Accelerating Inference for Pretrained Language Models by Unified Multi-Perspective Early Exiting

Jun Kong [†], **Jin Wang**[†][*], **Liang-Chih Yu**[‡][*] and **Xuejie Zhang**[†]
[†]School of Information Science and Engineering, Yunnan University, Yunnan, P.R. China
[‡]Department of Information Management, Yuan Ze University, Taiwan
Contact:wangjin@ynu.edu.cn, lcyu@saturn.yzu.edu.tw

Abstract

Conditional computation algorithms, such as the early exiting (EE) algorithm, can be applied to accelerate the inference of pretrained language models while maintaining competitive performance on resource-constrained devices. However, this approach is only applied to the vertical architecture to decide which layers should be used for inference. Conversely, the operation of the horizontal perspective is ignored, and the determination of which tokens in each layer should participate in the computation fails, leading to a high redundancy for adaptive inference. To address this limitation, a unified horizontal and vertical multi-perspective early exiting (MPEE) framework is proposed in this study to accelerate the inference of transformer-based models. Specifically, the vertical architecture uses recycling EE classifier memory and weighted self-distillation to enhance the performance of the EE classifiers. Then, the horizontal perspective uses recycling class attention memory to emphasize the informative tokens. Conversely, the tokens with less information are truncated by weighted fusion and isolated from the following computation. Based on this, both horizontal and vertical EE are unified to obtain a better tradeoff between performance and efficiency. Extensive experimental results show that MPEE can achieve higher acceleration inference with competent performance than existing competitive methods. The code for this paper is available at: https://github.com/JunKong5/MPEE.

1 Introduction

Pretrained language models (PLMs) (Devlin et al., 2019; Lan et al., 2019; Liu et al., 2019) have shown promising performance in many natural language processing tasks. The success of PLMs incurs computational consumption and long inference latency, which prevents these models from being deployed on resource-constrained devices, such as edge devices, or in time-sensitive scenarios.

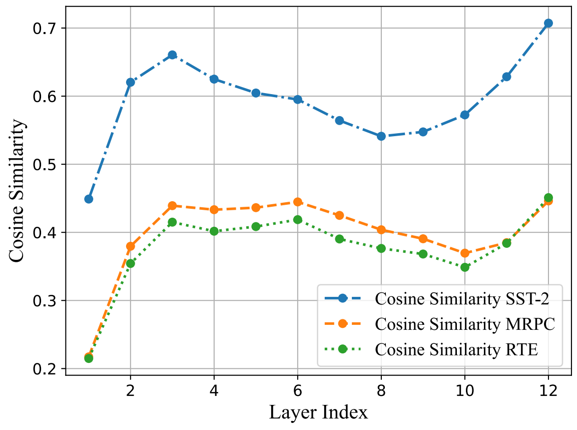


Figure 1: Tokens (cosine similarity) of different layers.

To address these issues, model compression techniques (Lin et al., 2020), including knowledge distillation (Hinton et al., 2015; Jiao et al., 2020), pruning (Sanh et al., 2020; Michel et al., 2019) and quantization (Zafrir et al., 2019), have been applied to PLMs to accelerate inference. However, these methods permanently remove some components of the model, leading to an inevitable decline in performance. Additionally, the complexity of these models cannot be adjusted based on the requirements of different hardware since the compression is implemented before the deployment. Conversely, several studies have suggested the use of conditional computation algorithms, such as the early exiting (EE) algorithm (Schwartz et al., 2020; Xin et al., 2020), in which each input sample uses a different part of the model so that the computation or latency is reduced on average. Based on this, an EE classifier is added between transformer layers. The shallow layer allows the *easy* samples to exit earlier without the need to perform computations up to the final layer. Computations related to deeper layers should only be reserved for *hard* samples.

There are two main limitations of existing EE

[*]Corresponding authors.

methods. For existing EE methods, there has been a focus on the exploration of accelerated inference for *easy* samples. However, for *hard* samples, acceleration inference is not achieved and still needs to be performed through the last layer. Furthermore, the computation consumption is quadratically proportional to the input sequence length. As a result, the average acceleration inference is limited.

In addition, the existing early exiting approach is only performed on the vertical perspective, and the operations of the horizon perspective are ignored. That is, each transformer encoder layer is regarded as a basic unit. The model determines how many transformer encoder layers are needed for inference but ignores which tokens in each layer are required to participate in the computation. Recent studies (Ethayarajh, 2019; Klafka and Ettinger, 2020) have shown that sequence tokens from the horizontal perspective have high redundancy. To further verify the redundant information from the horizontal perspective, we use the cosine distance to calculate the similarity between tokens in each layer, as shown in Figure 1. The token similarity in the shallow layer is low, indicating that the shallow layer is not fully encoded. The horizontal perspectives of deeper layers, especially the last layer, show high redundancy. This suggests that the previous EE approaches ignored the possibility of reducing the horizontal perspective redundancy to accelerate inference. Therefore, these EE approaches yield a suboptimal accelerated inference. To address this issue, an intuitive idea is to reduce the length of the sequence token as the layers grow. TR-BERT (Ye et al., 2021) uses a reinforcement learning (RL) method to select the tokens that need to be dynamically reduced. However, additional training is needed, which increases the cost of training. In addition, changes to the device platform require retraining of these components, which may greatly limit the application scenarios.

