A Good Learner can Teach Better: TEACHER-STUDENT COLLABORATIVE KNOWLEDGE DISTILLATION

Ayan Sengupta\textsuperscript{1}, Shantanu Dixit\textsuperscript{2}, Md Shad Akhtar\textsuperscript{2}, Tanmoy Chakraborty\textsuperscript{1}
\textsuperscript{1}Indian Institute of Technology Delhi, India
\textsuperscript{2}Indraprastha Institute of Information Technology Delhi, India
ayan.sengupta@ee.iitd.ac.in, \{shantanu2018, shad.akhtar\}@iiitd.ac.in, tanchak@iitd.ac.in

ABSTRACT

Knowledge distillation (KD) is a technique used to transfer knowledge from a larger “teacher” model into a smaller “student” model. Recent advancements in meta-learning-based knowledge distillation (MetaKD) emphasize that the fine-tuning of teacher models should be aware of the student’s need to achieve better knowledge distillation. However, existing MetaKD methods often lack incentives for the teacher model to improve itself. In this study, we introduce MPDistil, a meta-policy distillation technique, that utilizes novel optimization strategies to foster both collaboration and competition during the fine-tuning of the teacher model in the meta-learning step. Additionally, we propose a curriculum learning framework for the student model in a competitive setup, in which the student model aims to outperform the teacher model by self-training on various tasks. Exhaustive experiments on SuperGLUE and GLUE benchmarks demonstrate the efficacy of MPDistil compared to 20 conventional KD and advanced MetaKD baselines, showing significant performance enhancements in the student model – e.g., a distilled 6-layer BERT model outperforms a 12-layer BERT model on five out of six SuperGLUE tasks. Furthermore, MPDistil, while applied to a large language teacher model (DeBERTa-v2-xxlarge), significantly narrows the performance gap of its smaller student counterpart (DeBERTa-12) by just 4.6% on SuperGLUE. We further demonstrate how higher rewards and customized training curricula strengthen the student model and enhance generalizability.

1 INTRODUCTION

Large language models (LLMs) like GPT-3 (Brown et al., 2020), PaLM (Chowdhery et al., 2022), and LLaMA (Touvron et al., 2023) have demonstrated remarkable performance in a wide range of natural language tasks, showcasing their abilities in tasks requiring minimal, zero, or context-based prior knowledge (Kossen et al., 2023). These models underwent extensive pre-training on massive datasets to achieve such impressive capabilities, incurring substantial computational costs, as highlighted by Zhu et al. (2023). The scaling laws associated with language models (Kaplan et al., 2020) indicate that achieving superior predictive performance comes at the cost of a large number of model parameters, extensive training data, and substantial computational power. However, these costs typically do not account for inference expenses, which must be handled by resource-constrained devices when deploying these models at scale. One approach to address this challenge is “model compression” (Deng et al., 2020), a technique designed to reduce computational requirements and storage while preserving the original models’ performance.

Knowledge distillation (KD) (Hinton et al., 2015) is a widely adopted model compression method that entails transferring knowledge from a larger and more complex model, referred to as the “teacher model”, to a smaller and simpler model known as the “student model”. In traditional white-box knowledge distillation, the primary focus is on minimizing the “teacher-student margin” by narrowing the gap between the teacher’s distribution, \(t_\theta(y|x)\), and the student’s distribution, \(s_\theta(y|x)\). More advanced knowledge distillation techniques (Sun et al., 2019; Jiao et al., 2019) go further by aligning the intermediate feature maps between the teacher and student models. This alignment helps the student replicate the behaviors demonstrated by the teacher model. However, most of these techniques operate in a unidirectional manner, in which the teacher model remains fixed after pre-training or
fine-tuning, while the student model is trained to minimize the teacher-student margin. Zhou et al. (2021) pointed out several limitations of this approach. Existing approaches do not optimize the teacher explicitly for the distillation task or consider the student model’s capacity during training. This differs from the real-life teacher-student dynamic, in which a teacher continuously enhances her knowledge and teaching skills through repeated student interactions. To address this gap, Zhou et al. (2021) introduced MetaDistil, a meta-learning-based knowledge distillation framework that aims to overcome this limitation by considering the student’s predictive performance while fine-tuning the teacher model. In MetaDistil, the teacher model undergoes training on a separate quiz dataset to minimize the knowledge distillation loss between the teacher and student models along with the teacher’s task-specific loss. Although MetaDistil addresses several shortcomings of conventional KD methods, it is still far from imitating real-world teacher-student interactions. In a real-world educational context, teachers may not solely focus on narrowing the teacher-student knowledge gap to improve teaching; instead, they aim to maximize the joint performance of students and teachers. Furthermore, in a typical classroom setting, students study various subjects and follow a curriculum to optimize their learning across all subjects. For example, a student might aim to improve her understanding of Physics by studying selected concepts from Mathematics. However, existing knowledge distillation methods tend to distill knowledge for different tasks in isolation, which means they fail to capture the commonalities among tasks. Consequently, the distilled models struggle to generalize effectively to new tasks with limited training data. Another significant challenge with the sole optimization of the teacher-student margin is that it positions the teacher as the “sole role model” for the student without encouraging the student to surpass the teacher’s performance.

To address these limitations, we introduce MPDistil, a meta-policy knowledge distillation framework that employs a collaborative learning approach within the context of meta-knowledge distillation. Through empirical analyses, we underscore that optimizing a shared utility function can enhance the teacher’s predictive capabilities and ability to impart knowledge to the student model. Furthermore, we establish a meta-reinforcement learning-based “curriculum learning” paradigm in which the student model undergoes fine-tuning on a curriculum – a sequence of tasks. The aim is to enhance the student model’s generalization skills, enabling it to surpass the teacher model. This competitive approach empowers the student to outperform the teacher. We assess the effectiveness of MPDistil using two natural language understanding benchmarks, SuperGLUE (Wang et al., 2019) and GLUE (Wang et al., 2018), encompassing 15 NLU tasks. Following the experimental setup adopted by contemporary KD-related studies, we highlight the effectiveness of MPDistil with BERT-base (Devlin et al., 2018) as the teacher model and BERT 6L (with six layers) as the student model. On the SuperGLUE benchmark, the meta update enhances the BERT teacher’s performance with a maximum margin of +3%, resulting in a notable improvement in student performance by +5.9%. Surprisingly, the distilled student surpasses the teacher model on five of six SuperGLUE tasks within this competitive framework, with a maximum advantage of +7%. In contrast, the most competitive baseline, MetaDistil, exhibits an average deficit of −1.1%, indicating that the distilled student falls short of the teacher model’s performance. We also extend our evaluation to a large language model, DeBERTa-v2-xxlarge (He et al., 2020), as the teacher model and its smaller variant, DeBERTa-12, as the student model. Ours is the first large-scale study highlighting the effectiveness of distilling knowledge from large language models (>1B parameters) to smaller models. MPDistil, when applied to DeBERTa, improves the student model’s performance with a maximum margin of +2.2%, reducing the student-teacher margin to −4.6% (a negative student-teacher margin indicates that student underperforms the teacher), which is significantly higher than the other distillation methods (−7.9%).

