PCEE-BERT: Accelerating BERT Inference via Patient and Confident Early Exiting

Anonymous ACL submission

Abstract

BERT and other pre-trained language models (PLMs) are ubiquitous in the modern NLP. Even though PLMs are the state-of-the-art (SOTA) models for almost every NLP task (Qiu et al., 2020), the significant latency during inference forbids more widely industrial usage. In this work, we propose Patient and Confident Early Exiting BERT (PCEE-BERT), an off-the-shelf sample-dependent early exiting method that can work with different PLMs and can also work along with popular model compression methods. With a multi-exit BERT as the backbone model, PCEE-BERT will make the early exiting decision if enough numbers (patience parameter) of consecutive intermediate layers are confident about their predictions. The entropy value measures the confidence level of an intermediate layer's prediction. Experiments on the GLUE benchmark demonstrate that our method outperforms previous SOTA early exiting methods. Ablation studies show that: (a) our method performs consistently well on other PLMs, such as ALBERT and TinyBERT; (b) PCEE-BERT can make achieve different speed-up ratios by adjusting the patience parameter and the confidence threshold.

1 Introduction

Since BERT (Devlin et al., 2018), the pre-trained language models (PLMs) become the default state-of-the-art (SOTA) models for natural language processing (NLP). The 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), sequence labeling. Despite their excellent performances, there are two issues for PLMs.

First, previous studies show that PLMs such as BERT suffer from the over-thinking problem. (Zhou et al., 2020; Zhu et al., 2021) shows that in the sentence classification task, BERT's last few layers may be too deep for some samples. For a sentence classification task, if we insert a classifier on a certain intermediate layer and drop the deeper layers, these intermediate layers may outperform the last layer.

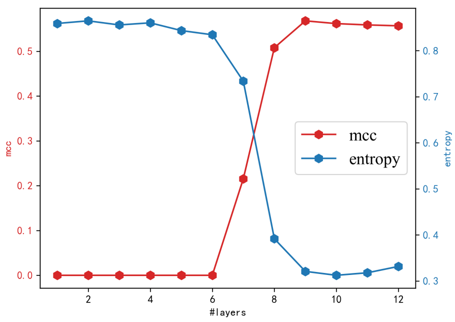
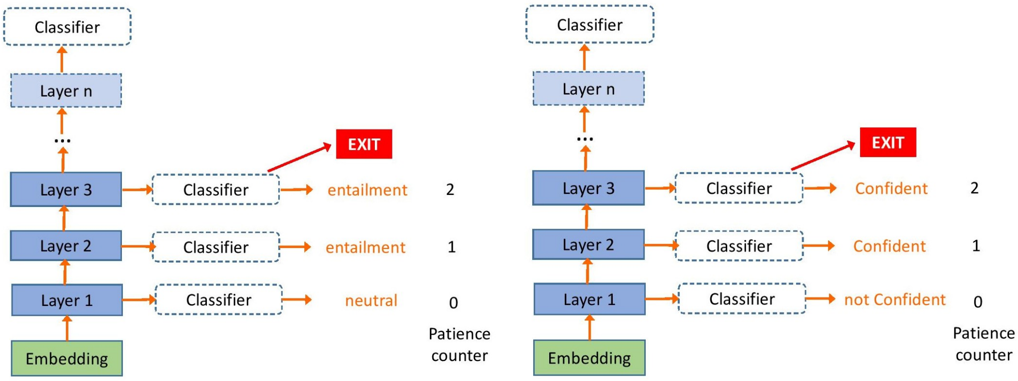


Figure 1: This figure demonstrates the overthinking problem in BERT when it is applied to the sentence classification task, such as CoLA from the GLUE benchmark.

The second drawback of PLMs is their high latency. Sentence classification (CLS) tasks play a central role in many application scenarios, such as dialogue systems, document analysis, content recommendation, etc. However, these applications are time-sensitive. For example, if a task-oriented dialogue (TOD) system takes a lot of time to respond, users will have no doubt stop using this system. User experience studies show that a response has to be made in between 0-100 ms. Thus, a CLS module should be efficient and accurate. In addition, a special feature of consumer queries is that there are times when the number of queries is extremely high. For example, during the flu season, online medical consultation will be used much often than usual. Thus, it is important for deployed models



(a) PABEE

(b) PCEE-BERT

Figure 2: Comparison between PABEE (Zhou et al., 2020) and our PCEE-BERT, a novel early exiting method that combines the score-based early exiting with the patience-based early exiting.

to adjust their latency dynamically. During peak hours, it switches to a low-latency mode to deal with more queries. And in other hours, it makes the best of itself to provide accurate answers. So how can we make model inference dynamically? The answer is adaptive inference.

