AdaBERT: Task-Adaptive BERT Compression with Differentiable Neural Architecture Search

Daoyuan Chen *, Yaliang Li *, Minghui Qiu, Zhen Wang, Bofang Li, Bolin Ding, Hongbo Deng, Jun Huang, Wei Lin, Jingren Zhou
Alibaba Group
{daoyuanchen.cdy, yaliang.li, minghui.qmh, jones.wz, bofang.lbf}@alibaba-inc.com
{bolin.ding, dhb167148, huangjun.hj, weilin.lw, jingren.zhou}@alibaba-inc.com

Abstract
Large pre-trained language models such as BERT have shown their effectiveness in various natural language processing tasks. However, the huge parameter size makes them difficult to be deployed in real-time applications that require quick inference with limited resources. Existing methods compress BERT into small models while such compression is task-independent, i.e., the same compressed BERT for all different downstream tasks. Motivated by the necessity and benefits of task-oriented BERT compression, we propose a novel compression method, AdaBERT, that leverages differentiable Neural Architecture Search to automatically compress BERT into task-adaptive small models for specific tasks. We incorporate a task-oriented knowledge distillation loss to provide search hints and an efficiency-aware loss as search constraints, which enables a good trade-off between efficiency and effectiveness for task-adaptive BERT compression. We evaluate AdaBERT on several NLP tasks, and the results demonstrate that those task-adaptive compressed models are 12.7x to 29.3x faster than BERT in inference time and 11.5x to 17.0x smaller in terms of parameter size, while comparable performance is maintained.

1 Introduction
Nowadays, pre-trained contextual representation encoders, such as ELMo [Peters et al., 2018], BERT and [Devlin et al., 2019], and RoBERTa [Liu et al., 2019c], have been widely adopted in a variety of Natural Language Processing (NLP) tasks. Despite their effectiveness, these models are built upon large-scale datasets and they usually have parameters in the billion scale. For example, the BERT-base and BERT-large models are with 109M and 340M parameters respectively. It is difficult to deploy such large-scale models in real-time applications that have tight constraints on computation resource and inference time.

To fulfill the deployment in real-time applications, recent studies compress BERT into a relatively small model to reduce the computational workload and accelerate the inference time. BERT-PKD [Sun et al., 2019] distills BERT into a small Transformer-based model that mimics intermediate layers from original BERT. TinyBERT [Jiao et al., 2019] uses two-stage knowledge distillation and mimics attention matrices and embedding matrices of BERT. And in [Michel et al., 2019], the authors propose a method to iteratively prune the redundant attention heads of BERT.

However, these existing studies compress BERT into a task-independent model structure, i.e., the same compressed BERT structure for all different tasks. Recall that BERT learns various knowledge from the large-scale corpus, while only certain parts of the learned knowledge are needed for a specific downstream task [Tenney et al., 2019]. Further, [Liu et al., 2019b] show that different hidden layers of BERT learned different levels of linguistic knowledge, and [Clark et al., 2019] demonstrates that the importance degrees of attention heads in BERT vary for different tasks. All these findings shed light on the task-adaptive BERT compression: different NLP tasks use BERT in different ways, and it is necessary to compress large-scale models such as BERT for specific downstream tasks respectively. By doing so, the task-adaptive compressed BERT can remove task-specific redundant parts in original large-scale BERT, which leads to better compression and faster inference.

Motivated by this, we propose a novel Adaptive BERT compression method, AdaBERT, that leverages differentiable Neural Architecture Search (NAS) to automatically compress BERT into task-adaptive small models for specific tasks. We incorporate a task-oriented knowledge distillation loss that depends on the original BERT model to provide search hints, as well as an efficiency-aware loss based on network structure as search constraints. These two loss terms work together and enable the proposed compression method to achieve a good trade-off between efficiency and effectiveness for different downstream tasks. To be specific, we adopt a lightweight CNN-based search space and explicitly model the efficiency metrics with respect to searched architectures, which has not been considered in previous BERT compression studies. Further, we hierarchically decompose the learned general knowledge of BERT into task-oriented useful knowledge with a set of probe models for knowledge distillation loss, such that the architecture search space can be reduced into a small task-oriented sub-space. Finally, by relaxing the discrete archi-
network architecture is stacked by the searched cell over pre-defined $K_{max}$ layers, where the cell structure parameter $\alpha_c$ is shared for all layers. In this work, we consider a macro search space over the entire network to enhance the structure exploration. Specifically, besides for searching shared cell parameter $\alpha_c$ for stacking, we also search the number of stacking layers $K \in \{1, 2, \ldots, K_{max}\}$. The $K$ is crucial for finding a trade-off between model expressiveness and efficiency, as a larger $K$ leads to higher model capacity but slower inference.