2 Related Work

Knowledge distillation. One of the earliest investigations on knowledge distillation by Hinton et al. (2015) introduced a technique in which the student model is trained to minimize the task-specific loss while also enhancing the similarity between the outputs of the teacher and student models. This enhancement is achieved by minimizing the difference between softened class probabilities generated by both the teacher and student model using temperature scaling. Sanh et al. (2019) distilled a more compact version of BERT-base by halving the number of layers and removing the pooller output and token type embeddings. In contrast to learning solely from the teacher's

1 Source code of MPDistil can be found at [https://github.com/notmymame16/MPDistil](https://github.com/notmymame16/MPDistil)
last layer, Sun et al. (2019) introduced a “patient learner” approach. The student learns from the last $k$ layers by incorporating a Mean Squared Error (MSE) loss between the teacher and student layer representations, leveraging the internal representations of the teacher model. TinyBERT (Jiao et al., 2019) also utilises the internal representations of BERT to distil knowledge. It achieves this by distilling the transformer, embedding, and classification layers. Yang et al. (2020) and Wu & Chiu (2020) introduced multiple teacher-based knowledge distillation – each teacher is assigned predetermined weights, and their weighted average probability distributions are used to train the student model. In a similar attempt, Yuan et al. (2021) proposed a reinforcement learning-based technique in which different weights are assigned to various teacher models during distillation using a policy function.

**Meta knowledge distillation.** Meta-learning-based knowledge distillation techniques utilize a meta-learning loop to enhance the teacher model. Established MetaKD approaches (Pan et al., 2020, Liu et al., 2022b) employ meta-teachers to adapt them to new domains and tasks. Liu et al. (2022b) argued that using fixed temperature can result in suboptimal knowledge transfer from teacher to student. They aimed to meta-learn the student and teacher’s temperature along with the distillation function. In contrast, Zhou et al. (2021) introduced a generalized framework, MetaDistil, focusing on teaching teachers how to facilitate knowledge distillation. Note that these techniques entail fine-tuning a meta-teacher, which is computationally equivalent to the teacher model. Fine-tuning a large pre-trained teacher model can be prohibitively expensive. Furthermore, the meta-learning objective optimized by the meta-teacher primarily aims to narrow the gap between the teacher and student models without encouraging them to surpass each other.

**Curriculum based knowledge distillation.** Curriculum learning (Bengio et al., 2009) involves increasing the level of training samples progressively during training. Xiang et al. (2020) designed curricula based on the confidence score of the mixture of expert models, which was later used to train a unified student model. Li et al. (2023) proposed learning a curriculum from easy to hard levels through a learnable temperature optimized through a reversed gradient. It involved increasing the distillation loss through a cosine schedule, leading to an increased difficulty for student learning.

Our work introduces collaborative and competitive utility optimization within the meta-knowledge-distillation paradigm and establishes a competitive reward system for the student using task based curricula. This competitive framework motivates both teacher and student models to outperform each other. The joint utility can also be optimized using a simpler feed-forward meta-teacher, eliminating the need to fine-tune the teacher backbone. This enhances the generality of our proposed method, making it suitable for distilling large pre-trained models and making our method more computationally efficient.

3 Proposed Methodology

Our proposed meta-policy knowledge distillation method, MPDistil, aims to distil knowledge from an arbitrary teacher model, parameterized by $\theta_T$ to an arbitrary student model, parameterized by $\theta_S$. Although the teacher and student models can belong to any family (e.g., encoder, encoder-decoder or decoder), we typically assume that they belong to the same architectural family. Additionally, we assume that both the teacher and student models can be extended to a multi-task framework, wherein the models can be optimized on any task $T \in T$. MPDistil consists of four steps – (i) fine-tuning the teacher model, (ii) distilling the teacher model to the student model, (iii) meta-teacher learning, and (iv) student curriculum learning. We illustrate these steps in Figure 1.

**Fine-tuning teacher model.** For any given task $T \in T$, we use the task-specific loss function $L_T$ to optimize and fine-tune the teacher model. For classification tasks, cross-entropy is used as the task-specific loss, whereas for regression tasks, MSE is used to calculate the loss. Precisely, for a given training data batch of size $N$, $D_{train} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, we calculate $L_{teacher} = \frac{1}{N} \sum_{i=1}^{N} L_T(y_i, \hat{y}_{i,T})$, with $\hat{y}_{i,T} = T(x_i; \theta_T)$, the output generated by the teacher model $T$ with parameters $\theta_T$. The teacher model parameters are then updated with gradients computed on $L_T$ with respect to $\theta_T$. Although we highlight the teacher fine-tuning with vanilla training procedure, for large language models, it can be replaced with parameter-efficient fine-tuning or even few-shot or zero-shot fine-tuning.
Figure 1: A schematic diagram of our proposed MPDistil framework. It consists of four steps – (i) teacher fine-tuning step, where the original teacher model is fine-tuned on task $T$, (ii) student distillation step, where the student model is fine-tuned with task-specific and knowledge distillation loss, (iii) meta-teacher learning step, where a feed-forward model, the meta-teacher, is trained to jointly optimize the outputs for the teacher and student hidden representations, and (iv) in student curriculum learning step, we train a student curriculum model to generate a suitable curriculum, a sequence of tasks $T' \in \mathbb{T}$ for which the distilled student is fine-tuned and calculated rewards.

Distilling teacher model to student model. In step 2 of our framework, we distil the student model $S$ with the knowledge acquired by the teacher model with fine-tuning. We use the task-specific loss $L_T$ along with a knowledge distillation loss $L_{KD}$ computed on the training dataset. MPDistil is a general framework and can handle any arbitrary $L_{KD}$ function, as long as it is differentiable with respect to the student model parameters $\theta_S$. Following Zhou et al. (2021), we use a convex combination of the MSE loss between the teacher and student logits and the mean squared differences between the teacher and student hidden representations. Formally, we calculate,

$$L_{T \text{student KD}} = \frac{1}{N} \sum_{i=1}^{N} \alpha L_T (y_i, \hat{y}_{(i,S)}) + (1 - \alpha) \left\| \Phi \left( \frac{\hat{y}_{(i,T)}}{\tau} \right) - \Phi \left( \frac{\hat{y}_{(i,S)}}{\tau} \right) \right\|_2^2 + \beta \left\| h_{(.,T)} - h_{(.,S)} \right\|_2$$

(1)

Here, $\Phi$ denotes the Softmax activation function. Although some studies (Hinton et al., 2015; Gu et al., 2023) use Kullback–Leibler Divergence (KLD) and reverse KLD between the teacher and student logits, following Kim et al. (2021), we use $L_2$ norm to calculate the difference between teacher and student logits, to stabilize the optimization. Hence, a student optimized with $L_{T \text{student KD}}$ is not only trained on the task $T$ standalone but also acquires the knowledge from the teacher through soft label association and shared latent, hidden representations.

Meta-teacher learning. In step 3 of MPDistil, we jointly update the teacher model parameters to optimize the teacher and student performances. To achieve this, we devise a meta-teacher $T'$, parametrized with $\theta_{T'}$. The meta-teacher follows a simple feed-forward network architecture with the ability to digest the hidden state representations from both the teacher and student models and generate the final output. In our white-box distillation framework, we assume that the output logits and hidden representations are accessible from both the teacher and student models. The meta-teacher, therefore, can act as a discriminator, separating the representations extracted by the original teacher and the student models. We train the meta-teacher parameters $\theta_{T'}$ as the meta-learner, whereas the student model parameters $\theta_S$ are learned as the inner-learner. It is interesting to notice that as opposed to the meta-teacher designed by Zhou et al. (2021), our meta-teacher has only an insignificant number (only 0.001% of the teacher model) of learnable parameters and, therefore,
is much easier to train in the meta-learning setup. We avoid a large pre-trained teacher model as cloning it to a meta-teacher and fine-tuning could be prohibitively expensive. Our meta-teacher is, therefore, computationally more efficient, even for distilling from computationally large pre-trained models.

