Early Exit with Disentangled Representation and Equiangular Tight Frame

Yixin Ji\textsuperscript{1}, Jikai Wang\textsuperscript{1}, Juntao Li\textsuperscript{1}, Qiang Chen\textsuperscript{2}, Wenliang Chen\textsuperscript{1}, Min Zhang\textsuperscript{1}
\textsuperscript{1}Institute of Computer Science and Technology, Soochow University, China
\textsuperscript{2}Alibaba Group
\{jiyixin169, risus254\}@gmail.com;
lapu.cq@alibaba-inc.com;
\{ljt, wlchen, minzhang\}@suda.edu.cn

Abstract

Dynamic early exit has demonstrated great potential in coping with the sharply increasing number of pre-trained language model parameters, which can achieve a good trade-off between performance and efficiency. The existing early exit paradigm relies on training parametrical internal classifiers at each intermediate layer to complete specific tasks. Based on the predictions of these internal classifiers, different methods are designed to decide when to exit. Under this circumstance, each intermediate layer takes on both generic language representation learning and task-specific feature extraction, which makes each intermediate layer struggle to balance two types of backward loss signals during training. To break this dilemma, we propose an adapter method to decouple the two distinct types of representation and further introduce a non-parametric simplex equiangular tight frame classifier (ETF) for improvement. Extensive experiments on monolingual and multilingual tasks demonstrate that our method gains significant improvements over strong PLM backbones and early exit methods.

1 Introduction

In recent years, fundamental models that rely on the scaling effect have penetrated different NLP scenarios (Radford et al., 2018; Devlin et al., 2019; Liu et al., 2019; Lan et al., 2019; Clark et al., 2020; Lewis et al., 2020; Raffel et al., 2020; Brown et al., 2020; He et al., 2021a; OpenAI, 2022). However, with the increasing number of the pre-trained model parameters, the expensive inference cost hinders their usage in practical applications. Besides, Overthinking problem (Kaya et al., 2019) also restricts the ability of PLMs. Specifically, since PLMs are overparameterized, they can give correct answers according to the shallow representations at earlier layers, while the high-level representation may instead contain too much over-complicated or irrelevant information to make the accurate prediction (Xin et al., 2020b; Liu et al., 2020).

To achieve a good trade-off between inference cost and performance, the early exit mechanism (Xin et al., 2020a; Zhou et al., 2020; Li et al., 2021b; He et al., 2021c; Xin et al., 2021; Banino et al., 2021; Balagansky and Gavrilov, 2022; Sun et al., 2022), a kind of adaptive inference strategy, has been proposed. These methods insert an internal classifier after each layer of the PLMs to predict the label of a given instance. In the inference stage, if the prediction is confident enough earlier, the sample will end the inference without going through the entire PLM. Nevertheless, how to train a competitive early exit PLM is not a trivial problem. Since each classifier tries to be optimized, different optimization procedures from different classifiers may conflict and interfere with each other (Phuong and Lampert, 2019). Exist-
ing methods can be divided into two categories: (1) jointly training all classifiers and using heuristics or learnable methods to weight loss functions (Zhou et al., 2020; Li et al., 2021b; Zhu, 2021; Liao et al., 2021); (2) training classifiers in different stages and freezing the Transformer layer when training internal classifiers.

However, in the early exit mechanism, each Transformer layer plays two roles: providing classifiable representation for the corresponding internal classifier and semantic features for subsequent Transformer layers. The former class of methods is obsessed with improving the classification ability of the middle layer of PLMs while ignoring its ability to capture other linguistic features. The latter class focuses on maintaining the ability to extract semantic representations while constraining the performance of internal classifiers. Although some work (Xin et al., 2021) has attempted to balance the two roles of the Transformer layer by alternating training, they have not explicitly decoupled the two distinct types of representation.

In this work, we propose a novel early exit method that enables the PLM’s intermediate representation to consider both the classification ability and the ability to capture semantic information. Specifically, our approach hands off the task-specific representation of the intermediate transformer layer to an adapter module (Houlsby et al., 2019; He et al., 2022) with a small number of parameters, thus disentangling the task-specific and universal representation (see Figure 1). In addition, due to the limited expressive power of the adapter, we use a simplex equiangular tight frame classifier (ETF) (Yang et al., 2022) to enhance the classification ability of the internal classifiers. Experimental results demonstrate that our proposed early exit method significantly improves the performance of monolingual and multilingual tasks over existing strong early exit methods.

