**	**86.1/85.0**
w/o pilot update	66M	1.94×	49.1	83.3/82.8	88.2/84.1	89.9	71.0/88.7	66.6	93.5	85.9/84.6

Table 1: Experimental results on the development set and the test set of GLUE. Numbers under each dataset indicate the number of training samples. All student models listed below have the same architecture of 66M parameters, 6 Transformer layers and 1.94× speed-up. The test results are from the official test server of GLUE. The best results for the task-specific setting are marked with **boldface**. †Results reported by us. The student is initialized with a 6-layer pretrained BERT (Turc et al., 2019) thus has a *better* performance than the original implementation. ‡TinyBERT has data augmentation. We re-run TinyBERT without data augmentation and denote it as *TinyBERT w/o DA*. All results reported by us are average performance of 3 runs with different random seeds.

from step 8,000, the teacher model underperforms the student while the student's accuracy keeps increasing. This verifies our assumption that a model with the best accuracy is not necessarily the optimal teacher. Also, MetaDistil is not naively optimizing the teacher's accuracy but its "teaching skills." This phenomenon suggests that beyond high accuracy, there could be more important properties of a good teacher that warrant further investigation.

In addition, we investigate the effect of meta-update for each iteration. We inspect (1) the validation loss of S' after the teaching experiment and that of S after the real distillation update, and (2) the KD loss, which describes the discrepancy between student and teacher, before and after the teacher update. We find that for 87% of updates, the student model's validation loss after real update (Line 7 in Algorithm 1) is smaller than that after the teaching experiment (Line 4 in Algorithm 1), which would be the update to the student S in the variant without pilot update. This confirms the effectiveness of the pilot update mechanism on better matching the student and teacher model. Moreover, we find that in 91% of the first half of the updates, the teacher becomes more similar to the student after the meta-update, which indicates that the teacher is learning to adapt to a low-performance student (like an elementary school teacher). However, in the second half of MetaDistil, this percentage drops to 63%. We suspect this is because in the later training stages, the teacher needs to actively evolve itself beyond the student to guide the student towards

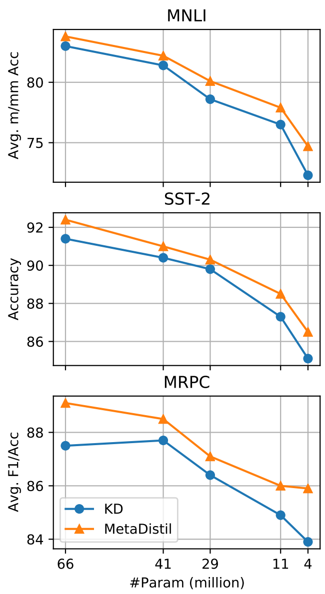

Figure 3: Results with different student architectures.

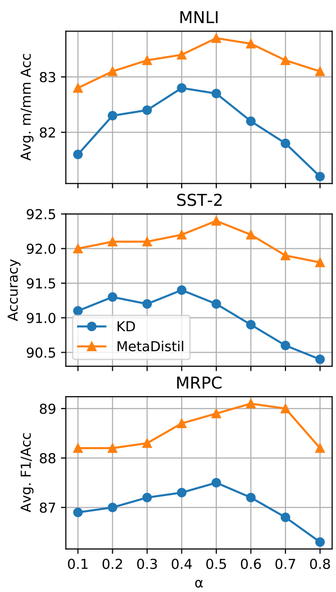

Figure 4: Results with different loss weight α.

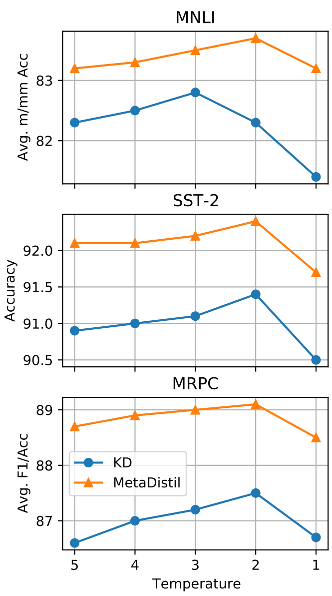

Figure 5: Results with different temperature.

further improvement (like a university professor). Finally, we try to apply a meta-learned teacher to a conventional static distillation and also to an unfamiliar student. We describe the results in details in Section A.3.