We adopt a quiz dataset \( Q_T \) for each task \( T \), separate from the original training dataset for meta-teacher learning. We formulate two different training objectives for the meta-teacher. In the first approach, we compute a collaborative loss, in which the meta-teacher optimizes the joint task-specific losses calculated with the teacher’s and student’s hidden representations. Given a training data \( (x_i, y_i) \), we obtain the hidden states from the teacher and student models as \( h_{(i,T)} \) and \( h_{(i,S)} \), respectively, and obtain the final output from the meta-teacher model as, \( \hat{y}_{(i,T)} = T'(h_{(i,T)}; \theta_T') \) and \( \hat{y}_{(i,S)} = T'(h_{(i,S)}; \theta_T') \), respectively. If the task \( T \) is a classification task, we use only the class probabilities of the true class \( \hat{y}_{(i,T)} = \Phi(\hat{y}_{(i,T)})_c \) and \( \hat{y}_{(i,S)} = \Phi(\hat{y}_{(i,S)})_c \) for \( c = \arg \max_k y(i,k) \), the true class label. The collaborative loss is formally defined as,

\[
L_T^\text{meta col} = \begin{cases} 
- \frac{1}{2N} \sum_{i=1}^{N} \left[ \log \hat{y}_{(i,T)} + \log \hat{y}_{(i,S)} \right], & \text{if } T \text{ is a classification task} \\
\frac{1}{2} \left\| y - \hat{y}_{(i,T)} \right\|_2^2 + \frac{1}{2} \left\| y - \hat{y}_{(i,S)} \right\|_2^2, & \text{if } T \text{ is a regression task}
\end{cases}
\]

The meta-teacher acts as a discriminator and does not understand whether the teacher or the student obtains the hidden representation. Moreover, the original student and teacher parameters are frozen during meta-teacher training, and the meta-teacher is trained. Therefore, the meta-teacher only learns the optimal function to jointly optimize the teacher’s and student’s predictive performances. We compute a competitive loss in another formulation between the teacher and student representations. The competitive loss is motivated by the Wasserstein loss used in WGAN (Arjovsky et al. [2017]) and calculates the Earth Mover’s (EM) distance between the teacher and student logits. Additionally, we add the meta-teacher’s task-specific loss so that the meta-teacher can minimize its loss, along with maximizing the margin from the student model. Using the Kantorovich-Rubenstein duality (Villani et al. [2009]), we compute the competitive loss as,

\[
L_T^\text{meta com} = \begin{cases} 
- \frac{1}{N} \sum_{i=1}^{N} \left[ 2 \log \hat{y}_{(i,T)} - \log \hat{y}_{(i,S)} \right], & \text{if } T \text{ is a classification task} \\
\left\| y - \hat{y}_{(i,T)} \right\|_2 - \frac{1}{2} \left\| y - \hat{y}_{(i,S)} \right\|_2, & \text{if } T \text{ is a regression task}
\end{cases}
\]

**Proposition 1.** For any classification task \( T \), having \( \mathbb{E}[L_T^\text{meta col}] < \mathbb{E}[L_T^\text{meta com}] \) ensures stronger student with \( \mathbb{E}[\hat{y}_S] > \mathbb{E}[\hat{y}_T] \).

**Student curriculum learning.** The final step of MPDistil is the student curriculum learning, in which the student model aims to outperform the meta-teacher by training itself on a suitable set of tasks, possibly different from the task at hand \( T \). Curriculum learning (Bengio et al. [2009]) involves gradually acquiring skills by initially focusing on simpler tasks before progressing to more complex ones. We use a policy learner, a curriculum model, to learn the curriculum for the student. The curriculum model is a feed-forward network that takes the current state of the student model and samples a task (action) \( T' \in \mathbb{T} \) in each forward pass. The student model is trained on a batch sampled from \( Q_{T'} \) and is used to calculate a reward on \( Q_T \). For an updated student model, we calculate the output \( \hat{y}_{(i,S)} = S(x_i; \theta_S) \) for each \( (x_i, y_i) \sim Q_T \). Similarly, we obtain the output \( \hat{y}_{(i,T')} \) from the meta-teacher. Analogous to the previous meta-teacher learning step, we obtain only the predicted probability of the true output class for classification tasks. We evaluate two different reward functions: (i) binary reward and (ii) real reward.

\[
R^\text{binary} = \begin{cases} 
1, & \text{if } T \text{ is a classification task} \\
1 - \frac{1}{2} \left\| y_i - \hat{y}_{(i,T')} \right\|_2^2, & \text{if } T \text{ is a regression task}
\end{cases}
\]

\[
\frac{1}{2} \left\| y - \hat{y}_{(i,T)} \right\|_2 - \frac{1}{2} \left\| y - \hat{y}_{(i,S)} \right\|_2
\]

\[
\mathbb{E}[\hat{y}_S] > \mathbb{E}[\hat{y}_T]
\]

\[
\text{All the proofs presented in the paper are supplied in Appendix A.}
\]
Here, \( \mathbb{1} \) is the indicator function. Similarly,

\[
R_{\text{real}}^r = \begin{cases} 
\hat{y}_t(S) - \hat{y}_t(T), & \text{if } T \text{ is a classification task} \\
\left\| y_t - \hat{y}_t(T) \right\|_2^2 - \left\| y_t - \hat{y}_t(S) \right\|_2^2, & \text{if } T \text{ is a regression task}
\end{cases}
\]

(5)

**Proposition 2.** For any classification task \( T \), \( \mathbb{E}[R_{\text{binary}}] \geq \mathbb{E}[R_{\text{real}}] \).

The expected reward for an entire episode is calculated as \( G_t = \sum_{k=t+1}^{E} \gamma^{k-t-1} R_k \), with a suitable discount factor \( \gamma \) and the length of the episode \( E \). In our case, the length of an episode is the number of quiz batches for task \( T \). We use the Monte Carlo policy gradient algorithm, REINFORCE (Williams, 1992; Sutton et al., 1999), to update the curriculum model parameters \( \theta_\lambda \) to maximize the expected reward. The curriculum model samples tasks that maximize the reward for the student model, i.e., increases the difference between the student and teacher models. In other words, having trained on these newly sampled tasks, the student model can achieve better results than the teacher when evaluated through meta-teacher, leading to higher rewards.

4 **Experimental Details and Results**

Following the contemporary knowledge distillation studies (Liu et al., 2022a; Zhou et al., 2021), we evaluate MPDistil on 15 different natural language understanding tasks from SuperGLUE (Wang et al., 2019) and GLUE (Wang et al., 2018) benchmarks. We choose the classification tasks – CB, COPA, RTE, WiC, WSC and BoolQ from the SuperGLUE benchmark. From the GLUE benchmark, we use MNLI, MNLI Mismatch, QNLI, QQP, WNLI, MRPC, RTE, SST-2 and STS-B tasks, out of which only STS-B is a regression task and the remaining are classification tasks. We elaborate on these tasks in Appendix B.1. We utilize the pre-trained BERT-base (Devlin et al., 2018) model as the teacher (111M parameters) and the pre-trained BERT 6-layer model as the student (66M parameters). Additionally, we evaluate MPDistil on a pre-trained large language model, DeBERTa-v2-xxlarge (He et al., 2020) (1.4B parameters) and its 12-layered variant (547M parameters) as the student model. As described in Section 5 for teacher, meta-teacher and student models, we use multiple output heads depending on the number of tasks. For SuperGLUE and GLUE, we use six and nine output heads (one for each task), respectively. However, all the models are trained on each task separately. The BERT-base model utilizes a pooling layer for learning a suitable projection from the output representation corresponding to the first token ([CLS] token) and uses it for downstream tasks. We use the student model pooling layer weight matrix and learn a projection vector on \( \mathbb{R}^{768} \) to represent the state of the student model in curriculum learning. Meta-teacher learning and student curriculum learning require a separate labelled dataset, for which we split the original training dataset provided in SuperGLUE and GLUE tasks into an updated training and quiz sets by 9 : 1. However, we use the original datasets provided in the benchmarks for dev and test. We furnish other experimental details in Appendix B.2. We evaluate MPDistil with 20 contemporary KD and MetaKD methods, some of which are elaborated in Appendix B.3.

