A Global Past-Future Early Exit Method for Accelerating Inference of Pre-trained Language Models

Kaiyuan Liao[†][*], **Yi Zhang**[‡][*], **Xuancheng Ren**[‡], **Qi Su**[‡§], **Xu Sun**[†‡], **Bin He**[⋆]
[†] Center for Data Science, Peking University
[‡] MOE Key Laboratory of Computational Linguistics, School of EECS, Peking University
[§] School of Foreign Languages, Peking University
[⋆] Huawei Noah's Ark Lab
{kyliao,zhangyi16,renxc,sukia,xusun}@pku.edu.cn
hebin.nlp@huawei.com

Abstract

Early exit mechanism aims to accelerate the inference speed of large-scale pre-trained language models. The essential idea is to exit early without passing through all the inference layers at the inference stage. To make accurate predictions for downstream tasks, the hierarchical linguistic information embedded in all layers should be jointly considered. However, much of the research up to now has been limited to use local representations of the exit layer. Such treatment inevitably loses information of the unused past layers as well as the high-level features embedded in future layers, leading to sub-optimal performance. To address this issue, we propose a novel Past-Future method to make comprehensive predictions from a *global* perspective. We first take into consideration all the linguistic information embedded in the past layers and further engage the future information which is originally inaccessible for predictions. Extensive experiments demonstrate that our method outperforms previous early exit methods by a large margin, yielding better and robust performance[1].

1 Introduction

Pre-trained language models (PLMs), e.g., BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019) and XLNet (Yang et al., 2019), have obtained remarkable success in a wide range of NLP tasks. Despite their impressive performance, PLMs are usually associated with large memory requirement and high computational cost. Such drawbacks slow down the inference and further encumber the application of PLMs in the scenarios where inference time and computation budget are restricted.

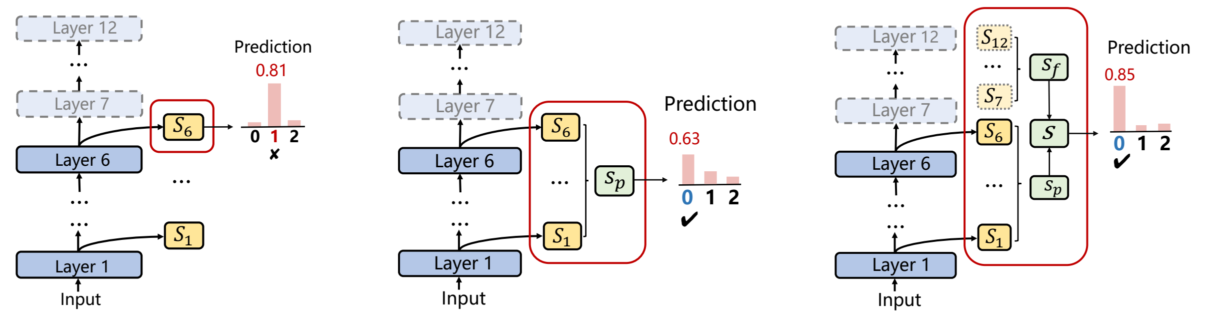
To address this issue, a growing number of studies focusing on improving model efficiency have emerged recently. Particularly, Kaya et al. (2019) point out that the current over-parameterized models conduct excessive computation for simple instances, which is actually undesirable and computationally wasteful. In light of this observation, an increasing amount of work seeks various early exit methods, of which the basic idea is to exit early without passing through the entire model during inference. Concretely, for NLP tasks, they couple branch classifiers with each layer of the pre-trained language models and stop forward propagation at an intermediate layer. Then the current branch classifier makes a prediction based on the representation of the token that is used as the aggregated sequence representation for classification tasks and is referred to as the state of the layer in this work.

However, existing work on early exit has two major drawbacks. First, existing work (Xin et al., 2020; Zhou et al., 2020) uses only local states in the early exit framework. They inevitably lose valuable features that are captured by passed layers but are ignored for prediction, leading to less reliable prediction results. Moreover, these methods abandon the potentially useful features captured by the future layers that have not been passed, which may hurt the performance of the instances requiring high-level features embedded in the deep layers. Consequently, their performance dramatically declines when the inference exits earlier for a higher speed-up ratio.

These two major drawbacks hinder the progress of early exit research and motivate us to develop a new mechanism using the hierarchical linguistic information embedded in all layers (Jawahar et al., 2019) from a global perspective. However, up to now, a global early exit mechanism remains a under-explored challenging problem. We extend the existing methods to their corresponding global versions and find that naive global strategies only result in poor performance. Meanwhile, the future states are originally inaccessible in the early exit

[*]Equal contribution
[1]The code is available at https://github.com/lancopku/Early-Exit

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 2013–2023
June 6–11, 2021. ©2021 Association for Computational Linguistics



(a) DeeBERT with local early exit method

(b) The Global Past version of our early exit method

(c) The Global Past-Future version of our early exit method

Figure 1: Comparison of the local early exit method and our proposed method. The red rectangles highlight the states that the models rely on to make predictions.

framework, which also remains a bottleneck for a global prediction considering both past and future states.

In this paper, we focus on the aforementioned problems and first put into practice a global Past-Future early exit mechanism. The term *global* is two-fold: (1) instead of using one or several local state(s) for prediction in previous work, all the available past states are effectively incorporated in our method; (2) furthermore, to grasp the features embedded in the deep layers, the originally inaccessible future states are approximated by imitation learning and are also engaged for prediction. The comparison of the previous method and our method is illustrated in Figure 1. By combining both past and future states, our model is able to make more accurate predictions for downstream tasks.

