BADGE: Speeding Up BERT Inference after Deployment via Block-wise BypAsses and DiverGence-Based Early Exiting

Wei Zhu[1], **Peng Wang**[2], **Yuan Ni**[3], **Guotong Xie**[3], **Xiaoling Wang**[1]*
[1]East China Normal University
[2] Tomorrow Advancing Life
[3] Pingan Health Tech

Abstract

Early exiting can reduce the average latency of pre-trained language models (PLMs) via its adaptive inference mechanism and work with other inference speed-up methods like model pruning, thus drawing much attention from the industry. In this work, we propose a novel framework, BADGE, which consists of two off-the-shelf methods for improving PLMs' early exiting. We first address the issues of training a multi-exit PLM, the backbone model for early exiting. We propose the novel architecture of block-wise bypasses, which can alleviate the conflicts in jointly training multiple intermediate classifiers and thus improve the overall performances of multi-exit PLM while introducing negligible additional flops to the model. Second, we propose a novel divergence-based early exiting (DGE) mechanism, which obtains early exiting signals by comparing the predicted distributions among the current layer and the previous layers' exits. Extensive experiments on three proprietary datasets and three GLUE benchmark tasks demonstrate that our method can obtain a better speedup-performance trade-off than the existing baseline methods.

1 Introduction

Since BERT (Devlin et al., 2019), the pre-trained language models (PLMs) have become the default state-of-the-art (SOTA) models for natural language processing (NLP). Recent years have witnessed the rise of many PLMs, such as GPT (Radford et al., 2019), XLNet (Yang et al., 2019), and ALBERT (Lan et al., 2020), and so forth. These BERT-style models achieved considerable improvements in many Natural Language Processing (NLP) tasks by pre-training on the unlabeled corpus and fine-tuning on labeled tasks, such as text classification, natural language inference (NLI), and named entity recognition (NER). Despite their outstanding performances, their industrial usage is still limited by the high latency during inference (Tambe et al., 2020; Zhu, 2021). In addition, a special feature of consumer queries is that there are time intervals when the number of queries is exceptionally high. For example, food search engines will be used more often during dinner hours than usual. Thus, deployed pre-trained models need to adjust their latency dynamically.

A branch of literature focuses on making PLMs' inference more efficient via adaptive inference (Zhou et al., 2020; Xin et al., 2020; Liu et al., 2020). The idea of adaptive inference is to process simple examples with only shallow layers of BERT and more difficult queries with deeper layers, thus significantly speeding up the inference time on average while maintaining high accuracy. Early exiting is one of the essential adaptive inference methods (Bolukbasi et al., 2017). As depicted in Figure 1, it implements adaptive inference by installing an early exit, i.e., an intermediate prediction layer, at each layer of PLM (multi-exit PLM) and early exiting "easy" samples to speed up inference. All the exits are jointly optimized at the training stage with BERT's parameters. At the inference stage, an early exiting strategy is designed to decide whether to exit at each layer given the currently obtained predictions (from previous and current layers) (Teerapittayanon et al., 2016; Kaya et al., 2019; Xin et al., 2020; Zhou et al., 2020). In this mode, different samples can exit at different depths. The average speedup ratio can be easily controlled with certain hyper-parameters without redeploying the model services or maintaining a group of models.

In this work, we propose a novel framework, BADGE, to improve the early exiting performances of PLMs. BADGE consists of two modifications to the current early exiting literature. First, we propose to add a block-wise bypass to each transformer block so that two different representations can be produced, one for the current layer's exit

*Corresponding author: xlwang@cs.ecnu.edu.cn.

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 5: Industry Track, pages 500–509

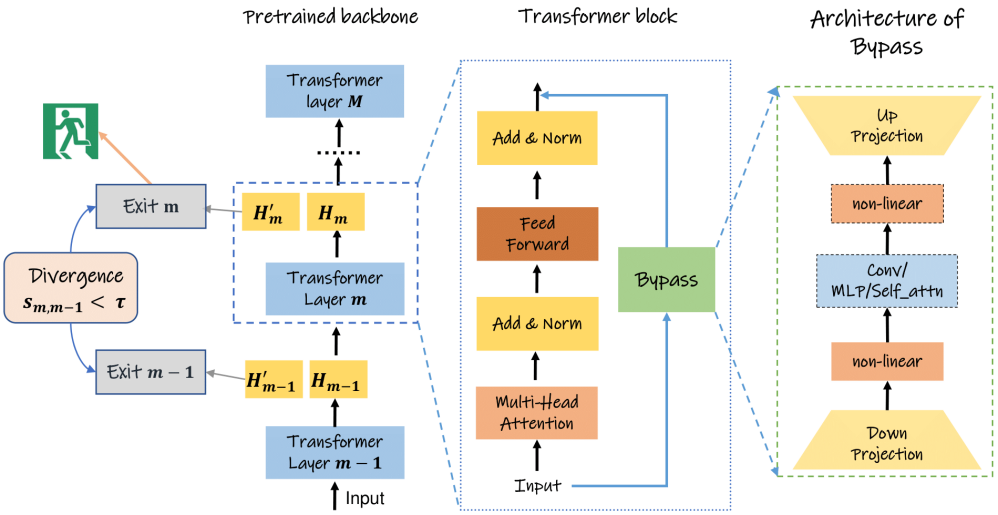


Figure 1: The overview of our BADGE framework.