**Results.** We report the performances of different models – the fine-tuned teacher model, the distilled student, the meta-teacher and the updated student with MPDistil on the SuperGLUE tasks in Table 1. Additionally, we report the results obtained from the student model when fine-tuned on the tasks without distillation. As an ablation to MPDistil, we evaluate the performance of an alternate student model that is only trained with \( \mathcal{L}_{KD} \) in step 4 (c.f. Figure 4), without curriculum learning. We denote this model as ‘(-) Curriculum learning’. On the SuperGLUE dev split, the meta-teacher obtains 0.7% better performance than the original teacher model. Although both the loss formulations are adequate for the meta-teacher, collaborative loss helps the meta-teacher obtain better performance, as opposed to the competitive loss. Collaborative loss is equally essential for better student distillation. On SuperGLUE dev, the student distilled with collaborative loss and binary reward achieves 5.9% and 1.5% better than the base student and teacher models, respectively. Although both reward functions are equally effective, the distilled student achieves 0.2% improvement with binary reward than with the real reward function. Without student curriculum learning, the ablation model achieves 2.4% better than the base student model. However, the improvement over the teacher model remains muted at −2.1%, indicating the positive impact of curriculum learning on the student.

The pre-trained models are obtained from [https://huggingface.co/models/](https://huggingface.co/models/).
Published as a conference paper at ICLR 2024

against the distilled student and the meta-teacher model. We highlight the cases in blue in which we observe the teacher and student base models trained on the original train set and compare them.

We report the accuracy scores for each of these classification tasks. We

MPDistil

3

of

MPDistil

performance drop with

and RTE tasks, the DeBERTa-v2-xxlarge model achieves nearly state-of-the-art results, whereas the

Even on the SuperGLUE test split, the BERT student model distilled with

MPDistil on GLUE dev set (Table 4). BERT-6L distilled with

methods – CKD (–) Curriculum learning + Comp loss

KD techniques on four tasks on both GLUE dev and test datasets.

Table 2 (with BERT models) and

in distilling knowledge

empower the student model to perform

As the performance of the distilled

student depends on the teacher’s performance, we derive an evaluation metric,

∆Margin = (Performance of student – Performance of teacher), to compare the distillation methods. The metric indicates the student’s performance in terms of the teacher (or the improvement over the teacher), which is agnostic to the original teacher’s performance.

It is worth noting that existing distillation techniques do not empower the student model to perform better than the teacher model, often leading to a negative ∆Margin. We report ∆Margin for different distillation techniques on the SuperGLUE dev in Table 2 (with BERT models) and Table 3 (with DeBERTa models). With the BERT teacher model, MPDistil shows a significant ∆Margin of +1.5% on average compared to two most competitive baselines, TinyBERT (average ∆Margin = –0.5%) and MetaDistil (average ∆Margin = –1.0%).

Even on the SuperGLUE test split, the BERT student model distilled with MPDistil outperforms the BERT teacher model on three out of six tasks (Table 5 in Appendix B.4). The margin between the teacher and student models is higher with the DeBERTa model. On COPA

Table 3 (with DeBERTa models). With

MPDistil

on GLUE dev set (Table 4). BERT-6L distilled with MPDistil achieves an average improvement of 3.1% over the base student model.

Similar improvements are observed on the GLUE test set (Table 7 in Appendix B.4). In terms of accuracy, the student model distilled with MPDistil outperforms the teacher BERT-base model on four GLUE tasks on both the dev and test split. Table 6 and Table 8 in Appendix B.4 show that MPDistil achieves the highest ∆Margin among all the KD techniques on four tasks on both GLUE dev and test datasets.

<table>
<thead>
<tr>
<th>Methods</th>
<th>BERTQ</th>
<th>CR</th>
<th>COPA</th>
<th>RTE</th>
<th>WSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teacher (BERT-base)</td>
<td>75.3</td>
<td>83.9</td>
<td>63.0</td>
<td>67.1</td>
<td>73.1</td>
</tr>
<tr>
<td>Teacher (DeBERTa-v2-xxlarge)</td>
<td>84.6</td>
<td>82.1</td>
<td>84.0</td>
<td>90.6</td>
<td>57.7</td>
</tr>
<tr>
<td>Student (BERT-base)</td>
<td>71.6</td>
<td>75.0</td>
<td>55.0</td>
<td>64.6</td>
<td>50.0</td>
</tr>
<tr>
<td>Student (DeBERTa-v2-12L)</td>
<td>82.1</td>
<td>82.1</td>
<td>64.0</td>
<td>70.4</td>
<td>57.8</td>
</tr>
<tr>
<td>Distilled Student</td>
<td>73.0</td>
<td>80.4</td>
<td>54.0</td>
<td>66.4</td>
<td>57.7</td>
</tr>
<tr>
<td>Distilled Student (DeBERTa-v2-xxlarge)</td>
<td>82.6</td>
<td>75.6</td>
<td>63.0</td>
<td>80.4</td>
<td>55.9</td>
</tr>
<tr>
<td>MPDistil (Ours)</td>
<td>86.6</td>
<td>82.1</td>
<td>85.0</td>
<td>90.8</td>
<td>55.9</td>
</tr>
</tbody>
</table>

Table 1: Results of our method (MPDistil) on the SuperGLUE dev set with the BERT and DeBERTa teacher models. We report the accuracy scores for each of these classification tasks. We observe the teacher and student base models trained on the original train set and compare them against the distilled student and the meta-teacher model. We highlight the cases in blue in which the meta-teacher outperforms the original teacher model. Scenarios highlighted with brown are the cases in which the distilled student model performs better than the teacher.

