Less is More: Task-aware Layer-wise Distillation for Language Model Compression

Chen Liang [1] **Simiao Zuo** [2] **Qingru Zhang** [1] **Pengcheng He** [2] **Weizhu Chen** [2] **Tuo Zhao** [1]

Abstract

Layer-wise distillation is a powerful tool to compress large models (i.e. teacher models) into small ones (i.e., student models). The student distills knowledge from the teacher by mimicking the hidden representations of the teacher at every intermediate layer. However, layer-wise distillation is difficult. Since the student has a smaller model capacity than the teacher, it is often under-fitted. Furthermore, the hidden representations of the teacher contain redundant information that the student does not necessarily need for the target task's learning. To address these challenges, we propose a novel Task-aware layEr-wise Distillation (TED). TED designs task-aware filters to align the hidden representations of the student and the teacher at each layer. The filters select the knowledge that is useful for the target task from the hidden representations. As such, TED reduces the knowledge gap between the two models and helps the student to fit better on the target task. We evaluate TED in two scenarios: continual pre-training and fine-tuning. TED demonstrates significant and consistent improvements over existing distillation methods in both scenarios. Code is available at https://github.com/cliang1453/task-aware-distillation.

1. Introduction

Large pre-trained language models have achieved state-of-the-art performances in many natural language processing tasks (Wang et al., 2019; Rajpurkar et al., 2016a). However, their deployment in resource-limited scenarios is hindered by their huge number of parameters (Raffel et al., 2019; Radford et al., 2019; Brown et al., 2020; He et al., 2020; 2023). Knowledge Distillation (KD) (Hinton et al., 2015) is a powerful tool to compress large models (i.e., teacher models) into small ones (i.e., student models) with a minimal loss of performance. This approach trains the student to match the output predictions of the teacher.

However, such a last-layer-only distillation approach does not exploit the intermediate layers of the teacher, which contain rich semantic and syntactic knowledge. To leverage such knowledge, researchers have proposed a *layer-wise distillation* approach, which trains the student to match the hidden representation of the teacher at each layer (Sun et al., 2019; Jiao et al., 2020; Sun et al., 2020b; Hou et al., 2020; Zuo et al., 2022). Such an approach often improves the generalization performance of the student model.

Nevertheless, layer-wise distillation faces two major challenges. First, the student may struggle to mimic the hidden representations of the teacher due to their large capacity gap. This often leads to large discrepancies between their hidden representations. Consequently, model training/optimization often favors reducing such large discrepancies over the training loss of the student (i.e., the target task's loss such as cross-entropy), resulting in an under-fitted student model. Second, mimicking the hidden representations may not be beneficial for the target task's learning. This is because the hidden representations of the teacher often contain redundant information (Dalvi et al., 2020; Durrani et al., 2020). Given the limited capacity of the student, such redundant information may compete with the useful information for distillation, hindering the useful knowledge from being distilled. Our empirical observations show that for some tasks, layer-wise distillation only marginally outperforms standard KD (Table 2).

To address these challenges, we propose a novel layer-wise distillation method, TED (**T**ask-aware lay**E**r-wise **D**istillation), which distills task-specific knowledge from the teacher to the student. We design a pair of task-aware filters for each layer of the teacher and student[1]. Each filter

[1]H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, U.S.A. [2]Microsoft, Redmond, U.S.A.. Correspondence to: Chen Liang <cliang73@gatech.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

[1]For simplicity, we assume that the student and teacher are of the same depth (number of layers) but different widths. The case of different depths will be elaborated in Section 2.

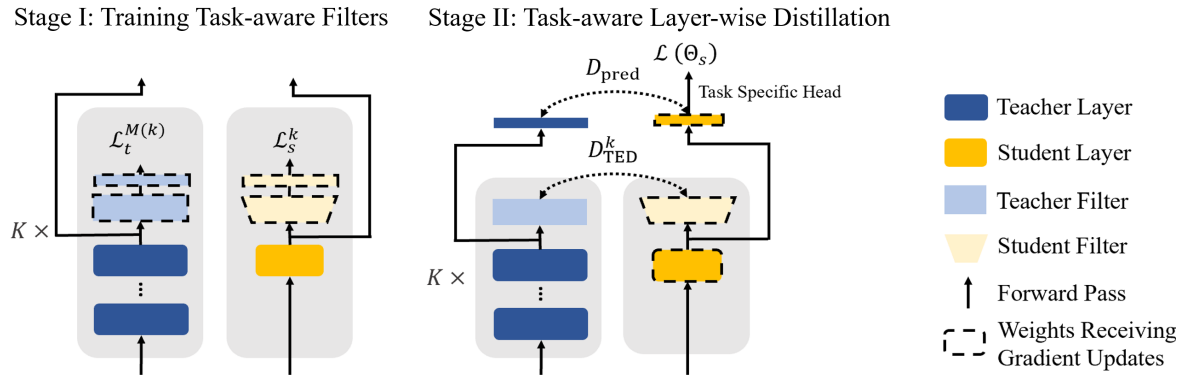

Figure 1. An illustration of TED's two-stage training framework. In *Stage I* (left), we fix the model parameters and only train the filters and task-specific heads based on the target task loss. In *Stage II* (right), we jointly train the student and its filters by aligning the filter outputs of each pair of the teacher and the student layers.

is a neural network with a task-specific head (e.g., a linear soft-max layer for classification), and is trained to extract the predictive knowledge from the hidden representation of the corresponding model. Figure 1 illustrates the training procedure of TED, which consists of two stages:

• Stage I: We train the task-aware filters for both the teacher and the student models, while keeping the model parameters frozen. At each layer, the filter takes the hidden representation of the model as input, and produces a target task's loss (e.g., cross-entropy) as output. The filter is subsequently optimized based on such a loss to capture the predictive knowledge from the hidden representation.

• Stage II: We jointly train the student model and its task-aware filters, while keeping the teacher and its filters fixed. At each layer, we feed the hidden representation of the teacher and the student to their respective filters (without the task-specific heads). Then, we adopt a regularizer that penalizes the discrepancy between the filtered representations. This regularizer encourages the student to learn the task-specific knowledge from the teacher, while ignoring the redundant information.

The task-aware filters serve as a selection mechanism that reduces the knowledge gap between the teacher and the student and encourages the distillation of task-specific knowledge. This makes distillation easier for the student.

We evaluate TED on two settings: continual pre-training and task-specific fine-tuning. In the continual pre-training setting, we distill a 6-layer GPT-2 student model (82M) from a 12-layer GPT-2 teacher model (125M) (Radford et al., 2019). We show that TED outperforms existing methods in both zero-shot and transfer learning settings on various downstream tasks (Paperno et al., 2016; Merity et al., 2017). In the task-specific fine-tuning setting, we distill a DeBERTaV3-xsmall student model (70M) from a DeBERTaV3-base teacher model (183M) (He et al., 2023). We demonstrate that TED achieves significant improvement on the GLUE benchmark (Wang et al., 2019) and the SQuAD v1.1/2.0 question answering datasets (Rajpurkar et al., 2016a; 2018).

The rest of the paper is organized as follows: Section 2 briefly reviews the background; Section 3 presents our proposed method; Section 4 presents experiments on language modeling; Section 5 presents experiments on natural language understanding; Section 6 presents analysis of models; and Section 8 discusses and concludes the paper.