and the other passed to the next transformer block. With this mechanism, an intermediate transformer block will not be distracted by the conflicting tasks in multi-exit BERT training, thus providing high-quality representations for the subsequent layers. As a result, the overall performance of the multi-exit BERT will improve. Second, we propose a novel divergence-based early exiting method (DGE), which is effective in mining early exiting signals by comparing the predicted probability distributions of the current and previous layers.

Extensive experiments and ablation studies are conducted on the GLUE benchmark datasets. The experimental results show that: (a) training multi-exit PLM with block-wise bypasses consistently performs better than the previous SOTA multi-exit model training methods, thus providing a better backbone for early exiting. (b) we show that, with the same multi-exit PLM backbone, our DGE method can provide better efficiency-performance trade-offs than the previous SOTA early exiting methods. Thus, with our framework, BADGE, a PLM can achieve significantly better early exiting performances.

The main contributions of our BADGE framework are two-fold:

(a) We propose a novel method, block-wise bypass, to improve the training of multi-exit PLMs.

(b) We propose a novel divergence-based early exiting method, DGE, which outperforms the previous SOTA early exiting methods.

2 Related Work

Due to the length limit, readers are referred to Appendix A for more related works on more inference speedup methods and dynamic early exiting mechanisms.

2.1 Early exiting

Early exiting requires a multi-exit network, a neural network backbone with an intermediate classifier (or exit) installed on each layer. Early exiting literature mainly focuses on the design of the early exiting strategies (Teerapittayanon et al., 2016; Xin et al., 2020; Kaya et al., 2019; Xin et al., 2021; Sun et al., 2022; Zhou et al., 2020; Schuster et al., 2021). However, the literature address less attention to the training of the multi-exit neural network. There are three types of training methods for training the multi-exit neural network: (a) joint training (JT) (Teerapittayanon et al., 2016; Zhou et al., 2020), that is, all the exits are jointed optimized together with the fine-tuning of BERT. (b) two-stage training method (2ST) (Liu et al., 2020; Xin et al., 2020), which first fine-tunes the backbone BERT and the last layer's exit till convergence and trains only the intermediate exits by distilling knowledge from the last exit. (c) BERxiT (Xin et al., 2021) propose an alternating training (ALT) method, combining 2ST and JT.

Our work complements the literature on early exiting by proposing the BADGE framework to improve early exiting performance via the novel design of block-wise bypasses and a novel early exiting mechanism.

3 Methodology

3.1 Block-wise bypasses

We now present the core of our novel BADGE framework: the block-wise bypasses (depicted in Figure 1). Denote the representations of the input sentence from the previous transformer block as H_{m-1}. Initially, H_{m-1} will go through the current transformer block, first the multi-head self-attention (MHSA) module with the residue to become $H_{m,MHA}$, then the positional feed-forward (FFN) module with residue, to become $H_{m,F}$, and finally a LayerNorm operation to output H_m.

We would like to employ an efficient bypass B_m to adjust the current layer's representations to fit the task better. B_m is simple in architecture (On the right side of Figure 1). Upon receiving the input H_{m-1}, B_m down-projects it to $H^{(1)}_{m,B}$, from dimension d to dimension r (where $r << d$) using a down-projection matrix $W_{down} \in \mathbb{R}^{d\times r}$. The $H^{(1)}_{m,B}$ will go through a non-linear activation function g_1, and become $H^{(2)}_{m,B}$. $H^{(2)}_{m,B}$ will then be up-projected to $H^{(3)}_{m,B}$ with dimension d, by an up-projection matrix $W_{up} \in \mathbb{R}^{r\times d}$. We refer to r as the bottleneck dimension. Formally, B_m can be expressed as:

$$H^{(3)}_{m,B} \leftarrow g_1(H_{m-1}W_{down})W_{up}. \quad (1)$$

Finally, the current transformer block will output two representations, H_m, which is the original intermediate representation, and $H^{'}_m$, which is modified by the bypass, via:

$$\begin{aligned} H^{'}_m &= \text{LayerNorm}(H^{(3)}_{m,B} + H_{m,F}), \\ H_m &= \text{LayerNorm}(H_{m,F}). \end{aligned} \quad (2)$$

H_m will be passed to the following transformer block as input, and $H^{'}_m$ is the modified representation which will be the input of the current layer's exit. We rely on the bypass B_m to extract task-specific representations to facilitate the learning of the exit. Note that the introduction of B_m will not bring little computational overhead since (a) the two representations H_m and $H^{'}_m$ can be calculated in a single forward pass; (b) the bottleneck dimension r can be very small, like 16, so that the added parameters or flops are negligible compared to the pre-trained backbone.