<table>
<thead>
<tr>
<th>Methods</th>
<th>BERTQ</th>
<th>CR</th>
<th>COPA</th>
<th>RTE</th>
<th>WSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>KD (Hinton et al. 2015)</td>
<td>–13.3</td>
<td>–19.1</td>
<td>–4.3</td>
<td>–3.7</td>
<td>–9.1</td>
</tr>
<tr>
<td>PD (Turc et al. 2019)</td>
<td>–9.6</td>
<td>–9.5</td>
<td>–0.3</td>
<td>–13.5</td>
<td>–6.9</td>
</tr>
<tr>
<td>Distillation (Hinton et al. 2015)</td>
<td>–1.7</td>
<td>–5.9</td>
<td>–6.0</td>
<td>–3.8</td>
<td>–20.4</td>
</tr>
<tr>
<td>Distillation (Park et al. 2019)</td>
<td>–7.7</td>
<td>–7.7</td>
<td>–1.0</td>
<td>–12.0</td>
<td>–5.8</td>
</tr>
<tr>
<td>Theorem (Hinton et al. 2015)</td>
<td>–1.6</td>
<td>–3.6</td>
<td>–4.3</td>
<td>–4.8</td>
<td>–1.8</td>
</tr>
<tr>
<td>TinyKD (Turc et al. 2019)</td>
<td>–4.4</td>
<td>–1.2</td>
<td>4.3</td>
<td>–3.7</td>
<td>1.7</td>
</tr>
<tr>
<td>MobileKD (Turc et al. 2020)</td>
<td>–4.8</td>
<td>–2.4</td>
<td>–0.7</td>
<td>–14.0</td>
<td>–2.3</td>
</tr>
<tr>
<td>MDI (Agapit et al. 2019)</td>
<td>–10.1</td>
<td>–17.3</td>
<td>–1.0</td>
<td>–18.4</td>
<td>–9.0</td>
</tr>
<tr>
<td>MultiKD (Turc et al. 2020)</td>
<td>–5.5</td>
<td>–11.9</td>
<td>–4.0</td>
<td>–5.3</td>
<td>–1.2</td>
</tr>
<tr>
<td>Universal-KD (Aguilar et al. 2020)</td>
<td>–2.7</td>
<td>–14.3</td>
<td>–4.0</td>
<td>–6.3</td>
<td>–2.5</td>
</tr>
<tr>
<td>Continual-KD (Bahl et al. 2021)</td>
<td>–2.2</td>
<td>–11.3</td>
<td>–5.3</td>
<td>–4.8</td>
<td>–1.3</td>
</tr>
<tr>
<td>LRC (Bahl et al. 2021)</td>
<td>–4.5</td>
<td>–9.5</td>
<td>–0.3</td>
<td>–16.4</td>
<td>–8.5</td>
</tr>
<tr>
<td>DIITO (Wu et al. 2021b)</td>
<td>–1.5</td>
<td>–3.9</td>
<td>–5.9</td>
<td>–14.0</td>
<td>–6.3</td>
</tr>
<tr>
<td>SID (Aguilar et al. 2020)</td>
<td>–8.8</td>
<td>–5.9</td>
<td>–3.3</td>
<td>–14.0</td>
<td>–6.3</td>
</tr>
<tr>
<td>CKD (Park et al. 2021)</td>
<td>–7.8</td>
<td>–6.6</td>
<td>–1.0</td>
<td>–11.7</td>
<td>–7.3</td>
</tr>
<tr>
<td>Universal-KD (Aguilar et al. 2020)</td>
<td>–1.8</td>
<td>–5.4</td>
<td>–7.3</td>
<td>–2.8</td>
<td>–0.6</td>
</tr>
<tr>
<td>DIITO (Wu et al. 2021b)</td>
<td>–3.0</td>
<td>–5.0</td>
<td>–6.0</td>
<td>–7.5</td>
<td>–5.4</td>
</tr>
<tr>
<td>Continual-KD (Bahl et al. 2021)</td>
<td>–8.0</td>
<td>–7.1</td>
<td>–2.7</td>
<td>–14.2</td>
<td>–7.9</td>
</tr>
<tr>
<td>ALP-KD (Passban et al. 2021)</td>
<td>–10.4</td>
<td>–7.7</td>
<td>0.7</td>
<td>–12.4</td>
<td>–5.8</td>
</tr>
<tr>
<td>Theseus (Xu et al. 2020)</td>
<td>–6.1</td>
<td>–6.6</td>
<td>–1.0</td>
<td>–7.0</td>
<td>–3.0</td>
</tr>
<tr>
<td>DIITO (Wu et al. 2021b)</td>
<td>–2.7</td>
<td>–1.8</td>
<td>1.0</td>
<td>–2.0</td>
<td>–1.6</td>
</tr>
<tr>
<td>MPDistil (Ours)</td>
<td>–1.9</td>
<td>0.0</td>
<td>7.0</td>
<td>0.4</td>
<td>2.5</td>
</tr>
<tr>
<td>(a) Curriculum learning</td>
<td>–2.8</td>
<td>–5.3</td>
<td>–4.0</td>
<td>–1.8</td>
<td>3.2</td>
</tr>
</tbody>
</table>

Table 2: ∆Margin reported for different distillation methods on SuperGLUE tasks dev split with BERT models. The higher the value of ∆Margin, the more effective the distillation technique. ∆Margin > 0 indicates that the student outperforms the teacher. The results highlighted with † are taken from [Tan et al. 2023]. For MPDistil and the ablation, we use the best scores obtained across different loss and reward configurations.
Figure 3: Student rewards for different tasks.

Table 4: Results of our method (MPDistil) on the GLUE dev set with BERT model.

Table 3: ∆Margin for different distillation methods on the dev split of SuperGLUE tasks with DeBERTa models.

5 DISCUSSION

Can better learners teach better? Figure 2 highlights the distribution of meta-teacher validation loss, and shows lower validation loss with collaborative loss for most tasks than competitive loss. As Proposition 1 suggested, lower collaborative loss than competitive loss asserts stronger students. We analyze how a lower loss enables a better meta-teacher and encourages a better student. Towards this, we observe the improvement shown by the meta-teacher over the original fine-tuned teacher. We analyze how a lower loss enables a better meta-teacher and encourages a better student. Towards this, we observe the improvement shown by the meta-teacher over the original fine-tuned teacher.

How does higher student reward impact student’s performance? Figure 3 highlights the distribution of rewards gathered by the distilled student across tasks. On the classification tasks, the expected binary reward obtained is 0.81, as opposed to the expected real reward of 0.35. As per Proposition 2, the expected binary reward on these classification tasks is always higher than the expected real reward. With binary reward formulation, the correlation between the performances of the distilled student and the meta-teacher is 0.77, significantly higher than the correlation with the real reward (0.60). On BoolQ, the expected binary reward is < 0.5, indicating that the student model is more underconfident than the teacher model. This propagates to the poorer dev performance by
the distilled student model. In contrast, on the tasks with the highest expected reward such as RTE, SST-2, CB and COPA, the distilled student outperforms the teacher model with the widest margins.

**How does a student select learning curricula?** We analyze the cumulative weightage of different tasks in the student curriculum in Figure 4a. On the low-performing tasks such as MNLI, MNLI-mm, and QNLI, the model explores all the different tasks with nearly uniform weightage. Compared to this, on high-performing tasks such as RTE, CB, and COPA, the model perfectly balances exploration and exploitation in selecting the suitable curriculum. For instance, on CB, the model only chooses the COPA task in its curriculum. On the other hand, on COPA, CB has the highest weightage in the curriculum. On the RTE task, the model selects the same task repeatedly in the curriculum. To understand how the model preserves the curriculum throughout its learning, we calculate a similarity metric, \( \text{sim}_{i,i+1} = \frac{|\text{LCS}(C_i, C_{i+1})|}{|C_i|} \), between two learned curricula \( C_i \) and \( C_{i+1} \) in two consecutive episodes. LCS denotes the longest common subsequence. We visualize the \( \text{sim}_{i,i+1} \) distribution for different tasks in Figure 4b. A high similarity score indicates that the model only exploits and shows reluctance to learn new curricula. Interestingly, the model has lower average similarity on the high-performing tasks, indicating more exploration. However, after learning the most suitable curriculum, the model stops exploration and exploits the same set of tasks to preserve high rewards. On the other hand, on low-performing tasks like MNLI, MNLI-mm and QNLI, the model shows higher average similarity with low variance, indicating more restrictive behavior.

