MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices

Zhiqing Sun[1]*, **Hongkun Yu**[2], **Xiaodan Song**[2], **Renjie Liu**[2], **Yiming Yang**[1], **Denny Zhou**[2]

[1]Carnegie Mellon University {zhiqings, yiming}@cs.cmu.edu
[2]Google Brain {hongkuny, xiaodansong, renjieliu, dennyzhou}@google.com

Abstract

Natural Language Processing (NLP) has recently achieved great success by using huge pre-trained models with hundreds of millions of parameters. However, these models suffer from heavy model sizes and high latency such that they cannot be deployed to resource-limited mobile devices. In this paper, we propose MobileBERT for compressing and accelerating the popular BERT model. Like the original BERT, MobileBERT is task-agnostic, that is, it can be generically applied to various downstream NLP tasks via simple fine-tuning. Basically, MobileBERT is a thin version of $\text{BERT}_{\text{LARGE}}$, while equipped with bottleneck structures and a carefully designed balance between self-attentions and feed-forward networks. To train MobileBERT, we first train a specially designed teacher model, an inverted-bottleneck incorporated $\text{BERT}_{\text{LARGE}}$ model. Then, we conduct knowledge transfer from this teacher to MobileBERT. Empirical studies show that MobileBERT is 4.3× smaller and 5.5× faster than $\text{BERT}_{\text{BASE}}$ while achieving competitive results on well-known benchmarks. On the natural language inference tasks of GLUE, MobileBERT achieves a GLUE score of 77.7 (0.6 lower than $\text{BERT}_{\text{BASE}}$), and 62 ms latency on a Pixel 4 phone. On the SQuAD v1.1/v2.0 question answering task, MobileBERT achieves a dev F1 score of 90.0/79.2 (1.5/2.1 higher than $\text{BERT}_{\text{BASE}}$).

1 Introduction

The NLP community has witnessed a revolution of pre-training self-supervised models. These models usually have hundreds of millions of parameters (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) shows substantial accuracy improvements. 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 efforts 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 generically fine-tuned on different downstream NLP tasks as 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 a task-specific teacher and then distill. Such a process is much more complicated (Wu et al., 2019) and costly than directly fine-tuning a task-agnostic compact model.

At first glance, it may seem straightforward to obtain a task-agnostic compact BERT. For example, one may just take a narrower or shallower version of BERT, and train it until convergence by minimizing a convex combination of the prediction loss and 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 is well-known 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 $\text{BERT}_{\text{LARGE}}$ while each layer is made much narrower via adopting bottleneck structures and balancing between self-attentions and feed-forward

*This work was done when the first author was an intern at Google Brain.

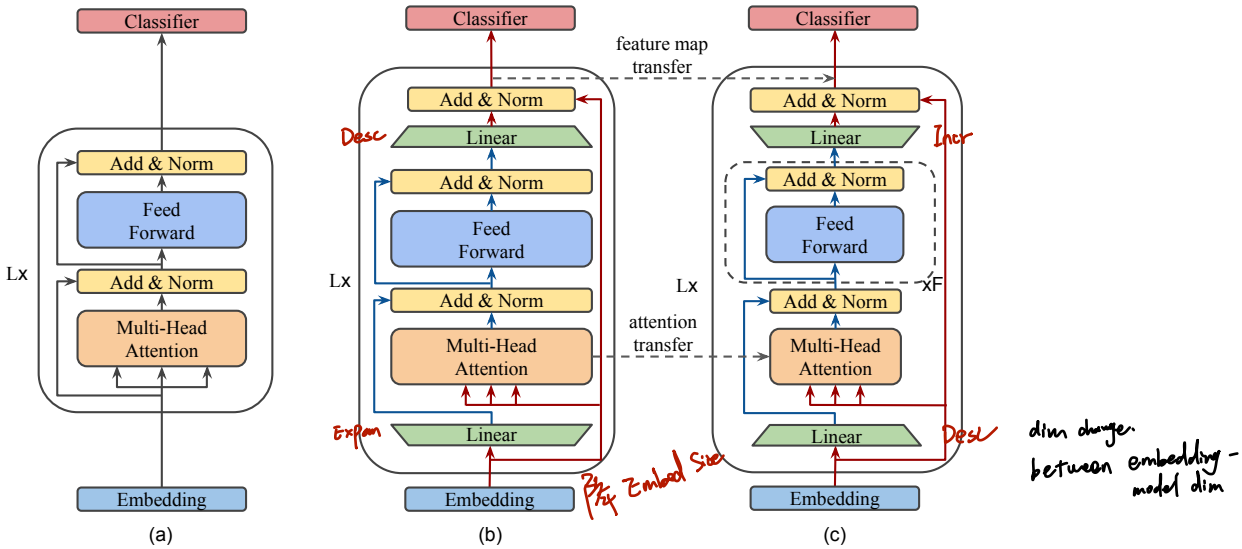

Figure 1: Illustration of three models: (a) BERT; (b) Inverted-Bottleneck BERT (IB-BERT); and (c) MobileBERT. In (b) and (c), red lines denote inter-block flows while blue lines intra-block flows. MobileBERT is trained by layer-to-layer imitating IB-BERT.

networks (Figure 1). To train MobileBERT, a deep and thin model, we first train a specially designed teacher model, an inverted-bottleneck incorporated $\text{BERT}_{\text{LARGE}}$ model (IB-BERT). Then, we conduct knowledge transfer from IB-BERT to MobileBERT. A variety of knowledge transfer strategies are carefully investigated in our empirical studies.

