LoRAExit: Empowering Dynamic Modulation of LLMs in Resource-limited Settings using Low-rank Adapters

Jiacheng Liu[1], **Peng Tang**[2], **Xiaofeng Hou**[2], **Chao Li**[2], **Pheng-Ann Heng**[1,3]
[1]Department of Computer Science and Engineering, The Chinese University of Hong Kong
[2]Department of Computer Science and Engineering, Shanghai Jiao Tong University
[3]Institute of Medical Intelligence and XR, The Chinese University of Hong Kong
{jcliu,pheng}@cse.cuhk.edu.hk,tttppp@sjtu.edu.cn,{hou-xf,lichao}@cs.sjtu.edu.cn

Abstract

Large Language Models (LLMs) have exhibited remarkable performance across various natural language processing tasks. However, deploying LLMs on resource-limited settings remains a challenge. While early-exit techniques offer an effective approach, they often require compromised training methods that result in sub-optimal performance. On the other hand, multi-model methods achieve improved results but suffer from significant inference latency and memory consumption. In this paper, we propose *LoRAExit*, a novel dynamic inference architecture that leverages low-rank adapters for efficient deployment of LLMs. *LoRAExit* decouples the training of multiple exit interfaces, enabling the separate optimization of each exit, thereby fundamentally addressing the performance issues of early-exit networks. Moreover, we introduce a superior-exit guided distillation method that effectively utilizes models of different sizes, thereby further enhancing the performance of early exits. Experimental results demonstrate that *LoRAExit* significantly improves LLM performance when deployed on resource-limited settings.

1 Introduction

Large Language Models (LLMs) have achieved remarkable success in various natural language processing tasks, demonstrating their potential for advancing the state-of-the-art in language understanding and generation (Chowdhery et al., 2022; Vaswani et al., 2017). However, the deployment of LLMs on resource-limited settings, such as edge devices and systems with constrained computational resources, presents significant challenges (Treviso et al., 2023; Hou et al., 2024). The computational demands of LLMs, characterized by their large size and high memory requirements, hinder their efficient deployment in such settings.

In recent years, there has been significant research exploring various approaches to enhance the efficiency of LLMs during inference. One prominent approach is adaptive inference, where computations are dynamically allocated based on data difficulty (Han et al., 2021). Among the effective methods, early-exit networks have garnered attention, enabling predictions at different inference stages and allowing for early termination when confident predictions are obtained (Teerapittayanon et al., 2016; Hou et al., 2023b). However, these early-exit techniques often rely on compromised training methods, which can lead to sub-optimal performance. Figure 1 illustrates three representative training methods. The joint training method optimizes all parameters simultaneously, while the two-stage training method involves training the LLM backbone in the first stage and the intermediate classifiers in the second stage. These methods suffer from conflicting gradient issues that can

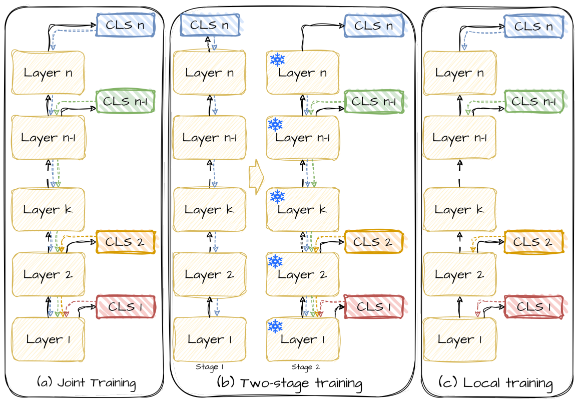


Figure 1: Limitations of existing training methods for early exit networks. (a) The joint training method optimizes all parameters simultaneously. (b) The two-stage training method involves training the backbone of the LLM in the first stage, followed by training the classifiers in the second stage. (c) The local training method employs heuristics to distribute each loss to specific layers, aiming to mitigate conflicting gradients. The black solid line represents the forward process, and the colored dotted line represents the backward process.

significantly impact model performance. The local training method mitigates this issue by heuristically distributing each loss to specific layers, attempting to alleviate the conflicting gradients. However, it remains challenging to accurately distribute the loss to specific parameters, and simple heuristics may inadvertently harm performance. Another notable approach is multi-model methods, which leverage combinations of models to achieve improved results (Mamou et al., 2023). Nevertheless, these methods suffer from increased inference latency and memory consumption, posing challenges in resource-limited settings.

Overcoming these limitations and building an efficient adaptive model present two key challenges. *Firstly, we need to train a model of different sizes without compromising its performance. Secondly, we must ensure that the model can run without significantly increasing the resource requirements.* These challenges are inherently conflicting because sharing parameters in early exits can degrade performance, while completely separating parameters will inevitably increase the resource requirements.