2. Background

Transformer-based Language Models. The Transformer architecture is a powerful neural network design for modeling sequential data, such as natural language (Vaswani et al., 2017; Devlin et al., 2019; Radford et al., 2019; He et al., 2023). It consists of multiple layers that are stacked on top of each other. Each layer performs two operations: a multi-head self-attention mechanism and a two-layer feed-forward neural network. We use $f(\cdot;\Theta)$ to denote a Transformer-based model f that has a set of parameters Θ, where f takes an input sequence x from the input sample space $\mathcal{X}$ and produces an output prediction. We define the loss function $\mathcal{L}(\Theta) = \mathbb{E}_{x\sim\mathcal{X}}[\ell(f(x;\Theta))]$, where ℓ is the target task loss. For example, ℓ is the causal language modeling loss for generative models (i.e., $\sum_{t=1}^{|x|} \log p(x_t|x_{<t};\Theta)$).

Knowledge Distillation is a powerful approach to compress large models (i.e., teacher models) into smaller models (i.e., student models) by transferring knowledge from the former to the latter (Hinton et al., 2015). The student is trained to mimic the output predictions of the teacher. Specifically, we denote the teacher as $f_t(\Theta_t)$ and the student as $f_s(\Theta_s)$ and

consider the following optimization problem:

$$\min_{\Theta_s} \mathcal{L}(\Theta_s) + \mathcal{D}_{\text{pred}}(\Theta_t, \Theta_s), \tag{1}$$

where $\mathcal{D}_{\text{pred}}(\Theta_t, \Theta_s)$ is the distillation loss, a distance metric between the output predictions of the teacher and the student. For example, $\mathcal{D}_{\text{pred}}$ can be the KL-divergence: $\texttt{KL}(f_t(\Theta_t)/T, f_s(\Theta_s)/T)$, where $T > 0$ is the temperature that controls the softness of the prediction probability distributions (Hinton et al., 2015). A commonly adopted distillation scheme is the offline distillation, where the teacher is fully-trained and fixed, and the student is optimized based on Eq. 1.

Layer-wise Distillation. In large Transformer-based models, the output predictions of the models may not capture all the semantic and syntactic knowledge encoded in the intermediate layers. Therefore, researchers propose a layer-wise distillation approach, which aligns the hidden representations of the student and the teacher at each layer (Romero et al., 2015; Sun et al., 2019; 2020b; Jiao et al., 2020; Hou et al., 2020; Zuo et al., 2022; Liang et al., 2023). Specifically, we denote the hidden representation at the k-th layer of a K-layer student as $H_s^k \in \mathbb{R}^{|x| \times d_s}$, and at the $M(k)$-th layer of the teacher as $H_t^{M(k)} \in \mathbb{R}^{|x| \times d_t}$. Here $|x|$ is the sequence length; d_s and d_t are the hidden dimensions of the student and the teacher, respectively. $M(\cdot)$ is a layer mapping function that determines from which layer in the teacher that a student layer should distill. For example, if we set $M(k) = 2k$, the student would distill from every other layer in the teacher. The layer-wise distillation loss is defined as:

$$\mathcal{D}_{\text{layer}}(\Theta_t, [\Theta_s, \mathcal{W}_s]) = \sum_{k=1}^{K} \texttt{MSE}(H_t^{M(k)}, H_s^k W_s^k). \tag{2}$$

Here $\texttt{MSE}(\cdot, \cdot)$ is the mean-squared error, $W_s^k \in \mathbb{R}^{d_s \times d_t}$ is a randomly initialized and learnable linear projection that projects H_s^k into the same space as $H_t^{M(k)}$, and $\mathcal{W}_s = \{W_s^k\}_{k=1}^K$. In practice, the student is often optimized using multiple distillation losses, e.g.,

$$\min_{\Theta_s, \mathcal{W}_s} \mathcal{L}(\Theta_s) + \alpha_1 \mathcal{D}_{\text{pred}}(\Theta_t, \Theta_s) + \alpha_2 \mathcal{D}_{\text{layer}}(\Theta_t, [\Theta_s, \mathcal{W}_s]). \tag{3}$$

where $\alpha_1, \alpha_2 \geq 0$ are hyper-parameters. Besides the intermediate layers, distilling knowledge from the attention scores and the embedding layers can also improve the distillation performance (Sun et al., 2020b; Jiao et al., 2020; Wang et al., 2020; 2021). Eq. 3 can be further extended by adding such losses.

3. Method

We introduce TED, a two-stage training framework that uses task-aware filters to distill knowledge from a teacher to a student. The task-aware filters are neural networks that learn to extract task-specific knowledge from the hidden representations of the teacher and the student. In the first stage, we add a task-aware filter to each layer of the teacher and the student. We train these filters using the task-specific loss while keeping the model parameters frozen. In the second stage, we fine-tune the student and its filters by minimizing the discrepancy between the filtered representations of the teacher and the student.

3.1. Stage I: Training Task-aware Filters

For a student that contains K layers, we select K corresponding layers from the teacher to match with the student using a layer mapping function, $M(\cdot)$, as defined in Section 2. We then equip each layer with a task-aware filter to extract the task-specific knowledge from the hidden representation of this layer. Each filter is a neural network with a task-specific head (e.g., a linear soft-max layer for classification). It takes in the hidden representation generated by this layer and outputs a prediction for the target task. For example, for a classification task, the filter outputs a probability distribution over the classes.

For simplicity, we only specify how to train task-aware filters for the teacher. The student is treated similarly (see Section 4 for details). To train the task-aware filters, we fix the parameters of the teacher, which is already pre-trained [2]. In other words, we only update the parameters of the filters. We denote the task-aware filter at the $M(k)$-th layer as $g_t^k(\cdot; W_t^k)$, where W_t^k is the filter's parameters. The filter takes in the hidden representation $H_t^{M(k)}$ at the $M(k)$-th layer, and outputs a task-specific loss

$$\mathcal{L}_t^k(\Theta_t^{M(k)}, W_t^k) = \mathbb{E}_{x \sim \mathcal{X}}[\ell(g_t^k(H_t^{M(k)}; W_t^k))], \tag{4}$$

where $\Theta_t^{M(k)}$ is the teacher's parameters up to the $M(k)$-th layer. The loss function ℓ depends on the task and the setting. For example, ℓ is the causal language modeling loss for continual pre-training and the cross-entropy loss for fine-tuning of classification tasks. Given the loss in Eq. 4, we train the K filters jointly:

$$\min_{\mathcal{W}_t} \sum_{k=1}^{K} \mathcal{L}_t^k(\Theta_t^{M(k)}, W_t^k), \tag{5}$$

where $\mathcal{W}_t = \{W_t^k\}_{k=1}^K$. By training the task-aware filters, we can reduce the redundant information in the hidden representations, and keep the information that is useful for learning the target task.

Remark 3.1. We can choose different neural network architectures to implement the task-aware filters, such as a simple

[2] We discuss in detail how to initialize the teacher and the student models in Section 4 and 5.

linear projection that maps the input to a lower-dimensional space, a multi-layer perceptron that applies a sequence of nonlinear transformations, or a stack of Transformer layers that encode the input with attention mechanism. We compare the performances of these architectures in Section 6.5.