Empirical evaluations[1] show that MobileBERT is 4.3× smaller and 5.5× faster than $\text{BERT}_{\text{BASE}}$, while it can still achieve competitive results on well-known NLP benchmarks. On the natural language inference tasks of GLUE, MobileBERT can achieve a GLUE score of 77.7, which is only 0.6 lower than $\text{BERT}_{\text{BASE}}$, with a latency of 62 ms on a Pixel 4 phone. On the SQuAD v1.1/v2.0 question answering task, MobileBER obtains a dev F1 score of 90.3/80.2, which is even 1.5/2.1 higher than $\text{BERT}_{\text{BASE}}$.

2 Related Work

Recently, compression of BERT has attracted much attention. Turc et al. (2019) propose to pre-train the smaller BERT models to improve task-specific knowledge distillation. Tang et al. (2019) distill BERT into an extremely small LSTM model. Tsai et al. (2019) distill a multilingual BERT into smaller BERT models on sequence labeling tasks. Clark et al. (2019b) use several single-task BERT models to teach a multi-task BERT. Liu et al. (2019a) distill knowledge from an ensemble of BERT models into a single BERT.

Concurrently to our work, Sun et al. (2019) distill BERT into shallower students through knowledge distillation and an additional knowledge transfer of hidden states on multiple intermediate layers. Jiao et al. (2019) propose TinyBERT, which also uses a layer-wise distillation strategy for BERT but in both pre-training and fine-tuning stages. Sanh et al. (2019) propose DistilBERT, which successfully halves the depth of BERT model by knowledge distillation in the pre-training stage and an optional fine-tuning stage.

In contrast to these existing literature, we only use knowledge transfer in the pre-training stage and do not require a fine-tuned teacher or data augmentation (Wu et al., 2019) in the down-stream tasks. Another key difference is that these previous work try to compress BERT by reducing its depth, while we focus on compressing BERT by reducing its width, which has been shown to be more effective (Turc et al., 2019).

3 MobileBERT

In this section, we present the detailed architecture design of MobileBERT and training strategies to efficiently train MobileBERT. The specific model settings are summarized in Table 1. These settings are obtained by extensive architecture search experiments which will be presented in Section 4.1.

[1]The code and pre-trained models are available at https://github.com/google-research/google-research/tree/master/mobilebert.

		BERT$_{\text{LARGE}}$	BERT$_{\text{BASE}}$	IB-BERT$_{\text{LARGE}}$	MobileBERT	MobileBERT$_{\text{TINY}}$
embedding	$h_{\text{embedding}}$	1024	768	128		
		no-op	no-op	3-convolution		
	h_{inter}	1024	768	512		
body Linear	h_{input} h_{output}	$\begin{bmatrix} 1024 \\ 16 \\ 1024 \\ 1024 \\ 4096 \\ 1024 \end{bmatrix} \times 24$	$\begin{bmatrix} 768 \\ 12 \\ 768 \\ 768 \\ 3072 \\ 768 \end{bmatrix} \times 12$	$\begin{bmatrix} 512 \\ 1024 \\ 512 \\ 4 \\ 1024 \\ 1024 \\ 4096 \\ 1024 \\ 1024 \\ 512 \end{bmatrix} \times 24$	$\begin{bmatrix} 512 \\ 128 \\ 512 \\ 4 \\ 128 \\ 128 \\ 512 \\ 128 \\ 128 \\ 512 \end{bmatrix} \times 4 \times 24$	$\begin{bmatrix} 512 \\ 128 \\ 128 \\ 4 \\ 128 \\ 128 \\ 512 \\ 128 \\ 128 \\ 512 \end{bmatrix} \times 2 \times 24$
MHA	h_{input} #Head h_{output}					
FFN	h_{input} h_{FFN} h_{output}					
Linear	h_{input} h_{output}					
#Params		334M	109M	293M	25.3M	15.1M

Table 1: The detailed model settings of a few models. h_{inter}, h_{FFN}, $h_{\text{embedding}}$, #Head and #Params denote the inter-block hidden size (feature map size), FFN intermediate size, embedding table size, the number of heads in multi-head attention, and the number of parameters, respectively.

3.1 Bottleneck and Inverted-Bottleneck

The architecture of MobileBERT is illustrated in Figure 1(c). It is as deep as BERT$_{\text{LARGE}}$, but each building block is made much smaller. As shown in Table 1, the hidden dimension of each building block is only 128. On the other hand, we introduce two linear transformations for each building block to adjust its input and output dimensions to 512. Following the terminology in (He et al., 2016), we refer to such an architecture as bottleneck.