Extensive experiments reveal that the proposal significantly outperforms previous early exit methods. Particularly, it surpasses the previous methods by a large margin when the speed-up ratio is relatively high. In addition, extensive experiments with different pre-trained language models as backbones demonstrate consistent improvement over the baseline methods, which verifies the generality of our method.

To summarize, our contributions are as follows:

- We propose a set of global strategies which effectively incorporate all available states and they achieve better performance compared to the existing naive global strategies.
- Our early exit method first utilizes the future states which are originally inaccessible at the inference stage, enabling more comprehensive global predictions.
- Experiments show that our proposal achieves better performance compared to the previous state-of-the-art early exit methods.

2 Related Work

Large-scale pre-trained language models (Devlin et al., 2019; Liu et al., 2019) based on the Transformer (Vaswani et al., 2017) architecture demonstrate superior performance in various NLP tasks. However, the impressive performance is on the basis of massive parameters, leading to large memory requirement and computational cost during inference. To overcome this bottleneck, increasing studies work on improving the efficiency of over-parameterized pre-trained language models.

Knowledge distillation (Hinton et al., 2015; Turc et al., 2019; Jiao et al., 2019; Li et al., 2020a) compacts the model architecture to obtain a smaller model that remains static for all instances at the inference stage. Sanh et al. (2019) focus on reducing the number of layers since their investigation reveals variations on hidden size dimension have a smaller impact on computation efficiency. Sun et al. (2019) learn from multiple intermediate layers of the teacher model for incremental knowledge extraction instead of only learning from the last hidden representations. Further, Wang et al. (2020) design elaborate techniques to drive the student model to mimic the self-attention module of teacher models. Xu et al. (2020) compress model by progressive module replacing, showing a new perspective of model compression. However, these static model compression methods treat the instances requiring different computational cost without distinction. Moreover, they have to distill a model from scratch to meet the varying speed-up ratio requirements.

To meet different constraints for acceleration, another line of work studies instance-adaptive methods to adjust the number of executed layers for different instances. Li et al. (2020b) select models in different sizes depending on the difficulty of input instance. Besides, early exit is a practical method to adaptively accelerate inference and is first proposed for computer vision tasks (Kaya et al., 2019; Teerapittayanon et al., 2016). Elbayad et al. (2020); Xin et al. (2020); Schwartz et al. (2020) follow the essential idea and leverage the method in NLP tasks. To prevent the error from one single classifier, Zhou et al. (2020) make the model stop inference when a cross-layer consistent prediction is achieved. However, researches on the subject has been mostly restricted to only use the local states around the exit layer.

3 Method

We first introduce the strategies to incorporate multiple states and the imitation learning method for generating approximations of future states. Then we introduce the merging gate to adaptively fuse past and future states. At last, we show the training process and the exit condition during inference.

3.1 Incorporation of Past States

Existing work (Xin et al., 2020) focuses on making exit decision based on a single branch classifier. The consequent unreliable result motivates the recent advance (Zhou et al., 2020) that uses consecutive states to improve the accuracy and robustness. However, the model prediction is still limited to use several local states. In contrast, we investigate how to incorporate all the past states from a global perspective. The existing strategy using consecutive consistent prediction labels can be easily extended to a global version that counts the majority of the predicted labels which is regarded as a voting strategy. Another alternative is the commonly-used ensemble strategy that averages the output probabilities for prediction. Besides these naive solutions, we explore the following strategies to integrate multiple states into a single one:

- Max-Pooling: The max-pooling operation is performed on all available states, resulting in the integrated state.
- Avg-Pooling: The average-pooling operation is performed on all available states, resulting in the integrated state.
- Attn-Pooling: The attentive-pooling takes the weighted summation of all available states as the integrated state. The attention weights are computed with the last state as the query.
- Concatenation: All available states are concatenated and then fed into a linear transformation layer to obtain the compressed state.
- Sequential Neural Network: All available states are sequentially fed into an LSTM and the hidden output of the last time-step is regarded as the integrated state.

Formally, the state of the i-th layer is denoted as $\boldsymbol{s}_i$. When forward propagation proceeds to the i-th intermediate layer, all the past states $\boldsymbol{s}_{1:i}$ are incorporated into a global past state $\boldsymbol{s}_p$:

$$\boldsymbol{s}_p = G(\boldsymbol{s}_{1:i}) \tag{1}$$

where $G(\cdot)$ refers to one of the state incorporation strategies.

3.2 Imitation of Future States

Existing work for early exit stops inference at an intermediate layer and ignores the underlying valuable features captured by the future layers. Such treatment is partly rationalized by the recent claim (Kaya et al., 2019) that shallow layers are adequate to make a correct prediction. However, Jawahar et al. (2019) reveal that the pre-trained language models capture a hierarchy of linguistic information from the lower to the upper layers, e.g., the lower layers learn the surface or syntactic features while the upper layers capture high-level information like the semantic features. We hypothesize that some instances not only rely on syntactic features but also require semantic features. It is actually undesirable to only consider features captured by shallow layers. Therefore, we propose to take advantage of both past and future states.