Overall, our contributions are shown below:

• We empirically study the ability to extract generic linguistic representation in the internal layers of early exit models. The study reveals that the internal layers have difficulty taking on both generic language representation learning and task-specific feature extraction.

• We propose an early exit method which use adapter modules to disentangle the two conflicting representations and utilize equiangular tight frame classifiers to improve representations for classification.

• Experimental results and analysis on monolingual and multilingual tasks demonstrate that our proposed early exit method performs better than previous methods. Our code is available at https://github.com/Jikai0Wang/DREE.

2 Preliminary Study

It is controversial whether the last classifier and the internal classifiers should be trained jointly in early exit. Some studies (Xin et al., 2020b, 2021; Liu et al., 2020) divide the training into two stages to train the last classifier and the internal classifiers, respectively, to preserve the best model ability for the final layer. While others (Zhou et al., 2020; Balagansky and Gavrilov, 2022) train the whole model simultaneously. In this section, we study the question: “Can the same representation serve both the classifier and subsequent layers?”

Following Durrani et al. (2021), we train layer-wise probes to check how much linguistic knowledge is preserved in each layer after finetuning. We evaluate the model on two linguistic tasks: POS tagging using the Universal Dependencies v2.5 English dataset from XTREME (Hu et al., 2020) and syntactic chunking using CoNLL 2000 shared task dataset (Tjong Kim Sang and Buchholz, 2000). We conduct experiments on PABEE (Zhou et al., 2020b).
As shown in Figure 2, the red lines refer to the performance of BERT without any finetuning. The orange and blue lines represent the performance of BERT finetuned on SST-2 with and without PABE. The model preserves the information of the downstream tasks in the higher layers with a corresponding loss of the linguistic knowledge learned in the pre-training after finetuning. Moreover, the gaps between the blue and orange lines are apparent, indicating that the training with PABE further disturbs linguistic knowledge. On the other hand, a representation trained only to serve the subsequent layers often has a poor performance for classification, especially in the lower layers. As a result, to maintain the internal representations’ capability and improve the classification performance, it is essential to train disentangled representations in the intermediate layers.

3 Method

In this section, we introduce our method for early exit, which first utilizes adapter modules to disentangle generic and task-specific representation. To further improve the classification representation, we replace learnable classifiers with equiangular tight frame (ETF) classifiers. Throughout this section, we consider the case of multi-class classification with samples \( \{(x_i, y_i)\}_{i=1}^n \), where \( x_i \) is a token sequence and \( y_i \) is its label.

3.1 Disentangled Internal Representations

Existing early exit methods utilize the intermediate layers to learn both task-specific representation for task prediction and extract generic linguistic representation for the subsequent layers, making the model struggle to balance the two learning objectives. To address such a dilemma, we investigate how to improve task-specific representations of internal Transformer layers without hindering the learning of the generic representation. Inspired by the recent emergence of efficient tuning that freezing most of the parameters of PLMs and training with a small number of parameters can achieve comparable performance to full-parameter finetuning, we propose to fix the internal Transformer layers to consistently extract linguistic representation and utilize additional adapter modules to learn task-specific representation. For a given sample pair \((x, y)\), each Transformer layer \( L_i \) outputs a hidden state \( h_i \) as the input to \( L_{i+1} \):

\[
\begin{align*}
  s_i &= \text{LN}(\text{FFN}(\text{SA}(h_{i-1})) + h_{i-1}) \\
  s'_i &= \text{FFN}(\text{FFN}(s_i)) + s_i  \\
  h_i &= \text{LN}(s'_i) 
\end{align*}
\]

(1)

where SA is a self-attention sub-layer, FFN is a feed-forward sub-layer, and LN means Layernorm. At the same time, the adapter module outputs another hidden state \( h' \) for classification:

\[
  h'_i = \text{LN}((\text{Adapter}(s_i) + s'_i)) 
\]

(2)