Our work is inspired by the recent work in parameter efficient tuning (He et al., 2021). However, our work is different in the following three aspects: (a) our work deals with the training of multi-exit BERT, not the parameter-efficient tuning settings. (b) Our work introduces the block-wise bypasses, while He et al. (2021) only considers the bypasses around the feed-forward layer or the self-attention layer of a transformer block. We will show in our experiments that block-wise bypasses perform better under our settings. (c) We add encoding operations inside the bypasses, which proves to be beneficial.

3.2 Divergence-based Early Exiting

We now introduce our divergence-based early exiting method, DGE. The inference procedure is illustrated in Figure 1. Assume the forward pass has reached layer m. Denote the divergence score between the prediction results of layer m and layer $m^{'}$ $(m > m^{'})$ as $s_{m,m'} = S(p_m, p_{m'}) \in \mathbf{R}$, where $S(\cdot,\cdot)$ is a divergence measure of two probability distributions. The smaller the value of $s_{m,m'}$, the predicted distributions p_m and $p_{m'}$ are more consistent with each other. Denote cnt_m as the number of previous layers that have a predicted distribution that is consistent with p_m. At layer m, cnt_m is calculated as:

$$cnt_m = \sum_{i=1}^{m-1} \mathbb{1}(s_{m,i} < \tau), \quad (3)$$

where $\tau > 0$ is a pre-defined threshold, and $\mathbb{1}(\cdot)$ is the indicator function. $\mathbb{1}(x)$ is equal to 1 if x is true, and equal to 0 if x is not true. If cnt_m reaches the pre-defined patience value t, the model stops inference and exits early. Otherwise, the model goes to the next layer. If the model does not exit early at intermediate layers, the model uses the final classifier f_M for prediction. Unless stated otherwise, we will mainly use the knowledge distillation objective (Hinton et al., 2015) as the divergence measures:[1]

$$S(p_i, p_j) = \mathcal{L}_{KD}(p_i, p_j) = -\sum_{k=1}^{K} p_i(k)\log(p_j(k)). \quad (4)$$

Note that the above-described framework is flexible and general, wrapping the patience-based method (Zhou et al., 2020) as a particular case. In Zhou et al. (2020), the value of $s_{i,i-1}$ is 0 only

[1] In our initial experiments, we find that different divergence measures like Kullback–Leibler divergence and Jensen-Shannon divergence perform comparably. Thus we adopt the one with the most simple form.

when the labels predicted by two layers are identical; otherwise, it is set to 1. However, compared to PABEE, our method has the following advantages: (a) even if the the current and previous layers give the same label, their output distributions might differ. Thus, PABEE might exit early even when the layers have not reached a consensus in the predictions. Thus, PABEE's early exiting performances with low patience parameters may not be reliable. Our method compares the intermediate layers in the distribution level, providing more reliable exiting signals. (b) PABEE can not adjust the speedup ratio flexibly, while our method can achieve different speedup ratios by setting patience and threshold parameters.

4 Experiments

4.1 Datasets

We evaluate our proposed approach to the classification tasks on three tasks from GLUE benchmark (Wang et al., 2018) (RTE, MRPC, MNLI) and three proprietary natural language understanding tasks (QID, DoS, QDR) we collected for developing and testing question-answering or dialogue systems. The detailed introductions of the datasets are put in Appendix B.

4.2 Baseline methods

For multi-exiting BERT training, we compare our BADGE framework with the following baselines: (a) joint training (JT) (Zhou et al., 2020; Teerapittayanon et al., 2016); (b) two-stage training (2ST) (Liu et al., 2020; Xin et al., 2020); (c) alternating training (ALT) (Xin et al., 2021); (d) Gradient equilibrium (GradEquil) (Li et al., 2019), which incorporates JT with gradient adjustments; (e) fine-tuning ALBERT with a single exit of different depths separately (Single-exits); (f) Global Past-Future, which develops a series of techniques to enhance the performances of lower layers by mimicking the deeper layers. We also implement JT with a multi-head attention exit (Liu et al., 2020) (JT+MHA-exit).

We compare the early exiting performances of our DGE method with the following early exiting methods: (a) Entropy-based method (Entropy) originated from (Teerapittayanon et al., 2016); (b) Maximum probability-based method (Maxprob) (Schwartz et al., 2020); (c) Patience-based method (Patience) (Zhou et al., 2020); (d) learning-to-exit based method (LTE) proposed by Xin et al. (2021), which incorporates a meta layer to evaluate the confidence of the current exit.