In this paper, we propose a unified horizontal and vertical multi-perspective early exiting framework to reduce the computation and latency for fast inference of transformer-based models. This framework contains layer-wise EE for the vertical perspective and sequential token-wise EE for the horizontal perspective. In sequential token-wise EE, different tokens are forced to exit at different layers to reduce computation by emphasizing the informative tokens to the downstream task. Conversely, the tokens with less information are truncated, which isolates them from the following computation. To measure the importance of tokens, we use class attention to learn the amount of prediction information. Due to the instability of the class attention in the shallow layer, recycling class attention memory is used to enhance the reuse of information across layers and to better identify informative tokens. The tokens with less information do not completely exit. Instead, they take a weighted fusion into an EE fusion token for subsequent calculations so that additional parameters and computations are not introduced.

For the vertical perspective EE, we introduce EE classifiers in the middle of two transformer layers. Samples exit early at the shallow layer when the confidence level is higher than the threshold and skip computation in other layers. This allows the input samples to be predicted with a shallow EE classifier rather than a deep EE classifier. Since the shallow EE classifier is weakly expressive, we introduce weighted ensemble self-distillation and recycling EE classifier memory to enhance the performance of the shallow EE classifier. Thus, a performance guarantee is provided at high acceleration inferences. The unified multi-perspective early exiting framework has horizontal sparsity and vertical sparsity. Based on extensive experiments, it is shown that the proposed model reduces computation and improves acceleration inference while maintaining high performance.

The rest of the paper is organized as follows. In Section 2, the preliminaries of PLMs are introduced. In Section 3, a detailed description of the proposed methods is provided. The empirical experiments are reported and analyzed in Section 4. Section 5 briefly introduces the previous studies on acceleration inference for PLMs. Conclusions are finally drawn in Section 6.

2 Preliminaries

Pretrained Language Model. A PLM consists of L transformer encoders (Vaswani et al., 2017). The sequence token $X = [x_{[\text{CLS}]}, x_1, x_2, ..., x_N]$ of the text processed by the tokenizer is input into the PLM. N is the number of sequence tokens (without the `[CLS]` token), and the `[CLS]` token denote the global text information and final classification. The transformer contains multi-head self-attention (MHSA) and feed-forward network (FFN) modules. The transformer coding process of each layer is defined as

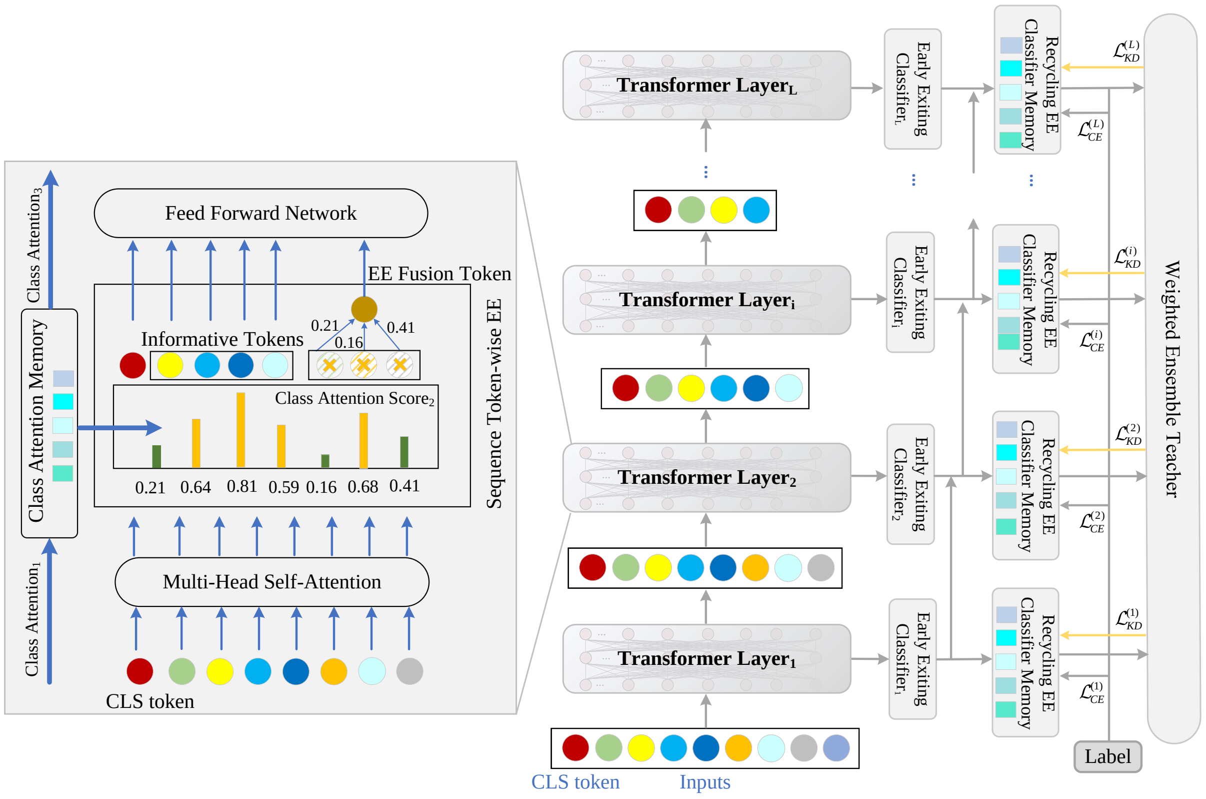


Figure 2: Overall architecture of the unified multi-perspective early exiting framework for accelerating.