It is challenging to train such a deep and thin network. To overcome the training issue, we first construct a teacher network and train it until convergence, and then conduct knowledge transfer from this teacher network to MobileBERT. We find that this is much better than directly training MobileBERT from scratch. Various training strategies will be discussed in a later section. Here, we introduce the architecture design of the teacher network which is illustrated in Figure 1(b). In fact, the teacher network is just BERT$_{\text{LARGE}}$ while augmented with *inverted*-bottleneck structures (Sandler et al., 2018) to adjust its feature map size to 512. In what follows, we refer to the teacher network as IB-BERT$_{\text{LARGE}}$. Note that IB-BERT and MobileBERT have the same feature map size which is 512. Thus, we can directly compare the layer-wise output difference between IB-BERT and MobileBERT. Such a direct comparison is needed in our knowledge transfer strategy.

It is worth pointing out that the simultaneously introduced bottleneck and inverted-bottleneck structures result in a fairly flexible architecture design. One may either only use the bottlenecks for MobileBERT (correspondingly the teacher becomes BERT$_{\text{LARGE}}$) or only the inverted-bottlenecks for IB-BERT (then there is no bottleneck in MobileBERT) to align their feature maps. However, when using both of them, we can allow IB-BERT$_{\text{LARGE}}$ to preserve the performance of BERT$_{\text{LARGE}}$ while having MobileBERT sufficiently compact.

3.2 Stacked Feed-Forward Networks

A problem introduced by the bottleneck structure of MobileBERT is that the balance between the Multi-Head Attention (MHA) module and the Feed-Forward Network (FFN) module is broken. MHA and FFN play different roles in the Transformer architecture: The former allows the model to jointly attend to information from different subspaces, while the latter increases the non-linearity of the model. In original BERT, the ratio of the parameter numbers in MHA and FFN is always 1:2. But in the bottleneck structure, the inputs to the MHA are from wider feature maps (of inter-block size), while the inputs to the FFN are from narrower bottlenecks (of intra-block size). This results in that the MHA modules in MobileBERT relatively contain more parameters.

To fix this issue, we propose to use stacked feed-forward networks in MobileBERT to re-balance the relative size between MHA and FFN. As illustrated in Figure 1(c), each MobileBERT layer contains one MHA but several stacked FFN. In MobileBERT, we use 4 stacked FFN after each MHA.

3.3 Operational Optimizations

By model latency analysis[2], we find that layer normalization (Ba et al., 2016) and `gelu` activation (Hendrycks and Gimpel, 2016) accounted for a considerable proportion of total latency. Therefore, we propose to replace them with new operations in our MobileBERT.

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

$$\texttt{NoNorm}(\mathbf{h}) = \boldsymbol{\gamma} \circ \mathbf{h} + \boldsymbol{\beta}, \quad (1)$$

where $\boldsymbol{\gamma}, \boldsymbol{\beta} \in \mathbb{R}^n$ 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 (Nair and Hinton, 2010).

3.4 Embedding Factorization

The embedding table in BERT models accounts for a substantial proportion of model size. To compress the embedding layer, as shown in Table 1, we reduce the embedding dimension to 128 in MobileBERT. Then, we apply a 1D convolution with kernel size 3 on the raw token embedding to produce a 512 dimensional output.

3.5 Training Objectives

We propose to use the following two knowledge transfer objectives, i.e., feature map transfer and attention transfer, to train MobileBERT. Figure 1 illustrates the proposed layer-wise knowledge transfer objectives. Our final layer-wise knowledge transfer loss $\mathcal{L}^{\ell}_{KT}$ for the ℓ^{th} layer is a linear combination of the two objectives stated below:

Feature Map Transfer (FMT) Since each layer in BERT merely takes the output of the previous layer as input, the most important thing in layer-wise knowledge transfer is that the feature maps of each layer should be as close as possible to those of the teacher. In particular, the mean squared error between the feature maps of the MobileBERT student and the IB-BERT teacher is used as the knowledge transfer objective:

$$\mathcal{L}^{\ell}_{FMT} = \frac{1}{TN}\sum_{t=1}^{T}\sum_{n=1}^{N}(H^{tr}_{t,\ell,n} - H^{st}_{t,\ell,n})^2, \quad (2)$$

where ℓ is the index of layers, T is the sequence length, and N is the feature map size. In practice, we find that decomposing this loss term into normalized feature map discrepancy and feature map statistics discrepancy can help stabilize training.

Attention Transfer (AT) The attention mechanism greatly boosts the performance of NLP and becomes a crucial building block in Transformer and BERT (Clark et al., 2019a; Jawahar et al., 2019). This motivates us to use self-attention maps from the well-optimized teacher to help the training of MobileBERT in augmentation to the feature map transfer. In particular, we minimize the KL-divergence between the per-head self-attention distributions of the MobileBERT student and the IB-BERT teacher:

$$\mathcal{L}^{\ell}_{AT} = \frac{1}{TA}\sum_{t=1}^{T}\sum_{a=1}^{A} D_{KL}(a^{tr}_{t,\ell,a} || a^{st}_{t,\ell,a}), \quad (3)$$

where A is the number of attention heads.