4.3 Exprimental settings

For the GLUE tasks, we use the open-sourced ALBERT-base (Lan et al., 2020) as the backbone. And for the three proprietary tasks, we use a pre-trained Chinese ALBERT-base[2]. In the ablation studies, we also consider BERT-base (Devlin et al., 2019) (in English and in Chinese). Our BADGE model with block-wise bypasses is optimized following the JT method with simple linear exits. Since we are training multi-exit models in which each layer's exit has a performance score, we mainly focus on improving S-avg, which denotes the cross-layer average score for a given metric S. We also report S-best, the best score among all the layers for the given metric S.

Following prior work on input-adaptive inference (Teerapittayanon et al., 2016; Kaya et al., 2019), inference is on a per-instance basis, i.e., the batch size for inference is set to 1.[3] The average speedup ratio on the test set of each task will be reported, which is defined as Speedup $= 1 - \frac{\sum_1^{N_{test}} t_i}{\sum_1^{N_{test}} T_i}$, where N_{test} is the number of samples on the test set, t_i is the inference time under early exiting, and T_i is the inference time without early exiting.

We implement our BADGE and all the baselines on the base of Hugging Face's Transformers (Wolf et al., 2020). Experiments are conducted on four Nvidia RTX 3090 GPUs. We report the median performance over five runs with different random seeds. More detailed settings regarding hyper-parameters can be found in Appendix C.

4.4 Overall comparison

We first compare our BADGE method with the previous best-performing training methods of multi-exit PLMs. Table 1 reports the results. The following takeaways can be made.

BADGE outperforms the baselines With the help of block-wise bypasses, our BADGE frame-

[2]This model is pre-trained by us, and this model is distilled from an open-sourced ALBERT-large model (available at `https://huggingface.co/uer/albert-large-chinese-cluecorpussmall`) on our corpus following the pretraining distillation pipeline of TinyBERT (Jiao et al., 2019)

[3]This setting is consistent with the common scenario in the industry where individual requests from different users (Schwartz et al., 2020) come at different time points.

	RTE		MRPC		MNLI		QID		DoS		QDR	
ALBERT-base fine-tuning												
	acc		acc-f1		acc		acc		mf1		f1	
ALBERT	77.6		89.1		83.7		91.9		88.0		87.3	
Multi-exit ALBERT fine-tuning												
	acc-avg	acc-best	acc-f1-avg	acc-f1-best	acc-avg	acc-best	acc-avg	acc-best	mf1-avg	mf1-best	f1-avg	f1-best
Single exits	67.5	77.6	85.4	89.1	76.4	83.7	89.5	92.3	82.9	88.0	83.3	87.3
2ST	68.9	77.6	83.0	89.1	76.2	83.7	89.3	91.9	82.5	88.0	81.9	87.3
JT	66.8	72.5	84.4	87.9	76.0	83.8	89.2	91.1	81.6	87.3	82.2	87.2
JT+MHA exit	67.5	75.9	84.8	87.9	76.8	83.2	89.3	91.5	81.7	87.6	82.3	87.1
GradEquil	67.3	77.4	84.2	88.4	76.5	83.6	89.2	91.8	82.4	88.0	82.4	87.0
ALT	68.5	77.8	84.6	88.3	76.6	82.9	88.6	90.6	82.3	87.8	81.5	86.8
Global Past-Future	68.1	78.4	84.3	88.1	75.9	82.9	89.1	92.3	82.4	87.9	82.4	87.1
BADGE	**70.2**	77.9	**86.0**	89.6	**77.8**	83.8	**90.0**	92.4	**83.2**	88.1	**84.3**	87.2

Table 1: Experimental results of models with ALBERT backbone (English and Chinese) on the GLUE benchmark datasets and our three proprietary datasets.

work consistently outperforms the baseline methods in terms of the average performances across all the intermediate layers and the cross-layer best scores. The experimental results are foreseeable: introducing the block-wise bypasses can help the intermediate transformer layer to concentrate on providing hidden representations, while the bypasses can provide features more suitable for the current layer's exit. In this way, both the cross-layer best and cross-layer average scores can improve.

Demonstrating that BADGE does not achieve improvements by merely adding more parameters It is natural to question whether the performance gains of BADGE are from additional parameters. With the bottleneck dimension $r = 16$, our bypass architecture introduces 26k parameters per layer. A multi-head attention exit (Liu et al., 2020) with a bottleneck dimension $r = 32$ introduces 98k parameters per layer. As shown in Table 1, JT with MHA exits performs better than JT (with simple linear exits). However, our BADGE method successfully outperforms JT with MHA exits with fewer parameters. The results demonstrate that BADGE's advantages come from designing our bypass mechanisms.

