MOBILEBERT: TASK-AGNOSTIC COMPRESSION OF BERT BY PROGRESSIVE KNOWLEDGE TRANSFER

Anonymous authors
Paper under double-blind review

ABSTRACT

The recent development of Natural Language Processing (NLP) has achieved great success using large pre-trained models with hundreds of millions of parameters. However, these models suffer from the heavy model size and high latency such that we cannot directly deploy them to resource-limited mobile devices. In this paper, we propose MobileBERT for compressing and accelerating the popular BERT model. Like BERT, MobileBERT is task-agnostic; that is, it can be universally applied to various downstream NLP tasks via fine-tuning. MobileBERT is a slimmed version of BERT_{LARGE} augmented with bottleneck structures and a carefully designed balance between self-attentions and feed-forward networks. To train MobileBERT, we use a bottom-to-top progressive scheme to transfer the intrinsic knowledge of a specially designed Inverted Bottleneck BERT_{LARGE} teacher to it. Empirical studies show that MobileBERT is $4.3 \times$ smaller and $4.0 \times$ faster than original BERT_{BASE} while achieving competitive results on well-known NLP benchmarks. On the natural language inference tasks of GLUE, MobileBERT achieves 0.6 GLUE score performance degradation, and 367 ms latency on a Pixel 3 phone. On the SQuAD v1.1/v2.0 question answering task, MobileBERT achieves a 90.0/79.2 dev F1 score, which is 1.5/2.1 higher than BERT_{BASE}.

1 INTRODUCTION

The NLP community has witnessed a revolution of pre-training self-supervised models. These models usually have hundreds of millions of parameters. They are trained on huge unannotated corpus and then fine-tuned for different small-data tasks (Peters et al., 2018; Radford et al., 2018; Devlin et al., 2018; Radford et al., 2019; Yang et al., 2019). Among these models, BERT (Devlin et al., 2018), which stands for Bidirectional Encoder Representations from Transformers (Vaswani et al., 2017), shows substantial accuracy improvements compared to training from scratch using annotated data only. However, as one of the largest models ever in NLP, BERT suffers from the heavy model size and high latency, making it impractical for resource-limited mobile devices to deploy the power of BERT in mobile-based machine translation, dialogue modeling, and the like.

There have been some works that task-specifically distill BERT into compact models (Turc et al., 2019; Tang et al., 2019; Sun et al., 2019; Tsai et al., 2019). To the best of our knowledge, there is not yet any work for building a task-agnostic lightweight pre-trained model, that is, a model that can be fine-tuned on downstream NLP tasks just like what the original BERT does. In this paper, we propose MobileBERT to fill this gap. In practice, task-agnostic compression of BERT is desirable. Task-specific compression needs to first fine-tune the original large BERT model into task-specific teachers and then distill. Such a process is way more complicated and costly than directly fine-tuning a task-agnostic compact model.

At first glance, it may seem straightforward to obtain a task-agnostic compact version of BERT. For example, one may just take a narrower or shallower architecture of BERT, and then train it with a prediction loss together with a distillation loss (Turc et al., 2019; Sun et al., 2019). Unfortunately, empirical results show that such a straightforward approach results in significant accuracy loss (Turc et al., 2019). This may not be that surprising. It aligns with a well-known observation that shallow networks usually do not have enough representation power while narrow and deep networks are difficult to train. Our MobileBERT is designed to be as deep as BERT_{LARGE} while each layer is made much narrower via adopting bottleneck structures and balancing between self-attentions and feed-
forward networks (Figure 1). To train MobileBERT, we use a bottom-to-top progressive scheme to transfer the intrinsic knowledge of a specially designed Inverted Bottleneck BERTLARGE (IB-BERT) teacher to it.

As a pre-trained NLP model, MobileBERT is both storage efficient (w.r.t model size) and computationally efficient (w.r.t latency) for mobile and resource-constrained environments. Experimental results on several NLP tasks show that while being $4.3 \times$ smaller and $4.0 \times$ faster, MobileBERT can still achieve competitive results compared to BERTBASE. On the natural language inference tasks of GLUE, MobileBERT can have only 0.6 GLUE score performance degradation with 367 ms latency on a Pixel 3 phone. On the SQuAD v1.1/v2.0 question answering task, MobileBERT obtains 90.3/80.2 dev F1 score which is 1.5/2.1 higher than BERTBASE.

2 RELATED WORK

2.1 BERT

BERT takes the embedding of source tokens as input. Each building block of BERT contains one Multi-Head self-Attention (MHA) module (Vaswani et al., 2017) and one Feed-Forward Network (FFN) module, which are connected by skip connections. The MHA module allows the model to jointly attend to information from different subspaces, while the position-wise FFN consists of a two-layer linear transformation with gelu activation (Hendrycks & Gimpel, 2016), which increase the representational power of the model. Figure 1(a) illustrates the original BERT architecture.

In the pre-training stage, BERT is required to predict the masked tokens in sentences (mask language modeling task), as well as whether one sentence is the next sentence of the other (next sentence prediction task). In the fine-tuning stage, BERT is further trained on task-specific annotated data.

2.2 KNOWLEDGE TRANSFER FOR MODEL COMPRESSION

Exploiting knowledge transfer to compress model size was first proposed by Bucilu et al. (2006). The idea was then adopted in knowledge distillation (Hinton et al., 2015), which requires the smaller student network to mimic the class distribution output of the larger teacher network. Fitnets (Romero et al., 2014) make the student mimic the intermediate hidden layers of the teacher to train narrow and deep networks. Luo et al. (2016) show that the knowledge of the teacher can also be obtained from the neurons in the top hidden layer. Similar to our proposed progressive knowledge transfer scheme, Yeo et al. (2018) proposed a sequential knowledge transfer scheme to distill knowledge from a deep teacher into a shallow student in a sequential way. Zagoruyko & Komodakis (2016)
proposed to transfer the attention maps of the teacher on images. Li et al. (2019) proposed to transfer the similarity of hidden states and word alignment from an autoregressive Transformer teacher to a non-autoregressive student.