$$Z^l = \text{Transformer}^l(h_{[\text{CLS}]}^{l-1}, h_1^{l-1}, ..., h_N^{l-1}) \quad (1)$$

where $Z_l \in \mathbb{R}^{n \times d}$ denotes the output feature of the l-th transformer, and N denotes the number of tokens in the l-th layer of MHSA and d denotes the model hidden size. The MHSA module is defined as

$$\begin{aligned}&\text{MHSA}(Z_{l-1})\\&\quad= \text{Concat}[\text{HA}_1(Z_{l-1}); ...; \text{HA}_H(Z_{l-1})]W^o\end{aligned} \quad (2)$$

where $W^o \in \mathbb{R}^{(H \cdot d_h) \times d}$ is the output projection, H is the number of multi-head and $d_h = d/H$. The single-head self-attention is denoted as

$$\begin{aligned}\text{HA}_h(Z_{l-1}) &= \text{Attention}(Q_h, K_h, V_h)\\&= \text{softmax}(\frac{Q_h K_h^T}{\sqrt{d_h}})V_h\end{aligned} \quad (3)$$

where $Q_h = Z_{l-1}W_h^Q, K_h = Z_{l-1}W_h^K, V_h = Z_{l-1}W_h^V$ are the matrices that package the query, key and value, and $W_h^{\{Q,K,V\}} \in \mathbb{R}^{d \times d_h}$ is the linear transformation. Two fully connected layers are applied on the output of the MHSA, and the FFN module is defined as

$$\text{FFN}(\tilde{Z}_l) = \sigma(\tilde{Z}_l W_1 + b_1)W_2 + b_2 \quad (4)$$

where σ denotes the GELU activation function, $\tilde{Z}_l$ is the MHSA output,$W_1 \in \mathbb{R}^{d \times 4d}$ and $W_2 \in \mathbb{R}^{4d \times d}$ are the projection matrix.

Computational Complexity. Given an input sequence $N \times D$, N is the number of input tokens and D is the embedding dimension of each token. From the above, the computational complexity of MHSA is $O(4ND^2 + 2N^2D)$ and FNN is $O(8ND^2)$. The total computational complexity of PLMs is $O(12ND^2 + 2N^2D)$. The complexity of MHSA and FFN is quadratically and linearly for the length N of the input sequence. Furthermore, existing studies (Ethayarajh, 2019) have shown that sequence tokens have high redundancy. An intuitive idea is to reduce redundant tokens to reduce computation and improve acceleration inference.

3 Multi-Perspective Early Exiting

Figure 2 shows an overview of the proposed unified multi-perspective early exiting framework, which consists of layer-wise and sequential token-wise early exiting. Layer-wise and sequential token-wise early exiting are orthogonal methods that exit unnecessary computations early by the layer-wise and token-wise of the model, respectively, to achieve faster inference. Therefore, we combine these two different perspectives to achieve accelerated inference and minimize computational cost.

3.1 Sequential Token-wise Early Exiting

The inference is accelerated by early exiting of the sequential token according to class attention, as shown in Figure 2. In PLMs, the `[CLS]` token is used for classification. According to Eq.(3), MHSA outputs of the `[CLS]` token, i.e., $h^l_{\text{[CLS]}}$, can be regarded as a weighted summation of value vectors with attention scores, which is defined as

$$\begin{aligned} A^l_{class} &= \text{MeanPooling}(\sum_h^H \text{HA}_h(Z^0_l)) \\ h^l_{\text{[CLS]}} &= V^l \cdot A^l_{class} \end{aligned} \tag{5}$$

where $\text{HA}_h(Z^0_l)$ is the *h*-th attention head and A^l_{class} denotes the mean values of all attention heads corresponding to the `[CLS]` token. Here, A^l_{class} represents the interaction of the `[CLS]` token for all other tokens and determines how much information from the *i*-th token is contributed to the `[CLS]` token. It is intuitive to denote A^l_{class} as the information that the `[CLS]` token transferred for downstream tasks. Therefore, each token is assigned to be informative or less informative according to the class attention score A^l_{class}. Due to the instability and weak representation of the shallow layer class attention, recycling class attention memory is used to enhance the reuse of information across layers and to better identify informative tokens and is defined as

$$A^l_m = (1-\beta)A^{l-1}_m + \beta A^l_{class} \tag{6}$$

where A^l_m is a class attention memory used to record the historical values of A^l_{class}. β denotes the weight that balance the class attention with different layers. We keep *m* informative tokens h^l_{in} for *m*-highest score in A^l_m, which is formulated as

