Early Exit with Disentangled Representation and Equiangular Tight Frame

Yixin Ji[1], **Jikai Wang**[1]*, **Juntao Li**[1]†, **Qiang Chen**[2], **Wenliang Chen**[1], **Min Zhang**[1]
[1]Institute of Computer Science and Technology, Soochow University, China
[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

* Equal contribution.
† Corresponding author.

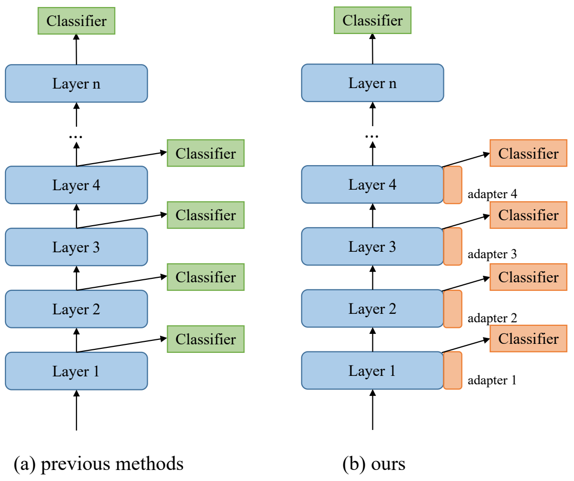


Figure 1: Comparison of our method with previous methods. Previous early exit methods allows the representation of the internal layer to extract generic linguistic and task-specific representation simultaneously. Our method uses an additional adapter module to undertake the learning of task-specific representation.

may instead contain too much over-complicated or irrelevant information to make the accurate prediction (Xin et al., 2020b; Liu et al., 2020).

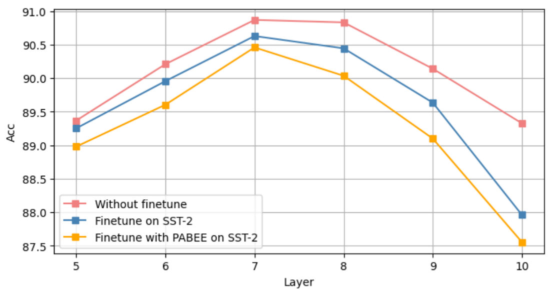
To achieve a good trade-off between inference cost and performance, the early exit mechanism (Xin et al., 2020a; Zhou et al., 2020; Li et al., 2021b; He et al., 2021c; Xin et al., 2021; Banino et al., 2021; Balagansky and Gavrilov, 2022; Sun et al., 2022), a kind of adaptive inference strategy, has been 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). Existing 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 alternate training, they have not explicitly decoupled the two distinct types of representation.

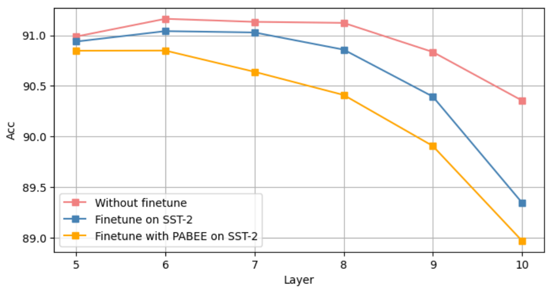
In this work, we propose a novel early exit method that enables the PLM's intermediate representation to consider both the classification ability and the ability to capture semantic information. Specifically, our approach hands off the task-specific representation of the intermediate transformer layer 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`.



(a) BERT-POS



(b) BERT-Chunking

Figure 2: Layer-wise probing performance on POS and Chunking with BERT-base as the backbone.

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., 2020), which jointly trains the last and internal classifiers. 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 PABEE. 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 PABEE 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 $\boldsymbol{h}_i$ as the input to L_{i+1}:

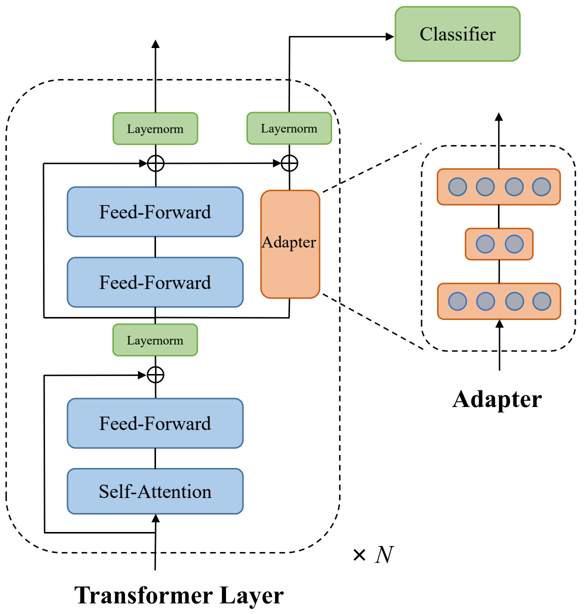


Figure 3: The structure of the intermediate transformer layer with the adapter module for early exit. N is the number of transformer layers.