Recently, knowledge transfer for BERT has attracted much attention. Researchers have distilled BERT into smaller pre-trained BERT models (Turc et al., 2019), an extremely small bi-directional LSTM (Tang et al., 2019), and smaller models on sequence labeling tasks (Tsai et al., 2019). Sun et al. (2019) distill BERT into shallower students through knowledge distillation and an additional knowledge transfer of hidden states on multiple intermediate layers. In contrast to these works, we only use knowledge transfer in the pre-training stage and do not require a fine-tuned teacher for task-specific knowledge in the downstream tasks. Moreover, compared to patient knowledge distillation (Sun et al., 2019) which transfers knowledge for all intermediate layers simultaneously to alleviate over-fitting in downstream task fine-tuning, we design a novel progressive knowledge transfer which eases the pre-training of our compact MobileBERT.

3 Bottom-to-Top Progressive Knowledge Transfer of BERT

The pre-training of BERT is challenging. This problem becomes more severe when we pre-train a compact BERT model from scratch (Frankle & Carbin, 2018). To tackle this problem, we propose a bottom-to-top progressive knowledge transfer scheme. Specifically, we first train a wider teacher network that is easier to optimize, and then progressively train the student network from bottom to top, requiring it to mimic the teacher network layer by layer. In our algorithm, the student and the teacher can be any multi-head attention encoder such as Transformer (Vaswani et al., 2017), BERT or XLNet (Yang et al., 2019). We take BERT as an example in the following description.

The progressive knowledge transfer is divided into $L$ stages, where $L$ is the number of layers. Figure 2 illustrates the diagram and algorithm of progressive knowledge transfer. The idea of progressive transfer is that when training the $(\ell+1)$th layer of the student, the $\ell$th layer is already well-optimized. As there are no soft target distributions that can be used for the intermediate states of BERT, we propose the following two knowledge transfer objectives, i.e., feature map transfer and attention transfer, to train the student network. Particularly, we assume that the teacher and the student have the same 1) feature map size, 2) the number of layers, and 3) the number of attention heads.

3.1 Feature Map Transfer (FMT)

Since each layer in BERT merely takes the output of the previous layer as input, the most important thing in progressively training the student network is that the feature maps of each layer should be as close as possible to those of the teacher, i.e., well-optimized. In particular, the mean squared error between the normalized feature maps of the student and the teacher is used as the objective:

$$L_{\text{FMT}}^{\ell} = \frac{1}{TN} \sum_{t=1}^{T} \sum_{n=1}^{N} (\text{LayerNorm}(H_{tr}^{\ell t})_n - \text{LayerNorm}(H_{st}^{\ell t})_n)^2,$$

where $\ell$ is the index of layers, $T$ is the sequence length, and $N$ is the feature map size. The layer normalization is added to stabilize the layer-wise training loss.

We also minimize two statistics discrepancies on mean and variance in feature map transfer:

$$L_{\mu}^{\ell} = \frac{1}{T} \sum_{t=1}^{T} (\mu(H_{tr}^{\ell t}) - \mu(H_{st}^{\ell t}))^2,$$

$$L_{\sigma^2}^{\ell} = \frac{1}{T} \sum_{t=1}^{T} |\sigma^2(H_{tr}^{\ell t}) - \sigma^2(H_{st}^{\ell t})|,$$

where $\mu$ and $\sigma^2$ represents mean and variance, respectively. Our empirical studies show that minimizing the statistics discrepancy is helpful when layer normalization is removed from BERT to reduce inference latency (see more discussions in Section 4.3).

3.2 Attention Transfer (AT)

The attention mechanism greatly boosts the performance of NLP and becomes a crucial building block in Transformer and BERT. Many papers (Clark et al., 2019; Jawahar et al., 2019) discover that the attention distributions in BERT detect reasonable semantic and syntactic relationships between
words. This motivates us to use self-attention maps from the well-optimized teacher to help the training of the student in augmentation to the feature map transfer. In particular, we minimize the KL-divergence between the per-head self-attention distributions of the teacher and the student:

\[
\mathcal{L}_{AT}^\ell = \frac{1}{TA} \sum_{t=1}^{T} \sum_{a=1}^{A} D_{KL}(a_{t,\ell,a}^{tr} || a_{t,\ell,a}^{st}),
\]

where \( A \) is the number of attention heads.

### 3.3 Bottom-to-Top Progressive Knowledge Transfer (PKT)

Our final progressive knowledge transfer loss \( \mathcal{L}_{PKT}^{\ell} \) for the \( \ell \)-th stage is a linear combination of the objectives stated above. As shown in the right panel of Figure 2, we progressively train each layer of the student by minimizing the knowledge transfer loss. In other words, when we train the \( \ell \)-th layer, we freeze all the trainable parameters in the layers below. We can somewhat soften the training process as follows. When training a layer, we further tune the lower layers with a small learning rate rather than entirely freezing them. Freezing the lower layers can be regarded as a special case of this softened process with the learning rate being zero. There is no knowledge transfer for the beginning embedding layer and the final classifier. They are the same for the student and teacher.