As depicted in Figure 2, the searched cell is represented as a directed acyclic graph. Each node within the cell indicates a latent state $h_i$ and the edge from node $i$ to node $j$ indicates operation $o_{i,j}$ that transforms $h_i$ to $h_j$. For the cell at $k$-th layer ($k > 1$), we define two input nodes $c_{k-2}$ and $c_{k-1}$ as layer-wise residual connections, and an output node $c_k$ that is obtained by attentively summarized over all the intermediate nodes. For the cell at first layer, node 0 and node 1 are task-dependent input embeddings. Formally, let’s denote $O$ as the set of candidate operations. We assume a topological order among $N$ intermediate nodes, i.e., $o_{i,j} \in O$ exists when $i < j$ and $j > 1$, and the search space $\alpha$ can be formalized as:

$$\alpha = \{K, \alpha_c\}, \quad K \leq K_{max},$$

$$\alpha_c = [o_{0,2}, o_{1,2}, \ldots, o_{i,j}, \ldots, o_{N+1,N+2}].$$

2.3 Task-oriented Knowledge Distillation

To encourage the learned structure to be suitable for the target task, we introduce the task-oriented knowledge distillation loss, denoted as $L_{KD}$ in Equation (1), to guide the structure search process.

**Task-useful Knowledge Probe.** We leverage a set of probe classifiers to hierarchically decompose the task-useful knowledge from the teacher model $BERT_t$, and then distill the knowledge into the compressed model. Specifically, we freeze the parameters of $BERT_t$ and train a Softmax probe classifier for each hidden layer w.r.t. the ground-truth task labels. In total we have $J$ classifiers, ($J = 12$ in BERT-base), and the classification logits of $j$-th classifier can be regarded as the learned knowledge from $j$-th layer. Given an input instance $m$, denote $C_j^{T,m}$ as the hidden representation from the $j$-th layer of $BERT_t$, $C_{i,m}^S$ as the attentively summed hidden state on $i$-th layer of the compressed student model, we distill the task-useful knowledge (classification logits) as:

$$L_{KD}^{m} = -P_j^T(C_j^{T,m}) \cdot \log(P_i^S(C_{i,m}^S)/T),$$

where $T$ is the temperature value, $P_j^T$ is the $j$-th teacher probe classifier, $P_i^S$ is the trainable student probe on $i$-th layer of

2 Methodology

2.1 Overview

As shown in Figure 1, we aim to compress a given large BERT model into an effective and efficient task-adaptive model for a specific task. The structures of the compressed models are searched in a differentiable manner with the help of task-oriented knowledge from the large BERT model while taking the model efficiency into consideration.

Formally, let’s denote a BERT model fine-tuned on the target data $D_t$ as $BERT_t$, an architecture searching space as $A$. Our task is to find an optimal architecture $\alpha \in A$ by minimizing the following loss function:

$$L = (1 - \gamma)L_{CE}(\alpha, w_\alpha, D_t) +$$

$$\gamma L_{KD}(\alpha, w_\alpha, BERT_t) + \beta L_E(\alpha),$$

(1)

where $w_\alpha$ is the trainable network weights of the architecture $\alpha$ (e.g., weights of a feed forward layer), $L_{CE}$, $L_{KD}$, and $L_E$ are losses for the target task, task-oriented knowledge distillation and efficiency respectively. Specifically, $L_{CE}$ is the cross-entropy loss w.r.t. labels from the target data $D_t$, $L_{KD}$ is the task-oriented knowledge distillation (KD) loss that provides hints to find suitable structures for the task, and $L_E$ is the efficiency-aware term to provide constraints to help search lightweight and efficient structures. $\gamma$ and $\beta$ are hyper-parameters to balance these loss terms.