$$\begin{aligned} \boldsymbol{s}_i &= \text{LN}(\text{FFN}(\text{SA}(\boldsymbol{h}_{i-1})) + \boldsymbol{h}_{i-1}) \\ \boldsymbol{s}'_i &= \text{FFN}(\text{FFN}(\boldsymbol{s}_i)) + \boldsymbol{s}_i \\ \boldsymbol{h}_i &= \text{LN}(\boldsymbol{s}'_i) \end{aligned} \tag{1}$$

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 $\boldsymbol{h}'$ for classification:

$$\boldsymbol{h}'_i = \text{LN}(\text{Adapter}(\boldsymbol{s}_i) + \boldsymbol{s}'_i) \tag{2}$$

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

$$\text{Adapter}(\boldsymbol{s}_i) = f_{up}(\text{ReLU}(f_{down}(\boldsymbol{s}_i))) \tag{3}$$

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 $\boldsymbol{h}$ into the next transformer layer and use $\boldsymbol{h}'$ 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.

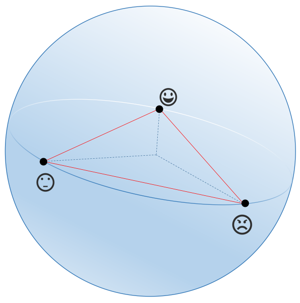


Figure 4: An illustration of the simplex equiangular tight frame (ETF) in the three-dimension space.

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* $\mathbf{w}_i \in \mathbb{R}^d$, $i = 1, 2, ..., K, d \geq K - 1$, *it satisfies:*

$$\boldsymbol{W}_{\text{ETF}} = \sqrt{\frac{K}{K-1}}\mathbf{U}\left(\mathbf{I}_K - \frac{1}{K}\mathbf{1}_K\mathbf{1}_K^T\right), \quad (4)$$

where $\boldsymbol{W}_{\text{ETF}} = [\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_K] \in \mathbb{R}^{d\times K}$, $\mathbf{I}_K \in \mathbb{R}^{K\times K}$ *is the identity matrix,* $\mathbf{1}_K \in \mathbb{R}^K$ *is a vector of all ones and* $\mathbf{U} \in \mathbb{R}^{K\times K}$ *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 $\boldsymbol{h}'_i$ from the Transformer layer L_i, the ETF classifier output the probability $p_i(y|x)$:

$$p_i(y|x) = \text{Softmax}(\boldsymbol{W}_{\text{ETF}}\boldsymbol{h}'_i) \quad (5)$$

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:

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

We use the mean squared error for regression tasks:

$$\mathcal{L}_{stage1} = (y - \widehat{y})^2. \quad (7)$$

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:

$$\mathcal{L}_i = \text{CE}(logits_i, y). \quad (8)$$

For regression, we also use the mean squared error:

$$\mathcal{L}_i = (y - \widehat{y}_i)^2. \quad (9)$$

The total loss $\mathcal{L}_{stage2}$ uses a weighted average:

$$\mathcal{L}_{stage2} = \frac{\sum_{j=1}^{n}(n-j)\cdot\mathcal{L}_j}{\sum_{j=1}^{n}(n-j)}, \quad (10)$$

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:

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

where y_i represents the prediction of the current layer and δ 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[1].

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 τ on the verification set. τ 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 δ 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-