3.2. Stage II: task-aware Layer-wise Distillation

In Stage II, we remove the task-specific heads in the task-aware filters, which are learned in Stage I. Then, we freeze the parameters of the teacher and its filters, and fine-tune the student and its filters by minimizing the discrepancy between the filtered representations at each layer of the two models.

Formally, we denote $g_s^k(\cdot, W_s^k)$ as the task-aware filters at the k-th layer of the student. Then the task-aware layer-wise distillation loss is defined as

$$\mathcal{D}_{\text{TED}}\left([\Theta_t, \mathcal{W}_t], [\Theta_s, \mathcal{W}_s]\right) = \sum_{k=1}^{K} \texttt{MSE}\left(g_t^k(H_t^{M(k)}; W_t^k), g_s^k(H_s^k; W_s^k)\right), \quad (6)$$

which measures the discrepancy between the filtered representations of the teacher and the student. Based on the distillation loss, the training objective for the student and its filters is

$$\min_{\Theta_s, \mathcal{W}_s} \mathcal{L}(\Theta_s) + \alpha_1 \mathcal{D}_{\text{pred}}(\Theta_t, \Theta_s) + \alpha_2 \mathcal{D}_{\text{TED}}([\Theta_t, \mathcal{W}_t], [\Theta_s, \mathcal{W}_s]), \quad (7)$$

where $\mathcal{L}$ is the target task's loss and $\mathcal{D}_{\text{pred}}$ is the prediction distillation loss defined in Eq 1 and $\alpha_1, \alpha_2 \geq 0$ are hyper-parameters. By using the task-aware filters, Eq. 7 imposes an easier requirement on the student than the conventional layer-wise distillation loss (Eq. 3). That is, Eq. 3 requires the student to match the teacher on the unfiltered hidden representations, regardless of their relevance to the target task.

Remark 3.2. We can also keep the task-specific heads in the task-aware filters and penalize the KL-divergence instead of the mean-squared error. We compare the performances of these two variants in Section 6.6.

4. Language Modeling

4.1. Data

First, we evaluate TED in the continual pre-training setting by distilling generative models on language modeling tasks. We use **Open WebText**[3] (Gokaslan et al., 2019), an open-source replication of the OpenAI WebText corpus (Radford et al., 2019) for open domain pre-training. It is a massive English corpus containing 8M training documents and around 38GB of texts extracted from 45M links of Reddit post urls. Data pre-processing details are deferred to Appendix A.1.1.

Second, we evaluate the distilled student model by conducting zero-shot and transfer learning experiments on two downstream tasks: **LAMBADA** (Paperno et al., 2016) and **WikiText-103** (Merity et al., 2017). LAMBADA evaluates the ability of language models in modeling long-range dependencies. The dataset consists of full texts of 2662 novels extracted from BookCorpus (Zhu et al., 2015). WikiText-103 is a collection of over 100M tokens extracted from the set of verified good and featured articles on Wikipedia.

4.2. Models

Teacher Model. We use a pre-trained GPT-2 (Radford et al., 2019) as the teacher model. GPT-2 is a Transformer-based model trained on Open WebText using a causal language modeling objective. We adopt the base version of GPT-2 (GPT-2_{12}, 125M parameters), which contains 12 layers and has a hidden dimension of $d_t = 768$.

Student Model. We initialize a 6-layer (i.e., $K = 6$) student model (GPT-2_6, 82M parameters) with a subset of layers from the teacher. We adopt the layer mapping function $M(k) = 2k - 1$ for $k \leq K/2$ and $M(k) = 2k$ for $k > K/2$ following Sanh et al. 2019. We further discuss how to initialize the student model when its architecture is not a shallow version of the teacher in Appendix A.4.

4.3. Training

Stage I. For the teacher model, we design each filter as a linear projection, i.e., $W_t^k \in \mathbb{R}^{d_t \times d_t}$, and randomly initialize a filter for each layer that is selected to match with a student layer. We fix the parameters of the teacher model and train the filters based on Eq 5 for one epoch. We use AdamW (Loshchilov & Hutter, 2019) as the optimizer and use 4k tokens as the batch size. We adopt a linear decay learning rate schedule with a learning rate of 2.5×10^{-4} and a warmup ratio of 0.05. Then, we directly take the trained filter at the $M(k)$-th layer of the teacher to be the filter at the k-th layer of the student without further training. It is intuitive that the trained filters of the teacher can serve as sufficiently good filters of the student because the student is initialized with a subset of layers from the teacher. Full implementation details are deferred to Appendix A.1.2.

Stage II. We train the student and its filters based on Eq 7 for four epochs. We follow the same hyper-parameter configurations as in Stage I, and set $\alpha_1 = 2.5$, $\alpha_2 = 0.1$ and temperature $T = 2.0$.

Baselines. We consider two baseline methods: 1) **KD** optimizes the student model based on $\mathcal{L}(\Theta_s) + \alpha_1 \mathcal{D}_{\text{pred}}(\Theta_t, \Theta_s)$ (Eq. 1), which is adopted by DistilGPT-2_6 (Sanh et al., 2019). 2) **LWD** optimizes the student model based on $\mathcal{L}(\Theta_s) +$

[3] https://huggingface.co/datasets/openwebtext

Table 1. Zero-shot and transfer learning performance of GPT-2_6 models on test sets. We report the results of DistilGPT-2 from Sanh et al. (2019), and the results of GPT-2_{12} from Radford et al. (2019). Other results are from our own implementation.

Method	Test	Zero-Shot			Transfer Learning		
	Open WebText	WikiText-103	LAMBADA		WikiText-103	LAMBADA	
	ppl↓	ppl↓	ppl↓	Acc↑	ppl↓	ppl↓	Acc↑
GPT-2_{12} (Teacher)	23.1	37.5	35.1	46.0	15.9	37.2	34.8
DistilGPT-2_6 (KD)	31.9	-	-	-	21.1	-	-
DistilGPT-2_6 (KD, Re-Imp)	29.1	49.0	87.9	22.9	19.3	50.1	31.7
GPT-2_6 (LWD)	29.7	51.9	91.9	22.0	19.3	50.6	31.5
GPT-2_6 (TED)	**28.5**	**48.1**	**87.2**	**23.0**	**19.0**	**48.6**	**32.1**

$\alpha_1 \mathcal{D}_{\text{pred}}(\Theta_t, \Theta_s) + \alpha_2 \mathcal{D}_{\text{layer}}(\Theta_t, [\Theta_s, \mathcal{W}_s])$ (Eq. 3).

4.4. Main Results

Table 1 shows the zero-shot and transfer learning performance of the GPT-2_6 models. For Open WebText, we split 5% for testing. For the zero-shot setting, we directly evaluate the student model on the test sets. For the transfer learning setting, we fine-tune the student model on the downstream language modeling tasks. We have the following observations: 1) LWD does not always lead to a better performance than KD, suggesting that the student may have difficulty mimicking the teacher at every layer. 2) TED can significantly improve model performance, especially on Open WebText. This suggests that distilling the task-specific knowledge to the student yields a better model.

