COST-EFF: Collaborative Optimization of Spatial and Temporal Efficiency with Slenderized Multi-exit Language Models

Bowen Shen[1,2], **Zheng Lin**[1,2]*, **Yuanxin Liu**[1,3], **Zhengxiao Liu**[1], **Lei Wang**[1]*, **Weiping Wang**[1]

[1]Institute of Information Engineering, Chinese Academy of Sciences
[2]School of Cyber Security, University of Chinese Academy of Sciences
[3]MOE Key Laboratory of Computational Linguistics, Peking University
{shenbowen, linzheng, liuzhengxiao, wanglei, wangweiping}@iie.ac.cn
liuyuanxin@stu.pku.edu.cn

Abstract

Transformer-based pre-trained language models (PLMs) mostly suffer from excessive overhead despite their advanced capacity. For resource-constrained devices, there is an urgent need for a spatially and temporally efficient model which retains the major capacity of PLMs. However, existing statically compressed models are unaware of the diverse complexities between input instances, potentially resulting in redundancy and inadequacy for simple and complex inputs. Also, miniature models with early exiting encounter challenges in the trade-off between making predictions and serving the deeper layers. Motivated by such considerations, we propose a collaborative optimization for PLMs that integrates static model compression and dynamic inference acceleration. Specifically, the PLM is slenderized in width while the depth remains intact, complementing layer-wise early exiting to speed up inference dynamically. To address the trade-off of early exiting, we propose a joint training approach that calibrates slenderization and preserves contributive structures to each exit instead of only the final layer. Experiments are conducted on GLUE benchmark and the results verify the Pareto optimality of our approach at high compression and acceleration rate with 1/8 parameters and 1/19 FLOPs of BERT.

1 Introduction

Pre-training generalized language models and fine-tuning them on specific downstream tasks has become the dominant paradigm in natural language processing (NLP) since the advent of Transformers (Vaswani et al., 2017) and BERT (Devlin et al., 2019). However, pre-trained language models (PLMs) are predominantly designed to be vast in the pursuit of model capacity and generalization. With such a concern, the model storage and inference time of PLMs are usually high, making them challenging to be deployed on resource-constrained devices (Sun et al., 2020).

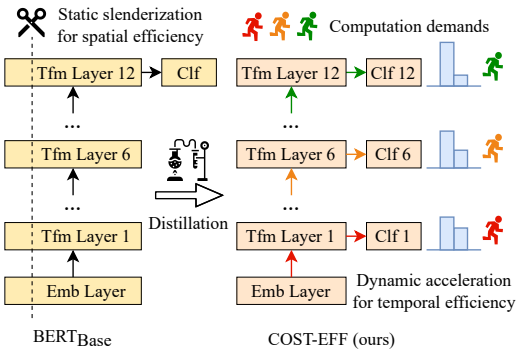

Figure 1: An illustration of COST-EFF model structure and inference procedure. Emb, Tfm and Clf are abbreviations of embedding, Transformer and classifier, respectively. Blue bar charts denote probability distribution output by classifiers.

Recent studies indicate that Transformer-based PLMs bear redundancy spatially and temporally which comes from the excessive width and depth of the model (Michel et al., 2019; Xin et al., 2021). With static compression methods including network pruning (Xia et al., 2022) and knowledge distillation (Jiao et al., 2020), spatial overheads of PLMs (i.e., model parameters) can be reduced to a fixed setting. From the perspective of input instances rather than the model, early exiting without passing all the model layers enables the dynamic acceleration at inference time and diminishes the temporal overheads (Zhou et al., 2020).

However, static compression can hardly find an optimal setting that is both efficient on simple input instances and accurate on complex ones, while early exiting cannot diminish the redundancy in model width and is impotent for reducing the actual volume of model. Further, interpretability studies indicate that the attention and semantic features across layers are different in BERT (Clark

* Zheng Lin and Lei Wang are the corresponding authors.

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 1719 - 1730
December 7-11, 2022

et al., 2019). Hence, deriving a multi-exit model from a pre-trained single-exit model like BERT incurs inconsistency in the training objective, where each layer is simultaneously making predictions and serving the deeper layers (Xin et al., 2021). Empirically, we find that the uncompressed BERT is not severely influenced by such inconsistency, whereas small capacity models are not capable of balancing shallow and deep layers. Plugging in exits after compression will lead to severe performance degradation, which hinders the complementation of the two optimizations.

To fully exploit the efficiency of PLMs and mitigate the above-mentioned issues, we design a slenderized multi-exit model and propose a Collaborative Optimization approach of Spatial and Temporal EFFiciency (COST-EFF) as depicted in Figure 1. Unlike previous works, e.g., DynaBERT (Hou et al., 2020) and CoFi (Xia et al., 2022), which obtain a squat model, we keep the depth intact while slenderizing the PLM. Superiority of slender architectures over squat ones is supported by (Bengio et al., 2007) and (Turc et al., 2019) in generic machine learning and PLM design. To address the inconsistency in compressed multi-exit model, we first distill an multi-exit BERT from the original PLM as both the teaching assistant (TA) and the slenderization backbone, which is more effective in balancing the trade-off between layers than compressed models. Then, we propose a collaborative approach that slenderizes the backbone with the calibration of exits. Such a slenderization diminishes less contributive structures to each exit as well as the redundancy in width. After the slenderization, task-specific knowledge distillation is conducted with the objectives of hidden representations and predictions of each layer as recovery. Specifically, the contributions of this paper are as follows.

- To comprehensively optimize the spatial and temporal efficiency of PLMs, we leverage both static slenderization and dynamic acceleration from the perspective of model scale and variable computation.
- We propose a collaborative training approach that calibrates the slenderization under the guidance of intermediate exits and mitigates the inconsistency of early exiting.
- Experiments conducted on the GLUE benchmark verify the Pareto optimality of our approach. COST-EFF achieves 96.5% performance of fine-tuned $\text{BERT}_{\text{Base}}$ with approximately $1/8$ parameters and $1/19$ FLOPs without any form of data augmentation.[1]

2 Related Work

The compression and acceleration of PLMs were recently investigated to neutralize the overhead of large models by various means.