[1]https://huggingface.co/

Method	Speed-up	CoLA	RTE	MRPC	QQP	SST-2	QNLI	MNLI	STS-B	Macro.
					Dev Set					
BERT-base	$\times$1.00	60.2	70.5	87.9	89.7	93.1	91.4	84.6	89.1	83.3
PABEE	$\times$1.39	50.3	65.3	84.8	**89.3**	91.2	**89.7**	83.4	88.6	80.3
PonderNet	$\times$1.43	46.5	62.8	82.0	87.1	90.3	85.6	79.9	83.5	77.2
PBERT	$\times$1.48	54.8	61.7	83.1	89.1	91.6	89.4	**83.5**	87.7	80.1
Ours	$\times$1.25	**55.7**	**69.7**	**87.5**	88.9	**92.3**	89.1	**83.5**	**88.9**	**82.0**
ALBERT-base	$\times$1.00	57.5	76.7	90.1	89.0	92.4	91.7	84.8	90.9	84.1
PABEE	$\times$1.43	54.1	73.9	86.9	**89.2**	92.3	**91.4**	**84.4**	90.0	82.8
PonderNet	$\times$1.34	51.1	72.2	86.9	87.6	91.0	88.8	81.8	88.3	81.0
PALBERT	$\times$1.23	55.1	75.8	88.4	88.8	92.3	91.2	83.7	89.7	83.1
Ours	$\times$1.27	**58.0**	**78.0**	**90.2**	88.8	**92.8**	91.2	83.9	**90.2**	**84.1**
					Test Set					
BERT-base	$\times$1.00	52.4	67.2	85.6	80.1	93.4	90.4	84.0	83.7	79.6
PABEE	$\times$1.40	45.8	64.8	82.5	**79.7**	92.3	**89.3**	**83.7**	83.7	77.7
PBERT	$\times$1.48	46.9	64.3	80.7	79.3	91.8	89.1	83.0	82.4	77.2
Ours	$\times$1.25	**51.7**	**66.8**	**83.8**	79.4	**92.9**	88.5	83.0	**84.0**	**78.8**
ALBERT-base	$\times$1.00	52.2	71.4	86.8	79.5	92.8	91.5	84.6	87.9	80.8
PABEE	$\times$1.45	48.7	69.5	85.7	**79.7**	91.8	**91.0**	**84.1**	86.5	79.6
PALBERT	$\times$1.21	47.1	**71.9**	86.1	79.1	91.4	90.9	83.2	85.1	79.4
Ours	$\times$1.26	**50.1**	71.4	**86.7**	79.3	**92.2**	**91.0**	83.5	**86.7**	**80.1**

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.

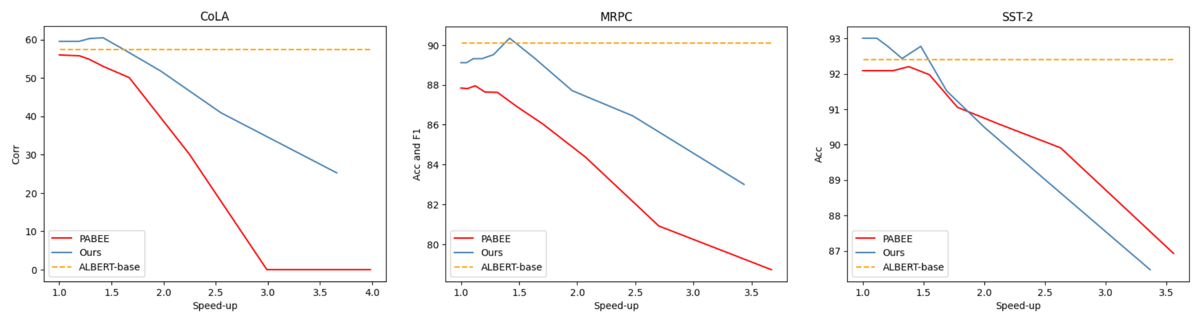


Figure 5: Speed-accuracy curves of PABEE and our approach with ALBERT backbone on CoLA, MRPC, and SST-2. The dashed lines mark the performance of ALBERT-base for reference.

tation cleaner and more meaningful. As a result, the performance penalty caused by accelerated inference reduces significantly at a speed-up ratio between 1 and 1.5. It is worth noting that there is an upward trend in the curves of our approach at a low speed-up ratio, indicating that our approach helps to solve the overthinking problem.

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.

Method	Speed-up	PAWSX	XNLI
mBERT	×1.00	83.1	65.4
PABEE	×1.20	82.9	66.0
Ours	×1.08	**83.3**	**66.5**

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.

Method	Speed-up	MNLI	SST-2	MRPC
Ours	×1.24	83.5	**92.3**	**87.5**
w/o ETF	×1.24	**83.6**	92.2	86.8
w/o adapters	×1.12	81.3	90.1	85.0
w/o 2-stage	×1.32	51.3	89.9	82.6

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

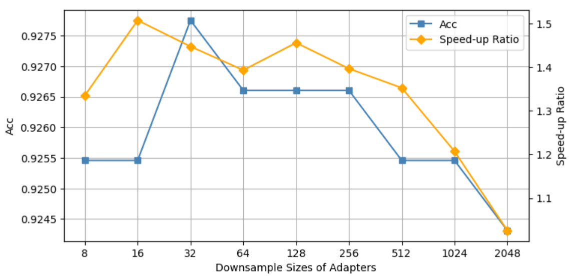


Figure 6: Influence of downsample size of adapters on accuracy and speed-up ratio with ALBERT-base backbone on SST-2. The orange and blue lines represent the speed-up ratio and accuracy.

