F-PABEE: Flexible-patience-based Early Exit for Single-label and Multi-label text Classification Tasks

Abstract

In recent years, pre-training language models (PLMs) such as BERT and ALBERT have achieved high accuracy in many single-label and multi-label classification tasks. However, their industrial usages are limited by their large inference latency. In order to increase the inference speed of BERT while maintaining good performances, this paper proposes a Flexible-patience-based early exiting (F-PABEE) method. F-PABEE makes predictions at each layer's intermediate classifier and will exit early if the current layer and the last few layers have similar (similarity score less than a threshold) predicted distributions. Our method is more flexible than the previous state-of-the-art early exiting method PABEE since it can achieve different speed-performance tradeoffs by adjusting the similarity score threshold and the patience parameter. We conduct a wide range of experiments on single-label classification tasks from the GLUE benchmark and four multi-label classification tasks and find that: (a) F-PABEE outperforms existing early exit models such as Brexit and PABEE on both single-label and multi-label tasks; (b) F-PABEE performs well on different PLMs such as ALBERT; (c) ablation studies show that as a similarity measure, Jenson-Shannon divergence works best for our F-PABEE method.[1]

1 Introduction

In recent years, fine-tuning pre-trained language models (PLMs) have become the standard solution for almost all Natural Language Processing tasks (Lin et al., 2021).Despite their state-of-the-art (SOTA) performances, BERT (Devlin et al., 2018) and its variants (Lan et al., 2019; Yang et al., 2019; Radford et al., 2019; Liu et al., 2019) still face a significant challenge in applications: high latency due to their large parameter numbers and model depth. Generally, pre-training models are mainly based on dozens of stacked transformers, and higher-accuracy models usually rely on deeper transformer architectures, which may require higher computational resources, capital, and time costs. In order to speed up the inference of PLMs, a series of methods have been proposed, such as attention head pruning, layer pruning, knowledge distillation, and input-adaptive inference methods. Early exiting, as an essential input-adaptive inference method, has attracted much attention.

Early exiting installs a classifier (or exit) at each transformer layer of BERT and evaluates at each layer whether the current layer's prediction is suited to be the final prediction. Thus, it can process different samples with different complexity and, as a result, strike a balance between model performance and speedup ratios. The core of early exiting is to design a strategy to determine the quality of the current intermediate layer's prediction. There are mainly two types of early exiting strategies. The first one is to evaluate the current layer's prediction based on specific confidence measurements, such as entropy BranchyNet (Teerapittayanon et al., 2016b) maximum probability mass Shallow-Deep (Kaya and Dumitras, 2018). Recently, BERxiT (Xin et al., 2021a) asks the model to learn to evaluate the confidence of intermediate layers' predictions. The second type of early exiting strategies is represented by PABEE (Zhou et al., 2020b). PABEE evaluates the quality of the prediction by comparing it with the previous layers', mimicking the idea of model early stopping.

We raise two issues for the current early exiting literature. Despite its SOTA performances, PABEE still faces a limitation for application: it can not flexibly adjust the speedup ratio on a given task and a fixed patience parameter, mainly caused by PABEE's strict cross-layer comparison strategy. Thus, we wonder whether we can combine PABEE with a softer cross-layer comparison strategy. Second, the current early exiting literature

[1]Code will be made publicly available upon acceptance.

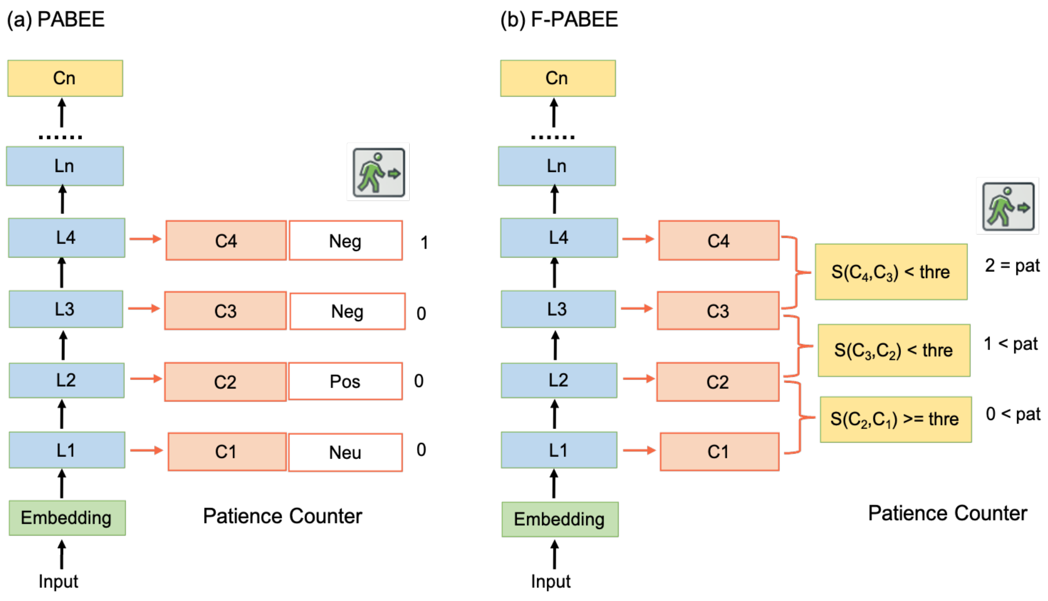

Figure 1: Architecture of PABEE and F-PABEE, C_i is classifier layer,S is score between adjacent layers, $thre$ is threshold, pat is patience in the model.

mainly focuses on the single-label classification (SLC) tasks, and multi-label classification (MLC) tasks are neglected. So can the current SOTA methods like BERxiT and PABEE speed up MLC tasks with good performances?

This paper proposes a Flexible-patience-based early exiting (F-PABEE) method to address the above issues. F-PABEE makes predictions at each intermediate layer's exit and will exit early if the current layer and the last few layers have similar (similarity score less than a threshold) predicted distributions. The similarity of predicted probability distributions from the current and previous layers are measured by divergence-based metrics like Kullback-Leibler Divergence (KL-divergence) and Jensen–Shannon divergence (JSD-dvergence). Our method can be seen as a natural extension of the PABEE method and is more flexible than PABEE since it can achieve different speed-performance tradeoffs by adjusting the similarity score threshold and the patience parameter. Our novel early exiting strategy can effortlessly extend to MLC tasks. An MLC task is usually regarded as a collection of binary classification tasks; thus, comparing the predictions across layers will summate similarity scores for binary predicted distributions.

