2023 26th International Conference on Computer Supported Cooperative Work in Design (CSCWD) | 979-8-3503-3168-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/CSCWD57460.2023.10152853

MetaBERT: Collaborative Meta-Learning for Accelerating BERT Inference

Yangyan Xu[1,2], Fangfang Yuan[1*], Cong Cao[1], Xiaoliang Zhang[1,2], Majing Su[3], Dakui Wang[1], Yanbing Liu[1,2]

[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3]The 6th Research Institute of China Electronic Corporations, Beijing, China
{xuyangyan,yuanfangfang,caocong,zhangxiaoliang,wangdakui,liuyanbing}@iie.ac.cn, sumj@ncse.com.cn

***Abstract*—Early exit methods are used to accelerate inference in pre-trained language models and maintain competitive performance on resource-constrained devices. However, existing methods for training early exit classifiers suffer from the problem of poor classifier representations in different layers, leading to difficulties in adapting to diverse natural language processing tasks. To address this issue, we propose MetaBERT: collaborative Meta-learning for accelerating BERT inference. The main goal of MetaBERT is to train early exit classifiers through collaborative meta-learning, in which case, few gradient updates can be quickly adapted to new tasks. Moreover, this novel meta-training approach produces good generalization performance, thus achieving an effective balance between the inference result and efficiency. Extensive experimental results show that our approach outperforms previous training methods by a large margin, and achieves state-of-the-art results compared to other competitive models.**

***Index Terms*—inference acceleration, early exit, meta-learning, BERT**

I. Introduction

Pre-trained language models (PLMs) based on transformer architecture [1], such as BERT [2], have become the dominant models in the natural language processing (NLP) community. Although all kinds of NLP tasks benefit from the transformer family of models [1]–[3], their inference speed is extremely slow. Slow inference speed can hinder easy deployment in real-world business scenarios, which becomes even more challenging when inference must be performed on edge devices.

In response to this limitation, studies focusing on improving the inference efficiency of PLMs have increasingly emerged to address the issue of slow inference. Existing model compression methods, such as knowledge distillation [4], pruning [5], quantization [6], module replacing [7], layer drop [8], etc., have been used to accelerate pre-trained models inference. However, these methods permanently remove some components of the model, resulting in a dramatic drop in performance. Furthermore, the structure of these models cannot be dynamically adjusted to meet the needs of different hardware because they are already compressed before the deployment. In light of this observation, conditional computation [9], such as early exit methods [10]–[15], are proposed for dynamic inference. Early exit models can dynamically decide at which

[*]Corresponding author.

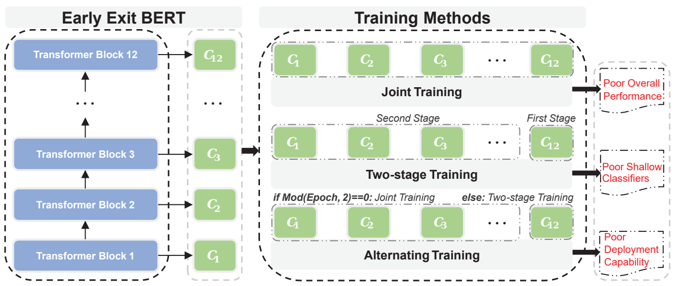

Fig. 1. The current mainstream training methods for early exit models. The joint training method sums the classification loss weights of exiting classifiers c_1 to c_{12}. For the two-stage training method, training c_{12} in the first stage, and training c_1 to c_{11} in the second stage. The alternating training method uses joint training in even epochs and two-stage training in odd epochs.

layer to output, depending on the difficulty of the samples, instead of only at the end of the transformer layer [2]. During fine-tuning, early exit classifiers added to each BERT transformer block are trained. During inference, the early exit classifier of a certain intermediate layer dynamically makes a prediction.

Early exit strategies have two major limitations that prevent their advancement: (1) Existing methods are dedicated to exploring inference acceleration for simple samples. However, for hard samples, it is still necessary to resort to the last layer. (2) The computational consumption increases in the same proportion as the length of the input sequence. Although existing early exit methods can dynamically exit, these dynamic models of classifiers are still trained in ways that do not adequately learn the semantic features of the samples. In the early exit approach of BERT, as shown in Fig. 1, there are currently three training methods: joint training [10], [11], [13], [16], [17], two-stage training [12], [14], and alternating training [18]. Among these three approaches, joint training tends to cause poor overall classifier performance (the performance of weak classifiers will pull down the performance of strong classifiers). Two-stage training can keep the performance of the classifier in the last layer better, but the performance of shallow classifiers will be poor. Alternating training has the disadvantages of both of these training methods, while not being efficient enough for practical deployment.