To tackle these challenges, we propose *LoRAExit*, a novel dynamic inference architecture specifically designed for efficient deployment of LLMs in resource-limited settings. *LoRAExit* utilizes low-rank adapters to decouple the training of multiple exit interfaces in LLMs. By enabling the separate optimization of each exit interface, *LoRAExit* effectively addresses the performance issues associated with early-exit networks. Additionally, we introduce a superior-exit guided distillation method in *LoRAExit*, which effectively leverages models of different sizes to aid the training of small models with early exits. In *LoRAExit*, the LoRA adapter enables the training of multiple exits without conflicting gradients. Concurrently, our novel batching mechanism facilitates efficient inference. The early exiting strategy further complements this by trimming inference time. The harmonization of these two approaches allows us to optimize both training and inference workflows collectively.

To assess the efficacy of *LoRAExit*, we conducted extensive experiments in resource-limited settings. Our evaluation focuses on real-system speedup as a performance metric, considering that LLMs commonly operate on modern accelerators like GPUs where computation (typically measured in FLOPs) is not the primary bottleneck, but rather factors like memory play a more crucial role (Miao et al., 2023). The experimental results demonstrate that *LoRAExit* significantly enhances LLM performance, surpassing existing approaches when deployed in such settings. In summary, we make the following contributions:

- We identify the performance, latency and memory dilemma of the existing adaptive inference algorithms.
- We propose *LoRAExit*, a novel adaptive inference architecture that builds upon parameter-efficient tuning techniques to fulfill the memory and latency requirement while not compromising the performance.
- A superior-exit guided distillation method is proposed to utilize the suitable information in different exits.
- We evaluate *LoRAExit* on various scenarios and demonstrate that it substantially outperforms the previous state-of-the-art solutions.

2 Background & Motivation

2.1 LLM and Adaptive Inference

LLMs have emerged as powerful tools for natural language processing (NLP) tasks, exhibiting remarkable performance across a wide range of applications (Bommasani et al., 2021). LLMs, such as BERT (Devlin et al., 2019) and GPT (Brown et al., 2020), are deep neural networks with millions or even billions of parameters, enabling them to capture intricate language patterns and generate coherent and contextually relevant text. However, the deployment of LLMs in resource-limited settings presents significant challenges due to their computational demands (Miao et al., 2023).

Adaptive inference techniques aim to address these challenges by optimizing the deployment of LLMs in resource-limited environments (Han et al., 2021). These techniques focus on improving the efficiency of LLMs to reduce computational requirements and enable real-time language processing on devices with limited resources. Two prominent approaches for adaptive inference are early-exit techniques and multi-model methods. These approaches make it possible to adjust the computational budget with different sample difficulties.

2.2 Performance Issue in Early-Exit

Early-exit techniques allow LLMs to make predictions at different stages of inference and terminate

	PABEE	PALBERT	DREE	Optimal
Layer 3	64.24	64.63	40.68	**72.27**
Layer 6	73.38	74.18	34.34	**78.95**
Layer 9	80.72	79.22	47.05	**82.26**
Layer 12	81.73	81.10	75.46	**82.60**

Table 1: The performance of each exit of different early-exit training methods. Bold value is the highest performance in each layer.

	Params	EE		MM	
		Mem.	Lat.	Mem.	Lat.
base	86	280	12.88	690	24.96
large	350	786	24.73	1,625	44.52
xlarge	710	1,717	22.72	3,934	41.7
xxlarge	1,320	3,023	44.17	5,997	74.83

Table 2: Comparison of memory and latency between EE and MM for various model sizes. The parameters are measured in millions, memory is indicated in megabytes, and latency is expressed in milliseconds.

early when confident predictions are obtained (Teerapittayanon et al., 2016). This approach aims to reduce computational costs by avoiding unnecessary processing for instances where accurate predictions can be made early in the inference process. While early-exit techniques can improve efficiency, they often require compromised training methods that may lead to sub-optimal performance compared to fully trained LLMs (Rotem et al., 2023; Zhu et al., 2023). Specifically, there are three mainstream early-exit training methods. The joint training method and two-stage training method involve merging the final prediction with intermediate classifiers, resulting in a "conflicting gradient" problem which occurs when different exits in a model share parameters, leading to gradients that conflict during the training process. This phenone is observed in diverse studies (Zhu et al., 2023; Rotem et al., 2023; Ji et al., 2023b). Recent works propose local training methods that assign a subset of parameters to each exit, addressing the conflicting gradient issue but potentially limiting the model's expressiveness (Rotem et al., 2023).

To empirically validate this observation, we conduct fine-tuning experiments on three representative early-exit models with a BERT BASE backbone, incorporating four exit points corresponding to layers [3, 6, 9, 12] (for detailed experimental settings, please refer to Section 4). Additionally, we add the performance of training a same-size model independently as the optimal performance. The experimental results, presented in Table 1, clearly demonstrate that existing training methods generally struggle to train high-performing models, particularly in the earlier exit points where the performance of these early-exit training methods falls significantly short of optimal levels.

2.3 Memory and Latency Issue in MM

Multi-model methods involve the combination of multiple models to enhance overall performance (Mamou et al., 2023). During the inference process, these methods gradually execute models of increasing sizes until satisfactory performance is achieved, at which point the process is terminated with the obtained predictions. However, a drawback of multi-model methods is the increased inference latency and higher memory consumption. This is primarily due to the necessity of loading and executing all the models in memory sequentially.