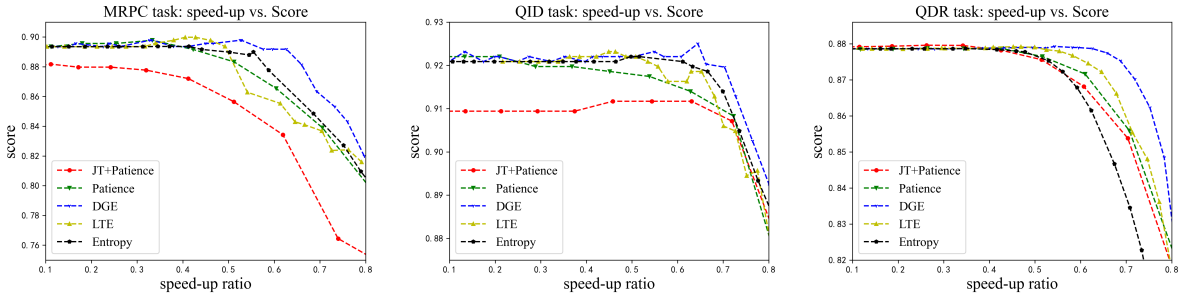
4.5 Dynamic early exiting performances

We compare our DGE method with the previous best-performing early exiting methods. For the patience-based method (Zhou et al., 2020), early exiting is run on different patience parameters. For the other methods, we run early exiting under different confidence thresholds or patience parameters so that the speedup-performance curves consist of at least 20 points evenly distributed across the interval $(0, 1)$ of speedup ratios. The speedup-performance curves for the MRPC, QID, and QDR tasks are plotted in Figure 2.

The following takeaways can also be made from Figure 2: (a) With the same backbone model trained with our BADGE framework, our DGE method achieves better speedup-performance tradeoffs than the previous SOTA early exiting methods, especially when the speedup ratio is large. (b) The comparison between Patience (with the model trained with BADGE) and JT+Patience (with the model trained with the JT method) demonstrates that our BADGE can provide superior backbones for early exiting and consistently result in superior performances under different speedup requirements. (c) Note that the pre-trained Chinese ALBERT-base model we use is a compressed model from a larger one. Our method can further speed up its inference, proving that it can work well with other model inference speedup methods like Jiao et al. (2019).

4.6 Ablation studies

Ablation on the placement of bypasses Since we want to provide separate hidden states in a single forward pass on the BERT backbone, our bypasses must be added to the end of the transformer layer. To show that the design of our block-wise bypasses is essential, we now consider another two placements of bypasses: (a) placed around the multi-head self-attention module, denoted as MHSA bypasses; (b) placed around the positional feed-forward module (FFN bypasses). Note that MHSA bypasses can not provide two different hidden states at the end of the transformer block within a single forward pass, thus slowing down inference speed. The results on RTE and QID tasks are reported in Table 2. We can see that: (a) FFN bypasses perform comparably with the block-wise bypasses; (b) MHSA bypasses perform worse than the block-wise bypasses since, in this setting, the



(a) MRPC (b) QID (c) QDR

Figure 2: The speedup-score curves on the MRPC, QID and QDR datasets.

	RTE		QID	
	acc-avg	acc-best	acc-avg	acc-best
block-wise bypasses	**70.2**	77.9	**90.0**	92.4
MHSA bypasses	68.4	77.3	89.4	91.7
FFN bypasses	69.6	77.8	89.7	92.1

Table 2: Experimental results for different placement settings for the bypasses.

	RTE		QID	
	acc-avg	acc-best	acc-avg	acc-best
r = 64	70.2	77.8	89.9	92.1
r = 32	70.1	77.6	90.0	92.2
r = 16	**70.2**	77.9	**90.0**	92.4
r = 8	70.0	78.0	89.8	92.3
r = 4	69.9	77.8	89.7	92.0

Table 3: Experimental results for comparing different bottleneck dimensions. The acc-avg and acc-best scores are reported.

FFN module in a transformer block still has to conduct multi-task learning.

Different bottleneck dimensions r We set the bottleneck dimension r to be 16 for the main part of the experiments (as in Table 1). We now conduct experiments for r on the RTE and QID tasks, and Table 3 reports the results. From Table 3, we can see that: (a) smaller bottleneck dimensions do not result in significant performance drops. (b) bottleneck dimensions do not provide performance improvements, demonstrating that the superior performances of our BADGE indeed come from our block-wise bypass mechanism instead of the additional parameters.

Ablation on different PLMs To show that our BADGE framework is off-the-shelf and can work well with other pre-trained models, we now switch the backbone model to BERT-base (Devlin et al., 2019), or ElasticBERT (Liu et al., 2022). Due to limited length, the results are reported in Table 6 of Appendix D. The results demonstrate that BADGE outperforms the JT and ALT methods under different backbones by clear margins.

Also, note that the cross-layer average scores of BERT-base and ElasticBERT are lower than that of ALBERT-base, showing that ALBERT is more suitable for early exiting. We hypothesize that the ALBERT model employs a cross-layer parameter-sharing strategy. Thus, the representations given by intermediate blocks are more similar to one another, and the performances of its intermediate layers will be closer.