Pre-training Distillation (PD) Besides layer-wise knowledge transfer, we can also use a knowledge distillation loss when pre-training MobileBERT. We use a linear combination of the original masked language modeling (MLM) loss, next sentence prediction (NSP) loss, and the new MLM Knowledge Distillation (KD) loss as our pre-training distillation loss:

$$\mathcal{L}_{PD} = \alpha\mathcal{L}_{MLM} + (1-\alpha)\mathcal{L}_{KD} + \mathcal{L}_{NSP}, \quad (4)$$

where α is a hyperparameter in $(0, 1)$.

3.6 Training Strategies

Given the objectives defined above, there can be various combination strategies in training. We discuss three strategies in this paper.

Auxiliary Knowledge Transfer In this strategy, we regard intermediate knowledge transfer as an auxiliary task for knowledge distillation. We use a single loss, which is a linear combination of knowledge transfer losses from all layers as well as the pre-training distillation loss.

[2] A detailed analysis of effectiveness of operational optimizations on real-world inference latency can be found in Section 4.6.1.

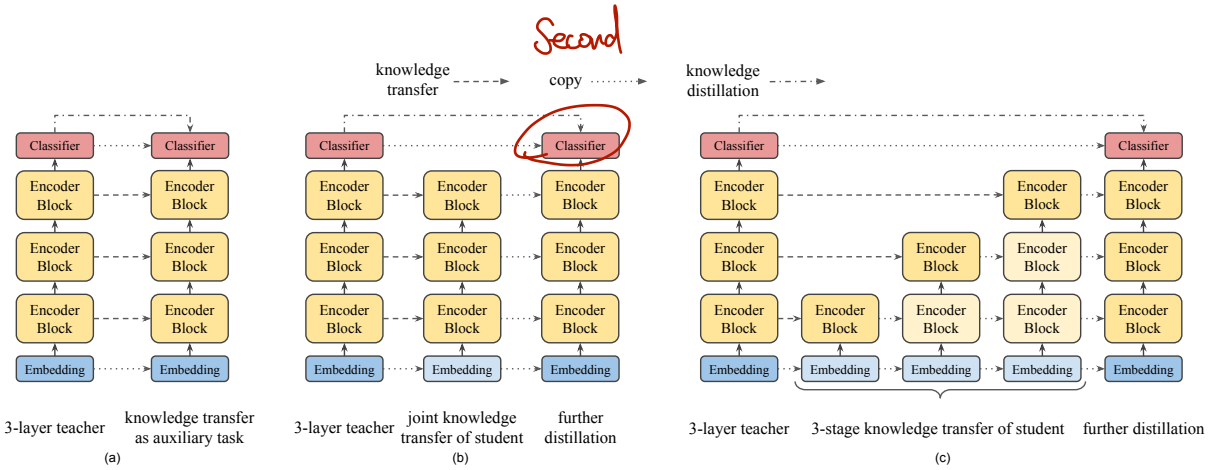

Figure 2: Diagrams of (a) auxiliary knowledge transfer (AKT), (b) joint knowledge transfer (JKT), and (c) progressive knowledge transfer (PKT). Lighter colored blocks represent that they are frozen in that stage.

Joint Knowledge Transfer However, the intermediate knowledge of the IB-BERT teacher (i.e. attention maps and feature maps) may not be an optimal solution for the MobileBERT student. Therefore, we propose to separate these two loss terms, where we first train MobileBERT with all layer-wise knowledge transfer losses jointly, and then further train it by pre-training distillation.

Progressive Knowledge Transfer One may also concern that if MobileBERT cannot perfectly mimic the IB-BERT teacher, the errors from the lower layers may affect the knowledge transfer in the higher layers. Therefore, we propose to progressively train each layer in the knowledge transfer. The progressive knowledge transfer is divided into L stages, where L is the number of layers.

Diagram of three strategies Figure 2 illustrates the diagram of the three strategies. For joint knowledge transfer and progressive knowledge transfer, there is no knowledge transfer for the beginning embedding layer and the final classifier in the layer-wise knowledge transfer stage. They are copied from the IB-BERT teacher to the MobileBERT student. Moreover, for progressive knowledge transfer, when we train the ℓ^{th} layer, we freeze all the trainable parameters in the layers below. In practice, we can 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.

4 Experiments

In this section, we first present our architecture search experiments which lead to the model settings in Table 1, and then present the empirical results on benchmarks from MobileBERT and various baselines.

	#Params	h_{inter}	h_{intra}	#Head	SQuAD
(a)	356M	1024	1024	16	88.2
(b)	325M	768	1024	16	88.6
(c)	293M	512	1024	16	88.1
(d)	276M	384	1024	16	87.6
(e)	262M	256	1024	16	87.0
(f)	293M	512	1024	4	88.3
(g)	92M	512	512	4	85.8
(h)	33M	512	256	4	84.8
(i)	15M	512	128	4	82.0

Table 2: Experimental results on SQuAD v1.1 dev F1 score in search of good model settings for the IB-BERT$_{\text{LARGE}}$ teacher. The number of layers is set to 24 for all models.

4.1 Model Settings

We conduct extensive experiments to search good model settings for the IB-BERT teacher and the MobileBERT student. We start with SQuAD v1.1 dev F1 score as the performance metric in the search of model settings. In this section, 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).