We performed a series of experiments utilizing varying sizes of the DeBERTa model (He et al., 2020), ranging from 86M to 1.3B parameters. The results are presented in Table 2, which indicate that as the model size increases, the multi-model approach exhibits a nearly twofold increase in memory requirements and runs approximately twice as slow when compared to the early-exit method.

2.4 Motivations

The challenges associated with deploying LLMs in resource-limited environments serve as a driving force for the exploration of efficient and high-performance approaches that overcome the limitations of early-exit techniques and multi-model methods. It is crucial to develop an approach that optimizes LLM deployment without compromising accuracy or increasing inference latency and memory consumption. Recent advancements in parameter-efficient tuning methods (Xu et al., 2023) provide a new opportunity to design a novel early-exit architecture that can be trained without conflicting gradients. Notably, extensive experiments have demonstrated the superior performance of the LoRA tuning method compared to full parameter fine-tuning in most scenarios (Hu et al., 2022). These promising findings inspire us to consider the potential of leveraging LoRA to construct an improved adaptive model.

3 Methodology

In this section, we introduce the proposed adaptive model architecture, which takes advantage of the

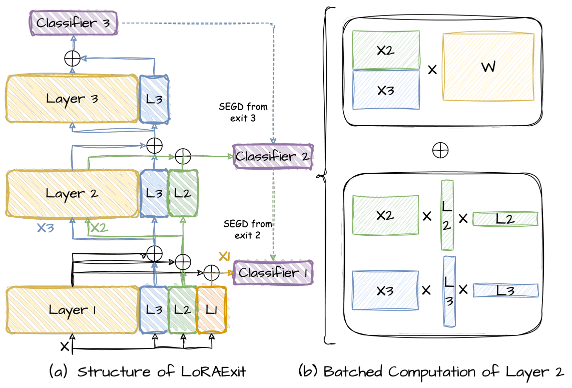


Figure 2: The architecture of a demonstrated three-layer transformer featuring the proposed LoRAExit for early exit. (a) The structure of the transformer layer, where multiple LoRA heads are individually added to maximize performance. In each layer, different LoRA heads are applied separately for different exits to eliminate the conflicting gradient problem. During training, SEGD is applied to each exit and its superior exit to enhance performance. (b) The computation pattern during inference for the second layer, shows how efficient batch processing enables early exit with minimal memory overhead.

powerful low-rank adapters to achieve high performance and low overhead (Section 3.1). Additionally, we introduce a superior-exit guided distillation method, specifically designed for the completely separated exit architecture, which effectively enhances the performance of intermediate exits (Section 3.2). Finally, we introduce the batched inference operator that can ddresses the computational overhead introduced by employing multiple LoRA heads (Section 3.3).

3.1 Model Architecture

Throughout this section, we focus on the discriminative LLMs (e.g., BERT) and address the multi-class classification scenario with samples $(x_i, y_i)_{i=1}^{n}$, where x_i represents a token sequence and y_i denotes its corresponding label. Given an input sequence, each transformer layer L_i computes the hidden state h_i based on the previous hidden state h_{i-1} using the following equation:

$$h_i = \mathrm{LN}(\mathrm{FFN}(\mathrm{LN}(\mathrm{SA}(h_{i-1}) + h_{i-1}))) \qquad (1)$$

where SA refers to the self-attention operator, FNN represents the feed-forward network, and LN denotes the layer norm layer. The computation of these layers can be computationally expensive, particularly for large models with numerous layers.

In this paper, we propose to maintain the expressiveness of the model by incorporating low-rank adapters (LoRA) and completely dissociating the effects of different exit branches, as depicted in Figure 2. For each exit, we introduce a series of LoRA modules in each layer leading up to the exit. Specifically, for an exit attached at backbone layer i, we add LoRA modules from layer 1 to layer i. For instance, in Figure 2, for exit classifier 2, LoRA modules are added at layer 1 and layer 2. For a d-dimensional input x, the calculation of the LoRA module is defined as:

$$x' = x(W_i + A_i B_i) \qquad (2)$$

where $W_i \in \mathbb{R}^{d\times d}$ represents the original weight, while $A_i \in \mathbb{R}^{d\times m}$ and $B_i \in \mathbb{R}^{m\times d}$ are the low-rank adapters, where $d \gg m$. The low-rank adapter is a novel technique that significantly reduces the number of parameters in the model while preserving its expressiveness (Hu et al., 2022). Following previous efforts, we only add LoRA weights in the query and value vectors in the self-attention layer. Subsequently, we introduce the early exit after the output of each layer, augmented by the LoRA output. The exit head is implemented as a feed-forward network.