We further calculate the count of each task in each curriculum and the chi-square distance between consecutive curricula to understand how different tasks are preserved within curricula. We report the distribution of chi-square distances in Figure 4c. For high-performing tasks, we observe a significantly lower chi-square distance between consecutive curricula. Lower chi-square and low average \( \text{sim} \) values conclude that for these tasks, the curriculum learning model does not explore newer tasks in the curriculum but rather explores the different sequences among them. It justifies that not only are the tasks important within curricula, but their sequence also equally matters.

6 **CONCLUSION**

This paper introduced MPDistil, a method for distilling meta-policy knowledge, emphasizing the importance of creating a suitable joint utility to enhance the teacher’s distillation capabilities. Furthermore, we underscored the necessity of employing multi-task curricula to produce more robust and effective student models through teacher knowledge distillation. It is essential to note that our framework is generic and can be applied to distilling large language models, regardless of their underlying architecture. While our empirical study primarily focused on natural language understanding tasks, our methodology can also be extended to other tasks involving natural language generation and reasoning. While our student curricula were designed with tasks featuring similar abstractions, it would be worth noting the potential impact of diverse tasks with varying levels of abstraction and complexity on enhancing the generalizability of student models.
REFERENCES


Wenxian Shi, Yuxuan Song, Hao Zhou, Bohan Li, and Lei Li. Learning from deep model via exploring local targets. 2020.


A PROOFS OF THE THEORETICAL RESULTS

A.1 PROOF OF PROPOSITION 1

The expected collaborative and competitive loss for any given teacher and student logit is defined as $\mathbb{E}[− \log \hat{y}_{i,T} − \log \hat{y}_{i,S}]$ and $\mathbb{E}[−2 \log \hat{y}_{i,T} + \log \hat{y}_{i,S}]$, respectively.
\[ \mathbb{E}[\mathcal{L}_T^\text{meta col}] < \mathbb{E}[\mathcal{L}_T^\text{meta com}] \]
\[ \iff \mathbb{E}[- \log \hat{y}_{(i,T)} - \log \hat{y}_{(i,S)}] < \mathbb{E}[-2 \log \bar{y}_{(i,T)} + \log \hat{y}_{(i,S)}] \]
\[ \iff \mathbb{E}[\log \bar{y}_{(i,T)}] < 2\mathbb{E}[\log \hat{y}_{(i,S)}] \]
\[ \iff \mathbb{E}[\log \bar{y}_{(i,T)}] < \mathbb{E}[\log \hat{y}_{(i,S)}] \]
\[ \iff \mathbb{E}[\hat{y}_{(i,S)}^2] - \log \bar{y}_{(i,T)} > 0 \]
\[ \iff \log \mathbb{E}[\hat{y}_{(i,S)}^2] - \log \mathbb{E}[\bar{y}_{(i,T)}] > 0 \text{ (Jensen's inequality)} \]
\[ \iff \log \frac{\mathbb{E}[\hat{y}_{(i,S)}^2]}{\mathbb{E}[\bar{y}_{(i,T)}]} > 0 \]
\[ \iff \log \frac{\mathbb{E}[\hat{y}_{(i,S)}^2]}{\mathbb{E}[\bar{y}_{(i,T)}]} > 1. \]

As \( \bar{y}_{(i,S)} > \hat{y}_{(i,S)}^2 \), it proves that \( \mathbb{E}[\hat{y}_{(i,S)}] > \mathbb{E}[\bar{y}_{(i,T)}] \). ■

### A.2 Proof of Proposition 2

\[ \mathbb{E}[\mathbf{1}_{\bar{y}_{(i,S)} > \hat{y}_{(i,T')}}] = \mathbb{P}_{\hat{y}_{(i,S)}, \hat{y}_{(i,T')}}[\hat{y}_{(i,S)} - \hat{y}_{(i,T')} > 0] \]
\[ = \int_{\hat{y}_{(i,S)} = 0}^{1} \int_{\hat{y}_{(i,T')} = 0}^{\hat{y}_{(i,S)}} d\hat{y}_{(i,S)} \cdot d\hat{y}_{(i,T')} \]

Being two probability distributions, the random variable \( \bar{y}_{(i,S)} - \hat{y}_{(i,T')} \) has its range \([-1, 1] \).

\[ \mathbb{E}[\bar{y}_{(i,S)} - \hat{y}_{(i,T')}] = \int_{\hat{y}_{(i,S)} = 0}^{1} \int_{\hat{y}_{(i,T')} = 0}^{\hat{y}_{(i,S)}} (\bar{y}_{(i,S)} - \hat{y}_{(i,T')}) \cdot d\hat{y}_{(i,S)} \cdot d\hat{y}_{(i,T')} \]
\[ = \int_{\hat{y}_{(i,S)} = \hat{y}_{(i,T')}}^{1} \int_{\hat{y}_{(i,T')} = 0}^{\hat{y}_{(i,S)}} (\bar{y}_{(i,S)} - \hat{y}_{(i,T')}) \cdot d\hat{y}_{(i,S)} \cdot d\hat{y}_{(i,T')} \]
\[ + \int_{\hat{y}_{(i,S)} = \hat{y}_{(i,T')}}^{1} \int_{\hat{y}_{(i,T')} = 0}^{\hat{y}_{(i,S)}} (\bar{y}_{(i,S)} - \hat{y}_{(i,T')}) \cdot d\hat{y}_{(i,S)} \cdot d\hat{y}_{(i,T')} \]

The second term in the above expression is less than or equal to 0 as \( \bar{y}_{(i,S)} - \hat{y}_{(i,T')} \leq 0 \) for any \( \bar{y}_{(i,S)} \in [0, \hat{y}_{(i,T')}]. \) Therefore,

\[ \mathbb{E}[\bar{y}_{(i,S)} - \hat{y}_{(i,T')}] \leq \int_{\hat{y}_{(i,S)} = \hat{y}_{(i,T')}}^{1} \int_{\hat{y}_{(i,T')} = 0}^{\hat{y}_{(i,S)}} (\bar{y}_{(i,S)} - \hat{y}_{(i,T')}) \cdot d\hat{y}_{(i,S)} \cdot d\hat{y}_{(i,T')} \]
\[ = \int_{\hat{y}_{(i,S)} = \hat{y}_{(i,T')}}^{1} \int_{\hat{y}_{(i,T')} = 0}^{\hat{y}_{(i,S)}} d\hat{y}_{(i,S)} \cdot d\hat{y}_{(i,T')} \text{ (Change of limits)} \]
\[ \leq \int_{\hat{y}_{(i,S)} = 0}^{1} \int_{\hat{y}_{(i,T')} = 0}^{\hat{y}_{(i,S)}} d\hat{y}_{(i,S)} \cdot d\hat{y}_{(i,T')} \]
\[ = \mathbb{E}[\mathbf{1}_{\bar{y}_{(i,S)} > \hat{y}_{(i,T')}}]. \]

### B Experimental Details and Results

#### B.1 Datasets

This section briefly describes the tasks of SuperGLUE [Wang et al., 2019] and GLUE [Wang et al., 2018] benchmarks.
B.1.1 SUPERGLUE BENCHMARK

**BoolQ**: Boolean Questions comprise of binary questions using the Google search engine as their source of questions; they are then paired with appropriate paragraphs from Wikipedia articles that contain the relevant answers (Clark et al., 2019).

**CB**: CommitmentBank comprises of short texts with embedded clauses. The examples are taken from sources like British National Corpus Fiction and Wall Street Journal. It involves a three-class textual entailment task. Each example includes a premise and the corresponding hypothesis along with the class label "contradiction", "neutral", or "entailment" (De Marneffe et al., 2019).

