Neurocomputing 558 (2023) 126690

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

PF-BERxiT: Early exiting for BERT with parameter-efficient fine-tuning and flexible early exiting strategy

Xiangxiang Gao, Yue Liu, Tao Huang, Zhongyu Hou *

National Key Laboratory of Science and Technology on Micro/Nano Fabrication, Department of Micro/Nano Electronics, School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, PR China

ARTICLE INFO

Communicated by Danilo Cavaliere

Keywords:
Parameter-efficient
Flexible early exiting
Fine-tuning
BERxiT

ABSTRACT

The industrial usage of huge pre-training language models such as BERT and ALBERT are limited by the computational probability problem in the fine-tuning process and overthinking problem in the inference process. PF-BERxiT has been proposed to optimize the pre-trained languages with a novel parameter-efficient fine-tuning method and a flexible early exiting strategy. Significantly, the new parameter-efficient fine-tuning model integrates a bottleneck adapter architecture parallel to the transformer architecture, and only the adapter's parameters are adjusted. In addition, we integrate an extra sub-learning module to learn the samples' characteristics, improving the accuracy and efficiency simultaneously. The flexible exiting strategy allows the model to exit early if the similarity score of adjacent layers is less than the threshold for pre-defined times. It is more flexible than previous early exiting methods, for it can simultaneously adjust the similarity score thresholds and patience parameters according to the request traffic. Extensive experiments are conducted on the GLUE benchmark, demonstrating that: (1) PF-BERxiT outperforms conventional training and parameter-efficient strategies with only a few parameters fine-tuned. (2) PF-BERxiT strikes a better balance between model performances and speedup ratios than previous state-of-the-art (SOTA) early exiting methods such as PABEE and BERxiT. (3) Ablation studies in the fine-tuning process demonstrate that the best bottleneck dimension r of the adapters is 32, and the adapters placed parallel to the feed-forward module are more efficient. (4) Ablation studies in the inference process demonstrate that for variants of PF-BERxiT with different similarity scores, PF-BERxiT-kl and PF-BERxiT-bikl attain better speedup-accuracy trade-offs than PF-BERxiT-rekl. Our PF-BERxiT helps attain a better trade-off between performance and efficiency, providing a reference for the efficient application of neural computing.

1. Introduction

Large-scale Pre-trained language models (PLMs) have achieved state-of-the-art performance on many natural language processing tasks [1–3]. Fine-tuning all the parameters of PLMs on specific tasks has been the paradigm in many natural language processing tasks [4,5]. However, PLMs contain hundreds of millions or even hundreds of billions of parameters, making them computationally expensive and inefficient regarding memory consumption and latency [6–9]. Another bottleneck of PLMs is the overthinking problem in the inference process [10–12]. That is, the representations of the model's shallow layers are adequate to make decisions, while the representations of the deep layers may be distracted by irrelevant and over-complicated features [13,14]. The overthinking problem slows the inference speed and wastes the computation resources [15,16].

To mitigate the computationally expensive and inefficient problem in the fine-tuning process, two-stage fine-tuning methods [17] and alternating fine-tuning methods [18] are proposed. However, the fine-tuned parameters are still redundant. Therefore, many researchers propose various parameter-efficient transfer learning methods to update only a small number of task-specific parameters while keeping most pre-trained parameters frozen [19]. The parameter-efficient model refers to integrates a simple module parallel to the transformer architecture, denoted as an adapter, which is a similar bottleneck architecture that imposes a low-rank constraint on the parameter updates. Only the parameters of the adapter modules are adjusted during the fine-tuning process, which only fine-tunes a small number of parameters to attain comparative performance. However, the performance of the parameter-

* Corresponding author.
E-mail address: zhyhou@sjtu.edu.cn (Z. Hou).

https://doi.org/10.1016/j.neucom.2023.126690
Received 27 December 2022; Received in revised form 8 June 2023; Accepted 13 August 2023
Available online 25 August 2023

efficient transfer learning methods mentioned is limited. It is necessary to propose an optimized solution to improve accuracy under similar tunable parameters.

In addition, inspired by early stopping [20,21], many researchers propose early exiting to mitigate the overthinking problem and accelerate the inference process [22,23]. Early exiting couples an internal classifier with each transformer layer [24] and dynamically stops inference when the intermediate predictions of the classifiers are stable for several steps [25]. There are mainly two types of early exiting strategies. The first type is the confidence-based early exiting strategy. It evaluates the current layer's intermediate predictions based on specific confidence measurements. The confidence-based early exiting strategies are based on the predicted probabilities, significantly saving inference time. However, Schwartz et al. [22] revealed that early exit based on predicted probabilities often leads to a significant drop in performance because many samples meet the early exit criteria by chance before reaching a steady state [18]. Zhou et al. [25] propose PABEE, a patience-based early exiting method that relies on inter-layer prediction instead of predicted probabilities to overcome this problem. However, due to its strict cross-layer comparison strategy, PABEE cannot flexibly adjust the speedup ratio on a given task and a fixed patience parameter [18].

Therefore, to alleviate the computationally expensive and inefficient problem in the fine-tuning and inference process, we propose PF-BERxiT, which optimizes the multi-exit BERT with a novel parameter-efficient method and a flexible early exiting module. The parameter-efficient method integrates bottleneck adapters parallel to the transformer architecture that imposes a low-rank constraint on the parameter updates, which decreases the tuned parameters [26–29]. The difference in our method is that sub-learning modules are integrated into the adapters to learn the hidden characteristics of the samples. This way, the model can fine-tune a small number of weights to attain strong performance [30,31]. The flexible early exiting module makes predictions at each intermediate layer. It calculates the similarity score and will exit early if the current layer and the adjacent layers have similar predicted distributions (similarity score less than a threshold) for pre-defined times (patience). The similarity scores are measured by the distillation object. The early exiting strategy can be seen as an extension of confidence-based and patience-based early exiting strategies. It is more flexible since it can make a trade-off between inference speed and performance by simultaneously adjusting the similarity score thresholds and patience parameters.

We conduct extensive experiments on the GLUE benchmark [32]. The results demonstrate that our PF-BERxiT outperforms the previous SOTA training and parameter-efficient transfer learning methods in the fine-tuning processes. In addition, it also makes a better performance and efficiency trade-off in the inference process than the previous SOTA early exiting methods such as PABEE and BERxiT. Our contributions are three-fold: (1) We propose PF-BERxiT, a novel model which optimizes PLMs with a novel parameter-efficient transfer learning method and a flexible early exiting method. (2) We integrate the sub-learning module into the bottleneck adapter to learn the hidden characteristics of the samples in the fine-tuning process. The model can attain strong performance by fine-tuning a small number of parameters. (3) We propose a flexible early exiting method that can balance efficiency and performance better because it is flexible in adjusting the similarity score and patience threshold simultaneously.

2. Related works

2.1. Conventional training and parameter-efficient strategies

To mitigate the computation expenses and inefficiency problem in the fine-tuning process of Joint training (fine-tunes all the parameters of the backbone and exits simultaneously). Two-stage fine-tuning methods [17] are proposed, which fine-tune the backbone and the final layer at the first stage. Then fine-tunes the intermediate exits on the fixed backbone. The alternating fine-tuning method alternates between joint and two-stage training, which outperforms two-stage fine-tuning methods. BERxiT [18] is an improved version of the alternating fine-tuning strategy, which optimizes the model in the fine-tuning and inference process (adding learning to the exit module). However, their parameters are still redundant, and efficiency improvements are limited.

Therefore, many researchers sought to mitigate this problem with parameter-efficient transfer learning methods [49–52]. For example, Neil et al. [33] insert adapter modules into each layer of the pre-trained model, and only the parameters of the adapter modules are adjusted during the fine-tuning process. Hu et al. [34] freeze the pre-trained model's parameters and inject trainable rank decomposition matrices into each transformer layer, significantly reducing the tunable parameters of specific tasks. Zhu et al. [35] propose a bottleneck architecture for the adapter module to save the parameters. The adapter modules are parallel connected to the transformer layers; the results show that parallel adapter outperforms series adapter. Besides reducing parameters, parameter-efficient transfer learning is robust [36] and can be adjusted rapidly to new tasks without catastrophic forgetting [37]. However, the performances of the parameter-efficient transformer learning methods could be improved.