Normally, we can directly fetch the past states, while using future information is intractable how since the future states are inaccessible before passing through the future layers. To bridge this gap, we propose a simple method to approximate the future states in light of imitation learning (Ross et al., 2011; Nguyen, 2016; Ho and Ermon, 2016). We couple each layer with an imitation learner. During training, the imitation learner is encouraged to mimic the representation of the real state of that layer. Through this layer-wise imitation, we can

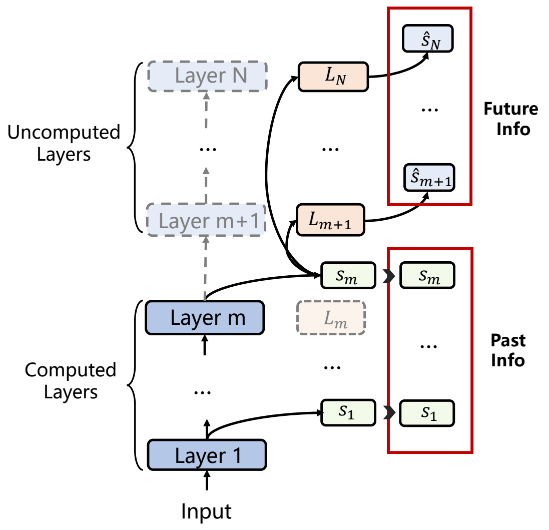


Figure 2: The illustration of the future imitation learning. m is the exit layer and the dashed line denotes the unused modules at inference stage.

obtain approximations of the future states with minimum cost. The illustration of the future imitation learning during inference is shown in Figure 2.

To be precise, we intend to obtain a state approximation of the j-th layer if the forward pass exits at the intermediate i-th layer for any $j > i$. During training, we pass through the entire n-layer model but we simulate the situation that the forward pass ends up at the i-th layer for any $i < n$. The j-th learner corresponding to the j-th layer takes $\boldsymbol{s}_i$ as input and outputs an approximation $\hat{\boldsymbol{s}}^i_j$ of the real state $\boldsymbol{s}_j$. Then $\boldsymbol{s}_j$ serves as a teacher to guide the j-th imitation learner. We adopt cosine similarity as the distance measurement and penalize the discrepancy between the real state $\boldsymbol{s}_j$ and the learned state $\hat{\boldsymbol{s}}^i_j$. Let $\mathcal{L}^i_{\text{cos}}$ denotes the imitation loss of the situation that the forward pass exits at the i-th layer, it is computed as the average of the similarity loss for any $j > i$. Since the exit layer i can be any number between 2 to n during inference, we go through all possible number i and average the corresponding $\mathcal{L}^i_{\text{cos}}$, resulting the overall loss $\mathcal{L}_{\text{cos}}$:

$$\hat{\boldsymbol{s}}^i_j = \text{Learner}^j(\boldsymbol{s}_i) \tag{2}$$

$$l^{i,j}_{\text{cos}}(\boldsymbol{s}_j, \hat{\boldsymbol{s}}^i_j) = 1 - \frac{\hat{\boldsymbol{s}}^i_j \cdot \boldsymbol{s}_j}{\|\hat{\boldsymbol{s}}^i_j\|\|\boldsymbol{s}_j\|} \tag{3}$$

$$\mathcal{L}^i_{\text{cos}} = \frac{1}{n-i}\sum\nolimits_{j=i+1}^{n} l^{i,j}_{\text{cos}}(\boldsymbol{s}_j, \hat{\boldsymbol{s}}^i_j) \tag{4}$$

$$\mathcal{L}_{\text{cos}} = \frac{1}{n-1}\sum\nolimits_{i=2}^{n} \mathcal{L}^i_{\text{cos}} \tag{5}$$

where $\|\cdot\|$ denotes the L_2 norm. Learner$^j(\cdot)$ is a simple feed-forward layer with learnable parameters $\boldsymbol{W}^i$ and $\boldsymbol{b}^i$.

During training, the forward propagation is computed on all layers and all imitation learners are encouraged to generate representations close to the real states. During inference, the forward propagation proceeds to the i-th intermediate layer and the subsequent imitation learners take the i-th real state as input to generate the approximations of future states. Then the approximations are incorporated into a comprehensive future state $\boldsymbol{s}_f$ with one of the global strategies introduced before:

$$\boldsymbol{s}_f = G(\hat{\boldsymbol{s}}^i_{i+1:n}) \tag{6}$$

where $\hat{\boldsymbol{s}}^i_{i+1:n}$ denotes the approximations of the states from the (i+1)-th layer to the n-th layer.

3.3 Adaptive Merging Gate

We then explore how to adaptively merge the past information and future information. Intuitively, the past state $\boldsymbol{s_p}$ and the future state $\boldsymbol{s_f}$ are of different importance since the authentic past states are more reliable than our imitated future states. In addition, different instances depend differently on high-level features learned by future layers. Therefore, it is indispensable to develop an adaptive method to automatically combine the past state $\boldsymbol{s}_p$ and the future state $\boldsymbol{s}_f$. In our work, we design an adaptive merging gate to automatically fuse the past state $\boldsymbol{s}_p$ and the future state $\boldsymbol{s}_f$. As the forward propagation proceeds to the i-th layer, we compute the reliability of the past state $\boldsymbol{s}_p$, and the final merged representation is a trade-off between these two states:

$$\alpha = \text{sigmoid}(\text{FFN}(\boldsymbol{s}_p)) \tag{7}$$

$$\boldsymbol{z}_i = \alpha \boldsymbol{s}_p + (1-\alpha)\boldsymbol{s}_f \tag{8}$$

where $\boldsymbol{z}_i$ is the merged final state and FFN$(\cdot)$ is a linear feed forward layer of the merging gate.