There exists a branch of literature focusing on making PLMs' inference more efficient via network pruning (Zhu and Gupta, 2018; Xu et al., 2020; Fan et al., 2020; Michel et al., 2019), knowledge distillation (Sun et al., 2019; Sanh et al., 2019; Jiao et al., 2020a), weight quantization (Zhang et al., 2020; Bai et al., 2020; Kim et al., 2021) and adaptive inference (Zhou et al., 2020; Xin et al., 2020; Liu et al., 2020). The adaptive inference has drawn much attention. The idea of adaptive inference is to deal with simple examples with only shallow layers of BERT and process more difficult queries with deeper layers, thus significantly speeding up the inference time on average while maintaining high accuracy. The speed-up ratio can be easily controlled with certain hyper-parameters to process significant changes in query traffic without re-deploying the model services or maintaining a group of models.

Early exiting is one of the most important adaptive inference methods (Bolukbasi et al., 2017). As depicted in Figure 2(b), it implements adaptive inference by installing an early exit, i.e., an intermediate prediction layer, at each layer of BERT and early exiting "easy" samples to speed up inference. At the training stage, all the exits are jointly optimized with BERT's parameters. At the inference stage, there are two different settings. First, in budgeted exiting mode, the model makes a prediction with a fixed exit for all queries. This mode deals with heavy traffic by assigning a shallower exit for prediction. The other one is dynamic exiting mode. That is, some strategies for early exiting are 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.

There are mainly three early exiting strategies for BERT dynamic exiting. The first one is score-based early exiting. BranchyNet (Teerapittayanon et al., 2016), FastBERT (Liu et al., 2020), and DeeBERT (Xin et al., 2020) calculated the entropy of the prediction probability distribution as an estimation for the confidence of exiting classifiers to enable dynamic early exiting. Shallow-Deep Nets (Kaya et al., 2019) and RightTool (Schwartz et al., 2020a) leveraged the maximum of the predicted distribution as the exiting signal. The second type is the learned exiting (Elbayad et al., 2020). In this type of work, an early exiting signal is generated by a learnable module in the neural network. For example, BERxiT (Xin et al., 2021) install a fully connected layer right after each transformer block of BERT to output a score that is used to decide whether the BERT should stop inference and exit early. The third type is patience-based early exiting, which relies on cross-layer comparison to formulate the exiting signal. PABEE (Zhou et al., 2020) propose a dynamic exiting strategy analogous to early stopping model training. That is, if the exits' predictions remain unchanged for a pre-defined number of times (patience), the model will stop inference and exit early. PABEE achieves SOTAs results for BERT early exiting.

Despite its state-of-the-art performances during early exiting, PABEE is inflexible in adjusting the speedup ratios. On a given task, once the multi-exit BERT is fine-tuned and the patience parameter is fixed, PABEE can only achieve a fixed average speedup ratio. Thus, PABEE can not achieve speedup ratios of certain values. This drawback makes PABEE inconvenient to use in real industrial scenarios. Thus, it is of great importance to come up with a method that can flexibly adjust its speedup ratios and performs comparable to or better than PABEE.

In this work, we propose Patiently Confidently Early Exiting BERT (PCEE-BERT), a novel early exiting method that combines the advantage of score-based methods and the patience based early exiting method. A multi-exit BERT is adopted as the backbone model, and an intermediate classifier (i.e., an exit) is installed right after each transformer black. PCEE-BERT will early exit if there are enough numbers (i.e., the patience parameter) of consecutive exits being confident for their predicted distributions. We mainly use entropy as the confidence measure. Intuitively, our method requires patience and confidence. It will not rush into an early exiting if we only see a couple of intermediate layers being confident. In addition, it allows the next layer to modify the predictions. In this way, our PCEE-BERT can exit with higher accuracy while maintaining flexibility.

Extensive experiments are conducted on the GLUE benchmark (Wang et al., 2018). The results show that our method outperforms the previous SOTA early exiting methods, especially in cases where the speedup ratio is large. In addition, one can adjust the patience and confidence threshold so that PCEE-BERT can arrive at different speedup ratios. A series of ablation studies are conducted, resulting in the following observations: (a) PCEE-BERT can work with different confidence measures; (b) our method performs consistently well on different PLMs, and can work alongside model compression methods to further speed up the BERT's inference; (c) our PCEE-BERT can also be applied to computer vision tasks.

The rest of the paper is organized as follows. First, we introduce the preliminaries for multi-exit BERT and early exiting. Second, we elaborate on our PCEE-BERT method. Third, we conduct experiments on the GLUE benchmark and conduct a series of ablations studies. Finally, we conclude with possible future works.

2 Preliminaries

In this section, we introduce the necessary background for BERT early exiting. Throughout this work, we consider the case of multi-class classification with samples $\{(x, y), x \in \mathcal{X}, y \in \mathcal{Y}, i = 1, 2, ..., N\}$, e.g., sentences, and the number of classes is K.

2.1 Backbone models

In this work, we adopt BERT as the backbone model. BERT is a multi-layer Transformer (Vaswani et al., 2017) network, which is pre-trained in a self-supervised manner on a large corpus. The number of transformer layers of our backbone is denoted as M, and the hidden dimension is d.

2.2 Early-exiting Architecture

As depicted in Figure 2, early exiting architectures are networks with exits at each transformer layer. With M exits, M classifiers $f^{(m)}(x; \theta^{(m)}) : \mathcal{X} \rightarrow \Delta^K$ $(m = 1, 2, ..., M)$ are designated at M layers of BERT, each of which maps its input to $p^{(m)}(x; \theta^{(m)})$, a probability distribution over the K classes. All the parameters of the transformer layers and exits are denoted as Θ.