5.2 Can the Meta Teacher Always Learn to Teach?

A motivation of MetaDistil is to enable the teacher to dynamically adjust its knowledge transfer in an optimal way. Similar to Adam (Kingma & Ba, 2015) vs. SGD (Sinha & Griscik, 1971; Kiefer et al., 1952) for optimization, with the ability of dynamic adjusting, it is natural to expect MetaDistil to be more insensitive and robust to changes of the settings. Here, we evaluate the performance of MetaDistil with students of various capability, and a wide variety of hyperparameters, including *loss weight* and *temperature*.

Student Capability To investigate the performance of MetaDistil under different student capacity, we experiment to distill BERT-Base into BERT-6L, Medium, Small, Mini and Tiny (Turc et al., 2019) with conventional KD and MetaDistil. We plot the performance with the student's parameter number in Figure 3. We can see MetaDistil outperforms conventional KD on every student and has a more gradual performance curve.

Loss Weight In KD, tuning the loss weight is non-trivial and often requires hyperparameter search. To test the robustness of MetaDistil under different loss weights, we run experiments with different α (Equation 1). As shown in Figure 4, MetaDistil consistently outperforms conventional KD and is less sensitive to different α.

Temperature Temperature is a re-scaling trick introduced in Hinton et al. (2015). We try different temperatures and illustrate the performance of KD and MetaDistil in Figure 5. MetaDistil shows better performance and robustness compared to KD.

6 Discussion

In this paper, we present MetaDistil, a knowledge distillation algorithm powered by meta learning that explicitly optimizes the teacher network to better transfer its knowledge to the student network. The extensive experiments verify the effectiveness and robustness of MetaDistil. One limitation is the training of MetaDistil is slower than conventional KD since it calculates second derivatives and makes additional teacher updates. However, since the goal is to produce an efficient student model for production, the computational cost for the distillation process is not a major concern. For future work, we would like to further investigate the teaching skills learned by the meta teacher from a theoretical perspective and use the insights to improve conventional knowledge distillation.

References

Sungsoo Ahn, Shell Xu Hu, Andreas C. Damianou, Neil D. Lawrence, and Zhenwen Dai. Variational information distillation for knowledge transfer. In *CVPR*, 2019.

Marcin Andrychowicz, Misha Denil, Sergio Gomez Colmenarejo, Matthew W. Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett (eds.), *NeurIPS*, 2016.

Atilim Gunes Baydin, Robert Cornish, David Martínez-Rubio, Mark Schmidt, and Frank Wood. Online learning rate adaptation with hypergradient descent. In *ICLR*, 2018.

Alexis Conneau and Douwe Kiela. Senteval: An evaluation toolkit for universal sentence representations. In *LREC*, 2018.

Jeffrey Cornelius-White. Learner-centered teacher-student relationships are effective: A meta-analysis. *Review of educational research*, 77(1):113–143, 2007.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

William B. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *IWP@IJCNLP*, 2005.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh (eds.), *ICML*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Byeongho Heo, Minsik Lee, Sangdoo Yun, and Jin Young Choi. Knowledge transfer via distillation of activation boundaries formed by hidden neurons. In *AAAI*, 2019.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*, 2019.

Xiao Jin, Baoyun Peng, Yichao Wu, Yu Liu, Jiaheng Liu, Ding Liang, Junjie Yan, and Xiaolin Hu. Knowledge distillation via route constrained optimization. In *ICCV*, 2019.

Jack Kiefer, Jacob Wolfowitz, et al. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3):462–466, 1952.

Jangho Kim, Seonguk Park, and Nojun Kwak. Paraphrasing complex network: Network compression via factor transfer. In *NeurIPS*, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

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

Benlin Liu, Yongming Rao, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Metadistiller: Network self-boosting via meta-learned top-down distillation. In *ECCV*, 2020.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. In *ICLR*, 2019.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Seyed-Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. In *AAAI*, 2020.