During training, each layer can generate the approximated states of future and obtain a merged final state which is used for prediction. Then the model will be updated with the layer-wise cross-entropy loss against the ground-truth label y. The merging gate adaptively learns to adjust the balance under the supervision signal given by ground-truth labels. However, with the layer-wise optimization objectives, the shallow layers will be updated more frequently since they receive more updating signals from higher layers. To address this issue, we heuristically re-weight the cross entropy loss of each layer depending on its depth i and get its weight w_i. The updating procedure is formalized as:

$$w_i = \frac{i}{\sum_{j=1}^{n} j} \tag{9}$$

$$p_i = \text{softmax}(\boldsymbol{z}_i) \tag{10}$$

$$\mathcal{L}_{\text{ce}}^{i} = -\sum\nolimits_{l \in \text{labels}} y(l) \log(p_i(l)) \tag{11}$$

$$\mathcal{L}_{\text{ce}} = \sum\nolimits_{i=1}^{n} w_i \mathcal{L}_{\text{ce}}^{i} \tag{12}$$

The overall loss is computed as follows:

$$\mathcal{L} = \mathcal{L}_{\text{ce}} + \mathcal{L}_{\text{cos}} \tag{13}$$

3.4 Fine-tuning and Inference

Here we introduce the fine-tuning technique and the exit condition at the inference stage.

Fine-tuning The representations learned by shallow layers have a big impact on performance in the early exit framework since the prediction largely depends on the states of shallow layers. Most existing work updates all of the model layers at each step during fine-tuning to adapt to the data of downstream tasks. However, we argue that such an aggressive updating strategy may undermine the well-generalized features learned in the pre-training stage. In our work, we try to balance the requirements of maintaining features learned in pre-training and adapting to data at the fine-tuning stage. Specifically, the parameters of a layer will be frozen with a probability p and the probability p linearly decreases from the first layer to the L-th layer in a range of 1 to 0.

Inference Following Xin et al. (2020), we quantify the prediction confidence e with the entropy of the output distribution p_i of i-th layer:

$$e(p_i) = \text{Entropy}(p_i) \tag{14}$$

The inference stops once the confidence $e(p_i)$ is lower than a predefined threshold τ. The hyper-parameter τ is adjusted according to the required speed-up ratios. If the exit condition is never reached, our model degrades into the common case of inference that the complete forward propagation is accomplished.

4 Experiments

4.1 Experimental Setup

Experimental Settings Following previous work (Xin et al., 2020), we evaluate our proposed method on six classification datasets from the GLUE benchmark (Wang et al., 2019): SST-2, MRPC, QNLI, RTE, QQP, and MNLI. We perform a grid search over the sets of learning rate as {1e-5, 2e-5, 3e-5, 5e-5}, batch size as {16, 32, 128} and number of frozen layers during fine-tuning as {0,1,2,3}. The maximum sequence length is fixed to 128. We employ a linear decay learning rate scheduler and the AdamW optimizer. In addition, we use the concatenation strategy to incorporate all available states for its best performance on the GLUE dev set.

Speed Measurement Since the measurement of runtime might not be stable, following Xin et al. (2020); Zhou et al. (2020), we manually adjust the exit threshold τ and calculate the speed-up ratio by comparing the actually executed layers in forward propagation and the required complete layers. For a n-layer model, the speed-up ratio is:

$$\text{speed-up ratio} = \frac{\sum_{i=1}^{n} n * m^i}{\sum_{i=1}^{n} i * m^i} \tag{15}$$

where m^i is the number of examples that exit at the i-th layer of the model.

4.2 Baselines

The proposed method can be practical for a range of existing pre-trained language models. Without losing generality, we conduct experiments with several well-known PLMs as backbones, namely, BERT, RoBERTa, and ALBERT (Lan et al., 2019). Both BERT and RoBERTa suffer from the problem of over-parameterization. ALBERT largely alleviates this problem and is very efficient in terms of model size, the results on which verify the effectiveness on such parameter-efficient models. We mainly compare our method with other methods targeting on reducing the depth of models, including the recent early exit methods and the method directly reducing model depth to m layers which is denoted as (AL)BERT-mL.

4.3 Overall Comparison

We compare our model performance with the baseline methods when different backbone models are adopted and show the result in Table 1 and Table 2. Both PABEE (Zhou et al., 2020) and DeeBERT (Xin et al., 2020) accelerate inference with a highest 2× speed-up ratio. To be consistent, we adjust the exit threshold to obtain a 2× speed-up ratio and report the results in Table 1. As shown, our