The structured pruning objects include, from small to large, hidden dimensions (Wang et al., 2020), attention heads (Michel et al., 2019), multi-head attention (MHA) and feed-forward network (FFN) modules (Xia et al., 2022) and entire Transformer layers (Fan et al., 2020). Considering the benefit of the overall structure, we keep all the modules while reducing their sizes. Besides pruning out structures, a fine-grained approach is unstructured pruning which prunes out weights. Unstructured pruning can achieve high sparsity of 97% (Xu et al., 2022) but is not yet adaptable to general computing platforms and hardware.

During the recovery training of compressed models, knowledge distillation objectives include predictions of classifiers (Sanh et al., 2020), features of intermediate representations (Jiao et al., 2020) and relations between samples (Tung and Mori, 2019). Also, the occasion of distillation varies from general pre-training and task-specific fine-tuning (Turc et al., 2019). Distillation enables the training without ground-truth labels complementing data augmentation. In this paper, data augmentation is not leveraged as it requires a long training time, but our approach is well adaptable to it if better performance is to be pursued.

Dynamic early exits come from BranchyNet (Teerapittayanon et al., 2016), which introduces exit branches after specific convolution layers of the CNN model. The idea is adopted to PLMs as Transformer layer-wise early exiting (Xin et al., 2021; Zhou et al., 2020; Liu et al., 2020). However, early exiting only accelerates inference but does not reduce the model size and the redundancy in width. Furthermore, owing to the inconsistency between shallow and deep layers, it is hard to achieve high speedup using early exiting alone.

The prevailing PLMs, e.g., RoBERTa (Liu et al., 2019) and XLNet (Yang et al., 2019) are variants of Transformer with similar overall structures,

[1]Code is available at https://github.com/sbwww/COST-EFF.

well-adaptable to the optimizations that we propose. Apart from PLMs with increasing size, ALBERT(Lan et al., 2020) is distinctive with a small volume of 18M (Million) parameters obtained from weight sharing of Transformer layers. Weight sharing allows the model to store the parameters only once, greatly reducing the storage overhead. However, the shared weights have no contribution to inference speedup. Instead, the time required for ALBERT to achieve BERT-like accuracy increases.

3 Methodology

In this section, we analyze the major structures of Transformer-based PLMs and devise corresponding optimizations. The proposed COST-EFF has three key properties, namely static slenderization, dynamic acceleration and collaborative training.

3.1 Preliminaries

In this paper, we focus on optimizing the Transformer-based PLM which mainly consists of embedding, MHA and FFN. Specifically, embedding converts each input token to a tensor of size H (i.e., hidden dimension). With a common vocabulary size of $|\mathbb{V}| = 30,522$, the word embedding matrix accounts for $< 22\%$ of $\text{BERT}_{\text{Base}}$ parameters. Inside the Transformer, MHA has four matrices $\boldsymbol{W}_Q$, $\boldsymbol{W}_K$, $\boldsymbol{W}_V$ and $\boldsymbol{W}_O$, all of which with input and output size of H. FFN has two matrices $\boldsymbol{W}_{FI}$ and $\boldsymbol{W}_{FO}$ with the size of $H \times F$. As the key components of Transformer, MHA and FFN account for $< 26\%$ and $< 52\%$ of $\text{BERT}_{\text{Base}}$ parameters, respectively.

Based on the analysis, we have the following slenderization and acceleration schemes. (1) The word embedding matrix $\boldsymbol{W}_t$ is decomposed into the multiplication of two matrices following (Lan et al., 2020). Thus, the vocabulary size $|\mathbb{V}|$ and hidden size H are not changed. (2) For the transformation matrices of MHA and FFN, structured pruning is adopted to reduce their input or output dimensions. (3) The inference is accelerated through early exiting as we retain the pre-trained model depth. To avoid introducing additional parameters, we remove the pre-trained pooler matrix before classifiers. (4) Knowledge distillation on prediction logits and hidden states of each layer is leveraged as a substitute for conventional fine-tuning. The overall architecture of COST-EFF is depicted in Figure 2.

3.2 Static Slenderization

3.2.1 Matrix Decomposition of Embedding

As mentioned before, the word embedding takes up more than 1/5 of $\text{BERT}_{\text{Base}}$ parameters. The output dimension of word embedding is equal to hidden size, which we don't modify, we use truncated singular value decomposition (TSVD) to internally compress the word embedding matrix.

TSVD first decomposes the matrix as $\boldsymbol{A}^{m\times n} = \boldsymbol{U}^{m\times m}\boldsymbol{\Sigma}^{m\times n}\boldsymbol{V}^{n\times n}$, where $\boldsymbol{\Sigma}^{m\times n}$ is the singular value diagonal matrix. After that, the three matrices are truncated to the given rank. Thus, the decomposition of word embedding is as

$$\begin{aligned}\boldsymbol{W}_t^{|\mathbb{V}|\times H} &\approx \boldsymbol{W}_{t1}^{|\mathbb{V}|\times R}\boldsymbol{W}_{t2}^{R\times H} \\ &= \left(\tilde{\boldsymbol{U}}^{|\mathbb{V}|\times R}\tilde{\boldsymbol{\Sigma}}^{R\times R}\right)\tilde{\boldsymbol{V}}^{R\times H},\end{aligned} \tag{1}$$

where we multiplies $\tilde{\boldsymbol{U}}$ and $\tilde{\boldsymbol{\Sigma}}$ matrices as the first embedding matrix $\boldsymbol{W}_{t1}^{|\mathbb{V}|\times R}$ and $\boldsymbol{W}_{t2}^{R\times H} = \tilde{\boldsymbol{V}}$ is a linear transformation with no bias.

3.2.2 Structured Pruning of MHA and FFN

To compress the matrices in MHA and FFN which contribute to most of the PLM's parameters, we adopt structured pruning to compress one dimension of the matrices. As depicted in Figure 2, the pruning granularity of MHA and FFN are attention head and hidden dimension, respectively.