Following Houlsby et al. (2019), the adapter module includes a stack of down- and up-scale fully connected neural network:

\[
  \text{Adapter}(s_i) = f_{up}(\text{ReLU}(f_{down}(s_i))) 
\]

(3)

where \( f_{down} \in \mathbb{R}^{d \times m} \), \( f_{up} \in \mathbb{R}^{m \times d} \) are the down and up projection layers. \( d \) is the dimension of the PLM, and \( m \) is the hidden size of the adapter. We pass \( h \) into the next transformer layer and use \( h' \) for classification. Unlike the standard adapter, to reduce the inference time overhead caused by the adapter module, we use the parallel adapter (He et al., 2022) and only add an adapter module to the feed-forward sub-layer, as shown in Figure 3. We have experimented with the sequential adapter, but its performance is inferior to the parallel adapter.
3.2 Improving Classifiable Representation

A recent study (Zhu et al., 2021; Ji et al., 2021; Tirer and Bruna, 2022; Yang et al., 2022) has shown a phenomenon called neural collapse that the within-class means of features and the classifier vectors converge to the vertices of a simplex equiangular tight frame (ETF) at the terminal phase of training (Figure 4 shows a simple example of the three-class classification). Galanti et al. (2021) demonstrate that this phenomenon shows better generalization. Now that the distribution of optimal classification representation is known theoretically, Yang et al. (2022) propose the ETF classifier, which is randomly initialized as an equiangular tight frame and free from training.

Definition 1 (ETF classifier). A simplex equiangular tight frame is a collection of vectors $w_i \in \mathbb{R}^d$, $i = 1, 2, ..., K, d \geq K - 1$, it satisfies:

$$W_{ETF} = \sqrt{\frac{K}{K-1}} U \left( I_K - \frac{1}{K} 1_K 1_K^T \right), \quad (4)$$

where $W_{ETF} = [w_1, w_2, ..., w_K] \in \mathbb{R}^{d \times K}$, $I_K \in \mathbb{R}^{K \times K}$ is the identity matrix, $1_K \in \mathbb{R}^K$ is a vector of all ones and $U \in \mathbb{R}^{K \times K}$ is orthonormal.

Due to the limited generalization of the adapter compared to full-parameter fine-tuning, we utilize the ETF classifier to induce better classifiable representation. For the hidden state $h_i^t$ from the Transformer layer $L_i$, the ETF classifier output the probability $p_i(y|x)$:

$$p_i(y|x) = \text{Softmax}(W_{ETF}h_i^t) \quad (5)$$

During the training phase, ETF classifiers are frozen, and we fine-tune the adapter parameters to align classification features with ETF classifiers, aiming to achieve neural collapse phenomenon with better generalization.

3.3 Training and Inference

Training. Following Xin et al. (2020b), we divide the training into two stages. The first stage trains the last classifier, i.e., without launching early exit, and the second stage trains internal classifiers. In the first stage, we fine-tune the whole parameters of the PLM and the last classifier with labeled data from downstream tasks without training the adapter modules and the internal classifiers. We use the cross-entropy loss for classification:

$$L_{stage1} = \text{CE}(\text{logits}, y). \quad (6)$$

We use the mean squared error for regression tasks:

$$L_{stage1} = (y - \hat{y})^2. \quad (7)$$

In the second stage, we freeze all parameters fine-tuned in the first stage and then train adapter modules and internal classifiers to enable the PLM early exit. The loss of an internal classifier is also calculated with cross-entropy for classification:

$$L_i = \text{CE}(\text{logits}_i, y). \quad (8)$$

For regression, we also use the mean squared error:

$$L_i = (y - \hat{y}_i)^2. \quad (9)$$

The total loss $L_{stage2}$ uses a weighted average:

$$L_{stage2} = \frac{\sum_{j=1}^{n}(n-j) \cdot L_j}{\sum_{j=1}^{n}(n-j)}, \quad (10)$$

where $n$ represents the number of hidden layers. Note that we give the early internal classifiers a bigger weight since they need more transformation to fit representations for classification.

Inference. Following Zhou et al. (2020), we adopt a patience-based strategy to decide which layer to exit. Specifically, we set the patience to $t$. Given a sample $x$, the early-exit model predicts from bottom to top. If the predictions of $t$ consecutive intermediate classifiers remain “unchanged”, the model exits at that layer and outputs the corresponding prediction. While for regression, we consider the prediction as “unchanged” by:

$$\text{unchanged} = \begin{cases} 
\text{True} & \text{if } |y_i - y_{i-1}| < \delta \\
\text{False} & \text{if } |y_i - y_{i-1}| \geq \delta 
\end{cases} \quad (11)$$

where $y_i$ represents the prediction of the current layer and $\delta$ is a pre-defined threshold.
4 Experiments

4.1 Datasets

We evaluate our proposed approach on monolingual tasks and multilingual tasks. For monolingual tasks, our experiments are conducted on the GLUE benchmark (Wang et al., 2018), including MRPC, QQP, SST-2, MNLI (matched/mismatched), QNLI, RTE, CoLA, and STS-B. Following Zhou et al. (2020), if a dataset has more than one metric, we report the arithmetic mean of the metrics. For multilingual tasks, we conduct experiments on paraphrase identification task (PAWS-X; Yang et al., 2019) and natural language inference (XNLI; Conneau et al., 2018). We show details of all the datasets in the Appendix.

4.2 Baselines

We compare our methods with the following competitive models:

**Backbone models.** We use BERT-base and ALBERT-base for monolingual tasks, which are widely used in studies of early exit. For multilingual tasks, we use multilingual BERT. All these pre-trained language models are released by HuggingFace.

**PABEE.** PABEE is a patience-based early exit method proposed by Zhou et al. (2020).

**PonderNet.** PonderNet is proposed by Banino et al. (2021), which treats the exit layer’s index as a latent variable.

**PBERT/PALBERT.** Balagansky and Gavrilov (2022) proposed a deterministic Q-exit criterion to improve the performance PonderNet. We apply the Q-exit criterion to BERT and ALBERT, and we denote them as PBERT and PALBERT.

4.3 Experimental Setup

**Training.** For all experiments, we set the batch size for training to 32. We search for the best learning rate in {1e-5, 2e-5, 3e-5, 5e-5} for all baselines and the first training stage of our method. The range of learning of the second training stage of our method is in {1e-3, 2e-3, 3e-3, 5e-3, 8e-3, 9e-3}. The downsample sizes of adapters are searched in {32, 64, 128, 256}. Since the effect of using the last layer for classification remains the same during the second training stage, we choose the best checkpoint on the development set with patience set to 6. To avoid the propensity of the model to use later layers for classification, which results in poor acceleration, we filter checkpoints whose average Flops (M) on the development set are greater than $\tau$ on the verification set. $\tau$ is chosen in {8000, 9000, 10000}. We conduct the experiments on one NVIDIA GTX3090 GPU.

**Inference.** We set the batch size for inference to 1. Patience is set to 6 for patience-based methods following Zhou et al. (2020), accelerating the inference while maintaining a satisfactory effect. For PonderNet, PBERT, and PALBERT, the threshold for early exit is set to 0.5 following Balagansky and Gavrilov (2022). For STS-B, we set the threshold $\delta$ to 0.5. We use Flops to calculate the speedup ratio.

4.4 Main Results

We report our experimental results with BERT and ALBERT backbone for monolingual tasks on GLUE in Table 1. Our method outperforms all compared approaches from the perspective of the macro score while maintaining the average speed-up ratio between 1.25 and 1.27 on both the development and test sets. Our approach works well on small datasets such as CoLA, RTE, MRPC, and STS-B, on which the previous approaches of early exit often have poor performance.

We also conduct experiments on multilingual tasks to examine the generality of our approach. Table 2 shows a comparison of our approach with baseline and PABEE. In addition to accelerating inference, our approach outperforms mBERT and PABEE on both PAWSX and XNLI.

Since our approach introduces an adapter in each intermediate layer, a small amount of computational overhead is added. In order to make a fairer and more comprehensive evaluation of our approach, we adjust the patience to obtain the effect at different speed-up ratios. We compare our approach with PABEE, as we both adopt a patience-based early exit mechanism, making it convenient to adjust the speed-up ratio. As shown in Figure 5, we experiment on 3 GLUE datasets with ALBERT backbone. The ability to capture semantic information is trained in the first training stage and maintained by freezing transformer layer parameters during the second training stage, improving performance when using the whole model for inference. Moreover, during training, the disentangled representations avoid conflicts between two different loss signals in the transformer layer. This division of functions makes the model represen-
<table>
<thead>
<tr>
<th>Method</th>
<th>Speed-up</th>
<th>CoLA</th>
<th>RTE</th>
<th>MRPC</th>
<th>QQP</th>
<th>SST-2</th>
<th>QNLI</th>
<th>MNLI</th>
<th>STS-B</th>
<th>Macro.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dev Set</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BERT-base</td>
<td>×1.00</td>
<td>52.4</td>
<td>67.2</td>
<td>85.6</td>
<td>80.1</td>
<td>93.4</td>
<td>90.4</td>
<td>84.0</td>
<td>83.7</td>
<td>79.6</td>
</tr>
<tr>
<td>PABEE</td>
<td>×1.40</td>
<td>45.8</td>
<td>64.8</td>
<td>82.5</td>
<td>79.7</td>
<td>92.3</td>
<td>89.3</td>
<td>83.7</td>
<td>83.7</td>
<td>77.7</td>
</tr>
<tr>
<td>PonderNet</td>
<td>×1.34</td>
<td>51.1</td>
<td>72.2</td>
<td>86.9</td>
<td>87.6</td>
<td>91.0</td>
<td>88.8</td>
<td>81.8</td>
<td>88.3</td>
<td>81.0</td>
</tr>
<tr>
<td>PBERT</td>
<td>×1.23</td>
<td>55.1</td>
<td>75.8</td>
<td>88.4</td>
<td>88.8</td>
<td>92.3</td>
<td>91.2</td>
<td>83.7</td>
<td>89.3</td>
<td>83.1</td>
</tr>
<tr>
<td>Ours</td>
<td>×1.27</td>
<td>58.0</td>
<td>78.0</td>
<td>90.2</td>
<td>88.8</td>
<td>92.8</td>
<td>91.2</td>
<td>83.9</td>
<td>90.2</td>
<td>84.1</td>
</tr>
<tr>
<td><strong>ALBERT-base</strong></td>
<td>×1.00</td>
<td>52.2</td>
<td>71.4</td>
<td>86.8</td>
<td>79.5</td>
<td>92.8</td>
<td>91.5</td>
<td>84.6</td>
<td>87.9</td>
<td>80.8</td>
</tr>
<tr>
<td>PABEE</td>
<td>×1.45</td>
<td>48.7</td>
<td>69.5</td>
<td>85.7</td>
<td>79.7</td>
<td>91.8</td>
<td>91.0</td>
<td>84.1</td>
<td>86.5</td>
<td>79.6</td>
</tr>
<tr>
<td>PALBERT</td>
<td>×1.21</td>
<td>47.1</td>
<td>71.9</td>
<td>86.1</td>
<td>79.1</td>
<td>91.4</td>
<td>90.9</td>
<td>83.2</td>
<td>85.1</td>
<td>79.4</td>
</tr>
<tr>
<td>Ours</td>
<td>×1.26</td>
<td>50.1</td>
<td>71.4</td>
<td>86.7</td>
<td>79.3</td>
<td>92.2</td>
<td>91.0</td>
<td>83.5</td>
<td>86.7</td>
<td>80.1</td>
</tr>
</tbody>
</table>

Table 1: Experimental results with BERT and ALBERT backbone on the development set and the test set of GLUE. We report the average result of five runs. The Macro score shows the average results across the eight tasks. Note that we apply learnable classifiers instead of ETF classifiers to our approach on STS-B since ETF classifiers do not support regression.

Figure 5: Speed-accuracy curves of PABEE and our approach with ALBERT backbone on CoLA, MRPC, and SST-2. The dashed lines mark the performance of ALBERT-base for reference.

5 Analysis and Discussion

5.1 Ablation Study

We conduct an ablation study with BERT backbone on MNLI, SST-2, and MRPC. Three experiments are performed based on our method:

- **w/o ETF**: We remove ETF classifiers and replace them with original trainable classifiers.
- **w/o adapters**: We remove all adapter modules and directly use the output of the layernorm after two feed-forward layers as the classifier’s input for each intermediate layer.
- **w/o two-stage**: We train the transformer layers and the adapter modules simultaneously instead of adopting the two-stage training strategy.
<table>
<thead>
<tr>
<th>Method</th>
<th>Speed-up</th>
<th>PAWSX</th>
<th>XNLI</th>
</tr>
</thead>
<tbody>
<tr>
<td>mBERT</td>
<td>×1.00</td>
<td>83.1</td>
<td>65.4</td>
</tr>
<tr>
<td>PABEE</td>
<td>×1.20</td>
<td>82.9</td>
<td>66.0</td>
</tr>
<tr>
<td>Ours</td>
<td>×1.08</td>
<td>83.3</td>
<td>66.5</td>
</tr>
</tbody>
</table>

Table 2: Test results for multilingual tasks with five different random seeds. We train the model on the English training set and test it on the test set of all languages. We report the average result of five runs.

<table>
<thead>
<tr>
<th>Method</th>
<th>Speed-up</th>
<th>MNLI</th>
<th>SST-2</th>
<th>MRPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours</td>
<td>×1.24</td>
<td>83.5</td>
<td>92.3</td>
<td>87.5</td>
</tr>
<tr>
<td>w/o ETF</td>
<td>×1.24</td>
<td>83.6</td>
<td>92.2</td>
<td>86.8</td>
</tr>
<tr>
<td>w/o adapters</td>
<td>×1.12</td>
<td>81.3</td>
<td>90.1</td>
<td>85.0</td>
</tr>
<tr>
<td>w/o 2-stage</td>
<td>×1.32</td>
<td>51.3</td>
<td>89.9</td>
<td>82.6</td>
</tr>
</tbody>
</table>

Table 3: The ablation study of our method. We report the results of BERT based model on the development set of MNLI, SST-2 and MRPC.

The results of the ablation study are shown in Table 3. As bridges between hidden states and classifiers, the adapters undertake to learn task-specific representations and convert intermediate layer representations into representations for classification. Note that the additional computation caused by the adapter modules is about 2% on BERT and 3% on ALBERT, which is acceptable considering the benefits it brings. Acceleration and performance are both degraded without using adapter modules. Moreover, the performance and stability of the model drop significantly and without using the two-stage training strategy, which indicates that the gradients returned by the higher layers mixed with the gradients returned by the classifiers confuse the model’s representation and mislead the optimization. ETF classifiers generally outperform learnable classifiers in performance and efficiency especially for low-resource datasets. They further improve the model’s performance by enhancing the intermediate layers’ classification ability while reducing the number of learnable parameters.

5.2 Impact of Adapter Hidden Size

The hidden size of adapters $m$ affect many aspects of our approach. Experiments are conducted to study the impact of $m$. We choose ALBERT as the backbone because it has a bigger hidden size. Figure 6 shows that the hidden size of adapters has little effect on the accuracy while it makes a difference in the speed-up ratio. The accuracy rate reaches the top with a good speed-up ratio when $m$ is set to 32. And a large size leads to a decrease both in performance and acceleration.

As shown in Figure 7, a bigger hidden size of adapters encourages the samples to exit at a lower layer. Based on this observation, we assume that a small size restricts the ability of adapters to express so that the intermediate classifiers have difficulties in giving a unified prediction. While a large size makes the lower layers too confident to wait for a high-level representation from the higher layers, resulting in a wrong decision about when to exit in some cases. This finding demonstrates that a proper hidden size of adapters gives the success to stop inference at a proper layer.

5.3 Error Analysis

In order to further explore the performance-boosting aspects of our method, we conduct an error analysis between PABEE and our approach. Suppose a model has a stable performance improvement compared to the baseline. In that case, the model should predict as correctly as possible for the samples that the baseline predicts correctly and that the model can give correct predictions for the instances that the baseline fails. Thus, we divide the samples in the development sets into two
Figure 8: Error analysis with ALBERT backbone on MRPC, RTE, and SST-2. The dark blue and dark red bars indicate the number of correct and wrong samples predicted by PABEE on the dataset, part of which is covered by light color bars. And we call them PABEE (Correct) and PABEE (Wrong), respectively. The light blue bars represent the number of correct samples predicted by our approach on PABEE (Correct). The light red bars represent the number of correct samples predicted by our approach on PABEE (Wrong).

classes PABEE (Correct) and PABEE (Wrong), according to the prediction results of PABEE. Then we test our approach on the two classes to observe the change in the distribution of correctly predicted samples. As shown in Figure 8, our method correctly predicts the vast majority of samples in PABEE (Correct) and about half in PABEE (Wrong). There can be two reasons for the prediction error: (1) The model stops inference too early; (2) Biased representations fail to make a correct understanding of the samples. So the experimental results indicate that our approach improves the representation accuracy and gives better answers to when to exit.

6 Related Work

Early exit Early exit focuses on enabling input-adaptive inference to reduce the computational cost, which has been proven effective on various NLP tasks (Elbayad et al., 2019; Xin et al., 2020a; Li et al., 2021b; He et al., 2021c; Xin et al., 2021; Sun et al., 2022; Schuster et al., 2022). Because the model needs to decide whether stop inference at a specific layer when using early exit, so there are two main problems: ‘How to decide whether stop inference at each layer?’ and ‘How to induce a better internal classifier for each layer?’.

To the former problem, Xin et al. (2020b); Schwartz et al. (2020); Xie et al. (2021) use confidence-based criterion; Zhou et al. (2020) propose a novel and effective patience mechanism; Sun et al. (2021) propose a voting-based strategy. In addition, Banino et al. (2021); Balagansky and Gavrilov (2022) utilize a latent variable to predict the exit layer’s index. For the latter one, Liu et al. (2020); Geng et al. (2021) use self-distillation to induce better internal classifiers; Zhu (2021) extent self-distillation to mutual distillation and use learnable weights to balance off different internal classifiers’ objectives; Liao et al. (2021) use global past and future information with imitation learning to train internal classifiers; Sun et al. (2021) maximize the mutual information of internal classifiers to enhance the diversity of classifiers, making it suitable for voting-based strategy. Besides, Li et al. (2021a) dynamically cascade proper-sized and complete models, enabling shallow layers with high-level semantic information. Liu et al. (2022) propose a novel pre-training method that encourages the intermediate layers of the pre-trained model to learn high-level semantics, making the pre-trained model more suitable for the early exiting mechanism.

Adapter With the increase in the number of parameters of PLMs, full-parameter fine-tuning faces difficulties in computing resources and is prone to the over-fitting problem (Phang et al., 2018; Dodge et al., 2020; Zhang et al., 2021). Therefore, research on efficient fine-tuning (Houlsby et al., 2019; Zaken et al., 2022; Hu et al., 2021; Li and Liang, 2021) is developing rapidly. Adapter is a promising efficient fine-tuning method. Adapter-based methods inject small-scale adapters to the Transformer layers and only tune these adapters. Many studies (Pfeiffer et al., 2020a,b; Guo et al., 2020; Rücklè et al., 2021; He et al., 2021b; Han et al., 2021) have shown that adapter-based methods can achieve comparable performance to the full-parameter fine-tuning. To further reduce the parameter amount of the adapter module, Karimi Mahabadi et al. (2021) propose Compacter, a more lightweight adapter that utilizes a combination of hypercomplex multiplication and parameter sharing. Due to the modularity of the adapter, adapter-based tuning is suitable for multi-task learning (Stickland and Murray, 2019; Mahabadi et al., 2021) and can provide different task-specific representations for various tasks. Adapterfusion Pfeiffer et al. (2021) proposes a fusion method named AdapterFusion that fuses adapter representations of different tasks and makes full use of cross-task knowledge.
7 Conclusion

In this paper, we investigate the internal representation of early exit models and observe that the internal layers have difficulty providing good generic linguistic representations for subsequent layers and good task-specific representations for internal classifiers. We propose an adapter-based method to disentangle the two conflicting representations and utilize equiangular tight frame classifiers to improve representations for classification. Experiments on the GLUE benchmark and two cross-lingual transfer tasks demonstrate that our proposed method performs better than existing methods. For future work, we would like to explore: (1) strengthening the information interaction between layers to make full use of the previous layers’ predictions, thus optimizing the representation for the internal classifiers; (2) optimizing the early exiting mechanism to further improve performance and accelerate inference; (3) applying our method to more advanced pre-trained models such as DeBERTa (He et al., 2020, 2021a), ElasticBERT (Liu et al., 2022), etc.

Limitations

Even though our work improves early exit performance effectively, some limitations are still listed below:

- Our approach focuses on making the intermediate representations of early exit models capable of general linguistic representation learning and task-specific representation extraction. Therefore, we did not fully use the model’s high-level representation and fuse representations of previous layers, which may restrict the performance of our method. For future work, we would like to strengthen the information interaction between layers to make full use of the previous layers’ predictions, thus optimizing the representation for the internal classifiers.
- Although our early exit method has achieved better performance, we have lost some inference speed due to the introduction of additional adapter modules. In the future, we will try more efficient adapter-based tuning.
- In recent years, the parameter size of generative pre-trained models has been continuously increasing, leading to remarkable performance on various NLP tasks. There is an urgent need to develop inference acceleration methods for generative pre-trained models. Unfortunately, our method is limited to discriminative pre-trained models. Our future work will investigate early exit strategies for generative pre-trained models.

Acknowledgements

This work is supported by the National Science Foundation of China (No. 62206194), the Natural Science Foundation of Jiangsu Province, China (Grant No. BK20220488), and JSSCBS20210661. This work is partially supported by the joint research project of Alibaba and Soochow University.

References


A Details of Datasets

We show details of datasets for monolingual tasks in Table 4 and multilingual tasks in Table 5.

### Table 4: Detailed description and statistics of datasets for monolingual tasks.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Task</th>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>SST-2</td>
<td>Sentiment</td>
<td>Acc.</td>
</tr>
<tr>
<td>MRPC</td>
<td>Paraphrase</td>
<td>Acc./F1</td>
</tr>
<tr>
<td>QQP</td>
<td>Paraphrase</td>
<td>Acc./F1</td>
</tr>
<tr>
<td>MNLI</td>
<td>NLI</td>
<td>Matched Acc./Mismatched Acc.</td>
</tr>
<tr>
<td>QNLI</td>
<td>QA/NLI</td>
<td>Acc.</td>
</tr>
<tr>
<td>RTE</td>
<td>NLI</td>
<td>Acc.</td>
</tr>
</tbody>
</table>

### Table 5: Detailed description and statistics of datasets for multilingual tasks.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Task</th>
<th></th>
<th>Languages</th>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>XNLI</td>
<td>NLI</td>
<td>15</td>
<td>Acc.</td>
<td></td>
</tr>
<tr>
<td>PAWS-X</td>
<td>Paraphrase Adversaries</td>
<td>7</td>
<td>Acc.</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Detailed description and statistics of datasets for monolingual tasks.

Table 5: Detailed description and statistics of datasets for multilingual tasks.
**ACL 2023 Responsible NLP Checklist**

**A** For every submission:

- ✔ A1. Did you describe the limitations of your work?  
  *the last section*

- ✔ A2. Did you discuss any potential risks of your work?  
  *the last section*

- ✔ A3. Do the abstract and introduction summarize the paper’s main claims?  
  *section 1*

- ✗ A4. Have you used AI writing assistants when working on this paper?  
  *Left blank.*

**B** ✔ Did you use or create scientific artifacts?

  *section 4.1*

- ✔ B1. Did you cite the creators of artifacts you used?  
  *section 4.1*

- ✔ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?  
  *section 4.1*

- ✔ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?  
  *section 4.1*

- ✗ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?  
  *No, The datasets we use are widely recognized public datasets.*

- ✔ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
  *Appendix A*

- ✔ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
  *Appendix A*

**C** ✔ Did you run computational experiments?

  *section 4.3*

- ✔ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?  
  *section 4.3*

_The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance._
C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?

section 4.3

C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?

section 4.4

C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?

section 4.3

D Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?

No response.

D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants’ demographic (e.g., country of residence)?

No response.

D3. Did you discuss whether and how consent was obtained from people whose data you’re using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?

No response.

D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?

No response.

D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?

No response.