This architecture effectively separates the gradients from each exit, simplifying the model training process and mitigating the issue of conflicting gradients (Zhu et al., 2023). Simultaneously, by leveraging the powerful low-rank adapters, our method preserves the expressiveness of the model. These characteristics differentiate our approach from existing early-exit methods. In contrast to conventional methods that simply attach the exit head to the backbone (Zhou et al., 2020), our method resolves the conflicting gradient issue. Furthermore, our approach differs from recently proposed local training methods, which only limit the training of each exit to a small fraction of parameters, significantly impacting performance (Rotem et al., 2023).

3.2 Superior-Exit Guided Distillation

The *LoRAExit* architecture offers the advantage of training the model without being influenced by conflicting gradients, thereby achieving improved performance. Furthermore, we found that this fundamental design allows for the integration of other effective techniques such as knowledge distillation (Gou et al., 2020). In a multiple-exit architecture, various exits are positioned at different locations within the model. We define a superior exit as the exit that immediately follows a given exit.

For instance, for exit k, the superior exit is $k+1$. Leveraging the hierarchical relationships among these exits, we propose a method called Superior-Exit Guided Distillation (SEGD) to enhance the performance of earlier exits. This enhancement is very important for better balancing accuracy and latency in *LoRAExit*.

Initially, we train the final layer (layer N) of the model with a classification head using conventional cross-entropy loss:

$$\mathcal{L}_N(\hat{y}_i) = -\sum \hat{y}_i \log(y_i^N) \tag{3}$$

where, y_i^N represents the output of the final exit for the i-th input, and $\hat{y}_i$ is the ground truth label. Subsequently, we progressively employ the predictions of the superior exit $k+1$ as soft labels to distill the next exit k using the following loss formulation:

$$\mathcal{L}_k(\hat{y}_i) = (1-\lambda)\mathcal{L}_k(y_i^k, \hat{y}_i) + \lambda\mathcal{KL}(\bar{y}_i^{k+1}, \bar{y}_i^k) \tag{4}$$

where λ is the weight of the distillation loss, $\mathcal{KL}$ is the Kullback-Leibler divergence, $\bar{y}_i^{k+1}$ and $\bar{y}_i^k$ is the soft label from the superior exit and this exit which is defined as,

$$\bar{y}_i^k = \frac{\exp(z_i^k/T)}{\sum_j \exp(z_j^k/T)} \tag{5}$$

where z_i^k represents the logits of the k-th exit, and T is the temperature parameter. By distilling from the superior exit rather than the final classification layer, *LoRAExit* achieves enhanced performance, as it reduces the gap between the teacher and student models (Mirzadeh et al., 2019), which is critical for the earlier layers.

3.3 Adaptive Inference

Upon training the *LoRAExit* model, achieving fast inference speed during model execution becomes crucial. The architecture of *LoRAExit* consists of multiple LoRA modules, allowing for the utilization of advanced accelerators (e.g., GPUs) equipped with highly efficient operators (Sheng et al., 2023; Chen et al., 2023). Specifically, we divide the computation of each layer into the backbone and the LoRA heads. The computation can be expressed as follows:

$$\begin{pmatrix} \vec{y_1} \\ \vdots \\ \vec{y_n} \end{pmatrix} = \begin{pmatrix} \vec{x_1} \\ \vdots \\ \vec{x_n} \end{pmatrix} W + \begin{pmatrix} \vec{x_1} A_1 B_1 \\ \vdots \\ \vec{x_n} A_n B_n \end{pmatrix} \tag{6}$$

The first part of the computation can be efficiently batched using the CUDA GEMM operator, while the second part can be effectively batched using the SGMV operator (Chen et al., 2023). Consequently, each layer of the *LoRAExit* architecture can be processed efficiently on modern hardware with minimal overhead. Once the features for each layer are obtained, we employ a confidence-based approach to determine the appropriate exit point.

4 Experiments and Results

This section presents the experimental setup, datasets used, and the results obtained through the evaluation of LoRAExit.

4.1 Datasets

Our evaluation of the proposed *LoRAExit* method is conducted over seven classification tasks from the GLUE benchmark (Wang et al., 2018), comprising CoLA, RTE, MRPC, QQP, SST-2, QNLI, and MNLI. For tasks featuring multiple metrics, we follow the approach of (Ji et al., 2023b), reporting the arithmetic mean of these metrics. Specifically, CoLA is evaluated using Matthew's correlation, QQP and MRPC are assessed by averaging accuracy and F1 scores, and the remaining tasks are evaluated based on accuracy alone. The details of these dataset can be found in Appendix A.1.

4.2 Baselines

To evaluate the performance of *LoRAExit*, baseline models and methods are compared. This subsection describes the baselines used and explains the rationale behind their selection.

We employ the commonly used BERT-base (Devlin et al., 2018) and ALBERT-base (Lan et al., 2019) to compare our method with current early exit studies fairly. In addition, we include a different model family the DeBERTa model (He et al., 2020), as for DeBERTa, we use DeBERTa-base, DeBERTa-large, and DeBERTa-xlarge which compose a benchmark to test *LoRAExit* in different model families and sizes.