**COPA**: Choice of Plausible Alternatives is a causal reasoning task which involves selecting the most plausible choice for a cause or effect given a premise. (Roemmele et al., 2011).

**WiC**: Word-in-Context is a task focused on word sense disambiguation, comprising of binary classification of pairs of sentences. In this task, two text snippets are provided, each containing a word that could have multiple meanings. The goal is to ascertain whether the word has the same meaning in both sentences (Pilevch & Camacho-Collados 2018).

**WSC**: Winograd Schema Challenge is a coreference resolution task where each sentence example includes a passage, a pronoun, and a choice of nouns from that sentence. The task objective is to determine whether the pronoun and noun refer to each other or not (Levesque et al., 2012).

B.1.2 GLUE BENCHMARK

**SST-2**: Stanford Sentiment Treebank is a dataset derived from movie reviews, annotated with sentiment labels. The goal is to predict the sentiment ("positive"/"negative") of the reviews, employing a two-way classification (Socher et al., 2013).

**MRPC**: Microsoft Research Paraphrase Corpus consists of sentence pairs obtained from online news sources and are human-annotated to ascertain their semantic equivalence. This task involves predicting whether two given sentences are paraphrases of each other or not (Dolan & Brockett, 2005).

**STS-B**: Semantic Textual Similarity Benchmark predicts the degree of similarity between two given sentences on a scale from 0-5. The dataset is compiled from diverse origins, including videos and image descriptions, news headlines, and natural language inference information. (Cer et al., 2017).

**QQP**: Quora Question Pairs involves predicting whether two given questions asked on the question-answering website Quora are semantically equivalent or not (Chen et al., 2018).

**MNLI**: Multi-Genre Natural Language Inference involves predicting whether a given hypothesis is "entailed", "neutral", or "contradicted" for a given premise. The dataset is collected from various sources, including fiction, government reports, and transcribed speech (Williams et al., 2017).

**QNLI**: Question-answering NLI is a task based on the Stanford Question Answering Dataset. It involves determining whether a given context sentence contains the answer to a corresponding question (Wang et al., 2018).

**RTE**: Recognizing Textual Entailment is compiled from collection of datasets that originate from a series of annual textual entailment challenges (Dagan et al., 2005; Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009). These datasets are constructed using news and Wikipedia text sources. The datasets are transformed into a two-class split, collapsing "neutral" and "contradiction" categories into "not entailment".

**WNLI**: Winograd NLI requires selecting the referent of a pronoun from choices by converting it into sentence pair classification. The task assesses whether substituting the pronoun with possible referents results in sentence entailment (Levesque et al., 2012).

B.2 EXPERIMENTAL SETUP

For all the SuperGLUE and GLUE tasks, we use a maximum sequence length of 128. All the results reported in the paper (except for the results obtained from other baselines) are obtained from running a grid search over the set of hyperparameters, teacher learning rate from \{2e-5, 3e-5\}, the student
B.3 Baselines

To evaluate the impact of our proposed distillation method, we undertake several competitive KD and MetaKD baselines.

**PKD** ([Sun et al., 2019]) uses a patient learning approach where instead of using the last layer of teacher for distilling knowledge into the student, the student utilises the intermediate layer representation of the teacher model.

**CKD** ([Park et al., 2021]) transfers knowledge from teacher to student, utilising the word representations contextual knowledge. It defines two contextual knowledge objectives – *Word Relation (WR)* and *Layer Transforming Relation (LTR)*, examining the relationships among words in a particular layer and across different layers, respectively.

**TinyBERT** ([Jiao et al., 2019]) employs a two-stage framework, general and task-specific distillation of BERT-base into a tiny 4-layer student model, using a multiple distillation objective. In addition to logit-based matching, the teacher transfers its knowledge through embedding layers, intermediate hidden states, and attention matrices.

**MetaDistil**: Zhou et al. (2021) argued that the teacher model is not optimized for transferring knowledge to the student model. MetaDistil proposes a trainable teacher setup wherein the student’s predictive performance is considered while fine-tuning the teacher model using a quiz dataset. The updated teacher is further used to distil knowledge into the student further.

B.4 Results

Table 5 and Table 7 highlight different model performances on SuperGLUE and GLUE test sets, respectively. BERT-6L distilled with MPDistil achieves better accuracy than the BERT-base teacher model in three SuperGLUE and four GLUE tasks. Table 6 and Table 8 highlight the $\Delta$Margin for MPDistil on GLUE dev and test set respectively. It is worth noting that most contemporary KD methods fail to improve the student’s performance over the teacher’s on GLUE test. In contrast, the student model distilled with MPDistil achieves positive $\Delta$Margin on MRPC, QQP, RTE and SST-2. The ablation model without curriculum learning achieves positive $\Delta$Margin on the STS-B task. These superior performances demonstrate the generalizability of the student model distilled with MPDistil.

<table>
<thead>
<tr>
<th>Methods</th>
<th>BoolQ</th>
<th>CB</th>
<th>COPA</th>
<th>RTE</th>
<th>WiC</th>
<th>WSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teacher (BERT-base)</td>
<td>74.6</td>
<td>83.6</td>
<td>56.6</td>
<td>63.2</td>
<td>54.8</td>
<td>65.1</td>
</tr>
<tr>
<td>Student (BERT-base 6L)</td>
<td>71.3</td>
<td>81.2</td>
<td>53.0</td>
<td>64.1</td>
<td>56.4</td>
<td>65.1</td>
</tr>
<tr>
<td>Distilled Student</td>
<td>72.1</td>
<td>82.0</td>
<td>52.8</td>
<td>64.3</td>
<td>53.9</td>
<td>65.1</td>
</tr>
<tr>
<td>Meta-teacher (+) Col loss</td>
<td>73.0</td>
<td>84.0</td>
<td>56.8</td>
<td>62.8</td>
<td>54.9</td>
<td>65.1</td>
</tr>
<tr>
<td>(+) Com loss</td>
<td>72.9</td>
<td>84.0</td>
<td>58.8</td>
<td>62.9</td>
<td>54.4</td>
<td>64.4</td>
</tr>
<tr>
<td>Student with MPDistil</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(+) Col loss + Binary reward</td>
<td>72.1</td>
<td>81.2</td>
<td>50.6</td>
<td>64.2</td>
<td>53.8</td>
<td>63.0</td>
</tr>
<tr>
<td>(+) Col loss + Real reward</td>
<td>71.1</td>
<td>82.4</td>
<td>53.6</td>
<td>64.2</td>
<td>53.8</td>
<td>64.4</td>
</tr>
<tr>
<td>(+) Com loss + Binary reward</td>
<td>71.2</td>
<td>82.0</td>
<td>51.6</td>
<td>64.5</td>
<td>55.0</td>
<td>65.8</td>
</tr>
<tr>
<td>(+) Com loss + Real reward</td>
<td>71.3</td>
<td>81.6</td>
<td>50.6</td>
<td>64.2</td>
<td>54.7</td>
<td>66.4</td>
</tr>
<tr>
<td>(-) Curriculum learning + Col loss</td>
<td>72.4</td>
<td>82.0</td>
<td>53.0</td>
<td>62.8</td>
<td>54.3</td>
<td>65.1</td>
</tr>
<tr>
<td>(-) Curriculum learning + Comp loss</td>
<td>72.7</td>
<td>82.4</td>
<td>53.0</td>
<td>61.2</td>
<td>55.2</td>
<td>65.8</td>
</tr>
</tbody>
</table>

Table 5: Results of our method on the SuperGLUE test sets with the BERT model.
Table 6: ∆Margin reported for different distillation methods on GLUE tasks dev split with the BERT model. The results highlighted with † are obtained from Zhou et al. [2021].