We conduct extensive experiments on SLC tasks from the GLUE benchmark and four MLC tasks: (Yang et al., 2018; Hemphill et al., 1990; Coucke et al., 2018). The experimental results show that our F-PABEE method outperforms the existing SOTA early exiting methods on both SLC and MLC tasks. Ablation studies show that: (a) F-PABEE performs well on PLM backbones other than BERT; (c) as a similarity measure, Jenson-Shannon divergence works best for our F-PABEE method.

Our contributions are three-fold: (1) We propose F-PABEE, a novel and effective inference mechanism which is also flexible in adjusting the speedup ratios of BERT; (2) Our empirical results on the GLUE and four benchmark MLC tasks show that our method can effectively speed up inference across different tasks while maintaining good performances; (3) As far as we know, we are the first to investigate the early exiting of MLC tasks, and our F-PABEE is suitable for this type of task.

2 Related Works

Existing BERT inference speedup methods mainly include two categories: (1) Static approaches that compress the heavy models into a fixed smaller one; (2) Adaptive inference approaches that make no changes to the architecture and encode different samples with different parts of the model. Usually they can adjust certain hyper-parameters to dynamic controlling the average latency in product environments.

2.1 Static inference approach

Since the rise of BERT, there are a quite large numbers of literature devoting themselves in speeding up the inference of BERT. HeadPrune (Michel et al., 2019) propose to rank the attention heads and progressively pruning them, so that the inference latency is reduced. LayerDrop (Fan et al., 2019) further pre-trains and fine-tunes BERT with some of the layers skipped, so that we can reduce the layers on demand. DynaBERT (Hou et al., 2020) conduct pruning on attention heads, neurons in the feed-forward layers and layer depth. Knowledge distillation (KD) plays an important role in BERT model compression. (Sun et al., 2019) investigate the best practices of distilling knowledge from BERT into smaller-sized models. (Sanh et al., 2019) initialize the student model weights from the BERT teacher. (Jiao et al., 2019) applies attention KD, feature KD along with soft-label based KD, and introduce KD in the pre-training of student models. Based on lightweight integer-only approximation methods for nonlinear operations, I-BERT (Kim et al., 2021) performs an end-to-end integer-only BERT inference without any floating point calculation.

Note that despite some static model compression methods along can already speedup the BERT's inferences, they are still in the form of deep neural networks with multiple stacked layers. Thus, other inference speedup methods, like sample adaptive inference, can also be applied to further improve the efficiency.

2.2 Early exiting approaches

Recently, adaptive inference has attracted the attention of both academia and industries. One of the major methodology for adaptive inference is early exiting. It is a method to dynamically adjust certain hyper-parameters in response to changes in request traffic. Shallow transformer layers are used to infer simple requests, and deep transformer layers to infer complex requests. This method does not need to make significant changes to the original model structure or weight bits, nor does it need to train different teacher-student learning networks, which saves computing resources. In addition, recent work point out deep transformers turn to be over-thinking, and early exiting could be beneficial for inference accuracy and generalization ability (Dehghani et al., 2018).

There are mainly three early exiting strategies for BERT dynamic exiting. The first one is score-based early exiting. BranchyNet (Teerapittayanon et al., 2016b), 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 and Dumitras, 2018) 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., 2019). 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., 2021a) 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., 2020b) 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.

This work complements the literature by proposing a more flexible extension to the PABEE method. Our method tends out to be performant in both SLC and MLC tasks. In addition, our method can flexibly adjust the speedup ratios to meet different industrial requirements.

3 Flexible patience-based early exiting

3.1 Motivation

Despite its SOTA performances, PABEE may suffer from two drawbacks. First, PABEE's exiting criterion is quite strict, since it asks the predicted labels of a few consecutive layers to be exactly the same. Thus, 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. Second, intuitively, PABEE acts like a student taking an exam. It checks the answer to a query again and again using the intermediate exits. He will be assured if the answers remain the same. However, what if the later intermediate exit find something wrong and want to change the answer? Thus, PABEE's strict exiting criterion may miss out some early exiting opportunities.

We also notice that the literature on early exiting mainly focuses on SLC tasks, and the investigations on whether the existing early exiting methods work well on MLC tasks are lacking. Intuitively, exit criteria of patience-based methods PABEE is too strict, and the exiting criteria on MLC tasks may be difficult to meet.

In summary, a more flexible early exiting mechanism should be designed, so that it is flexible than PABEE, and can perform well on both SCL tasks and MLC tasks.

3.2 Inference procedure for SLC and MLC tasks

We now introduce our flexible PABEE (F-PABEE) method. The architecture of F-PABEE is shown in Figure 1(b), which is an improved version of PABEE(Figure1(a)). Where L_i is the transformer blocks of the model, n is the number of transformer layers in the model, C_i is the inserted classifier layer, and S is the similarity score between adjacent classifier layers, $thre$ is the threshold, and pat is the pre-defined patience.

The inference process of the model is obvious; that is, the input sentences first pass embedded to the vector by the embedding layer:

$$h_0 = Embedding(x) \quad (1)$$

The vector is then passed through $L_1...L_n$ layers to extract features and compute its hidden state h, after which we use an internal classifier connected to each $L_1 \ldots L_n$ layer predictors $C_1...C_n$ to predict probability p:

$$h_i = L_i(h_{i-1}); \quad (2)$$

$$p_i = C_i(h_i) \quad (3)$$

We denote the similarity score between the prediction results of layer i and layer $i+1$ as $s_{i,i+1} = S(p_i, p_{i+1})$ ($s_{i,i+1} \in \mathbf{R}$). The smaller the value of $s_{i,i+1}$, the prediction distributions p_i and p_{i+1} are more consistent with each other. The premise of the model's early exit is that the comparison scores between successive layers are relatively small; that is to say, the output results of these layers are relatively similar, and the model is more confident in its prediction results. The threshold of the score τ is a hyper parameter. We use pat_i to store the number of times that the cross-layer comparison score is consecutively less than the threshold when the model reaches the current layer:

$$pat_{i+} = \begin{cases} pat_i + 1 & s_{i,i+1} < \tau \\ 0 & s_{i,i+1} >= \tau \end{cases} \quad (4)$$

Where τ is a predefined threshold, if $s_{i,i+1}$ is less than the predefined threshold, then increase the patience counter by 1. Otherwise, reset the patience counter to 0. This process is repeated until pat reaches the predefined patience value P_0. The model dynamically stops inference and exits early. However, if this condition is never met, the model uses the last layer of classifiers to make predictions. In this way, the model can stop inference early without going through all layers.