Model	MNLI-m		MNLI-mm		QQP		QNLI		SST-2		MRPC		RTE		Macro
	Acc	Spd-up	Acc	Spd-up	F1/Acc	Spd-up	Acc	Spd-up	Acc	Spd-up	F1/Acc	Spd-up	Acc	Spd-up	
							BERT								
BERT-base (Devlin et al., 2019)	84.6	1.00×	83.4	1.00×	71.2/ -	1.00×	90.5	1.00×	93.5	1.00×	88.9/ -	1.00×	66.4	1.00×	-
BERT-6L	80.8	2.00×	79.9	2.00×	69.7/88.3	2.00×	86.7	2.00×	91.0	2.00×	85.1/78.6	2.00×	63.9	2.00×	80.5
DeeBERT (Xin et al., 2020)	-		-	-	69.4/ -	1.96×	87.9	1.79×	91.5	1.89×	85.2/ -	1.79×	-	-	-
DeeBERT	74.4	1.87×	73.1	1.88×	70.4/88.8	2.13×	85.6	2.09×	90.2	2.00×	84.4/77.4	2.07×	64.3	1.95×	74.7
PABEE	79.8	2.07×	78.7	2.08×	70.4/88.6	2.09×	88.0	1.87×	89.3	1.95×	84.4/77.4	2.01×	64.0	1.81×	80.0
Ours	**83.3**	1.96×	**82.7**	1.96×	**71.2/89.4**	2.18×	**89.8**	1.97×	**92.8**	2.02×	**87.0/81.8**	1.98×	**64.5**	2.04×	**82.5**
							RoBERTa								
RoBERTa-base (Xin et al., 2020)	87.0	1.00×	86.3	1.00×	71.8/ -	1.00×	92.4	1.00×	94.3	1.00×	90.4/ -	1.00×	67.5	1.00×	-
RoBERTa-6L	84.4	2.00×	83.4	2.00×	71.6/89.2	2.00×	90.4	2.00×	93.5	2.00×	**89.3/85.5**	2.00×	58.0	2.00×	82.5
DeeBERT	64.2	1.87×	64.7	1.87×	**72.0/89.3**	2.05×	83.8	2.01×	86.9	2.02×	88.7/84.3	1.86×	**60.8**	1.90×	75.4
Ours	**86.6**	1.92×	**86.2**	1.93×	**72.0/89.3**	2.54×	**91.7**	2.11×	**94.5**	1.98×	**89.3/85.5**	1.95×	58.0	2.11×	**83.6**
							ALBERT								
ALBERT-base	85.2	1.00×	84.7	1.00×	70.5/88.7	1.00×	92.0	1.00×	93.3	1.00×	89.0/84.8	1.00×	72.0	1.00×	84.8
ALBERT-6L	82.4	2.00×	81.7	2.00×	69.8/88.3	2.00×	90.0	2.00×	91.8	2.00×	87.0/82.4	2.00×	65.8	2.00×	82.2
PABEE	84.2	1.90×	83.5	1.81×	**70.7/88.9**	2.11×	90.9	1.98×	92.4	1.80×	87.6/82.6	1.91×	66.8	2.06×	83.2
Ours	**84.8**	1.94×	**84.1**	1.95×	70.4/88.6	2.35×	**91.9**	1.97×	**92.8**	2.13×	**88.3/84.6**	1.95×	**72.0**	1.93×	**84.5**

Table 1: Model performance on the GLUE test set with different PLMs as backbone. The speed-up ratio (Spd-up) is approximately 2.00 × and our method significantly outperforms previous early exit methods.

Model	MNLI-m		MNLI-mm		QQP		QNLI		SST-2		MRPC		RTE		Macro
	Acc	Spd-up	Acc	Spd-up	F1/Acc	Spd-up	Acc	Spd-up	Acc	Spd-up	F1/Acc	Spd-up	Acc	Spd-up	
							BERT								
BERT-base (Devlin et al., 2019)	84.6	1.00×	83.4	1.00×	71.2/ -	1.00×	90.5	1.00×	93.5	1.00×	88.9/ -	1.00×	66.4	1.00×	-
BERT-4L	77.6	3.00×	77.2	3.00×	67.7/87.5	3.00×	85.4	3.00×	88.7	3.00×	82.9/74.9	3.00×	**63.0**	3.00×	78.4
DeeBERT	61.0	2.80×	59.8	2.84×	66.1/86.9	3.19×	80.8	2.88×	84.7	2.71×	83.5/75.5	2.61×	60.5	2.90×	71.8
PABEE	75.9	2.70×	75.3	2.71×	69.5/88.2	2.57×	82.6	3.04×	85.2	3.15×	82.6/73.1	2.72×	60.5	2.38×	76.6
Ours	**78.4**	2.99×	**77.4**	3.02×	**70.4/89.2**	3.16×	**87.3**	2.78×	**91.1**	2.97×	**84.5/77.7**	2.87×	**63.0**	2.88×	**79.7**
							RoBERTa								
RoBERTa-base (Xin et al., 2020)	87.0	1.00×	86.3	1.00×	71.8/ -	1.00×	92.4	1.00×	94.3	1.00×	90.4/ -	1.00×	67.5	1.00×	-
RoBERTa-4L	80.3	3.00×	79.6	3.00×	69.8/88.4	3.00×	86.0	3.00×	91.3	3.00×	85.0/78.1	3.00×	53.2	3.00×	80.2
DeeBERT	55.1	2.31×	56.6	2.27×	67.1/88.1	3.24×	76.0	2.82×	72.3	2.67×	85.9/79.4	2.87×	-	-	-
Ours	81.4	2.97×	80.5	3.02×	71.9/89.3	3.12×	89.2	2.83×	93.5	2.67×	87.1/82.2	2.75×	54.1	3.01×	80.6
							ALBERT								
ALBERT-base	85.2	1.00×	84.7	1.00×	70.5/88.7	1.00×	92.0	1.00×	93.3	1.00×	89.0/84.8	1.00×	72.0	1.00×	84.8
ALBERT-4L	80.1	3.00×	79.2	3.00×	68.9/88.1	3.00×	87.6	3.00×	89.5	3.00×	84.4/78.9	3.00×	61.2	3.00×	79.7
PABEE	79.6	2.95×	78.9	2.96×	**70.8/88.8**	2.61×	87.9	3.25×	91.9	2.64×	83.6/75.1	2.66×	64.6	2.69×	80.3
Ours	**82.5**	2.93×	**82.0**	2.95×	70.3/88.6	3.17×	**91.0**	2.92×	**92.5**	2.88×	**87.6/82.8**	2.72×	**68.1**	2.92×	**83.0**

Table 2: Model performance on the GLUE test set with different PLMs as backbone. The speed-up ratio (Spd-up) is approximately 3.00× and our method significantly outperforms previous early exit methods.

method maintains a comparable result with the original models on most datasets. We also notice that directly reducing layers performs well and serves as a strong baseline. Nevertheless, our proposal significantly outperforms such a method as well as the other two early exit methods.