5. Natural Language Understanding

5.1. Data

We further evaluate TED on natural language understanding (NLU) tasks. We consider the widely used General Language Understanding Evaluation (**GLUE**, Wang et al. 2019) benchmark, which contains nine NLU tasks, including textual entailment, sentiment analysis and text similarity. We also evaluate TED on **SQuAD v1.1/2.0** (Rajpurkar et al., 2016a; 2018), which are widely used question answering datasets. Details about the datasets are deferred to Appendix A.2.1.

5.2. Models

We use DeBERTaV3 models (He et al., 2023) as the teacher and student models. DeBERTaV3 is pre-trained in an ELECTRA-style (Clark et al., 2020) on 160GB open-domain corpus (Gokaslan et al., 2019; Trinh & Le, 2018; Nagel, 2016). It improves BERT (Devlin et al., 2019) with disentangled attention and enhanced mask decoder, and achieves the state-of-the-art downstream performance.

Teacher Model. We initialize the teacher model for each task with a DeBERTaV3-base model that has been fine-tuned on the target task. The model has 183M parameters, 12 layers and a hidden dimension of 768 (i.e., $d_t = 768$). We fine-tune the model using AdamW as the optimizer. We adopts a linear decay learning rate schedule with a warmup ratio in $\{0.05, 0.1\}$. We choose the learning rate from $\{1, 1.5, 2, 2.5, 3\} \times 10^{-5}$, the batch size from $\{16, 32, 64\}$, the number of training epochs from $\{3, 6, 8\}$ and the dropout ratio from $\{0.05, 0.1\}$. Full implementations details are deferred to Appendix A.2.2.

Student Model. We initialize the student model for each task with a DeBERTaV3-xsmall model that has been fine-tuned on the target task. The model contains a total 70M parameters (including 22M backbone parameters). It has 12 layers and a hidden dimension of 384 (i.e., $d_s = 384$). We further discuss how to initialize the student model when there does not exist a pre-trained or fine-tuned model with the desired architecture in Appendix A.4.

5.3. Training

Stage I. Since the teacher and student have different hidden dimensions, we set $W_t^k \in \mathbb{R}^{d_t \times d_t}$ and $W_s^k \in \mathbb{R}^{d_s \times d_t}$. We randomly initialize a filter for each layer of the student and the teacher (recall that they have the same number of layers). We freeze the model parameters of the teacher and the student, and train their filters on the target task following the same hyper-parameter configurations in Section 5.2. Full implementations details are deferred to Appendix A.2.3.

Stage II. We then train the student and its filters based on Eq 7 on the target task. We follow the same hyper-parameter configurations as in Stage I. We choose $\alpha_1 \in \{1.0, 2.5, 5.0, 10.0\}$, choose $\alpha_2 \in \{10, 20, 50, 100, 200, 500, 1000\}$, and set the temperature $T = 2.0$.

Table 2. Evaluation results on GLUE dev set. The teacher is a fine-tuned DeBERTaV3-base model (183M) and the student is a DeBERTaV3-xsmall model (70M). Results of "Fine-tune" are obtained by directly fine-tuning the DeBERTaV3-xsmall model on the target task without distillation.

Method	MNLI-m/mm Acc	QQP Acc/F1	QNLI Acc	SST-2 Acc	RTE Acc	CoLA Mcc	MRPC Acc/F1	STSB P/S	Avg. Score
$\text{Teacher}_{\text{base}}$	90.5/90.6	92.3/89.7	94.2	96.0	86.1	68.8	90.8/93.5	92.4/92.2	88.9
$\text{Fine-tune}_{\text{xs}}$	88.3/88.1	91.7/88.8	92.5	93.5	79.7	68.3	90.2/93.0	90.9/90.5	86.9
KD_{xs}	88.5/88.1	91.7/88.8	92.9	93.9	80.5	66.3	**91.2/93.7**	91.0/90.8	87.0
LWD_{xs}	**88.8**/88.3	91.8/89.0	92.9	93.9	80.2	66.8	90.2/93.0	91.0/90.6	87.0
TED_{xs}	**88.8/88.7**	**92.2/89.5**	**93.1**	**94.2**	**81.8**	**68.5**	90.4/93.2	**91.3/91.1**	**87.5**

Table 3. Evaluation results on SQuAD v1.1 and SQuAD v2.0 validation sets. The teacher is a fine-tuned DeBERTaV3-base model and the student is a DeBERTaV3-xsmall model.

Method	SQuAD v1.1 EM/F1	SQuAD v2.0 EM/F1	Avg. Score
$\text{Teacher}_{\text{base}}$	87.1/93.1	85.4/88.4	90.8
$\text{Fine-tune}_{\text{xs}}$	83.5/90.4	82.0/84.8	87.6
KD_{xs}	84.8/91.4	82.6/85.5	88.5
LWD_{xs}	84.9/91.5	82.8/85.6	88.6
TED_{xs}	**85.4/91.7**	**83.0/85.8**	**88.8**

5.4. Main Results

Table 2 and Table 3 show the evaluation results of the student on the GLUE benchmark and SQuAD v1.1/2.0 datasets, respectively. TED achieves consistent and significant gains over nine out of ten tasks over the best distillation baseline. For example, TED achieves a gain of 0.5 on some large datasets, e.g., QQP, and a gain of 1.0 on some small datasets, e.g., RTE. For certain small datasets (e.g., RTE, MRPC, STS-B), LWD does not always produce a better performance than KD. In contrast, TED improves upon KD in two out of three cases.

5.5. BERT Experiments

To compare with the state-of-the-art task-specific distillation baselines, we present in Table 4 the results of a 6-layer BERT-base student model (BERT-base_6, 66M) distilled from a fine-tuned 12-layer BERT-base teacher model (BERT-base_{12}, 109M). TED achieves comparable performance with noticeable benefits over the existing baselines on three NLU tasks. All implementation details are deferred to Appendix A.2.4.

Table 4. Evaluation results on GLUE dev set. The teacher is a fine-tuned BERT-base_{12} (12 layers), and the student is a BERT-base_6 (6 layers), except for CoDIR, which uses a RoBERTa-base as the teacher. References: PKD (Sun et al., 2019), BERT-of-Thesus (Xu et al., 2020), MixKD (Liang et al., 2020), ProKT (Shi et al., 2021), CoDIR (Sun et al., 2020a).

Method	MNLI-m/mm	SST-2	RTE
Teacher_{12}	84.5/84.7	92.6	71.3
KD_6	82.1/82.3	90.8	65.4
LWD_6	82.7/83.1	90.9	67.6
PKD_6	81.3/-	91.3	66.5
Thesus_6	82.3/-	91.5	68.2
MixKD_6	82.5/-	92.1	67.9
ProKT_6	82.8/83.2	91.3	68.4
CoDIR_6	**83.6**/82.8	**93.6**	65.6
TED_6	83.4/**84.0**	91.7	**68.8**

6. Analysis

We further verify that the task-aware filters can capture the task-specific knowledge and ease distillation. All implementation details are deferred to Appendix A.3.

6.1. Filters Capture Task-Specific Knowledge

Table 5 shows the evaluation results of a student model trained on the target task with their task-aware filters replaced by the filters trained on a different task. If the filters

Table 5. Evaluation results on GLUE dev set. The teacher is fine-tuned DeBERTaV3-base model and the student is a DeBERTaV3-xsmall model.