### 3.4 Pre-training Distillation (PD)

After the progressive knowledge transfer, we further pre-train MobileBERT until convergence. We use a linear combination of the original masked language modeling (MLM) loss, next sentence prediction (NSP) loss, and the new knowledge distillation loss as our pre-training distillation loss:

\[
\mathcal{L}_{MLM-KD} = \sum_{i \in [N]} D_{KL}(P^{tr}(i), P^{st}(i)),
\]

\[
\mathcal{L}_{PD} = \alpha \mathcal{L}_{MLM-KD} + (1 - \alpha) \mathcal{L}_{MLM-KD} + \mathcal{L}_{NSP},
\]

where \([N]\) is the set of masked tokens, \(P^{tr}(i)\) and \(P^{st}(i)\) are two predicted distributions respectively from the teacher and student model on the masked tokens, and \(\alpha\) is a hyperparameter in \((0, 1)\). We do not perform knowledge distillation on the next sentence prediction (NSP) task as it has been shown to be unimportant (Yang et al., 2019; Liu et al., 2019).

### 4 MobileBERT: Compact Architecture Design of BERT

In this section, we present the MobileBERT architecture and the underlining design principle, i.e., how to exploit the benefits of the proposed progressive knowledge transfer.
4.1 Bottleneck and Inverted Bottleneck

MobileBERT is a much slimmed version of BERT\textsubscript{LARGE}. As illustrated in Figure 1(c), to align its feature maps with the teacher’s, it is augmented with the bottleneck modules (He et al., 2016), which have additional shortcut connections outside the original non-linear modules. Through the bottleneck modules, MobileBERT can increase the dimension of its block outputs by a linear transformation, while decreasing the dimension of its block inputs by another linear transformation. So the intra-block hidden size (hidden size of the original non-linear modules) stays unchanged. Symmetrically, to align with the student’s feature maps, we can also place the inverted bottleneck modules (Sandler et al., 2018) in the BERT\textsubscript{LARGE} teacher (Figure 1b). We refer this variant of BERT\textsubscript{LARGE} as IB-BERT. Through the inverted bottleneck modules, we can effectively reduce the feature map size of the teacher without losing its representational power.

We may either only use bottleneck for the student or only the inverted bottleneck for the teacher to align their feature maps. However, when using both of them, we have a chance to search for a better feature map size for the teacher and student to obtain a more compact student model while not hurting the performance of the teacher.

4.2 Stacked Feed-Forward Networks

A problem introduced by the bottleneck structure of MobileBERT is that the balance between self-attentions and feed-forward networks is broken. In original BERT, the ratio of the parameter numbers in self-attentions and feed-forward networks is always 1:2. But in the bottleneck structure, the inputs to the self-attentions are from wider feature maps (of inter-block size), while the inputs to the feed-forward networks are from narrower bottlenecks (of intra-block size). This results in that the self-attentions in MobileBERT will contain more parameters than normally. Therefore, we propose to use stacked feed-forward networks in MobileBERT to re-balance it. As illustrated in 1(c), each MobileBERT layer contains one self-attention but several stacked feed-forward networks.

4.3 Further Operational Optimizations

By model latency analysis\textsuperscript{1} we find that layer normalization and gelu activation accounted for a considerable proportion of total latency. Therefore, we replace them with new operations in our MobileBERT.

**Remove layer normalization** We replace the layer normalization of a $n$-channel hidden state $h$ with an element-wise linear transformation:

$$\text{NoNorm}(h) = \gamma \circ h + \beta,$$

where $\gamma, \beta \in \mathbb{R}^n$ and $\circ$ denotes the Hadamard product. Please note that NoNorm has different properties from LayerNorm even in test mode since the original layer normalization is not a linear operation for a batch of vectors.

**Use relu activation** We replace the gelu activation with simpler relu activation.

5 Experiments

5.1 Model Settings of the Teacher and the Student

We conduct extensive experiments to search good model settings for the IB-BERT teacher and the MobileBERT student. We replace the original embedding table by a 3-convolution from a smaller embedding table with embedding size 128 to keep the number of embedding parameters in different model settings the same. We start with SQuAD v1.1 dev F1 score as the metric to measure the performance of different model settings. Since BERT pre-training is time and resource consuming, in the architecture search stage, we only train each model for 125k steps with 2048 batch size, which halves the training schedule of original BERT (Devlin et al., 2018; You et al., 2019).

\textsuperscript{1}A detailed analysis of effectiveness of operational optimizations on real-world inference latency can be found in Appendix C.
Table 1: Experimental results on SQuAD v1.1 dev F1 score in search of good model settings for the IB-BERT\_LARGE teacher (left) and the MobileBERT student (right). The number of layers is set to 24 for all models. Especially, in the right table, the inter-block hidden size of all models is set to 512, which is the same as the chosen teacher model, and the total parameter numbers in these models are all roughly 24M.

<table>
<thead>
<tr>
<th>#Params</th>
<th>h_inter</th>
<th>h_intra</th>
<th>#Head</th>
<th>SQuAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) 356M</td>
<td>1024</td>
<td>1024</td>
<td>16</td>
<td>88.2</td>
</tr>
<tr>
<td>(b) 325M</td>
<td>768</td>
<td>1024</td>
<td>16</td>
<td>88.6</td>
</tr>
<tr>
<td>(c) 293M</td>
<td>512</td>
<td>1024</td>
<td>16</td>
<td>88.1</td>
</tr>
<tr>
<td>(d) 276M</td>
<td>384</td>
<td>1024</td>
<td>16</td>
<td>87.6</td>
</tr>
<tr>
<td>(e) 262M</td>
<td>256</td>
<td>1024</td>
<td>16</td>
<td>87.0</td>
</tr>
<tr>
<td>(f) 293M</td>
<td>512</td>
<td>1024</td>
<td>4</td>
<td>88.3</td>
</tr>
<tr>
<td>(g) 92M</td>
<td>512</td>
<td>512</td>
<td>4</td>
<td>85.8</td>
</tr>
<tr>
<td>(h) 33M</td>
<td>512</td>
<td>256</td>
<td>4</td>
<td>84.8</td>
</tr>
<tr>
<td>(i) 15M</td>
<td>512</td>
<td>128</td>
<td>4</td>
<td>82.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>h_inter</th>
<th>#Head (#Params)</th>
<th>h_FFN (#Params)</th>
<th>SQuAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>192</td>
<td>6 (8M)</td>
<td>1 (7M)</td>
<td>82.5</td>
</tr>
<tr>
<td>160</td>
<td>5 (6.5M)</td>
<td>2 (10M)</td>
<td>83.4</td>
</tr>
<tr>
<td>128</td>
<td>4 (5M)</td>
<td>4 (12.5M)</td>
<td>83.4</td>
</tr>
<tr>
<td>96</td>
<td>3 (4M)</td>
<td>8 (14M)</td>
<td>81.6</td>
</tr>
</tbody>
</table>