2.2.1 Training

At the training stage, all the exits are jointly optimized with a summed loss function. Following Huang et al. (2017) and Zhou et al. (2020), the loss function is the weighted average of the cross-entropy (CE) losses given by

$$\mathcal{L} = \frac{\sum_{m=1}^{M} m * \mathcal{L}^{(m)}}{\sum_{m=1}^{M} m}, \tag{1}$$

where $\mathcal{L}^{(m)} = \mathcal{CE}(y, p^{(m)}(x; \theta^{(m)}))$ denotes the cross-entropy loss of the m-th exit. Note that the weight m corresponds to the relative inference cost of exit m.

2.2.2 Inference

During inference, the multi-exit BERT can exit early in two different modes, depending on whether the computational budget to classify an example is known or not.

Budgeted Exiting. If the computational budget is known, we can directly appoint a suitable exit m^* of BERT, $f^{(m^*)}(x; \theta^{(m^*)})$, to predict all queries.

	RTE	QNLI	MRPC
patience=1	3.24	2.25	2.00
patience=2	4.96	3.87	3.00
patience=3	6.69	5.32	4.18
patience=4	7.77	6.50	5.60
patience=5	8.78	7.61	6.81
patience=6	9.75	8.64	7.91
patience=7	10.68	9.54	8.83
patience=8	11.47	10.36	9.72
patience=9	11.79	11.04	10.51
patience=10	11.92	11.57	11.26
patience=11	12.00	12.00	12.00

Table 1: Average inference layers of PABEE on the RTE, QNLI and MRPC tasks.

Dynamic Exiting. Under this mode, after receiving a query input x, the model starts to predict on the classifiers $f^{(1)}(x;\theta^{(1)})$, $f^{(2)}(x;\theta^{(2)})$, ..., in turn in a forward pass, reusing computation where possible. It will continue to do so until it receives a signal to stop early at an exit $m^* < M$, or arrives at the last exit M. At this point, it will output the final predictions based on the current and previous predictions. Note that under this early exit setting, different samples might exit at different layers.

3 PCEE-BERT

3.1 Motivation

PABEE achieves the SOTA performances for BERT early exiting by applying an early exiting decision-making process that mimics the early stopping of model training. However, one drawback of PABEE is that it can not flexibly adjust the average inference layers (i.e., speed-ups) for a given dataset once its patience parameter is set. Table 1 shows PABEE can not achieve certain values for average inference layers, such as around 4.0, 6.0, or 9.0 on RTE. This drawback may limit the industrial usage of early exiting techniques. Thus, it is of great importance to develop a new method that performs comparably with PABEE and is more flexible than PABEE.

3.2 PCEE-BERT: a novel dynamic exiting method

The inference process of PCEE-BERT is illustrated in Figure 2(b). Assume the feed forward process for predicting sample x has gone through layers 1, ..., $m-1$, and we are now at layer m. After going through the transformer layer m, the intermediate classifier $f^{(m)}(x;\theta^{(m)})$ predicts a class label distribution $p^{(m)}(x;\theta^{(m)})$. The confidence level of layer m is measured by the entropy value of distribution $p^{(m)}(x;\theta^{(m)})$:

$$C^{(m)} = \frac{\sum_{k=1}^{K} p_k^{(m)} \log p_k^{(m)}}{\log(1/K)}, \qquad (2)$$

where $p_k^{(m)}$ is the probability mass for k-th class label. If $C^{(m)}$ is smaller than a pre-defined threshold τ, the predictions of layer m is considered confident. Otherwise, it is considered in-confident.

We use a patience counter pct to store the number of times that the predictions remain confident in consecutive layers. Formally, at layer m, $pct^{(m)}$ is calculated as

$$pct^{(m)} = \begin{cases} pct^{(m-1)} + 1, & \text{if } C^{(m)} < \tau, \\ 0, & \text{otherwise.} \end{cases} \qquad (3)$$

We stop inference early at layer m when $pct^{(m)}$ reaches a predefined integer number t (the patience parameter). If this condition is never fulfilled, we use the final classifier M for prediction. In this way, the model can make an early exit without passing through all layers to make a prediction.

Our method draws advantages from the previous score-based early exiting method (Teerapittayanon et al., 2016) and patience-based method (Zhou et al., 2020) and overcomes their shortcomings. First, the score-based early exiting method relies on the confidence score from only the current layer. However, as revealed by Szegedy et al. (2014); Jiang et al. (2018), prediction of probability distributions (i.e., softmax scores) suffers from being over-confident to one class, making it an unreliable metric to represent confidence. In our method, early exiting occurs when a group of consecutive layers is confident, thus making the early exiting decision more reliable. Second, with a patience-based early exiting method like PABEE, when a deeper layer tries to correct the predictions, the patience count resets to zero. As a result, PABEE is less efficient than our PCEE-BERT. Third, since our method is a combination of PABEE and the score-based method, one can conveniently adjust the threshold and patience parameters to control the speed-up ratios, which makes our method more flexible than PABEE.