Method	RTE	SST-2	MRPC	STS-B
LWD_{xs}	80.2	93.9	90.2	91.0
TED_{xs} (Filters Learned on MNLI)	80.5	93.4	90.0	91.2
TED_{xs} (Filters Learned on QNLI)	81.6	93.5	89.1	91.4
TED_{xs} (Filters Learned on SST-2)	79.7	**94.2**	90.2	90.8
TED_{xs} (Filters Learned on the Target Task)	**81.8**	**94.2**	**90.4**	**91.3**

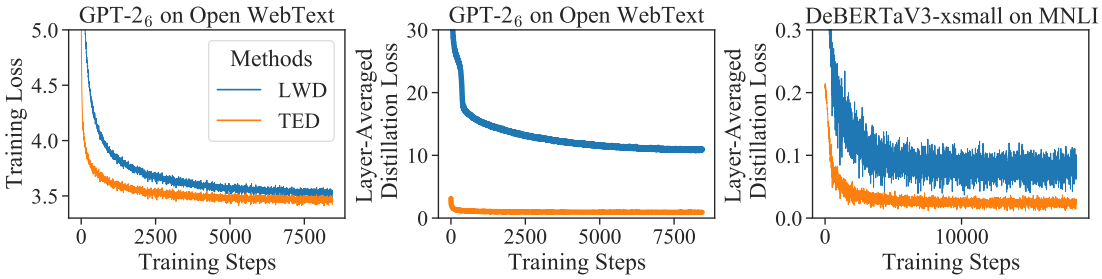

Figure 2. Left: Training loss of the GPT-2_6 student on the target task (i.e., language modeling) on Open WebText. *Middle and Right*: Distillation loss averaged by the number of layers of the GPT-2_6 student on Open WebText and the DeBERTaV3-xsmall student on MNLI, respectively.

Table 6. Evaluation results of different types of filter initialization. We evaluate the DeBERTaV3-xsmall student on the GLUE dev set and the GPT-2_6 student on the Open WebText test set. "N/A" is because we initialize $\mathcal{W}_s$ of GPT-2_6 from $\mathcal{W}_t$.

Method	$\mathcal{W}_t$ Trained on target task?	$\mathcal{W}_s$ Trained on target task?	Target Tasks MNLI-m/mm Acc↑	SST-2 Acc↑	RTE Acc↑	Open WebText ppl↓
Abl.1	✓	No $\mathcal{W}_s$	87.4/86.9	92.0	77.2	31.82
Abl.2	No $\mathcal{W}_t$	✗	88.7/88.5	93.5	79.5	29.74
Abl.3	✓	✗	88.7/88.6	93.9	79.9	29.66
Abl.4	No $\mathcal{W}_t$	✓	88.8/88.6	94.0	81.5	N/A
TED	✓	✓	**88.8/88.7**	**94.2**	**81.8**	**28.49**

are trained on the target task, TED shows consistent gains over LWD. In contrast, if the filters are trained on a different task, the gains become smaller and vary significantly across tasks, suggesting the task-aware filters can learn task-specific knowledge[4].

6.2. TED Alleviates Under-fitting and Eases Distillation

Figure 2 (*Left*) shows the training loss of the student on the target task (i.e., language modeling) during distillation. TED leads to a faster convergence and a lower training loss than LWD, which suggests that TED improves the fitting of the student on the target task. Figure 2 (*Middle* and *Right*) shows the distillation loss averaged by the number of layers. The distillation loss in TED has a smaller magnitude and a lower variance than LWD. This suggests that TED effectively eases the distillation.

6.3. Contribution of the Filters

To investigate the contribution of the filters, we initialize the filters with trained weights (✓), randomly initialized weights (✗), or no filters at all (No $\mathcal{W}_t$/$\mathcal{W}_s$). Table 6 shows that: 1) Using trained filters for the teacher significantly improves the distillation performance, as long as the student has a

[4]The cause of high variance could be that the filters trained on tasks that are more similar to the target task perform better. For example, on the RTE task, filters trained on MNLI and QNLI perform better than those trained on SST-2, likely because the task-relevant knowledge can be transferred across NLI tasks.

Table 7. Evaluation results on GLUE dev set. The teacher is DeBERTaV3-large (435M), and the student is DeBERTaV3-xsmall (70M).

Method	MNLI-m/mm	SST-2	RTE
TED_{xs} ($\text{Teacher}_{\text{base}}$)	**88.8**/88.7	94.2	**81.8**
$\text{Teacher}_{\text{large}}$	91.7/91.8	96.3	91.4
$\text{Fine-tune}_{\text{xs}}$ ($\text{Teacher}_{\text{large}}$)	88.3/88.1	93.5	79.7
KD_{xs} ($\text{Teacher}_{\text{large}}$)	88.4/88.4	94.4	77.9
LWD_{xs} ($\text{Teacher}_{\text{large}}$)	88.5/88.5	93.6	79.4
TED_{xs} ($\text{Teacher}_{\text{large}}$)	**88.8/88.8**	**94.6**	81.4

Table 8. Evaluation results of the GPT-2_6 student using different filter architectures.

Filter Architectures	Zero-Shot			Transfer Learning		
	WikiText-103	LAMBADA		WikiText-103	LAMBADA	
	ppl↓	ppl↓	Acc↑	ppl↓	ppl↓	Acc↑
Linear Projection	48.12	87.27	22.99	19.03	48.63	32.08
Two-layer MLP	**47.78**	**86.88**	**23.09**	19.03	48.61	32.08
One Subsequent Layer	48.05	87.01	23.02	**19.02**	48.50	32.08
All Subsequent Layers	48.13	87.79	22.86	19.17	**48.48**	**32.04**

set of filters that can learn to match the teacher's filtered output. In other words, TED can be still beneficial even if the student filters are not trained in Stage I but randomly initialized for Stage II. 2) If the student filters are initialized from trained weights instead of randomly initialized, the distillation performance can be further improved.

6.4. TED Alleviates the Capacity Gap Issue

Table 7 shows the performance of a DeBERTaV3-xsmall student (70M) distilled from a 6 times larger DeBERTaV3-large model (435M), which contains 24 layers and has a hidden dimension of 1024. All implementation details are deferred to Appendix A.3.1. When using this large teacher, LWD performs worse, e.g., the student achieves only 79.4 in RTE while achieving 80.2 when using the DeBERTaV3-base teacher (Table 2). In contrast, TED maintains a comparable performance, e.g., the student achieves a 0.4 points of gain on SST-2.

6.5. Filter Architectures

Table 8 shows the student performance when the filters are initialized with different architectures: 1) a linear projection; 2) a two-layer perceptron with GeLU as non-linearity (Hendrycks & Gimpel, 2016); 3) layer(s) initialized from the weights of the subsequent layer(s), e.g., for the first layer of the model, we use the second (or from second to last) layer(s) as the filter. By introducing non-linearity in the filters, the zero-shot performance slightly improves while the transfer learning performance remains insensitive. Further increasing the filter complexity exhibits little benefits.

6.6. Design of Distillation Loss

If we keep the task specific head of each filter trained in Stage I and bring it to Stage II, then the filtered output would be a prediction probability distribution instead of a hidden representation. Then we can substitute the `MSE` between the two hidden representations in Eq. 6 with the KL-divergence between the two probability distributions. Table 9 shows that such an approach also shows noticeable improvements over the baselines. This suggests that the prediction probability can also preserve some task-specific knowledge.

Table 9. Evaluation results under different designs of the task-aware distillation loss. The teacher is DeBERTaV3-base and the student is DeBERTaV3-xsmall.

Method	MNLI m/mm	SQuAD 2.0 EM/F1
KD	88.5/88.1	81.0/84.2
LWD	**88.8**/88.3	81.5/84.4
TED (`KL`)	88.6/88.7	81.9/84.7
TED (`MSE`)	**88.8/88.7**	**82.0/84.9**

6.7. Hyper-parameter Study

We further investigate whether TED is sensitive to α_2, the hyper-parameter that controls the strength of $\mathcal{D}_{\text{TED}}$. Fig-

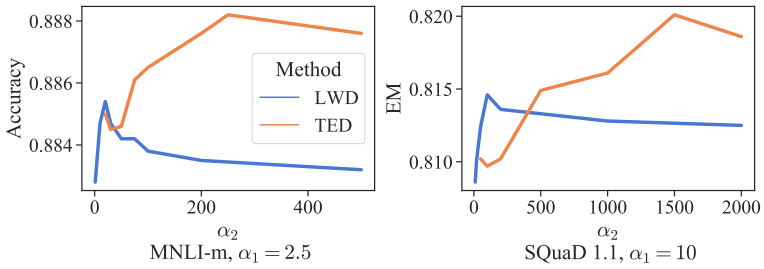

Figure 3. Evaluation performance of DeBERTaV3-xsmall under different values of α_2.

ure 3 shows the performance of the DeBERTaV3-xsmall student on MNLI-m and SQuAD v1.0 under different values of α_2. TED shows consistent gains over a wide range of values of α_2.

7. Discussion

Computational Costs of TED. In the training phase, TED incurs an additional computational overhead beyond what is required by layer-wise distillation (LWD). This is due to the training of task-aware filters in Stage I. However, this overhead is relatively moderate, accounting for approximately 10% of the computational cost of LWD. This is because the number of filter parameters is around 2%-4% of the model parameters, and the training of the filters does not require any back-propagation on the model parameters. Despite the overhead during training, TED retains the same inference speed as LWD during the model deployment phase. This is because all filters are discarded at this stage, with only model parameters being utilized for inference.

Exploring Task-aware Distillation in Multi-task Setting. We design TED for *task-specific distillation*, where a task-specific student is trained by distilling knowledge from a target task. However, task-specific distillation exhibits several practical limitations: 1) It lacks scalability as one needs to distill a new student for every new task. 2) Certain tasks have zero or few training samples, making them unsuitable for distillation. To resolve these limitations, one potential direction is to explore the idea of task-aware distillation in the *multi-task setting* (Sanh et al., 2021; Longpre et al., 2023). In this setting, one can leverage knowledge from hundreds of tasks to distill a single multi-task student which generalizes well on various seen and unseen tasks. One possible strategy, for example, is to design multiple filters, each serving as an expert specialized in extracting knowledge from a group of relevant tasks. During the distillation process, each input sample can be routed to its most task-relevant filter(s).

Filtering and Distilling Knowledge from Large Language Models. Pre-trained large language models (LLMs), with up to hundreds of billions of parameters, have demonstrated remarkable generalizability on a wide range of tasks. How to effectively transfer the knowledge from these powerful LLMs into smaller models has therefore become an area of research interest (Hsieh et al., 2023; Jiang et al., 2023; Taori et al., 2023; Peng et al., 2023). However, it is challenging to directly apply TED, an layer-wise distillation (LWD) approach, to LLMs due to two primary reasons: 1) LWD requires the access to the layer-wise hidden representations, making it incompatible with models that are closed-source. 2) The computational cost of LWD scales with the model depth and hidden dimension, which is prohibitively expensive in such models. Yet, the underlying idea of selecting and transferring task-relevant knowledge could be useful for LLM distillation, particularly given the significant teacher-student capacity gap. For example, a possible strategy is to use a LLM teacher to generate task-relevant input and output samples in a controllable manner, and then use these samples to distill the student model.

8. Conclusion

Layer-wise distillation is challenging as the student may struggle to mimic the hidden representations of a much larger teacher. We propose TED, which first learns task-aware filters to extract the task-specific knowledge from the teacher and the student, then minimizes the discrepancy between the filtered outputs. This encourages the student to learn filtered knowledge, which contains more task-relevant signals. Our experiments verify that the filters can effectively capture task-specific knowledge and ease layer-wise distillation.

References

Bar-Haim, R., Dagan, I., Dolan, B., Ferro, L., and Giampiccolo, D. The second PASCAL recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, 2006.

Bentivogli, L., Dagan, I., Dang, H. T., Giampiccolo, D., and Magnini, B. The fifth pascal recognizing textual

entailment challenge. In *In Proc Text Analysis Conference (TAC'09)*, 2009.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., and Specia, L. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pp. 1–14, Vancouver, Canada, 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2001.

Clark, K., Luong, M., Le, Q. V., and Manning, C. D. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Dagan, I., Glickman, O., and Magnini, B. The pascal recognising textual entailment challenge. In *Proceedings of the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*, MLCW'05, pp. 177–190, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-33427-0, 978-3-540-33427-9. doi: 10.1007/11736790_9.

Dalvi, F., Sajjad, H., Durrani, N., and Belinkov, Y. Analyzing redundancy in pretrained transformer models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4908–4926, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.398.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423.

Dolan, W. B. and Brockett, C. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

Durrani, N., Sajjad, H., Dalvi, F., and Belinkov, Y. Analyzing individual neurons in pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4865–4880, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.395.

Giampiccolo, D., Magnini, B., Dagan, I., and Dolan, B. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pp. 1–9, Prague, 2007. Association for Computational Linguistics.

Gokaslan, A., Cohen, V., Pavlick, E., and Tellex, S. Openwebtext corpus, 2019.

He, P., Liu, X., Gao, J., and Chen, W. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

He, P., Gao, J., and Chen, W. DeBERTav3: Improving deBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=sE7-XhLxHA.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

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

Hou, L., Huang, Z., Shang, L., Jiang, X., Chen, X., and Liu, Q. Dynabert: Dynamic BERT with adaptive width and depth. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Hsieh, C.-Y., Li, C.-L., Yeh, C.-K., Nakhost, H., Fujii, Y., Ratner, A., Krishna, R., Lee, C.-Y., and Pfister, T. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. *arXiv preprint arXiv:2305.02301*, 2023.

Jiang, Y., Chan, C., Chen, M., and Wang, W. Lion: Adversarial distillation of closed-source large language model. *arXiv preprint arXiv:2305.12870*, 2023.

Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F., and Liu, Q. TinyBERT: Distilling BERT for

natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4163–4174, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.372.

Liang, C., Jiang, H., Li, Z., Tang, X., Yin, B., and Zhao, T. Homodistil: Homotopic task-agnostic distillation of pretrained transformers. *arXiv preprint arXiv:2302.09632*, 2023.