3.3 Similarity measures for SLC tasks

Note that our F-PABEE framework is flexible and general, wrapping the PABEE method as a particular case. In PABEE, the value of $s_{i,i+1}$ is 1 only when the labels predicted by two adjacent layers are the same; otherwise, it is 0.

Under the framework of F-PABEE, we can adopt different similarity measures for predicted probability distributions. In this work, we will compare a series of divergence based measurements:

- F-PABEE-KL. This version of F-PABEE adopts the KL divergence from probability mass distribution p^l to distribution p^{l+1}:

$$KL^{(l,l+1)} = \sum_{j=1}^{k} p_j^l log(p_j^{l+1}); \quad (5)$$

- F-PABEE-ReKL. It also adopts the KL divergence, but from the reversed direction, that is, the KL divergence from p^{l+1} to p^l:

$$KL^{(l+1,l)} = \sum_{j=1}^{k} p_j^{l+1} log(p_j^l) \quad (6)$$

- F-PABEE-SymKL. Note that the usual KL divergence is actually asymmetrical, and we can make it symmetrical with following equation:

$$SymKL = KL^{(l,l+1)} + KL^{(l+1,l)}; \quad (7)$$

- F-PABEE-JSD. This version of F-PABEE employs the Jenson-Shannon divergence:

$$JSD^{(l,l+1)} = 1/2(KL(p^l/\tilde{p}) + KL(p^{l+1}/\tilde{p})), \quad (8)$$

  where $\tilde{p} = 0.5 * (p^l + p^{l+1})$ is the average distribution of p^l and p^{l+1}. Note that JSD is also a symmetrical divergence measure.

3.4 Similarity measures for MLC tasks

For MLC tasks, the judgment criterion based on Patience requires the prediction results of each label in the adjacent layer to be the same, which is too strict. Many results are correct on most labels, and wrong results on a few labels will be excluded, leading to difficulties in exiting.

F-PABEE can compare the prediction results, allowing a few different labels between different layers to achieve a balance between accuracy and efficiency. Therefore, our method also has significant advantages in multi-label.

For MLC tasks, we transform it into multiple binary classification problems and sum the divergences of all categories, and the formula is:

- F-PABEE-KL:

$$KL^{(l,l+1)} = \sum_{j=1}^{k}\sum_{i=1}^{2} p_{ji}^{l} log(p_{ji}^{l+1}); \quad (9)$$

- F-PABEE-ReKL:

$$KL^{(l+1,l)} = \sum_{j=1}^{k}\sum_{i=1}^{2} p_{ji}^{l+1} log(p_{ji}^{l}) \quad (10)$$

- F-PABEE-SymKL:

$$BiKL = KL^{(l,l+1)} + KL^{(l+1,l)}; \quad (11)$$

- F-PABEE-JSD:

$$JS^{(l,l+1)} = 1/2 \sum_{j=1}^{k}\sum_{i=1}^{2} (KL(p_{ji}^{l}/\tilde{p}) + KL(p_{ji}^{l+1}/\tilde{p})), \quad (12)$$

where $\tilde{p} = 0.5 * (p_{ji}^{l} + p_{ji}^{l+1})$ is the average distribution of p_{ji}^{l} and p_{ji}^{l+1}.

3.5 Training procedure

The model's training process is as follows: when the input sample is received, each output will calculate the loss according to its prediction result, and the sum of the losses will be optimized simultaneously for all outputs. Our model is trained and inferred on both SLC and MLC tasks, where the activation and loss functions are different for both tasks. We use the sigmoid activation function for the MLC function and calculate the loss using the cross-entropy function for binary classification. While, for the SLC tasks, we use the softmax function as the activation function and calculate the cross-entropy function according to the tasks.

After that, we optimize the model parameters by minimizing the overall loss function L ,which is the weighted average of the loss terms from all exits

$$L = \sum_{j=1}^{n} jL_j / \sum_{j=1}^{n} j \quad (13)$$

4 Experiments

4.1 Tasks and Datasets

We evaluated our proposed F-PABEE to the SLC tasks on the GLUE benchmark. We excluded STS-B since it is a regression task, and we also exclude the WNLI task follwing previous work. (Devlin et al., 2018) .Meanwhile, for MLC tasks, we evaluated on four datasets : MixATS (Hemphill et al., 1990) , MixSNLPS (Coucke et al., 2018) , AAPD (Yang et al., 2018) , and Stack overflow (Atwood, 2009) datasets. the size of the dataset is shown in the table 2 in the Appendix.

4.2 Baselines

We compare our F-PABEE with three groups of baselines.

Backbone models: We mainly choose the BERT-base and ALBERT-base model open-sourced by (Devlin et al., 2018) as the backbone model.

Budgeted exiting: If the computational budget is known, Budgeted exiting method are appointed a suitable exit layer of BERT, to predict all queries. we compare the model with different Budgeted exiting layers: Budgeted-Exit-3L, Budgeted-Exit-6L, Budgeted-Exit-9L.

Dynamic exiting: We compare our methods with a seriese of dynamic exiting methods, including BrachyNet (Teerapittayanon et al., 2016a), Shallow-Deep (Kaya et al., 2019b), BERxiT (Xin et al., 2021b) and PABEE (Zhou et al., 2020a). 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 F-PABEE so that all methods' number of average inference layers are close.