To improve the performance of parameter-efficient transformer learning methods, we integrate sub-learning modules into the adapters and improve the model performance under a similar number of tunable parameters to conventional parameter-efficient methods such as parallel adapters. Our proposed PF-BERxiT can be seen as an improved version of BERxiT, significantly improving fine-tuning efficiency.

2.2. Early exiting strategies in the inference process

Early exiting is a method that dynamically adjusts certain hyperparameters in response to the samples. It does not need to make huge changes to the original model structure, which saves computing resources [38]. It mainly includes two kinds of early exiting methods: confidence-based and patience-based early exiting methods. The confidence-based early exiting methods evaluate the current layer's intermediate predictions based on specific confidence measurements. The patience-based early exiting methods rely on inter-layer prediction instead of predicted probabilities.

The confidence-based early exiting includes BERxiT, BranchyNet [39], and DeeBERT [23], which utilize the entropy of the intermediate predictions as a measure of confidence. If the entropy exceeds the threshold, the model is confident in the prediction, and the sample exits the network. In addition, Shallow-deep [10] denotes the maximum probability mass as the confidence. However, it may occasionally exit before the model reaches the steady state, which leads to a performance drop. The patience-based early exiting method is proposed by Zhou et al. [25], it will stop inference and exit early if the predicted results of the internal classifiers remain unchanged for a predefined number of steps. However, the exit criteria are too strict, which reduces efficiency.

To overcome the drawbacks proposed above, we propose PF-BERxiT, which will exit early if the similarity score of adjacent layers is less than the threshold for pre-defined times. PF-BERxiT is more flexible because it can adjust the similarity score threshold and patience parameters simultaneously. It not only mitigates the performance drop for occasional exits before reaching a steady state of confidence-based early exiting methods. But also alleviates the inefficiency of too strict exiting criteria of patience-based early exiting methods, making a better speed-performance trade-off.

3. Methods

PF-BERxiT is a more efficient and flexible version of BERxiT, it optimizes the BERxiT from two aspects: (1) It proposes a novel parameter-efficient transfer learning model to mitigate the computationally expensive and inefficient problem in the fine-tuning process. The

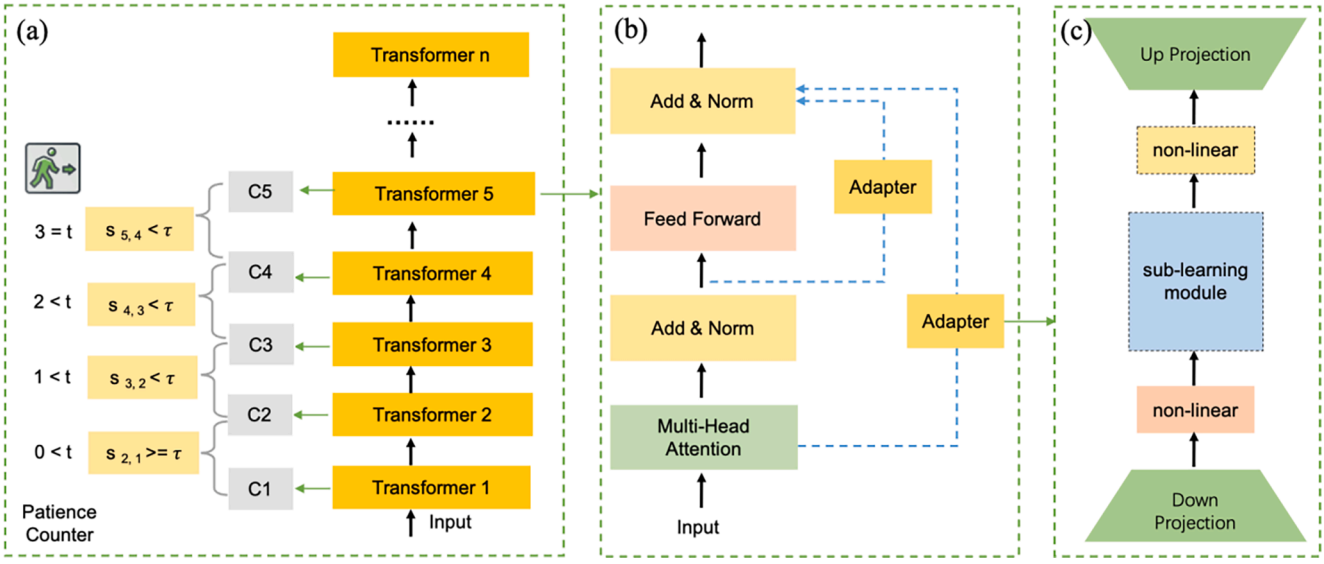


Fig. 1. The parameter-efficient fine-tuning and inference architecture of the PF-BERxiT framework.

parameter-efficient model integrates an adapter to the model, a bottleneck architecture that integrates a sub-learning module to learn the hidden characteristics. (2) It proposes a flexible early exiting method that makes a better trade-off between inference speed and performance to mitigate inefficiency and overthinking problem in the inference process.

3.1 Model structure

In this section, we present the novel parameter-efficient transfer learning and flexible early exiting architecture of PF-BERxiT. As shown in Fig. 1, the backbone consists of n stacked Transformer blocks. In addition, a classifier is attached to each transformer layer to enable early exiting. The classifier is a one-layer fully connected network, which takes the hidden state of each transformer layer and outputs a prediction. Each classifier can make a prediction, and the model can early exit dynamically according to the prediction of each classifier. Moreover, the learning-to-exit (LTE) is connected to each classifier to improve the inference efficiency [18]. The LTE module is a simple one-layer fully-connected network, which takes the hidden states as the input and outputs a certain level of the sample of each layer.

Fig. 1 (b) shows that each transformer block consists of a multi-head attention module and position-wise feed-forward. In addition, a residual connection is employed, followed by the Normalization module [40]. Firstly, the transformer blocks are trained on the entire corpus, and its parameters are frozen in the fine-tuning process. Then, the adapters are connected with the attention module and the feed-forward module of each transformer block, and the parameters of the adapters are trained in the fine-tuning process.

The architecture of the adapter is shown in Fig. 1 (c). The adapter is a bottleneck architecture. It firstly projects the original d dimension input representations H_{i-1}^{PA} to a small dimension $r(r \ll d)$ by a down-projection matrix $W_{down} \in R^{d \times r}$, followed by a non-linear activation function $g(.)$, and a sub-learning module $f(.)$ to learn the hidden characteristics, then projects the r dimension representations back to d dimensions by an up-projection matrix $W_{up} \in R^{r \times d}$. We denote r as the bottleneck dimension and H_i^{PA} as the output of each adapter module. The formulation of each adapter module is denoted as follows:

$$H_i^{PA} = f(g(H_{i-1}^{PA} W_{down}))W_{up} \tag{1}$$

Among them, the non-linear activation function $g(.)$ is usually RELU [41],GELU [42], and Tanh [43], or an identity transformation. In addition, we propose four different sub-learning modules $f(.)$ to learn the hidden features of the samples, and they are denoted as:

(1) PF-BERxiT(origin-adapter): The sub-learning module is a simple identity mapping. The architecture of this adapter module is the conventional adapters presented in parallel adapters [35].
(2) PF-BERxiT(self-attn-adapter): The sub-learning module is a self-attention layer [24], followed by a non-linear activation function.
(3) PF-BERxiT(conv-adapter): The sub-learning module is a 1-d convolutional layer [44], followed by a non-linear activation function.
(4) PF-BERxiT(mlp-adapter): The sub-learning module is a simple fully-connected layer [45].

During the fine-tuning process, only the parameters of adapters are trained, alleviating the computational prohibitively and inefficiency problem. In addition, the added sub-learning module proved beneficial in learning the samples' hidden characteristics and improving the model's performance.