Haojie Pan, Chengyu Wang, Minghui Qiu, Yichang Zhang, Yaliang Li, and Jun Huang. Meta-kd: A meta knowledge distillation framework for language model compression across domains. *arXiv preprint arXiv:2012.01266*, 2020.

Dae Young Park, Moon-Hyun Cha, Changwook Jeong, Daesin Kim, and Bohyung Han. Learning student-friendly teacher networks for knowledge distillation. *arXiv preprint arXiv:2102.07650*, 2021.

Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *CVPR*, 2019.

Nikolaos Passalis and Anastasios Tefas. Learning deep representations with probabilistic knowledge transfer. In *ECCV*, 2018.

Baoyun Peng, Xiao Jin, Dongsheng Li, Shunfeng Zhou, Yichao Wu, Jiaheng Liu, Zhaoning Zhang, and Yu Liu. Correlation congruence for knowledge distillation. In *ICCV*, 2019.

Hieu Pham, Zihang Dai, Qizhe Xie, Minh-Thang Luong, and Quoc V Le. Meta pseudo labels. *arXiv preprint arXiv:2003.10580*, 2020.

Hieu Pham, Xinyi Wang, Yiming Yang, and Graham Neubig. Meta back-translation. *arXiv preprint arXiv:2102.07847*, 2021.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*, 2016.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In Yoshua Bengio and Yann LeCun (eds.), *ICLR*, 2015.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Wenxian Shi, Yuxuan Song, Hao Zhou, Bohan Li, and Lei Li. Learning from deep model via exploring local targets, 2021. URL https://openreview.net/forum?id=5slGDu_bVc6.

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

Naresh K. Sinha and Michael P. Griscik. A stochastic approximation method. *IEEE Trans. Syst. Man Cybern.*, 1(4):338–344, 1971.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, 2013.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for BERT model compression. In *EMNLP-IJCNLP*, 2019.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. In *ICLR*, 2020.

Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation. In *ICCV*, 2019.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Well-read students learn better: The impact of student initialization on knowledge distillation. *arXiv preprint arXiv:1908.08962*, 13, 2019.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*, 2019.

Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. Minilmv2: Multi-head self-attention relation distillation for compressing pretrained transformers. *arXiv preprint arXiv:2012.15828*, 2020a.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *NeurIPS*, 2020b.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. Neural network acceptability judgments. *TACL*, 2019.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL-HLT*, 2018.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *EMNLP (Demos)*, pp. 38–45. Association for Computational Linguistics, 2020.

Gloria Brown Wright. Student-centered learning in higher education. *International Journal of Teaching and Learning in Higher Education*, 23(1):92–97, 2011.

Canwen Xu, Wangchunshu Zhou, Tao Ge, Furu Wei, and Ming Zhou. Bert-of-theseus: Compressing BERT by progressive module replacing. In *EMNLP*, 2020.

Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *ICLR*, 2017.

Ying Zhang, Tao Xiang, Timothy M. Hospedales, and Huchuan Lu. Deep mutual learning. In *CVPR*, 2018.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian J. McAuley, Ke Xu, and Furu Wei. BERT loses patience: Fast and robust inference with early exit. In *NeurIPS*, 2020.

Wangchunshu Zhou, Tao Ge, Ke Xu, and Furu Wei. Improving sequence-to-sequence pre-training via sequence span rewriting. *arXiv preprint arXiv:2101.00416*, 2021.

A MetaDistil for Image Classification

In addition to BERT compression, we also provide results on image classification. Also, we conduct experiments of static teaching and cross teaching, to further verify the effectiveness of MetaDistil of adapting to different students.

A.1 Experimental Settings

For CV, following the settings in Tian et al. (2020), we experiment with the image classification task on CIFAR-100 (Krizhevsky et al., 2009) with student-teacher combinations of different capacity and architectures, including ResNet (He et al., 2016) and VGG (Simonyan & Zisserman, 2015). Additionally, we run a distillation experiment between different architectures (a ResNet teacher to a VGG student). We report the top-1 test accuracy of the compressed student networks. We inherit all hyperparameters from Tian et al. (2020) except for the teacher learning rate, which is grid searched from {1e-4, 2e-4, 3e-4}. We randomly split the original training set to a new training set and the quiz set by $9 : 1$. We compare our results with a state-of-the-art distillation method, CRD (Tian et al., 2020) and other commonly used knowledge distillation methods (Hinton et al., 2015; Romero et al., 2015; Zagoruyko & Komodakis, 2017; Tung & Mori, 2019; Peng et al., 2019; Ahn et al., 2019; Park et al., 2019; Passalis & Tefas, 2018; Heo et al., 2019; Kim et al., 2018) including ProKT (Shi et al., 2021) which has a trainable teacher.