4 Experiments

4.1 Datasets

We evaluate our proposed approach to the classification tasks on the GLUE benchmark. We only exclude the STS-B task since it is a regression task, and we exclude the WNLI task following previous work (Devlin et al., 2018; Xu et al., 2020).

4.2 Baselines

We compare our approaches with three groups of baselines.

Backbone models: We mainly choose the BERT-base model open-sourced by Devlin et al. (2019) as the backbone model. We also investigate whether our method is applicable across different backbones, so we also run ablation experiments with ALBERT base (Lan et al., 2020) and TinyBERT_6 (Jiao et al., 2020b).

Budgeted exiting: In the section 2.2 we have introduced how to train a multi-exit BERT. Once the multi-exit BERT, we can conduct budgeted early exiting, that is, asking a designated intermediate layer to encode and predict all the samples. Budgeted exiting is a direct way to speed up BERT's inference, but it is instance adaptive. Some of the samples may not need to go through many of the BERT's layers, and the others may be more difficult and require deeper feature encoding from the deeper layers of BERT.

Dynamic exiting: In this part, we compare our methods with a series of strong baselines, including BranchyNet (Teerapittayanon et al., 2016), Shallow-Deep (Kaya et al., 2019), BERxiT (Xin et al., 2021), and PABEE (Zhou et al., 2020). Note that PABEE can not flexibly adjust the average inference layers on a task once the patience parameter is set. So we will adjust the thresholds in the other baselines and our PCEE-BERT so that all methods' number of average inference layers are close.

4.3 Evaluation of early exiting method

In this work, we strictly follow the GLUE benchmark to report the performances metrics on each task. Note that this work focuses on investigating the early exiting of PLMs. Thus we have to consider the trade-offs between performance and efficiency. Following PABEE (Zhou et al., 2020), we mainly report the speedup ratio as the efficiency metric. Assume the PLM backbone has N layers in total. For each test sample x_i ($i \in \{0, 1, ..., N\}$), the early exiting layer is m_i, then the average speedup ratio on the test set is calculated by

$$\textbf{Speedup} = 1 - \frac{\sum_1^N m_i}{\sum_1^N M}. \quad (4)$$

We choose this efficiency metric for the following reason: (1) it is linear w.r.t. the actual amount of computation; (2) according to our experiments, it is proportional to actual wall-clock runtime and is also more stable across different runs compared with actual runtime due to randomness by other processes on the same machine.

4.4 Experimental settings

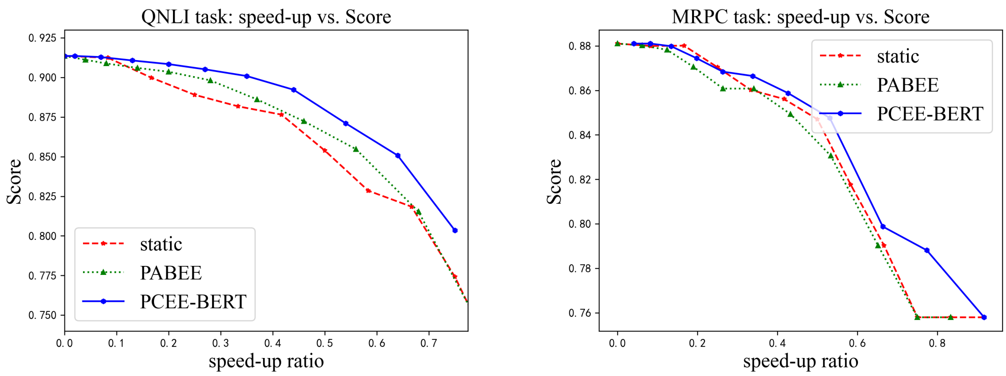
Training We add a linear output layer after each intermediate layer of the pre-trained BERT or other backbone models as the internal classifiers. We perform grid search over batch sizes of 16, 32, 128, and learning rates of 1e-5, 2e-5, 3e-5, 5e-5 with an Adam optimizer. The hyper-parameters are selected via the 5-fold cross validation on the train set of GLUE tasks. We implement PCEE-BERT on the base of Hugging Face's Transformers (Wolf et al., 2020). Experiments are conducted on a single Nvidia V100 16GB GPU.

Inference 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. This is a common scenario in the industry where individual requests from different users (Schwartz et al., 2020b) come at different time points. We report the median performance over five runs with different random seeds.

4.5 Main results

In Table 2, we report the performance comparisons of each method on the GLUE benchmark under three different speedup settings. The three speedup settings are: (1) 74% to 82% speedup; (2) 46% to 54% speedup; (3) 23% to 28% speedup. Since PABEE can not flexibly adjust the speedup ratios for a given patience parameter and a given task, we adjust the hyper-parameters (such as entropy threshold) of our PCEE-BERT and the other baselines to achieve similar speedups with PABEE. The results in table 2 clearly show that our PCEE-BERT method outperforms the baseline methods under different speedup ratios. Table 2 also shows that the PABEE method is the best performing baseline. Thus, in order to further analyze and better visualize the results, we draw the score-speedup curves (in Figure 3) for budgeted early exiting, PABEE and PCEE-BERT, on the QNLI and MRPC tasks. [1] With Table 2 and Figure 3, we can make the

[1] The score-speedup curves for the other five GLUE tasks can be found in the appendix.