4.3 Evaluation of early exiting method

For SLC tasks, we follow the GLUE benchmark to report the performances metrics on each

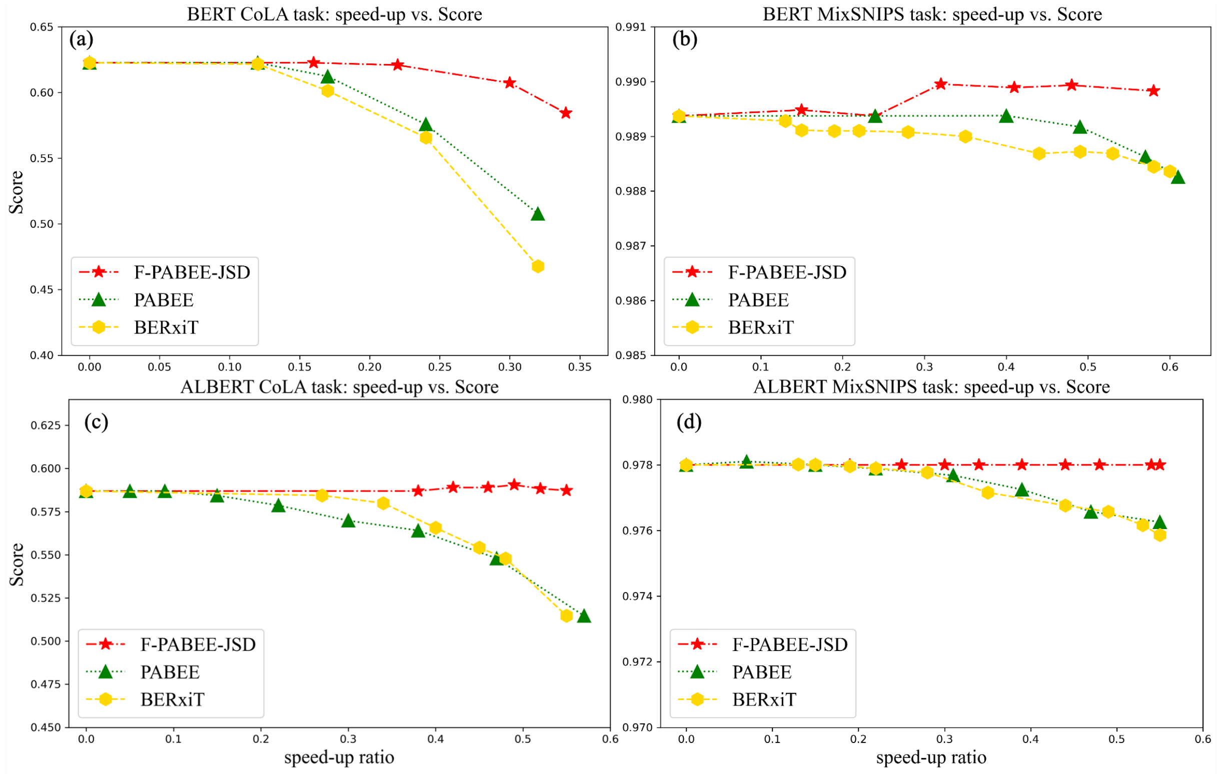

Figure 2: Speed-accuracy curves of F-PABEE, PABEE,BERxiT on the CoLA (a SLC task) and MixSNIPS (a MLC task) tasks, when the BERT base is the backbone model.

	CoLA		MNLI		MRPC		QNLI		QQP		RTE		SST-2	
	score	speedup	score	speedup	score	speedup	score	speedup	score	speedup	score	speedup	score	speedup
BERT base	54.2	0%	83.1	0%	86.8	0%	89.8	0%	89.2	0%	69.1	0%	91.3	0%
Budgeted-Exit-3L	0.0	75%	70.0	75%	75.8	75%	77.4	75%	81.8	75%	54.7	75%	81.0	75%
Budgeted-Exit-6L	0.0	50%	79.6	50%	84.7	50%	85.3	50%	89.3	50%	68.1	50%	88.6	50%
Budgeted-Exit-9L	51.9	25%	83.0	25%	87.0	25%	88.4	25%	90.3	25%	69.0	25%	91.2	25%
BranchyNet	0.0,	74%	63.8	76%	75.7	76%	74.2	80%	71.6	80%	54.7	76%	79.9	76%
	0.0	51%	78.3	53%	83.0	52%	87.1	47%	89.3	50%	67.4	47%	88.3	49%
	52.1	27%	83.0	25%	85.8	24%	89.3	27%	90.1	26%	68.0	26%	91.2	24%
Shallow-Deep	0.0	75%	64.1	77%	75.6	76%	74.3	78%	71.4	79%	54.7	76%	79.5	77%
	0.0	52%	78.2	51%	82.8	51%	87.2	49%	89.6	51%	67.2	48%	88.4	48%
	52.3	26%	82.9	26%	85.7	25%	89.3	26%	90.1	27%	67.8	26%	91.2	25%
BERxiT	0.0	76%	63.5	76%	75.6	76%	73.3	78%	68.2	80%	55.3	77%	79.5	76%
	12.3	52%	78.4	51%	82.9	51%	87.0	48%	89.1	49%	67.3	47%	88.3	49%
	52.2	25%	83.2	26%	86.2	26%	89.6	27%	90.1	26%	68.1	27%	91.4	24%
PABEE	0.0	75%	63.9	77%	75.8	75%	73.6	81%	68.6	82%	55.8	75%	79.9	77%
	0.0	50%	78.9	52%	83.1	53%	87.2	46%	89.6	49%	67.7	46%	88.7	48%
	52.4	26%	83.4	24%	86.1	26%	89.8	28%	90.4	24%	68.3	28%	91.7	22%
F-PABEE	0.0	75%	66.9	72%	81.5	77%	76.2	75%	79.6	82%	56.0	76%	80.5	76%
	13.6	52%	83.9	53%	87.3	53%	88.6	54%	90.8	49%	68.1	47%	92.3	48%
	62.8	25%	84.2	28%	88.0	26%	90.9	27%	91.2	25%	68.9	25%	92.4	25%

Table 1: Experimental results of different early exiting methods with the same fine-tuned BERT backbone on the GLUE benchmark. The results show that F-PABEE is effective in accelerating BERT's inference with less performance loss compared with the baseline methods.

task. For MLC tasks, we use accuracy as the performance metrics. The efficiency metric follows PABEE (Zhou et al., 2020b),we mainly evaluate the speedup ratio. Assuming the model has M layers, for each instances $x_i (i \in 0, 1, ..., N)$, the exiting layer is m_i, the average speedup ratio is :

$$Speedup = 1 - \sum_{i=1}^{i=N} m_i / \sum_{i=1}^{i=N} M \qquad (14)$$

4.4 Experimental setting

Training We add a linear output layer after each intermediate layer of BERT and ALBERT back-