2.2 Search Space $A$

Most neural architecture search methods focus on cell-based micro search space [Zoph and Le, 2017; Pham et al., 2018; Liu et al., 2019a]. That is, the searching target is a cell and the architecture parameters into continuous distribution, the proposed method can efficiently find task-adaptive compression structures through the gradient-based optimization.

To evaluate the proposed method, we compress BERT for several NLP tasks including sentiment classification, entailment recognition, and semantic equivalence classification. Empirical results on six datasets show that the proposed compression method can find task-adaptive compression models that are 12.7x to 29.3x faster than BERT in inference time and 11.5x to 17.0x smaller than BERT in terms of parameter size while maintaining comparable performance.

Figure 1: The overview of AdaBERT.

Figure 2: Search space including stacked layers and stacked cells.
the compressed model. Clearly, $L_{KD}^{i,m}$ represents the mastery degree of teacher knowledge from the $i$-th layer of the compressed model, i.e., how similar are the logits of the two models predict for the same input $m$. Here we set $j$ to be proportional to the layers of the two models, i.e., $j = \lfloor i \times J/K \rfloor$, such that the compressed model can learn the decomposed knowledge smoothly and hierarchically.

**Attentive Hierarchical Transfer.** The usefulness of each layer of BERT is varied for different tasks as shown in [Liu et al., 2019]. Here we attentively combine the decomposed knowledge for all layers as:

$$L_{KD} = \sum_{m=0}^{M} \sum_{i=1}^{K} w_{i,m} \cdot L_{KD}^{i,m},$$

$$w_{i,m} = \frac{\exp[y_m \cdot \log P_T^{j}(C^S)]}{\sum_j \exp[y_m \cdot \log P_T^{j}(C^S)]},$$

where $M$ is the total number of training instances, $y_m$ is the label of instance $m$, $w_{i,m}$ is set as the normalized weight according to the negative cross-entropy loss of the teacher probe $P_T^{j}$, so that probe classifiers making preciser predictions (smaller loss) gain higher weights. Besides, to enrich task-useful knowledge, we perform data augmentation on target task datasets with the augmentation process used in [Jiao et al., 2019], which uses BERT to replace original words.

### 2.4 Efficiency-Aware Loss

Recall that we aim to compress the original BERT model into efficient compressed models. To achieve this, we further incorporate model efficiency into loss function from two aspects, i.e., parameter size and inference time. To be specific, for searched architecture $\alpha_i$ and $K$, we define the efficiency-aware loss in Equation (1) as:

$$L_E = \frac{K}{K_{\text{max}}} \sum_{i,j} \text{SIZE}(\alpha_{i,j}) + \text{FLOPs}(\alpha_{i,j}),$$

where $K_{\text{max}}$ is the pre-defined maximum number of layers, $\text{SIZE}(\cdot)$ and $\text{FLOPs}(\cdot)$ are the normalized parameter size and the number of floating point operations (FLOPs) for each operation. The sum of FLOPs of searched operations serves as an approximation to the actual inference time of the compressed model.

### 2.5 Differentiable Architecture Searching

A major difference between the proposed method and existing BERT compression methods is that the proposed Ad-aBERT method seeks to find task-adaptive structures for different tasks. Now we will discuss the task-adaptive structure searching via a differentiable structure search method with the aforementioned loss terms.

**Search Space Setting**

Before diving into the details of the search method, we first illustrate the search space for our method in Figure 2. To make it easy to stack cells and search for network layers, we keep the same shapes for input and output nodes of each layer. In the first layer, we adopt different input settings for single text tasks such as sentiment classification and text pair tasks such as textual entailment. As shown in Figure 2, the inputs $c_{k-2}$ and $c_{k-1}$ are set as the same text input for single text tasks, and set as two input texts for text pair tasks. This setting helps to explore self-attention or self-interactions for single text task, and pair-wise cross-interactions for text pair tasks.

For the candidate operations in the cell, we adopt lightweight CNN-based operations as they are effective in NLP tasks [Bai et al., 2018], and CNN-based operations have shown inference speed superiority over RNN-based models and self-attention based models [Shen et al., 2018] due to the fact that they are parallel-friendly operations. Specifically, the candidates operation set $O$ include convolution, pooling, skip (identity) connection and zero (discard) operation. For convolution operations, both 1D standard convolution and dilated convolution with kernel size $\{3, 5, 7\}$ are included, among which the dilated convolution can be used to enhance the capability of capturing long-dependency information and each convolution is applied as a Relu-Conv-BatchNorm structure. Pooling operations include averaging pooling and max pooling with kernel size 3. The skip and zero operations are used to build residual connection and discard operation respectively, which are helpful to reduce network redundancy. Besides, for both convolution and pooling operations, we apply the “SAME” padding to make the output length the same as the input.

**Search Algorithm**

Directly optimizing the overall loss $L$ in Equation (1) by brute-force enumeration of all the candidate operations is impossible, due to the huge search space with combinatorial operations and the time-consuming training on its $w_o$ w.r.t. $\alpha$. Here we solve this problem by modeling searched architecture $\{K, \alpha_{i,j}\}$ as discrete variables that obey discrete probability distributions $P_K = [\theta^K_1, \ldots, \theta^K_{K_{\text{max}}}]$ and $P_o = [\theta^o_1, \ldots, \theta^o_{O}]$. $K$ and $\alpha_{i,j}$ are thus modeled as one-hot variables and sampled from layer range $[1, K_{\text{max}}]$ and candidate operation set $O$ respectively. However, $L$ is non-differentiable as the discrete sampling process makes the gradients cannot propagate back to the learnable parameters $P_K$ and $P_o$. Inspired by [Xie et al., 2019; Wu et al., 2019], we leverage Gumbel Softmax technique [Jang et al., 2017; Maddison et al., 2017] to relax the categorical samples into continuous sample vectors $y^K \in \mathbb{R}^{K_{\text{max}}}$ and $y^o \in \mathbb{R}^O$ as:

$$y^K_i = \frac{\exp[(\log(\theta^K) + g_i)/\tau]}{\sum_{j=1}^{K_{\text{max}}} \exp[(\log(\theta^K) + g_j)/\tau]},$$

$$y^o_i = \frac{\exp[(\log(\theta^o) + g_i)/\tau]}{\sum_{j=1}^{O} \exp[(\log(\theta^o) + g_j)/\tau]},$$

where $g_i$ is a random noise drawn from Gumbel(0, 1) distribution, $\tau$ is the temperature to control the degree of approximating Gumbel-Softmax to argmax, i.e., as $\tau$ approaches 0, the samples become one-hot. By this way, $y^K$ and $y^o$ are differentiable proxy variables to discrete samples, and then we can efficiently optimize $L$ directly using gradient information. Specifically, we use one-hot sample vectors $\text{argmax}(y^K)$ and $\text{argmax}(y^o)$ in the forward stage while use continuous
Table 1: The compression results including model efficiency and accuracy from the GLUE test server, and the MNLI result is evaluated for matched-accuracy (MNLI-m). BERT$_{12}$ indicates the results of the fine-tuned BERT-base from [Devlin et al., 2019] and BERT$_{12}$-T indicates the results of the fine-tuned BERT-base in our implementation. The results of BERT-PKD are from [Sun et al., 2019], the results of DistilBERT$_{4}$ and TinyBERT$_{4}$ are from [Jiao et al., 2019], and the results of BiLSTM$_{SOFT}$ is from [Tang et al., 2019]. The number of model parameters includes the embedding size, and the inference time is tested with a batch size of 128 over 50,000 samples. The bold numbers and underlined numbers indicate the best and the second-best performance respectively.

Table 2: The number of layers, parameters and inference speedups of searched structures by AdaBERT for different tasks.

3 Experiments

3.1 Setup

Datasets. We evaluate the proposed AdaBERT method on six datasets from GLUE [Wang et al., 2019a] benchmark. Specifically, we consider three types of NLP tasks, namely sentiment classification, semantic equivalence classification, and entailment recognition. SST-2 is adopted for sentiment classification, whose goal is to label movie reviews as positive or negative. MRPC and QQP are adopted for semantic equivalence classification, whose sentence pairs are extracted from news sources and online website respectively. MNLI, QNLI and RTE are adopted for textual entailment recognition, whose premise-hypothesis pairs vary in domains and scales.

Baselines. We compare the proposed AdaBERT method with several state-of-the-art BERT compression methods including BERT-PKD [Sun et al., 2019], DistilBERT [Sanh et al., 2019], TinyBERT [Jiao et al., 2019] and BiLSTM$_{SOFT}$ [Tang et al., 2019]. Since our approach searches architecture from a space including the number of network layers $K$, here we also compare with several different versions of these baselines with the different number of layers for a comprehensive comparison.

AdaBERT Setup. We fine-tune the BERT-base model [Devlin et al., 2019] on the six adopted datasets respectively as teacher models for knowledge distillation. For input text, following [Lan et al., 2020], we factorize the WordPiece embedding of BERT$_{t}$ into smaller embeddings whose dimension is 128 and set the max input length as 128. For the optimization of operation parameters, we adopt SGD with momentum as 0.9 and learning rate from 2e-2 to 5e-4 scheduled by cosine annealing. For the optimization of architecture distribution parameters $P_o$, we use Adam with a learning rate of 3e-4 and weight decay of 1e-3. For AdaBERT, we set $\gamma = 0.8$, $\beta = 4$, $T = 1$, inner node $N = 3$ and search layer $K_{max} = 8$. We search $P_o$ for 80 epochs and derive the searched structure with its trained operation weights.

3.2 Overall Results

Compression Results

The compression results on the six adopted datasets, including parameter size, inference speedup and classification accuracy, are summarized in Table 1. Detailed results of AdaBERT method for different tasks are reported in Table 2.

Overall speaking, on all the evaluated datasets, the proposed AdaBERT method achieves significant efficiency improvement while maintaining comparable performance. Compared to the BERT$_{12}$-T, the compressed models are 11.5x to 17.0x smaller in parameter size and 12.7x to 29.3x smaller in parameter size.
faster in inference speed with an average performance degradation of 2.79%. This demonstrates the effectiveness of AdaBERT to compress BERT into task-dependent small models.

Comparing with different Transformers-based compression baselines, the proposed AdaBERT method is 1.35x to 3.12x faster than the fastest baseline, TinyBERT, and achieves comparable performance with the two baselines that have the best averaged accuracy, BERT-PKD and TinyBERT. Further, as shown in Table 2, AdaBERT searches suitable layers and structures for different tasks, e.g., the searched structure for SST-2 task is lightweight since this task is relatively easy and a low model capacity is enough to mimic task-useful knowledge from the original BERT. This observation confirms that AdaBERT can automatically search small compressed models that adapt to downstream tasks.

Comparing with another structure-heterogeneous method, BiLSTM, AdaBERT searches CNN-based models and achieves much better improvements, especially on the MNLI dataset. This is because AdaBERT searches different models for different downstream tasks (as Table 2 shows), and adopts a flexible searching manner to find suitable structures for different tasks while BiLSTM uses a Siamese structure for all different tasks. This shows the flexibility of AdaBERT to derive task-oriented compressed models for different tasks, and we will investigate more about this in the following part.

Adaptiveness Study

Cross-Task Validation. In order to further examine the adaptiveness of searched structures by AdaBERT, we apply the searched compression model structures across different downstream tasks. For example, the searched structure for task SST-2 (we denote this searched structure as AdaBERT-SST-2) is applied to all different tasks. For such cross-task validation, we randomly initialize the weights of each searched structure and re-train its weights using corresponding training data to ensure a fair comparison. The results of cross-task validation is summarized in Table 3.

From Table 3, we can observe that: The searched structures achieve the best performance on their original target tasks compared with other tasks, in other words, the performance numbers along the diagonal line of this table are the best. Further, the performance degradation is quite significant across different kinds of tasks (for example, applying the searched structures of sentiment classification tasks to entailment recognition task, or vice verse), while the performance degradations within the same kind of tasks (for example, MRPC and QQP for semantic equivalence classification) are relatively small, since they have the same input format (i.e., a pair of sentences) and similar targets. All these observations verify that AdaBERT can search task-adaptive structures for different tasks with the guidance of task-specific knowledge.

We also conduct another set of experiment: for each task, we randomly sample a model structure without searching, and then train such structures on the corresponding datasets. From the last row of Table 3, we can see that the randomly sampled structures perform worse than the searched structures and their performances are not stable. This shows the necessity of the proposed adaptive structure searching.

Architecture Study. In order to examine the adaptiveness of searched structures, we also visualize the basic cells of searched structures for different tasks in Figure 3.

By comparing the structure searched for single-text task such as sentiment classification (Figure 3a) with the one for sentiment equivalence classification task (Figure 3b), we can find that the former searched structure has more aggregation operations (maxpool1d_3) and smaller feature filters (std_conv_3 and dil_conv_3) since encoding local features are good enough for the binary sentiment classification task, while the latter searched structure has more interactions between the two input nodes as it deals with text pairs.

For text-pair tasks, compared with sentiment equivalence classification task (Figure 3b), the searched structure for the

<table>
<thead>
<tr>
<th>Structure</th>
<th>Task</th>
<th>SST-2</th>
<th>MRPC</th>
<th>QQP</th>
<th>MNLI</th>
<th>QNLI</th>
<th>RTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdaBERT-SST-2</td>
<td>91.9</td>
<td>78.1</td>
<td>58.6</td>
<td>64.0</td>
<td>74.1</td>
<td>53.8</td>
<td></td>
</tr>
<tr>
<td>AdaBERT-MRPC</td>
<td>81.5</td>
<td>84.7</td>
<td>68.9</td>
<td>75.9</td>
<td>82.2</td>
<td>60.3</td>
<td></td>
</tr>
<tr>
<td>AdaBERT-QQP</td>
<td>81.9</td>
<td>84.1</td>
<td>70.5</td>
<td>76.3</td>
<td>82.5</td>
<td>60.5</td>
<td></td>
</tr>
<tr>
<td>AdaBERT-MNLI</td>
<td>82.1</td>
<td>81.5</td>
<td>66.8</td>
<td>81.3</td>
<td>86.1</td>
<td>63.2</td>
<td></td>
</tr>
<tr>
<td>AdaBERT-QNLI</td>
<td>81.6</td>
<td>82.3</td>
<td>67.7</td>
<td>79.2</td>
<td>87.2</td>
<td>62.9</td>
<td></td>
</tr>
<tr>
<td>AdaBERT-RTE</td>
<td>82.9</td>
<td>81.1</td>
<td>66.5</td>
<td>79.8</td>
<td>86.0</td>
<td>64.1</td>
<td></td>
</tr>
<tr>
<td>Random</td>
<td>80.4 ± 4.3</td>
<td>79.2 ± 2.8</td>
<td>61.8 ± 4.9</td>
<td>69.7 ± 6.7</td>
<td>78.2 ± 5.5</td>
<td>55.3 ± 4.1</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Accuracy comparison on the dev sets with the searched compression structures applying to different tasks. For Random, 5-times averaging results with standard deviations are reported.
entailment recognition task (Figure 3c) has more diverse operations such as avg_pool_3 and skip_connect, and more early interactions among all the three subsequent nodes (node 0, 1 and 2). This may be justified by the fact that textual entailment requires different degrees of reasoning and thus the searched structure has more complex and diverse interactions.

The above comparisons among the searched structures for different tasks confirm that AdaBERT can search task-adaptive structure for BERT compression. Next, we conduct ablation studies to examine how knowledge losses and efficiency-aware loss affect the performance of AdaBERT.

3.3 Ablation Study

Knowledge Losses

We evaluate the effect of knowledge distillation ($\mathcal{L}_{KD}$) and the supervised label knowledge ($\mathcal{L}_{CE}$) by conducting experiments on different tasks. The results are shown in Table 4.

The Base-KD (line 1 in Table 4) is a naive knowledge distillation version in which only the logits of the last layer are distilled without considering hidden layer knowledge and supervised label knowledge. By incorporating the probe models, the performance (line 2 in Table 4) is consistently improved, indicating the benefits from hierarchically decomposed task-oriented knowledge. We then leverage Data Augmentation (DA) to enrich task-oriented knowledge and this technique also improves performance for all tasks, especially for tasks that have a limited scale of data (i.e., MRPC and RTE). DA is also adopted in existing KD-based compression studies [Tang et al., 2019; Jiao et al., 2019].

When taking the supervised label knowledge ($\mathcal{L}_{CE}$) into consideration, the performance is further boosted, showing that this term is also important for AdaBERT by providing focused search hints.

Efficiency-aware Loss

Last, we test the effect of efficiency-aware loss $L_{E}$ by varying its corresponding coefficient, including the standard case ($\beta = 4$), without efficiency constraint ($\beta = 0$), and strong efficiency constraint ($\beta = 8$). The model performance and corresponding model size are reported in Table 5.

On the one hand, removing the efficiency-aware loss ($\beta = 0$) leads to the increase in model parameter size, on the other hand, a more aggressive efficiency preference ($\beta = 8$) results in the small model size but degraded performance, since a large $\beta$ encourages the compressed model to adopt more lightweight operations such as zero and skip which hurt the performance. A moderate efficiency constraint ($\beta = 4$) provides a regularization, guiding the AdaBERT method to achieve a trade-off between the small parameter size and the good performance.

4 Related Work

Existing efforts to compress BERT model can be broadly categorized into four lines: knowledge distillation, parameter sharing, pruning and quantization.

For knowledge distillation based methods, in [Tang et al., 2019], BERT is distilled into a simple BiLSTM and achieves comparable results with ELMo. A dual distillation is proposed to reduce the vocabulary size and the embedding size in [Zhao et al., 2019]. PKD-BERT [Sun et al., 2019] and DistilBERT [Sanh et al., 2019] distill BERT into shallow Transformers in the fine-tune stage and the pre-train stage respectively. TinyBERT [Jiao et al., 2019] further distills BERT with a two-stage knowledge distillation. For parameter sharing based methods, the multi-head attention is compressed into a tensorized Transformer in [Ma et al., 2019]. AIBERT [Lan et al., 2020] leverages cross-layer parameter sharing to speed up the training. Different from these existing methods, the proposed AdaBERT incorporates both task-oriented knowledge distillation and efficiency factor, and automatically compresses BERT into task-adaptive small structures instead of a task-independent structure in existing methods.

For pruning and quantization based methods, [Michel et al., 2019] and [Wang et al., 2019b] prune attention heads and weight matrices respectively. Q8BERT [Zafir et al., 2019] quantizes matrix multiplication operations into 8-bit operations, while Q-BERT [Shen et al., 2019] quantizes BERT with Hessian based mix-precision. These methods and the proposed AdaBERT compress BERT from different aspects that are complementary, that is, one can first distill BERT into a small model, and then prune or quantize the small model.

5 Conclusion

In this work, motivated by the strong need to compress BERT into small and fast models, we propose AdaBERT, an effective and efficient model that adaptively compresses BERT for various downstream tasks. By leveraging Neural Architecture Search, we incorporate two kinds of losses, task-useful knowledge distillation loss and model efficiency-aware loss, such that the task-suited structures of compressed BERT can be automatically and efficiently found using gradient information. We evaluate the proposed AdaBERT on six datasets involving three kinds of NLP tasks. Extensive experiments demonstrate that AdaBERT can achieve comparable performance while significant efficiency improvements, and find task-suited models for different downstream tasks.
References