Following (Molchanov et al., 2017), COST-EFF has the pruning objective of minimizing the difference between pruned and original model, which is calculated by first-order Taylor expansion

$$\begin{aligned}|\Delta(\boldsymbol{S})| &= |\mathcal{L}(\boldsymbol{X}) - \mathcal{L}(\boldsymbol{X}|\boldsymbol{h}_i = 0, \boldsymbol{h}_i \in \boldsymbol{S})| \\ &= \left|\sum_{\boldsymbol{h}_i\in\boldsymbol{S}} \frac{\delta\mathcal{L}}{\delta\boldsymbol{h}_i}(\boldsymbol{h}_i - 0) + R^{(1)}\right| \\ &\approx \left|\sum_{\boldsymbol{h}_i\in\boldsymbol{S}} \frac{\delta\mathcal{L}}{\delta\boldsymbol{h}_i}\boldsymbol{h}_i\right|,\end{aligned} \tag{2}$$

where $\boldsymbol{S}$ denotes a specific structure, i.e., a set of weights, $\mathcal{L}(\cdot)$ is the loss function and $\frac{\delta\mathcal{L}}{\delta\boldsymbol{h}_i}$ is the gradient of loss to weight $\boldsymbol{h}_i$. $|\Delta(\boldsymbol{S})|$ is the importance of structure $\boldsymbol{S}$ measured by absolute value of the first-order term. For simplicity, we ignore the remainder $R^{(1)}$ in Taylor expansion.

In each Transformer layer, the structure $\boldsymbol{S}$ of MHA is the attention head while that of FFN is the hidden dimension as depicted in the lower part of Figure 2. Specifically, the output dimensions

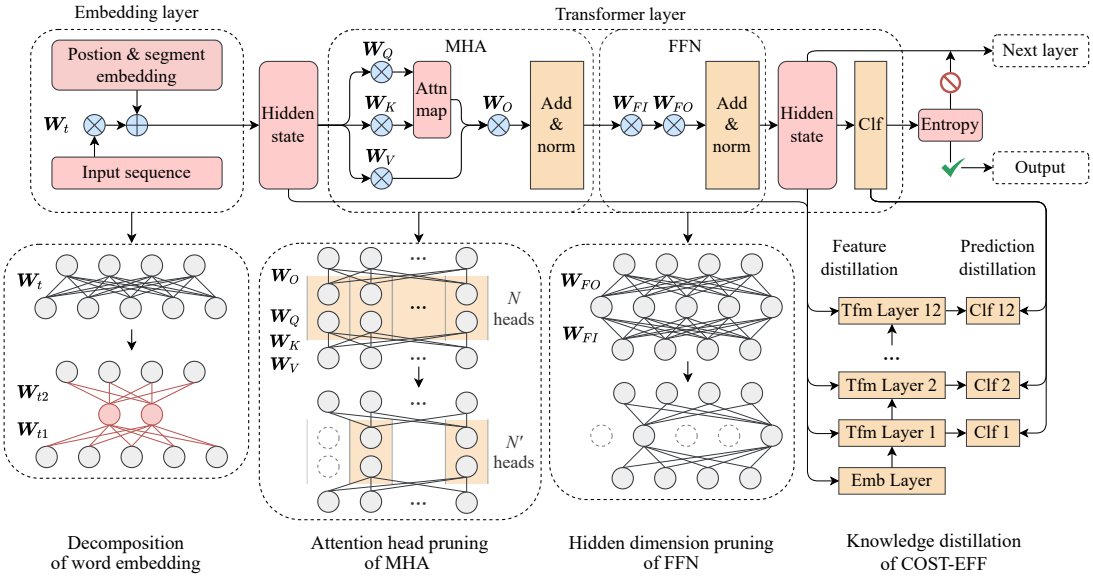

Figure 2: Illustration of COST-EFF. The upper part is the general architecture and forward procedure of the model. The lower part is the slenderization details of corresponding modules, where grey circles denote the input and output dimensions of matrices and the lines connecting them are weights.

of $\boldsymbol{W}_Q$, $\boldsymbol{W}_K$, $\boldsymbol{W}_V$ and $\boldsymbol{W}_{FI}$ are compressed. On the contrary, the input dimensions of $\boldsymbol{W}_O$ and $\boldsymbol{W}_{FO}$ are compressed. Thus, the dimension of the hidden states remains intact in COST-EFF. Also, as a single but drastic pruning would usually cause damage hard to recover, we use iterative pruning (Tan and Motani, 2020) in COST-EFF which gradually prunes out insignificant modules.

3.3 Dynamic Acceleration

3.3.1 Inference with Early Exiting

Unlike static compression, early exiting dynamically determines the computation at inference time, depending on the complexity of inputs and the perplexity of the model. Specifically, we use layer-wise early exiting, as shown in Figure 1, by plugging in a classifier at each Transformer layer.

Following the experimental results of ElasticBERT (Liu et al., 2022), entropy-based exiting generally outperforms patience-based, we use entropy of the classifier output as the exit condition defined as $\mathrm{H}(x) = -\sum_{i=1}^{C} \mathrm{p}(i) \ln \mathrm{p}(i)$, where $\mathrm{p}(\cdot)$ is the probability distribution calculated by softmax function and $H(x)$ is the entropy of the probability distribution x. If the entropy is greater than a given threshold H_T, the model is hard to make a prediction at that state. Conversely, the model tends to make a certain prediction with small entropy, where the difference in the probability distribution is large and dominant.

3.3.2 Training Multiple Exits

When training the model with multiple exits, the loss function of each exit is taken into account. DeeBERT (Xin et al., 2020) introduced a two-stage training scheme where the backbone model and exits are separately trained. However, only with the loss of the final classifier and the gradients that back-propagate, shallow layers of the backbone model are not capable of making confident predictions but rather serve the deep layers. Thus, it is necessary to introduce the loss of intermediate classifiers while training and calculating the Taylor expansion-based structure importance as Equation 2 in COST-EFF.