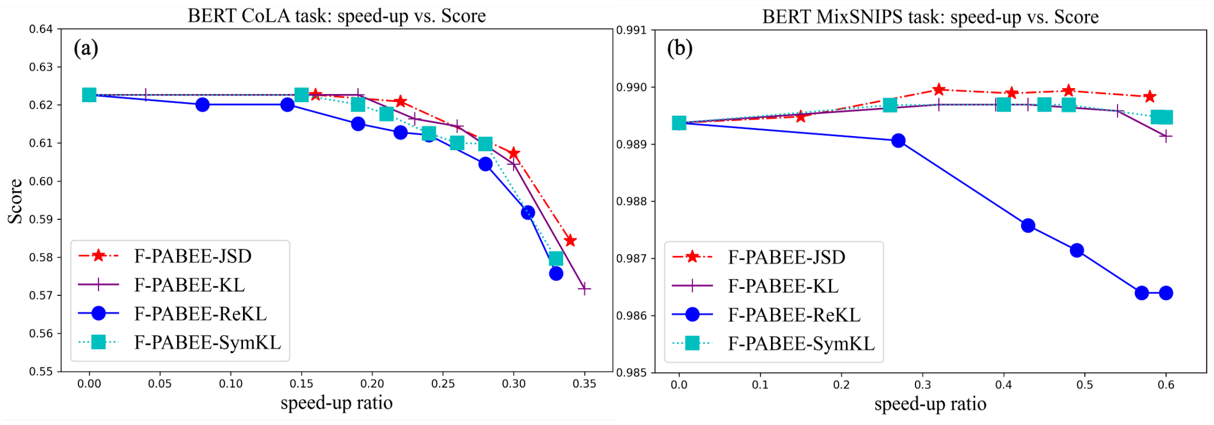

Figure 3: Speed-accuracy curves of different similarity score calculation methods on SLC and MLC tasks

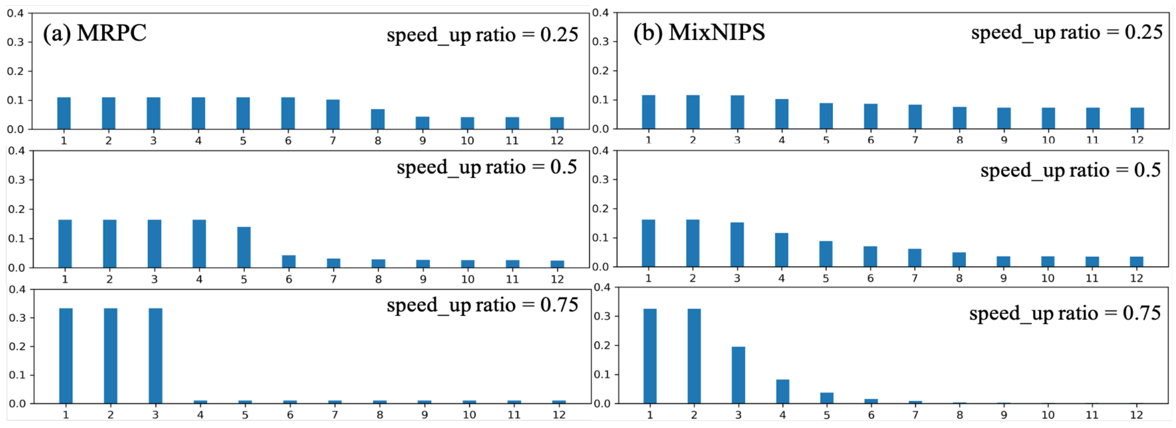

Figure 4: The distribution of executed layers of MRPC and MixSNIPS on average at three different speeds(0.25,0.5,0.75)

bone models as the internal classifiers. In addition, we perform grid search over the batch size of {16, 32, 128}, and learning rate of {1e-5, 2e-5, 3e-5, 5e-5} with an AdamW optimizer (Loshchilov and Hutter, 2019). The hyper-parameters are selected via cross validation on the training set of GLUE. We implement F-PABEE on the bases of Hugging-Face Transformers (Wolf et al., 2020). We conduct our experiments on a single Nvidia TITAN X 24GB GPU.

Inference Following prior work on input-adaptive inference (Teerapittayanon et al., 2016a; Kaya et al., 2019a), 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.

5 Results and discussions

5.1 Overall comparison

In Table 1, we report the performance comparisons of each method on the GLUE benchmark under three different speedup settings: (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 our similarity score threshold) of our F-PABEE and the other baselines to achieve similar speedups with PABEE. The results in table 1 clearly show that our PCEE-BERT method outperforms the baseline methods under different speedup ratios. Table 1 also shows that the PABEE method is the best performing baseline.

In order to further analyze and better visualize the results, we draw the score-speedup curves (in Figure 2) for BERxiT, PABEE and our F-PABEE, on the CoLA and MixSNIPS tasks. [2] With Table 1

[2]The score-speedup curves for the other tasks can be found

and Figure 2, we can make the following observations:

- Our F-PABEE method consistently outperforms the baseline methods on both the SLC and MLC tasks.
- 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 a softer score mechanism across layers.
- Our F-PABEE consistently performs better than the baseline methods, especially when the speedup ratio is large. Note that our F-PABEE also consistently outperforms the budgeted exiting speedup ratios, which the other baselines do not achieve.
- The overthinking problem is prevailing in the benchmark classification tasks, and our F-PABEE early exiting can effectively take advantage of this phenomenon. For 2 of the GLUE tasks, PCEE-BERT can outperform BERT-base with a 50% (or more than) speedup ratio. The speedup ratios without any performance losses are also high in the four MLC tasks.

Putting performance comparisons aside, one benefit of F-PABEE over the previous SOTA method, PABEE, 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.

5.2 Ablation studies

Ablation of PLM backbones In the main experiments, we use BERT as the pre-trained backbone model. However, F-PABEE is off-the-shelf, and can also work with the other types of pre-trained backbones, such as ALBERT base. We conduct the experiments on the CoLA and MixSNIPS tasks with ALBERT base as the backbone, and results are presented in the lower row of Figure 2. We can see that when using the other pre-trained backbones, PCEE-BERT also performs better than the baseline methods.