Algorithm 1: Our meta-learning approach

```
Input  : $p(\mathcal{T})$: distribution over meta-training tasks;
         $\alpha$, $\beta$: meta-learning hyper-parameters
Output: the model parameters $\theta$
1 Randomly initialize $\theta$;
2 while not done do
3     Sample batch of meta-training tasks $\mathcal{T}_i \sim p(\mathcal{T})$;
4     for each task in $\mathcal{T}_i$ do
5         Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K
            examples;
6         Compute adapted parameters $\theta_i'$ by Eq. (2);
7     end
8     Update $\theta$ by Eq. (4);
9 end
```

Algorithm 2: The training process of MetaBERT

```
Input  : $c_i$: the early exit classifier;
         $L$: total number of transformer layers
Output: $\mathcal{L}_{Total}$
1 repeat
2     A set of optimized parameters $\theta$ is obtained from
        meta-learning in Algorithm 1;
3     We pass $\theta$ into the Adam optimizer for gradient
        update;
4     Compute the cross-entropy loss $\mathcal{L}_i^{CE}$ for each exit
        classifier $c_i$;
5     Update total loss:
          $$\mathcal{L}_{Total} = \sum_{i=1}^{L} \mathcal{L}_i^{CE}$$
6 until Epoch stopping criterion is satisfied;
```

To tackle the above problems, we propose **MetaBERT**: collaborative **Meta**-learning for accelerating **BERT** inference. Collaborative meta-learning plays a role in two aspects: (1) Unlike earlier work, which used one or several shallow layers for prediction, our strategy successfully trains all available past layers. (2) Additionally, in order to understand semantic information contained in deep layers, previously inaccessible features of new tasks are approximated via collaborative meta-learning and used for prediction. Our strategy can produce more accurate predictions for various downstream tasks by integrating shallow and deep levels of information. In summary, all early exit classifiers are trained through meta-learning, so that instead of learning results, classifiers learn how to extract semantic features from samples. Meanwhile, with this idea of learning to learn, MetaBERT gets a better set of initialization parameters. This set of parameters learns representations that can be quickly adapted to new tasks with few gradient iterations, which provides the prerequisite for efficient BERT inference.

Extensive experiments reveal that the proposed MetaBERT significantly outperforms the state-of-the-art early exit methods. Particularly, it surpasses the previous static models by a large margin when the acceleration ratio is relatively high. Furthermore, experiments with different training techniques show significant improvement over baseline approaches, confirming the applicability of our strategy.

To summarize, our contributions are as follows:

- In contrast to current training methods, we provide a collaborative meta-learning training strategy that successfully incorporates all early exit classifiers and achieves improved performance.
- Our approach takes advantage of new task features that were previously inaccessible at the inference stage to produce more comprehensive predictions.
- Extensive experimental results demonstrate that our proposed approach performs better than the most recent state-of-the-art early exit strategies.

II. Related Work

There are extensive studies on compressing and accelerating the inference of neural networks, such as knowledge distillation [19], pruning [20], quantization [21], module replacing [7], layer drop [8], and early exit [18], [22]. A comprehensive survey is provided in [23]. Here, we summarize the approaches that are most relevant to our research.

A. Early Exit

Conditional computation [9], [24], which selectively activates only parts of the model conditioned on a given input rather than pursuing a more compact static model. [9] introduces an end-to-end halting mechanism, Adaptive Computation Time, to carry out the input-adaptive computation, which is later employed in Universal Transformer [25].

Early exit approaches can be seen as an instance of conditional computation. In particular, early exit models allow distinct samples to exit early in certain layers based on the features of the input samples. They first appear in computer vision, such as Shallow-Deep Network [11] and BranchyNet [26]. Recently, with the advent of PLMs, early exit mechanisms are also used to accelerate inference of transformer-based models, such as GAML-BERT [13], Depth-Adaptive Transformer [27], FastBERT [14], DeeBERT [12], PABEE [10], and Right Tool [28]. However, studies on the training methods of early exit classifiers have largely focused on shallow surface features near the exit layer.