(a) QNLI task

(b) MRPC task

Figure 3: Performance–efficiency trade-offs using different exiting strategies. We can see that our PCEE-BERT consistently outperforms the strong baseline, PABEE, especially when the speed-up ratio is large.

	CoLA		MNLI		MRPC		QNLI		QQP		RTE		SST-2	
	score	speedup	score	speedup	score	speedup	score	speedup	score	speedup	score	speedup	score	speedup
BERT base	0.5426	0%	0.8312	0%	0.8687	0%	0.8988	0%	0.8923	0%	0.6917	0%	0.9135	0%
Budgeted-Exiting-3L	0.0	75%	0.7004	75%	0.7580	75%	0.7741	75%	0.8181	75%	0.5470	75%	0.8107	75%
Budgeted-Exiting-6L	0.0	50%	0.7968	50%	0.8470	50%	0.8538	50%	0.8934	50%	0.6814	50%	0.8864	50%
Budgeted-Exiting-9L	0.5195	25%	0.8306	25%	0.8704	25%	0.8840	25%	0.9029	25%	0.6898	25%	0.9116	25%
BranchyNet	0.0,	74%	0.6381	76%	0.7568	76%	0.7416	80%	0.7159	80%	0.5465	76%	0.7986	76%
	0.0	51%	0.7827	53%	0.8298	52%	0.8711	47%	0.8927	50%	0.6738	47%	0.8831	49%
	0.5213	27%	0.8297	25%	0.8579	24%	0.8926	27%	0.9005	26%	0.6795	26%	0.9124	24%
Shallow-Deep	0.0,	75%	0.6406	77%	0.7557	76%	0.7432	78%	0.7143	79%	0.5471	76%	0.7947	77%
	0.0	52%	0.7818	51%	0.8279	51%	0.8715	49%	0.8956	51%	0.6721	48%	0.8843	48%
	0.5232	26%	0.8288	26%	0.8568	25%	0.8931	26%	0.9012	27%	0.6778	26%	0.9115	25%
BERxiT	0.0,	76%	0.6354	76%	0.7562	76%	0.7331	78%	0.6828	80%	0.5531	77%	0.7953	76%
	0.1232	52%	0.7842	51%	0.8298	51%	0.8705	48%	0.8914	49%	0.6731	47%	0.8829	49%
	0.5218	25%	0.8321	26%	0.8617	26%	0.8958	27%	0.9012	26%	0.6812	27%	0.9138	24%
PABEE	0.0,	75%	0.6392	77%	0.7580	75%	0.7355	81%	0.6863	82%	0.5579	75%	0.7993	77%
	0.0	50%	0.7885	52%	0.8306	53%	0.8723	46%	0.8956	49%	0.6770	46%	0.8876	48%
	0.5241	26%	0.8342	24%	0.8608	26%	0.8981	28%	0.9043	24%	0.6834	28%	0.9174	22%
PCEE-BERT (ours)	0.0975	79%	0.7336	72%	0.7881	77%	0.8034	75%	0.7961	82%	0.5840	76%	0.8360	76%
	0.2323	57%	0.7999	53%	0.8476	53%	0.8710	54%	0.9084	49%	0.6942	47%	0.9036	48%
	0.5283	27%	0.8335	28%	0.8684	26%	0.9051	27%	0.9118	25%	0.6970	30%	0.9186	23%

Table 2: Experimental results of different early exiting methods with the same fine-tuned BERT backbone on the GLUE benchmark. The results show that PCEE-BERT is effective in accelerating BERT's inference with less performance loss compared with the baseline methods.

following observations:

- Although it is clear that PABEE performs better than the other baselines when the speedup ratio is around 50% or 25%, its advantages over the other baselines with the 75% speedup ratio is relatively small. With the 75% speedup ratio for seven GLUE tasks, it performs better than the score-based methods only on three tasks. This observation motivates us to improve PABEE by combining its patience-based early exiting mechanism with the score-based ones.

- Our PCEE-BERT consistently performs better than the baseline methods, especially when the speedup ratio is large. Note that our PCEE-BERT also consistently outperforms the budgeted exiting speedup ratios, which the other baselines do not achieve. Figure 3(a) and 3(b) show that score-speedup curve for PABEE is interleaving with that of the budgeted exiting. However, the score-speedup curve for PCEE-BERT distances itself from the others for most of the GLUE tasks.

- The overthinking problem is prevailing in the GLUE benchmark, and our PCEE-BERT early exiting can effectively take advantage of this phenomenon. For 6 of the GLUE tasks, PCEE-BERT can outperform BERT-base with a 25% (or more than) speedup ratio. And for 2 of the GLUE tasks, PCEE-BERT can outperform BERT-base with a 50% (or more than) speedup ratio.