Table 2: The detailed model settings of a few models. $\mathcal{L}$, $h_{\text{inter}}$, $h_{\text{intra}}$, $h_{\text{FFN}}$, $h_{\text{embedding}}$, #Head, #FFN, and #Params denote the number of layers, inter-block hidden size (feature map size), intra-block hidden size, FFN intermediate size, embedding table size, the number of heads in multi-head attention, the number of FFN layers, and the number of parameters, respectively.

<table>
<thead>
<tr>
<th>$\mathcal{L}$</th>
<th>$h_{\text{inter}}$</th>
<th>$h_{\text{intra}}$</th>
<th>$h_{\text{FFN}}$</th>
<th>$h_{\text{embedding}}$</th>
<th>#Head</th>
<th>#FFN</th>
<th>#Params</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT_LARGE</td>
<td>24</td>
<td>1024</td>
<td>1024</td>
<td>4096</td>
<td>16</td>
<td>1</td>
<td>334M</td>
</tr>
<tr>
<td>BERT_BASE</td>
<td>12</td>
<td>768</td>
<td>768</td>
<td>3072</td>
<td>12</td>
<td>1</td>
<td>109M</td>
</tr>
<tr>
<td>BERT_SMALL*</td>
<td>12</td>
<td>384</td>
<td>384</td>
<td>1536</td>
<td>12</td>
<td>1</td>
<td>25.3M</td>
</tr>
<tr>
<td>IB-BERT_LARGE</td>
<td>24</td>
<td>512</td>
<td>1024</td>
<td>4096</td>
<td>12</td>
<td>4</td>
<td>293M</td>
</tr>
<tr>
<td>MobileBERT</td>
<td>24</td>
<td>512</td>
<td>128</td>
<td>512</td>
<td>12</td>
<td>4</td>
<td>25.3M</td>
</tr>
</tbody>
</table>

Architecture Search of the Teacher As shrinking the inter-block size can effectively compress the model while maintaining its representational power (Sandler et al., 2018), our design philosophy for the teacher model is to use as small inter-block hidden size (feature map size) as possible as long as there is no accuracy loss. Under this guideline, we design experiments to manipulate the inter-block size of a BERT\_LARGE-sized IB-BERT, and the results are shown in the left panel of Table 1 with labels (a)-(e). As can be seen, decreasing the inter-block hidden size doesn’t damage the performance of BERT until the inter-block size is smaller than 512. As a result, we choose the IB-BERT\_LARGE with its inter-block hidden size being 512 as the teacher model.

One may wonder whether we can also shrink the intra-block hidden size of the teacher, as this may bridge the gap between the student and teacher (Mirzadeh et al., 2019). We conduct experiments and the results are shown in the left panel of Table 1 with labels (f)-(i). We can see that when the intra-block hidden size is reduced, the model performance is dramatically worse. This means that the intra-block hidden size, which represents the representation power of non-linear modules, plays a crucial role in BERT. Therefore, unlike the inter-block hidden size, we do not shrink the intra-block hidden size of our teacher model. Besides, by comparing (a) and (f) in Table 1, we can see that reducing the number of heads from 16 to 4 does not harm the performance of BERT. This is in line with the observation in the recent literature (Michel et al., 2019; Voita et al., 2019).

Architecture Search of the Student We seek a compression ratio of $4 \times$ for BERT\_BASE, so we design a set of MobileBERT models all with approximately 25M parameters but different ratios of the parameter numbers in MHA and FFN to select a good student model. The right part of Table 1 shows our experimental results. They have different balances between self-attentions and feed-forward networks. From the table, we can see that the model performance reaches the peak when the ratio of parameters in MHA and FFN is $0.4 \sim 0.6$. This may justify why the original Transformer chooses the parameter ratio of self-attention and feed-forward networks to 0.5.

We choose the architecture with 128 intra-block hidden size and 4 stacked FFNs as the student model in consideration of model accuracy and training efficiency. We also accordingly set the number of attention heads in the teacher model to 4 in preparation for the progressive knowledge transfer. Table 2 demonstrates the model settings of our IB-BERT\_LARGE teacher and MobileBERT student.
Table 3: The test results on the GLUE benchmark (except WNLI). The number below each task denotes the number of training examples. The metrics for these tasks can be found in the GLUE paper (Wang et al., 2018). For tasks with multiple metrics, the metrics are arithmetically averaged to compute the GLUE score. “OPT” denotes the operational optimizations introduced in Section 4.3.