Comparisons between different similarity score calculation methods: In our main experiments, we main set JSD as our F-PABEE's similarity measure (F-PABEE-JSD). However, as we have shown in Section 3, F-PABEE is off-the-shelf, and it can work with other similarity measure as well. Thus, we consider the following variants of F-PABEE: F-PABEE-KL, F-PABEE-ReKL, and F-PABEE-SymKL. Their early exiting results are presented in Figure 3.[3]

We think that it is intuitive that F-PABEE-JSD works best for F-PABEE. JSD is symmetric, and its numerical values is more stable then the other measures. To understand the working mechanism of F-PABEE-JSD, we plot the distributions of exiting depth at the speedup around 25%, 50% and 75% (in Figure 4). We can see that the distribution of layer depths are quite balanced, showing that F-PABEE-JSD is good at determine which samples should be paid more attention, and the others should exit at shallow layers.

6 Conclusion

In this paper, we proposed F-PABEE, a novel and efficient method combining the PABEE with a softer cross-layer comparison strategy. F-PABEE is more flexible than PABEE since it can achieve different speed-performance tradeoffs by adjusting the similarity score threshold and the patience parameter. In addition, the similarity score is measured by divergence-based metrics like KL-divergence and JSD-divergence from the current and previous layers. Extensive experiments on SLC and MLC tasks demonstrate that: (1) Our F-PABEE performs better than the previous SOTA adaptive early exit methods for SLC tasks. (2) Our F-PABEE performs better than other early exit methods for MLC tasks. As far as we know, we are the first to investigate the early exiting method of MLC tasks. (3) F-PABEE performs well on different PLMs such as BERT and ALBERT. (4) ablation studies show that JSD-divergence's similarity calculation method outperforms other similarity score calculation methods for our F-PABEE model.

in the appendix.

[3]See more comparisons on the similarity measure in Appendix.

7 Ethical considerations

F-PABEE can be seen as a natural extension of the PABEE method. It is more flexible than PABEE since it can adjust the similarity score and the patience threshold according to the data characteristics and request traffic. Moreover, it is more applicable for pre-trained models to be deployed on real-time search engines or software for public communication and edge devices with limited hardware, such as mobile phones and smartwatches. Furthermore, It does not introduce new ethical concerns, but more work is needed to determine its effect on biases encoded in PLMs.

8 Limitation

Although F-PABEE can improve both efficiency and accuracy, it also has limitations. For example, this method is only suitable for classification problems, not regression problems, which restricts the application of F-PABEE. Moreover, more suitable similarity score calculation methods may exist that can improve efficiency. Therefore, for future work, we would like to explore efficient models for regression tasks and more similar evaluation metrics for more tasks and models.

References

Jeff Atwood. 2009. Stack overflow creative commons data dump. https://archive.org/details/stackexchange.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv*, abs/1805.10190.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2018. Universal transformers. *arXiv*, abs/1807.03819.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. 2019. Depth-adaptive transformer. *arXiv*, abs/1910.10073.

Angela Fan, Edouard Grave, and Armand Joulin. 2019. Reducing transformer depth on demand with structured dropout. *arXiv*, abs/1909.11556.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*.

Lu Hou, Lifeng Shang, Xin Jiang, and Qun Liu. 2020. Dynabert: Dynamic BERT with adaptive width and depth. *arXiv*, abs/2004.04037.

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*, abs/1909.10351.

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

Yigitcan Kaya and Tudor Dumitras. 2018. How to stop off-the-shelf deep neural networks from overthinking. *arXiv*, abs/1810.07052.

Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. 2019b. Shallow-deep networks: Understanding and mitigating network overthinking. In *International conference on machine learning*, pages 3301–3310. PMLR.

Sehoon Kim, Amir Gholami, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. 2021. I-BERT: integer-only BERT quantization. *arXiv*, abs/2101.01321.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. *arXiv*, abs/1909.11942.

Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. 2021. A survey of transformers. *arXiv*, abs/2106.04554.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Haotang Deng, and Qi Ju. 2020. Fastbert: a self-distilling BERT with adaptive inference time. *arXiv*, abs/2004.02178.

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. *arXiv*, abs/1907.11692.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *ICLR*.

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

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 Noah A. Smith. 2020a. The right tool for the job: Matching model and instance complexities. *arXiv*, abs/2004.07453.

Roy Schwartz, Gabriel Stanovsky, Swabha Swayamdipta, Jesse Dodge, and Noah A Smith. 2020b. The right tool for the job: Matching model and instance complexities. *arXiv preprint arXiv:2004.07453*.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for BERT model compression. *arXiv*, abs/1908.09355.

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

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

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 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.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang 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. 2021a. 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.

Ji Xin, Raphael Tang, Yaoliang Yu, and Jimmy Lin. 2021b. 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.

Pengcheng Yang, Xu Sun, Wei Li, Shuming Ma, Wei Wu, and Houfeng Wang. 2018. SGM: sequence generation model for multi-label classification. *arXiv*, abs/1806.04822.

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

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

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

A Appendix

A.1 Details of the datasets

Here we presented the in table 2 the details of the datasets for both SLC and MLC tasks.

A.2 comparson of different adaptive early exit methods

In the main context, we present the Speed-accuracy curves for SLC task CoLA and MLC task MixSNIPS, and we present the Speed-accuracy curves of other SLC and MLC tasks as shown in Figure 5 and 6.

A.3 comparison of different similarity score calculation methods

In the main context, we present the ablation results for different similarity calculation methods of SLC task CoLA and MLC task MixSNIPS. We also present the ablation results of other SLC and MLC tasks here. as shown in Figure 7 and 8.

Table 2: Statistics of single and multi label datasets

Dataset	Labels	Train/Dev/Test
COLA	2	8.5k/1.0k/1.0k
RTE	2	2.5k/0.3k/3.0k
MRPC	2	3.7k/0.4k/1.7k
SST-2	2	67k/0.9k/1.8k
QNLI	2	105k/5.5k/5.5k
MNLI	3	393k/9.8k/9.8k
AAPD	54	53k/6.7k/6.7k
Stackflow	50	80k/10k/10k
MixATS	17	18k/1k/1k
MixSNLPS	7	45k/2.5k/2.5k

