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 $\text{BERT}_{\text{LARGE}}$ 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 $\text{BERT}_{\text{LARGE}}$ teacher to it. Empirical studies show that MobileBERT is $4.3\times$ smaller and $4.0\times$ faster than original $\text{BERT}_{\text{BASE}}$ 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 $\text{BERT}_{\text{BASE}}$.

1 INTRODUCTION

The NLP community has witnessed a revolution of pre-training self-supervised models. These models usually have hundreds of millions of parameters. 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 $\text{BERT}_{\text{LARGE}}$ while each layer is made much narrower via adopting bottleneck structures and balancing between self-attentions and feed-

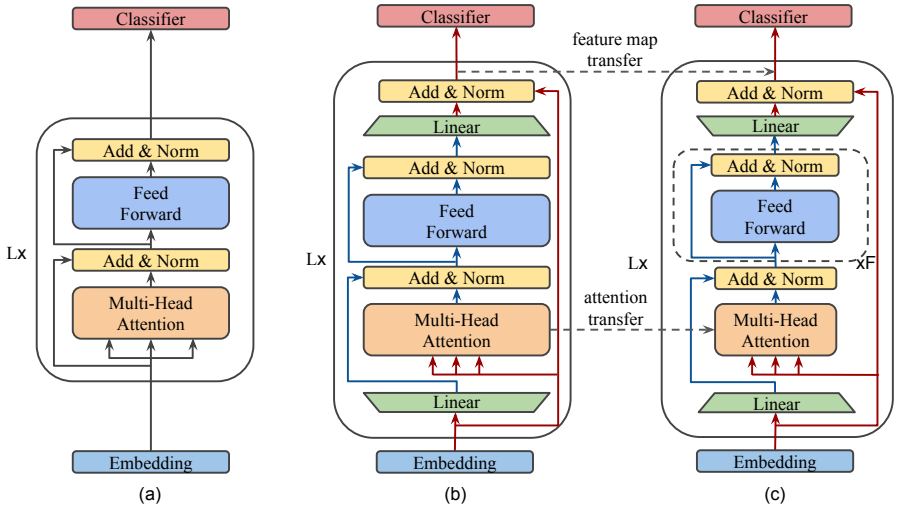


Figure 1: Illustration of three models: (a) BERT; (b) Inverted Bottleneck BERT (IB-BERT); and (c) MobileBERT. In (b) and (c), red lines denote inter-block flows while blue lines intra-block flows. MobileBERT is trained by progressively transferring knowledge from IB-BERT.

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 $\text{BERT}_{\text{LARGE}}$ (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× smaller and 4.0× faster, MobileBERT can still achieve competitive results compared to $\text{BERT}_{\text{BASE}}$. 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 $\text{BERT}_{\text{BASE}}$.

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 down-stream tasks. Moreover, compared to patient knowledge distillation (Sun et al., 2019) which transfers knowledge for all intermediate layers simultaneously to alleviate over-fitting in down-stream 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 ℓ^{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:

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

where ℓ 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.

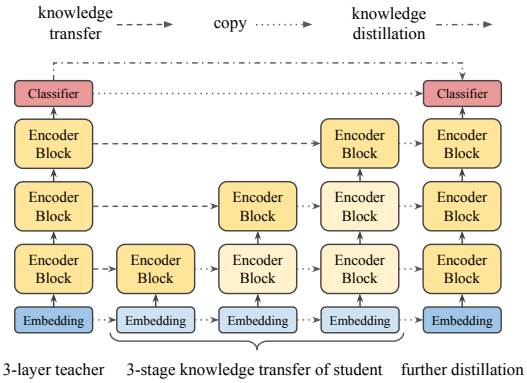
We also minimize two statistics discrepancies on mean and variance in feature map transfer:

$$\mathcal{L}^{\ell}_{\mu} = \frac{1}{T}\sum_{t=1}^{T}(\mu(H^{tr}_{t,\ell}) - \mu(H^{st}_{t,\ell}))^2, \quad \mathcal{L}^{\ell}_{\sigma} = \frac{1}{T}\sum_{t=1}^{T}|\sigma^2(H^{tr}_{t,\ell}) - \sigma^2(H^{st}_{t,\ell})|, \tag{2}$$

where μ and σ^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



Input : the teacher model M_{tr}
Output: the student model M_{st}
`// Copy the embedding layerd`
Embedding(M_{st}) $\leftarrow$ Embedding(M_{tr});
`// Progressive knowledge transfer`
for each layer ℓ in M_{st} **do**
  $M_{st} \leftarrow$ Train(M_{st}, $\mathcal{L}^{\ell}_{PKT}(M_{st}, M_{tr})$)
`// Copy the classifier layer`
Classifier(M_{st}) $\leftarrow$ Classifier(M_{tr});
`// Further pre-training distillation`
$M_{st} \leftarrow$ Train(M_{st}, $\mathcal{L}_{PD}(M_{st}, M_{tr})$);
return M_{st}

Figure 2: Left panel: diagram of progressive knowledge transfer. Lighter colored blocks represent that they are frozen in that stage. Right panel: algorithm of progressive knowledge transfer.

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:

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

where A is the number of attention heads.

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

Our final progressive knowledge transfer loss $\mathcal{L}^{\ell}_{PKT}$ for the ℓ^{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 ℓ^{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 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:

$$\mathcal{L}_{MLM-KD} = \sum_{i\in[N]} D_{KL}(P^{tr}(i), P^{st}(i)), \tag{4}$$

$$\mathcal{L}_{PD} = \alpha\mathcal{L}_{MLM} + (1-\alpha)\mathcal{L}_{MLM-KD} + \mathcal{L}_{NSP}, \tag{5}$$

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 α 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 $\text{BERT}_{\text{LARGE}}$. 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 $\text{BERT}_{\text{LARGE}}$ teacher (Figure 1b). We refer this variant of $\text{BERT}_{\text{LARGE}}$ 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[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 $\mathbf{h}$ with an element-wise linear transformation:

$$\texttt{NoNorm}(\mathbf{h}) = \boldsymbol{\gamma} \circ \mathbf{h} + \boldsymbol{\beta}, \tag{6}$$

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.

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).

[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$_{\text{LARGE}}$ 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.

	#Params	$\mathbf{h_{inter}}$	$\mathbf{h_{intra}}$	**#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

$\mathbf{h_{intra}}$	**#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 2: The detailed model settings of a few models. $\mathcal{L}$, $\mathbf{h_{inter}}$, $\mathbf{h_{intra}}$, $\mathbf{h_{FFN}}$, $\mathbf{h_{embedding}}$, **#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.