Architecture Search for IB-BERT 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$_{\text{LARGE}}$-sized IB-BERT, and the results are shown in Table 2 with labels (a)-(e). We can see that reducing the inter-block hidden size doesn't damage the performance

h_{intra}	#Head (#Params)	#FFN (#Params)	SQuAD
192	6 (8M)	1 (7M)	82.6
160	5 (6.5M)	2 (10M)	83.4
128	4 (5M)	4 (12.5M)	83.4
96	3 (4M)	8 (14M)	81.6

Table 3: Experimental results on SQuAD v1.1 dev F1 score in search of good model settings for the MobileBERT student. The number of layers is set to 24 and the inter-block hidden size is set to 512 for all models.

of BERT until it is smaller than 512. Hence, we choose IB-BERT$_{\text{LARGE}}$ 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. We conduct experiments and the results are shown in Table 2 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.

Architecture Search for MobileBERT We seek a compression ratio of 4$\times$ for BERT$_{\text{BASE}}$, 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 MobileBERT student model. Table 3 shows our experimental results. They have different balances between MHA and FFN. 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 MHA and FFN to 0.5.

We choose the architecture with 128 intra-block hidden size and 4 stacked FFNs as the MobileBERT 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 layer-wise knowledge transfer. Table 1 demonstrates the model settings of our IB-BERT$_{\text{LARGE}}$ teacher and MobileBERT student.

One may wonder whether reducing the number of heads will harm the performance of the teacher model. By comparing (a) and (f) in Table 2, we can see that reducing the number of heads from 16 to 4 does not affect the performance of IB-BERT$_{\text{LARGE}}$.

4.2 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$_{\text{LARGE}}$ teacher reach the same accuracy as original BERT$_{\text{LARGE}}$, we train IB-BERT$_{\text{LARGE}}$ on 256 TPU v3 chips for 500k steps with a batch size of 4096 and LAMB optimizer (You et al., 2019). For a fair comparison with the original BERT, we do not use training tricks in other BERT variants (Liu et al., 2019b; Joshi et al., 2019). For MobileBERT, we use the same training schedule in the pre-training distillation stage. Additionally, we use progressive knowledge transfer to train MobileBERT, which takes additional 240k steps over 24 layers. In ablation studies, we halve the pre-training distillation schedule of MobileBERT to accelerate experiments. Moreover, in the ablation study of knowledge transfer strategies, for a fair comparison, joint knowledge transfer and auxiliary knowledge transfer also take additional 240k 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) * 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.

4.3 Results on GLUE

The General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018) is a collection of 9 natural language understanding tasks. We compare MobileBERT with BERT$_{\text{BASE}}$ and a few state-of-the-art pre-BERT models on the GLUE leaderboard[3]: OpenAI GPT (Radford et al., 2018) and ELMo (Peters et al., 2018). We also compare with three recently proposed compressed BERT models: BERT-PKD (Sun et al., 2019), and DistilBERT (Sanh et al., 2019). To further show the advantage of MobileBERT over recent small BERT models, we also evaluate a smaller variant of our

[3] https://gluebenchmark.com/leaderboard

	#Params	#FLOPS	Latency	CoLA 8.5k	SST-2 67k	MRPC 3.7k	STS-B 5.7k	QQP 364k	MNLI-m/mm 393k	QNLI 108k	RTE 2.5k	GLUE
ELMo-BiLSTM-Attn	-	-	-	33.6	90.4	84.4	72.3	63.1	74.1/74.5	79.8	58.9	70.0
OpenAI GPT	109M	-	-	47.2	93.1	87.7	84.8	70.1	80.7/80.6	87.2	69.1	76.9
$\text{BERT}_{\text{BASE}}$	109M	22.5B	342 ms	**52.1**	**93.5**	**88.9**	**85.8**	71.2	**84.6/83.4**	90.5	66.4	78.3
$\text{BERT}_{\text{BASE}}$-6L-PKD*	66.5M	11.3B	-	-	92.0	85.0	-	70.7	81.5/81.0	89.0	65.5	-
$\text{BERT}_{\text{BASE}}$-4L-PKD†*	52.2M	7.6B	-	24.8	89.4	82.6	79.8	70.2	79.9/79.3	85.1	62.3	-
$\text{BERT}_{\text{BASE}}$-3L-PKD*	45.3M	5.7B	-	-	87.5	80.7	-	68.1	76.7/76.3	84.7	58.2	-
$\text{DistilBERT}_{\text{BASE}}$-6L†	62.2M	11.3B	-	-	92.0	85.0		70.7	81.5/81.0	89.0	65.5	-
$\text{DistilBERT}_{\text{BASE}}$-4L†	52.2M	7.6B	-	32.8	91.4	82.4	76.1	68.5	78.9/78.0	85.2	54.1	-
TinyBERT*	14.5M	1.2B	-	43.3	92.6	86.4	79.9	**71.3**	82.5/81.8	87.7	62.9	75.4
$\text{MobileBERT}_{\text{TINY}}$	15.1M	3.1B	40 ms	46.7	91.7	87.9	80.1	68.9	81.5/81.6	89.5	65.1	75.8
MobileBERT	25.3M	5.7B	62 ms	50.5	92.8	88.8	84.4	70.2	83.3/82.6	90.6	66.2	77.7
MobileBERT w/o OPT	25.3M	5.7B	192 ms	51.1	92.6	88.8	84.8	70.5	84.3/**83.4**	**91.6**	**70.4**	**78.5**

Table 4: 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). "OPT" denotes the operational optimizations introduced in Section 3.3. †denotes that the results are taken from (Jiao et al., 2019). *denotes that it can be unfair to directly compare MobileBERT with these models since MobileBERT is task-agnostically compressed while these models use the teacher model in the fine-tuning stage.