In addition, to mitigate the computational consumption and overthinking problem in the inference process. We propose flexible early exiting strategies, as shown in Fig. 1(a). Among them, C_n is the classifier attached to each transformer layer, s is the similarity score between different layers, τ is the similarity score threshold, and t is the predefined patience threshold.

The inference process is evident. The input samples are firstly embedded as vectors by the embedding layer:

$$h_0 = Embedding(x) \tag{2}$$

Next, the vectors pass through the transformer blocks $L_1 \cdots L_i$ to extract the hidden features and compute its hidden states h_i. Then, the probability p is calculated by each classifier $C_1 \cdots C_i$:

$$h_i = L_i h_{i-1} \tag{3}$$

$$p_i = C_i h_i \tag{4}$$

After that, the similarity score s between layer i and j is calculated based on the predicted probability, and whether the model is confident in its prediction results is evaluated, we designed three different evaluation methods, which are shown as follows:

(1) PF-BERxiT-v1: When the model reaches the current layer i, we calculate the similarity score $s_{i,i-1}$ between the prediction results of layer i and i-1. The smaller the value of $s_{i,i-1}$, the predicted distribution p_i and

Table 1
Cross-layer average scores of different fine-tuning strategies with the ALBERT backbone.

	CoLA avg-mcc	RTE avg-acc	MRPC avg-acc-f1	SST-2 avg-acc	QNLI avg-acc	MNLI avg-acc	QQP avg-acc
JOINT	41.77	64.26	84.05	88.71	86.27	79.37	86.61
ALT	43.90	67.27	83.68	90.01	86.21	79.71	88.43
2ST	32.03	66.96	77.78	87.51	82.72	67.09	81.64
BERxiT	**44.73**	66.61	84.88	89.00	86.14	79.16	88.48
PF (orig)	44.08	66.61	85.14	89.74	86.33	79.17	88.96
PF (conv)	43.16	**69.40**	84.64	89.57	86.33	79.10	88.96
PF (attn)	42.74	64.83	**85.24**	88.45	**86.49**	79.17	**89.08**
PF (mlp)	44.03	66.37	85.09	**90.19**	86.42	**79.82**	88.98

p_{i-1} are more consistent with each other, and the model is more confident in the prediction results. In addition, τ is denoted as the similarity score threshold, and cnt_i is utilized to store the number of times that the similarity scores consecutively less than the threshold:

$$cnt_i = \begin{cases} cnt_{i-1} + 1 s_{i,i-1} < \tau \\ 0 s_{i,i-1} \geq \tau \end{cases} \tag{5}$$

If the similarity score $s_{i,i-1}$ is less than the pre-defined threshold τ, increase the patience counter by 1. Otherwise, reset the patience counter to 0. This process is repeated until *cnt* reaches the pre-defined patience value. The model will stop inference and exit early. However, if this condition is never met, the model will use the final classifier to make predictions.

(2) PF-BERxiT-v2: When the model reaches the current layer *i*, we calculate the similarity score between layer *i* and its previous layers i-1…, i-m. If the similarity score *s* consecutively less than the threshold τ, increase the patience counter by 1. Otherwise, reset it to 0. This process is repeated until *cnt* reaches the pre-defined patience value. The model will stop inference and exit early. Otherwise, the model will use the final classifier to make predictions.

(3) PF-BERxiT-v3: When the model reaches the current layer *i*, we calculate the similarity score between layer *i* and its previous layers i-1, … i-m. Suppose the number of times the similarity scores *E* between the current layer and previous layers less than the threshold reaches the pre-defined patience τ. In that case, the model will stop inference and exit early. Otherwise, the model will use the final classifier to make predictions. PF-BERxiT-v3 differs from PF-BERxiT-v2 in that PF-BERxiT-v2 requires the similarity scores of the current layer and previous layers are continuously less than the threshold, while PF-BERxiT-v3 does not.

Among them, PF-BERxiT-v1 performs better than other early exiting strategies, therefore, we present it in Fig. 1(a).

Under the framework of PF-BERxiT, we adopt three different distillation-object-based similarity score measuring methods, shown below:

(1) This version of PF-BERxiT adopts the distillation object from probability distribution p_i of *i* layer to p_j of *j* layer:

$$E(p_i, p_j) = -\sum_{k=1}^{K} p_i(k) \log(p_j(k)) \tag{6}$$

(2) PF-BERxiT-rekl: This version of PF-BERxiT also adopts the distillation object, but in the reverse direction, from probability distribution p_j of *j* layer to p_i of *i* layer:

$$E(p_j, p_i) = -\sum_{k=1}^{K} p_j(k) \log(p_i(k)) \tag{7}$$

(3) PF-BERxiT-bikl: Note that the distillation object is usually asymmetrical so that we can evaluate the performance of symmetrical entropy:

$$SymE(p_j, p_i) = E(p_i, p_j) + E(p_j, p_i) \tag{8}$$

Our flexible early exiting can solve the occasional exits before reaching a steady state of the confidence-based early exiting methods and the too strict exiting criteria problem of patience-based early exiting methods, attaining a better balance between the inference speed and performance.

4. Experiments

4.1. Datasets

We evaluated the performance of our PF-BERxiT on the GLUE benchmark [32], including (1) Natural language inference benchmarks (MNLI, RTE, QNLI), which predict whether the sentence entails, contradicts, or is neutral [46]. (2) Sentiment classification benchmark predicts a sentence's sentiment (SST-2) [47]. (3) Paraphrase detection benchmark (MRPC, QQP), and (4) Linguistic acceptability benchmark (CoLA).

4.2. Baselines

We mainly choose the ALBERT-base-v2 [7] as the backbone model for baselines. For the performance comparison in the fine-tuning process, we compare our PF-BERxiT with four different groups of baselines:

(1) **JOINT:** all model parameters are trained in the fine-tuning process [2].
(2) **2ST:** Two-stage training, fine-tuning the backbone and final exit in the first stage and fine-tuning the intermediate exits on the fixed backbone in the second stage [17].
(3) **ALT:** Alternating training, alternating between joint training and two-stage training [18].
(4) **BERxiT:** The fine-tuning strategy is alternating training, and the intermediate layer parameters are tuned simultaneously to ensure an efficient exit [18].

For the performance comparison in the inference process, we compare PF-BERxiT with four different baselines:

(1) Appointed early exiting: The inference exit layers of the model are appointed. We compare our PF-BERxiT with appointed early exiting layers: Appointed-Exit-3L, Appointed-Exit-6L, and Appointed-Exit-9L.
(2) Confidence-based early exiting: We compare our PF-BERxiT with confidence-based early exiting methods, including BranchyNet [39], Shallow-deep [10], and BERxiT [18].
(3) Patience-based early exiting: We compare our PF-BERxiT with patience-based early exiting methods PABEE [25].
(4) Confidence-window-based early exit: We compare our PF-BERxiT with confidence-window-based early exiting methods ELBERT-12L [53].

4.3. Training and inference details

A linear classification layer is attached to the intermediate layer of the backbone model. We fine-tune the backbone with efficient

Table 2
Fine-tuned parameters and accuracy of different models on the RTE task.

Method	All Parameters	Fine-tuned Parameters	Cross-layer Accuracy
ALBERT	11,683,584	11,683,584	64.26
PABEE	11,702,040	11,702,040	65.31
Branchy Net	11,702,040	11,702,040	63.21
BERxiT	11,711,268	11,711,268	66.61
PF- BERxiT (orig)	12,329,124	617,856	66.61
PF- BERxiT (conv)	12,342,564	631,296	**69.40**
PF- BERxiT (attn)	12,380,580	669,312	64.83
PF- BERxiT (mlp)	12,341,796	630,528	66.37

parameter architectures. The bottleneck dimension r is 32. In addition, we fine-tune the model with the Adam optimization and warm-up. Moreover, we perform a grid search over the batch size of {16, 32, 64, 128}, and the learning rate of {1e-5, 2e-5, 3e-5, 5e-5}. And the best checkpoints in the training process are selected to be used in the inference process. All the experiments are conducted on two Nvidia TITANX24 GB GPUs. The inference process is based on the best checkpoints in the fine-tuning process. The inference of PF-BERxiT is on a per-instance basis, and the batch size is set to 1. In addition, we present the median speed-accuracy performance of the model over five runs with different random seeds.