	$\mathcal{L}$	$\mathbf{h_{inter}}$	$\mathbf{h_{intra}}$	$\mathbf{h_{FFN}}$	$\mathbf{h_{embedding}}$	**#Head**	**#FFN**	**#Params**
BERT$_{\text{LARGE}}$	24	1024	1024	4096	1024	16	1	334M
BERT$_{\text{BASE}}$	12	768	768	3072	768	12	1	109M
BERT$_{\text{SMALL}^*}$	12	384	384	1536	128	12	1	25.3M
IB-BERT$_{\text{LARGE}}$	24	512	1024	4096	128	4	1	293M
MobileBERT	24	512	128	512	128	4	4	25.3M

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$_{\text{LARGE}}$-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$_{\text{LARGE}}$ 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, 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$_{\text{BASE}}$, 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 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$_{\text{LARGE}}$ 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.

	#Params	#FLOPS	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m/mm	QNLI	RTE	GLUE
			8.5k	67k	3.7k	5.7k	364k	393k	108k	2.5k	
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	**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	-
$\text{BERT}_{\text{BASE}}$-3L-PKD	45.3M	5.7B	-	87.5	80.7	-	68.1	76.7/76.3	84.7	58.2	-
MobileBERT	25.3M	5.7B	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	51.1	92.6	88.8	84.8	70.5	84.3/**83.4**	**91.6**	**70.4**	**78.5**

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-$\text{BERT}_{\text{LARGE}}$ teacher reach the same accuracy as original $\text{BERT}_{\text{LARGE}}$, we train IB-$\text{BERT}_{\text{LARGE}}$ on 256 TPU v3 chips for 500k steps with a batch size of 4096 and LAMB optimizer (You et al., 2019). For 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) * 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[2] of GLUE to get the test results. We compare MobileBERT with $\text{BERT}_{\text{BASE}}$ 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.[3]

We can see that our MobileBERT is quite competitive with the original $\text{BERT}_{\text{BASE}}$. It outperforms $\text{BERT}_{\text{BASE}}$ 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 $\text{BERT}_{\text{BASE}}$ 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.

[2] `https://gluebenchmark.com/leaderboard`

[3] 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.

	#Params	#FLOPS	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	71.2B	80.8	88.5	74.2†	77.1†
MobileBERT	25.3M	18.2B	82.9	90.0	76.2	79.2
MobileBERT w/o OPT	25.3M	18.2B	**83.4**	**90.3**	**77.6**	**80.2**

Table 5: Ablation on the dev sets of GLUE benchmark. $\text{BERT}_{\text{BASE}}$, $\text{BERT}_{\text{SMALL}}*$, 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.

	#Params	#FLOPS	Latency	MNLI-m	QNLI	MRPC	SST-2
$\text{BERT}_{\text{LARGE}}$ (Devlin et al., 2018)	334M	79.2B	5023 ms	86.6	92.1†	**87.8**	93.7
IB-$\text{BERT}_{\text{LARGE}}$	293M	75.8B	5077 ms	**87.0**	**93.2**	87.3	**94.1**
$\text{BERT}_{\text{BASE}}$ (Devlin et al., 2018)	109M	22.5B	1468 ms	**84.4**	91.1†	86.7	**92.9**
$\text{BERT}_{\text{MEDIUM}}$ (Turc et al., 2019)	41.1M	6.8B	-	81.2	-	-	89.6
$\text{BERT}_{\text{SMALL}}$ (Turc et al., 2019)	28.2M	3.4B	-	78.8	-	-	88.4
$\text{BERT}_{\text{SMALL}}*$	25.3M	5.8B	466 ms	80.2	88.1	84.1	89.8
MobileBERT (bare)	25.3M	5.7B	620 ms	80.8	88.2	84.3	90.1
+ PD	25.3M	5.7B	620 ms	81.1	88.9	85.5	91.7
+ PD + FMT	25.3M	5.7B	620 ms	83.8	91.1	87.0	92.2
+ PD + FMT + AT	25.3M	5.7B	620 ms	**84.4**	**91.5**	87.0	92.5
+ PD + FMT + AT + OPT	25.3M	5.7B	367 ms	83.9	91.0	**87.5**	92.1

We evaluate MobileBERT only on the SQuAD dev datasets, as there is nearly no single model submission on SQuAD test leaderboard[4]. We compare our MobileBERT with $\text{BERT}_{\text{BASE}}$ and a strong baseline DocQA (Clark & Gardner, 2017). As shown in Table 4, MobileBERT outperforms a large margin over $\text{BERT}_{\text{BASE}}$ and DocQA. We notice that MobileBERT also outperforms $\text{BERT}_{\text{BASE}}$ 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 $\text{BERT}_{\text{SMALL}}*$, which is the best model setting we can find with roughly 25M parameters under the original BERT architecture. The detailed model setting of $\text{BERT}_{\text{SMALL}}*$ 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[5] 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 $1.68\times$. In architecture comparison, we find that although specifically designed for progressive knowledge transfer, our MobileBERT architecture alone is still quite competitive. It outperforms $\text{BERT}_{\text{SMALL}}*$ and $\text{BERT}_{\text{SMALL}}$ on all compared tasks, while outperforming the $1.7\times$ sized $\text{BERT}_{\text{MEDIUM}}$ on the SST-2 task. Finally, we can

[4] https://rajpurkar.github.io/SQuAD-explorer/

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

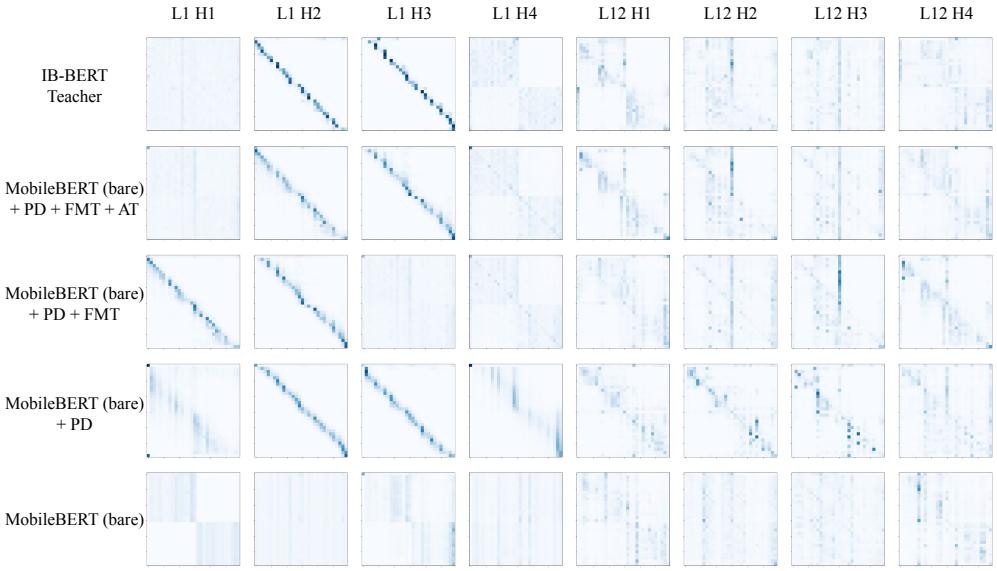


Figure 3: The visualization of the attention distributions in some attention heads of the IB-BERT teacher and different MobileBERT models. We provide the visualization of attention distributions of all 24 layer in Appendix G to readers for further details.

find that although augmented with the powerful progressive knowledge transfer, our MobileBERT still degrades greatly when compared to the IB-BERT$_{\text{LARGE}}$ teacher.