$$m = N(1 - R_{ee}) \tag{7}$$

where R_{ee} denotes the sequence EE ratio. For *p* early exiting tokens that are less informative, $p = N - m$. Less informative tokens do not completely exit early from the sequence. To reduce the missing information due to early exiting, the less informative tokens are fused into an EE fusion token according to the corresponding class attention weights. The EE fusion token and the informative tokens are spliced in the subsequent computation, which is denoted as

$$\begin{aligned} h^l_{eef} &= \alpha_i h^l_i \\ \tilde{Z}_l &= \text{Concat}[h^l_{CLS}; h^l_{in}; h^l_{eef}] \end{aligned} \tag{8}$$

where h^l_i represents tokens with less informative and α_i denotes the corresponding class attention weight. [;] indicates a connection operation.

3.2 Layer-wise Early Exiting

In addition to sequential token-wise EE, layer-wise EE is used to further reduce the layers and computation of PLMs from another perspective, as shown in Figure 2. The EE classifier is added between every two transformer layers. The logit distribution C^l_z of the EE classifier is defined as

$$C^l_z = \tanh(W^l_z h^l_{\text{[CLS]}} + b^l_z) \tag{9}$$

where $W^l_z \in \mathbb{R}^{C\times d}$ and $b^l_z \in \mathbb{R}^J$ represent the weight and bias of the *l*-th EE classifier, respectively, and *J* denotes the number of classes.

Each EE classifier in the existing EE approaches makes its decision independently, ignoring information from previous EE classifiers and discarding potentially valuable information. Recycling EE classifier memory is used to fuse useful information from previous EE classifiers into the current EE classifier and obtain useful information from different layers of EE classifiers to help make accurate and reliable predictions and is defined as

$$\begin{aligned} C^l_{zm} &= (1-\lambda)\text{C}^{l-1}_{zm} + \lambda C^l_z \\ \hat{y}^l_z &= \text{softmax}(C^l_{zm}) \end{aligned} \tag{10}$$

where $\hat{y}^l_z$ denotes the probability distribution of the EE classifier and C^l_{zm} denotes EE classifier memory. The cross-entropy training objectives of these EE classifiers are defined as follows:

$$\mathcal{L}_{CE} = -\sum_{l=1}^{L} \mathbb{I}(y) \circ \log(\hat{y}^l_z) \tag{11}$$

where *y* denotes the corresponding ground-truth label, $\mathbb{I}(y)$ denotes a one-hot label with the *y*-th element being one, and $\circ$ represents an element-wise multiplication operation.

The weak expressiveness of the shallow EE classifier and the lack of higher-level semantic information lead to poor performance of the shallow EE classifier. Thus, weighted ensemble self-distillation is applied to further improve the representation abilities and obtain rich semantic information from the

EE classifiers. Based on the importance of students, i.e., EE classifiers, to the final prediction, an attention mechanism is used to combine all students into a teacher model and is defined as

$$\eta^l = \frac{\exp(V^T \tanh(W_\eta C_{zm}^l + b_\eta))}{\sum_{l=1}^{L} \exp(V^T \tanh(W_\eta C_{zm}^l + b_\eta))} \qquad (12)$$
$$t = \sum_{l=1}^{L} \eta^l C_{zm}^l$$

where V is the context vector, W_α and b_α are the weight and bias, respectively, and t is the teacher for self-distillation, which is accomplished by minimizing the distribution between the teacher model and the student models and is defined as

$$\mathcal{L}_{KD} = \sum_{l=1}^{L} \tau^2 \mathrm{KL}(\mathrm{softmax}(C_{zm}^l/\tau)||\mathrm{softmax}(t/\tau)) \qquad (13)$$

where $\mathrm{KL}(\cdot||\cdot)$ is the Kullback-Leibler divergence function and τ is the temperature parameter that measures the smoothness of the distribution. τ^2 compensates for the size of the gradient scaled by the soft target, ensuring that there is no negative impact on the gradient size. The overall loss of the proposed MPEE is denoted as

$$\mathcal{L} = \mathcal{L}_{CE} + \mathcal{L}_{KD} \qquad (14)$$

In MPEE, the informative token is first selected and the less informative token exits. Furthermore, less informative tokens are fused into EE fusion tokens to continue the computation in the following layer. Then, the decision of whether to exit the whole computation at the current layer is based on whether the entropy value of the current EE classifier is greater than the preset threshold F.

4 Experiments

4.1 Datasets

To evaluate the acceleration effect of the proposed method on the inference of PLMs, we conducted experiments on the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019), which includes SST-2 (Socher et al., 2013) for sentiment analysis containing sentences from movie reviews. QQP and MRPC (Dolan and Brockett, 2005) for similarity and paraphrase. QNLI (Rajpurkar et al., 2016) and MNLI (Williams et al., 2018) for natural language inference, and the RTE for textual entailment.

4.2 Performance and Acceleration Metrics

For performance metrics, accuracy (Acc) and F_1-score (F_1) were used as evaluation metrics for MRPC. For all other tasks, accuracy was used as the evaluation metric.