To evaluate *LoRAExit* 's performance, we compare it with following state-of-the-art methods,

PABEE. PABEE is a patience-based early exit strategy, which stops the inference process when the number of times that the predictions remain "unchanged" reaches the predefined value (Zhou et al., 2020). This approach optimizes computational efficiency by eliminating unnecessary processing once the model's outputs stabilize.

	Lat.	Layer	CoLA	RTE	MRPC	QQP	SST2	QNLI	MNLI	Avg.
BERT-base	1.00	12.00	61.10	70.75	89.22	89.82	93.11	91.85	83.96	82.83
PABEE	0.94	8.59	57.54	65.34	85.55	89.18	91.39	88.96	83.39	80.19
PALBERT	1.18	9.67	58.22	65.70	86.19	89.16	91.74	90.07	83.56	80.66
DREE	1.27	9.77	49.43	67.87	86.55	88.80	91.62	88.37	76.12	78.39
LoRAExit	**0.78**	**6.56**	**58.45**	**69.67**	**87.13**	**89.33**	**91.86**	**90.53**	**83.58**	**81.51**
ALBERT-base	1.00	12.00	51.71	76.90	89.44	80.11	91.51	88.30	74.93	78.99
PABEE	0.88	8.64	43.23	76.53	89.11	**80.46**	88.30	87.04	72.91	76.80
PALBERT	1.12	9.65	51.24	73.65	**91.27**	80.21	89.11	87.13	73.70	78.04
DREE	1.24	9.47	51.83	**77.62**	89.75	80.12	89.56	**88.38**	41.02	74.04
LoRAExit	**0.79**	**7.19**	**53.03**	77.26	90.99	79.25	**91.40**	87.31	**73.95**	**79.03**

Table 3: Experimental results with BERT and ALBERT backbone on the GLUE benchmark. Latency is measured as the normalized inference time of each method, calculated by dividing the method's time by that of the base model. The term "Layer" refers to the average number of layers utilized for inference across methods. "Average" denotes the mean performance score across all tasks.

PALBERT. PALBERT improves the performance of PonderNet with a novel deterministic Q-exit criterion and a revisited model architecture (Balagansky and Gavrilov, 2023). Given the absence of the BERT model implementation within PALBERT, we adapted its principles to BERT following (Ji et al., 2023b).

DREE. DREE is an adapter-based method for Early Exit to disentangle two conflicting representations, namely generic linguistic representations for subsequent layers and task-specific representations for internal classifiers (Ji et al., 2023b).

SWEET. SWEET is an Early-Exit fine-tuning method that assigns each classifier its own set of unique model weights, not updated by other classifiers, which relieves the conflicts between different exit classifiers (Rotem et al., 2023).

MM. MM uses a set of independent models of increasing capacity, each fine-tuned separately for the same task. We use the implementation of MM in (Rotem et al., 2023).

To enable a fair comparison with other baselines, we adopt specific settings considering the variations in backbone usage across different methods. In particular, we evaluate our proposed *LoRAExit* against PABEE, PALBERT, and DREE using both BERT and ALBERT backbones, as these methods provide implementations for these backbones. Additionally, we compare *LoRAExit* against SWEET and MM, specifically on the DeBERTa backbone.

The detailed training and inference setup of these models can be found in Appendix A.2.

4.3 Main Results

We first present the results using BERT and ALBERT backbones on GLUE datasets, comparing our approach with PABEE, PALBERT, and DREE, as shown in Table 3. Our analysis indicates that our technique necessitates a minimal number of layers and achieves the quickest inference times in all cases. Remarkably, it demands only 78% and 79% of the inference duration needed by the BERT-base and ALBERT-base models, respectively. Conversely, despite PALBERT and DREE employing around 10 layers for inference, their inference durations surpass those of the base models due to the computational burdens introduced by early exit mechanisms. Although PABEE expends less time compared to both PALBERT and DREE, owing to its reduced layer count, it still consumes about 10% more time than our strategy.

From a performance perspective, our method registers the topmost scores on all tasks using the BERT backbone. In the case of the ALBERT backbone, while our lead is not consistent across every task, we attain the highest overall average score, eclipsing even that of the base model. These demonstrate the effectiveness of the novel architecture that fundamentally addresses the conflicting gradient issue. Among the competitors, PALBERT notches the best average scores aside from ours, albeit at the cost of excessive layer usage. PABEE finds a more equitable compromise between layer count and performance outcomes. DREE, particularly underwhelming on the MNLI task, ends up with the lowest composite average score due to its

	Vanilla Train	KD	SEGD
Layer 3	71.05	71.47	**71.74**
Layer 6	76.66	78.34	**78.61**
Layer 9	80.93	**82.13**	**82.13**

Table 4: The performance of different training schemes. Layer 12 (the last layer) is omitted since SEGD does not affect this layer.

	BERT	ALBERT	DeBERTa
LoRAExit	**7.83**	**9.08**	**14.90**
ForLoop	12.58	12.47	19.00

Table 5: Inference time (ms) of once inference the entire model with *LoRAExit* and conventional ForLoop.

limited model expressiveness. Coupled with its high-layer requisites and the additional computational expense of its adapters, DREE's extended inference time culminates in the least favorable comprehensive assessment.