	#Params	SQuAD v1.1 EM	SQuAD v1.1 F1	SQuAD v2.0 EM	SQuAD v2.0 F1
DocQA + ELMo	-	-	-	65.1	67.6
$\text{BERT}_{\text{BASE}}$	109M	80.8	88.5	74.2†	77.1†
$\text{DistilBERT}_{\text{BASE}}$-6L	66.6M	79.1	86.9	-	-
$\text{DistilBERT}_{\text{BASE}}$-6L‡	66.6M	78.1	86.2	66.0	69.5
$\text{DistilBERT}_{\text{BASE}}$-4L‡	52.2M	71.8	81.2	60.6	64.1
TinyBERT	14.5M	72.7	82.1	65.3	68.8
$\text{MobileBERT}_{\text{TINY}}$	15.1M	81.4	88.6	74.4	77.1
MobileBERT	25.3M	82.9	90.0	76.2	79.2
MobileBERT w/o OPT	25.3M	**83.4**	**90.3**	**77.6**	**80.2**

Table 5: The results on the SQuAD dev datasets. †marks our runs with the official code. ‡denotes that the results are taken from (Jiao et al., 2019).

	MNLI-m	QNLI	MRPC	SST-2	SQuAD
$\text{MobileBERT}_{\text{TINY}}$	**82.0**	**89.9**	**86.7**	**91.6**	**88.6**
+ Quantization	**82.0**	89.8	86.3	**91.6**	88.4
MobileBERT	**83.9**	**91.0**	**87.5**	**92.1**	**90.0**
+ Quantization	**83.9**	90.8	87.0	91.9	**90.0**

Table 6: Results of MobileBERT on GLUE dev accuracy and SQuAD v1.1 dev F1 score with 8-bit Quantization.

model with approximately 15M parameters called $\text{MobileBERT}_{\text{TINY}}$[4], which reduces the number of FFNs in each layer and uses a lighter MHA structure. Besides, to verify the performance of MobileBERT on real-world mobile devices, we export the models with TensorFlow Lite[5] APIs and measure the inference latencies on a 4-thread Pixel 4 phone with a fixed sequence length of 128. The results are listed in Table 4. [6]

From the table, we can see that MobileBERT is very competitive on the GLUE benchmark. MobileBERT achieves an overall GLUE score of 77.7, which is only 0.6 lower than $\text{BERT}_{\text{BASE}}$, while being 4.3× smaller and 5.5× faster than $\text{BERT}_{\text{BASE}}$. Moreover, It outperforms the strong OpenAI GPT baseline by 0.8 GLUE score with 4.3× smaller model size. It also outperforms all the other compressed BERT models with smaller or similar model sizes. Finally, we find that the introduced operational optimizations hurt the model performance a bit. Without these optimizations, MobileBERT can even outperforms $\text{BERT}_{\text{BASE}}$ by 0.2 GLUE score.

4.4 Results on SQuAD

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. 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 $\text{BERT}_{\text{BASE}}$, DistilBERT, and a strong baseline DocQA (Clark and Gardner, 2017).

[4]The detailed model setting of $\text{MobileBERT}_{\text{TINY}}$ can be found in Table 1 and in the appendix.

[5]https://www.tensorflow.org/lite

[6]We follow Devlin et al. (2018) to skip the WNLI task.

Setting	#FLOPS	Latency
`LayerNorm` & `gelu`	5.7B	192 ms
`LayerNorm` & `relu`	5.7B	167 ms
`NoNorm` & `gelu`	5.7B	92 ms
`NoNorm` & `relu`	5.7B	62 ms

Table 7: The effectiveness of operational optimizations on real-world inference latency for MobileBERT.

	MNLI-m	QNLI	MRPC	SST-2	SQuAD
AKT	83.0	90.3	86.8	91.9	88.2
JKT	83.5	90.5	**87.5**	92.0	89.7
PKT	**83.9**	**91.0**	**87.5**	**92.1**	**90.0**

Table 8: Ablation study of MobileBERT on GLUE dev accuracy and SQuAD v1.1 dev F1 score with Auxiliary Knowledge Transfer (AKT), Joint Knowledge Transfer (JKT), and Progressive Knowledge Transfer (PKT).

As shown in Table 5, MobileBERT outperforms a large margin over all the other models with smaller or similar model sizes.

4.5 Quantization

We apply the standard post-training quantization in TensorFlow Lite to MobileBERT. The results are shown in Table 6. We find that while quantization can further compress MobileBERT by 4×, there is nearly no performance degradation from it. This indicates that there is still a big room in the compression of MobileBERT.