For acceleration metrics, following Ye et al. (2021), we used the number of floating operations (FLOPs) as the acceleration inference ratio. Because FLOPs evaluate acceleration inference independently of the runtime environment and hardware, it is easy to compare the acceleration inference of the models. The acceleration ratio ρ is defined as the rate of the total FLOPs by the original model to the FLOPs actually executed by the model and is defined as

$$\rho = \frac{FLOPs_{exec}}{FLOPs_{total}} \qquad (15)$$

where FLOPs are the sum of floating operations of all inference sample. The individual sample FLOPs are calculated during inference. The performance with different acceleration ratios is obtained by adjusting the threshold F and sequence EE ratio R_{ee}.

4.3 Baselines

To demonstrate the effectiveness of the proposed method, several previous methods are implemented for comparison for PLMs accelerated inference. The baselines include the BERT backbone model, model compression methods, early exiting methods, and sequence reduction methods. Other model compression methods, such as BERT-6L, DistilBERT (Sanh et al., 2019), BERT-PKD (Sun et al., 2019) and LayerDrop (Fan et al., 2019) are also applied. In addition, early exiting methods, including DeeBERT (Xin et al., 2020), FastBERT (Liu et al., 2020) and PABEE (Zhou et al., 2020) and sequence length reduction methods, including PoWER-BERT (Goyal et al., 2020) and TR-BERT (Ye et al., 2021) are introduced for comparison. The implementation and the optimal hyperparameters are fine-tune by using a grid search strategy to train the baseline. To make a fair comparison, we use the same parameters for the baseline and calculated PoWER-BERT in a single sample of FLOPs.

4.4 Implementation Details

The implementation of the proposed model is based on HuggingFace's transformers (Wolf et al., 2020). `BERT-base-uncased` is used as the backbone

Model	MNLI-m		SST-2		MRPC		QQP		MNLI-mm		QNLI		RTE	
	Acc	ρ	Acc	ρ	F_1/Acc	ρ	Acc	ρ	Acc	ρ	Acc	ρ	Acc	ρ
BERT-Base	83.9	1.00×	92.1	1.00×	90.3/86.3	1.00×	91.1	1.00×	83.8	1.00×	91.2	1.00×	71.1	1.00×
BERT-6L	80.3	2.00×	90.1	2.00×	86.6/80.3	2.00×	88.0	2.00×	80.6	2.00×	86.9	2.00×	64.5	2.00×
DistilBERT	79.0	2.00×	90.7	2.00×	87.5/-	2.00×	88.5	2.00×	81.5	2.00×	85.3	2.00×	59.9	2.00×
BERT-PKD	81.3	2.00×	91.3	2.00×	85.7/-	2.00×	88.4	2.00×	-	2.00×	88.4	2.00×	66.5	2.00×
LayerDrop	80.7	2.00×	90.7	2.00×	85.9/-	2.00×	88.3	2.00×	-	2.00×	88.4	2.00×	65.2	2.00×
DeeBERT	74.3	1.92×	90.4	1.97×	86.5/79.9	1.95×	88.2	1.96×	74.5	1.91×	86.4	1.95×	63.8	1.96×
FastBERT	74.5	1.93×	90.8	1.98×	86.6/80.3	1.96×	88.4	1.96×	74.8	1.92×	86.7	1.92×	65.0	1.98×
PABEE	79.0	1.95×	90.1	1.98×	86.8/80.4	1.80×	89.1	1.98×	79.5	1.93×	87.6	1.85×	63.9	1.96×
PoWER-BERT	81.9	2.47×	91.1	2.33×	88.4/82.8	3.11×	89.7	3.27×	81.6	2.52×	89.3	2.21×	68.2	2.56×
TR-BERT	82.1	2.96×	90.3	1.63×	84.9/75.9	1.23×	89.4	3.15×	81.9	3.02×	88.2	1.57×	67.8	2.63×
MPEE	**82.6**	**3.38×**	**91.7**	**4.4×**	**88.7/83.3**	**3.47×**	**90.0**	**4.33×**	**82.3**	**3.41×**	**90.2**	**2.45×**	**69.0**	**3.55×**

Table 1: Compare experimental results in the baseline methods with BERT backbone on GLUE.

of our model, where the transformer layer is 12 layers, the attention head is 12, and the hidden dimension is 768. The training batch sizes are 64 and 128. The inference batch size for the proposed model and baseline is 1. We use the grid search to find 0.7 for R_{ee} and 0.5 and 0.9 for β and λ, respectively. The model is optimized using Adam, and the learning rate is 2e-5.

4.5 Comparative Results

Table 1 shows the performance and acceleration inference of the proposed method and baseline methods. The proposed MPEE method outperforms all baseline methods in improving acceleration inference while maintaining better performance, verifying the effectiveness of the proposed model. The accuracy degradation is within a relatively small range compared to BERT, while the acceleration inference is significantly improved on most datasets. Especially, the acceleration ratio ρ is 4.4× on SST-2.