<table>
<thead>
<tr>
<th>Model</th>
<th>#Params</th>
<th>FLOPS</th>
<th>CoLA</th>
<th>SST-2</th>
<th>MRPC</th>
<th>STS-B</th>
<th>QQP</th>
<th>MNLI-m/mm</th>
<th>QNLI</th>
<th>RTE</th>
<th>GLUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELMo-BiLSTM-Attn</td>
<td>-</td>
<td>-</td>
<td>33.6</td>
<td>90.4</td>
<td>84.4</td>
<td>72.3</td>
<td>63.1</td>
<td>74.1/74.5</td>
<td>79.8</td>
<td>58.9</td>
<td>70.0</td>
</tr>
<tr>
<td>OpenAI GPT</td>
<td>109M</td>
<td></td>
<td>47.2</td>
<td>93.1</td>
<td>87.7</td>
<td>84.8</td>
<td>70.1</td>
<td>80.7/80.6</td>
<td>87.2</td>
<td>69.1</td>
<td>76.9</td>
</tr>
<tr>
<td>BERTBASE</td>
<td>109M</td>
<td>22.5B</td>
<td>52.1</td>
<td>93.5</td>
<td>88.9</td>
<td>85.8</td>
<td>71.2</td>
<td>84.6/83.4</td>
<td>90.5</td>
<td>66.4</td>
<td>78.3</td>
</tr>
<tr>
<td>BERTBASE-6L-PKD</td>
<td>66.5M</td>
<td>11.3B</td>
<td>-</td>
<td>92.0</td>
<td>85.0</td>
<td>-</td>
<td>70.7</td>
<td>81.5/81.0</td>
<td>89.0</td>
<td>65.5</td>
<td>-</td>
</tr>
<tr>
<td>BERTBASE-3L-PKD</td>
<td>45.3M</td>
<td>5.7B</td>
<td>-</td>
<td>87.5</td>
<td>80.7</td>
<td>-</td>
<td>68.1</td>
<td>76.7/76.3</td>
<td>84.7</td>
<td>58.2</td>
<td>-</td>
</tr>
<tr>
<td>MobileBERT</td>
<td>25.3M</td>
<td>5.7B</td>
<td>50.5</td>
<td>92.8</td>
<td>88.8</td>
<td>84.4</td>
<td>70.2</td>
<td>83.3/82.6</td>
<td>90.6</td>
<td>66.2</td>
<td>77.7</td>
</tr>
<tr>
<td>MobileBERT w/o OPT</td>
<td>25.3M</td>
<td>5.7B</td>
<td>51.1</td>
<td>92.6</td>
<td>88.8</td>
<td>84.8</td>
<td>70.5</td>
<td>84.3/83.4</td>
<td>91.6</td>
<td>70.4</td>
<td>78.5</td>
</tr>
</tbody>
</table>

5.2 IMPLEMENTATION DETAILS

Following BERT (Devlin et al., 2018), we use the BooksCorpus (Zhu et al., 2015) and English Wikipedia as our pre-training data. To make the IB-BERT\textsubscript{LARGE} teacher reach the same accuracy as original BERT\textsubscript{LARGE}, we train IB-BERT\textsubscript{LARGE} on 256 TPU v3 chips for 500k steps with a batch size of 4096 and LAMB optimizer (You et al., 2019). For MobileBERT, we also use the same training schedule. Besides, progressive knowledge transfer of MobileBERT over 24 layers takes 240k steps, so that each layer of MobileBERT is trained for 10k steps.

For the downstream tasks, all reported results are obtained by simply fine-tuning MobileBERT just like what the original BERT does. To fine-tune the pre-trained models, we search the optimization hyperparameters in a search space including different batch sizes (16/32/48), learning rates ((1-10) \times e^{-5}), and the number of epochs (2-10). The search space is different from the original BERT because we find that MobileBERT usually needs a larger learning rate and more training epochs in fine-tuning. We select the model for testing according to their performance on the development (dev) set.

5.3 GLUE DATASET

The General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018) is a collection of 9 natural language understanding tasks. We briefly describe these tasks in Appendix F. Following BERT (Devlin et al., 2018), we use the final hidden vector corresponding to the first input token as model output, and introduced a new linear classification layer for the final predictions.

We submit the predictions of MobileBERT and MobileBERT without operational optimizations to the online test evaluation system\footnote{https://gluebenchmark.com/leaderboard} of GLUE to get the test results. We compare MobileBERT with BERT\textsubscript{BASE} and a few other state-of-the-art pre-BERT models on the GLUE leaderboard: OpenAI GPT (Radford et al., 2018) and ELMo (Peters et al., 2018). We also compare with a recent work on compressing BERT: BERT-PKD (Sun et al., 2019). The results are listed in Table 3.

We can see that our MobileBERT is quite competitive with the original BERT\textsubscript{BASE}. It outperforms BERT\textsubscript{BASE} a bit on QNLI and RTE tasks, while the overall GLUE score performance gap is only 0.6. Moreover, It outperform the strong OpenAI GPT baseline by 0.8 GLUE score with 4.3 \times smaller model size. We also find that the introduced operational optimizations hurt the model performance a bit. Without these optimizations, MobileBERT can even outperform BERT\textsubscript{BASE} by 0.2 GLUE score.

5.4 SQUAD DATASET

SQuAD is a large-scale reading comprehension datasets. SQuAD1.1 (Rajpurkar et al., 2016) only contains questions that always have an answer in the given context, while SQuAD2.0 (Rajpurkar et al., 2018) contains unanswerable questions. Following BERT (Devlin et al., 2018), we treat questions that do not have an answer as having an answer span with start and end at the sentence classification token to fine-tune a MobileBERT on SQuAD2.0.

\footnote{We follow Devlin et al. (2018) to skip the WNLI task.}
Table 4: The results on the SQuAD dev datasets. †marks our runs with the official code.