5 Conclusion

In this work, we propose a novel framework, BADGE, to improve the early exiting of pre-trained language models. First, we propose to add block-wise bypasses to facilitate the joint training of the intermediate exits, thus improving the overall performance of multi-exit PLMs. Second, we propose a novel divergence-based early exiting method, DGE, which can effectively mine the exiting signals by comparing the predicted distributions of the current and previous intermediate layers. DGE achieves better efficiency-performance trade-offs than the previous SOTA early exiting methods under the same multi-exit backbone. Our BADGE is off-the-shelf and can effectively speed up the inferences of PLMs with less performance degradation.

6 Acknowledgements

This work was supported by NSFC grants (No. 61972155 and 62136002), National Key R&D Program of China (No. 2021YFC3340700), Shanghai Trusted Industry Internet Software Collaborative Innovation Center, and the Research, Development, Demonstration and Application Project of Artificial Intelligence (No. GJHZ20200731095001005).

Limitations

Although our BADGE framework is shown to be effective in improving the multi-exit model training and early exiting, it still has certain limitations that need to be addressed in the future: (a) blockwise bypasses indeed introduce new parameters and additional flops. We would like to explore more parameter-efficient methods to improve multi-exit model training in future works. (b) In this work, we demonstrate our framework's performance on sentence classification or pair classification tasks. In future works, we would like to extend our work to broader tasks such as sequence labeling, relation extraction, and text generation. We would like to explore this aspect in future work.

Ethics Statement

Our BADGE framework is designated to improve the training of multi-exit BERT and dynamic early exiting performances. Our work can facilitate the deployment and applications of pre-trained models on devices with less powerful computation capabilities, making the state-of-the-art models accessible for everyone. In addition, we hope this technology can help reduce the carbon footprints of NLP-based applications. Furthermore, the GLUE datasets we experiment with are widely used in previous work. Thus, to our knowledge, our work does not introduce new ethical concerns.

References

Haoli Bai, Wei Zhang, Lu Hou, Lifeng Shang, Jing Jin, Xin Jiang, Qun Liu, Michael Lyu, and Irwin King. 2020. Binarybert: Pushing the limit of bert quantization. *arXiv preprint arXiv:2012.15701*.

Tolga Bolukbasi, J. Wang, O. Dekel, and Venkatesh Saligrama. 2017. Adaptive neural networks for efficient inference. In *ICML*.

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. 2019. Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556*.

Mitchell A Gordon, Kevin Duh, and Nicholas Andrews. 2020. Compressing bert: Studying the effects of weight pruning on transfer learning. *arXiv preprint arXiv:2002.08307*.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a unified view of parameter-efficient transfer learning. *ArXiv*, abs/2110.04366.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531.

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

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. *ArXiv*, abs/1909.10351.

Y. Kaya, Sanghyun Hong, and T. Dumitras. 2019. Shallow-deep networks: Understanding and mitigating network overthinking. In *ICML*.

Sehoon Kim, Amir Gholami, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. 2021. I-bert: Integer-only bert quantization. In *International conference on machine learning*, pages 5506–5518. PMLR.

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. *ArXiv*, abs/1909.11942.

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

Weijie Liu, P. Zhou, Zhe Zhao, Zhiruo Wang, Haotang Deng, and Q. Ju. 2020. Fastbert: a self-distilling bert with adaptive inference time. *ArXiv*, abs/2004.02178.

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.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. In *NeurIPS*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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

Tal Schuster, Adam Fisch, Tommi Jaakkola, and Regina Barzilay. 2021. Consistent accelerated inference via confident adaptive transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4962–4979, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Roy Schwartz, Gabriel Stanovsky, Swabha Swayamdipta, Jesse Dodge, and Noah A. Smith. 2020. The right tool for the job: Matching model and instance complexities. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6640–6651, Online. Association for Computational Linguistics.

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

Tianxiang Sun, Xiangyang Liu, Wei Zhu, Zhichao Geng, Lingling Wu, Yilong He, Yuan Ni, Guotong Xie, Xuanjing Huang, and Xipeng Qiu. 2022. A simple hash-based early exiting approach for language understanding and generation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2409–2421, Dublin, Ireland. Association for Computational Linguistics.

Thierry Tambe, Coleman Hooper, Lillian Pentecost, En-Yu Yang, Marco Donato, Victor Sanh, Alexander M. Rush, David M. Brooks, and Gu-Yeon Wei. 2020. Edgebert: Optimizing on-chip inference for multi-task nlp. *ArXiv*, abs/2011.14203.

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

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

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

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.

Canwen Xu, Wangchunshu Zhou, Tao Ge, Furu Wei, and Ming Zhou. 2020. Bert-of-theseus: Compressing bert by progressive module replacing. *arXiv preprint arXiv:2002.02925*.