Putting performance comparisons aside, one benefit of PCEE-BERT is that it is flexible since by adjusting the threshold and the patience parameter, it can easily control the average inference layers and cover (or achieve values close to) any speedup ratios.[2]

4.6 Ablation studies

4.6.1 Ablation on the confidence measures

Note that our PCEE-BERT is a novel combination of PABEE and BranchyNet. Thus PCEE-BERT mainly uses the entropy of predicted distributions as the confidence measure of an intermediate layer. However, can PCEE-BERT work with the other confidence measures, such as Shallow-Deep? We switch the entropy-based confidence level $C^{(M)}$ (Equation 2) with that from Shallow-Deep (Kaya et al., 2019):

$$C^{(M)} = \text{Argmax}_k p_k^{(m)}, \tag{5}$$

and we will call this version of PCEE-BERT as PCEE-BERT-v1. Note that PCEE-BERT-v1 does not require a newly fine-tuned model.

With BERxiT, we can come up with PCEE-BERT-v2. Following BERxiT, PCEE-BERT-v2 fine-tunes the multi-exit BERT with a fully connected layer right after each transformer block designated to evaluate the confidence score $C^{(M)}$ for early exiting at that layer. $C^{(M)}$ is learned along with the training of intermediate classifiers. Note that PCEE-BERT-v2 can not reuse the fine-tuned checkpoints used in PCEE-BERT and requires one to fine-tune the BERT backbones on the task at hand.

We conduct the experiments on the QNLI tasks, and the results are reported in Figure 4. We can see that PCEE-BERT-v1 and PCEE-BERT-v2 perform comparably to PCEE-BERT. The results show that the proposed PCEE-BERT early exiting mechanism is off-the-shelf, and the reason for the success of our PCEE-BERT is its early exiting mechanism, that is, early exit if a group of consecutive exits is confident for their predictions.

4.6.2 Ablation of PLM backbones

In the main experiments, we use BERT as the pre-trained backbone model. However, PCEE-BERT can also work with the other types of pre-trained backbones, such as ALBERT base (Lan et al., 2020) and TinyBERT$_6$ (Jiao et al., 2020b). We conduct the experiments on the QNLI task with these two

[2] See the Appendix for demonstration on MRPC.

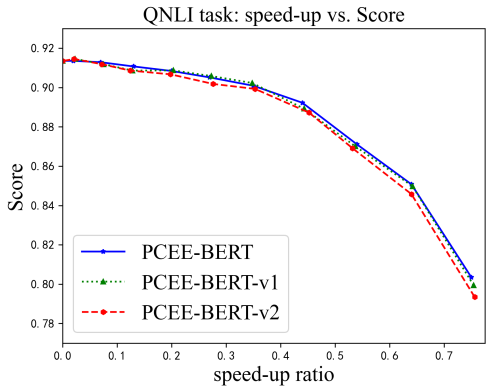


Figure 4: This figure demonstrates that PCEE-BERT can work with other confidence measures.

backbone models, and results are presented in Figure 5(a) and 5(b). We can see that when using the other pre-trained backbones, PCEE-BERT also performs better than the baseline methods.

The results for PCEE-BERT on the TinyBERT also convey an important message: as an inference speedup method, our PCEE method can work alongside the model compression methods to further reduce the latency of BERT.

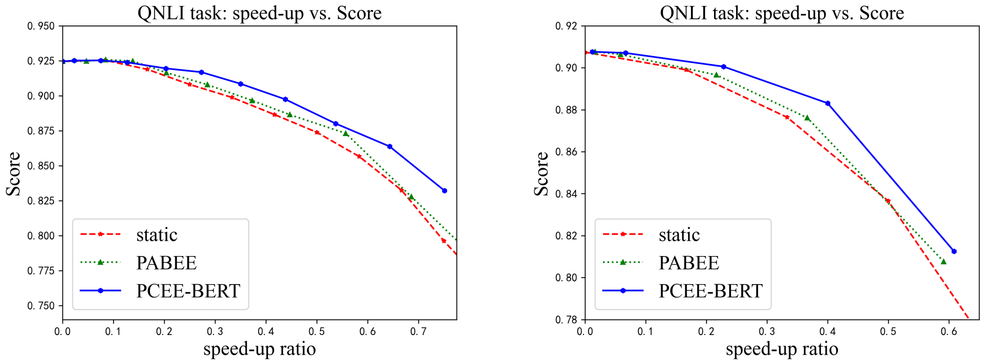
4.6.3 Ablation of cross-layer ensemble

Since we have a prediction module at each layer of BERT, we can conduct model ensemble across layers that the forward pass has gone through already. In Figure 6, we conduct the ablation studies on the RTE and QNLI tasks. According to Figure 6, cross-layer ensemble leads to performance degradation when the speedup ratio is large, while when the average inference layers is close to the number of BERT's transformer blocks M, cross-layer ensemble results in slight improvements. In conclusion, the cross-layer ensemble does not result in consistent performance improvements.