We then adopt a more aggressive 3.00× speed-up ratio to verify the effectiveness of our method. According to Table 2, the performance of PABEE and DeeBERT deteriorates badly. In contrast, our model exhibits more robust and stable performance, showing its superiority over previous early exit methods. Particularly, ALBERT is already very efficient in model size owing to its layer-sharing mechanism. Results shown in the bottom of Table 2 suggest that our model can obtain a good result with minimum performance loss on such a parameter-efficient model.

The success of our proposal might be attributed to the global perspective for prediction. DeeBERT makes prediction with the help of the state of a single branch classifier, leading to less reliable results. Although PABEE employs cross-layer prediction to prevent error from one single classifier, they ignore much available information of past states as well as the high-level semantic features captured by future layers. Different from those methods, our method jointly takes into consideration the hierarchical linguistic information embedded in all layers and thus is able to produce more accurate results.

4.4 Performance-Efficiency Trade-Off

To further verify the robustness and efficiency of our method, we visualize the performance-efficiency trade-off curves in Figure 3 on a representative subset of the GLUE dev set. The backbone

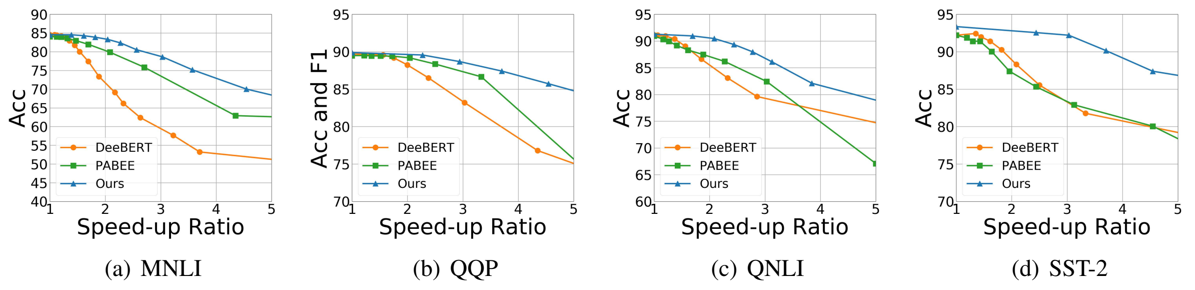


Figure 3: Performance and efficiency trade-off for early exit methods with BERT as backbone. Our method outperforms previous early exit methods by a large margin especially under high speed-up ratios.

Method	**MNLI-m** Acc	**QNLI** Acc	**SST-2** Acc	**MRPC** F1/Acc	**Macro**
Naive global strategies					
Voting	71.33	87.19	89.56	87.71/82.35	83.28
Ensemble	70.92	87.85	90.37	87.44/81.13	83.41
Our global strategies					
Avg-Pooling	81.11	90.43	92.43	88.48/83.09	87.44
Max-Pooling	82.86	90.18	92.32	87.85/82.11	87.59
SequentialNN	82.52	90.17	92.09	**89.35/85.05**	88.00
Attn-Pooling	83.02	90.37	**93.00**	87.83/81.86	87.81
Concatenation	**83.30**	**90.46**	92.89	88.44/83.08	**88.10**

Table 3: The performance of different strategies to incorporate multiple states on the GLUE dev set. The speed-up ratio is approximately 2.00× (±4%).

model is BERT. Please refer to the Appendix A for results of RoBERTa and ALBERT. As can be seen from Figure 3, the performance of previous state-of-the-art early exit methods drops dramatically when the speed-up ratio increases, which limits their practicality for higher acceleration requirements. By comparison, our method demonstrates more tolerance of speed-up ratio. It significantly improves performance compared to previous best-performing early exit models under the same speed-up ratio, especially in the case that the speed-up ratio is high, indicating that it can be applied in a wider range of acceleration scenarios.

4.5 Analysis

4.5.1 Effect of Global Strategies

The results of different global strategies on a representative subset of GLUE dev are shown in Table 3. The naive global strategies including voting and ensemble perform poorly, which demonstrates that existing global strategies can only achieve sub-optimal performance. In contrast, we design simple yet effective global strategies to incorporate past states which bring significant improvement compared to baselines. In addition, we empirically find that the concatenation strategy works best from an overall point of view. We assume that such a strategy allows interaction among different states, yielding better performance. In addition, the effect of the merging gate can be found in Appendix B.