A.2 Experimental Results

We show the experimental results of MetaDistil distilling ResNet (He et al., 2016) and VGG (Simonyan & Zisserman, 2015) with five different teacher-student pairs. MetaDistil achieves comparable performance to CRD (Tian et al., 2020), the current state-of-the-art distillation method on image classification while outperforming all other baselines with complex features and loss functions. Notably, CRD introduces additional negative sampling and contrastive training while our method achieves comparable performance without using these tricks. Additionally, we observe a substantial performance drop without pilot update, again verifying the importance of this mechanism.

A.3 Static Teaching and Cross Teaching

In MetaDistil, the student is trained in a dynamic manner. To investigate the effect of such a dynamic distillation process, we attempt to use the teacher at the end of MetaDistil training to perform a static conventional KD, to verify the effectiveness of our dynamic distillation strategy. As shown in Table 3, on both experiments, dynamic MetaDistil outperforms conventional KD and static distillation with the teacher at the end of MetaDistil training.

As mentioned in Section 3.2, a meta teacher is optimized to transfer its knowledge to a specific student network. To justify this motivation, we conduct experiments using a teacher optimized for the ResNet-32 student to statically distill to the ResNet-20 student, and also in reverse. As shown in Table 3, the cross-taught students underperform the static students taught by their own teachers by 0.27 and 0.12 for ResNet-32 and ResNet-20, respectively. This confirms our motivation that the meta teacher in MetaDistil can adjust itself according to its student.

Teacher **Student**	ResNet-56 ResNet-20	ResNet-110 ResNet-20	ResNet-110 ResNet-32	VGG-13 VGG-8	ResNet-50* VGG-8
Teacher	72.34	74.31	74.31	74.64	79.34
Student	69.06	69.06	71.14	70.36	70.36
KD (2015)	70.66	70.67	73.08	72.98	73.81
FitNet (2015)	69.21	68.99	71.06	71.02	70.69
AT (2017)	70.55	70.22	72.31	71.43	71.84
SP (2019)	69.67	70.04	72.69	72.68	73.34
CC (2019)	69.63	69.48	71.48	70.71	70.25
VID (2019)	70.38	70.16	72.61	71.23	70.30
RKD (2019)	69.61	69.25	71.82	71.48	71.50
PKT (2018)	70.34	70.25	72.61	72.88	73.01
AB (2019)	69.47	69.53	70.98	70.94	70.65
FT (2018)	69.84	70.22	72.37	70.58	70.29
ProKT (2021)	70.98	70.74	72.95	73.03	73.90
CRD (2020)	<u>71.16</u>	**71.46**	**73.48**	**73.94**	<u>74.30</u>
MetaDistil	**71.25**	<u>71.40</u>	<u>73.35</u>	<u>73.65</u>	**74.42**
w/o pilot update	71.02	70.96	73.31	73.48	74.05

Table 2: Experimental results on the test set of CIFAR-100. The best and second best results are marked with **boldface** and <u>underline</u>, respectively. All baseline results except ProKT are reported in Tian et al. (2020). *ResNet for ImageNet. Other ResNets are ResNet for CIFAR (He et al., 2016).

Teacher	**Student**	**Acc@1**
KD (ResNet-110)	ResNet-32 (static)	73.08
	ResNet-20 (static)	70.67
MetaDistil (ResNet-110→ResNet-32)	ResNet-32 (dynamic)	73.35
	ResNet-32 (static)	73.16
	ResNet-20 (static, cross)	70.82
MetaDistil (ResNet-110→ResNet-20)	ResNet-20 (dynamic)	71.40
	ResNet-20 (static)	70.94
	ResNet-32 (static, cross)	72.89

Table 3: Experimental results of static teaching and cross teaching.