We visualize the attention distributions of the 1^{st} and the 12^{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

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pp. 153–160, 2007.

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

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

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

Z. Chen, H. Zhang, X. Zhang, and L. Zhao. Quora question pairs. *Quora*, 2018. URL `https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs`.

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

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

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

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

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Fei Gao, Lijun Wu, Li Zhao, Tao Qin, Xueqi Cheng, and Tie-Yan Liu. 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)*, pp. 799–808, 2018.

Tobias Glasmachers. Limits of end-to-end learning. *arXiv preprint arXiv:1704.08305*, 2017.

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

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

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

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

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

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

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

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

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

Valentin Khrulkov, Oleksii Hrinchuk, Leyla Mirvakhabova, and Ivan Oseledets. Tensorized embedding layers for efficient model compression. *arXiv preprint arXiv:1901.10787*, 2019.

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

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

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

Guillaume Lample, Alexandre Sablayrolles, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Large memory layers with product keys. *CoRR*, abs/1907.05242, 2019. URL `http://arxiv.org/abs/1907.05242`.

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

Zhuohan Li, Di He, Fei Tian, Tao Qin, Liwei Wang, and Tie-Yan Liu. Hint-based training for non-autoregressive translation, 2019. URL `https://openreview.net/forum?id=r1gGpjActQ`.

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

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

Xindian Ma, Peng Zhang, Shuai Zhang, Nan Duan, Yuexian Hou, Dawei Song, and Ming Zhou. A tensorized transformer for language modeling. *arXiv preprint arXiv:1906.09777*, 2019.

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

Avner May, Jian Zhang, Tri Dao, and Christopher Ré. On the downstream performance of compressed word embeddings. *arXiv preprint arXiv:1909.01264*, 2019.

Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? *arXiv preprint arXiv:1905.10650*, 2019.

Seyed-Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant: Bridging the gap between student and teacher. *arXiv preprint arXiv:1902.03393*, 2019.

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

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

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

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

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

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

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

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. Q-bert: Hessian based ultra low precision quantization of bert. *arXiv preprint arXiv:1909.05840*, 2019.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP.*, pp. 1631–1642, 2013.

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

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

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

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

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103. ACM, 2008.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*, 2019.

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

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

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

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

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

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

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

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

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

Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 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*, pp. 19–27, 2015.

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 5.5, 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.

	Setting	**#FLOPS**	**Latency**
MobileBERT	`LayerNorm` & `gelu`	5.7B	620 ms
	`NoNorm` & `gelu`	5.7B	449 ms
	`LayerNorm` & `relu`	5.7B	538 ms
	`NoNorm` & `relu`	5.7B	367 ms

D EXTRA EXPERIMENTAL SETTINGS

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

$$\mathcal{L}^{\ell}_{PKT} = \lambda\mathcal{L}^{\ell}_{AT} + \mu\mathcal{L}^{\ell}_{FMT} + \beta\mathcal{L}^{\ell}_{\mu} + \gamma\mathcal{L}^{\ell}_{\sigma} \tag{7}$$

$(\lambda, \mu, \beta, \gamma)$ 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 (Kudo & 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., 2019). 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 & Rush, 2016), we set α to 0.5.

E COMPARISON WITH DISTILBERT

We notice that recently there is an unpublished work[6] that also propose a task-agnosticly compressed BERT, called DistilBERT. Basically, DistilBERT is a 6-layer truncated $\text{BERT}_{\text{BASE}}$, which is distilled from $\text{BERT}_{\text{BASE}}$ 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.

[6]It is actually a blog: "Smaller, faster, cheaper, lighter: Introducing DistilBERT, a distilled version of BERT", `https://medium.com/huggingface/distilbert-8cf3380435b5`, Sept. 23, 2019.

QQP The Quora Question Pairs[7] (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).

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

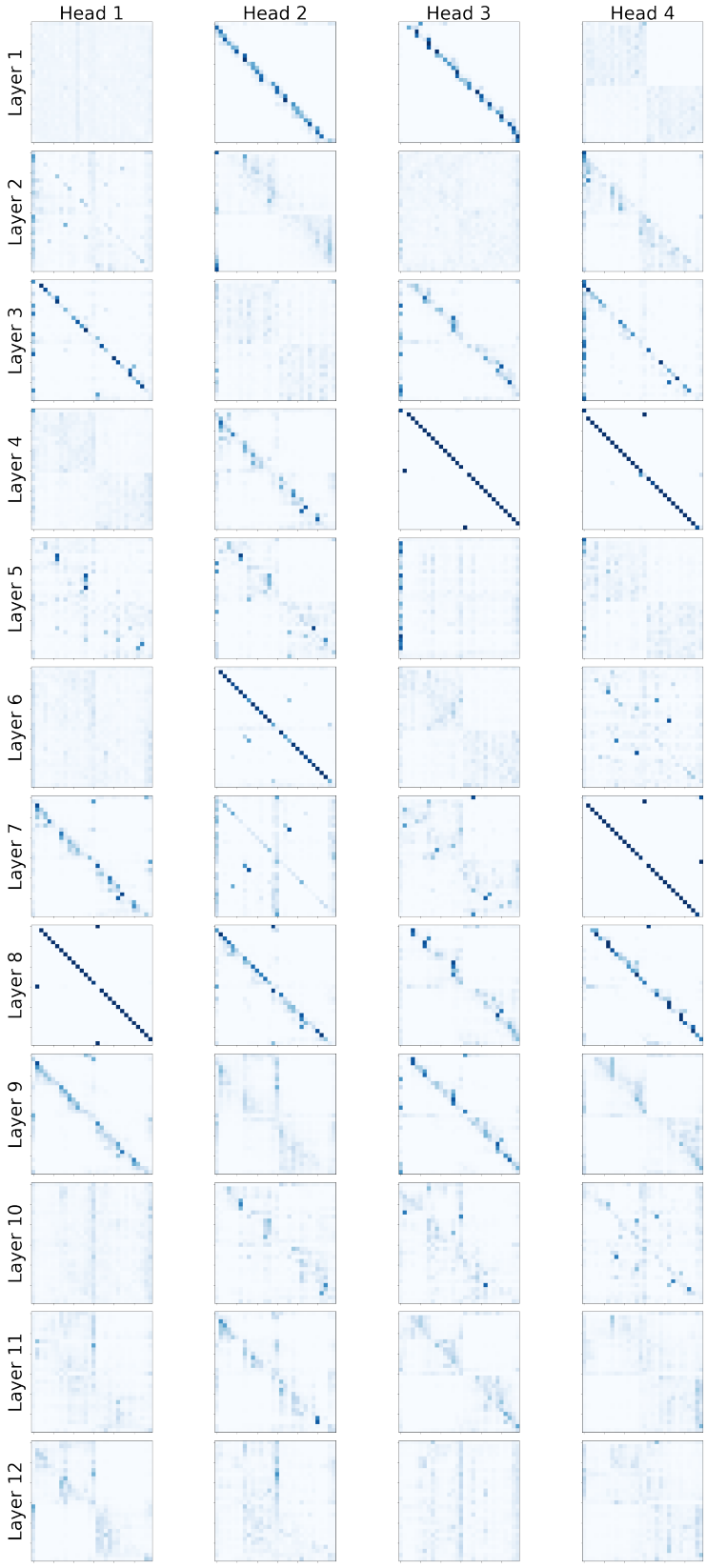


Figure 4: The visualization of attention distributions in the first 12 layers of IB-BERT teacher.