B. Meta-Learning

The basic concept of meta-learning is learning to learn, which may be rapidly applied to different learning tasks. Meta-learning usually involves a bi-level optimization process, in which the internal learner provides feedback for the optimization of the meta-learner. Successful meta-learning applications include learning better initialization [29], architecture research [30], learning to optimize the learning rate schedule [31], and learning to optimize [32]. In order to learn better initialization

TABLE I
COMPARISON OF METABERT WITH OTHER STATIC AND DYNAMIC MODELS.

Model	QQP	MNLI	QNLI	SST-2	MRPC	RTE	CoLA	Average
				Base Model				
BERT-base	88.7	83.5	86.8	91.5	88.2	67.1	54.7	80.1
				Static Models				
BERT-6L	84.8	77.1	80.2	88.6	83.9	60.3	36.4	73.0
DistilBERT	86.4	79.8	82.2	89.3	85.3	63.1	46.5	76.1
LayerDrop	86.2	80.1	81.7	89.1	85.2	62.8	43.2	75.5
BERT-PKD	86.8	80.6	82.5	89.7	85.5	63.4	47.4	76.6
BERT-of-Theseus	86.5	80.7	82.4	89.6	85.4	63.6	44.5	76.1
				Dynamic Models				
GAML-BERT	89.1	83.2	85.1	91.9	88.1	66.8	51.1	79.3
Shallow-Deep	87.2	81.1	82.6	90.1	85.7	64.1	44.5	76.5
FastBERT	88.1	82.1	83.6	90.8	86.7	64.9	48.6	77.8
DeeBERT	87.3	81.2	82.5	90.0	85.8	63.9	43.4	76.3
PABEE	87.5	81.5	83.1	90.5	86.2	64.5	45.2	76.9
MetaBERT(Ours)	**90.1**	**85.6**	**92.3**	**93.5**	**91.7**	**70.8**	**58.1**	**83.2**

parameters, we adopt a collaborative meta-learning approach to improve the ability of early exit classifiers to extract semantic knowledge.

III. METHOD

We first start with a BERT that has been pre-trained as a backbone. Then, we introduce a collaborative meta-learning training method. In particular, we get the optimized initialization parameters θ according to Algorithm 1. Next, with the help of these parameters, we train each early exit classifier by Algorithm 2.

A. Backbone Model

In the original BERT [2], it is straightforward to pick up a classifier at the end of the transformer layer, and update the backbone model and classifier on the downstream task. The classifier at the last layer is a fully connected network that takes the hidden state h of the last layer as input, and then outputs the prediction. To implement the idea of early exit, we add a classifier after each transformer layer, e.g., BERT has twelve classifiers for twelve transformer layers. Each early exit classifier can give its own prediction result to facilitate the inference process by exiting early.

Specifically, we use the multi-exit structure of BERT as the backbone, i.e., we add additional classifiers to all transformer layers of BERT. The $[CLS]$ token corresponding to the hidden state of the i_{th} layer is denoted as

$$h_i = f_i(x; \varOmega_1, \varOmega_2, \varOmega_3, ..., \varOmega_i), \tag{1}$$

where x is the input sequence, $\varOmega_i$ is the parameters of the i_{th} transformer layer of BERT, and f_i is the mapping from the input to the hidden state of the i_{th} layer.

B. Training of Classifiers

As mentioned in the literature [33], for the BERT series of models, the shallow layers learn surface features, the middle layers learn syntactic features, and the high layers learn semantic features. In order to improve the representation ability of the early exit classifiers and adapt it to the requirements of different NLP tasks, we present a training method of meta-learning. The illustration of our meta-learning approach and the training process of MetaBERT during inference is shown in Algorithm 1 and Algorithm 2.

Formally, we first consider a model represented by a parameterized function f_θ denoted by the parameters θ. For a new task $\mathcal{T}_i$, the parameters θ of the model becomes θ_i'. The vector θ_i' is computed using one or more gradient updates on task $\mathcal{T}_i$:

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}\left(f_\theta\right), \tag{2}$$

where α is the step size.

The model parameters are trained by optimizing for the performance of $f_{\theta_i'}$ with regard to θ across tasks sampled from $p(\mathcal{T})$. At this point, the meta-objective becomes

$$\min_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}\left(f_{\theta_i'}\right) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}\left(f_{\theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i(f_\theta)}}\right), \tag{3}$$

here, the meta-optimization is performed on the model parameters θ, while the objective is computed from the updated model parameters θ'.