Further, the proposed MPEE method has 4.4× acceleration ratio ρ on SST-2, but still maintains a 91.6% accuracy. The proposed MPEE outperforms the existing EE methods due to the shallow layer learns higher layer semantic information, allowing it to improve its expressiveness, which leads to decreased model performance. Ignoring horizontal perspective redundancy limits accelerated inference. On SST-2, MPEE has a 1.4% higher performance than TR-BERT with respect to accuracy, but still has a 2.86× faster acceleration ratio ρ on inference. TR-BERT discards some token information and reduces the sequence length in only two layers to ease the convergence of the model with RL and reduce the search space. Therefore, its acceleration effect is relatively insignificant, thus its performance is degraded. PoWER-BERT completely removes the token, leading to partial information loss and decreasing model performance. This sequence reduction approach requires the computation to be executed through the last layer and ignores vertical perspective redundancy, resulting in limited acceleration. The proposed approach significantly reduces computation and accelerates inference by simultaneous early exits from both horizontal and vertical perspectives, and the EEs of the two perspectives are orthogonal. The EE token information is preserved, and the performance of the model is maintained by weighted self-distillation.

4.6 Ablation Study

We conducted several ablation studies to better demonstrate the effectiveness of the proposed modules, including recycling class attention memory (Att-memory), EE fusion token (EE-fusion), recycling EE classifier memory (EE-memory), weighted self-distillation (WSD), layer-wise EE (Layer-EE) and sequential token-wise EE (Token-EE). To demonstrate their effectiveness, we have removed each module individually to show that the performance is degraded, as observed in Table 2.

The removal of EE fusion leads to performance degradation because most of the semantic information is lost in this setup. Similarly, the absence of Att-memory leads to performance degradation, proving the ability of proposed model to effectively enhance the selected informative tokens by combining class attention at different layers. The removal of Token-EE and Layer-EE leads to a significant reduction in acceleration. The rational reason for these results is that vertical and horizontal perspec-

Model	SST-2		MRPC		RTE		QQP		QNLI	
	Acc	ρ	F_1/Acc	ρ	Acc	ρ	Acc	ρ	Acc	ρ
BERT	92.1	1.00×	90.3/86.3	1.00×	71.1	1.00×	91.1	1.00×	91.2	1.00×
MPEE	**91.7**	**4.40×**	**88.7/83.3**	**3.47×**	**69.0**	**3.55×**	**90.0**	**4.33×**	**90.2**	**2.45×**
w/o Att-memory	91.4	4.22×	88.3/82.9	3.26×	68.1	3.26×	89.5	4.12×	89.7	2.26×
w/o EE-memory	91.3	4.27×	88.2/83.1	3.18×	68.3	3.19×	89.3	4.26×	89.8	2.32×
w/o WSD	91.2	3.92×	87.8/83.0	2.96×	67.9	3.08×	89.1	4.02×	89.5	2.06×
w/o EE-fusion	91.0	4.18×	88.0/82.5	3.23×	68.4	3.28×	89.2	4.08×	89.6	2.37×
w/o Layer-EE	91.3	2.43×	86.8/81.1	2.62×	68.2	2.43×	87.3	2.43×	90.0	1.74×
w/o Token-EE	91.5	2.47×	87.2/81.5	2.83×	68.6	1.72×	89.9	2.88×	89.9	1.71×

Table 2: Results of the ablation study of the proposed MPEE model.

Method	SST-2		QNLI	
	Acc	ρ	Acc	ρ
Class Attention	91.6	4.40×	90.1	2.45×
All-token Attention	91.3	4.27×	89.7	2.36×

Table 3: Different methods of selecting the informative token.

tive EE simultaneously lead to significant acceleration inference.

The informative tokens were selected based on class attention, which is convenient and does not introduce additional parameters and calculations. To show the validity of class attention, we compare it with another choice of informative token. Another way to select informative tokens is to sum the attention weights of all tokens, which is noted as the all token attention, as shown in Table 3. Using class attention has better performance, especially for similarity acceleration inference. The results show that class attention is a better guide for selecting informative tokens because the model prediction is based on the [CLS] token and averaging all token attention when there is too much redundant information will dilute the important token weights down to the selection informative token confusion.

4.7 Analysis

Performance-Acceleration Tradeoff. To further demonstrate the performance and efficiency tradeoff between the proposed method and baseline methods, Figure 3 shows the performance and acceleration tradeoff curves. Different accelerations can be obtained by changing the threshold F and sequence EE ratio R_{ee}. In addition, in TR-BERT, the parameters need to be changed to retrain to obtain

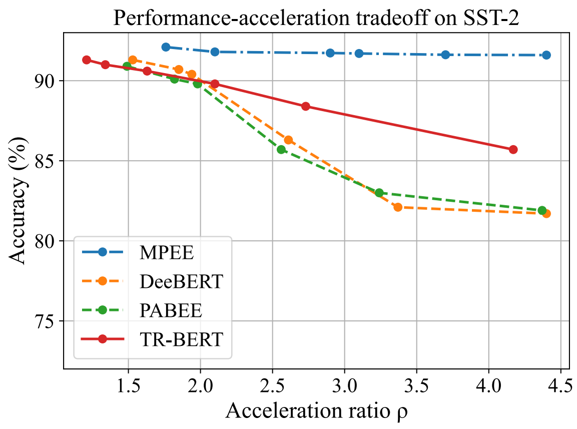


Figure 3: Performance-acceleration tradeoff for MPEE, DeeBERT, TR-BERT and PABEE.