4.4 Ablation Study

We initially demonstrate the effectiveness of our superior-exit guided distillation (SEGD) approach by contrasting its performance with that of vanilla training without distillation (Vanilla Train) and distillation from the final layer (KD). The outcomes, presented in Table 4, reflect the average scores across all tasks in the GLUE benchmark utilized in our study. Analysis of the results at layers 3 and 6 indicates that our method outperforms the other two approaches. Moreover, distillation proves more effective than direct training at layers 3, 6, and 9. For layer 9, both superior-exit guided distillation and final-layer distillation employ layer 12 as the teacher model, resulting in identical outcomes.

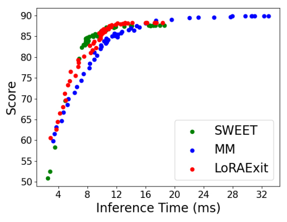
Furthermore, to highlight the benefits of the inference algorithm in *LoRAExit*, we compared it against traditional ForLoop computation across the three backbones. The findings, presented in Table 5, demonstrate that ForLoop computation leads to substantial overhead, whereas the SGMV operator achieves a speedup of 38%, 27% and 22% respectively, underscoring its efficiency and the advantage of using SGMV for adding the LoRA adapter.

4.5 Different Model Sizes

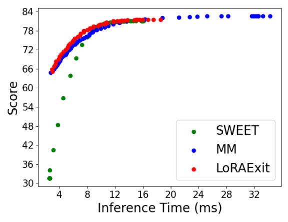
To further evaluate the capability of our method across different model sizes, we compare its performance with SWEET on three sizes of the DeBERTa model: base, large, and extra-large (He et al., 2020). As shown in Table 6, *LoRAExit* consistently outperforms SWEET at all exit layers across the three model sizes. Particularly for smaller exit layers, such as layers 1, 4, and 6, *LoRAExit* achieves significantly higher accuracy than SWEET regardless of the model size. Furthermore, Table 7 presents the performance and inference latency of both methods across low and high ranges. The results clearly indicate that *LoRAExit* not only delivers better performance but also maintains lower inference latency and a reduced number of layers. Specifically, *LoRAExit* shows more pronounced advantages in low-range and smaller models, making it suitable for environments with limited computing resources. Overall, these findings demonstrate that *LoRAExit* provides robust performance across various model sizes.

Size	Method	Exit Layer			
BASE		1	4	6	12
	SWEET	42.97	66.01	75.02	80.38
	LoRAExit	**57.88**	**73.82**	**77.96**	**80.78**
LARGE		1	6	12	24
	SWEET	41.96	65.81	75.92	83.59
	LoRAExit	**57.33**	**73.55**	**79.40**	**85.10**
XLARGE		1	12	24	48
	SWEET	42.62	77.58	84.61	84.94
	LoRAExit	**57.91**	**78.56**	**85.74**	**86.00**

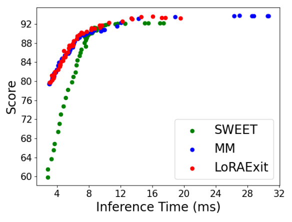
Table 6: The accuracy of each exit of SWEET and *LoRAExit* on three model sizes. Bold value is the highest accuracy in each layer.



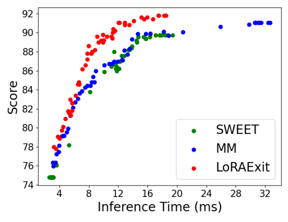
(a) QNLI



(b) QQP



(c) SST2



(d) MRPC

Figure 3: Time-accuracy curve of four tasks obtained using a DeBERTa model.

Size	Method	Low Range			High Range		
		Latency (ms)	Layer	Accuracy	Latency (ms)	Layer	Accuracy
BASE	SWEET	5.75	2.65	59.18	15.92	10.44	80.40
	LoRAExit	**4.12**	**1.60**	**62.14**	**15.60**	**9.06**	**80.64**
LARGE	SWEET	10.13	5.49	64.04	25.08	18.34	83.57
	LoRAExit	**8.52**	**3.77**	**67.56**	**24.97**	**16.49**	**84.65**
XLARGE	SWEET	10.79	6.85	64.12	35.22	27.69	84.89
	LoRAExit	**8.97**	**4.86**	**65.78**	**35.12**	**26.98**	**85.71**

Table 7: The performance of SWEET and *LoRAExit* on three model sizes. Latency is the average inference time for a single sample. "Low Range" indicates selections with low accuracy and low latency, while "High Range" indicates selections with high accuracy and high latency.