A possible application of the above results is to apply the cross-layer ensemble when a low speedup ratio is applied. And when we ask the model to exit early in the shallow layers, the cross-layer ensemble is not used.

4.6.4 PCEE-BERT are effective for image classification

Our main experiments are conducted on BERT, a pre-trained language model, and the GLUE benchmark, a series of natural language understanding tasks. However, our PCEE-BERT method is a plug-and-play early exiting and can be applied to models and tasks of different modalities. To demon-



(a) ALBERT base as backbone

(b) TinyBERT$_6$ as backbone

Figure 5: Ablation study on alternative PLMs.

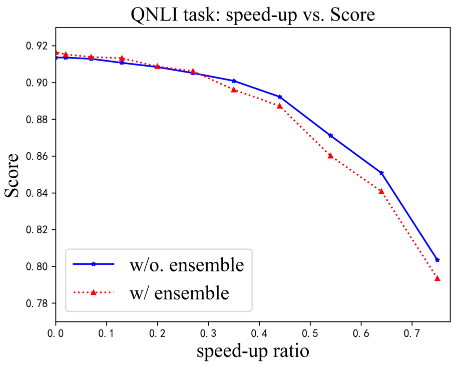


Figure 6: Results for ablation study of whether PCEE-BERT should apply the cross-layer ensemble.

Method	CIFAR-10		CIFAR-100	
	speed-up	Acc.	speed-up	Acc.
ResNet-56	0.0	91.8	0.0	68.6
PABEE	77%	78.3	76%	51.2
	52%	86.7	48%	62.5
	26%	91.9	24%	69.2
PCEE-BERT	76%	81.2	74%	55.6
	51%	87.3	49%	64.8
	25%	92.1	24%	69.4

Table 3: Experimental results of PCEE-BERT when applied in the image classification tasks.

strate the effectiveness of PCEE-BERT on the image classification task, we follow the experimental settings in PABEE (Zhou et al., 2020). We conduct experiments on two image classification datasets, CIFAR-10 and CIFAR-100 (Krizhevsky, 2009). The ResNet-56 model (He et al., 2016) serves as the backbone, and we compare PCEE-BERT with PABEE. We place an exiting classifier at every two convolutional layers. We set the batch size to 128 and use an SGD optimizer with a learning rate of 0.1.

Table 3 reports the results. PCEE-BERT outperforms PABEE when early exiting at different speedup ratios. In addition, the performance advantages of PCEE-BERT are larger when the speedup ratio is large, which is also observed in the NLP tasks. And PCEE-BERT outperforms the original ResNet-56 on both tasks even when it provides around 25% speedup.

5 Conclusion

In this work, we propose PCEE-BERT, a novel efficient inference method that can yield a better performance-speed trade-off than the existing early exiting methods. PCEE-BERT adopts BERT as the backbone model and makes the exiting decision if there are enough intermediate layers to make confident predictions. The confidence level is measured by the entropy of the predicted distributions. Experiments on the GLUE benchmark demonstrate that our method outperforms the previous SOTA early exiting methods, especially when the speedup ratio is large. In addition, PCEE-BERT can achieve different speedup ratios by adjusting the patience parameter and the confidence threshold, which makes it more flexible in industrial usage. Ablation studies show that: (a) our PCEE-BERT can adopt different confidence measures, such as maximum probability mass; (b) our method performs consistently well on different PLMs and can work together with model compression methods to speed up the BERT's inference; (c) our PCEE-BERT also performs well on computer vision tasks.

References

Haoli Bai, Wei Zhang, L. Hou, L. Shang, Jing Jin, X. Jiang, Qun Liu, Michael R. Lyu, and Irwin King. 2020. Binarybert: Pushing the limit of bert quantization. *ArXiv*, abs/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. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. 2020. Depth-adaptive transformer. *ArXiv*, abs/1910.10073.

Angela Fan, E. Grave, and Armand Joulin. 2020. Reducing transformer depth on demand with structured dropout. *ArXiv*, abs/1909.11556.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Gao Huang, Danlu Chen, T. Li, Felix Wu, L. V. D. Maaten, and Kilian Q. Weinberger. 2017. Multi-scale dense convolutional networks for efficient prediction. *ArXiv*, abs/1703.09844.

Heinrich Jiang, Been Kim, and Maya R. Gupta. 2018. To trust or not to trust a classifier. In *NeurIPS*.

Xiaoqi Jiao, Y. Yin, L. Shang, Xin Jiang, X. Chen, Linlin Li, F. Wang, and Qun Liu. 2020a. Tinybert: Distilling bert for natural language understanding. *ArXiv*, abs/1909.10351.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020b. 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.

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

Se-Hoon Kim, Amir Gholami, Zhewei Yao, M. W. Mahoney, and K. Keutzer. 2021. I-bert: Integer-only bert quantization. *ArXiv*, abs/2101.01321.

A. Krizhevsky. 2009. Learning multiple layers of features from tiny images.

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.

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.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *NeurIPS*.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *ArXiv*, abs/2003.08271.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

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

Roy Schwartz, Gabi Stanovsky, Swabha Swayamdipta, Jesse Dodge, and N. A. Smith. 2020a. The right tool for the job: Matching model and instance complexities. In *ACL*.