The cross-task meta-optimization is performed by stochastic gradient descent, and θ are updated in the following way:

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}\left(f_{\theta_i'}\right), \tag{4}$$

where β is the meta step size.

The optimized model parameters θ are obtained from Algorithm 1, and then we pass them into the Adam optimizer as a way to train the early exit classifiers. To be specific, the loss of the early exit classifier for each transformer layer can be expressed as

$$\mathcal{L}_i(x, y) = M\left(y, c_i\left(h_i; w_i\right)\right), \tag{5}$$

where x is the input sequence, y is its corresponding label, c_i is the classifier of the i_{th} exit layer, w_i is the parameters corresponding to c_i, and M is the task-specific loss function. Here we use cross-entropy loss.

TABLE II
COMPARISON BETWEEN DIFFERENT TRAINING METHODS.

Model	CoLA	RTE	MRPC	SST-2	Average
BERT-base	54.7	67.1	88.2	91.5	75.4
Training Methods					
Joint	55.5	65.3	89.4	90.8	75.3
Two-stage	56.5	67.8	91.5	92.7	77.1
Alternating	55.7	68.2	91.3	91.8	76.8
MetaBERT(Ours)	**58.1**	**70.8**	**91.7**	**93.5**	**78.5**

Following [12], we calculate the prediction confidence $Entr$ using the entropy of the output distribution P_i of i_{th} layer:

$$Entr = entropy(P_i). \tag{6}$$

The inference is terminated when the confidence $Entr$ falls below a predefined threshold γ. The hyper-parameter γ is adjusted according to the required speed-up ratio.

IV. EXPERIMENTS

A. Datasets and Evaluation Metrics

To evaluate the acceleration effect of the proposed MetaBERT on the inference of PLMs, we conduct experiments on the General Language Understanding Evaluation (GLUE) benchmark [34]. The GLUE benchmark includes QQP and MRPC [35] for similarity and paraphrase. QNLI [36] and MNLI [37] for natural language inference. SST-2 [38] for sentiment analysis. RTE for textual entailment and CoLA [39] for linguistic acceptability. For convenience, the average of accuracy and F1 score is used as the evaluation metric for QQP and MRPC, matthews correlation coefficient is used as the evaluation metric for CoLA, while accuracy is used as the evaluation metric for the remaining tasks.

B. Baselines

The baselines include the BERT backbone model, static models, and dynamic models. Static models, such as BERT-6L, DistilBERT [40], LayerDrop [8], BERT-PKD [41], and BERT-of-Theseus [7] are applied. In addition, dynamic models, including GAML-BERT [13], Shallow-Deep [11], FastBERT [14], DeeBERT [12], and PABEE [10] are introduced for comparison.

C. Implementation Details

The implementation of our proposed method is based on the HuggingFace Transformers library [42] and the learn2learn library [43]. The results of all experiments are obtained from a single NVIDIA V100 16GB GPU. Following previous work [12], the training batch size is 32. The inference batch size for the proposed method and baselines is 1. The model is optimized with Adam, and the learning rate is 2e-5. For the hyper-parameter settings in the meta-learning algorithm, we are consistent with the literature [29]. For the training epochs, we set it to 5. For the setting of the threshold γ, we follow [12].

D. Performance Comparison

Overall Performance. To demonstrate the effectiveness of the proposed approach, we compare our model performance with the state-of-the-art approaches and show the results in Table I and Table II. As shown, our method achieves better results than the original models on most datasets. The success of our model might be attributed to the training method of collaborate meta-learning. Our approach, which differs from other approaches, takes into account the linguistic knowledge inherent in every layer and is able to deliver more accurate predictions.

Comparison with Static and Dynamic Models. In order to verify the robustness and efficiency of MetaBERT, we first compare it with the mainstream static and dynamic models in Table I. As can be seen in Table I, our proposed approach achieves state-of-the-art results. Concretely, we address the problem of lack of semantic knowledge of early exit classifiers, and through the idea of meta-learning, we can achieve a balance between the inference result and efficiency.