To balance the gradient from multiple classifier losses, we use gradient equilibrium following (Li et al., 2019) and scale the gradient of layer k to

$$\nabla'_{\boldsymbol{w}_k} \mathcal{L} = \frac{1}{L-k+1} \sum_{i=k}^{L} \nabla_{\boldsymbol{w}_k} \mathcal{L}_i, \qquad (3)$$

where L is the model depth, $\nabla_{\boldsymbol{w}_k} \mathcal{L}_i$ is the gradient propagates from layer i down to layer k and $\nabla'_{\boldsymbol{w}_k} \mathcal{L}$ is the rescaled gradient.

3.4 Collaborative Training of COST-EFF

3.4.1 Training with Knowledge Distillation

The small size and capacity of the compressed model make it hard to restore performance only with fine-tuning. Whereas knowledge distillation

is used as a complement that transfers the knowledge from the original teacher model to the compressed student model. In this paper, we aim to distill the prediction and intermediate features (i.e., hidden states) as depicted in Figure 2.

As the inconsistency between layers is observed (Xin et al., 2021), simply using ground-truth labels to train a compressed multi-exit model would result in severe contradictions. Given this, we first distill the original model into a multi-exit $\text{BERT}_{\text{Base}}$ model with the same layers as the TA. Then, each layer output of TA is used as soft labels of the corresponding layer in COST-EFF as

$$\mathcal{L}_{pred} = \sum_{i=1}^{L} \text{CELoss}\left(\boldsymbol{z}_i^{TA}/T, \boldsymbol{z}_i^{CE}/T\right), \quad (4)$$

where $\boldsymbol{z}_i^{TA}$ and $\boldsymbol{z}_i^{CE}$ are the prediction outputs of TA and COST-EFF at the i-th layer, respectively. T is the temperature factor usually set as 1. Besides distilling the predictions, COST-EFF distills hidden states to effectively transfer the representations of TA to the student model. The hidden state outputs, denote as $\boldsymbol{H}_i (i = 0, 1, \cdots, L+1)$ including embedding output $\boldsymbol{H}_0$ and each Transformer layer output, are optimized as

$$\mathcal{L}_{feat} = \sum_{i=0}^{L+1} \text{MSELoss}\left(\boldsymbol{H}_i^{TA}, \boldsymbol{H}_i^{CE}\right). \quad (5)$$

3.4.2 COST-EFF Procedure

As mentioned in Section 3.4.1, COST-EFF first distills the model into a multi-exit TA model with the same number of layers. Specifically, we distill the predictions at this stage. Although feature distillation is typically more powerful, representations of the single-exit model are not aligned with the multi-exit model and will introduce inconsistencies during training. Such distillation masks the trivial implementations of different PLMs to be compressed, as well as preliminarily mitigates the inconsistency between layers with a larger and more robust model. Then, the TA model is used as both the slenderization backbone and the teacher of further knowledge distillation.

During slenderization, we integrate the loss of exits into Taylor expansion-based structure importance calculation. Compared to simply using the loss of the final classifier, multi-exit loss helps calibrate the slenderization by weighing structures' contribution to each subsequent exit instead of only the final layer. In this way, the trade-off between layers can be better balanced in the slenderized model. After slenderization, the recovery training is a layer-wise knowledge transferring from TA to COST-EFF with the objective of minimizing the sum of $\mathcal{L}_{pred}$ and $\mathcal{L}_{feat}$ which mitigates the contradictions of ground-truth label training on the slenderized multi-exit model.

4 Experimental Evaluation

4.1 Experiment Setup

Datasets We use four tasks of GLUE benchmark (Wang et al., 2019), namely SST-2, MRPC, QNLI and MNLI. The details of these tasks are shown in Table 1 and most categories of GLUE are covered.

Task	Category	Labels	Metric
SST-2	Single-sentence	2	Acc
MRPC	Paraphrase	2	F1
QNLI	Inference	2	Acc
MNLI	Inference	3	Acc

Table 1: Details of the datasets.

Comparative Methods We compare the following baselines and methods. (1) Different size of BERT models, namely $\textbf{BERT}_{\textbf{Base}}$, $\textbf{BERT}_{\textbf{6L-768H}}$ and $\textbf{BERT}_{\textbf{8L-256H}}$, fine-tuned based on the pre-trained models of (Turc et al., 2019). (2) Representative static compression methods. **DistilBERT** (Sanh et al., 2020) and **TinyBERT** (Jiao et al., 2020). (3) Dynamic accelerated methods. **DeeBERT** (Xin et al., 2020), **PABEE** (Zhou et al., 2020) and the pre-trained multi-exit model **ElasticBERT** (Liu et al., 2022).

Model Settings As the number of parameters profoundly impacts the capacity and performance, we have two comparison groups with similar model sizes inside each group. Models in the first group are with less than 20M parameters and the second group of models are of larger size above 50M parameters. The details of model settings can be found in Table 2. Notably, the results of DistilBERT are extracted from the original paper and the others are implemented by ourselves as the experiments involve different backbone models and training data. The implementation is with AdamW optimizer on a single 24GB RTX 3090 GPU, while train batch size is in {32, 64} and learning rate is in {2e-5, 3e-5, 4e-5} varying from tasks.

Model	Size				EE
	L	H	A	F	
$\text{BERT}_{\text{Base}}$	12	768	12×64	3072	
$\text{BERT}_{\text{8L-256H}}$	8	256	4×64	1024	
TinyBERT_4	4	312	12×26	1200	
$\text{DeeBERT}_{\text{12L-256H}}$	12	256	4×64	1024	✓
$\text{PABEE}_{\text{12L-256H}}$	12	256	4×64	1024	✓
$\text{COST-EFF}_{8\times}$	12	768	2×64	256	✓
$\text{BERT}_{\text{6L-768H}}$	6	768	12×64	3072	
TinyBERT_6	6	768	12×64	3072	
DistilBERT_6	6	768	12×64	3072	
$\text{ElasticBERT}_{\text{6L}}$	6	768	12×64	3072	✓
$\text{DeeBERT}_{\text{12L-512H}}$	12	512	8×64	2048	✓
$\text{PABEE}_{\text{12L-512H}}$	12	512	8×64	2048	✓
$\text{COST-EFF}_{2\times}$	12	768	6×64	1536	✓

Table 2: Settings of compressed models. L is the number of layers and H is the dimension of hidden states. A denotes the MHA size as $head\_num \times head\_size$, and the intermediate size of FFN is F. Models with a check sign in the EE column adopt early exiting.

