MoEBERT: from BERT to Mixture-of-Experts via Importance-Guided Adaptation

Anonymous ACL submission

Abstract

Pre-trained language models have demonstrated superior performance in various natural language processing tasks. However, these models usually contain hundreds of millions of parameters, which limits their practicality because of latency requirements in real-world applications. Existing methods train small compressed models via knowledge distillation. However, performance of these small models drops significantly compared with the pre-trained models due to their reduced model capacity. We propose MoEBERT, which uses a Mixture-of-Experts structure to increase model capacity and inference speed. We initialize MoEBERT by adapting the feed-forward neural networks in a pre-trained model into multiple experts. As such, representation power of the pre-trained model is largely retained. During inference, only one of the experts is activated, such that speed can be improved. We also propose a layer-wise distillation method to train MoEBERT. We validate the efficiency and efficacy of MoEBERT on natural language understanding and question answering tasks. Results show that the proposed method outperforms existing task-specific distillation algorithms. For example, our method outperforms previous approaches by over 2% on the MNLI (mismatched) dataset. Our code will be publicly available.

1 Introduction

Pre-trained language models have demonstrated superior performance in various natural language processing tasks, such as natural language understanding (Devlin et al., 2019; Liu et al., 2019; He et al., 2021b) and natural language generation (Radford et al., 2019; Brown et al., 2020). These models can contain billions of parameters, e.g., T5 (Raffel et al., 2019) contains up to 11 billion parameters, and GPT-3 (Brown et al., 2020) consists of up to 175 billion parameters. Their extreme sizes bring challenges in serving the models to real-world applications due to latency requirements.

Model compression through knowledge distillation (Romero et al., 2015; Hinton et al., 2015) is a promising approach that reduces the computational overhead of pre-trained language models while maintaining their superior performance. In knowledge distillation, a large pre-trained language model serves as a teacher, and a smaller student model is trained to mimic the teacher's behavior. Distillation approaches can be categorized into two groups: task-agnostic (Sanh et al., 2019; Jiao et al., 2020; Wang et al., 2020, 2021; Sun et al., 2020a) and task-specific (Turc et al., 2019; Sun et al., 2019; Li et al., 2020; Hou et al., 2020; Sun et al., 2020b; Xu et al., 2020). Task-agnostic distillation pre-trains the student and then fine-tunes it on downstream tasks; while task-specific distillation directly fine-tunes the student after initializing it from a pre-trained model. Note that task-agnostic approaches are often combined with task-specific distillation during fine-tuning for better performance (Jiao et al., 2020). We focus on task-specific distillation in this work.

One major drawback of existing knowledge distillation approaches is the drop in model performance caused by the reduced representation power. That is, because the student model has fewer parameters than the teacher, its model capacity is smaller. For example, the student model in DistilBERT (Sanh et al., 2019) has 66 million parameters, about half the size of the teacher (BERT-base, Devlin et al. 2019). Consequently, performance of DistilBERT drops significantly compared with BERT-base, e.g., over 2% on MNLI (82.2 v.s. 84.5) and over 3% on CoLA (54.7 v.s. 51.3).

We resort to the Mixture-of-Experts (MoE, Shazeer et al. 2017) structure to remedy the representation power issue. MoE models can increase model capacity while keeping the inference computational cost constant. A layer of a MoE model

(Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2021; Yang et al., 2021; Zuo et al., 2021) consists of an attention mechanism and multiple feed-forward neural networks (FFNs) in parallel. Each of the FFNs is called an expert. During training and inference, an input adaptively activates a fixed number of experts (usually one or two). In this way, the computational cost of a MoE model remains constant during inference, regardless of the total number of experts. Such a property facilitates compression without reducing model capacity.

However, MoE models are difficult to train-from-scratch and usually require a significant amount of parameters, e.g., 7.4 billion parameters for Switch-base (Fedus et al., 2021). We propose MoEBERT, which incorporates the MoE structure into pre-trained language models for fine-tuning. Our model can speedup inference while retaining the representation power of the pre-trained language model. Specifically, we incorporate the expert structure by adapting the FFNs in a pre-trained model into multiple experts. For example, the hidden dimension of the FFN is 3072 in BERT-base (Devlin et al., 2019), and we adapt it into 4 experts, each has a hidden dimension 768. In this way, the amount of *effective parameters* (i.e., parameters involved in computing the representation of an input) is cut by half, and we obtain a $\times 2$ speedup. We remark that MoEBERT utilizes more parameters of the pre-trained model than existing approaches, such that it has greater representation power.

To adapt the FFNs into experts, we propose an importance-based method. Empirically, there are some neurons in the FFNs that contribute more to the model performance than the other ones. That is, removing the important neurons causes significant performance drop. Such a property can be quantified by the *importance score* (Molchanov et al., 2019; Xiao et al., 2019; Liang et al., 2021). When initializing MoEBERT, we share the most important neurons (i.e., the ones with the highest scores) among the experts, and the other neurons are distributed evenly. This strategy has two advantages: first, the shared neurons preserve performance of the pre-trained model; second, the non-shared neurons promote diversity among experts, which further boost model performance. After initialization, MoEBERT is trained using a layer-wise task-specific distillation algorithm.