Comparison of Training Methods. To further demonstrate the performance and efficiency tradeoff between our proposed model and baseline models, we compare four training methods: joint training, two-stage training, alternating training, and MetaBERT. From Table II, MetaBERT outperforms and dominates the other three training methods across the board on the selected tasks. Joint training performs the worst of the four training methods due to the fact that it treats all early exit classifiers equally, resulting in poor overall performance. Two-stage training achieves good performance, but is still inferior to our proposed method. Its disadvantage is that, on the one hand, it increases the performance of the final layer classifier at the expense of other shallow classifiers. On the other hand, it is inefficient, requiring two phases to complete classifier training. As for alternating training, although it combines the advantages of joint training and two-stage training, it does not reach the optimum among all training methods.

From Fig. 2, it can be seen that MetaBERT significantly exceeds the other three training methods. To be more specific, we observe a performance boost as the inaccessible features of new tasks are approximated by collaborative meta-learning. This shows that the high-layer semantic knowledge incorporated in the deep layers helps to improve the performance through our training framework.

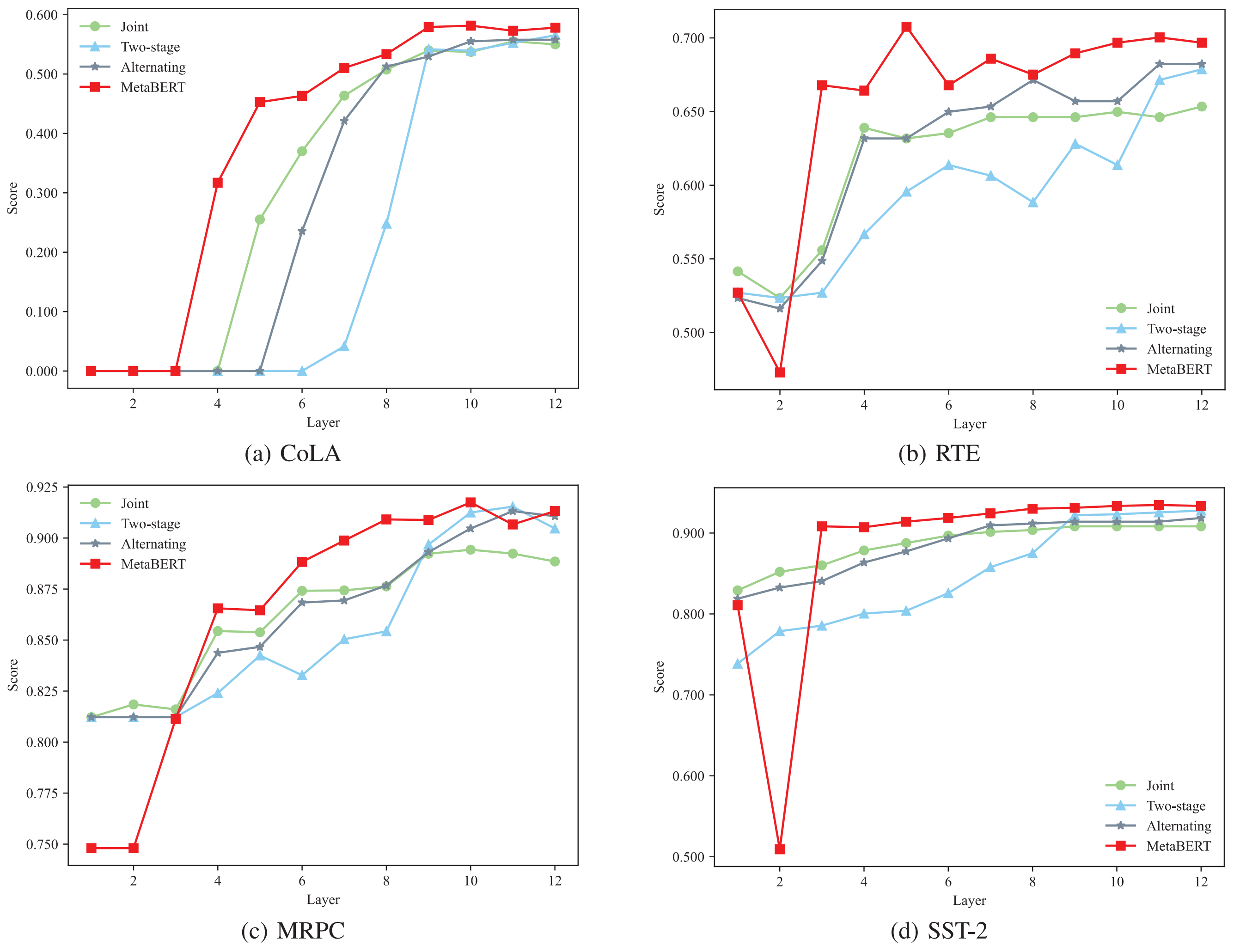

Fig. 2. Comparison of four training methods. These training methods include: joint training, two-stage training, alternating training, and MetaBERT. The performance of the twelve classifiers connected to the transformer layers is shown above. *Score* refers to the evaluation metric for the corresponding task.

V. Conclusion

In this paper, we propose a collaborative meta-learning training strategy that significantly decreases computation consumption while improving the speed of inference with a minor performance loss. Unlike previous work, our model employs all available linguistic information for prediction and engages features from new tasks which are originally inaccessible for prediction. Extensive experimental results show that the proposed method provides a better tradeoff between the model performance and inference efficiency. In the future, it will be very interesting to combine our model with pruning or other early exit methods.

Acknowledgment

This work is supported by Key National Defense Science and Technology Innovation Program of the Chinese Academy of Sciences (No. KGFZD-145-22-16), Strategic Priority Research Program of the Chinese Academy of Sciences (No.XDC02030400).

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "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, pp. 4171–4186.

[3] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," in *International Conference on Learning Representations, ICLR*, 2020.

[4] H. Ye, S. Lu, and D. Zhan, "Generalized knowledge distillation via relationship matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 2, pp. 1817–1834, 2023.

[5] L. Cai, J. Wang, L. Yu, B. Yan, Y. Tao, and Y. Yang, "Accelerating neural-ode inference on fpgas with two-stage structured pruning and history-based stepsize search," in *Proceedings of the 2023 ACM/SIGDA International Symposium on Field Programmable Gate Arrays, FPGA*, 2023, pp. 177–183.

[6] P. Wang, W. Chen, X. He, Q. Chen, Q. Liu, and J. Cheng, "Optimization-based post-training quantization with bit-split and stitching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 2, pp. 2119–2135, 2023.

[7] C. Xu, W. Zhou, T. Ge, F. Wei, and M. Zhou, "BERT-of-theseus: Compressing BERT by progressive module replacing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP*, 2020, pp. 7859–7869.

[8] A. Fan, E. Grave, and A. Joulin, "Reducing transformer depth on demand with structured dropout," in *International Conference on Learning Representations, ICLR*, 2019.

[9] A. Graves, "Adaptive computation time for recurrent neural networks," *arXiv preprint arXiv:1603.08983*, 2016.

[10] W. Zhou, C. Xu, T. Ge, J. J. McAuley, K. Xu, and F. Wei, "BERT loses patience: Fast and robust inference with early exit," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 18 330–18 341.

[11] Y. Kaya, S. Hong, and T. Dumitras, "Shallow-deep networks: Understanding and mitigating network overthinking," in *Proceedings of the 36th International Conference on Machine Learning, ICML*, vol. 97, 2019, pp. 3301–3310.

[12] J. Xin, R. Tang, J. Lee, Y. Yu, and J. Lin, "Deebert: Dynamic early exiting for accelerating bert inference," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL*, 2020, pp. 2246–2251.

[13] W. Zhu, X. Wang, Y. Ni, and G. Xie, "Gaml-bert: Improving bert early exiting by gradient aligned mutual learning," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP*, 2021, pp. 3033–3044.

[14] W. Liu, P. Zhou, Z. Wang, Z. Zhao, H. Deng, and Q. Ju, "Fastbert: a self-distilling bert with adaptive inference time," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL*, 2020, pp. 6035–6044.

[15] Y. Matsubara, M. Levorato, and F. Restuccia, "Split computing and early exiting for deep learning applications: Survey and research challenges," *ACM Comput. Surv.*, vol. 55, no. 5, pp. 90:1–90:30, 2023.

[16] J. Kong, J. Wang, and X. Zhang, "Accelerating pretrained language model inference using weighted ensemble self-distillation," in *Natural Language Processing and Chinese Computing - 10th CCF International Conference, NLPCC*, vol. 13028, 2021, pp. 224–235.

[17] J. Kong, J. Wang, L. Yu, and X. Zhang, "Accelerating inference for pretrained language models by unified multi-perspective early exiting," in *Proceedings of the 29th International Conference on Computational Linguistics, COLING*, 2022, pp. 4677–4686.