5. Overall comparisons

5.1. Comparisons of different fine-tuning strategies

We compare the performance between PF-BERxiT and other previous SOTA fine-tuning strategies with the ALBERT backbone. Table.1 shows the cross-layer average scores of different fine-tuning strategies on seven tasks of the GLUE benchmark. It can be seen that with the help of adapter modules, PF-BERxiT outperforms the previous SOTA fine-tuning strategies by a large margin. In addition, the performances of PF-BERxiT (conv-adapter), PF-BERxiT(self-attn-adapter), and PF-BERxiT(mlp-adapter) are better than PF-BERxiT(origin-adapter), indicating the integrated sub-learning module can help learn the hidden features of the samples. In addition, the fine-tuned parameters and cross-layer accuracy are compared between different fine-tuning and early-exit models (take the RTE task as an example), as shown in Table 2. All parameters on the RTE task of PF-BERxiT (orig), PF-BERxiT (conv), PF-BERxiT (attn), and PF-BERxiT (mlp) are 12329124, 12342564, 12,380,580 and 12,341,796 respectively. And the fine-tuned parameters on the RTE task of PF-BERxiT (orig), PF-BERxiT (conv), PF-BERxiT (attn), and PF-BERxiT (mlp) are 617856, 631296, 669,312 and 630,528 respectively, which are 5.0%, 5.1%, 5.4% and 5.2% parameters of the joint training model respectively. Because in the fine-tuning process, the parameters of the backbone are frozen, and only the parameters of the adapters are fine-tuned. Therefore, the fine-tuned parameters of PF-BERxiT are much smaller than the joint training model. At the same time, the cross-layer accuracy of PF-BERxiT is higher than other fine-tuning methods, indicating that our model can attain strong performance by fine-tuning fewer parameters, making a better trade-off between accuracy and efficiency. In addition, PF-BERxiT only increases fewer than 19 K new parameters for judging whether the model needs to exit, which is about 0.15% of the original parameter, which can be ignored. Moreover, compared to the previous early-exit methods, the fine-tuned parameters of confidence-based early exit method BERxiT and patience-based exit method (PABEE) are 11,711,268 and 11,702,040 respectively, which is far more than the parameters of PF-BERxiT. However, PF-BERxiT achieves the highest cross-layer accuracy, indicating that PF-BERxiT makes better trade-offs between performance and effectiveness. Furthermore, it can find that for the RTE task, PF-BERxiT (conv) attains the highest accuracy by fine-tuning 5.1% parameters of the joint training model.

The layers-score curves with different fine-tuning strategies on the GLUE benchmark are shown in Fig. 2. It is obvious that PF-BERxiT outperforms other fine-tuning strategies by a large margin, especially in the shallow layers, which helps the model exit earlier in the inference process. All the results above indicate that PF-BERxiT outperforms

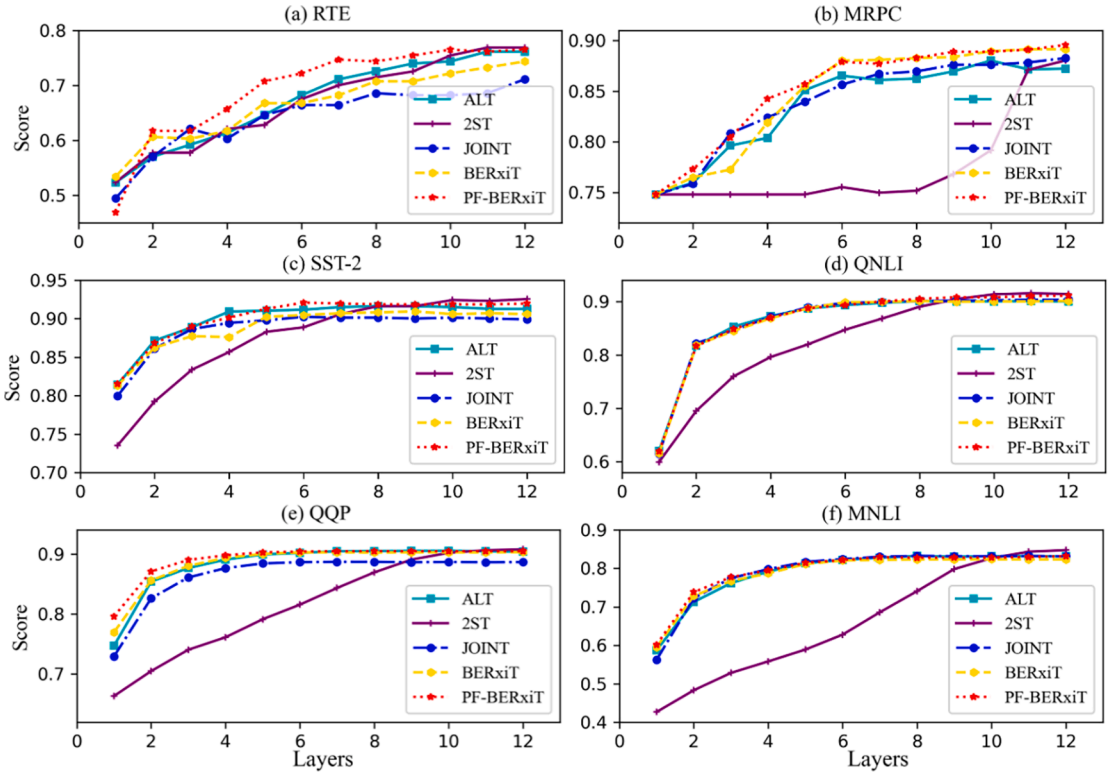


Fig. 2. The layers-score curves of different fine-tuning strategies on GLUE benchmark with ALBERT backbone.

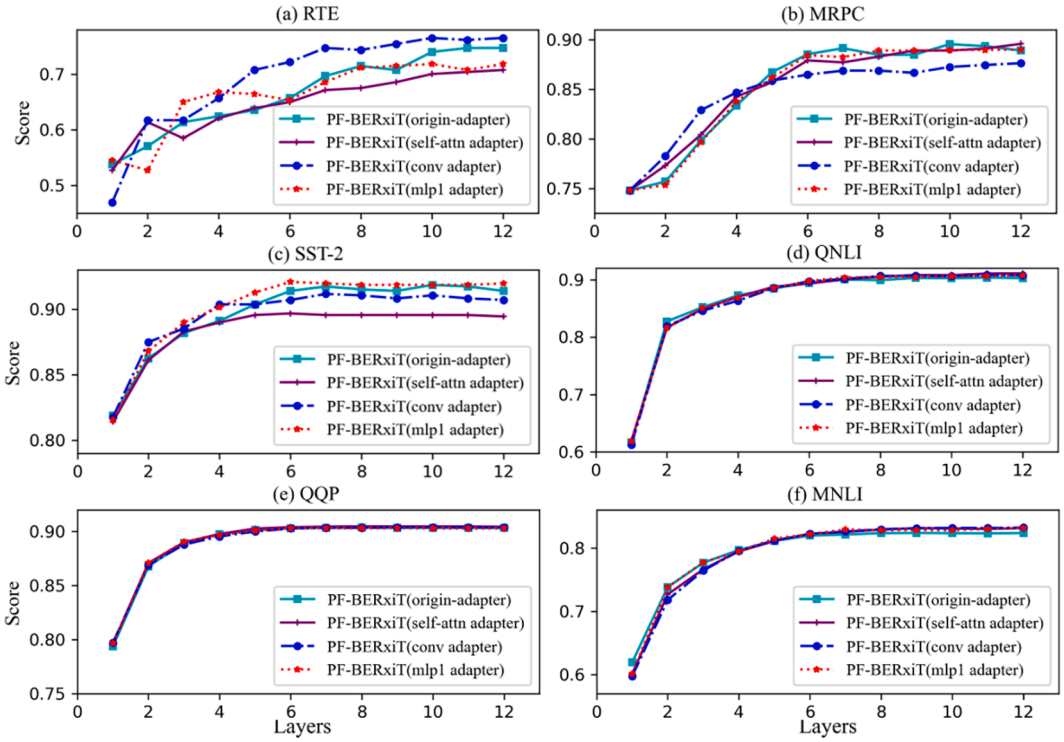


Fig. 3. Layer-score curves of different sub-learning modules on GLUE benchmark with ALBERT backbone.

conventional training strategies such as 2ST, JOINT, ALT, and conventional parameter-efficient methods with origin adapters by fine-tuning only a few parameters. Our PF-BERxiT can achieve competitive performance on the GLUE benchmark by tuning fewer parameters of the previous SOTA training strategies. In addition, the integrated sub-learning modules help learn the hidden characteristics and further improve the average accuracy of the model (Fig. 3).