4.2 Experiment Results

4.2.1 Overall Results

The results of COST-EFF and comparative methods are listed in Table 3. When counting parameters, we include the parameters of embeddings and use the vocabulary size of 30,522 as default. The FLOPs are evaluated by PyTorch profiler with input sequences padded or truncated to the default length of 128 tokens and are averaged by tasks.

In the first group, the models are highly compressed and accelerated, while the performance is retained at approximately 96.5% by $\text{COST-EFF}_{8\times}$, which is much better than the conventional pre-training and fine-tuning of $\text{BERT}_{\text{8L-256H}}$. Specifically, $\text{COST-EFF}_{8\times}$ out-performs TinyBERT_4 in all four tasks, suggesting that a slenderized model preserving all the layers is superior to a squat one. The slenderized architecture is more likely to extract hierarchical features for hard instances while expeditiously processing simple instances. For larger models, TinyBERT_6 with general distillation gains a slight advantage over $\text{COST-EFF}_{2\times}$. Whereas $\text{COST-EFF}_{2\times}$ has a smaller volume than TinyBERT_6 and does not require general distillation, the performance gap is not significant. Meanwhile, TinyBERT_6 without general distillation is dominated by $\text{COST-EFF}_{2\times}$ in both efficiency and effectiveness, indicating the necessity of TinyBERT general distillation. However, a large effort is required by general distillation which pre-trains a single model of a fixed size and computation. In case the computation demand changes, pre-training yet another model can be extremely time-consuming. Compared to TinyBERT, COST-EFF has advantages in both performance and flexible inference.

To demonstrate the effect of dynamic acceleration, we empirically select simple instances from the development set which are shorter (i.e., lower than the median non-padding length after tokenization). The results on simple instances exhibit extra improvements attributed to dynamic inference, which are hard to obtain with static models. Notably, shorter length does not always indicate simplicity. For entailment tasks like QNLI, shorter inputs would contain less information, which potentially aggravate the perplexity of language models. Also, we plot performance curves with respect to GLUE scores and FLOPs in Figure 3 and 4. The performance curves are two-dimensional and exhibit the optimality of different methods. Aiming at obtaining the model with smaller computation and performance, we focus on the models in the upper left part of the figure, which is the Pareto frontier plotted in dashed blue lines.

As depicted in Figure 3 and 4, both $\text{COST-EFF}_{8\times}$ and $\text{COST-EFF}_{2\times}$ outperform DistilBERT, DeeBERT, PABEE and BERT baselines. Compared with TinyBERT and ElasticBERT, COST-EFF is generally optimal. We find that early exiting reduces the upper bound of NLI performance, where both $\text{COST-EFF}_{2\times}$ and $\text{ElasticBERT}_{\text{6L}}$ are inferior to TinyBERT_6. This issue may stem from the inconsistency between layers. Given that the complex samples in the NLI task rely on high-level semantics, the shallow layers should serve the deeper layers rather than solving the task by themselves. However, this issue does not affect global optimality. As shown in Figure 3, $\text{COST-EFF}_{8\times}$ has non-dominated performance against TinyBERT_4 on QNLI and MNLI, demonstrating the flexibility of our approach.

The performance of models incorporating early exiting is substantially affected by each exit. In Figure 5, we plot the layer-wise performance of models with early exiting in the first group and the final performance of TinyBERT_4. $\text{COST-EFF}_{8\times}$ achieves the dominant performance compared to DeeBERT and PABEE. Compared to

Model	Params reduc.	FLOPs reduc.	SST-2	MRPC	QNLI	MNLI-m/mm
$\text{BERT}_{\text{Base}}$	1.0×	1.0×	93.1	90.5	91.7	84.4 / 84.5
$\text{BERT}_{\text{8L-256H}}$	7.6×	13.5×	88.4	84.7	86.6	77.5 / 78.4
TinyBERT_4	7.6×	18.6×	89.7	86.7	87.0	81.2 / 81.6
$\text{DeeBERT}_{\text{12L-256H}}$	6.0×	14.9×	87.5	85.0	86.8	77.6 / 78.5
$\text{PABEE}_{\text{12L-256H}}$	6.3×	14.5×	88.1	85.4	86.0	78.6 / 78.3
$\text{COST-EFF}_{8\times}$ (ours)	**7.9**×	**19.0**×	**90.6**	**87.1**	**87.8**	**81.3 / 81.8**
$\text{BERT}_{\text{6L-768H}}$	1.6×	2.0×	91.1	88.1	89.6	81.5 / 82.0
DistilBERT_6	1.6×	2.0×	91.3	-	89.2	82.2 / -
TinyBERT_6	1.6×	2.0×	91.6	89.2	**91.3**	**84.2 / 84.4**
TinyBERT_6 w/o GD	1.6×	2.0×	91.2	88.5	90.0	83.5 / 83.4
$\text{ElasticBERT}_{\text{6L}}$	1.6×	2.4×	91.2	89.4	90.5	83.2 / 83.2
$\text{DeeBERT}_{\text{12L-512H}}$	1.9×	2.2×	89.8	89.0	89.8	81.8 / 82.6
$\text{PABEE}_{\text{12L-512H}}$	2.0×	2.2×	89.7	86.9	89.2	81.6 / 81.9
$\text{COST-EFF}_{2\times}$ (ours)	**2.0**×	**2.4**×	**92.0**	**89.7**	90.9	83.7 / 83.8
simple input instances						
TinyBERT_4	7.6×	18.6×	90.1 (+0.4)	83.6 (-3.1)	87.0 (+0.0)	81.4 / **83.4** (+0.9)
$\text{COST-EFF}_{8\times}$ (ours)	**7.9**×	**20.3**×	**91.5** (+0.9)	**88.8** (+1.7)	**87.9** (+0.1)	**82.3 / 83.4** (+1.3)