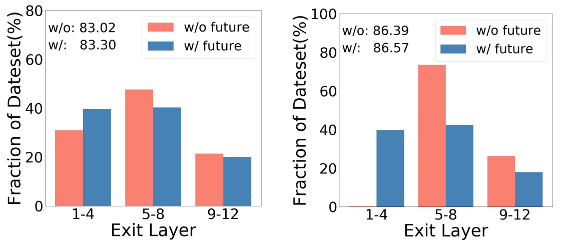
4.5.2 Analysis of Future Information

To assess whether and how future information contributes to the prediction, we first evaluate the Global Future version of our early exit method where all the approximations of futures states are incorporated through the concatenation strategy. Effect of future information is backed with the results shown in Table 4. We observe that the Global Future mechanism brings improvement on most datasets for both 2× speed-up ratio and 3× speed-up ratio, which confirms that the approximations of future states help enhance the model ability in prediction. Beyond that, the future states can be especially advantageous for the models with a higher speed-up ratio. Recall that approximations of future states complement the high-level semantic information and the exit at shallow layers loses more semantic information in comparison with the exit at deep layers. Therefore, the benefit of future information is more significant compared to the exit at shallow layers, which is validated by the larger improvement gap with a 3× speed-up ratio.

We also investigate the effect of future information on exit time. Figure 4 demonstrates the distribution of exit layers with and without future information. When future information is engaged, we observe that the proportion of exit at shallow layers increases. The observation conforms with our intuition: with the approximations of future states supplemented for prediction, the merged state at a shallow layer is able to make a confident and correct prediction. Thus the exit time is earlier compared to situations without future states, result-

Model	MNLI-m		MNLI-mm		QQP		QNLI		SST-2		MRPC		RTE		Macro
	Acc	Spd-up	Acc	Spd-up	F1/Acc	Spd-up	Acc	Spd-up	Acc	Spd-up	F1/Acc	Spd-up	Acc	Spd-up	
≈ 2.00× speed-up															
BERT-local	81.97	1.96×	82.47	1.96×	88.18/91.21	1.85×	89.90	2.00×	92.09	2.00×	86.84/80.39	2.08×	66.43	1.96×	83.73
+Global Future	82.14	2.04×	82.93	1.96×	88.01/91.12	1.89×	90.04	2.04×	92.32	2.08×	87.19/80.64	2.08×	66.78	1.92×	83.96
≈ 3.00× speed-up															
BERT-local	76.72	2.78×	77.64	2.78×	85.80/89.53	3.03×	86.62	2.86×	90.25	2.94×	85.35/77.45	2.94×	62.82	2.86×	80.45
+Global Future	79.06	2.70×	78.86	2.70×	85.65/89.61	3.03×	86.93	2.86×	91.40	2.94×	86.12/78.43	2.86×	62.45	2.86×	81.23

Table 4: Effect of the approximated future states. BERT-local denotes the early exit method using only current state and Global Future represents the incorporation of future states. Results are on the GLUE dev set.



(a) MNLI (b) MRPC

Figure 4: The distribution of exit layers with and without future states on the MNLI and MRPC tasks. The exit threshold for the same task is fixed. When future states are engaged for prediction, we observe an increase of exit at shallow (1-4) layers as well as a performance boost.

Method	MNLI-m	QQP	QNLI	SST-2	Macro
	Acc	F1/Acc	Acc	Acc	
DistilBERT	81.9	70.0/88.4	88.2	92.1	85.4
BERT-PKD (Sun et al., 2019)	81.5	70.7/88.9	89.0	92.0	85.6
PD-BERT (Turc et al., 2019)	82.8	70.4/88.9	88.9	91.8	85.8
BERT-of-Theseus (Xu et al., 2020)	82.4	**71.6/89.3**	89.6	92.2	86.2
TinyBERT‡	81.9	70.0/88.6	88.6	92.0	85.5
Ours	**83.3**	71.2/89.4	**89.8**	**92.8**	**86.6**

Table 5: Comparison with distillation methods on the GLUE test set. TinyBERT‡ is our implementation that removes general distillation and additional fine-tuning resources to match the settings of other methods. The speed-up ratio is approximately 2.00× (±4%).

ing in a higher speed-up ratio. To be more specific, for MRPC, the speed-up ratios with and without future states are 1.69 and 1.99, and are 1.92 and 2.04 for MNLI, respectively. Meanwhile, we observe a performance boost with future states involved. It confirms our assumption that the high-level semantic features embedded in future states help improve performance in early exit framework.

4.5.3 Comparison with Distillation Methods

As an alternative method to accelerate inference, knowledge distillation also exhibits promising performance for NLP tasks. We provide comparison with typical knowledge distillation methods in Table 5. Existing model TinyBERT (Jiao et al., 2019) exerts multiple elaborate strategies to achieve the state-of-the-art results, including the expensive general distillation process and a vast amount of augmented data for fine-tuning. We remove these two techniques to exclude the effect of extra training data. Under the same settings, we observe that our method outperforms the distillation methods with the same speed-up ratio.

In general, early exit and distillation methods improve inference efficiency from different perspectives. The distillation methods are more efficient in saving memory usage, but the downside is that such static methods suffer from high computation cost to adapt to different speed-up ratios. A new student model has to be trained from scratch if the speed-up requirement changes. By contrast, dynamic methods are more flexible to meet different acceleration requirements. Concretely, simple instances will be processed by passing through fewer layers and complex instances may require more layers. Moreover, the speed-up ratio can be easily adjusted depending on the acceleration requests. Nevertheless, early exit and distillation accelerate inference from different perspectives and these two kinds of techniques can be integrated to further compress the model size and accelerate the inference time.

5 Conclusions

We propose a novel Past-Future early exit method from a global perspective. Unlike previous work using only local states for prediction, our model employs all available past states for prediction and propose a novel approach to engage the future states which are originally inaccessible for prediction. Experiments illustrate that our method achieves significant improvement over baseline methods with different models as backbones, suggesting the superiority of our early exit method.