5.2. Ablation studies on different sub-learning modules

We further investigate the performance of different sub-learning modules to improve the performance of PF-BERxiT. As demonstrated before, Table 1 also presents the average scores of PF-BERxiT with different sub-learning modules integrated into the adapters. It is evident that PF-BERxiT(self-attn-adapter) performs best for MRPC, QNLI, and QQP tasks, PF-BERxiT(mlp-adapter) performs best for SST-2 and MNLI tasks, and PF-BERxiT(conv-adapter) performs best for RTE tasks. Interestingly, different sub-modules are suitable for different tasks. It may be because the neural units of the convolutional neural network are locally connected. That is, the neural units of each layer are only connected with some of the neural units of the previous layer. Each neural unit only responds to the region within the receptive field and does not care about the region outside the receptive field. Therefore, it has the strongest response and accuracy for simple tasks (RTE) that emphasize local patterns. However, it performs poorer than a fully connected network when faced with complex tasks that require learning the correlation properties of data in different locations. Because a fully connected network comprises many neurons connected to other neurons, each connection is evaluated by a weight coefficient that reflects the importance of a given link in the neural network. Therefore, compared with the convolutional neural network, it can learn the correlation characteristics of data at different locations. Therefore, it performs better in more complex tasks like SST-2 and MNLI. However, compared with the fully connected network, the attentional mechanism module can aggregate the limited local distribution information into the global distribution of the whole space, which can better learn the global information. It performs well in complex and simple tasks, performing best in most tasks, such as MRPC, QNLI, and QQP.

Table 3
Experimental results of different bottleneck dimensions with the ALBERT backbone on the RTE, MRPC, and SST-2 tasks.

	2	4	8	16	32	64
RTE (avg-acc)	67.09	65.85	68.92	67.51	**69.31**	68.38
MRPC (avg-acc-f1)	85.21	85.35	85.22	84.37	**85.24**	84.01
SST-2(avg-acc)	89.61	89.36	89.83	89.57	**90.19**	89.56

5.3. Ablation studies on different bottleneck dimensions

The bottleneck dimension r has a significant influence on the performance of the model. Therefore, we conduct ablation studies to investigate the model's performances under different bottleneck dimensions r on the RTE, MRPC, and SST-2 tasks. The results are shown in Table 3, we can see that smaller bottleneck dimensions do not result in significant performance drops. Our model can achieve good performance even though the dimension of the bottleneck is small, demonstrating that the adapters help attain strong performance with fewer fine-tuned parameters. In addition, the model performs best when the bottleneck dimension is 32. Therefore, we set the bottleneck dimensions r as 32 for further ablation studies and inference processes.

5.4. Ablation studies on placements of the adapters

The placed position of the adapters also affects the model's performance. Therefore, we investigate three different placements of the adapters: (a) The adapters are placed parallel to the multi-head attention module, denoted as Attn-adapter; (b) The adapters are placed parallel to the feed-forward module, denoted as FFN-adapter in this paper. (c) The adapters are inserted between the multi-head attention layer and the Add&Norm layer, denoted as sequence-adapter in this paper (Seq-adapter). The experiment results of different adapter positions on RTE,

Table 4
Experimental results of different adapter placements on RTE, MRPC, and SST-2 tasks with the ALBERT backbone.

	RTE (avg-acc)	MRPC (avg-acc-f1)	SST-2 (avg-acc)
Attn-adapter	69.31	**85.96**	89.29
FFN-adapter	**69.40**	85.24	**90.19**
Seq-adapter	68.30	84.12	87.95

MRPC, and SST-2 tasks are shown in Table 4. It shows that the parallel adapter performs better than the sequencing adapter. Based on the superior results of parallel adapters over sequential adapters, we focus on parallel adapter results in the following sections. In addition, among parallel adapters, it is evident that FFN-adapter performs better on the RTE and SST-2 tasks, and Attn-adapter performs better on the MRPC task. The results show that the FFN-adapter performs better than the Attn-adapter, indicating that the FFN-adapter can utilize the fine-tuned parameters more effectively than the Attn-adapter. We suppose it may

Table.5
scores and speedup of different early exiting methods with the same fine-tuned ALBERT backbone on the GLUE benchmark. The results show that PF-BERxiT effectively accelerates ALBERT's inference speed with less performance loss compared with the baseline methods.

	CoLA		MNLI		MRPC		QNLI		QQP		RTE		SST-2	
	Score	speed	Score	speed	Score	speed	Score	speed	Score	speed	Score	speed	Score	speed
Backbone	54.2	0%	83.1	0%	86.8	0%	89.8	0%	89.2	0%	69.1	0%	91.3	0%
Exit-3L	0.0	75%	70.0	75%	75.8	75%	77.4	75%	81.8	75%	54.7	75%	81.0	75%
Exit-6L	0.0	50%	79.6	50%	84.7	50%	85.3	50%	89.3	50%	68.1	50%	88.6	50%
Exit-9L	51.9	25%	83.0	25%	87.0	25%	88.4	25%	90.3	25%	69.0	25%	91.2	25%
Branchy	0.0	74%	63.8	76%	75.7	76%	74.2	80%	71.6	80%	54.7	76%	79.9	76%
Net	0.0	51%	78.3	53%	83.0	52%	87.1	47%	89.3	50%	67.2	48%	88.3	49%
	52.1	27%	83.0	25%	85.8	24%	89.3	27%	90.1	26%	67.8	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%
ELBERT-12L	32.3	74%	79.6	73%	81.7	77%	86.5	74%	88.7	72%	67.5	75%	85.3	76%
	51.3	52%	81.7	49%	87.2	50%	90.1	53%	90.2	53%	71.8	49%	90.1	51%
	53.1	24%	83.6	27%	88.4	25%	90.5	26%	90.5	23%	72.1	25%	90.3	24%
PF-BERxiT	**40.1**	75%	**81.3**	72%	**83.8**	77%	**88.2**	75%	**90.4**	82%	**73.0**	76%	**91.1**	76%
	53.7	53%	**82.9**	53%	**89.4**	53%	**91.1**	54%	**90.5**	49%	**76.1**	47%	**92.0**	48%
	53.4	25%	**83.4**	28%	**89.6**	26%	**91.1**	28%	**90.5**	25%	**76.9**	25%	**92.0**	25%
	54.2	0%	**83.6**	0%	**89.8**	0%	**91.2**	0%	**90.5**	0%	**76.9**	0%	**92.1**	0%