<table>
<thead>
<tr>
<th></th>
<th>#Params</th>
<th>#FLOPS</th>
<th>SQuAD v1.1</th>
<th>SQuAD v2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>EM</td>
<td>F1</td>
</tr>
<tr>
<td>DocQA + ELMo</td>
<td>-</td>
<td>-</td>
<td>80.8</td>
<td>88.5</td>
</tr>
<tr>
<td>BERTBASE</td>
<td>109M</td>
<td>71.2B</td>
<td>65.1</td>
<td>67.6</td>
</tr>
<tr>
<td>MobileBERT</td>
<td>25.3M</td>
<td>18.2B</td>
<td>62.9</td>
<td>70.0</td>
</tr>
<tr>
<td>MobileBERT w/o OPT</td>
<td>25.3M</td>
<td>18.2B</td>
<td>83.4</td>
<td>90.3</td>
</tr>
</tbody>
</table>

Table 5: Ablation on the dev sets of GLUE benchmark. BERTBASE, BERTSMALL*, and the bare MobileBERT (i.e., w/o PD, FMT, AT, FMT & OPT) use the standard BERT pre-training scheme. PD, AT, FMT, and OPT denote Pre-training Distillation, Attention Transfer, Feature Map Transfer, and operational OPTimizations, respectively. †marks our runs with the official code.

<table>
<thead>
<tr>
<th></th>
<th>#Params</th>
<th>#FLOPS Latency</th>
<th>MNLI-m</th>
<th>QNLI</th>
<th>MRPC</th>
<th>SST-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERTLARGE (Devlin et al., 2018)</td>
<td>334M</td>
<td>79.2B 5023 ms</td>
<td>86.6</td>
<td>92.1†</td>
<td>87.8</td>
<td>93.7</td>
</tr>
<tr>
<td>IB-BERTLARGE</td>
<td>293M</td>
<td>75.8B 5077 ms</td>
<td>87.0</td>
<td>93.2</td>
<td>87.3</td>
<td>94.1</td>
</tr>
<tr>
<td>BERTBASE  (Devlin et al., 2018)</td>
<td>109M</td>
<td>22.5B 1468 ms</td>
<td>84.4</td>
<td>91.1†</td>
<td>86.7</td>
<td>92.9</td>
</tr>
<tr>
<td>BERTMEDIUM (Turc et al., 2019)</td>
<td>41.1M</td>
<td>6.5B</td>
<td>81.2</td>
<td>-</td>
<td>-</td>
<td>89.6</td>
</tr>
<tr>
<td>BERTSMALL (Turc et al., 2019)</td>
<td>28.2M</td>
<td>3.4B</td>
<td>78.8</td>
<td>-</td>
<td>-</td>
<td>88.4</td>
</tr>
<tr>
<td>BERTSMALL*</td>
<td>25.3M</td>
<td>5.8B 466 ms</td>
<td>80.2</td>
<td>88.1</td>
<td>84.1</td>
<td>89.8</td>
</tr>
<tr>
<td>MobileBERT (bare)</td>
<td>25.3M</td>
<td>5.7B 620 ms</td>
<td>80.8</td>
<td>88.2</td>
<td>84.3</td>
<td>90.1</td>
</tr>
<tr>
<td>+ PD</td>
<td>25.3M</td>
<td>5.7B 620 ms</td>
<td>81.1</td>
<td>88.9</td>
<td>85.5</td>
<td>91.7</td>
</tr>
<tr>
<td>+ PD + FMT</td>
<td>25.3M</td>
<td>5.7B 620 ms</td>
<td>83.8</td>
<td>91.1</td>
<td>87.0</td>
<td>92.2</td>
</tr>
<tr>
<td>+ PD + FMT + AT</td>
<td>25.3M</td>
<td>5.7B 620 ms</td>
<td>84.4</td>
<td>91.5</td>
<td>87.0</td>
<td>92.5</td>
</tr>
<tr>
<td>+ PD + FMT + AT + OPT</td>
<td>25.3M</td>
<td>5.7B 367 ms</td>
<td>83.9</td>
<td>91.0</td>
<td>87.5</td>
<td>92.1</td>
</tr>
</tbody>
</table>

We evaluate MobileBERT only on the SQuAD dev datasets, as there is nearly no single model submission on SQuAD test leaderboard. We compare our MobileBERT with BERTBASE and a strong baseline DocQA (Clark & Gardner, 2017). As shown in Table 4, MobileBERT outperforms a large margin over BERTBASE and DocQA. We notice that MobileBERT also outperforms BERTBASE on QNLI, a question-answering GLUE task. This may be due to that since we search the model settings on SQuAD, MobileBERT may be over-fitted to question answering tasks.

5.5 ABLATION STUDY AND DISCUSSION

We perform an ablation study to investigate how each component of MobileBERT contributes to its performance on the dev data of a few GLUE tasks with diverse characteristics. To accelerate the experiment process, we halve the original pre-training schedule in the ablation study.

We conduct a set of ablation experiments with regard to Attention Transfer (AT), Feature Map Transfer (FMT) and Pre-training Distillation (PD). The operational OPTimizations (OPT) are removed in these experiments. Moreover, to investigate the effectiveness of the proposed novel architecture of MobileBERT, we compare MobileBERT with two compact BERT models from Turc et al. (2019). For a fair comparison, we also design our own BERT baseline BERTSMALL*, which is the best model setting we can find with roughly 25M parameters under the original BERT architecture. The detailed model setting of BERTSMALL* can be found in Table 2.

Besides these experiments, to verify the performance of MobileBERT on real-world mobile devices, we export the models with Tensorflow Lite APIs and measure the inference latencies on a single large core of a Pixel 3 phone with a fixed sequence length of 128.