Table 3: Results on GLUE development set. $\text{BERT}_{\text{Base}}$ is used as the baseline to evaluate the average compression and acceleration rate, i.e., Params reduc. and FLOPs reduc., which are the higher the better. TinyBERT is implemented by conducting task-specific distillation without data augmentation on the public general distilled models, while TinyBERT_6 w/o GD is initialized from pre-trained $\text{BERT}_{\text{6L-768H}}$ without general distillation. $\text{ElasticBERT}_{\text{6L}}$ is initialized from the first 6 layers of ElasticBERT without pooler. The best results are in bold and the second best results are underlined.

TinyBERT_4, $\text{COST-EFF}_{8\times}$ can achieve better performance from the 7th to 12th layer, further verifying our claim that slender models are superior to squat models in performance, benefiting from the preserved architecture and its ability to extract high-level semantics. Another way to obtain powerful multi-exit models is alternating the backbone from BERT to the pre-trained ElasticBERT (Liu et al., 2022). In view of fairness, we uniformly use BERT as the backbone of COST-EFF and comparative methods. Notably, our approach is well-adaptable to ElasticBERT and the advanced performance is exhibited in Appendix A.

4.2.2 Ablation Studies

Impact of knowledge distillation The ablation experiments of distillation strategies aim to evaluate the effectiveness of prediction and feature distillation. In this ablation study, the comparison methods are (1) ablating feature distillation and (2) alternating prediction distillation with ground-truth label training. The results shown in Table 4 indicate that both objectives are crucial.

Model	SST-2	MRPC	QNLI
$\text{COST-EFF}_{8\times}$	90.6	87.1	87.8
$-\mathcal{L}_{feat}$	87.5	86.8	86.4
$-\mathcal{L}_{pred}$	88.6	82.4	84.2

Table 4: Ablation results on GLUE development set with $8\times$ compression. Feature distillation is ablated in $-\mathcal{L}_{feat}$, while ground-truth label is used to replace prediction distillation in $-\mathcal{L}_{pred}$. FLOPs of two ablated methods are ensured more than $\text{COST-EFF}_{8\times}$.

Attributing to the imitation of hidden representations, $\text{COST-EFF}_{8\times}$ has an advantage of 1.6% in performance compared to training without feature distillation. Without prediction distillation, the performance drops more than 3.4%. Previous works of static compression, e.g., TinyBERT (Jiao et al., 2020) and CoFi (Xia et al., 2022), are generally not sensitive to prediction distillation in GLUE tasks, as the output distribution of the single-exit teacher model is generally in accordance with the ground-truth label. However,

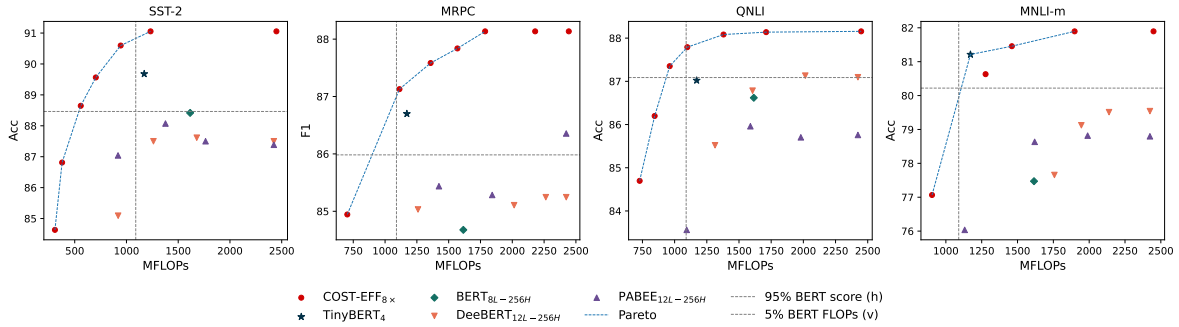

Figure 3: Performance curves of models with $8\times$ compression rate on GLUE development set. Horizontal grey line indicates the 95% of $\text{BERT}_{\text{Base}}$ performance and vertical line indicates 5% $\text{BERT}_{\text{Base}}$ FLOPs.

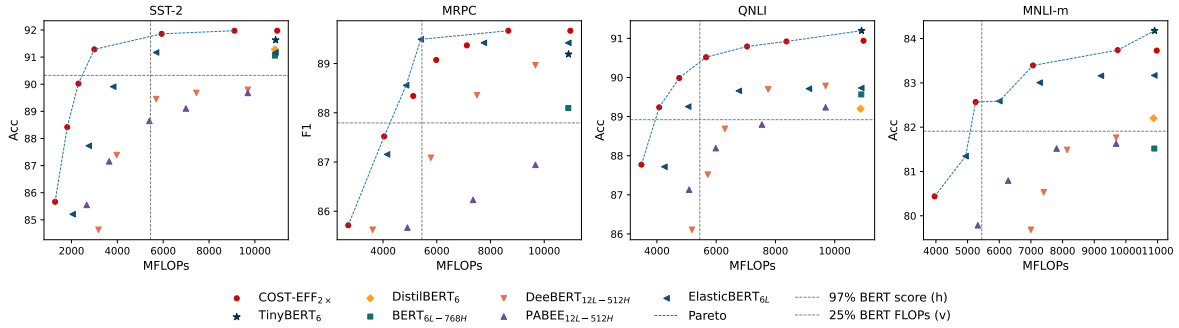

Figure 4: Performance curves of models with $2\times$ compression rate on GLUE development set. Horizontal grey line indicates the 97% of $\text{BERT}_{\text{Base}}$ performance and vertical line indicates 25% $\text{BERT}_{\text{Base}}$ FLOPs.

a large decrease in COST-EFF performance is observed in Table 4 if prediction distribution is ablated. The result indicates that pursuing the ground truth at shallow layers can deteriorate the performance of deep layers. Such inconsistency between shallow and deep layers commonly exists in early exiting models, which is particularly hard to balance by compressed models with small capacity. Instead, COST-EFF introduces an uncompressed TA model to mitigate the contradiction at an early stage and transfer the balance through prediction distillation.