Z. Yang, Zihang Dai, Yiming Yang, J. Carbonell, R. Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q. Weinberger, and Yoav Artzi. 2020a. Revisiting few-sample bert fine-tuning. *ArXiv*, abs/2006.05987.

Wei Zhang, Lu Hou, Yichun Yin, Lifeng Shang, Xiao Chen, Xin Jiang, and Qun Liu. 2020b. Ternarybert: Distillation-aware ultra-low bit bert. *arXiv preprint arXiv:2009.12812*.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. 2020. Bert loses patience: Fast and robust inference with early exit. *ArXiv*, abs/2006.04152.

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

Wei Zhu. 2021. Autonlu: Architecture search for sentence and cross-sentence attention modeling with redesigned search space. In *NLPCC*.

A Appendix: Related work

A.1 Inference acceleration methods

Since the rise of BERT, there are quite large numbers of literature devoting themselves to speeding up the inference of BERT. Standard method include direct network pruning (Zhu and Gupta, 2017; Xu et al., 2020; Fan et al., 2019; Gordon et al., 2020), distillation (Sun et al., 2019; Sanh et al., 2019; Jiao et al., 2020), Weight quantization (Zhang et al., 2020b; Bai et al., 2020; Kim et al., 2021) and Adaptive inference (Zhou et al., 2020; Xin et al., 2020; Liu et al., 2020). Among them, adaptive inference has drawn much attention. Adaptive inference aims to deal with simple examples with only shallow layers of PLMs, thus speeding up inference time on average.

A.2 Early exiting mechanisms

Early exiting requires a multi-exit BERT, a BERT backbone with an intermediate classifier (or exit) installed on each layer. Early exiting literature mainly focuses on the design of the early exiting strategies, that is, determining when an intermediate exit's prediction is suitable as the final model prediction. Score based strategies (Teerapittayanon et al., 2016; Xin et al., 2020; Kaya et al., 2019; Xin et al., 2021), prior based strategies (Sun et al., 2022) and patience based strategies (Zhou et al., 2020) have been proposed. Teerapittayanon et al. (2016) uses the entropy of an intermediate layer's predicted distribution to measure the in-confidence level and decide whether to exit early. PABEE determines whether to exit by comparing the current layer's prediction with the previous layers.

A.3 Introductions to more multi-exit BERT training methods

A.3.1 Two-stage training method

The two-stage (2ST) (Xin et al., 2020; Liu et al., 2020) training strategy divides the training procedure into two stages. The first stage is identical to the vanilla BERT fine-tuning, updating the backbone model and only the final exit. In the second stage, we freeze all parameters updated in the first stage and fine-tune the remaining exits separately:

$$\text{Stage1} : \mathcal{L}_{stage1} = \mathcal{L}_M^{CE}(y_i, f_M(x_i; \theta_M)) \quad (5)$$

$$\text{Stage2} : \mathcal{L}_{stage2} = \mathcal{L}_m^{CE}, m = 1, ..., M-1. \quad (6)$$

where $\mathcal{L}_m^{CE} = \mathcal{L}_m^{CE}(y_i, f_m(x_i; \theta_m))$ denotes the cross-entropy loss of m-th exit.

A.3.2 Alternating training

This method alternates between joint training and two-stage training across different optimization steps, and it was proposed by BERxiT (Xin et al., 2021):

$$\text{Odd} : \mathcal{L}_{stage1} = \mathcal{L}_M^{CE}(y_i, f_M(x_i; \theta_M)) \quad (7)$$

$$\text{Even} : \mathcal{L}_{joint} = \sum_{m=1}^{M} \mathcal{L}_m^{CE} \quad (8)$$

B Appendix for datasets and evaluation metrics

B.1 Introduction to our proprietary tasks

In this work, we experiment with three proprietary datasets collected by our teams. The samples in these datasets are collected from users of a real-world production environment and are manually annotated by an data annotation team.

Query intent classification in dialogues (denoted as QID) This task asks a model predict the intent of an utterance from a user in a dialogue. The dialogue history are also provided as a part of the input text sequence. The samples of this task is collected from the dialogues between customers and clerks in grocery stores, hotels, pharmaceutical stores and gyms. All the participants of the dialogues have signed the data collection agreements. We assign a single label to each user utterance, and the number of intent labels are 78. The evaluation metric is accuracy (acc).

Query-doc ranking task (denoted as QDR) In this task, for a given search query in the scientific domain (including topics like bio-medicine, drug development and artificial intellegence), a pool of documents are given. The documents are either blog posts from our institution or abstracts of academic journal papers from Medline[4]. The pool size is between 12 to 128. The relevant documents are labeled as 1, and the other are labeled as 0. The annotation is based on whether the content of a document can provide sufficient answers to the query. In this task, a model is expected to predict the relevancy label for each query-document pair with high accuracy. The evaluation metric for this task is F1 of query-document relevancy labels (f1).