4.6 Ablation Studies

4.6.1 Operational Optimizations

We evaluate the effectiveness of the two operational optimizations introduced in Section 3.3, i.e., replacing layer normalization (`LayerNorm`) with `NoNorm` and replacing `gelu` activation with `relu` activation. We report the inference latencies using the same experimental setting as in Section 4.6.1. From Table 7, we can see that both `NoNorm` and `relu` are very effective in reducing the latency of MobileBERT, while the 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).

4.6.2 Training Strategies

We also study how the choice of training strategy, i.e., auxiliary knowledge transfer, joint knowledge transfer, and progressive knowledge transfer, can affect the performance of MobileBERT. As shown in Table 8, progressive knowledge transfer consistently outperforms the other two strategies. We notice that there is a significant performance gap between auxiliary knowledge transfer and the other two strategies. We think the reason is that the intermediate layer-wise knowledge (i.e., attention maps and feature maps) from the teacher may not be optimal for the student, so the student needs an additional pre-training distillation stage to fine-tune its parameters.

	MNLI-m	QNLI	MRPC	SST-2
$\text{BERT}_{\text{LARGE}}$	86.6	92.1†	**87.8**	93.7
$\text{IB-BERT}_{\text{LARGE}}$	**87.0**	**93.2**	87.3	**94.1**
$\text{BERT}_{\text{BASE}}$	**84.4**	91.1†	86.7	**92.9**
MobileBERT (bare)	80.8	88.2	84.3	90.1
+ PD	81.1	88.9	85.5	91.7
+ PD + FMT	83.8	91.1	**87.0**	92.2
+ PD + FMT + AT	**84.4**	**91.5**	**87.0**	92.5

Table 9: Ablation on the dev sets of GLUE benchmark. $\text{BERT}_{\text{BASE}}$ 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.

4.6.3 Training Objectives

We finally 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 to make a fair comparison between MobileBERT and the original BERT. The results are listed in Table 9.

We can see that the proposed Feature Map Transfer contributes most to the performance improvement of MobileBERT, while Attention Transfer and Pre-training Distillation also play positive roles. We can also find that our $\text{IB-BERT}_{\text{LARGE}}$ teacher is as powerful as the original $\text{IB-BERT}_{\text{LARGE}}$ while MobileBERT degrades greatly when compared to its teacher. So we believe that there is still a big room in the improvement of MobileBERT.

5 Conclusion

We have presented MobileBERT which is a task-agnostic compact variant of BERT. Empirical results on popular NLP benchmarks show that MobileBERT is comparable with $\text{BERT}_{\text{BASE}}$ while being much smaller and faster. MobileBERT can

enable various NLP applications[7] to be easily deployed on mobile devices.

In this paper, we show that 1) it is crucial to keep MobileBERT deep and thin, 2) bottleneck/inverted-bottleneck structures enable effective layer-wise knowledge transfer, and 3) progressive knowledge transfer can efficiently train MobileBERT. We believe our findings are generic and can be applied to other model compression problems.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo. Magnini. 2009. The fifth PASCAL recognizing textual entailment challenge. *TAC*.

Cristian Bucilu, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541. ACM.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

Z. Chen, H. Zhang, X. Zhang, and L. Zhao. 2018. Quora question pairs. *Quora*.

Christopher Clark and Matt Gardner. 2017. Simple and effective multi-paragraph reading comprehension. *arXiv preprint arXiv:1710.10723*.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019a. What does bert look at? an analysis of bert's attention. *arXiv preprint arXiv:1906.04341*.

Kevin Clark, Minh-Thang Luong, Urvashi Khandelwal, Christopher D Manning, and Quoc V Le. 2019b. Bam! born-again multi-task networks for natural language understanding. *arXiv preprint arXiv:1907.04829*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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

[7] https://tensorflow.org/lite/models/bert_qa/overview

Fei Gao, Lijun Wu, Li Zhao, Tao Qin, Xueqi Cheng, and Tie-Yan Liu. 2018. Efficient sequence learning with group recurrent networks. 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 799–808.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *arXiv*.

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

Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. 2019. Searching for mobilenetv3. *arXiv preprint arXiv:1905.02244*.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. 2016. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*.

Ganesh Jawahar, Benoît Sagot, Djamé Seddah, Samuel Unicomb, Gerardo Iñiguez, Márton Karsai, Yannick Léo, Márton Karsai, Carlos Sarraute, Éric Fleury, et al. 2019. What does bert learn about the structure of language? In *57th Annual Meeting of the Association for Computational Linguistics (ACL), Florence, Italy*.

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

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2019. Spanbert: Improving pre-training by representing and predicting spans. *arXiv preprint arXiv:1907.10529*.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*.

Oleksii Kuchaiev and Boris Ginsburg. 2017. Factorization tricks for lstm networks. *arXiv preprint arXiv:1703.10722*.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Hector J Levesque, Ernest Davis, and Leora. Morgenstern. 2011. The Winograd schema challenge. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning.*, volume 46, page 47.

Zhuohan Li, Di He, Fei Tian, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2019. Hint-based training for non-autoregressive translation.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.

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

Ping Luo, Zhenyao Zhu, Ziwei Liu, Xiaogang Wang, and Xiaoou Tang. 2016. Face model compression by distilling knowledge from neurons. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Brian W Matthews. 1975. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*.

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

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2014. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520.