[18] J. Xin, R. Tang, Y. Yu, and J. Lin, "BERxiT: Early exiting for BERT with better fine-tuning and extension to regression," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL*, 2021, pp. 91–104.

[19] Z. Sun, H. Yu, X. Song, R. Liu, Y. Yang, and D. Zhou, "Mobilebert: a compact task-agnostic BERT for resource-limited devices," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL*. Association for Computational Linguistics, 2020, pp. 2158–2170.

[20] H. Sajjad, F. Dalvi, N. Durrani, and P. Nakov, "Poor man's bert: Smaller and faster transformer models," *arXiv preprint arXiv:2004.03844*, 2020.

[21] Z. Gao, Y. Yao, S. Zhang, J. Yang, M. Lei, and I. McLoughlin, "Extremely low footprint end-to-end ASR system for smart device," in *Interspeech 2021, 22nd Annual Conference of the International Speech Communication Association*. ISCA, 2021, pp. 4548–4552.

[22] T. Sun, X. Liu, W. Zhu, Z. Geng, L. Wu, Y. He, Y. Ni, G. Xie, X. Huang, and X. Qiu, "A simple hash-based early exiting approach for language understanding and generation," in *Findings of the Association for Computational Linguistics: ACL*, 2022, pp. 2409–2421.

[23] S. Laskaridis, A. Kouris, and N. D. Lane, "Adaptive inference through early-exit networks: Design, challenges and directions," in *EMDL@MobiSys 2021: Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning*, 2021, pp. 1–6.

[24] A. S. Davis and I. Arel, "Low-rank approximations for conditional feedforward computation in deep neural networks," in *2nd International Conference on Learning Representations, ICLR*, 2014.

[25] M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and Ł. Kaiser, "Universal transformers," *arXiv preprint arXiv:1807.03819*, 2018.

[26] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016, pp. 2464–2469.

[27] M. Elbayad, J. Gu, E. Grave, and M. Auli, "Depth-adaptive transformer," in *8th International Conference on Learning Representations, ICLR*. OpenReview.net, 2020.

[28] R. Schwartz, G. Stanovsky, S. Swayamdipta, J. Dodge, and N. A. Smith, "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*. Association for Computational Linguistics, 2020, pp. 6640–6651.

[29] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 36th International Conference on Machine Learning, ICML*, vol. 70, 2017, pp. 1126–1135.

[30] H. Liu, K. Simonyan, and Y. Yang, "DARTS: differentiable architecture search," in *7th International Conference on Learning Representations, ICLR*. OpenReview.net, 2019.

[31] A. G. Baydin, R. Cornish, D. Martínez-Rubio, M. Schmidt, and F. Wood, "Online learning rate adaptation with hypergradient descent," in *6th International Conference on Learning Representations, ICLR*. OpenReview.net, 2018.

[32] M. Andrychowicz, M. Denil, S. G. Colmenarejo, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas, "Learning to learn by gradient descent by gradient descent," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems*, 2016, pp. 3981–3989.

[33] G. Jawahar, B. Sagot, and D. Seddah, "What does BERT learn about the structure of language?" in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL*, 2019, pp. 3651–3657.

[34] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP*, 2018, pp. 353–355.

[35] W. B. Dolan and C. Brockett, "Automatically constructing a corpus of sentential paraphrases," in *Proceedings of the Third International Workshop on Paraphrasing, IWP@IJCNLP*. Asian Federation of Natural Language Processing, 2005.

[36] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100, 000+ questions for machine comprehension of text," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP*. The Association for Computational Linguistics, 2016, pp. 2383–2392.

[37] A. Williams, N. Nangia, and S. R. Bowman, "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*. Association for Computational Linguistics, 2018, pp. 1112–1122.

[38] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP*. ACL, 2013, pp. 1631–1642.

[39] A. Warstadt, A. Singh, and S. R. Bowman, "Neural network acceptability judgments," *Trans. Assoc. Comput. Linguistics*, vol. 7, pp. 625–641, 2019.

[40] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[41] S. Sun, Y. Cheng, Z. Gan, and J. Liu, "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, pp. 4323–4332.

[42] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP*, 2020, pp. 38–45.

[43] S. M. Arnold, P. Mahajan, D. Datta, I. Bunner, and K. S. Zarkias, "learn2learn: A library for Meta-Learning research," *arXiv preprint arXiv:2008.12284*, 2020.