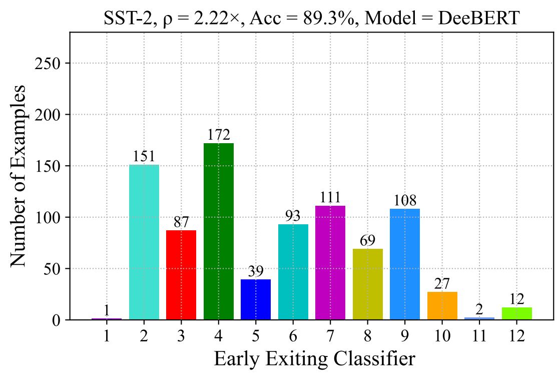
different accelerations, which increases the computational resources. This limits the application of TR-BERT on different mobile devices. As shown in Figure 3, the performance of the existing EE methods decreases sharply with increasing acceleration ratios, which is also due to the poor performance of shallow EE classifiers caused by the weak representation. MPEE outperforms TR-BERT in terms of acceleration and performance. The proposed method simultaneously accelerates inference in multiple perspectives. Preserving EE token information and weighted self-distillation maintains the performance of the model. Another reason is that TR-BERT reduces the sequence length in two layers, which requires discarding more tokens to reach similar acceleration, while we early exit tokens in more layers. MPEE can achieve a better tradeoff between performance and efficiency.

Sample Distribution of Early Exiting. To better demonstrate the superiority of the proposed method over other methods in accelerating the inference of PLMs, Figure 4 shows the statistical information of the number of samples exiting early in different layers. The proposed MPEE tends to exit the model inference at an earlier classifier than DeeBERT with a higher model performance. This indicates that the proposed model can exit early while maintaining high performance. Another observation is that nearly half of the samples of the proposed model exit the model immediately at the first layer, but the acceleration ratio ρ is 4×. This is because the horizontal perspective EE further reduces the computation and improves the acceleration inference and because the multi-perspective early exiting is orthogonal.

Performance of Different Class Attention Scores. To verify whether the class attention has the ability to discriminate the information token. We exit the tokens of the top 70% and the bottom 70% class attention score tokens separately to obtain the model performance, as shown in Figure 5. The performance of tokens with high scores is better than the performance of tokens with low scores. The selection of the bottom 70% class attention score tokens leads to increasingly low performance, which is due to the model selecting tokens with less information layer by layer. This indicates that class attention can be used to obtain the more informative tokens.

5 Related Work

Model Compression. Knowledge distillation (Sun et al., 2020) refers to the training of smaller student models using the knowledge supervision of pre-trained larger teacher models. The student model uses fewer layers to learn knowledge from the teacher's hidden units and logits, e.g., DistilBERT (Sanh et al., 2019) and BERT-PKD (Sun et al., 2019). Pruning (Wang et al., 2020) refers to the removal of less important weights or computational units. Voita et al. (2019) analyzed multi-head self-attention importance and removed it. Sajjad et al. (2020) discarded unimportant layers in the fine-tuning process. PoWER-BERT (Goyal et al., 2020) progressively reduces the sequence length and accelerates the BERT model. However, ignoring the deleted token leads to information loss and thus reduces the performance of the model. Ye et al. (2021) proposed using reinforcement learning to select unimportant tokens to reduce the sequence length. This approach requires additional strategies to converge the model. Quantification (Shen et al., 2020) is the process of reducing the number of bits needed to represent the model weights. Gao et al. (2021) proposed to quantize the activation function and the weight parameters simultaneously to reduce quantization errors.



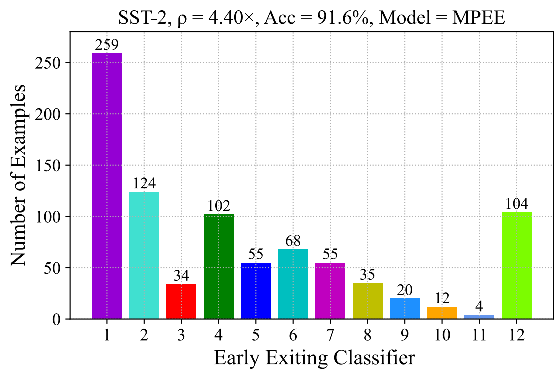


Figure 4: Distribution of different performances of early exiting classifiers on SST-2.

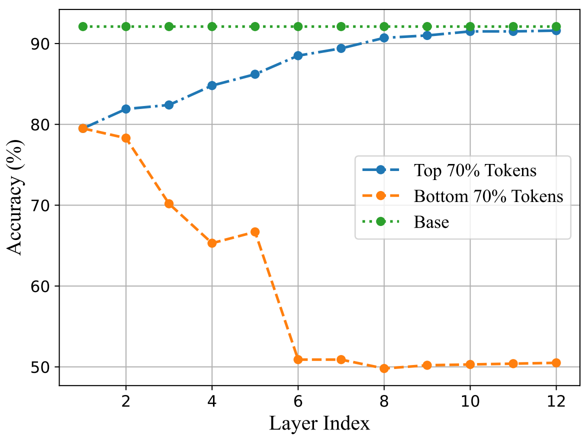


Figure 5: The performance of the top 70% and bottom 70% class attention score on SST-2.