Table 7: Results of our method (MPDistil) on the GLUE test set with the BERT model.

Table 8: ∆Margin reported for different distillation methods on GLUE tasks test split with the BERT model. The results highlighted with † are obtained from Zhou et al. [2021].

Table 9: Performance of our method (MPDistil) along with standard deviation on the SuperGLUE dev sets with the BERT model.
Table 10: Performance of our method (along with standard deviation) including performance variance on the SuperGLUE dev sets with the DeBERTa model.

<table>
<thead>
<tr>
<th>Methods</th>
<th>BoolQ</th>
<th>CB</th>
<th>COPA</th>
<th>RTE</th>
<th>WiC</th>
<th>WSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student with MPDistil</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(+) Col loss + Binary reward</td>
<td>82.7 ± 2.86</td>
<td>76.8 ± 3.91</td>
<td>67.0 ± 2.58</td>
<td>79.8 ± 2.25</td>
<td>61.0 ± 0.96</td>
<td>66.3 ± 0.76</td>
</tr>
<tr>
<td>(+) Col loss + Real reward</td>
<td>82.7 ± 0.18</td>
<td>76.7 ± 1.31</td>
<td>67.0 ± 2.87</td>
<td>78.0 ± 1.08</td>
<td>60.0 ± 0.53</td>
<td>65.4 ± 4.40</td>
</tr>
<tr>
<td>(+) Com loss + Binary reward</td>
<td>82.4 ± 0.49</td>
<td>76.8 ± 2.56</td>
<td>69.0 ± 3.90</td>
<td>80.9 ± 2.27</td>
<td>59.7 ± 0.47</td>
<td>65.4 ± 2.94</td>
</tr>
<tr>
<td>(+) Com loss + Real reward</td>
<td>82.6 ± 0.22</td>
<td>76.8 ± 1.27</td>
<td>68.0 ± 3.28</td>
<td>77.6 ± 1.28</td>
<td>60.0 ± 0.45</td>
<td>64.4 ± 3.87</td>
</tr>
</tbody>
</table>

Table 11: Performance of our method (MPDistil) along with standard deviation on the GLUE dev set with BERT model.

<table>
<thead>
<tr>
<th>Methods</th>
<th>BoolQ</th>
<th>CB</th>
<th>COPA</th>
<th>RTE</th>
<th>WiC</th>
<th>WSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student with MPDistil</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(+) Col loss + Binary reward</td>
<td>80.5 ± 1.38</td>
<td>80.8 ± 0.79</td>
<td>81.4 ± 1.05</td>
<td>86.7 ± 1.13</td>
<td>90.2 ± 0.28</td>
<td>86.9 ± 0.44</td>
</tr>
<tr>
<td>(+) Com loss + Binary reward</td>
<td>80.4 ± 1.15</td>
<td>80.7 ± 0.54</td>
<td>81.6 ± 1.38</td>
<td>86.8 ± 1.67</td>
<td>90.3 ± 0.51</td>
<td>87.0 ± 0.49</td>
</tr>
<tr>
<td>(+) Com loss + Real reward</td>
<td>80.3 ± 1.42</td>
<td>81.0 ± 0.61</td>
<td>81.5 ± 0.86</td>
<td>86.8 ± 0.98</td>
<td>90.3 ± 0.12</td>
<td>87.0 ± 0.32</td>
</tr>
</tbody>
</table>

Table 11: Performance of our method (MPDistil) along with standard deviation on the GLUE dev set with BERT model.
### Table 12: Performance of our method (MPDistil) on the SuperGLUE dev set with OPT model.

<table>
<thead>
<tr>
<th>Methods</th>
<th>BoolQ</th>
<th>CB</th>
<th>COPA</th>
<th>RTE</th>
<th>WiC</th>
<th>WSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teacher (OPT-1.3B)</td>
<td>64.9</td>
<td>83.9</td>
<td>55.0</td>
<td>53.4</td>
<td>56.7</td>
<td>63.5</td>
</tr>
<tr>
<td>Student (OPT 12L)</td>
<td>63.4</td>
<td>75.0</td>
<td>54.0</td>
<td>53.1</td>
<td>53.8</td>
<td>63.5</td>
</tr>
<tr>
<td>Distilled Student</td>
<td>65.6</td>
<td>67.9</td>
<td>55.0</td>
<td>52.7</td>
<td>54.7</td>
<td>63.5</td>
</tr>
<tr>
<td>Meta-teacher</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(+) Col loss</td>
<td>64.9</td>
<td>80.4</td>
<td>55.0</td>
<td>50.9</td>
<td>56.1</td>
<td>63.5</td>
</tr>
<tr>
<td>(+) Com loss</td>
<td>65.4</td>
<td>82.1</td>
<td>55.0</td>
<td>53.8</td>
<td>56.0</td>
<td>63.5</td>
</tr>
<tr>
<td>Student with MPDistil</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(+) Col loss + Binary reward</td>
<td>65.7</td>
<td>71.4</td>
<td>55.0</td>
<td>55.6</td>
<td>54.7</td>
<td>63.5</td>
</tr>
<tr>
<td>(+) Col loss + Real reward</td>
<td>65.6</td>
<td>73.2</td>
<td>55.0</td>
<td>54.9</td>
<td>56.1</td>
<td>63.5</td>
</tr>
<tr>
<td>(+) Com loss + Binary reward</td>
<td>65.6</td>
<td>76.8</td>
<td>55.0</td>
<td>53.8</td>
<td>56.1</td>
<td>63.5</td>
</tr>
<tr>
<td>(+) Com loss + Real reward</td>
<td>65.6</td>
<td>73.2</td>
<td>55.0</td>
<td>53.8</td>
<td>56.7</td>
<td>63.5</td>
</tr>
<tr>
<td>(-) Curriculum learning + Col loss</td>
<td>65.6</td>
<td>73.2</td>
<td>55.0</td>
<td>52.7</td>
<td>54.7</td>
<td>63.5</td>
</tr>
<tr>
<td>(-) Curriculum learning + Comp loss</td>
<td>64.3</td>
<td>71.4</td>
<td>54.0</td>
<td>54.5</td>
<td>54.9</td>
<td>63.5</td>
</tr>
</tbody>
</table>

### Table 13: ∆Margin for different distillation methods on SuperGLUE dev tasks with OPT model.

<table>
<thead>
<tr>
<th>Methods</th>
<th>BoolQ</th>
<th>CB</th>
<th>COPA</th>
<th>RTE</th>
<th>WiC</th>
<th>WSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>PKD Sun et al. (2019)</td>
<td>-0.5</td>
<td>-8.9</td>
<td>0.0</td>
<td>0.7</td>
<td>-2.8</td>
<td>0.0</td>
</tr>
<tr>
<td>MPDistil (Ours)</td>
<td><strong>0.8</strong></td>
<td><strong>-7.1</strong></td>
<td>0.0</td>
<td><strong>1.5</strong></td>
<td><strong>0.0</strong></td>
<td>0.0</td>
</tr>
<tr>
<td>(-) Curriculum learning</td>
<td>0.7</td>
<td>-10.7</td>
<td>0.0</td>
<td>1.0</td>
<td>-1.8</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Table 13: ∆Margin for different distillation methods on SuperGLUE dev tasks with OPT model.