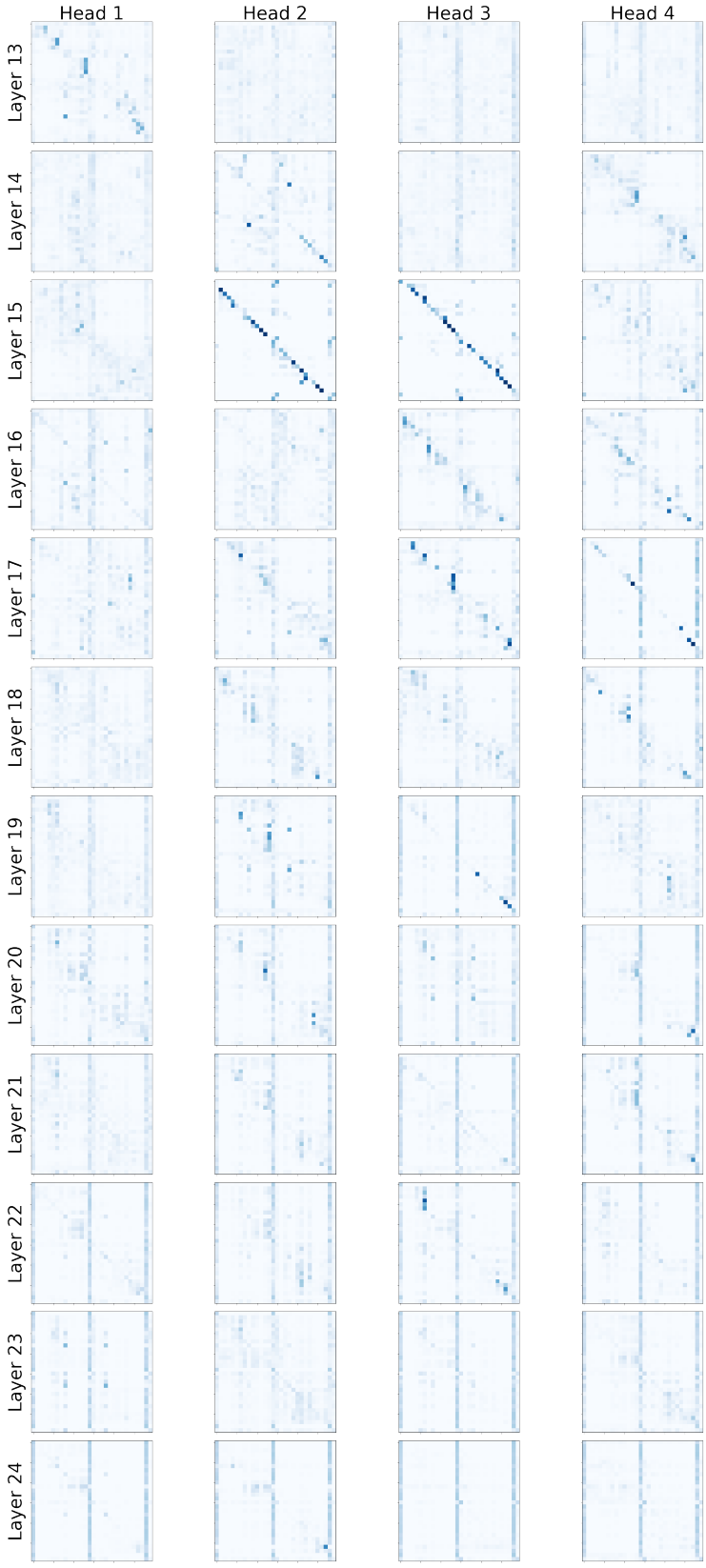


Figure 5: The visualization of attention distributions in the last 12 layers of IB-BERT teacher.

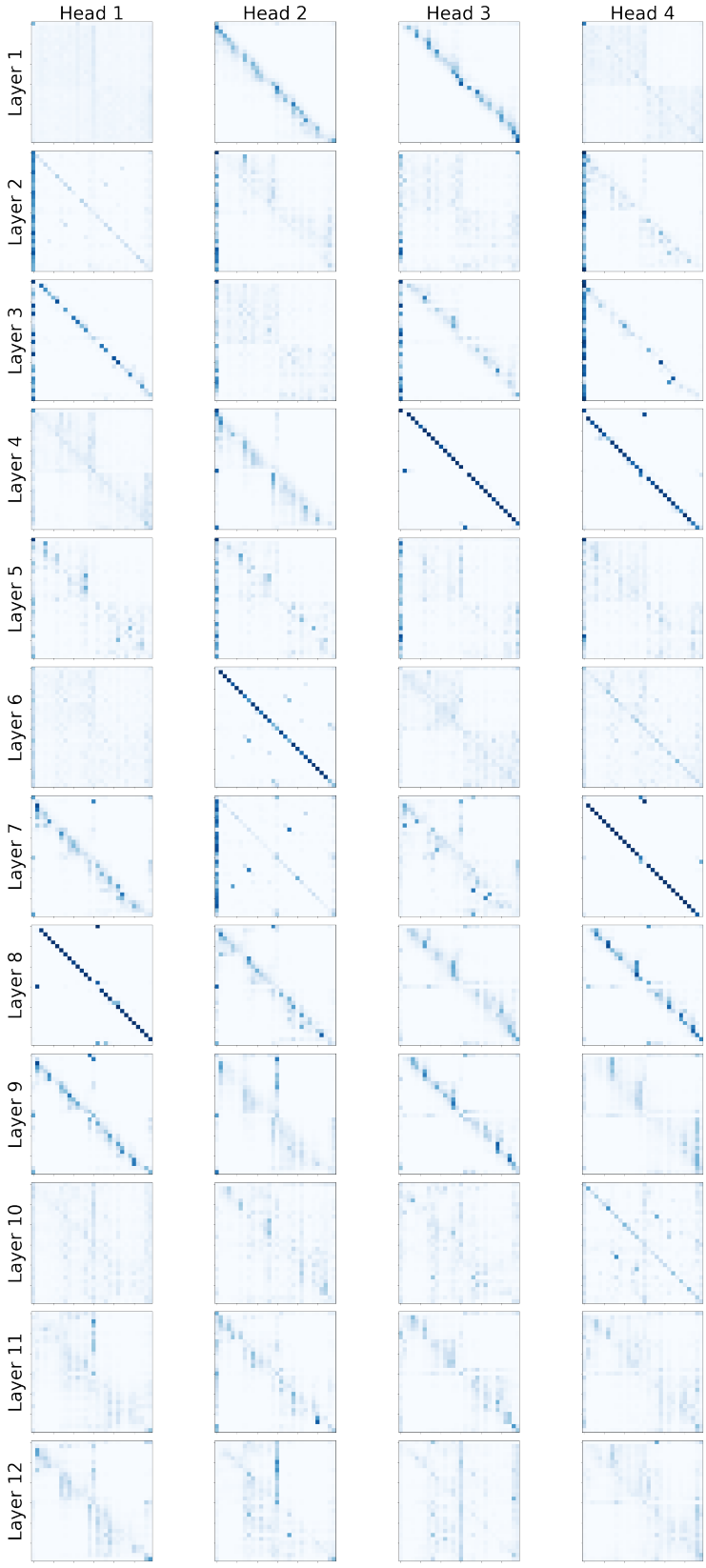


Figure 6: The visualization of attention distributions in the first 12 layers of MobileBERT (bare) + PD + FMT + AT.

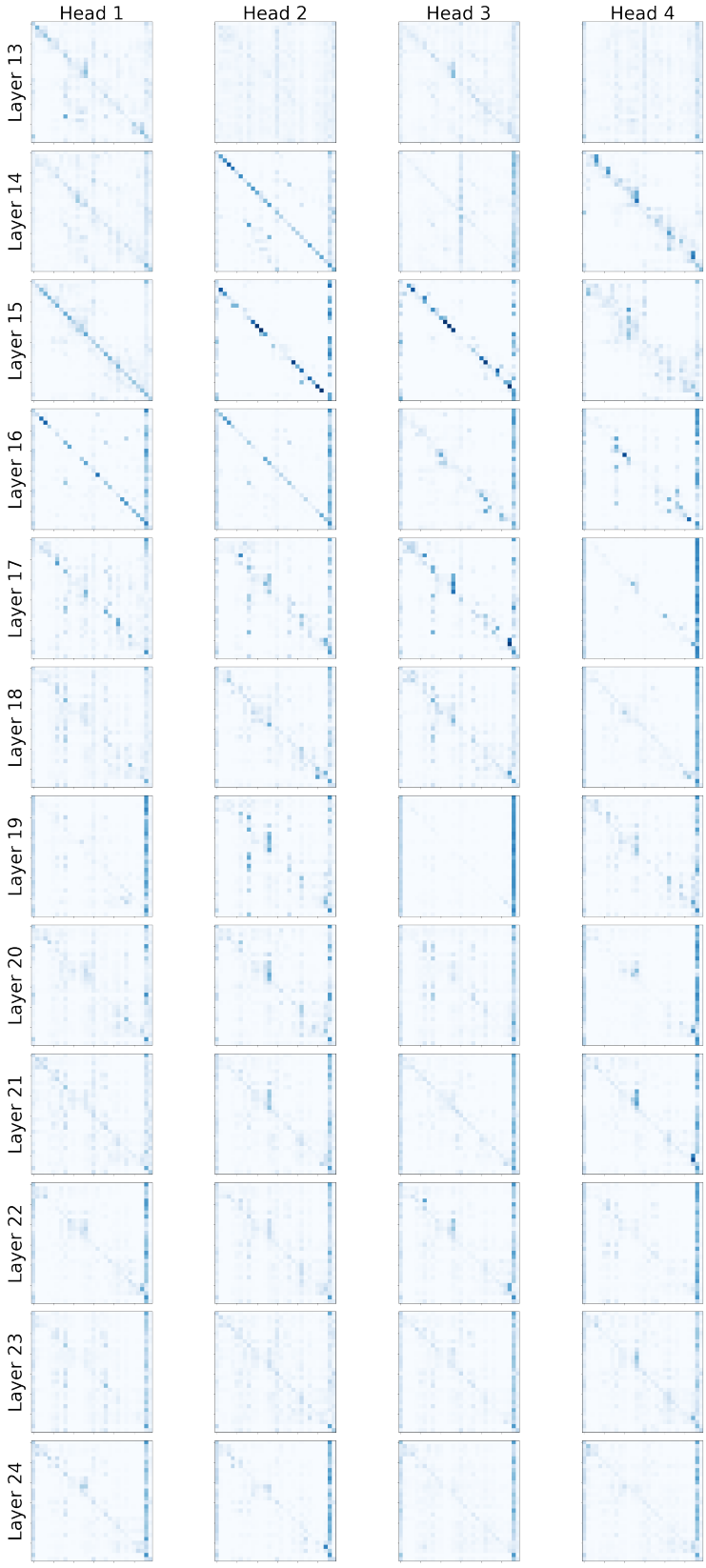


Figure 7: The visualization of attention distributions in the last 12 layers of MobileBERT (bare) + PD + FMT + AT.

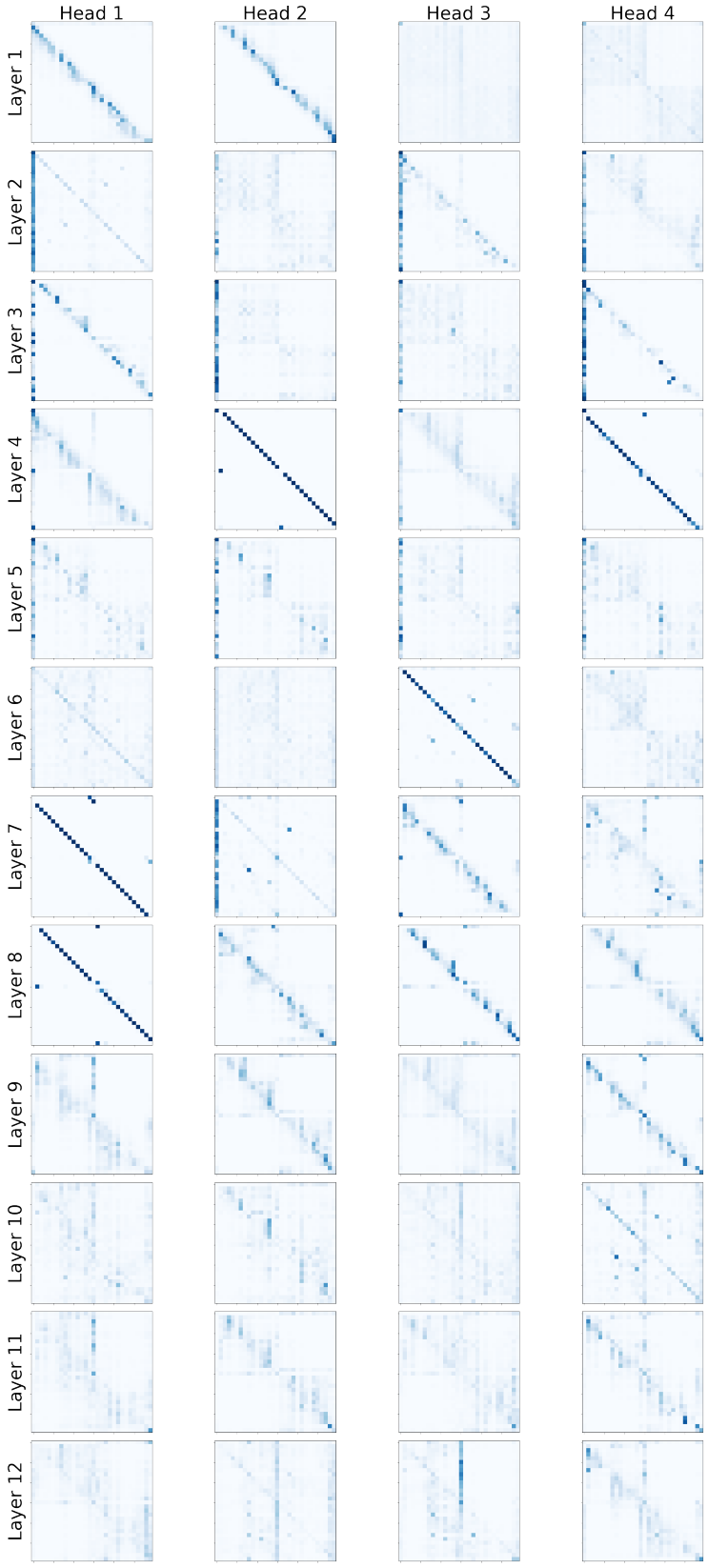


Figure 8: The visualization of attention distributions in the first 12 layers of MobileBERT (bare) + PD + FMT.

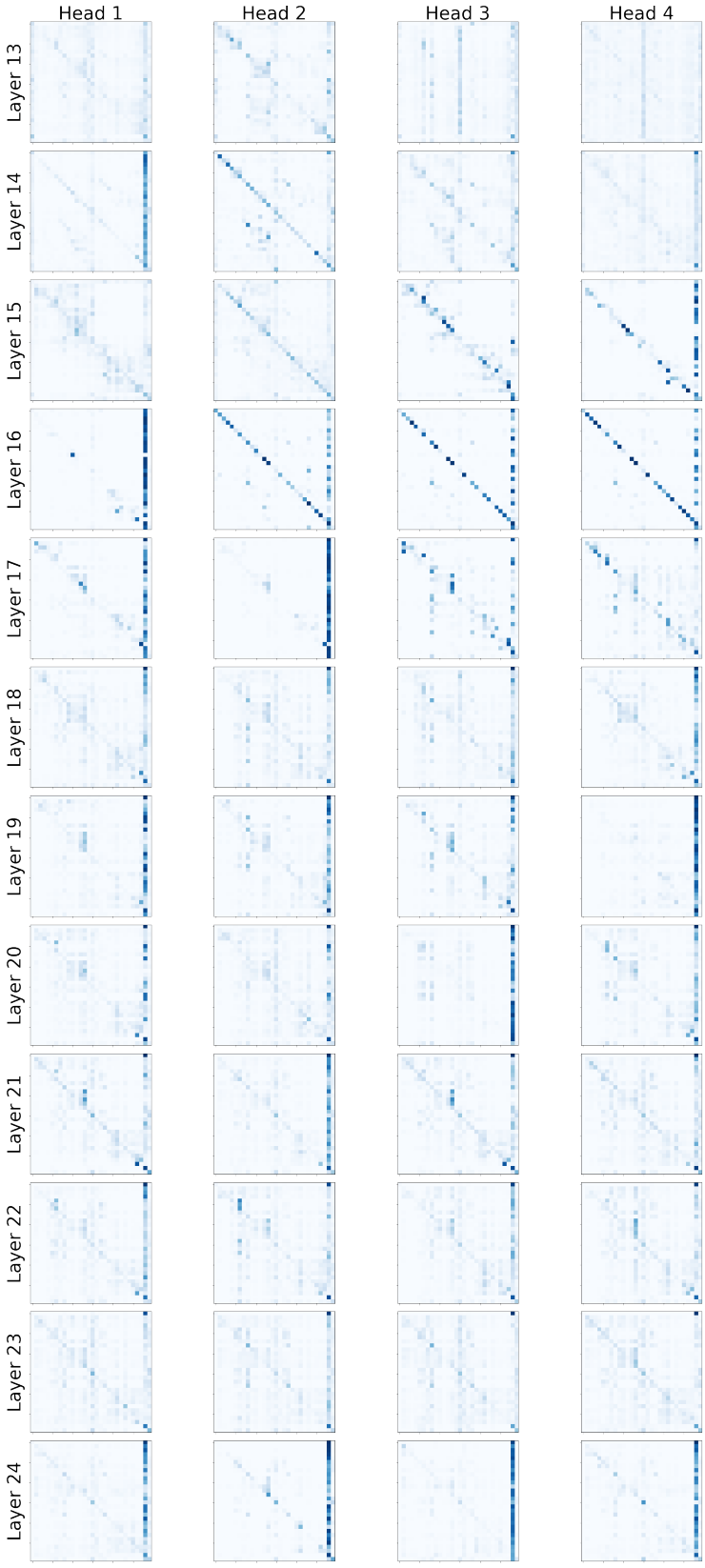


Figure 9: The visualization of attention distributions in the last 12 layers of MobileBERT (bare) + PD + FMT.

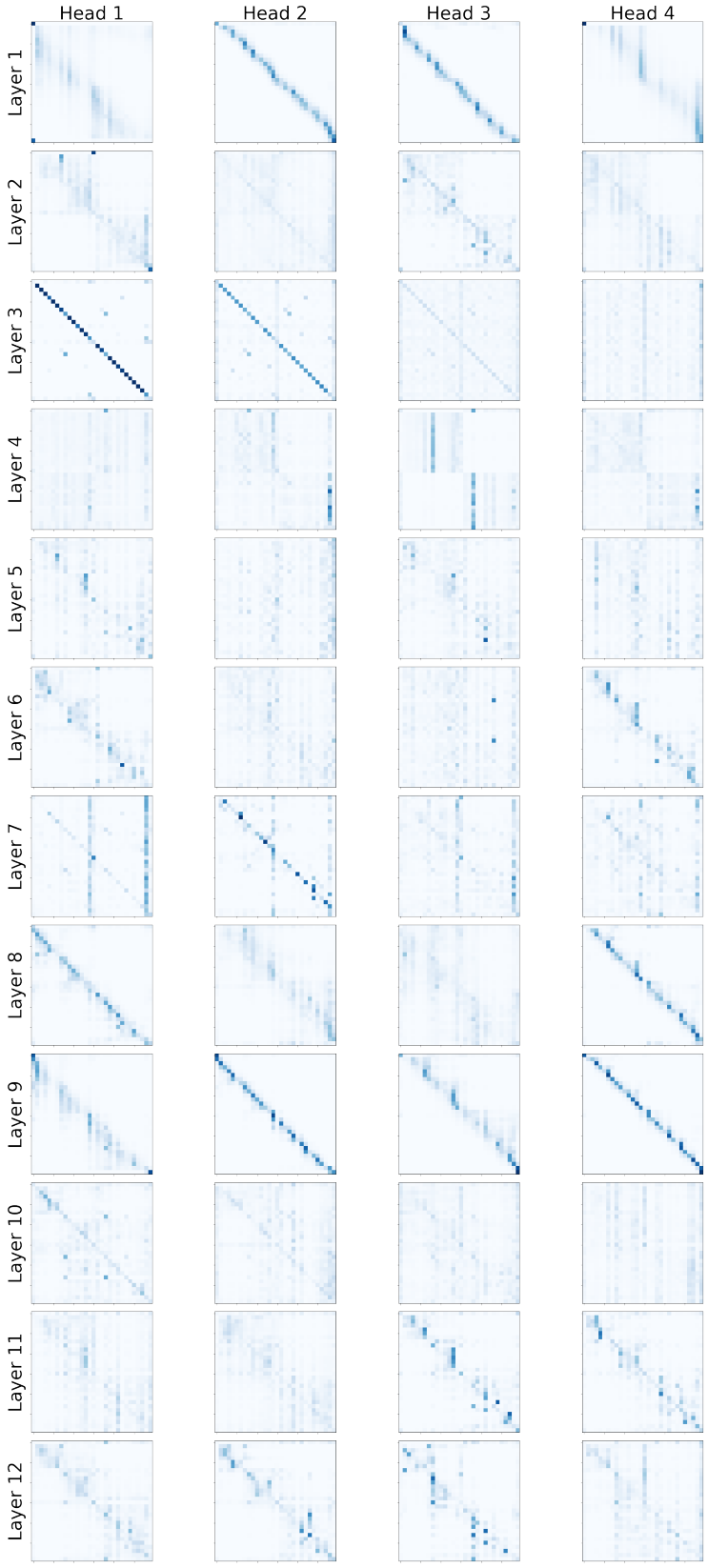


Figure 10: The visualization of attention distributions in the first 12 layers of MobileBERT (bare) + PD.

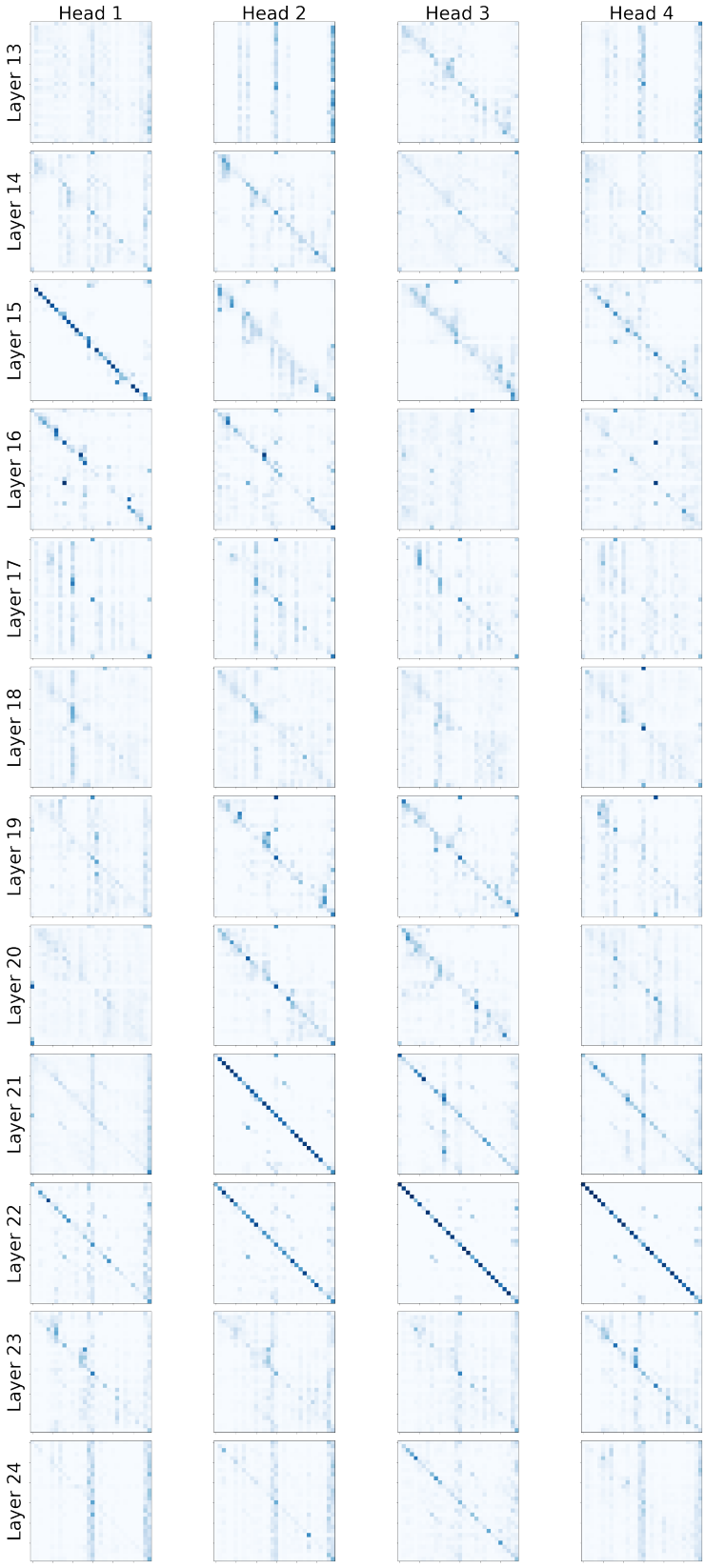


Figure 11: The visualization of attention distributions in the last 12 layers of MobileBERT (bare) + PD.

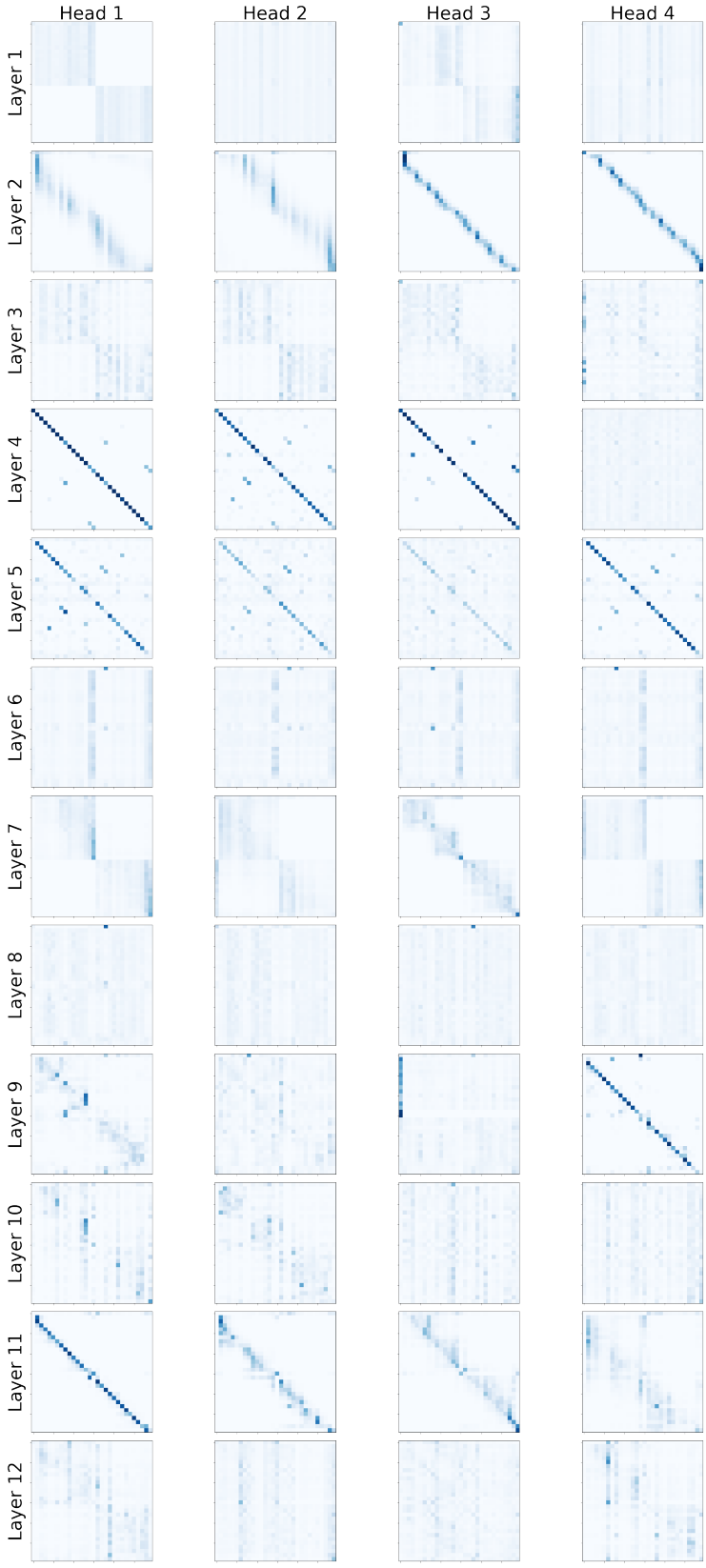


Figure 12: The visualization of attention distributions in the first 12 layers of MobileBERT (bare).

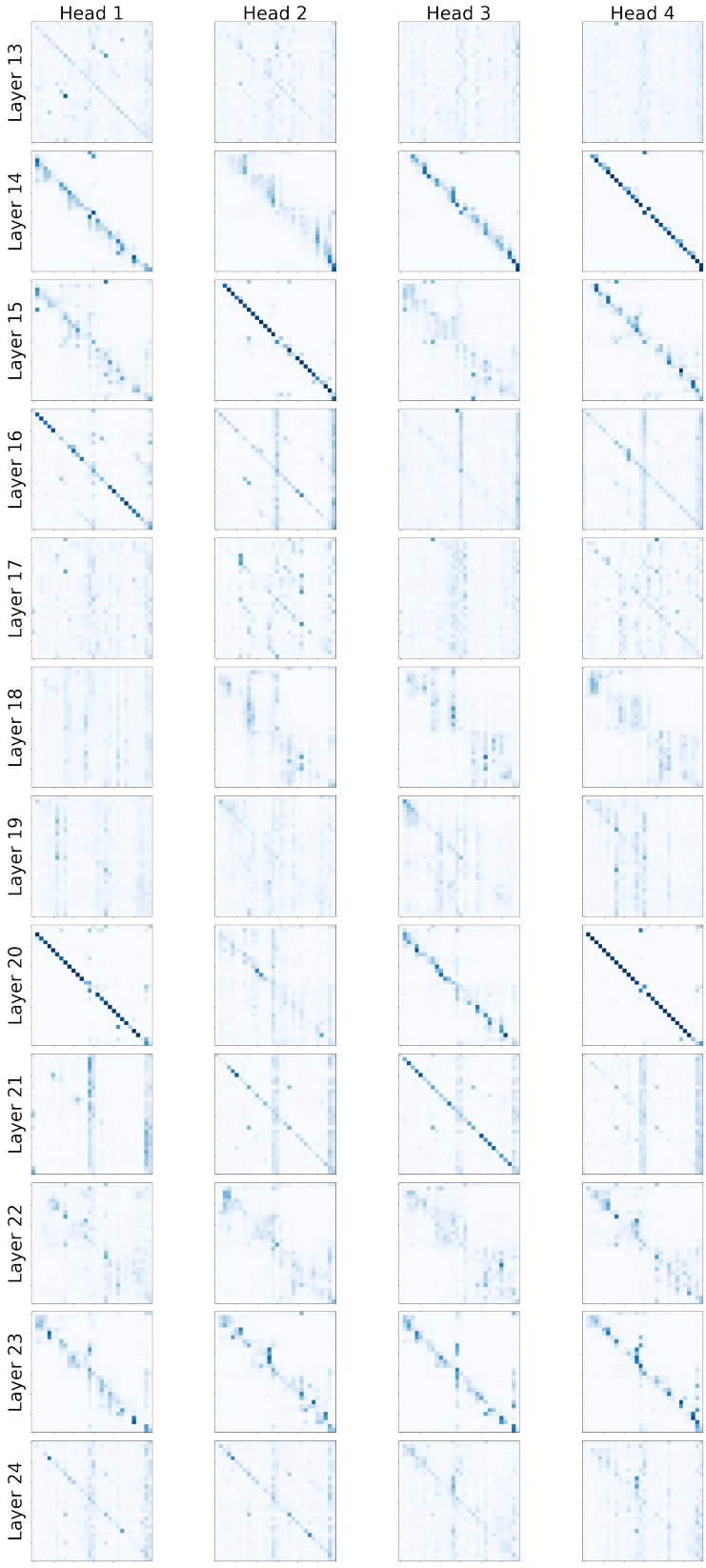


Figure 13: The visualization of attention distributions in the last 12 layers of MobileBERT (bare).