We demonstrate efficiency and efficacy of MoEBERT on natural language understanding and question answering tasks. On the GLUE (Wang et al., 2019) benchmark, our method significantly outperforms existing distillation algorithms. For example, MoEBERT exceeds performance of state-of-the-art task-specific distillation approaches by over 2% on the MNLI (mismatched) dataset. For question answering, MoEBERT increases F1 by 2.6 on SQuAD v1.1 (Rajpurkar et al., 2016) and 7.0 on SQuAD v2.0 (Rajpurkar et al., 2018) compared with existing algorithms.

The rest of the paper is organized as follows: we introduce background and related works in Section 2; we describe MoEBERT in Section 3; experimental results are provided in Section 4; and Section 5 concludes the paper.

2 Background

2.1 Backbone: Transformer

The Transformer (Vaswani et al., 2017) backbone has been widely adopted in pre-trained language models. The model contains several identically-constructed Transformer layers. Each layer has a multi-head self-attention mechanism and a two-layer feed-forward neural network (FFN).

Suppose the output of the attention mechanism is $\mathbf{A}$. Then, the FFN is defined as:

$$\mathbf{H} = \sigma(\mathbf{A}\mathbf{W}_1 + \mathbf{b}_1),\ \mathbf{X}^\ell = \mathbf{W}_2\mathbf{H} + \mathbf{b}_2, \quad (1)$$

where $\mathbf{W}_1 \in \mathbb{R}^{d\times d_h}$, $\mathbf{W}_2 \in \mathbb{R}^{d_h\times d}$, $\mathbf{b}_1 \in \mathbb{R}^{d_h}$ and $\mathbf{b}_2 \in \mathbb{R}^d$ are weights of the FFN, and σ is the activation function. Here d denotes the embedding dimension, and d_h denotes the hidden dimension of the FFN.

2.2 Mixture-of-Experts Models

Mixture-of-Experts models consist of multiple expert layers, which are similar to the Transformer layers. Each of these layers contain a self-attention mechanism and multiple FFNs (Eq. 1) in parallel, where each FFN is called an expert.

Let $\{E_i\}_{i=1}^N$ denote the experts, and N denotes the total number of experts. Similar to Eq. 1, the experts in layer ℓ take the attention output $\mathbf{A}$ as the input. For each $\mathbf{a}_t$ (the t-th row of $\mathbf{A}$) that corresponds to an input token, the corresponding output $\mathbf{x}_t^\ell$ of layer ℓ is

$$\mathbf{x}_t^\ell = \sum_{i\in\mathcal{T}} p_i(\mathbf{a}_t)E_i(\mathbf{a}_t). \quad (2)$$

Here, $\mathcal{T} \subset \{1\cdots N\}$ is the activated set of experts with $|\mathcal{T}| = K$, and p_i is the weight of expert E_i.

Different approaches have been proposed to construct $\mathcal{T}$ and compute p_i. For example, Shazeer et al. (2017) take

$$p_i(\mathbf{a}_t) = [\text{softmax}(\mathbf{a}_t \mathbf{W}_g)]_i , \quad (3)$$

where $\mathbf{W}_g$ is a weight matrix. Consequently, $\mathcal{T}$ is constructed as the experts that yield top-K largest p_i. However, such an approach suffers from load imbalance, i.e., $\mathbf{W}_g$ collapses such that nearly all the inputs are routed to the same expert. Existing works adopt various ad-hoc heuristics to mitigate this issue, e.g., adding Gaussian noise to Eq. 3 (Shazeer et al., 2017), limiting the maximum number of inputs that can be routed to an expert (Lepikhin et al., 2021), imposing a load balancing loss (Lepikhin et al., 2021; Fedus et al., 2021), and using linear assignment (Lewis et al., 2021). In contrast, Roller et al. 2021 completely remove the gate and pre-assign tokens to experts using hash functions, in which case we can take $p_i = 1/K$.

In Eq. 2, a token only activates K instead of N experts, and usually $K \ll N$, e.g., $K = 2$ and $N = 2048$ in GShard (Lepikhin et al., 2021). As such, the number of FLOPs for one forward pass does not scale with the number of experts. Such a property paves the way for increasing inference speed of a pre-trained model without decreasing the model capacity, i.e., we can adapt the FFNs in a pre-trained model into several smaller components, and only activate one of the components for a specific input token.

2.3 Pre-trained Language Models

Pre-trained language models (Peters et al., 2018; Devlin et al., 2019; Raffel et al., 2019; Liu et al., 2019; Brown et al., 2020; He et al., 2021b,a) have demonstrated superior performance in various natural language processing tasks. These models are trained on an enormous amount of unlabeled data, such that they contain rich semantic information that benefits downstream tasks. Fine-tuning pre-trained language models achieves state-of-the-art performance in tasks such that natural language understanding (He et al., 2021a) and natural language generation (Brown et al., 2020).

2.4 Knowledge Distillation

Knowledge distillation (Romero et al., 2015; Hinton et al., 2015) compensates for the performance drop caused by model compression. In knowledge distillation, a small student model mimics the behavior of a large teacher model. For example, DistilBERT (Sanh et al., 2019) uses the teacher's soft prediction probability to train the student model; TinyBERT (Jiao et al., 2020) aligns the student's layer outputs (including attention outputs and hidden states) with the teacher's; MiniLM (Wang et al., 2020, 2021) utilizes self-attention distillation; and CoDIR (Sun et al., 2020a) proposes to use a contrastive objective such that the student can distinguish positive samples from negative ones according to the teacher's outputs.

There are also heated discussions on the number of layers to distill. For example, Wang et al. (2020, 2021) distill the attention outputs of the last layer; Sun et al. (2019) choose specific layers to distill; and Jiao et al. (2020) use different weights for different transformer layers.

There are two variants of knowledge distillation: task-agnostic (Sanh et al., 2019; Jiao et al., 2020; Wang et al., 2020, 2021; Sun et al., 2020a) and task-specific (Turc et al., 2019; Sun et al., 2019; Li et al., 2020; Hou et al., 2020; Sun et al., 2020b; Xu et al., 2020). The former requires pre-training a small model using knowledge distillation and then fine-tuning on downstream tasks, while the latter directly fine-tunes the small model. Note that task-agnostic approaches are often combined with task-specific distillation for better performance, e.g., TinyBERT (Jiao et al., 2020). In this work, we focus on task-specific distillation.

3 Method

In this section, we first present an algorithm that adapts a pre-trained language model into a MoE model. Such a structure enables inference speedup by reducing the number of parameters involved in computing an input token's representation. Then, we introduce a layer-wise task-specific distillation method that compensates for the performance drop caused by model compression.

3.1 Importance-Guided Adaptation of Pre-trained Language Models

Adapting the FFNs in a pre-trained language model into multiple experts facilitates inference speedup while retaining model capacity. This is because in a MoE model, only a subset of parameters are used to compute the representation of a given token (Eq. 2). These activated parameters are referred to as *effective parameters*. For example, by adapting the FFNs in a pre-trained BERT-base (Devlin et al., 2019) (with hidden dimension 3072) model into 4 experts (each has hidden dimension 768), the

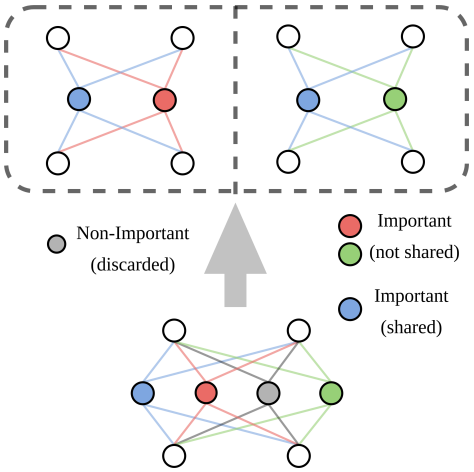


Figure 1: Adapting a two-layer FFN into two experts. The blue neuron is the most important one, and is shared between the two experts. The red and green neurons are the second and third important ones, and are assigned to expert one and two, respectively.

number of effective parameters reduces by half, such that we obtain a $\times 2$ speedup.

Empirically, we find that randomly converting a FFN into experts works poorly (see Figure 4a in the experiments). This is because there are some columns in $\mathbf{W}_1 \in \mathbb{R}^{d \times d_h}$ (correspondingly some rows in $\mathbf{W}_2$ in Eq. 1) contribute more than the others to model performance.

The importance score (Molchanov et al., 2019; Xiao et al., 2019; Liang et al., 2021), originally introduced in model pruning literature, measures such parameter importance. For a dataset $\mathcal{D}$ with sample pairs $\{(x, y)\}$, the score is defined as

$$I_j = \sum_{(x,y)\in\mathcal{D}} \Big| (\mathbf{w}_j^1)^\top \nabla_{\mathbf{w}_j^1} \mathcal{L}(x, y) + (\mathbf{w}_j^2)^\top \nabla_{\mathbf{w}_j^2} \mathcal{L}(x, y) \Big|. \quad (4)$$

Here $\mathbf{w}_j^1 \in \mathbb{R}^d$ is the j-th column of $\mathbf{W}_1$, $\mathbf{w}_j^2$ is the j-th row of $\mathbf{W}_2$, and $\mathcal{L}(x, y)$ is the loss.

The importance score in Eq. 4 indicates variation of the loss if we remove the neuron. That is,

$$|\mathcal{L}_{\mathbf{w}} - \mathcal{L}_{\mathbf{w}=\mathbf{0}}| \approx \Big| (\mathbf{w} - \mathbf{0})^\top \nabla_{\mathbf{w}} \mathcal{L}_{\mathbf{w}} \Big| = |\mathbf{w}^\top \nabla_{\mathbf{w}} \mathcal{L}_{\mathbf{w}}|,$$

where $\mathcal{L}_{\mathbf{w}}$ is the loss with neuron[1] $\mathbf{w}$ and $\mathcal{L}_{\mathbf{w}=\mathbf{0}}$ is the loss without neuron $\mathbf{w}$. Here the approximation is based on the first order Taylor expansion of $\mathcal{L}_{\mathbf{w}}$ around $\mathbf{w} = \mathbf{0}$.

After computing I_j for all the columns, we adapt $\mathbf{W}_1$ into experts.[2] The columns are sorted in ascending order according to their importance scores as $\mathbf{w}^1_{(1)} \cdots \mathbf{w}^1_{(d_h)}$, where $\mathbf{w}^1_{(1)}$ has the largest I_j and $\mathbf{w}^1_{(d_h)}$ the smallest. Empirically, we find that sharing the most important columns benefits model performance. Based on this finding, suppose we share the top-s columns and we adapt the FFN into N experts, then expert e contains columns $\{\mathbf{w}^1_{(1)}, \cdots, \mathbf{w}^1_{(s)}, \mathbf{w}^1_{(s+e)}, \mathbf{w}^1_{(s+e+N)}, \cdots\}$. Note that we discard the least important columns to keep the size of each expert as $\lfloor d/N \rfloor$. Figure 1 is an illustration of adapting a FFN in a pre-trained model into two experts.

3.2 Layer-wise Distillation

To remedy the performance drop caused by adapting a pre-trained model to a MoE model, we adopt a layer-wise task-specific distillation algorithm. We use BERT-base (Devlin et al., 2019) as both the student (i.e., the MoE model) and the teacher. We distill both the Transformer layer output $\mathbf{X}^\ell$ (Eq. 2) and the final prediction probability.

For the Transformer layers, the distillation loss is the mean squared error between the teacher's layer output $\mathbf{X}^\ell_{\text{tea}}$ and the student's layer output $\mathbf{X}^\ell$ obtained from Eq. 2.[3] Concretely, for an input x, the Transformer layer distillation loss is

$$\mathcal{L}_{\text{trm}}(x) = \sum_{\ell=0}^{L} \text{MSE}(\mathbf{X}^\ell, \mathbf{X}^\ell_{\text{tea}}), \quad (5)$$

where L is the total number of layers. Notice that we include the MSE loss of the embedding layer outputs $\mathbf{X}^0$ and $\mathbf{X}^0_{\text{tea}}$.

Let f denotes the MoE model and f_{tea} the teacher model. We obtain the prediction probability for an input x as $p = f(x)$ and $p_{\text{tea}} = f_{\text{tea}}(x)$, where p is the prediction of the MoE model and p_{tea} is the prediction of the teacher model. Then the distillation loss for the prediction layer is

$$\mathcal{L}_{\text{pred}}(x) = \frac{1}{2}\left(\text{KL}(p||p_{\text{tea}}) + \text{KL}(p_{\text{tea}}||p)\right), \quad (6)$$

where KL is the Kullback–Leibler divergence.

The layer-wise distillation loss is the sum of Eq. 5 and Eq. 6, defined as

$$\mathcal{L}_{\text{distill}}(x) = \mathcal{L}_{\text{trm}}(x) + \mathcal{L}_{\text{pred}}(x). \quad (7)$$

We will discuss variants of Eq. 7 in the experiments.

[1] A neuron $\mathbf{w}$ contains two weights $\mathbf{w}^1$ and $\mathbf{w}^2$ as in Eq. 4.

[2] The other parameters in the FFN: $\mathbf{W}_2$, $\mathbf{b}_1$ and $\mathbf{b}_2$ are treated similarly according to I_j.

[3] Note that Eq. 2 computes the layer output of one token $\mathbf{x}^\ell_t$, i.e., one row in $\mathbf{X}^\ell$.

3.3 Model Training

We employ the random hashing strategy (Roller et al., 2021) to train the experts. That is, each token is pre-assigned to a random expert, and this assignment remains the same during training and inference. We will discuss more about other routing strategies of the MoE model in the experiments.

Given the training dataset $\mathcal{D}$ and samples $\{(x, y)\}$, the training objective is

$$\mathcal{L} = \sum_{(x,y)\in\mathcal{D}} \mathrm{CE}(f(x), y) + \lambda_{\text{distill}}\mathcal{L}_{\text{distill}}(x),$$

where CE is the cross-entropy loss and λ_{distill} is a hyper-parameter.

4 Experiments

In this section, we evaluate the efficacy and efficiency of the proposed algorithm on natural language understanding and question answering tasks. We implement our algorithm using the *Huggingface Transformers*[4] (Wolf et al., 2019) code-base. All the experiments are conducted on NVIDIA V100 GPUs.

4.1 Datasets

GLUE. We evaluate performance of the proposed method on the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019), which is a collection of nine natural language understanding tasks. The benchmark includes two single-sentence classification tasks: SST-2 (Socher et al., 2013) is a binary classification task that classifies movie reviews to positive or negative, and CoLA (Warstadt et al., 2019) is a linguistic acceptability task. GLUE also contains three similarity and paraphrase tasks: MRPC (Dolan and Brockett, 2005) is a paraphrase detection task; STS-B (Cer et al., 2017) is a text similarity task; and QQP is a duplication detection task. There are also four natural language inference tasks in GLUE: MNLI (Williams et al., 2018); QNLI (Rajpurkar et al., 2016); RTE (Dagan et al., 2006; Bar-Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009); and WNLI (Levesque et al., 2012). Following previous works on model distillation, we exclude STS-B and WNLI in the experiments. Dataset details are summarized in Appendix A.

Question Answering. We evaluate the proposed algorithm on two question answering datasets: SQuAD v1.1 (Rajpurkar et al., 2016) and SQuAD v2.0 (Rajpurkar et al., 2018). These tasks are treated as a sequence labeling problem, where we predict the probability of each token being the start and end of the answer span. Dataset details can be found in Appendix A.

[4] https://github.com/huggingface/transformers

4.2 Baselines

We compare our method with both task-agnostic and task-specific distillation methods.

In task-agnostic distillation, we pre-train a small language model through knowledge distillation, and then fine-tune on downstream tasks. The fine-tuning procedure also incorporates task-specific distillation for better performance.

DistilBERT (Sanh et al., 2019) pre-trains a small language model by distilling the temperature-controlled soft prediction probability.

TinyBERT (Jiao et al., 2020) is a task-agnostic distillation method that adopts layer-wise distillation.

MiniLM (Wang et al., 2020, 2021) pre-trains a small language model by aligning the attention distribution between the teacher and the student models.

CoDIR (Contrastive Distillation, Sun et al. 2020a) proposes a framework that distills knowledge through intermediate Transformer layers of the teacher via a contrastive objective.

In task-specific distillation, a pre-trained language model is directly compressed and fine-tuned.

PKD (Patient Knowledge Distillation, Sun et al. 2019) proposes a method where the student patiently learns from multiple intermediate Transformer layers of the teacher.

BERT-of-Theseus (Xu et al., 2020) proposes a progressive module replacing method for knowledge distillation.

4.3 Implementation Details

In the experiments, we use BERT-base (Devlin et al., 2019) as both the student model and the teacher model. That is, we first transform the pre-trained model into a MoE model, and then apply layer-wise task-specific knowledge distillation. We set the number of experts in the MoE model to 4, and the hidden dimension of each expert is set to 768, a quarter of the hidden dimension of BERT-base. The other configurations remain unchanged. We share the top-512 important neurons among the experts (see Section 3.1). The number of effective

	RTE Acc	CoLA Mcc	MRPC F1/Acc	SST-2 Acc	QNLI Acc	QQP F1/Acc	MNLI m/mm
BERT-base	63.5	54.7	89.0/84.1	92.9	91.1	88.3/90.9	84.5/84.4
Task-agnostic							
DistilBERT	59.9	51.3	87.5/-	92.7	89.2	-/88.5	82.2/-
TinyBERT (w/o aug)	72.2	42.8	88.4/-	91.6	90.5	-/90.6	83.5/-
MiniLMv1	71.5	49.2	88.4/-	92.0	91.0	-/91.0	84.0/-
MiniLMv2	72.1	52.5	88.9/-	92.4	90.8	-/91.1	84.2/-
CoDIR (pre+fine)	67.1	53.7	89.6/-	**93.6**	90.1	-/89.1	83.5/82.7
Task-specific							
PKD	65.5	24.8	86.4/-	92.0	89.0	-/88.9	81.5/81.0
BERT-of-Theseus	68.2	51.1	89.0/-	91.5	89.5	-/89.6	82.3/-
CoDIR (fine)	65.6	53.6	89.4/-	**93.6**	90.4	-/89.1	83.6/82.8
Ours (task-specific)							
MoEBERT	**74.0**	**55.4**	**92.6/89.5**	93.0	**91.3**	**88.4/91.4**	**84.5/84.8**

Table 1: Experimental results on the GLUE development set. The best results are shown in **bold**. All the models are trained without data augmentation. All the models have $66M$ parameters, except BERT-base ($109M$ parameters). We report mean over three runs. Model references: BERT (Devlin et al., 2019), DistilBERT (Sanh et al., 2019), TinyBERT (Jiao et al., 2020), MiniLMv1 (Wang et al., 2020), MiniLMv2 (Wang et al., 2021), CoDIR (Sun et al., 2020a), PKD (Sun et al., 2019), BERT-of-Theseus (Xu et al., 2020).

parameters of the MoE model is $66M$ (v.s. $106M$ for BERT-base), which is the same as the baseline models. We use the random hashing strategy (Roller et al., 2021) to train the MoE model, we will discuss more later. Detailed training and hyperparameter settings can be found in Appendix B.

4.4 Main Results

Table 1 summarizes experimental results on the GLUE benchmark. Notice that our method outperforms all of the baseline methods in $6/7$ tasks. In general task-agnostic distillation behaves better than task-specific algorithms because of the pre-training stage. For example, the best-performing task-specific method (BERT-of-Theseus) has a 68.2 accuracy on the RTE dataset, whereas accuracy of MiniLMv2 and TinyBERT are greater than 72. Using the proposed method, MoEBERT obtains a 74.0 accuracy on RTE without any pre-training, indicating the efficacy of the MoE architecture. We remark that MoEBERT behaves on par or better than the vanilla BERT-base model in all of the tasks. This shows that there exists redundancy in pre-trained language models, which paves the way for model compression.

Table 2 summarizes experimental results on two question answering datasets: SQuAD v1.1 and SQuAD v2.0. Notice that MoEBERT significantly outperforms all of the baseline methods in terms of both evaluation metrics: exact match (EM) and F1. Similar to the findings in Table 1, task-agnostic distillation methods generally behave better than task-specific ones. For example, PKD has a 69.8 F1 score on SQuAD 2.0, while performance of MiniLMv1 and MiniLMv2 is over 76. Using the proposed MoE architecture, performance of our method exceeds both task-specific and task-agnostic distillation, e.g., the F1 score of MoEBERT on SQuAD 2.0 is 76.8, which is 7.0 higher than PKD (task-specific) and 0.4 higher than MiniLMv2 (task-agnostic).

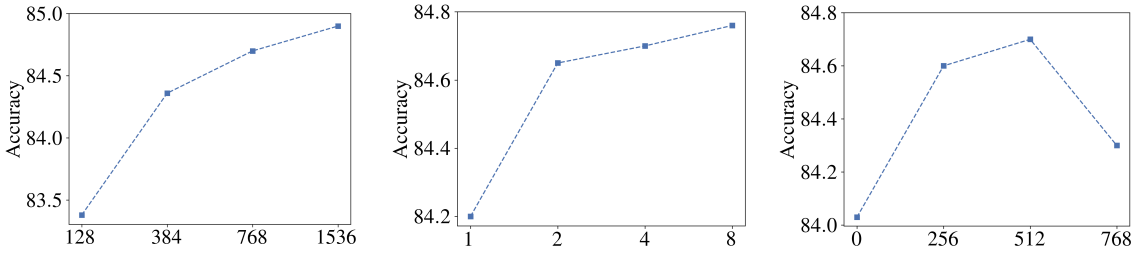
4.5 Ablation Study

Expert dimension. We examine the affect of expert dimension, and experimental results are illustrated in Figure 2a. As we increase the dimension of the experts, model performance improves. This is because of the increased model capacity due to a larger number of effective parameters.

Number of experts. Figure 2b summarizes experimental results when we modify the number of experts. As we increase the number of experts, model performance improves because we effectively enlarge model capacity. We remark that having only one expert is equivalent to compressing

	SQuAD v1.1		SQuAD v2.0	
	EM	F1	EM	F1
BERT-base (Devlin et al., 2019)	80.7	88.4	74.5	77.7
Task-agnostic				
DistilBERT (Sanh et al., 2019)	78.1	86.2	66.0	69.5
TinyBERT (w/o aug) (Jiao et al., 2020)	-	-	-	73.1
MiniLMv1 (Wang et al., 2020)	-	-	-	76.4
MiniLMv2 (Wang et al., 2021)	-	-	-	76.3
Task-specific				
PKD (Sun et al., 2019)	77.1	85.3	66.3	69.8
Ours (task-specific)				
MoEBERT	**80.4**	**87.9**	**73.6**	**76.8**

Table 2: Experimental results on SQuAD v1.1 and SQuAD v2.0. The best results are shown in **bold**. All the models are trained without data augmentation. All the models have $66M$ parameters, except BERT-base ($109M$ parameters). Here *EM* means exact match.



(a) Expert dimension. (b) Number of experts. (c) Shared dimension.

Figure 2: Ablation study on MNLI. We report the average accuracy of MNLI-m and MNLI-mm. As default settings, we have expert dimension 768, number of experts 4, and shared dimension 512.

	RTE	MNLI	SQuAD v2.0
	Acc	m/mm	EM/F1
MoEBERT	74.0	84.5/84.8	73.6/76.8
-distill	73.3	83.2/84.0	72.5/76.0

Table 3: Efficacy of layer-wise distillation.

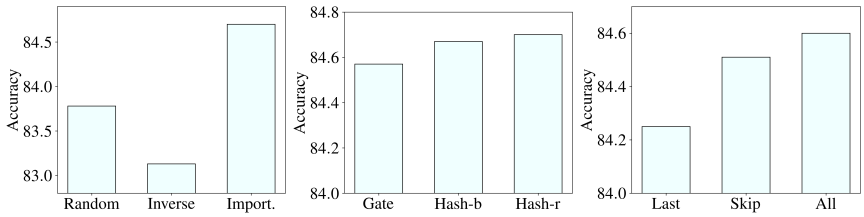
the model without incorporating MoE. In this case performance is unsatisfactory because of the limited representation power of the model.

Shared dimension. Recall that we share important neurons among the experts when adapting the FFNs. In Figure 2c we examine the effect of varying the number of shared neurons. Notice that sharing no neurons yields the worst performance, indicating the efficacy of the sharing strategy. Also notice that performance of sharing all the neurons is also unsatisfactory. We attribute this to the lack of diversity among the experts.

4.6 Analysis

Efficacy of distillation. After adapting the FFNs in the pre-trained BERT-base model into experts, we train MoEBERT using layer-wise knowledge distillation. In Table 3, we examine the efficacy of the proposed distillation method. We show experimental results on RTE, MNLI and SQuAD v2.0, where we remove the distillation and directly fine-tune the adapted model. Results show that by removing the distillation module, model performance significantly drops, e.g., accuracy decreases by 0.7 on RTE and the exact match score decreases by 1.1 on SQuAD v2.0.