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

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 EMNLP.*, pages 1631–1642.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for bert model compression. *arXiv preprint arXiv:1908.09355*.

Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. 2019. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2820–2828.

Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. 2019. Distilling task-specific knowledge from bert into simple neural networks. *arXiv preprint arXiv:1903.12136*.

Henry Tsai, Jason Riesa, Melvin Johnson, Naveen Arivazhagan, Xin Li, and Amelia Archer. 2019. Small and practical bert models for sequence labeling. *arXiv preprint arXiv:1909.00100*.

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

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2018. Neural network acceptability judgments. *arXiv preprint 1805.12471*.

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

Xing Wu, Shangwen Lv, Liangjun Zang, Jizhong Han, and Songlin Hu. 2019. Conditional bert contextual augmentation. In *International Conference on Computational Science*, pages 84–95. Springer.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Doyeob Yeo, Ji-Roon Bae, Nae-Soo Kim, Cheol-Sig Pyo, Junho Yim, and Junmo Kim. 2018. Sequential knowledge transfer in teacher-student framework using densely distilled flow-based information. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 674–678. IEEE.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, and Cho-Jui Hsieh. 2019. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*.

Sergey Zagoruyko and Nikos Komodakis. 2016. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*.

Ting Zhang, Guo-Jun Qi, Bin Xiao, and Jingdong Wang. 2017. Interleaved group convolutions. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4373–4382.

Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. 2018. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6848–6856.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

Appendix for "MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices"

A Extra Related Work on Knowledge Transfer

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 and 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.

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 and Ginsburg, 2017; Gao et al., 2018), which employs the idea of group convolution (Zhang et al., 2017, 2018) into Recurrent Neural Networks (RNN).

C Visualization of Attention Distributions

We visualize the attention distributions of the 1^{st} and the 12^{th} layers of a few models in the ablation study for further investigation. They are shown in Figure 3. We find that 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 the ablation study table, since **the alignment of feature maps lead to the alignment of attention distributions**.

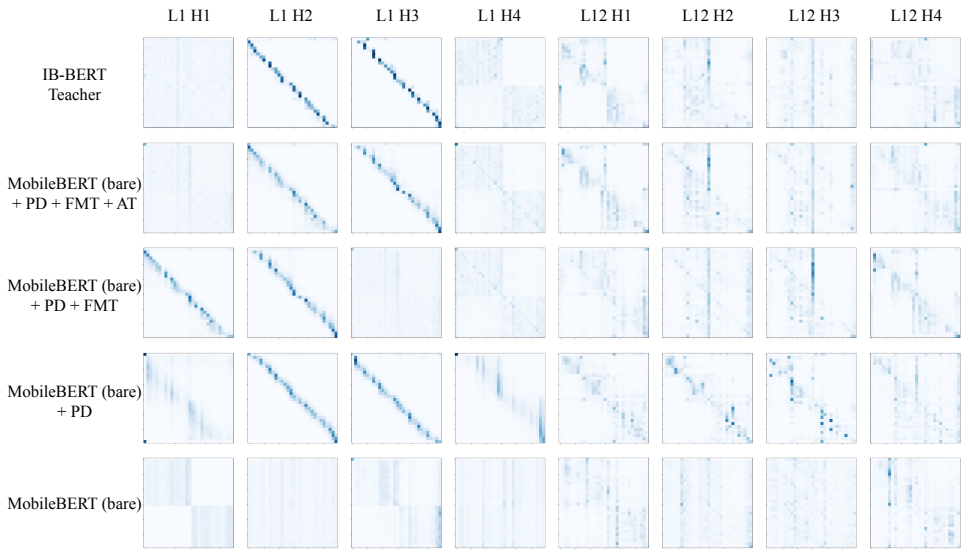

Figure 3: The visualization of the attention distributions in some attention heads of the IB-BERT teacher and different MobileBERT models.

D Extra Experimental Settings

For a fair comparison with original BERT, we follow the same pre-processing scheme as BERT, where we mask 15% of all WordPiece (Kudo and Richardson, 2018) 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., 2019b). We leave it for future work.

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

E Architecture of MobileBERT$_{\text{TINY}}$

We use a lighter MHA structure for MobileBERT$_{\text{TINY}}$. As illustrated in Figure 4, in stead of using hidden states from the inter-block feature maps as inputs to MHA, we use the reduced intra-block feature maps as key, query, and values in MHA for MobileBERT$_{\text{TINY}}$. This can effectively reduce the parameters in MHA modules, but might harm the model capacity.

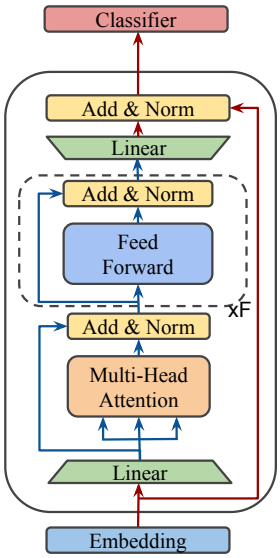

Figure 4: Illustration of MobileBERT$_{\text{TINY}}$. red lines denote inter-block flows while blue lines intra-block flows.

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 and 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[8] (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.

[8] https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs