F-PABEE: FLEXIBLE-PATIENCE-BASED EARLY EXITING FOR SINGLE-LABEL AND MULTI-LABEL TEXT CLASSIFICATION TASKS

Xiangxiang Gao[1], *Wei Zhu*[2*]

[1] Shanghai Jiaotong University, China
[2] East China Normal University, China

Jiasheng Gao[3], *Congrui Yin*[4]

[3] Shenzhen University, China
[4] Nanchang University, China

ABSTRACT

Computational complexity and overthinking problems have become the bottlenecks for pre-training language models (PLMs) with millions or even trillions of parameters. A Flexible-Patience-Based Early Exiting method (F-PABEE) has been proposed to alleviate the problems mentioned above for single-label classification (SLC) and multi-label classification (MLC) tasks. F-PABEE makes predictions at the classifier and will exit early if predicted distributions of cross-layer are consecutively similar. It is more flexible than the previous state-of-the-art (SOTA) early exiting method PABEE because it can simultaneously adjust the similarity score thresholds and the patience parameters. Extensive experiments show that: (1) F-PABEE makes a better speedup-accuracy balance than existing early exiting strategies on both SLC and MLC tasks. (2) F-PABEE achieves faster inference and better performances on different PLMs such as BERT and ALBERT. (3) F-PABEE-JSKD performs best for F-PABEE with different similarity measures.

***Index Terms*—** F-PABEE, PABEE, Early Exiting, Multi-label Classification, Single-label Classification

1. INTRODUCTION

Fine-tuning PLMs has become the de-facto paradigm in natural language processing [1]. Despite SOTA performances, BERT [2] and its variants [3, 4, 5] still face significant application challenges: cumbersome computation and overthinking problems due to huge parameters and deep models. Early exiting attracts much attention as an input-adaptive method to speed up inference [6]. Early exiting installs a classifier at each transformer layer to evaluate the predictions and will exit when meeting the criterion. Three different early exiting strategies exist: (1) The confidence-based strategy evaluates the predictions based on specific confidence measurements. (2) The learned-based strategy learns a criterion for early exiting. (3) The patience-based strategy exits when consecutive classifiers make the exact predictions. Among them, the patience-based strategy PABEE [7] achieves SOTA results.

We raise two issues for the current SOTA strategy: (1) PABEE faces a limitation for application: it can not flexibly adjust the speedup ratio on a given task and fixed patience parameter, mainly caused by a strict cross-layer comparison strategy. Thus, we wonder whether we can combine PABEE with a softer cross-layer comparison strategy. (2) Current early exiting strategies mainly focus on SLC tasks, while the MLC tasks are neglected. So can they speed up MLC tasks?

Therefore, we propose a Flexible-Patience-Based Early Exiting method (F-PABEE) to address the above issues. F-PABEE makes predictions at each classifier and will exit early if the current layer and the last few layers have similar (similarity score less than a threshold) predicted distributions. F-PABEE can be seen as a natural extension of PABEE and is more flexible since it can achieve better speed-accuracy trade-offs by adjusting the similarity score thresholds and patience parameters. It can also extend to MLC tasks effortlessly.

Our contributions are summarized as follows: (1) We propose F-PABEE, a novel and effective inference mechanism that is flexible in adjusting the speedup ratios of PLMs. (2) The results show that our method can accelerate inference effectively while maintaining good performances across different SLC and MLC tasks. (3) We are the first to investigate the early exiting of MLC tasks, and F-PABEE is suitable for this type of task.

2. RELATED WORKS

2.1. Static inference approach

The static inference approach compresses the heavy model into a smaller one, including pruning, knowledge distillation, quantization, and weight sharing [8, 9, 10]. For example, HeadPrune [11] ranks the attention heads and prunes them to reduce inference latency. PKD [12] investigates the best practices of distilling knowledge from BERT into smaller-sized models. I-BERT [13] performs an end-to-end BERT inference without any floating point calculation. ALBERT [3] shares the cross-layer parameters. Note that the static models are still in the form of deep neural networks with multiple

*Wei Zhu contributes equally with Xiangxiang Gao, and he is the corresponding author. Email: wzhu@stu.ecnu.edu.cn.

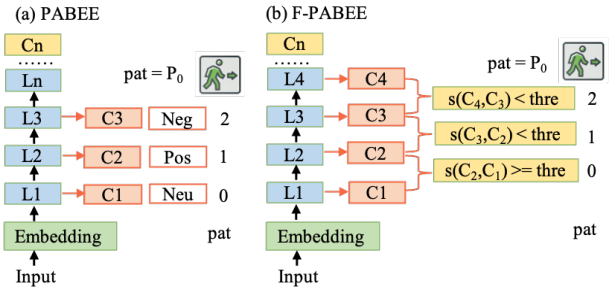

Fig. 1. Inference procedure of PABEE and F-PABEE, C_i is the classifier, $thre$ is threshold, P_0 is pre-defined patience.

stacked layers. The computational path is invariable for all examples in the inference process, which is not flexible.

2.2. Dynamic early exiting

Orthogonal to the static inference approach, early exiting dynamically adjusts hyper-parameters in response to changes in request traffic. It does not need to make significant changes to the original model structure or weight bits, nor does it need to train different teacher-student learning networks, which saves computing resources [14]. There are mainly three groups of dynamic early exiting strategies. The first type is confidence-based early exiting. For example, BranchyNet [15], FastBERT [16], and DeeBERT [17] calculate the entropy of the prediction probability distribution to estimate the confidence of classifiers to enable dynamic early exiting. Shallow-deep [18] and RightTool [19] leverage the maximum of the predicted distribution as the exiting signal. The second type is the learned-based exiting, such as BERxiT [20] and CAT [21]. They learn a criterion for early exiting. The third type is patience-based early exiting, such as PABEE [7], which stops inference and exits early if the classifiers' predictions remain unchanged for pre-defined times. Among them, patience-based PABEE achieves SOTA performance. However, PABEE suffers from too strict cross-layer comparison, and the applications on MLC tasks are neglected.

F-PABEE is a more flexible extension to PABEE, which can simultaneously adjust the confidence thresholds and patience parameters to meet different requirements. In addition, it outperforms other existing early exiting strategies on both SLC and MLC tasks.

3. FLEXIBLE PATIENCE-BASED EARLY EXITING

3.1. Inference procedure for SLC and MLC tasks

The inference procedure of F-PABEE is shown in Fig 1(b), which is an improved version of PABEE (Fig 1(a)), where L_i is the transformer block of the model, n is the number of transformer layers, C_i is the inserted classifier layer, s is the cross-layer similarity score, $thre$ is the similarity score threshold, P_0 is the pre-defined patience value in the model.

The input sentences are first embedded as the vector:

$$h_0 = Embedding(x). \tag{1}$$

The vector is then passed through transformer layers ($L_1 ... L_n$) to extract features and compute its hidden state h. After which, we use internal classifiers ($C_1 ... C_n$), which are connected to each transformer layer to predict probability p:

$$p_i = C_i(h_i) = C_i(L_i(h_{i-1})). \tag{2}$$

We denote the similarity score between the prediction results of layer $i-1$ and i as $s(p_{i-1}, p_i)$ ($s(p_{i-1}, p_i) \in \mathbf{R}$). The smaller the value of $s(p_{i-1}, p_i)$, the prediction distributions are more consistent with each other. The premise of the model's early exit is that the comparison scores between successive layers are relatively small; The similarity threshold $thre$ is a hyper-parameter. We use pat_i to store the times that the cross-layer comparison scores are consecutively less than the threshold $thre$ when the model reaches current layer i:

$$pat_i = \begin{cases} pat_{i-1} + 1 & s(p_{i-1}, p_i) < thre \\ 0 & s(p_{i-1}, p_i) >= thre \end{cases} \tag{3}$$

If $s(p_{i-1}, p_i)$ is less than the similarity score threshold $thre$, then increase the patience counter by 1. Otherwise, reset the patience counter to 0. This process is repeated until pat reaches the pre-defined patience value P_0. The model dynamically stops inference and exits early. However, if this condition is never met, the model uses the final classifier layer to make predictions. This way, the model can stop inference early without going through all layers.

3.2. Similarity measures for SLC and MLC tasks

Under the framework of F-PABEE, we can adopt different similarity measures for predicted probability distributions. This work uses the knowledge distillation objectives as the similarity measures [22]. When the model reaches the current layer l, for SLC tasks, we compare a series of similarity measures of F-PABEE, denoted as:

F-PABEE-KD: It adopts the knowledge distillation objective from probability mass distribution p^{l-1} to p^l:

$$s(p^{l-1}, p^l) = -\sum_{j=1}^{k} p_j^{l-1} log(p_j^l); \tag{4}$$

F-PABEE-ReKD: It adopts the knowledge distillation objective in the reverse direction, from probability mass distribution p^l to p^{l-1}:

$$s(p^l, p^{l-1}) = -\sum_{j=1}^{k} p_j^l log(p_j^{l-1}); \tag{5}$$

	CoLA		MNLI		MRPC		QNLI		QQP		RTE		SST-2	
	score	speedup	score	speedup	score	speedup	score	speedup	score	speedup	score	speedup	score	speedup
BERT base	54.2	0%	83.1	0%	86.8	0%	89.8	0%	89.2	0%	69.1	0%	91.3	0%
Fixed-Exit-3L	0.0	75%	70.0	75%	75.8	75%	77.4	75%	81.8	75%	54.7	75%	81.0	75%
Fixed-Exit-6L	0.0	50%	79.6	50%	84.7	50%	85.3	50%	89.3	50%	68.1	50%	88.6	50%
BranchyNet	0.0	74%	63.8	76%	75.7	76%	74.2	80%	71.6	80%	54.7	76%	79.9	76%
	0.0	51%	78.3	53%	83.0	52%	87.1	47%	89.3	50%	67.4	47%	88.3	49%
Shallow-Deep	0.0	75%	64.1	77%	75.6	76%	74.3	78%	71.4	79%	54.7	76%	79.5	77%
	0.0	52%	78.2	51%	82.8	51%	87.2	49%	89.6	51%	67.2	48%	88.4	48%
BERxiT	0.0	76%	63.5	76%	75.6	76%	73.3	78%	68.2	80%	55.3	77%	79.5	76%
	12.3	52%	78.4	51%	82.9	51%	87.0	48%	89.1	49%	67.3	47%	88.3	49%
PABEE	0.0	75%	63.9	77%	75.8	75%	73.6	81%	68.6	82%	55.8	75%	79.9	77%
	0.0	50%	78.9	52%	83.1	53%	87.2	46%	89.6	49%	67.7	46%	88.7	48%
F-PABEE	0.0	75%	**66.9**	72%	**81.5**	77%	**76.2**	75%	**79.6**	82%	**56.0**	76%	**80.5**	76%
	13.6	52%	**83.9**	53%	**87.3**	53%	**88.6**	54%	**90.8**	49%	**68.1**	47%	**92.3**	48%

Table 1. Experimental results of different early exiting methods with BERT backbone on the GLUE benchmark.

F-PABEE-SymKD: It adopts a symmetrical knowledge distillation objective:

$$SymKD = s(p^{l-1}, p^l) + s(p^l, p^{l-1}); \quad (6)$$

F-PABEE-JSKD: It adopts another symmetrical distillation objective, similar to Jenson-Shannon divergence:

$$JSKD = \frac{1}{2}s(p^{l-1}, \frac{p^{l-1}+p^l}{2}) + \frac{1}{2}s(p^l, \frac{p^{l-1}+p^l}{2}) \quad (7)$$

In addition, for MLC tasks, we transform them into multiple binary classification problems and sum the similarity scores of all categories, and the formulas are denoted as:
F-PABEE-KD:

$$s(p^{l-1}, p^l) = -\sum_{j=1}^{k}\sum_{i=1}^{2} p_{ji}^{l-1} log(p_{ji}^l); \quad (8)$$

F-PABEE-ReKD:

$$s(p^l, p^{l-1}) = -\sum_{j=1}^{k}\sum_{i=1}^{2} p_{ji}^{l} log(p_{ji}^{l-1}); \quad (9)$$

The formulations of F-PABEE-SymKD and F-PABEE-JSKD for MLC tasks are similar to those of SLC tasks.

3.3. Training procedure

F-PABEE is trained on SLC and MLC tasks, while the activation and loss functions are different. For SLC tasks, we use the softmax activation function and cross-entropy function according to the tasks. In contrast, we use the sigmoid activation function and binary cross-entropy function for MLC tasks.

After that, we optimize the model parameters by minimizing the overall loss function L, which is the weighted average of the loss terms from all classifiers:

$$L = \sum_{j=1}^{n} jL_j / \sum_{j=1}^{n} j \quad (10)$$

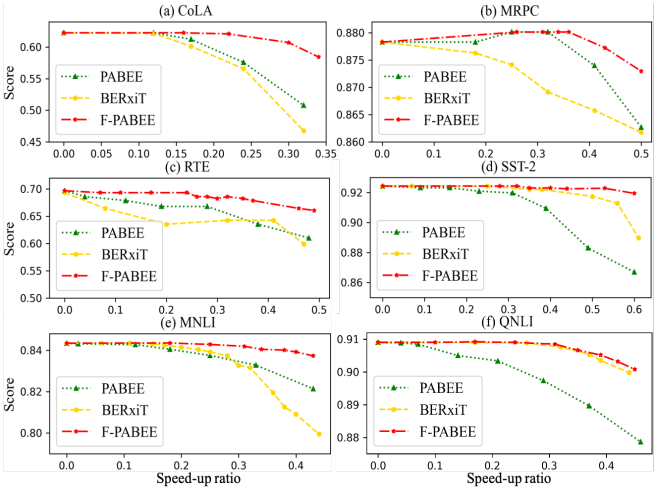

Fig. 2. Speed-accuracy curves of F-PABEE, PABEE and BERxiT on SLC tasks with BERT backbone.

4. EXPERIMENTS

4.1. Tasks and Baselines

We evaluate F-PABEE on GLUE benchmark [23] for SLC tasks and four datasets for MLC tasks: MixSNLPS [24], MixATS [25], AAPD [26], and Stackoverflow [27]. we compare F-PABEE with three groups of baselines: (1) BERT-base; (2) Static exiting; (3) Dynamic exiting methods, including BrachcyNet [28], Shallow-Deep [18], BERxiT [20], and PABEE. Considering the flops of inferencing one with the whole BERT as the base, the speed-up ratio is defined as the average ratio of reduced flops due to early exiting.

4.2. Experimental setting

In training process, we perform grid search over the batch size of {16, 32, 128}, and learning rate of {1e-5, 2e-5, 3e-5, 5e-5} with an AdamW optimizer [29] . The batch size in the inference process is 1. We implement F-PABEE on the bases of HuggingFace Transformers [30]. All experiments are conducted on two Nvidia TITAN X 24GB GPUs.

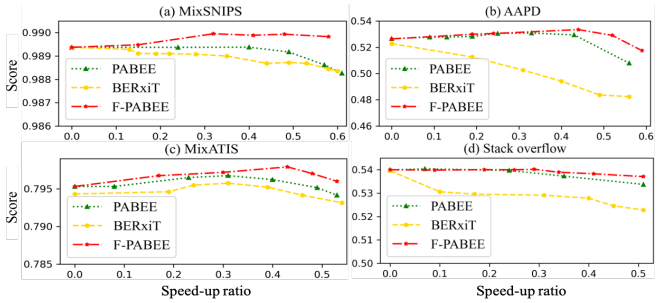

Fig. 3. Speed-accuracy curves of F-PABEE, PABEE and BERxiT on MLC tasks with BERT backbone.

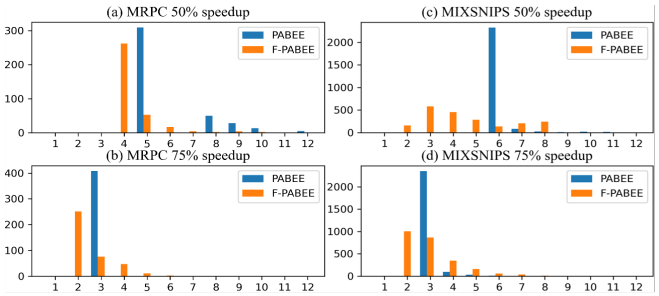

Fig. 4. The distribution of executed layers of MRPC and MixSNIPS on average at different speeds (50%, 75%).

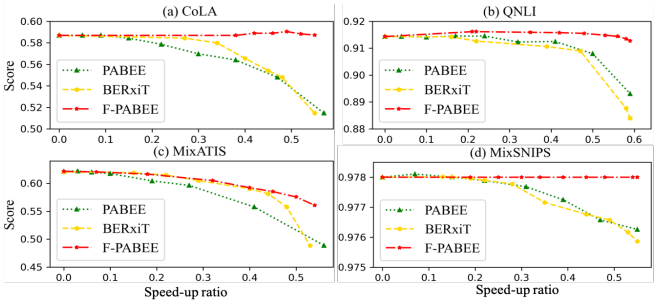

Fig. 5. Speed-accuracy curves of F-PABEE, PABEE and BERxiT on SLC and MLC tasks with ALBERT backbone.

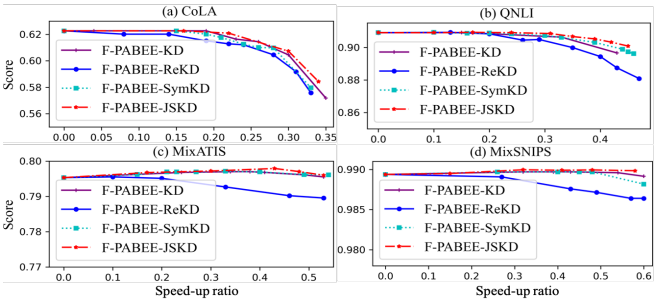

Fig. 6. Speed-accuracy curves of different similarity measures on SLC and MLC tasks with BERT backbone.

4.3. Overall comparisons

In Table 1, we compare F-PABEE with other early exiting strategies. We adjust the hyper-parameters of F-PABEE and other baselines to ensure similar speedups with PABEE. It shows that F-PABEE balances speedup and performance better than baselines, especially for a large speedup ratio. Moreover, we draw the score-speedup curves for BERxiT, PABEE, and F-PABEE. It shows that F-PABEE outperforms the baseline models on both SLC (Fig 2) and MLC tasks(Fig 3). Furthermore, the distribution of executed layers (Fig 4) indicates that F-PABEE can choose the faster off-ramp and achieve a better trade-off between accuracy and efficiency by flexibly adjusting similarity score thresholds and patience parameters.

4.4. Ablation studies

Ablation on different PLMs F-PABEE is flexible and can work well with other pre-trained models, such as ALBERT. Therefore, to show the acceleration ability of F-PABEE with different backbones, we compare F-PABEE to other early exiting strategies with ALBERT base as the backbone. The results in Fig 5 show that F-PABEE outperforms other early exiting strategies under different backbones by large margins on both SLC and MLC tasks, indicating that F-PABEE can accelerate the inference process for numerous PLMs.

Comparisons between different similarity measures We consider F-PABEE with different similarity measures, denoted as F-PABEE-KD, F-PABEE-ReKD, F-PABEE-SymKD, and F-PABEE-JSKD, and the results are presented in Fig 6. F-PABEE-JSKD performs best on both SLC and MLC tasks. We suppose that F-PABEE-JSKD is symmetric, and the similarity discrimination is more accurate than asymmetric measures. Therefore, it is good to determine which samples should exit at shallow layers and which should go through deep layers.

5. CONCLUSIONS

We proposed F-PABEE, a novel and efficient early exiting method that combines PABEE with a softer cross-layer comparison strategy. F-PABEE is more flexible than PABEE since it can achieve different speed-performance tradeoffs by adjusting the similarity score thresholds and patience parameters. In addition, we investigate the acceleration ability of F-PABEE with different backbones. Moreover, we compare the performances of F-PABEE with different similarity measures. Extensive experiments on SLC and MLC demonstrate that: (1) F-PABEE performs better than the previous SOTA adaptive early exiting strategies for both SLC and MLC tasks. As far as we know, we are the first to investigate the early exiting methods for MLC tasks. (2) F-PABEE performs well on different PLMs such as BERT and ALBERT. (3) Ablation studies show that F-PABEE-JSKD performs best for F-PABEE with different similarity measures.

6. REFERENCES

[1] Tianyang Lin et al., "A survey of transformers," *arXiv*, vol. abs/2106.04554, 2021.

[2] Jacob Devlin et al., "BERT: pre-training of deep bidirectional transformers for language understanding," *ArXiv*, vol. abs/1810.04805, 2018.

[3] Zhenzhong Lan et al., "ALBERT: A lite BERT for self-supervised learning of language representations," *arXiv*, vol. abs/1909.11942, 2019.

[4] Zhilin Yang et al., "Xlnet: Generalized autoregressive pretraining for language understanding," *arXiv*, vol. abs/1906.08237, 2019.

[5] Yinhan Liu et al., "Roberta: A robustly optimized BERT pretraining approach," *arXiv*, vol. abs/1907.11692, 2019.

[6] Canwen et al. Xu, "A survey on dynamic neural networks for natural language processing," *arXiv*, vol. abs/2202.07101, 2022.

[7] Wangchunshu Zhou et al., "BERT loses patience: Fast and robust inference with early exit," *arXiv*, vol. abs/2006.04152, 2020.

[8] Canwen et al., "Bert-of-theseus: Compressing bert by progressive module replacing," *EMNLP*, 2020.

[9] Victor Sanh et al., "Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter," *arXiv*, vol. abs/1910.01108, 2019.

[10] Angela Fan et al., "Reducing transformer depth on demand with structured dropout," *arXiv*, vol. abs/1909.11556, 2019.

[11] Paul Michel et al., "Are sixteen heads really better than one?," *arXiv*, vol. abs/1905.10650, 2019.

[12] Siqi Sun et al., "Patient knowledge distillation for BERT model compression," *arXiv*, vol. abs/1908.09355, 2019.

[13] Sehoon Kim et al., "I-BERT: integer-only BERT quantization," *arXiv*, vol. abs/2101.01321, 2021.

[14] Mostafa Dehghani et al., "Universal transformers," *arXiv*, vol. abs/1807.03819, 2018.

[15] Teerapittayanon et al., "Branchynet: Fast inference via early exiting from deep neural networks," in *2016 23rd International Conference on Pattern Recognition (ICPR)*, 2016, pp. 2464–2469.

[16] Weijie Liu et al., "Fastbert: a self-distilling BERT with adaptive inference time," *arXiv*, vol. abs/2004.02178, 2020.

[17] Ji Xin et al., "Deebert: Dynamic early exiting for accelerating BERT inference," *arXiv*, vol. abs/2004.12993, 2020.

[18] Yigitcan Kaya et al., "How to stop off-the-shelf deep neural networks from overthinking," *arXiv*, vol. abs/1810.07052, 2018.

[19] Roy Schwartz et al., "The right tool for the job: Matching model and instance complexities," *arXiv*, vol. abs/2004.07453, 2020.

[20] Ji et al. Xin, "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*, Online, Apr. 2021, pp. 91–104, Association for Computational Linguistics.

[21] Tal Schuster et al., "Consistent accelerated inference via confident adaptive transformers," *CoRR*, vol. abs/2104.08803, 2021.

[22] Geoffrey E. Hinton et al., "Distilling the knowledge in a neural network," *ArXiv*, vol. abs/1503.02531, 2015.

[23] Alex Wang et al., "GLUE: A multi-task benchmark and analysis platform for natural language understanding," *CoRR*, vol. abs/1804.07461, 2018.

[24] Alice Coucke et al., "Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces," *arXiv*, vol. abs/1805.10190, 2018.

[25] Hemphill et al., "The ATIS spoken language systems pilot corpus," in *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*, 1990.

[26] Pengcheng Yang et al., "SGM: sequence generation model for multi-label classification," *arXiv*, vol. abs/1806.04822, 2018.

[27] Jeff Atwood, "Stack overflow creative commons data dump," https://archive.org/details/stackexchange, 2009.

[28] Haoli Bai et al., "Binarybert: Pushing the limit of bert quantization," vol. abs/2012.15701, 2020.

[29] Ilya Loshchilov et al., "Decoupled weight decay regularization," in *ICLR*, 2019.

[30] Wolf et al., "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, 2020, pp. 38–45.