Efficacy of importance-based adaptation. Recall that we adapt the FFNs in BERT-base into experts according to the neurons' importance scores (Eq. 4). We examine the method's efficacy by experimenting on two different strategies: randomly split the FFNs into experts (denoted *Random*), and adapt (and share) the FFNs according to the inverse importance, i.e., we share the neurons with



(a) Adaptation methods. (b) Routing methods in MoE. (c) Distillation methods.

Figure 3: Experimental results of model variants on MNLI (average of m and mm). Our methods are denoted *Import*, *Hash-r* and *All* in the subfigures, respectively.

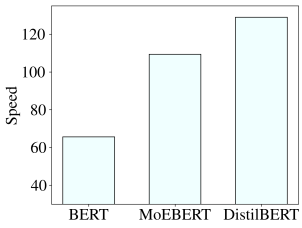


Figure 4: Inference speed (examples/second) on the MNLI dataset.

the smallest scores (denoted *Inverse*). Figure 4a illustrated the results. Notice that performance significantly drops when we apply random splitting compared with *Import* (the method we use). Moreover, performance of *Inverse* is even worse than random splitting, which further demonstrates the efficacy of the importance metric.

Different routing methods. We use a random hashing strategy (denoted *Hash-r*) to route input tokens to experts. That is, each token in the vocabulary is pre-assigned to a random expert, and this assignment remains the same during training and inference. We examine other routing strategies: (1) we employ a trainable gate as in Eq. 3 (denoted *Gate*); (2) we use a balanced hash list (Roller et al., 2021), i.e., tokens are pre-assigned to experts according to frequency, such that each expert receives approximately the same amount of inputs (denoted *Hash-b*). From Figure 4b, we see that all the methods yield similar performance. Therefore, MoEBERT is robust to routing strategies.

Different distillation methods. MoEBERT is trained using a layer-wise distillation method (Eq. 7), where we add a distillation loss to every intermediate layer (denoted *All*). We examine two variants: (1) we only distill the hidden states of the last layer (denoted *Last*); (2) we distill the hidden states of every other layer (denoted *Skip*). Figure 4c shows experimental results. We see that only distilling the last layer yields unsatisfactory performance; while the *Skip* method obtains similar results compared with *All* (the method we use).

Inference speed. We examine inference speed of BERT, DistilBERT and MoEBERT on the MNLI dataset, and Figure 4 illustrates the results. We see that the speed of MoEBERT is slightly slower than DistilBERT, but significantly faster than BERT.

Compressing larger models. Task-specific distillation methods do not require pre-training. Therefore, these methods can be easily applied to other model architectures and sizes beyond BERT-base. We compress the BERT-large model. Specifically, we adapt the FFNs in BERT-large (with hidden dimension 4096) into four experts, such that each expert has hidden dimension 1024. We share the top-512 neurons among experts according to the importance score. After compression, the number of effective parameters is reduces by half. Table 5 demonstrates experimental results on RTE and MNLI. We see that similar to the findings in Table 1, MoEBERT behaves on par or better than BERT-large in all of the experiments.

	RTE Acc	**MNLI-m** Acc	**MNLI-mm** Acc
$\text{BERT}_{\text{large}}$	71.1	**86.3**	86.2
$\text{MoEBERT}_{\text{large}}$	**72.2**	**86.3**	**86.5**

Table 5: Distilling BERT-large on RTE and MNLI.

5 Conclusion

We present MoEBERT, which uses a Mixture-of-Experts structure to distill pre-trained language models. Our proposed method can speedup inference by adapting the feed-forward neural networks (FFNs) in a pre-trained language model into multiple experts. Moreover, the proposed method largely retains model capacity of the pre-trained model. This is in contrast to existing approaches, where the representation power of the compressed model is limited, resulting in unsatisfactory performance. To adapt the FFNs into experts, we adopt an importance-based method, which identifies and shares the most important neurons in a FFN among the experts. We further propose a layer-wise task-specific distillation algorithm to train MoEBERT . We conduct systematic experiments on natural language understanding and question answering tasks. Results show that the proposed method outperforms existing distillation approaches.

References

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, and Danilo Giampiccolo. 2006. The second PASCAL recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. 2009. The fifth pascal recognizing textual entailment challenge. In *In Proc Text Analysis Conference (TAC'09*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. 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, pages 177–190, Berlin, Heidelberg. Springer-Verlag.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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

William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *ArXiv preprint*, abs/2101.03961.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 1–9, Prague. Association for Computational Linguistics.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021a. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing. *ArXiv preprint*, abs/2111.09543.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021b. Deberta: decoding-enhanced bert with disentangled attention. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

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

Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. 2020. Dynabert: Dynamic BERT with adaptive width and depth. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2021. Gshard: Scaling giant models with conditional computation and automatic sharding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. 2021. BASE layers: Simplifying training of large, sparse models. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 6265–6274. PMLR.

Jianquan Li, Xiaokang Liu, Honghong Zhao, Ruifeng Xu, Min Yang, and Yaohong Jin. 2020. BERT-EMD: Many-to-many layer mapping for BERT compression with earth mover's distance. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3009–3018, Online. Association for Computational Linguistics.

Chen Liang, Simiao Zuo, Minshuo Chen, Haoming Jiang, Xiaodong Liu, Pengcheng He, Tuo Zhao, and Weizhu Chen. 2021. Super tickets in pre-trained language models: From model compression to improving generalization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6524–6538, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv preprint*, abs/1907.11692.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. 2019. Importance estimation for neural network pruning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 11264–11272. Computer Vision Foundation / IEEE.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *ArXiv preprint*, abs/1910.10683.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. 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)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Stephen Roller, Sainbayar Sukhbaatar, Arthur Szlam, and Jason Weston. 2021. Hash layers for large sparse models. *ArXiv preprint*, abs/2106.04426.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2015. Fitnets: Hints for thin deep nets. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

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

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. 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)*, pages 4323–4332, Hong Kong, China. Association for Computational Linguistics.

Siqi Sun, Zhe Gan, Yuwei Fang, Yu Cheng, Shuohang Wang, and Jingjing Liu. 2020a. Contrastive distillation on intermediate representations for language model compression. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 498–508, Online. Association for Computational Linguistics.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020b. MobileBERT: a compact task-agnostic BERT for resource-limited devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2158–2170, Online. Association for Computational Linguistics.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models. *ArXiv preprint*, abs/1908.08962.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. 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.

Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. 2021. MiniLMv2: Multi-head self-attention relation distillation for compressing pretrained transformers. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2140–2151, Online. Association for Computational Linguistics.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. 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)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv preprint*, abs/1910.03771.

Xia Xiao, Zigeng Wang, and Sanguthevar Rajasekaran. 2019. Autoprune: Automatic network pruning by regularizing auxiliary parameters. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 13681–13691.

Canwen Xu, Wangchunshu Zhou, Tao Ge, Furu Wei, and Ming Zhou. 2020. BERT-of-theseus: Compressing BERT by progressive module replacing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7859–7869, Online. Association for Computational Linguistics.

An Yang, Junyang Lin, Rui Men, Chang Zhou, Le Jiang, Xianyan Jia, Ang Wang, Jie Zhang, Jiamang Wang, Yong Li, et al. 2021. Exploring sparse expert models and beyond. *ArXiv preprint*, abs/2105.15082.

Simiao Zuo, Xiaodong Liu, Jian Jiao, Young Jin Kim, Hany Hassan, Ruofei Zhang, Tuo Zhao, and Jianfeng Gao. 2021. Taming sparsely activated transformer with stochastic experts. *arXiv preprint arXiv:2110.04260*.

A Dataset details

Statistics of the GLUE benchmark is summarized in Table 7. Statistics of the question answering datasets (SQuAD v1.1 and SQuAD v2.0) are summarized in Table 6.

	#Train	#Validation
SQuAD v1.1	87,599	10,570
SQuAD v2.0	130,319	11,873

Table 6: Statistics of the SQuAD dataset.

B Training Details

We use Adam (Kingma and Ba, 2015) as the optimizer with parameters $(\beta_1, \beta_2) = (0.9, 0.999)$. We employ gradient clipping with a maximum gradient norm 1.0, and we choose weight decay from $\{0, 0.01, 0.1\}$. The learning rate is chosen from $\{1 \times 10^{-5}, 2 \times 10^{-5}, 3 \times 10^{-5}, 4 \times 10^{-5}\}$, and we do not use learning rate warm-up. We train the model for $\{3, 4, 5, 10\}$ epochs with a batch size chosen from $\{8, 16, 32, 64\}$. The weight of the distillation loss λ_{distil} is chosen from $\{1, 2, 3, 4, 5\}$.

Hyper-parameters for distilling BERT-base is summarized in Table 8. We use Adam (Kingma and Ba, 2015) as the optimizer with parameters $(\beta_1, \beta_2) = (0.9, 0.999)$. We employ gradient clipping with a maximum gradient norm 1.0. We do not use learning rate warm-up. For the GLUE benchmark, we use a maximum sequence length of 512 except MNLI and QQP, where we set the maximum sequence length to 128. For the SQuAD datasets, the maximum sequence length is set to 384.

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

Table 7: Summary of the GLUE benchmark.

	lr	batch	epoch	decay	λ_{distill}
RTE	1×10^{-5}	1×8	10	0.01	1.0
CoLA	2×10^{-5}	1×8	10	0.0	3.0
MRPC	3×10^{-5}	1×8	5	0.0	2.0
SST-2	2×10^{-5}	2×8	5	0.0	1.0
QNLI	2×10^{-5}	4×8	5	0.0	2.0
QQP	3×10^{-5}	8×8	5	0.0	1.0
MNLI	5×10^{-5}	8×8	5	0.0	5.0
SQuAD v1.1	3×10^{-5}	4×8	5	0.01	2.0
SQuAD v2.0	3×10^{-5}	2×8	4	0.1	1.0

Table 8: Hyper-parameters for distilling BERT-base. From left to right: learning rate; batch size (2×8 means we use a batch size of 2 and 8 GPUs); number of training epochs; weight decay; and weight of the distillation loss.