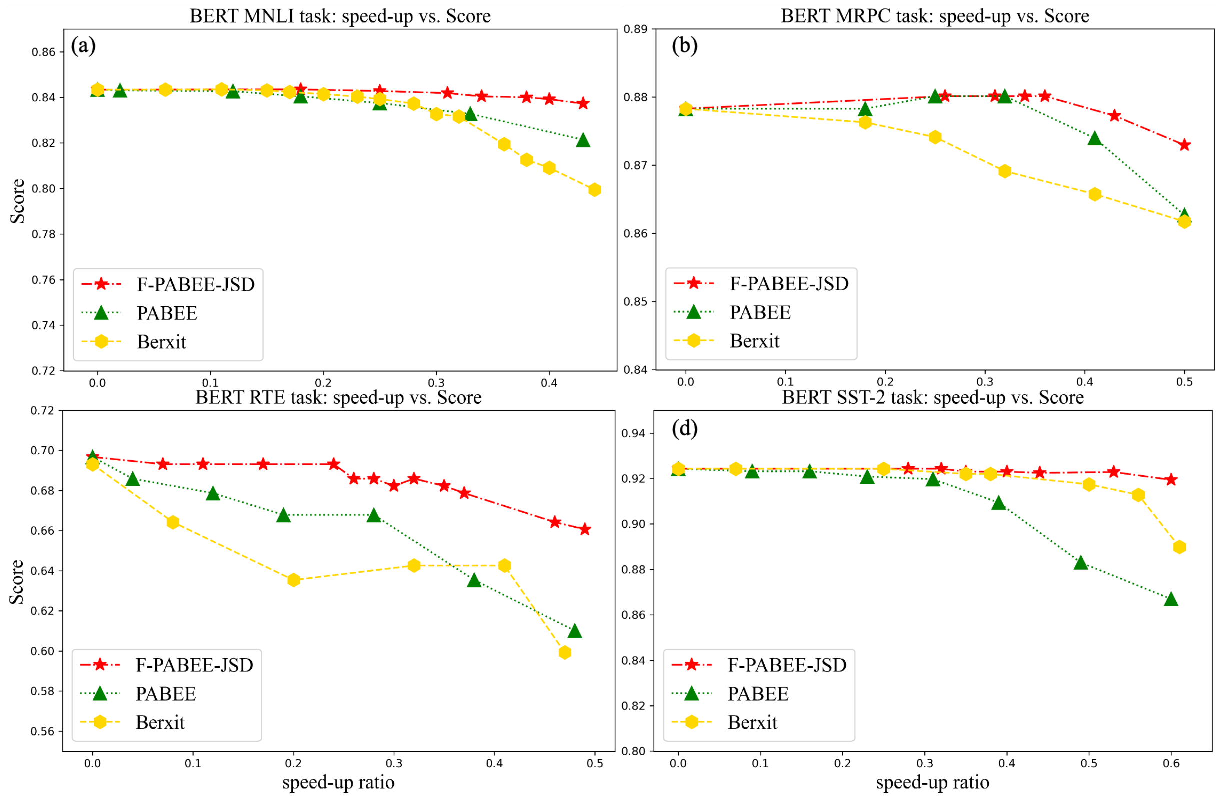

Figure 5: Speed-accuracy curves of F-PABEE, PABEE,BERxiT on SLC tasks with BERT as the backbone

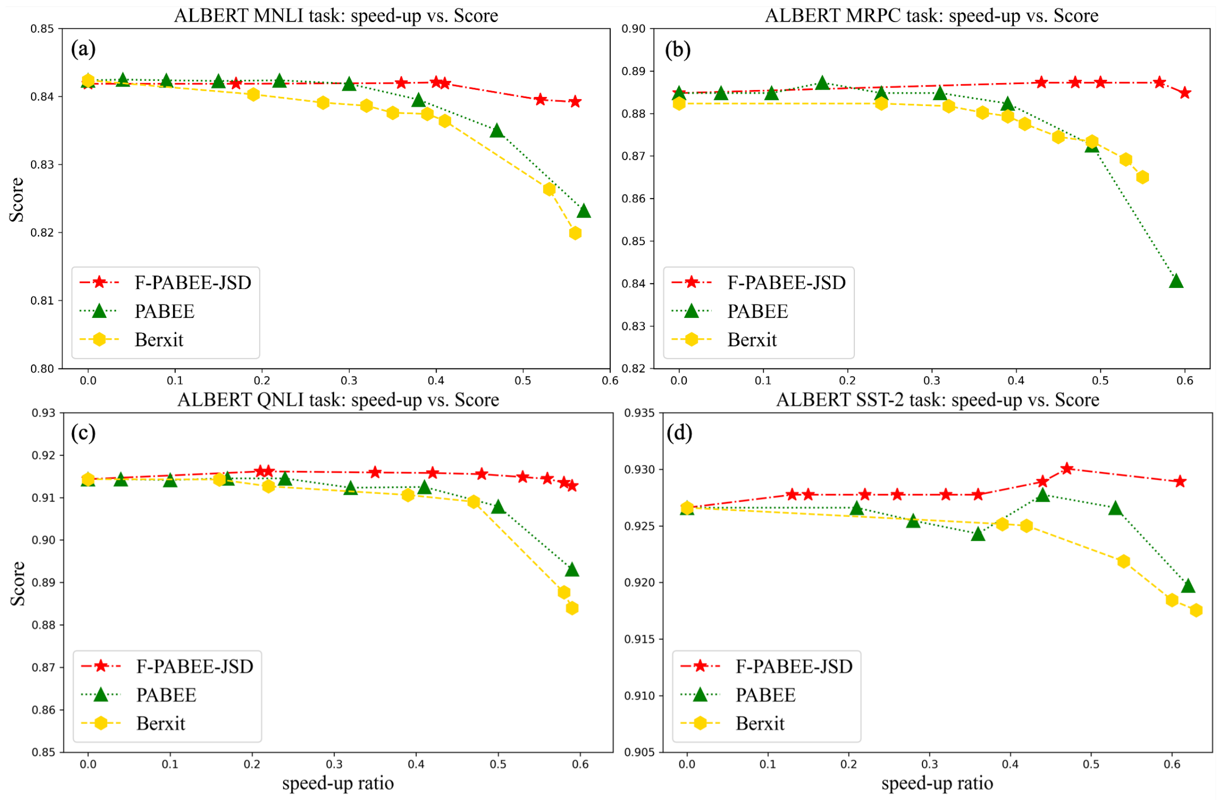

Figure 6: Speed-accuracy curves of F-PABEE, PABEE,BERxiT on SLC tasks with ALBERT as the backbone