Impact of collaborative training In this paper, we propose a collaborative approach for model slenderization and exit training, intended to calibrate the pruning of shallow modules. To validate the effectiveness of the training strategy, we ablate the collaborative training at different times. First, we implemented a two-stage training mode as DeeBERT does. Also, we implement COST-EFF$_{8\times}$ with exit loss ablated before and during slenderization. The layer-wise comparison of the above methods is shown in Figure 6.

Intuitively, two-stage training has an advantage on the final layer over collaborative training, as the inconsistency between layers is not introduced. However, the advantage diminishes in shallow layers, leaving the general performance unacceptable. Compared to slenderizing without exit loss, our approach has an advantage of 1.1% to 2.3%. Notably, slenderizing without calibration of exit can still achieve similar performance to COST-EFF at shallow layers, suggesting that the distillation-based training is effective in restoring performance. However, the inferior performance of deep layers indicates that the trade-off between layers is not well-balanced, since the slenderization is conducted aiming at optimizing the performance of the final classifier.

5 Conclusion

In this paper, we statically slenderize and dynamically accelerate PLMs in the pursuit of inference efficiency as well as preserving the capacity. To integrate the perspectives, we propose a collaborative optimization approach that achieves a mutual gain of static slenderization and dynamic acceleration. Specifically, the size of PLM is reduced in model width, and the inference is adaptable to the complexity of inputs without introducing redun-

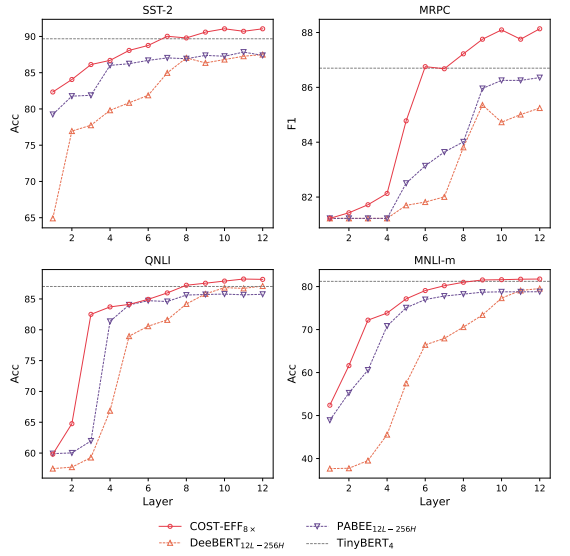

Figure 5: Layer-wise performance on GLUE development set. Horizontal lines indicate the final classifier performance of TinyBERT$_4$.

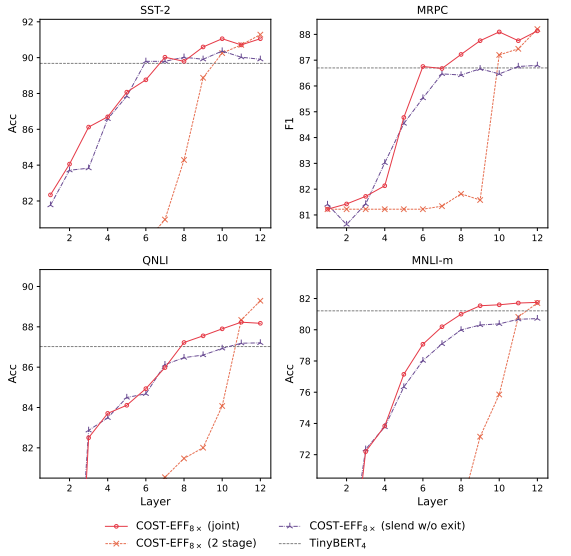

Figure 6: Layer-wise performance (zoomed) of collaborative training ablation study on GLUE development set. Horizontal lines indicate the final classifier performance of TinyBERT$_4$. COST-EFF$_{8\times}$ (slend w/o exit) is slenderized without the calibration of exits.

dancy for simple inputs and inadequacy for hard inputs. Comparative experiments are conducted on GLUE benchmark and verify the Pareto optimality of our approach at high compression and acceleration rate.

Limitations

COST-EFF currently has the following limitations. If they are addressed in future works, the potential capabilities of COST-EFF can be unleashed. (1) During the inference of dynamic early exiting models, the conventional practice is to set batch size as 1 to better adjust the computational according to individual input samples. However, such a setting is not always effective as a larger batch size is likely to reduce inference time, whereas input complexities inside a batch may differ significantly. Thus, it is inspiring to investigate a pipeline that gathers samples with similar expectations of complexity into a batch while controlling the priority of batches with different complexities to achieve parallelism. (2) We choose natural language understanding (NLU) tasks to study compression and acceleration following the strong baselines TinyBERT (Jiao et al., 2020) and ElasticBERT (Liu et al., 2022). However, the extensibility of COST-EFF is yet to be explored in more complex tasks including natural language generation, translation, etc. So far, static model compression is proved to be effective in complex tasks (Gupta and Agrawal, 2022) and we are seeking the extension of dynamic inference acceleration on different tasks using models with an iterative process.

Acknowledgements

This work was supported by National Natural Science Foundation of China (No. 61976207, No. 61906187).

References

Yoshua Bengio, Yann LeCun, et al. 2007. Scaling learning algorithms towards ai. *Large-scale kernel machines*, 34(5):1–41.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Angela Fan, Edouard Grave, and Armand Joulin. 2020. Reducing transformer depth on demand with structured dropout. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Manish Gupta and Puneet Agrawal. 2022. Compression of deep learning models for text: A survey. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 16(4):1–55.