4.6 Speeds-Accuracy Tradeoff

To evaluate the trade-off between speed and accuracy of *LoRAExit*, we conducted a comparison with SWEET and MM, utilizing the DeBERTa-base backbone. Following (Rotem et al., 2023), we evaluate all methods across a spectrum of inference speeds by employing 51 threshold values, evenly distributed within the range $(\frac{1}{\text{\# of labels}}, 1)$, where *# of labels* denotes the number of labels for a given task. We focus our analysis on four distinct tasks: two larger datasets, QNLI and QQP, and two smaller datasets, SST2 and MRPC. Illustrated in Figure 3, our findings indicate that *LoRAExit* achieves a superior trade-off between speed and accuracy compared to the other methods. Notably, *LoRAExit* excels in achieving the highest score under conditions of limited inference time. In contrast, SWEET exhibits poor performance at shorter inference times due to inadequate training of its lower layers. Meanwhile, MM requires significantly longer inference times.

5 Related Work

5.1 Adaptive Inference

Adaptive inference has emerged as an effective means to enhance the efficiency of pre-trained language models during inference (Miao et al., 2023). A naive approach to addressing the limitations of single-model inference is the use of multi-model approaches (Li et al., 2020). These methods sequentially try a series of models, ranging from simple to complex. While this approach can effectively handle a batch of input examples, it can result in longer inference latency when processing a single sample. Additionally, it may require more memory to store all the models, which can be impractical for LLMs. In contrast, early exit strategies involve incorporating auxiliary classifiers at different layers of the model. To train models that perform well for both intermediate and final classifiers, various techniques have been proposed (Teerapittayanon et al., 2016; Hou et al., 2023a,c; Pu et al., 2024). Joint training methods simultaneously train all classifiers using weighted loss functions to combine losses from different classifiers (Zhou et al., 2020; Zhu et al., 2023; Zhang et al., 2022). Two-stage methods separate training of the backbone and intermediate classifiers, considering it unreasonable to merge the last layer with intermediate heads (Xin et al., 2020, 2021; Zhou et al., 2020). Addressing conflicts in training all classifiers, local training methods assign losses from each classifier to specific subsets of parameters (Rotem et al., 2023; Ji et al., 2023a). These adaptive inference techniques have shown promise in improving the efficiency of LLMs during inference. However, achieving a balance between the final classifier and intermediate classifiers remains a challenge, requiring intricate tradeoffs.

5.2 Parameter Efficient Fine Tuning

The computational challenges entailed in fine-tuning LLMs have prompted the exploration of parameter-efficient techniques (Xu et al., 2023). Adapter-based methods have emerged as notable approaches, involving the integration of small-scale adapter modules into LLM Transformer layers, with updates limited to these adapters during fine-tuning (Hu et al., 2023). Among these, the low-rank adapter (LoRA) has shown promise in fine-tuning LLMs by effectively reducing parameters and computational costs while preserving model performance (Hu et al., 2022). Extensive research, such as that by (Pfeiffer et al., 2020), has demonstrated that adapter-based methods can achieve performance on par with full-parameter fine-tuning across diverse NLP tasks. In an effort to further minimize the parameter count of adapter modules, (Davison, 2021) proposed Compacter, a lightweight adapter employing a combination of hypercomplex multiplication and parameter sharing. These advancements have significantly contributed to the efficient fine-tuning of LLMs, enabling their broader adoption in various applications (Liu et al., 2023). However, existing works have primarily focused on the training phase, overlooking the potential of these techniques in the inference phase.

6 Conclusion

In this work, we have introduced *LoRAExit*, a novel approach that addresses the limitations of existing early exit networks. By combining a low-rank adapter with the early exit network and implementing a superior-exit guided distillation method, we have demonstrated the effectiveness of *LoRAExit* in improving the efficiency of LLMs during inference. Our experimental results showcase a better speed-accuracy tradeoff compared to existing methods, positioning *LoRAExit* as a promising solution for deploying LLMs in resource-limited settings. This advancement opens new avenues for real-time, on-device language processing applications, offering practical benefits without degrading performance.

7 Limitations

While our *LoRAExit* framework has demonstrated effectiveness in improving the performance of early-exit models, it is important to acknowledge certain limitations that should be addressed in future research: (a) *LoRAExit* introduces additional parameters and computational overhead of the LoRA modules. Investigating novel execution methods that can alleviate this overhead is an avenue worth investigating in future work. (b) Our work primarily focuses on the performance of *LoRAExit* in classification tasks. To broaden the scope of our research, it is crucial to extend our investigations to encompass a wider range of tasks, such as sequence labeling, relation extraction, and text generation. (c) In our current work, we exclusively utilize discriminative LLMs as the backbone. Investigating the use of generative LLMs would be an intriguing direction for future research. Exploring the potential benefits and implications of incorporating generative LLMs (e.g., GPT) into our framework could yield valuable insights. In addition, the recently proposed Mamba model (Gu and Dao, 2023; Dao and Gu, 2024) tends to be a better backbone for *LoRAExit* since it can work without the memory-intensive KV cache.