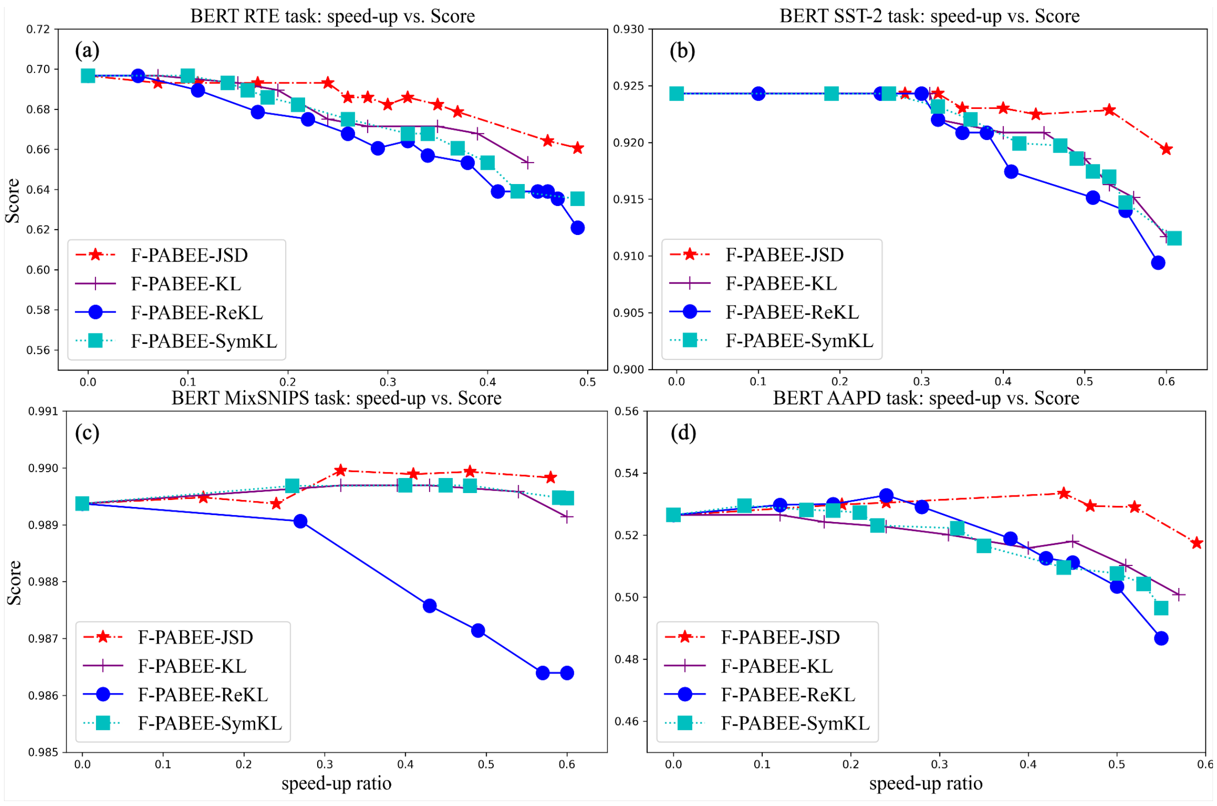

Figure 7: Speed-accuracy curves of different similarity score calculation methods on SLC and MLC tasks with BERT as backbone

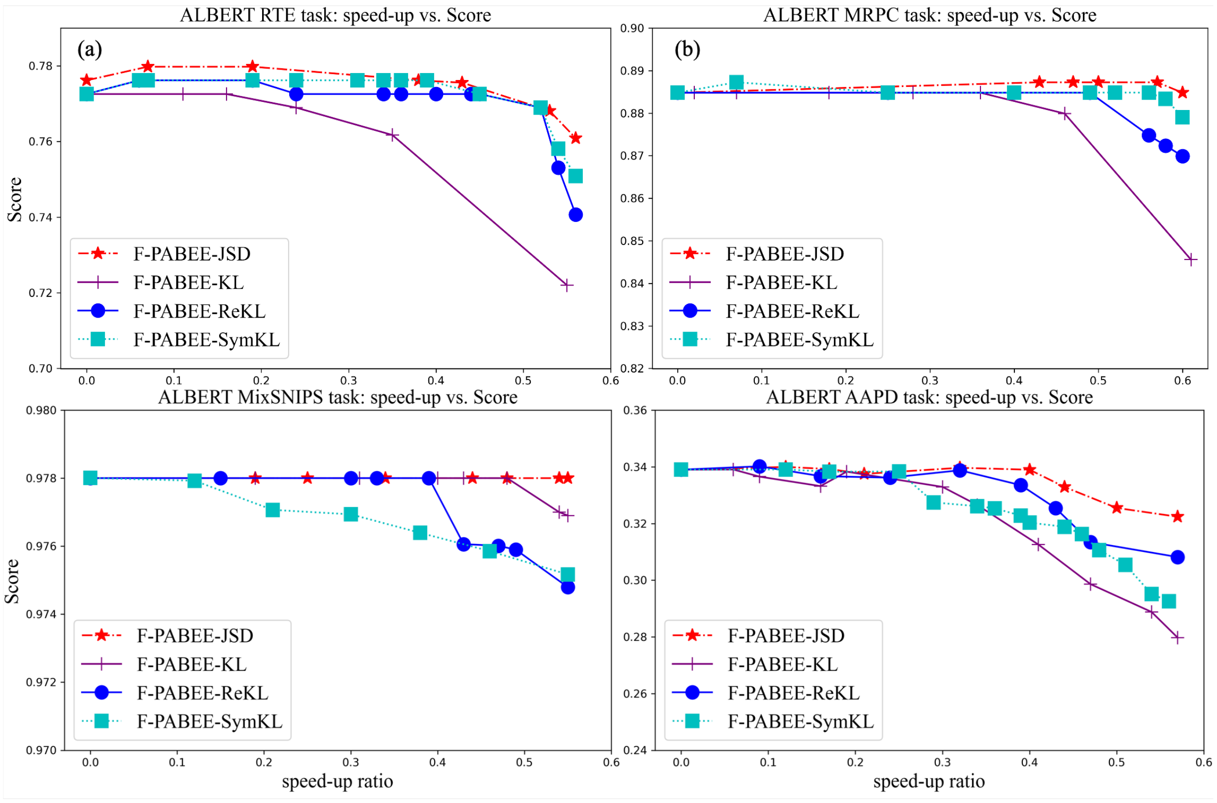

Figure 8: Speed-accuracy curves of different similarity score calculation methods on SLC and MLC tasks with ALBERT as backbone