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

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

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Hao Li, Hong Zhang, Xiaojuan Qi, Ruigang Yang, and Gao Huang. 2019. Improved techniques for training adaptive deep networks. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 1891–1900. IEEE.

Weijie Liu, Peng Zhou, Zhiruo Wang, Zhe Zhao, Haotang Deng, and Qi Ju. 2020. FastBERT: a self-distilling BERT with adaptive inference time. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6035–6044, Online. Association for Computational Linguistics.

Xiangyang Liu, Tianxiang Sun, Junliang He, Jiawen Wu, Lingling Wu, Xinyu Zhang, Hao Jiang, Zhao Cao, Xuanjing Huang, and Xipeng Qiu. 2022. Towards efficient NLP: A standard evaluation and a strong baseline. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3288–3303, Seattle, United States. Association for Computational Linguistics.

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

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 14014–14024.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. 2017. Pruning convolutional neural networks for resource efficient inference. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv:1910.01108 [cs]*. ArXiv: 1910.01108.

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

Chong Min John Tan and Mehul Motani. 2020. Dropnet: Reducing neural network complexity via iterative pruning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 9356–9366. PMLR.

Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. 2016. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2464–2469. IEEE.

Frederick Tung and Greg Mori. 2019. Similarity-preserving knowledge distillation. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 1365–1374. IEEE.

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

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

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th*

International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net.

Ziheng Wang, Jeremy Wohlwend, and Tao Lei. 2020. Structured pruning of large language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6151–6162, Online. Association for Computational Linguistics.

Mengzhou Xia, Zexuan Zhong, and Danqi Chen. 2022. Structured pruning learns compact and accurate models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1513–1528.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2246–2251, Online. Association for Computational Linguistics.

Ji Xin, Raphael Tang, Yaoliang Yu, and Jimmy Lin. 2021. BERxiT: Early exiting for BERT with better fine-tuning and extension to regression. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 91–104, Online. Association for Computational Linguistics.

Runxin Xu, Fuli Luo, Chengyu Wang, Baobao Chang, Jun Huang, Songfang Huang, and Fei Huang. 2022. From dense to sparse: Contrastive pruning for better pre-trained language model compression. In *Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI)*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. 2020. Bert loses patience: Fast and robust inference with early exit. *Advances in Neural Information Processing Systems*, 33:18330–18341.

A ElasticBERT as Backbone

We implemented COST-EFF with the backbone of ElasticBERT and obtain better performance than BERT backbone. The global results are listed in Table 5. Also, we plot performance curves and layer-wise performance in Figure 7 and Figure 8, respectively.

Model	Params reduc.	FLOPs reduc.	SST-2	MRPC	QNLI	MNLI-m/mm
$\text{BERT}_{\text{Base}}$	1.0×	1.0×	93.1	90.5	91.7	84.4 / 84.5
$\text{COST-EFF}_{8\times}$ (BERT)	7.9×	19.0×	90.6	87.1	87.8	81.3 / 81.8
$\text{COST-EFF}_{8\times}$ (ElasticBERT)	7.9×	**19.1**×	**90.8**	**88.1**	**89.0**	**81.6 / 82.3**

Table 5: Results of COST-EFF on GLUE development set with BERT and ElasticBERT as the backbone.

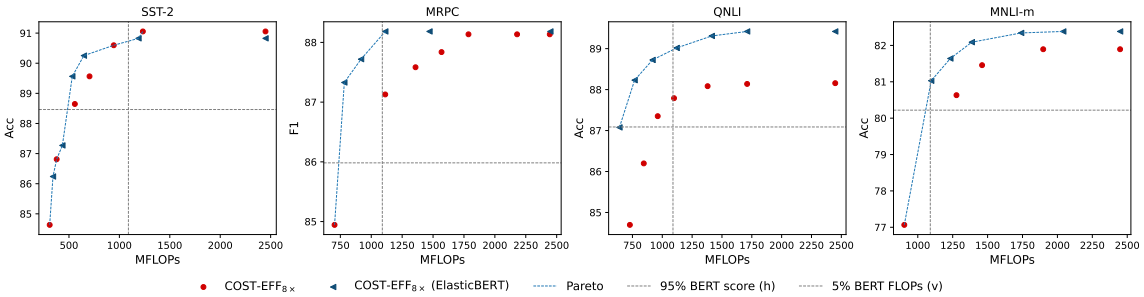

Figure 7: Performance curves COST-EFF on GLUE development set with BERT and ElasticBERT. Horizontal grey line indicates the 95% of $\text{BERT}_{\text{Base}}$ performance and vertical line indicates 5% $\text{BERT}_{\text{Base}}$ FLOPs.

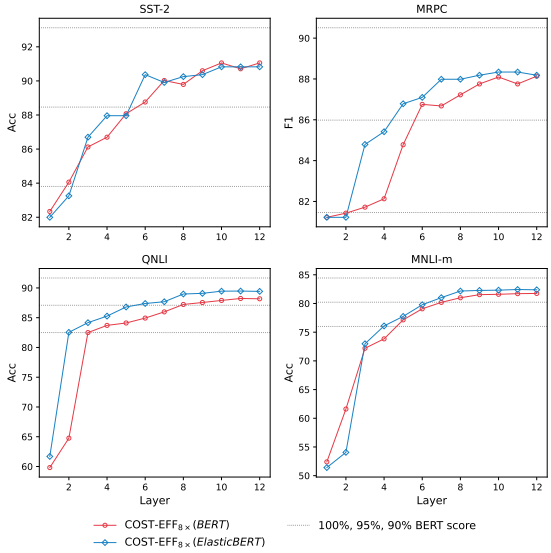

Figure 8: Layer-wise performance on GLUE development set. Horizontal lines indicate 100%, 95% and 90% of $\text{BERT}_{\text{Base}}$ performance from top to bottom.