Liang, K. J., Hao, W., Shen, D., Zhou, Y., Chen, W., Chen, C., and Carin, L. Mixkd: Towards efficient distillation of large-scale language models. *arXiv preprint arXiv:2011.00593*, 2020.

Longpre, S., Hou, L., Vu, T., Webson, A., Chung, H. W., Tay, Y., Zhou, D., Le, Q. V., Zoph, B., Wei, J., et al. The flan collection: Designing data and methods for effective instruction tuning. *arXiv preprint arXiv:2301.13688*, 2023.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Nagel, S. Cc-news. *URL: http://web.archive. org/save/http://commoncrawl. org/2016/10/newsdatasetavailable*, 2016.

Paperno, D., Kruszewski, G., Lazaridou, A., Pham, N. Q., Bernardi, R., Pezzelle, S., Baroni, M., Boleda, G., and Fernández, R. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534, Berlin, Germany, 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1144.

Peng, B., Li, C., He, P., Galley, M., and Gao, J. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*, 2023.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, 2016a. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, 2016b. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264.

Rajpurkar, P., Jia, R., and Liang, P. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 784–789, Melbourne, Australia, 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-2124.

Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. Fitnets: Hints for thin deep nets. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Sanh, V., Webson, A., Raffel, C., Bach, S. H., Sutawika, L., Alyafeai, Z., Chaffin, A., Stiegler, A., Scao, T. L., Raja, A., et al. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*, 2021.

Sennrich, R., Haddow, B., and Birch, A. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1715–1725, Berlin, Germany, 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162.

Shi, W., Song, Y., Zhou, H., Li, B., and Li, L. Follow your path: a progressive method for knowledge distillation. In *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part III 21*, pp. 596–611. Springer, 2021.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 1631–1642, Seattle, Washington, USA, 2013. Association for Computational Linguistics.

Sun, S., Cheng, Y., Gan, Z., and Liu, J. Patient knowledge distillation for BERT model compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4323–4332, Hong Kong, China, 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1441.

Sun, S., Gan, Z., Fang, Y., Cheng, Y., Wang, S., and Liu, J. Contrastive distillation on intermediate representations for language model compression. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 498–508, Online, 2020a. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.36.

Sun, Z., Yu, H., Song, X., Liu, R., Yang, Y., and Zhou, D. MobileBERT: a compact task-agnostic BERT for resource-limited devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2158–2170, Online, 2020b. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.195.

Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., and Hashimoto, T. B. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Trinh, T. H. and Le, Q. V. A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., and Zhou, M. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *arXiv preprint arXiv:2002.10957*, 2020.

Wang, W., Bao, H., Huang, S., Dong, L., and Wei, F. MiniLMv2: Multi-head self-attention relation distillation for compressing pretrained transformers. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 2140–2151, Online, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.188.

Warstadt, A., Singh, A., and Bowman, S. R. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019. doi: 10.1162/tacl_a_00290.

Williams, A., Nangia, N., and Bowman, S. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1112–1122, New Orleans, Louisiana, 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1101.

Xu, C., Zhou, W., Ge, T., Wei, F., and Zhou, M. BERT-of-theseus: Compressing BERT by progressive module replacing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7859–7869, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.633.

Zhu, Y., Kiros, R., Zemel, R. S., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pp. 19–27. IEEE Computer Society, 2015. doi: 10.1109/ICCV.2015.11.

Zuo, S., Zhang, Q., Liang, C., He, P., Zhao, T., and Chen, W. Moebert: from bert to mixture-of-experts via importance-guided adaptation. *arXiv preprint arXiv:2204.07675*, 2022.

A. Appendix

A.1. Language Modeling Experiment

A.1.1. Data

Open WebText is an open source effort to reproduce OpenAI's WebText dataset. The dataset is created by extracting Reddit post urls from the Reddit submissions dataset[5]. These links are then deduplicated, filtered to exclude non-html content, and shuffled randomly. Near-duplicate documents are identified using local-sensitivity hashing. They are hashed into sets of 5-grams and all documents that had a similarity threshold of greater than 0.5 were removed. All language modeling datasets were tokenized based on byte-level BPE (Sennrich et al., 2016) with a vocabulary size of 50257 (Radford et al., 2019). The max sequence length of the input training sample is 1024.

A.1.2. Training

Our implementation is based on Huggingface Transformers [6]. The GPT-2 base model consists 12 layers and has 12 attention heads in each attention module. The input and intermediate hidden dimension in the feed-forward network is 768 and 1024, respectively. We use mixed precision training and train on 8 80G Nvidia A100 GPUs. Detailed hyper-parameters are summarized in Table 10.

Table 10. Hyper-parameters for training GPT-2_6 on Open WebText.

Hyper-parameters	Stage I	Stage II
Dropout	0.1	0.1
Warmup Ratio	0.05	0.05
Learning Rates	0.00025	0.00025
Batch Size	4000	4000
Weight Decay	0	0
Training Epochs	1	4
Learning Rate Decay	Linear	Linear
Adam ϵ	1×10^{-6}	1×10^{-6}
Adam β_1	0.9	0.9
Adam β_2	0.98	0.98

A.2. Natural Language Understanding Experiment

A.2.1. Data

GLUE is a commonly used natural language understanding benchmark containing nine tasks. The benchmark includes question answering (Rajpurkar et al., 2016b), linguistic acceptability (CoLA, Warstadt et al. 2019), sentiment analysis (SST, Socher et al. 2013), text similarity (STS-B, Cer et al. 2017), paraphrase detection (MRPC, Dolan & Brockett 2005), and natural language inference (RTE & MNLI, Dagan et al. 2006; Bar-Haim et al. 2006; Giampiccolo et al. 2007; Bentivogli et al. 2009; Williams et al. 2018) tasks. Details of the GLUE benchmark, including tasks, statistics, and evaluation metrics, are summarized in Table 14.

SQuAD 1.1/2.0 is the Stanford Question Answering Dataset (SQuAD) v1.1 and v2.0 (Rajpurkar et al., 2018; 2016a), two popular machine reading comprehension benchmarks from approximately 500 Wikipedia articles with questions and answers obtained by crowdsourcing. The SQuAD v2.0 dataset includes unanswerable questions about the same paragraphs.

A.2.2. Model

We initialize the teacher for each target task as a DeBERTaV3-base model fine-tuned on the target task. We fine-tune the model by adding a target task classification head on top of the last layer. The detailed hyper-parameters are listed in Table 13. We initialize the student for each target task as a pre-trained DeBERTaV3-xsmall model.

[5] https://files.pushshift.io/reddit/submissions/

[6] https://github.com/huggingface/transformers/tree/v4.17.0

A.2.3. TRAINING

We follow the hyper-parameter configurations listed in Table 13 for both the Stage I and Stage II training. Our implementation is based on Huggingface Transformers. We use mixed precision training and train on 8 32G Nvidia V100 GPUs.

For Stage I, we empirically observe that if we first fine-tune the student on the target task, then train the filters on top of the fine-tuned student, the distillation performance would improve. We hypothesize that the student filters can learn to capture more task-relevant knowledge if the student is properly initialized on the target task. As a result, we also fine-tune the student model following the hyper-parameter configuration listed in Table 13 before Stage I. As shown in Table 11, initializing the student model with fine-tuned weights will not largely influence the final distillation performance.