The results are listed in Table 5. We first can see that the propose Feature Map Transfer contributes most to the performance improvement of MobileBERT, while Attention Transfer and Pre-training Distillation also play positive roles. As expected, the proposed operational OPTimizations hurt the model performance a bit, but it brings a crucial speedup of 1.68×. In architecture comparison, we find that although specifically designed for progressive knowledge transfer, our MobileBERT architecture alone is still quite competitive. It outperforms BERTSMALL* and BERTSMALL on all compared tasks, while outperforming the 1.7× sized BERTMEDIUM on the SST-2 task. Finally, we can
find that although augmented with the powerful progressive knowledge transfer, our MobileBERT still degrades greatly when compared to the IB-BERT\textsubscript{LARGE} teacher.

We visualize the attention distributions of the 1\textsuperscript{st} and the 12\textsuperscript{th} layers of a few models in Figure 3 for further investigation. The proposed attention transfer can help the student mimic the attention distributions of the teacher very well. Surprisingly, we find that the attention distributions in the attention heads of "MobileBERT(bare)+PD+FMT" are exactly a re-order of those of "MobileBERT(bare)+PD+FMT+AT" (also the teacher model), even if it has not been trained by the attention transfer objective. This phenomenon indicates that multi-head attention is a crucial and unique part of the non-linearity of BERT. Moreover, it can explain the minor improvements of Attention Transfer in ablation table 5 since the alignment of feature maps lead to the alignment of attention distributions.

6 Conclusion

We have presented MobileBERT which is a task-agnostic compact variant of BERT. It is built upon a progressive knowledge transfer method and a conjugate architecture design. Standard model compression techniques including quantization (Shen et al., 2019) and pruning (Zhu & Gupta, 2017) can be applied to MobileBERT to further reduce the model size as well as the inference latency. In addition, although we have utilized low-rank decomposition for the embedding layer, it still accounts for a large part in the final model. We believe there is a big room for extremely compressing the embedding table (Khrulkov et al., 2019; May et al., 2019).

References


Appendix

A Extra Related Work on Layer-wise Training of Neural Networks

Layer-wise pre-training of neural networks can be dated back to Deep Belief Networks (DBN) (Hinton et al., 2006) and stacked auto-encoders (Vincent et al., 2008). Bengio et al. (2007) showed that the unsupervised pre-training of DBN helps to mitigate the difficult optimization problem of deep networks by better initializing the weights of all layers. Although they made essential breakthrough in the application of neural networks, they are widely considered to be obsolete. A more popular way today is to train deep neural networks in an end-to-end fashion. However, Glasmachers (2017) recently showed that end-to-end learning can sometimes be very inefficient.

In this paper, we propose a progressive knowledge transfer scheme to combine the best of both worlds. Compared to previous layer-wise methods, we use a well-optimized wider teacher to guide the layer-wise pre-training of the narrower student, rather than a greedy layer-wise unsupervised way, which makes better use of labels and rewards. Our method also tackle the difficult training problem of end-to-end training from scratch.

B Extra Related Work on Compact Architecture Design

While much recent research has focused on improving efficient Convolutional Neural Networks (CNN) for mobile vision applications (Iandola et al., 2016; Howard et al., 2017; Zhang et al., 2017; 2018; Sandler et al., 2018; Tan et al., 2019; Howard et al., 2019), they are usually tailored for CNN. Popular lightweight operations such as depth-wise convolution (Howard et al., 2017) cannot be directly applied to Transformer or BERT. In the NLP literature, the most relevant work can be group LSTMs (Kuchaiev & Ginsburg, 2017; Gao et al., 2018), which employs the idea of group convolution (Zhang et al., 2017; 2018) into Recurrent Neural Networks (RNN).

Recently, compressing or accelerating Transformer or BERT has attracted much attention. Ma et al. (2019) apply Block-Term Tensor Decomposition on the self-attention modules of Transformer and achieve a compression of 2.5 on the machine translation task, but they don’t consider how to compress the feed-forward networks, which constrains the compression ratio. Lample et al. (2019) use structured memory layers to replace feed-forward networks in BERT and get better perplexity by half the computation, but they cannot compress the model size. Compared to these work, MobileBERT reduces overheads in both self-attentions and feed-forward networks of BERT by bottleneck structure, while achieves efficiency with regard to both storage and computation.

C Latency Analysis of Operational Optimizations

We evaluate the effectiveness of our two operational optimizations for MobileBERT introduced in Section 4.3 replacing layer normalization (LayerNorm) with NoNorm and replacing gelu activation with relu activation. We use the same experimental setting as in Section 5.3, where the models are exported to Tensorflow Lite format and evaluated on a single large core of a Pixel 3 phone with a fixed sequence length of 128. From Table 6, we can see that both NoNorm and relu are very effective in reducing the latency of MobileBERT, even if these two operational optimizations do not reduce FLOPS. This reveals the gap between the real-world inference latency and the theoretical computation overhead (i.e., FLOPS).

Table 6: The effectiveness of operational optimizations on real-world inference latency.

<table>
<thead>
<tr>
<th>Setting</th>
<th>#FLOPS</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>MobileBERT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LayerNorm &amp; gelu</td>
<td>5.7B</td>
<td>620 ms</td>
</tr>
<tr>
<td>NoNorm &amp; gelu</td>
<td>5.7B</td>
<td>449 ms</td>
</tr>
<tr>
<td>LayerNorm &amp; relu</td>
<td>5.7B</td>
<td>538 ms</td>
</tr>
<tr>
<td>NoNorm &amp; relu</td>
<td>5.7B</td>
<td>367 ms</td>
</tr>
</tbody>
</table>
D Extra Experimental Settings

**Progressive Knowledge Transfer** Our final progressive knowledge transfer loss $L_{PKT}^\ell$ in Section 3.3 for the $\ell^{th}$ stage can be written as:

$$L_{PKT}^\ell = \lambda L_{AT}^\ell + \mu L_{FMT}^\ell + \beta L_{\mu}^\ell + \gamma L_{\sigma}^\ell$$  (7)

($\lambda, \mu, \beta, \gamma$) are hyperparameters to balance the different loss terms. Specifically, we use $\lambda = 1, \mu = 100, \beta = 5000, \gamma = 5$ in our all experiments.

**Pre-train MobileBERT** For a fair comparison with original BERT, we follow the same pre-processing scheme as BERT, where we mask 15% of all WordPiece tokens in each sequence at random and use next sentence prediction. Please note that MobileBERT can be potentially further improved by several training techniques recently introduced, such as span prediction (Joshi et al., 2019) or removing next sentence prediction objective (Liu et al., 2019). We leave it for future work.

In pre-training distillation, the hyperparameter $\alpha$ is used to balance the original masked language modeling loss and the distillation loss. Following (Kim & Rush, 2016), we set $\alpha$ to 0.5.

E Comparison with DistilBERT

We notice that recently there is an unpublished work[6] that also propose a task-agnostically compressed BERT, called DistilBERT. Basically, DistilBERT is a 6-layer truncated BERTBASE, which is distilled from BERTBASE on unannotated data with masked language modeling target. The distillation process of DistilBERT is quite similar to the pre-training distillation described in Section 3.4. In comparison, in this paper, we propose a pair of conjugate architectures to help knowledge transfer and design a progressive knowledge transfer scheme which transfers the intrinsic knowledge of intermediate layers from the teacher to the student in a bottom-to-top progressive way.

F GLUE Dataset

In this section, we provide a brief description of the tasks in the GLUE benchmark (Wang et al., 2018).

**CoLA** The Corpus of Linguistic Acceptability (Warstadt et al., 2018) is a collection of English acceptability judgments drawn from books and journal articles on linguistic theory. The task is to predict whether an example is a grammatical English sentence and is evaluated by Matthews correlation coefficient (Matthews, 1975).

**SST-2** The Stanford Sentiment Treebank (Socher et al., 2013) is a collection of sentences from movie reviews and human annotations of their sentiment. The task is to predict the sentiment of a given sentence and is evaluated by accuracy.

**MRPC** The Microsoft Research Paraphrase Corpus (Dolan & Brockett, 2005) is a collection of sentence pairs automatically extracted from online news sources. They are labeled by human annotations for whether the sentences in the pair are semantically equivalent. The performance is evaluated by both accuracy and F1 score.

**STS-B** The Semantic Textual Similarity Benchmark (Cer et al., 2017) is a collection of sentence pairs drawn from news headlines, video and image captions, and natural language inference data. Each pair is human-annotated with a similarity score from 1 to 5. The task is to predict these scores and is evaluated by Pearson and Spearman correlation coefficients.

QQP  The Quora Question Pairs [Chen et al., 2018] dataset is a collection of question pairs from the community question-answering website Quora. The task is to determine whether a pair of questions are semantically equivalent and is evaluated by both accuracy and F1 score.

MNLI  The Multi-Genre Natural Language Inference Corpus [Williams et al., 2018] is a collection of sentence pairs with textual entailment annotations. Given a premise sentence and a hypothesis sentence, the task is to predict whether the premise entails the hypothesis (entailment), contradicts the hypothesis (contradiction), or neither (neutral) and is evaluated by accuracy on both matched (in-domain) and mismatched (cross-domain) sections of the test data.

QNLI  The Question-answering NLI dataset is converted from the Stanford Question Answering Dataset (SQuAD) [Rajpurkar et al., 2016]. The task is to determine whether the context sentence contains the answer to the question and is evaluated by the test accuracy.

RTE  The Recognizing Textual Entailment (RTE) datasets come from a series of annual textual entailment challenges [Bentivogli et al., 2009]. The task is to predict whether sentences in a sentence pair are entailment and is evaluated by accuracy.

WNLI  The Winograd Schema Challenge [Levesque et al., 2011] is a reading comprehension task in which a system must read a sentence with a pronoun and select the referent of that pronoun from a list of choices. We follow Devlin et al. [2018] to skip this task in our experiments, because few previous works do better than predicting the majority class for this task.

G  Visualization of Attention Distributions

We show the visualization of attention distributions in all 24 layers of a few models in Figure 4-5 (for IB-BERT teacher), Figure 6-7 (for MobileBERT w/o OPT), Figure 8-9 (for MobileBERT w/o OPT & AT), Figure 10-11 (for MobileBERT w/o OPT, AT & FMT), and Figure 12-13 (for MobileBERT w/o OPT, AT FMT & PD).

https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs
Figure 4: The visualization of attention distributions in the first 12 layers of IB-BERT teacher.
Figure 5: The visualization of attention distributions in the last 12 layers of IB-BERT teacher.
Figure 6: The visualization of attention distributions in the first 12 layers of MobileBERT (bare) + PD + FMT + AT.
Figure 7: The visualization of attention distributions in the last 12 layers of MobileBERT (bare) + PD + FMT + AT.
Figure 8: The visualization of attention distributions in the first 12 layers of MobileBERT (bare) + PD + FMT.
Figure 9: The visualization of attention distributions in the last 12 layers of MobileBERT (bare) + PD + FMT.
Figure 10: The visualization of attention distributions in the first 12 layers of MobileBERT (bare) + PD.
Figure 11: The visualization of attention distributions in the last 12 layers of MobileBERT (bare) + PD.
Figure 12: The visualization of attention distributions in the first 12 layers of MobileBERT (bare).
Figure 13: The visualization of attention distributions in the last 12 layers of MobileBERT (bare).