Early Exiting. The early exiting approach refers to allowing different samples to exit early in different layers depending on the properties of the input samples. DeeBERT (Xin et al., 2020) performs early prediction by introducing multiple EE classifiers and exits early if the confidence is greater than the threshold, and conversely passes to the next layer to continue the computation. FastBERT (Liu et al., 2020) uses self-distillation to train EE classifiers. PABEE (Zhou et al., 2020) achieves early exit inference when EE classifier predictions are held continuously constant.

6 Conclusions

In this paper, we propose a unified multi-perspective early exiting framework that significantly reduces the computation cost and improves acceleration inference within a small performance loss. The multi-perspective early exit framework includes horizontal and vertical perspectives early exiting. It has horizontal sparsity and vertical sparsity for faster inference. Extensive experimental results show that compared to previous approaches, the proposed model provides a better tradeoff between model performance and inference efficiency. Future work attempt to extend the proposed approach to vision and language pretrained models, taking into account the properties of different modalities.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (NSFC) under Grants No. 61966038, and the Ministry of Science and Technology, Taiwan, ROC, under Grant No. MOST 111-2628-E-155-001-MY2. The authors would like to thank the anonymous reviewers for their constructive comments.

References

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 (NAACL-HLT 2019)*, pages 4171–4186.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP 2005)*, pages 9–16.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMO, and GPT-2 embeddings. 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 2019)*, pages 55–65.

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

Zhifu Gao, Yiwu Yao, Shiliang Zhang, Jun Yang, Ming Lei, and Ian McLoughlin. 2021. Extremely low footprint end-to-end ASR system for smart device.

Saurabh Goyal, Anamitra Roy Choudhury, Saurabh M. Raje, Venkatesan T. Chakaravarthy, Yogish Sabharwal, and Ashish Verma. 2020. PoWER-BERT: Accelerating BERT inference via progressive word-vector elimination. In *Proceedings of 37th International Conference on Machine Learning (ICML 2020)*, volume PartF16814, pages 3648–3657.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling bert for natural language understanding. In *Findings of the Association for Computational Linguistics (EMNLP 2020)*, pages 4163—-4174.

Josef Klafka and Allyson Ettinger. 2020. Spying on your neighbors: Fine-grained probing of contextual embeddings for information about surrounding words. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020)*, pages 4801–4811.

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

Zi Lin, Jeremiah Liu, Zi Yang, Nan Hua, and Dan Roth. 2020. Pruning redundant mappings in transformer models via spectral-normalized identity prior. In *Findings ofthe Association for Computational Linguistics (EMNLP 2020)*, pages 719–730.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Haotang Deng, and Qi Ju. 2020. FastBERT: A self-distilling BERT with adaptive inference time.

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*.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Proceedings of the 33th International Conference on Neural Information Processing Systems (NeurIPS-2019)*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 2383–2392.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. 2020. Poor man's BERT: Smaller and faster transformer models hassan. *arXiv preprint arXiv:2004.03844*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Victor Sanh, Thomas Wolf, and Alexander Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning. In *Advances in Neural Information Processing Systems (NeurIPS 2020)*, volume 33, pages 20378–20389.

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 (ACL 2020)*, pages 6640–6651.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian May, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. 2020. Q-BERT: Hessian based ultra low precision quantization of BERT. In *Proceedings of The 34th AAAI Conference on Artificial Intelligence (AAAI 2020)*, pages 8815–8821.

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank richard. In *In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, pages 1631–1642.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for BERT model compression. In *Proceedings ofthe 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP/IJCNLP 2019)*, pages 4323–4332.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. MobileBERT: A compact task-agnostic BERT for resource-limited devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020)*, pages 2158–2170.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31th International Conference on Neural Information Processing Systems (NeurIPS 2017)*, pages 5998–6008.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)*, pages 5797–5808.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of The 7th International Conference on Learning Representations (ICLR 2019)*, pages 353–355.

Ziheng Wang, Jeremy Wohlwend, and Tao Lei. 2020. Structured pruning of large language models. In *Proceedings ofthe 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020)*, pages 6151–6162.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2018)*, pages 1112–1122.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick Von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

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

Deming Ye, Yankai Lin, Yufei Huang, and Maosong Sun. 2021. TR-BERT: Dynamic token reduction for accelerating BERT inference. In *Proceedings ofthe 2021 Conference ofthe North American Chapter ofthe Association for Computational Linguistics (ACL 2021)*, pages 5798–5809.

Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. 2019. Q8BERT: Quantized 8Bit BERT. *arXiv preprint arXiv:1910.06188*.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. 2020. BERT loses patience: Fast and robust inference with early exit. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS 2020)*, pages 18330—-18341.