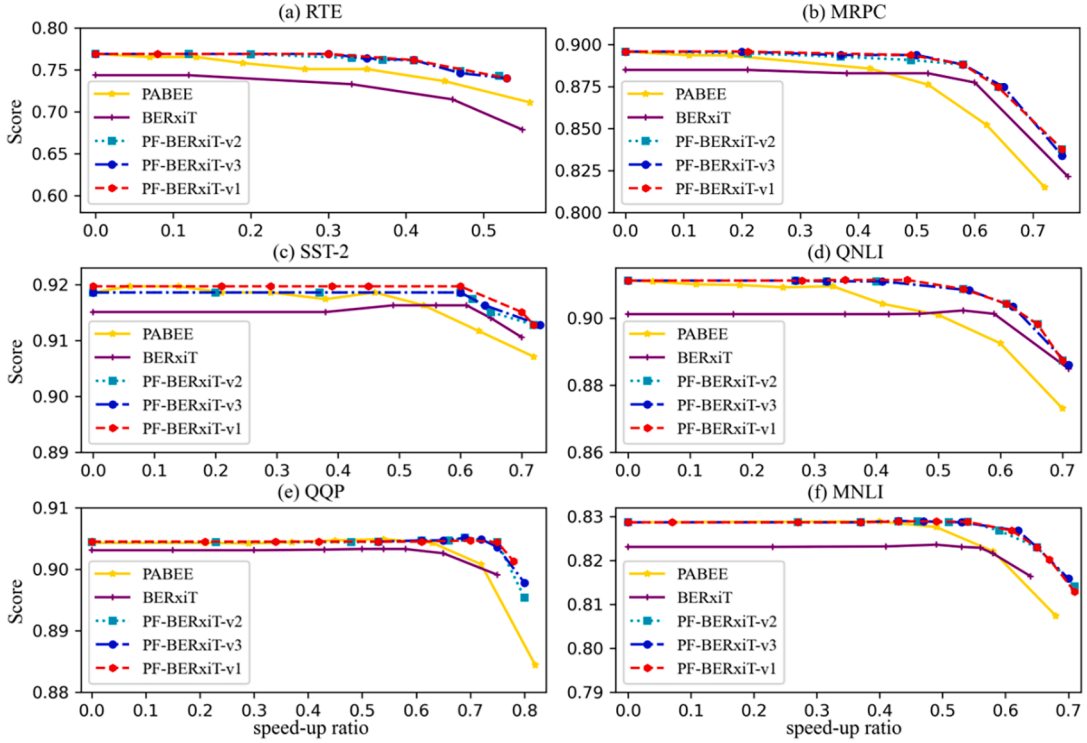


Fig. 4. Speed-accuracy curves of different early exiting strategies on GLUE benchmark with the ALBERT backbone.

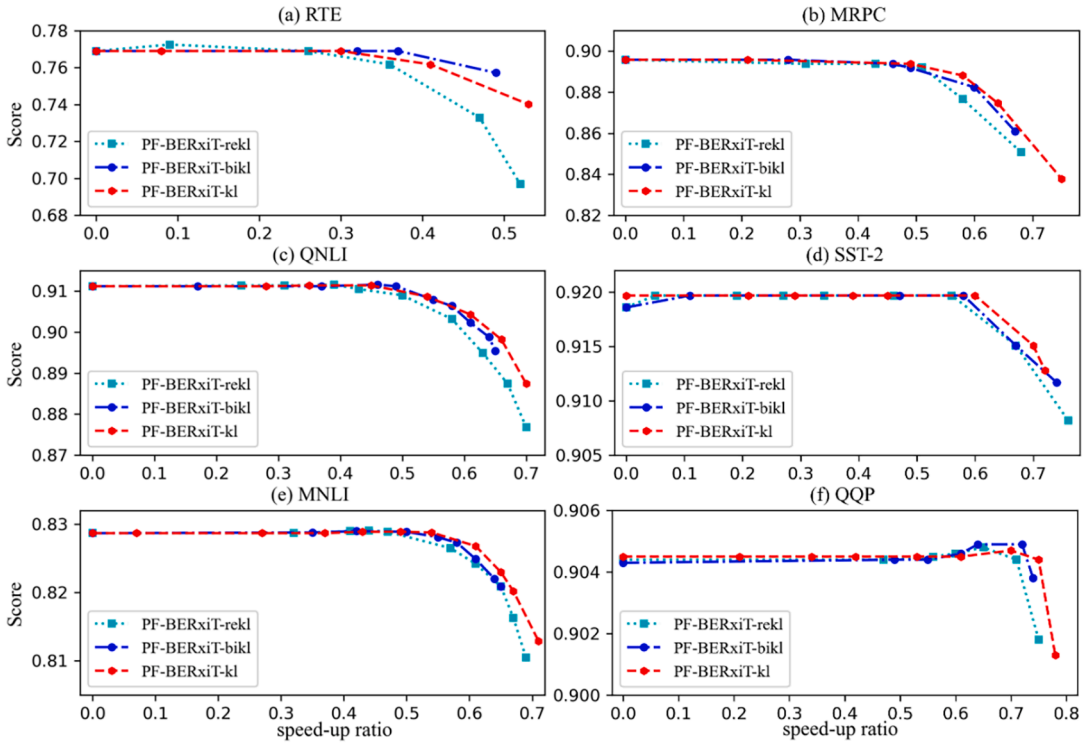


Fig. 5. Speed-accuracy curves of different similarity score calculation methods on GLUE benchmark with the ALBERT backbone.

be because the feed-forward module learns task-specific patterns, while the attention module learns the positional interactions, which is not advanced in learning new tasks [48].

5.5. Comparisons of different inference strategies

In the inference process, the best checkpoints in the fine-tuned process are selected to be used in the inference process to combine with the flexible early exiting strategy to make a trade-off between performance and efficiency. We compare the inference performance between our PF-BERxiT with the previous SOTA early exiting strategies on the GLUE benchmark of three different speedup settings: (1) 72% to 82% speedup; (2) 46% to 54% speedup; (3) 25% to 28% speedup. The previous early exiting strategies included appointed early exiting, which fixes the exit layers, including appointed-Exit-3L, appointed-Exit-6L, and appointed-Exit-9L, and confidence-based early exit methods, including Branchy Net, Shallow-deep, and BERxiT, and patience-based early exit method PABEE. In addition, ELBERT [53] model, which significantly improves the average inference speed compared to ALBERT due to the proposed confidence-window-based early exit mechanism without introducing additional parameters or extra training overhead, is also compared with PF-BERxiT. Among them, the original ELBERT model consists of 24 Transformer layers and uses ALBERT-large as the backbone, while our PF-BERxiT consists of 12 Transformer layers and uses ALBERT-base-v2 as the backbone. Therefore, to ensure a fair comparison, we compare the performances of PF-BERxiT with the variants of the ELBERT model, which consists of 12 Transformer layers, denoted as ELBERT-12L, in this paper. Moreover, since PABEE cannot flexibly adjust the speedup ratios, we adjust the hyper-parameters (similarity score threshold and patience parameter) of our PF-BERxiT and the other baselines to achieve similar speedups with PABEE.

Table 5 shows the score and speedup of different exiting strategies on the GLUE benchmark. The speed-up ratio equal to 0 means that the model goes through all layers without using a flexible early exiting strategy to speed up the inference process, compared with the model with flexible exiting strategies. It shows that PF-BERxiT with 0 speedups performs better than the backbone model, indicating that our model can perform better with fewer fine-tuned parameters. In addition, PF-BERxiT can attain a better score than other inference strategies (including static early exit methods, confidence-based early exit methods, patience-based early exit methods and confidence-window based early exit mechanism ELBERT-12L) under ~25%, ~50%, and 75% speed up respectively, indicating that PF-BERxiT is effective in accelerating ALBERT's inference speed with less performance loss compared with the baseline exiting strategies.

In addition, we plot the speedup-score curves for the patience-based early exiting method PABEE, confidence-based early exiting method BERxiT, and our PF-BERxiT, to further analyze the performance of different inference strategies. Fig. 4 shows the speedup-score curves of different inference strategies on GLUE benchmarks. The speed-up ratio equal to 0 means that the model goes through all layers without using a flexible early exiting strategy to speed up the inference process, compared with the model with flexible exiting strategies. It is clearly shown that our PF-BERxiT outperforms the baseline early exiting methods on all tasks of the GLUE benchmark. Moreover, the majority of the computation complexity is from encoding sentences with the backbone, which is quadratic for sequence length, linear for the depth of intermediate exits, and the model's hidden size and the parameters of the adapter are about 5% of the initial model. Therefore, the adapter's impact on the computational complexity is negligible. Furthermore, we count the inference latency of the backbone and our PF-BERxiT, and we compare the inference time between PF-BERxiT and the original model. The results indicate that our method can significantly alleviate the model's inference latency. For example, the inference latency is 0.71x of the backbone for the RTE task, the inference latency is 0.53x of the backbone for the MRPC task, the inference latency is 0.40x of the backbone for the SST-2 task, the inference latency is 0.45x of backbone for QNLI task, the inference latency is 0.39x of backbone for QQP task, and the inference latency is 0.48x of backbone for MNLI task. We suppose that PF-BERxiT can overcome the performance drop of confidence-based early exiting strategies for the occasional exit before it reaches the steady state and can overcome the inefficiency problem of patience-

Table 6
Scores and speedup of different batch sizes with the same fine-tuned ALBERT backbone on the GLUE benchmark.

Batch size	MNLI		MRPC		QNLI		QQP		RTE		SST-2	
	Score	speed	Score	speed	Score	speed	Score	speed	Score	speed	Score	speed
Base	83.1	0%	86.8	0%	89.8	0%	89.2	0%	69.1	0%	91.3	0%
1	81.3	72%	83.8	72%	91.1	75%	90.4	82%	73.0	76%	91.1	76%
	82.9	53%	89.4	53%	91.1	54%	90.5	49%	76.1	47%	92.0	48%
	83.4	28%	89.6	26%	91.1	28%	90.5	25%	76.9	25%	92.0	25%
2	78.9	72%	83.3	72%	86.2	72%	89.4	75%	72.9	73%	91.2	72%
	82.8	50%	88.9	54%	90.8	49%	90.5	48%	72.2	49%	91.5	48%
	83.2	25%	89.4	26%	91.2	26%	90.5	24%	75.8	25%	91.7	26%
4	77.1	73%	84.3	72%	85.8	73%	88.3	73%	72.0	74%	91.1	75%
	81.7	51%	88.3	50%	90.2	50%	90.5	47%	72.9	48%	91.7	54%
	83.2	25%	89.4	25%	91.2	26%	90.5	25%	75.5	24%	91.7	25%
8	76.2	73%	82.8	73%	84.8	73%	87.7	72%	71.2	73%	90.8	73%
	81.8	50%	87.2	52%	89.8	49%	90.1	52%	72.2	49%	91.7	53%
	83.2	24%	89.3	25%	91.0	28%	90.5	25%	75.5	25%	91.9	25%
16	76.0	73%	81.3	72%	84.3	72%	87.4	73%	70.2	74%	90.3	74%
	81.5	50%	87.4	48%	89.7	52%	90.1	52%	72.9	48%	92.0	50%
	82.9	25%	89.3	24%	90.8	25%	90.5	25%	74.7	28%	92.0	25%

based early exiting strategies for too strict exiting criteria for the reason that it can flexibly adjust the similarity score threshold and patience threshold. Our PF-BERxiT can better balance the performance and speed up, alleviating the overthinking problem of huge pre-trained language models without any performance loss.

5.6. Ablation studies on different similarity score calculation methods

We adopt three different relative entropy-based similarity score measuring methods of PF-BERxiT in this paper, denoted as PF-BERxiT-kl, PF-BERxiT-rekl, and PF-BERxiT-bikl. The speedup-score curves of PF-BERxiT-kl, PF-BERxiT-rekl, and PF-BERxiT-bikl on GLUE benchmark are shown in Fig. 5. It can be seen that the PF-BERxiT-kl and PF-BERxiT-bikl attain better speedup-accuracy trade-offs than PF-BERxiT-rekl. We suspect it may be because the PF-BERxiT-rekl is asymmetric, not an accurate distance measurement. In addition, compared to PF-BERxiT-kl, it has a wide range, which increases the difficulty of tuning the hyper-parameters. PF-BERxiT-bikl is a symmetric divergence, and the similarity discrimination is more accurate than asymmetric divergence. Moreover, its divergence range is between 0 and 1, making hyper-parameters easier to tune. Therefore, PF-BERxiT-bikl and kl perform better.

5.7. Ablation studies on different batch sizes

In the previous inference process, we present the performance of the model inference process when the batch size is 1. However, the batch size may be greater than 1 in the actual industrial scenario. Therefore, we conduct ablation experiments to explore the inference performance of PF-BERxiT when batch size is greater than 1. We explore the inference performance of the model under different speedup settings (72% to 82% speedup, 46% to 54% speedup, 25% to 28% speedup) on the GLUE benchmark when the batch size was 1, 2, 4, 8, 16, respectively. The results are shown in Table.6. It can be seen that even under different batch sizes, the inference accuracy of PF-BERxiT is very close, with almost no difference when the acceleration ratio is ~25%, ~50% and ~75%, respectively. This is because we can dynamically adjust the model's similarity score threshold and patience parameters based on the complexity of the data, making it easier for the model to find the optimal exit. The above results indicate that our PF-BERxiT can also guarantee excellent inference acceleration performance in practical acceleration scenarios.

6. Conclusion

To mitigate the redundant computational problems in the fine-tuning process, we propose PF-BERxiT, which optimizes the pre-trained language model in the fine-tuning and inference process. Significantly, the parameter-efficient fine-tuning method integrates a sub-learning module to the adapters to learn the characteristics of the samples, and the flexible exiting strategy makes the model exit early if the times that the similarity score of cross-layers is less than the threshold reaches the predefined patience. Extensive experiments are conducted on the GLUE benchmark, demonstrating that:

(1) PF-BERxiT outperforms conventional training strategies such as 2ST, JOINT, ALT, and conventional parameter-efficient methods with origin adapters by fine-tuning only a few parameters. Our model can achieve good performance even though the dimension of the bottleneck is small, demonstrating that the adapters with sub-learning module help attain strong performance with fewer fine-tuned parameters.

(2) PF-BERxiT strikes a better balance between model performances and speedup ratios than previous state-of-the-art (SOTA) early exiting methods such as PABEE and BERxiT because it can dynamically adjust the similarity score thresholds and patience parameters according to the request traffics.

(3) Ablation studies in the fine-tuning process demonstrate that the best bottleneck dimension r of the adapters is 32. The adapters placed parallel to the feed-forward module are more efficient. In addition, ablation studies in the inference process demonstrate that PF-BERxiT-kl and PF-BERxiT-bikl attain better speedup-accuracy trade-offs than PF-BERxiT-rekl.

In summary, PF-BERxiT helps attain a better trade-off between accuracy and efficiency, providing a reference for the efficient application of artificial intelligence.

CRediT authorship contribution statement

Xiangxiang Gao: Writing – original draft, Methodology, Investigation. **Yue Liu:** Formal analysis, Data curation. **Tao Huang:** Validation, Investigation. **Zhongyu Hou:** Writing – review & editing, Supervision.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that has been used is confidential.

Acknowledgments

This work was financially supported by the National Natural Science Foundation of China (60906053, 61204069), Shanghai Science Innovation Project (15DZ1160800), CASIC Frontier Technology Topic (2022-QY-12), Fundamental Research Funds of non-profit Central Institutes (No. ZX2230).

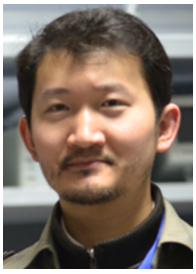
References

[1] M.E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer, Deep contextualized word representations, CoRR abs/1802.05365. arXiv: 1802.05365.
[2] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, CoRR abs/1810.04805. arXiv: 1810.04805.
[3] S. Kim, A. Gholami, Z. Yao, M. W. Mahoney, K. Keutzer, I-BERT: integer-only BERT quantization, arXiv abs/2101.01321. arXiv:2101.01321.
[4] M. Lewis, et al. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, CoRR abs/1910.13461. arXiv:1910.13461.
[5] W. Fedus, B. Zoph, N. Shazeer, Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, CoRR abs/2101.03961. arXiv: 2101.03961.
[6] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized BERT pretraining approach, arXiv:1907.11692.
[7] Z. Lan, et al, ALBERT: A lite BERT for self-supervised learning of language represen tations, arXiv:1909.11942.
[8] A. Radford, et al, Language models are unsupervised multi task learners, 2019.
[9] V. Sanh, et al, Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter, arXiv:1910.01108.
[10] Y. Kaya, T. Dumitras, How to stop off-the shelf deep neural networks from overthinking, arXiv:1810.07052.
[11] D. Jin, Z. Jin, J. T. Zhou, P. Szolovits, Is BERT really robust? natural language attack on text classification and entailment, arXiv:1907.11932.
[12] P. Michel, et al. Are sixteen heads really better than one? arXivabs/1905.10650.
[13] S. Sun, et al. Patient knowledge distillation for BERT model compression, arXiv: 1908.09355.
[14] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, Q. Liu, Tinybert: Distilling BERT for natural language understanding, arXiv:1909.10351.
[15] A. Fan, E. Grave, A. Joulin, Reducing transformer depth on demand with structured dropout, arXivabs/1909.11556.
[16] L. Hou, L. Shang, X. Jiang, Q. Liu, Dynabert: Dynamic BERT with adaptive width and depth,arXivabs/2004.04037.
[17] W. Liu, P. Zhou, Z. Zhao, Z. Wang, H. Deng, Q. Ju, Fastbert: a self-distilling BERT with adaptive inference time, arXiv:2004.02178.
[18] J.Xin, R.Tang, Y.Yu, J.J.Lin, Berxit: Early exiting for bert with better fine-tuning and extension to regression, in:EACL, 2021.
[19] J. He, et al, Towards a unified view of parameter-efficient transfer learning, arXiv: 2110. 04366.
[20] N. Morgan, et al., Generalization and Parameter Estimation in Feedforward Nets: Some Experiments, in: Proceedings of the 2nd International Conference on Neural Information Processing Systems, NIPS'89, MIT Press, Cambridge, MA, USA, 1989, pp. 630–637.
[21] L. Prechelt, Early stopping but when? Neural Networks: Tricks of the Trade, Vol.1524 of Lecture Notes in Computer Science, Springer, 1996, pp. 55–69.
[22] R. Schwartz, G. Stanovsky, S. Swayamdipta, J. Dodge, N. A. Smith, The right tool for the job: Matching model and instance complexities, arXiv:2004.07453.
[23] J. Xin, R. Tang, J. Lee, Y. Yu, J. Lin, Deebert: Dynamic early exiting for accelerating BERT inference, CoRR abs/2004.12993. arXiv:2004.12993.
[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, CoRR abs/1706.03762. arXiv:1706.03762.
[25] W. Zhou, et al, BERT loses patience: Fast and robust inference with early exit, arXiv:2006.04152.
[26] X. Liu, et al, GPT understands, too, arXiv:2103.10385.
[27] Y. Li, Y. Liang, A. Risteski, Recovery guarantee of weighted low-rank approximation via alternating minimization, arXiv:1602.02262.
[28] S. Oymak, Z. Fabian, M. Li, M. Soltanolkotabi, Generalization guarantees for neural networks via harnessing the low-rank structure of the jacobian, arXiv:1906.05392.
[29] Z. Allen-Zhu, et al, A convergence theory for deep learning via over-parameterization, arXiv:1811.03962.
[30] E.B. Zaken, et al, Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language models, arXiv:2106.10199.
[31] R. K. Mahabadi, J. Henderson, S. Ruder, Compacter: Efficient low-rank hypercomplex adapter layers, arXiv:2106.04647.
[32] A. Wang, et al, GLUE: A multi-task benchmark and analysis platform for natural language understanding, arXiv:1804.07461.
[33] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, S. Gelly, Parameter-efficient transfer learning for NLP, arXiv:1902.00751.
[34] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, W. Chen, Lora: Low-rank adaptation of large language models, arXiv:2106.09685.
[35] Y. Zhu, J. Feng, C. Zhao, M. Wang, L. Li, Serial or parallel? plug able adapter for multilingual machine translation, arXiv:2104.08154.
[36] X.L. Li, P. Liang, Prefix-tuning: Optimizing continuous prompts for generation, arXiv:2101.00190.
[37] J. Pfeiffer, A. Kamath, A. Ru¨ckle´, K. Cho, I. Gurevych, Adapter fusion: Non-destructive task composition for transfer learning, arXiv:2005.00247.
[38] M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, L. Kaiser, Universal transformers, arXivabs/1807.03819.
[39] S.Teerapittayanon, B.McDanel, H.Kung, Branchynet: Fast inference via early exiting from deep neural networks, in: 2016, 23rd International Conference on Pattern Recognition (ICPR), 2016, pp. 2464–2469. doi:10.1109/ICPR.2016.7900006.
[40] G.E.H.JimmyLeiBa, Jamie RyanKiros, Layer normalization, CoRRabs/1607.06450.
[41] K. Fukushima, Cognitron: A self-organizing multilayered neural network, Biol. Cybernetics 20 (3-4) (1975) 121–136.
[42] D. Hendrycks, K. Gimpel, Bridging nonlinearities and stochastic regularizers with gaussian error linear units, arXiv:1606.08415.
[43] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[44] Y.Kim, Convolutional neural networks for sentence classification, arXiv: 1408.5882.
[45] D. Svozil, V. Kvasnicka, J. Pospichal, Introduction to multi-layer feed forward neural networks, Chemometr. Intelligent Labor. Syst. 39 (1) (1997) 43–62.
[46] A. Williams, N. Nangia, S. R. Bowman, A broad-coverage challenge corpus for sentence understanding through inference, arXiv:1704.05426.
[47] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. Manning, A. Ng, C. Potts, Recursive deep models for semantic compositionality over a sentiment treebank, EMNLP 1631 (2013) 1631–1642.
[48] M. Geva, R. Schuster, J. Berant, O. Levy, Transformer feed-forward layers are key-value memories, arXiv:2012.14913.
[49] Dongze Lian, Daquan Zhou, Jiashi Feng, Xinchao Wang, Scaling & Shifting Your Features: A New Baseline for Efficient Model Tuning, NeurIPS2022, arXiv: 2210.08823.
[50] Shengding Hu, Zhen Zhang, Ning Ding, Yadao Wang, Yasheng Wang, Zhiyuan Liu, Maosong Sun, sparse structure search for parameter-efficient tuning, NeurIPS2033, arXiv:2206.07382.
[51] Yi-Lin Sung, Jaemin Cho, Mohit Bansal. LST: Ladder Side-Tuning for Parameter and Memory Efficient Transfer Learning. NeurIPS2022, arXiv:2206.06522.
[52] Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, Xipeng Qiu, Black-Box Tuning for Language-model-as-a-service. ICML2022, arXiv:2201.03514.
[53] Xie, Keli, et al. Elbert: Fast albert with confidence-window based early exit. ICASSP 2021-2021. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2021.

XiangXiang Gao received a B.S. degree in Metal Materials and Engineering from the school of mechanical and electrical engineering, HoHai University, China, in 2018. She is currently pursuing a Ph.D. degree in electronic science and technology at Shanghai Jiao Tong University, China. Her research interest includes the engineering applications of artificial intelligence.

Yue Liu received her B.S. degree from the Dalian University of Technology in 2008. She received her Ph.D. degree from Shanghai Jiao Tong University in 2013. From March 2014 to December 2015, Dr. Liu worked as a junior researcher with the European laser research project (ELI). In January 2016, she joined the Department of Micro/Nano-Electronics of Shanghai Jiao Tong University as an assistant professor. Her research directions include laser-plasma interaction, electron and proton accelerations, radiation source generation, etc. Her accomplishments have been published in many academic journals, such as Physical Review Letter and New Journal of Physics, as well as some international conferences.

Tao Huang received a B.S. degree in material forming and control engineering from the Beijing Institute of Technology. He is currently pursuing a Master's degree in Integrated Circuit Engineering at Shanghai Jiao Tong University. His research interest includes the design of neural network accelerators.



Zhongyu Hou received a Ph.D. in electronic science and engineering from Shanghai Jiao Tong University. He is currently a Professor at Shanghai Jiao Tong University. His research interests include plasma science and technology in the application context of microwave technology, aerodynamics, environment, and MEMS-based sensors and actuators. He is also working on the development of a new research methodology incorporating artificial intelligent techniques.