Table 11. Performance comparison of initializing the student with fine-tuned and pre-trained weights.

Method	Θ_s Fine-tuned?	MNLI-m/mm Acc	QQP Acc	QNLI Acc	SST-2 Acc	RTE Acc	Avg Score
LWD	✗	88.8/88.3	91.8	92.9	93.9	80.2	89.5
Abl.	✓	88.7/88.5	92.0	92.8	93.5	79.5	89.3

A.2.4. BERT EXPERIMENTS

Model. We initialize the teacher model with a pre-trained 12-layer BERT-base model that has been fine-tuned on the target task (BERT-base$_{12}$). The teacher model contains 110M parameters and has a hidden dimension of $d_t = 768$. We initialize the student model with 6 selected layers from the fine-tuned teacher model (BERT-base$_6$). Specifically, we define the layer mapping function $M(k) = 2k - 1$ for $k \leq K/2$ and $M(k) = 2k$ for $k > K/2$, which is the same as Sanh et al. 2019. The fine-tuning hyper-parameters are listed in Table 12.

Stage I. We initialize each task-aware filter of the teacher with size $d_t \times d_t$. We fix the fine-tuned teacher and train the filters following the hyper-parameter configurations listed in Table 12. We directly take the trained k-th filter of the teacher as the k-th filter of the student without further training.

Stage II. We distill the student model and its filters following the hyper-parameter configurations listed in Table 12. Our implementation is based on Huggingface Transformers. We conduct all experiments using mixed precision training on 8 32G Nvidia V100 GPUs.

Table 12. Hyper-parameters for fine-tuning BERT-base$_{12}$ on MNLI.

Hyper-parameters	BERT-base
Dropout of Task Layer	0.1
Warmup Steps	1000
Learning Rates	3×10^{-5}
Batch Size	32
Weight Decay	0
Training Epochs	3
Learning Rate Decay	Linear
Adam ϵ	1×10^{-6}
Adam β_1	0.9
Adam β_2	0.98

A.3. Experiments in Analysis

A.3.1. EXPERIMENTS IN SECTION 6.2

Model. We initialize the teacher model with a pre-trained 24-layer DeBERTaV3-large model that has been fine-tuned on the target task. The teacher model contains 435M parameters and has a hidden dimension $d_t = 1024$. We initialize the student model with a 12-layer DeBERTav3-xsmall model. The student model contains 70M parameters and has a hidden dimension $d_s = 384$. We define the layer mapping function $M(k) = 2k - 1$ for $k \leq K/2$ and $M(k) = 2k$ for $k > K/2$, which is the

same as Sanh et al. 2019. The fine-tuning hyper-parameters are listed in Table 13.

Stage I. We initialize each filter of the teacher model with the size $d_t \times d_t$ and each filter of the student model with the size $d_s \times d_t$. We fix the model parameters of the teacher and the student and train their filters following the hyper-parameters summarized in Table 13.

Stage II. We distill the student model and its filters following the hyper-parameters listed in Table 13. Our implementation is based on Huggingface Transformers. We conduct all experiments using mixed precision training on 8 32G Nvidia V100 GPUs.

Table 13. Hyper-parameters for fine-tuning DeBERTaV3 models on the downstream tasks.

Hyper-parameters	DeBERTaV3-large	DeBERTaV3-base	DeBERTaV3-xsmall
Dropout of Task Layer	$\{0.05, 0.1\}$	$\{0.05, 0.1, 0.15\}$	$\{0.05, 0.1, 0.15\}$
Learning Rates	$\{6, 7, 10\} \times 10^{-6}$	$\{1, 1.5, 2, 2.5, 3, 4, 5\} \times 10^{-5}$	$\{3, 3.5, 5, 6, 8, 9\} \times 10^{-5}$
Batch Size	{32, 64}	{12, 16, 32, 64}	{12,16,32, 64}
Weight Decay	0	0	0
Training Epochs	{2,6,8}	{2,3,6,8}	{2,3,6,8}
Learning Rate Decay	Linear	Linear	Linear
Adam ϵ	1×10^{-6}	1×10^{-6}	1×10^{-6}
Adam β_1	0.9	0.9	0.9
Adam β_2	0.98	0.98	0.98

Table 14. Summary of the GLUE benchmark.

Corpus	Task	#Train	#Dev	#Test	#Label	Metrics
Single-Sentence Classification (GLUE)						
CoLA	Acceptability	8.5k	1k	1k	2	Matthews corr
SST	Sentiment	67k	872	1.8k	2	Accuracy
Pairwise Text Classification (GLUE)						
MNLI	NLI	393k	20k	20k	3	Accuracy
RTE	NLI	2.5k	276	3k	2	Accuracy
QQP	Paraphrase	364k	40k	391k	2	Accuracy/F1
MRPC	Paraphrase	3.7k	408	1.7k	2	Accuracy/F1
QNLI	QA/NLI	108k	5.7k	5.7k	2	Accuracy
Text Similarity (GLUE)						
STS-B	Similarity	7k	1.5k	1.4k	1	Pearson/Spearman corr

A.4. Discussion on the Model Initialization

The model initialization is critical to the learning of the task-aware filters. If the model parameters have not been properly initialized and the filters are directly trained upon such parameters, the filters may fail to learn sufficient task-relevant knowledge and become useless. Below we list our recommended practices for model initialization under different scenarios:

Distillation in the pre-training setting. This setting considers a pre-trained model as the teacher and produces a pre-trained model as the student.

Case 1. If there exists a pre-trained model with the desired student architecture, we can directly initialize the student with its weights and proceed to Stage I.

Case 2. If there does not exist a pre-trained model with the desired student architecture, we consider the following three cases: *Case 2.1.* If there is sufficient computational budget, we can pre-train the student from scratch and then proceed to Stage I. *Case 2.2.* If there is no pre-training budget but the desired student architecture is a shallow version of the teacher (like in the GPT-2_6 case), we can initialize the student with a subset of teacher layers. We can directly adopt the filters of the teacher at the corresponding layers as the filters of the student, and proceed to Stage II. *Case 2.3.* Otherwise, we recommend

directly proceeding to Stage II.

Distillation in the fine-tuning setting. This setting considers a fine-tuned model as the teacher and produces a fine-tuned model as the student.

Case 1. If there exists a pre-trained model with the desired student architecture (like the DeBERTaV3-xsmall case), we can fine-tune the pre-trained model on the target task, initialize the student with its weights, and proceed to Stage I.

Case 2. If there does not exist a pre-trained model with the desired student architecture, we consider the following three cases: *Case 2.1*. If there is sufficient computational budget, we can pre-train and fine-tune the student from scratch and then proceed to Stage I. *Case 2.2*. If there is limited computational budget, but the desired student architecture is a shallow version of the teacher (like the BERT-base$_6$ case), we can initialize the student with a subset of teacher layers. We directly adopt the filters of the teacher at the corresponding layers as the filters of the student, and proceed to Stage II. *Case 2.3*. Otherwise, we recommend directly proceeding to Stage II.