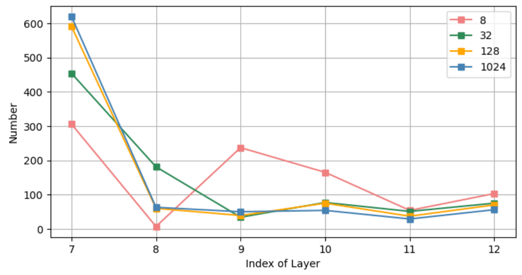


Figure 7: Distribution of early exit layers with ALBERT-base backbone on SST-2 with different downsample sizes of adapters. We set patience to 6 in this experiment.

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

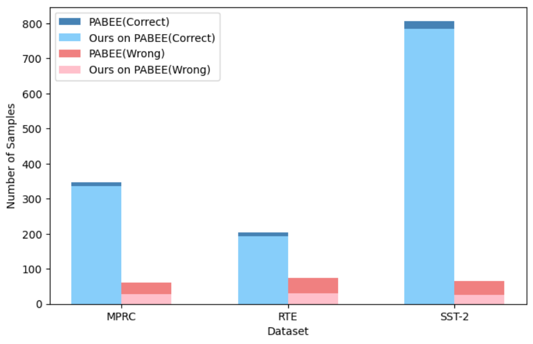


Figure 8: Error analysis with ALBERT backbone on MRPC, RTE, and SST-2. The dark blue and dark red bars indicate the number of correct and wrong samples predicted by PABEE on the dataset, part of which is covered by light color bars. And we call them PABEE (Correct) and PABEE (Wrong), respectively. The 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; Rücklé 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

Nikita Balagansky and Daniil Gavrilov. 2022. PALBERT: Teaching ALBERT to ponder. In *Advances in Neural Information Processing Systems*.

Andrea Banino, Jan Balaguer, and Charles Blundell. 2021. Pondernet: Learning to ponder. In *8th ICML Workshop on Automated Machine Learning (AutoML)*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*.

Nadir Durrani, Hassan Sajjad, and Fahim Dalvi. 2021. How transfer learning impacts linguistic knowledge in deep NLP models? In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4947–4957, Online. Association for Computational Linguistics.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. 2019. Depth-adaptive transformer. In *International Conference on Learning Representations*.

Tomer Galanti, András György, and Marcus Hutter. 2021. On the role of neural collapse in transfer learning. In *International Conference on Learning Representations*.

Shijie Geng, Peng Gao, Zuohui Fu, and Yongfeng Zhang. 2021. Romebert: Robust training of multi-exit bert. *arXiv preprint arXiv:2101.09755*.

Junliang Guo, Zhirui Zhang, Linli Xu, Hao-Ran Wei, Boxing Chen, and Enhong Chen. 2020. Incorporating bert into parallel sequence decoding with adapters. *Advances in Neural Information Processing Systems*, 33:10843–10854.

Wenjuan Han, Bo Pang, and Ying Nian Wu. 2021. Robust transfer learning with pretrained language models through adapters. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 854–861.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021a. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced BERT with disentangled attention. *CoRR*, abs/2006.03654.

Ruidan He, Linlin Liu, Hai Ye, Qingyu Tan, Bosheng Ding, Liying Cheng, Jiawei Low, Lidong Bing, and Luo Si. 2021b. On the effectiveness of adapter-based tuning for pretrained language model adaptation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2208–2222.

Xuanli He, Iman Keivanloo, Yi Xu, Xiang He, Belinda Zeng, Santosh Rajagopalan, and Trishul Chilimbi. 2021c. Magic pyramid: Accelerating inference with early exiting and token pruning. *arXiv preprint arXiv:2111.00230*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4411–4421. PMLR.

Wenlong Ji, Yiping Lu, Yiliang Zhang, Zhun Deng, and Weijie J Su. 2021. An unconstrained layer-peeled perspective on neural collapse. In *International Conference on Learning Representations*.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. *Advances in Neural Information Processing Systems*, 34:1022–1035.

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

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Lei Li, Yankai Lin, Deli Chen, Shuhuai Ren, Peng Li, Jie Zhou, and Xu Sun. 2021a. CascadeBERT: Accelerating inference of pre-trained language models via calibrated complete models cascade. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 475–486, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th*

International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 4582–4597.

Xiaonan Li, Yunfan Shao, Tianxiang Sun, Hang Yan, Xipeng Qiu, and Xuan-Jing Huang. 2021b. Accelerating bert inference for sequence labeling via early-exit. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 189–199.

Kaiyuan Liao, Yi Zhang, Xuancheng Ren, Qi Su, Xu Sun, and Bin He. 2021. A global past-future early exit method for accelerating inference of pre-trained language models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2013–2023.

Weijie Liu, Peng Zhou, Zhiruo Wang, Zhe Zhao, Haotang Deng, and Qi Ju. 2020. Fastbert: a self-distilling bert with adaptive inference time. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6035–6044.

Xiangyang Liu, Tianxiang Sun, Junliang He, Jiawen Wu, Lingling Wu, Xinyu Zhang, Hao Jiang, Zhao Cao, Xuanjing Huang, and Xipeng Qiu. 2022. Towards efficient NLP: A standard evaluation and a strong baseline. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3288–3303, Seattle, United States. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. 2021. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 565–576.

OpenAI. 2022. Chatgpt: Optimizing language models for dialogue. *Open AI, blog*.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. Adapterfusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020a. Adapterhub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020b. Mad-x: An adapter-based framework for multi-task cross-lingual transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673.

Jason Phang, Thibault Févry, and Samuel R Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*.

Mary Phuong and Christoph H Lampert. 2019. Distillation-based training for multi-exit architectures. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1355–1364.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *Open AI, blog*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2021. Adapterdrop: On the efficiency of adapters in transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7930–7946.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. 2022. Confident adaptive language modeling. In *Advances in Neural Information Processing Systems*.

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

Asa Cooper Stickland and Iain Murray. 2019. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning. In *International Conference on Machine Learning*, pages 5986–5995. PMLR.

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

Tianxiang Sun, Yunhua Zhou, Xiangyang Liu, Xinyu Zhang, Hao Jiang, Zhao Cao, Xuanjing Huang, and Xipeng Qiu. 2021. Early exiting with ensemble internal classifiers. *arXiv preprint arXiv:2105.13792*.

Tom Tirer and Joan Bruna. 2022. Extended unconstrained features model for exploring deep neural collapse. In *international conference on machine learning (ICML)*.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task chunking. In *Fourth Conference on Computational Natural Language Learning and the Second Learning Language in Logic Workshop*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Keli Xie, Siyuan Lu, Meiqi Wang, and Zhongfeng Wang. 2021. Elbert: Fast albert with confidence-window based early exit. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7713–7717. IEEE.

Ji Xin, Rodrigo Nogueira, Yaoliang Yu, and Jimmy Lin. 2020a. Early exiting bert for efficient document ranking. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 83–88.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020b. Deebert: Dynamic early exiting for accelerating bert inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2246–2251.

Ji Xin, Raphael Tang, Yaoliang Yu, and Jimmy Lin. 2021. Berxit: Early exiting for bert with better fine-tuning and extension to regression. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 91–104.

Yibo Yang, Shixiang Chen, Xiangtai Li, Liang Xie, Zhouchen Lin, and Dacheng Tao. 2022. Inducing neural collapse in imbalanced learning: Do we really need a learnable classifier at the end of deep neural network? In *Advances in Neural Information Processing Systems*.

Yinfei Yang, Yuan Zhang, Chris Tar, and Jason Baldridge. 2019. PAWS-X: A cross-lingual adversarial dataset for paraphrase identification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3687–3692, Hong Kong, China. Association for Computational Linguistics.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. 2021. Revisiting few-sample bert fine-tuning. In *International Conference on Learning Representations*.

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

Wei Zhu. 2021. Leebert: Learned early exit for bert with cross-level optimization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2968–2980.

Zhihui Zhu, Tianyu Ding, Jinxin Zhou, Xiao Li, Chong You, Jeremias Sulam, and Qing Qu. 2021. A geometric analysis of neural collapse with unconstrained features. *Advances in Neural Information Processing Systems*, 34:29820–29834.

A Details of Datasets

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

Dataset	**Task**	**Metrics**
CoLA	Acceptability	Matthews Corr.
SST-2	Sentiment	Acc.
MRPC	Paraphrase	Acc./F1
STS-B	Sentence Similarity	Pearson/Spearman Corr.
QQP	Paraphrase	Acc./F1
MNLI	NLI	Matched Acc./Mismatched Acc.
QNLI	QA/NLI	Acc.
RTE	NLI	Acc.

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

Dataset	**Task**	**|Languages|**	**Metrics**
XNLI	NLI	15	Acc.
PAWS-X	Paraphrase Adversaries	7	Acc.

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.