Roy Schwartz, Gabriel Stanovsky, Swabha Swayamdipta, Jesse Dodge, and Noah A. Smith. 2020b. 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.

S. Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for bert model compression. In *EMNLP/IJCNLP*.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, D. Erhan, Ian J. Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. *CoRR*, abs/1312.6199.

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.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, L. Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *ArXiv*, abs/1706.03762.

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. In *BlackboxNLP@EMNLP*.

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.

J. Xin, Raphael Tang, J. Lee, Y. Yu, and Jimmy Lin. 2020. Deebert: Dynamic early exiting for accelerating bert inference. *ArXiv*, abs/2004.12993.

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.

Canwen Xu, Wangchunshu Zhou, Tao Ge, Furu Wei, and M. Zhou. 2020. Bert-of-theseus: Compressing bert by progressive module replacing. In *EMNLP*.

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

W. Zhang, L. Hou, Y. Yin, L. Shang, X. Chen, X. Jiang, and Qun Liu. 2020. Ternarybert: Distillation-aware ultra-low bit bert. *ArXiv*, abs/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.

M. Zhu and S. Gupta. 2018. To prune, or not to prune: exploring the efficacy of pruning for model compression. *ArXiv*, abs/1710.01878.

Wei Zhu, Xiaoling Wang, Yuan Ni, and Guotong Xie. 2021. GAML-BERT: Improving BERT early exiting by gradient aligned mutual learning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3033–3044, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

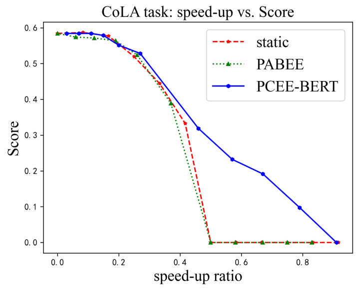
A Appendix

A.1 Quality–efficiency trade-offs on GLUE benchmark tasks.

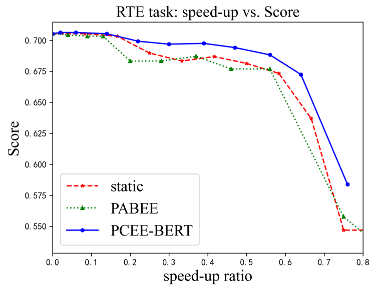
In the main content, we present the quality–efficiency trade-offs curves for 2 GLUE tasks. And here we put the results of the other five tasks in Figure 7.

A.2 Demonstrating PCEE-BERT can cover (or achieve values close to) any speedup ratios

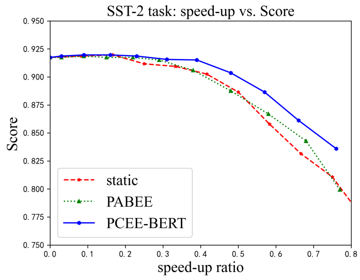
PCEE-BERT's speedup ratio can be conveniently adjusted by setting different values for the patience parameter and the confidence threshold. To validate our claim, we alternate the threshold among 100 points between 0.0 to 1.0 when the patience parameter takes the value of 1, 2, 3, 6. The average numbers of inference layers are reported in the scatter plot (Figure 8). We can see that by adjusting the threshold and the patience parameter, one can easily control the average inference layers and cover (or achieve values close to) any speedup ratios.



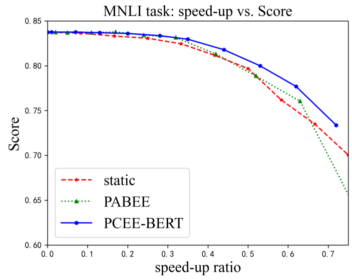
(a) CoLA task



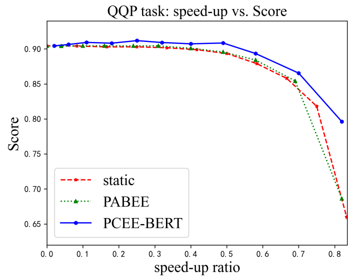
(b) RTE task



(c) SST-2 task



(d) MNLI task



(e) QQP task

Figure 7: Performance–efficiency trade-offs using different exiting strategies. We can see that our PCEE-BERT consistently outperforms the strong baseline, PABEE, especially when the speed-up ratio is large.

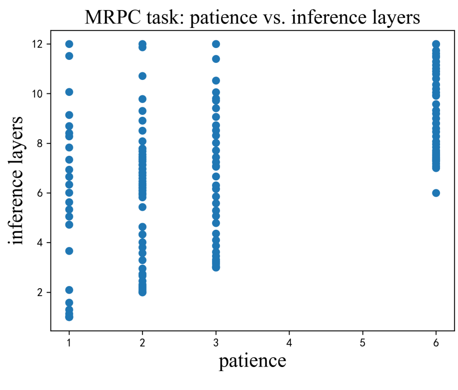


Figure 8: This figure demonstrates that PCEE-BERT can cover (or achieve values close to) any speedup ratios.