Document classification (denoted as DoS) In this task, a pre-defined label is assign to each document, to help guide the users to find articles of interests. The collection of documents have the same sources with QDR. The number of labels are 109. In this task, a classification model reads the text contents and predicts the label of a document. We develop this dataset in the hope that the model can help us automatically organize the documents, or at least facilitate the human workers to do so to reduce costs. The evaluation metric for this task is macro F1, since this task has in-balanced label distributions (macro-f1).

B.2 Evaluation metrics of GLUE tasks

For RTE, the metric is accuracy (acc). For MRPC task, the metric the average of the accuracy and F1 score (acc-f1). For MNLI, the metric is the average of the accuracy score on the matched and mis-matched test subsets (denoted as acc).

[4] https://www.medline.com/

Category	Datasets	|train|	|dev|	|test|	$\|\mathcal{Y}\|$	Type	Labels
Single-sentence	QID	125365	15670	15670	78	intent classification	0, 1, ..., 77
	DoS	1083278	135410	135410	109	document classification	0, 1, ..., 108
Sentence-pair	MNLI	390702	2000	19647	3	NLI	entailment, neutral, contradiction
	RTE	2490	138	139	2	NLI	entailment, not entailment
	MRPC	3668	204	204	2	paraphrase	equivalent, not equivalent
	QDR	3568930	446116	446116	2	text matching	0, 1

Table 4: The statistics of datasets evaluated in this work. For MNLI task, the number of samples in the test set is summed by matched and mismatched samples. $|\mathcal{Y}|$ is the number of classes for a dataset.

B.3 Dataset splits

For the QID, QDR and DoS tasks, we randomly split the annotation samples train/dev/test splits with 8:1:1 ratio. Since the original test sets of GLUE are not publicly available, we follow Zhang et al. (2020a) and Mahabadi et al. (2021) to construct the train/dev/test splits as follows: (a) for datasets with fewer than 10k samples (RTE, MRPC), we divide the original validation set in half, using one half for validation and the other for testing. (b) for larger datasets (MNLI), we split 2k samples from the training set as the development set, and use the original development set as the test set. The detailed dataset statistics are presented in Table 4.

C Appendix for experimental settings

We add a classification layer after each intermediate layer of the PLM as the exits. The exit is a simple linear layer by default, but we also consider the MHA exit suggested by Liu et al. (2020). We fine-tune the multi-exit BERT with one of the four bypasses architectures (in Section 3.1) or without bypasses, and the bottleneck dimension r is set as 16. The bottleneck dimension for multi-head attentional exit is set to be 32.

We fine-tune models for at most 25 epochs with an Adam optimizer and warm-up. The warm-up steps are set to be 10% of the optimization steps. Early stopping with patience eight is performed, and the best checkpoint is selected based on the dev set performances. Since we are training multi-exit models in which each layer's exit have a performance score, we mainly focus on improving S-avg, which denotes the cross-layer average score for a given metric S. We also report S-best, the best score among all the layers for the given metric S.

We perform grid search for the following hyper-parameter search space: the {GELU, ReLU, SWISH, Tanh, Identity} for the activation functions in the bypasses; {16, 32, 64, 128, 512, 1024} for the batch size; {1e-5, 2e-5, 3e-5, 5e-5} for the learning rate. We implement our BADGE and all the baselines on the base of Hugging Face's Transformers (Wolf et al., 2020). Experiments are conducted on four Nvidia RTX 3090 GPUs.

	batch size	lr	g_1
RTE	16	1e-5	GELU
MRPC	16	2e-5	Tanh
MNLI	1024	5e-5	GELU
QID	128	1e-5	GELU
DoS	128	2e-5	ReLU
QDR	1024	3e-5	LeakyReLU

Table 5: Hyper-prameters adopted by our best BADGE models on each GLUE tasks.

	RTE		QID	
	acc-avg	acc-best	acc-avg	acc-best
	ElasticBERT as backbone			
JT	62.3	70.7	88.6	92.0
ALT	63.1	70.9	88.4	91.6
BADGE	**64.5**	72.3	**89.3**	92.1
	BERT-base as backbone			
JT	61.5	68.9	87.5	91.3
ALT	61.9	69.3	87.7	91.2
BADGE	**62.7**	70.5	**88.9**	92.5

Table 6: Experimental results of different PLM backbones.

Table 5 presents the detailed hyper-parameters for the best performing BADGE model. The hyper-parameters we present are: (a) training batch size (bsz); (b) learning rate (lr); (c) activation functions g in the bypasses. From Table 5, many tasks favor the GELU activation function.

D Ablation studies on different PLMs

In the experiments of the main content, the PLM backbone is the ALBERT-base. In Table 6, we report the results using BERT-base and ElasticBERT as backbones. The results show that our BADGE works well with different pre-trained models.