Acknowledgements

We thank all the anonymous reviewers for their constructive comments. This work is partly supported by National Key R&D Program of China No. 2019YFC1521200 and Beijing Academy of Artificial Intelligence (BAAI). Xu Sun is the corresponding author.

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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. 2020. Depth-adaptive transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4565–4573.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3651–3657. Association for Computational Linguistics.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling BERT for natural language understanding. *CoRR*, abs/1909.10351.

Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. 2019. Shallow-deep networks: Understanding and mitigating network overthinking. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, 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. *CoRR*, abs/1909.11942.

Jianquan Li, Xiaokang Liu, Honghong Zhao, Ruifeng Xu, Min Yang, and Yaohong Jin. 2020a. BERT-EMD: many-to-many layer mapping for BERT compression with earth mover's distance. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 3009–3018. Association for Computational Linguistics.

Lei Li, Yankai Lin, Shuhuai Ren, Deli Chen, Xuancheng Ren, Peng Li, Jie Zhou, and Xu Sun. 2020b. Accelerating pre-trained language models via calibrated cascade. *CoRR*, abs/2012.14682.

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. *CoRR*, abs/1907.11692.

Khanh Nguyen. 2016. Imitation learning with recurrent neural networks. *CoRR*, abs/1607.05241.

Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, volume 15 of *JMLR Proceedings*, pages 627–635. JMLR.org.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

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, Online, July 5-10, 2020*, pages 6640–6651. Association for Computational Linguistics.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for BERT model compression. 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, Hong Kong, China, November 3-7, 2019*, pages 4322–4331. Association for Computational Linguistics.

Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. 2016. Branchynet: Fast inference via early exiting from deep neural networks. In *23rd International Conference on Pattern Recognition, ICPR 2016, Cancún, Mexico, December 4-8, 2016*, pages 2464–2469. IEEE.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-Read Students Learn Better:

On the Importance of Pre-training Compact Models. *arXiv e-prints*, page arXiv:1908.08962.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

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 *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *CoRR*, abs/2002.10957.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. Deebert: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 2246–2251. Association for Computational Linguistics.

Canwen Xu, Wangchunshu Zhou, Tao Ge, Furu Wei, and Ming Zhou. 2020. Bert-of-theseus: Compressing BERT by progressive module replacing. *CoRR*, abs/2002.02925.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 5754–5764.

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

A More Performance-Efficiency Trade-Off Curves

Performance-efficiency curves with RoBERTa and ALBERT as backbones are shown in Figure 5 and Figure 6 respectively. Similar to the observation with BERT as backbone, the performance of DeeBERT and PABEE becomes progressively worse as the speed-up ratio increases. In contrast, our past-future early exit method shows more robust results.

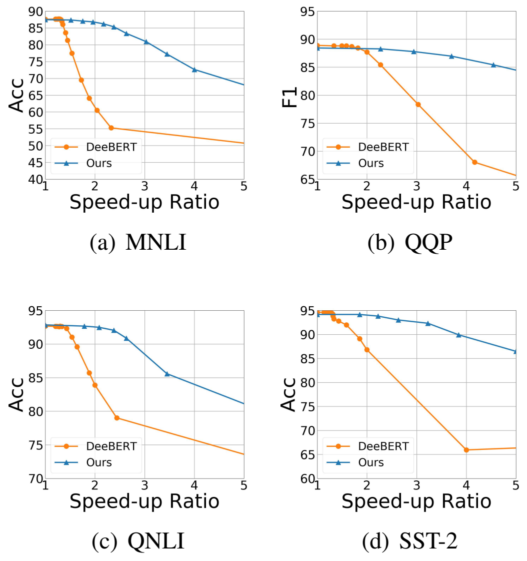


Figure 5: Performance-efficiency trade-off for early exit method DeeBERT with RoBERTa as backbone.

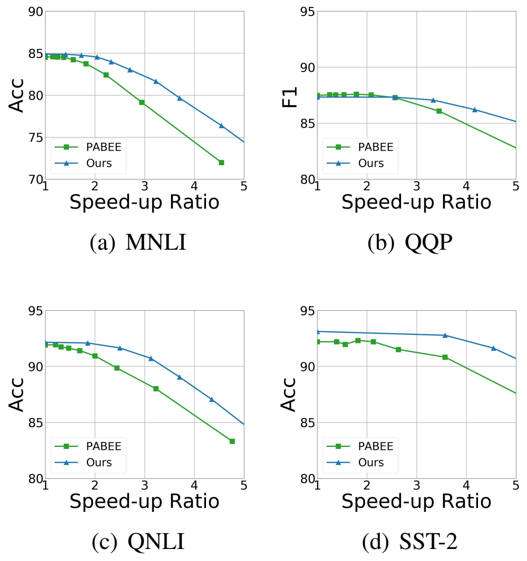


Figure 6: Performance-efficiency trade-off for early exit method PABEE with ALBERT as backbone.

B Effect of Merging Gate

Method	**MNLI-m** Acc	**QNLI** Acc	**SST-2** Acc	**MRPC** F1/Acc	**Macro**
Ours	83.30	90.46	92.89	88.44/83.08	88.10
-merging gate	83.15	90.61	92.43	86.86/80.64	87.49

Table 6: Ablation study of the merging gate. The speed-up ration is approximately 2.00× and the model implementation is based on BERT.

We conduct ablation study to show the effect of the merging gate and report the result in Table 6. We can see that the performance drops when we remove the merging gate from our model, suggesting that the merging gate plays an important role in keeping the balance between past information and future information.