8 Ethics Statement

Our proposed *LoRAExit* aims to enhance the efficiency of LLMs. By improving efficiency, our work has the potential to facilitate the deployment and utilization of pre-trained models on devices with limited computational capabilities, thereby increasing accessibility to state-of-the-art models for a broader range of users. We also anticipate that the adoption of our technology can contribute to reducing the carbon footprints associated with NLP-based applications. By optimizing the resource utilization and improving the efficiency of LLMs, we aim to minimize the environmental impact of these applications. Furthermore, it is worth noting that the GLUE datasets utilized in our experiments are widely employed in previous research. As such, we have taken precautions to ensure that our work does not introduce any new ethical concerns related to data collection, usage, or privacy.

Acknowledgment

We sincerely thank all the anonymous reviewers for their valuable comments. This work is supported in part by the Shanghai Pujiang Program (No. 23PJ1405100). This work was also partially supported by a grant from the STCSM International S&T Cooperation Project (23510713200). This work was also partially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project Number: T45-401/22-N). The corresponding authors are Xiaofeng Hou and Chao Li.

References

Nikita Balagansky and Daniil Gavrilov. 2023. Palbert: Teaching albert to ponder.

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, and Danilo Giampiccolo. 2006. The second PASCAL recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. 2009. The fifth pascal recognizing textual entailment challenge. In *In Proc Text Analysis Conference (TAC'09*.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma

Brunskill, Erik Brynjolfsson, S. Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen A. Creel, Jared Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas F. Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, O. Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Benjamin Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, J. F. Nyarko, Giray Ogut, Laurel J. Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Robert Reich, Hongyu Ren, Frieda Rong, Yusuf H. Roohani, Camilo Ruiz, Jack Ryan, Christopher R'e, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishna Parasuram Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei A. Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. 2021. On the opportunities and risks of foundation models. *ArXiv*, abs/2108.07258.

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

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

Lequn Chen, Zihao Ye, Yongji Wu, Danyang Zhuo, Luis Ceze, Arvind Krishnamurthy University of Washington, and Duke University. 2023. Punica: Multi-tenant lora serving. *ArXiv*, abs/2310.18547.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. PaLM: Scaling language modeling with pathways. arXiv:2204.02311.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Proceedings of the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*, MLCW'05, Berlin, Heidelberg.

Tri Dao and Albert Gu. 2024. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *International Conference on Machine Learning (ICML)*.

Joe Davison. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. In *Neural Information Processing Systems*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 1–9, Prague. Association for Computational Linguistics.

Jianping Gou, B. Yu, Stephen J. Maybank, and Dacheng Tao. 2020. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789 – 1819.

Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*.

Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. 2021. Dynamic neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44:7436–7456.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.

Xiaofeng Hou, Jiacheng Liu, Xuehan Tang, Chao Li, Jia Chen, Luhong Liang, Kwang-Ting Cheng, and Minyi Guo. 2023a. Architecting efficient multi-modal aiot systems. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*.

Xiaofeng Hou, Jiacheng Liu, Xuehan Tang, Chao Li, Kwang-Ting Cheng, Li Li, and Minyi Guo. 2023b. Mmexit: Enabling fast and efficient multi-modal dnn inference with adaptive network exits. In *Proceedings the 29th International European Conference on Parallel and Distributed Computing (Euro-par)*.

Xiaofeng Hou, Peng Tang, Chao Li, Jiacheng Liu, Cheng Xu, Kwang-Ting Cheng, and Minyi Guo. 2023c. Smg: A system-level modality gating facility for fast and energy-efficient multimodal computing. In *IEEE Real-Time Systems Symposium*.

Xiaofeng Hou, Xuehan Tang, Jiacheng Liu, Chao Li, Luhong Liang, and Kwang-ting Cheng. 2024. Wasp: Efficient power management enabling workload-aware, self-powered aiot devices. In *IEEE Transactions on Parallel and Distributed Systems*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. 2023. LLM-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Singapore. Association for Computational Linguistics.

Yixin Ji, Jikai Wang, Juntao Li, Qiang Chen, Wenliang Chen, and M. Zhang. 2023a. Early exit with disentangled representation and equiangular tight frame. In *Annual Meeting of the Association for Computational Linguistics*.

Yixin Ji, Jikai Wang, Juntao Li, Qiang Chen, Wenliang Chen, and Min Zhang. 2023b. Early exit with disentangled representation and equiangular tight frame. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 14128–14142. Association for Computational Linguistics.

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.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

Lei Li, Yankai Lin, Deli Chen, Shuhuai Ren, Peng Li, Jie Zhou, and Xu Sun. 2020. Cascadebert: Accelerating inference of pre-trained language models via calibrated complete models cascade. *arXiv preprint arXiv:2012.14682*.

Qidong Liu, Xian Wu, Xiangyu Zhao, Yuanshao Zhu, Derong Xu, Feng Tian, and Yefeng Zheng. 2023. Moelora: An moe-based parameter efficient fine-tuning method for multi-task medical applications. *ArXiv*, abs/2310.18339.

Jonathan Mamou, Oren Pereg, Moshe Wasserblat, and Roy Schwartz. 2023. TangoBERT: Reducing inference cost by using cascaded architecture. In *In Proc. EMC^2*.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Hongyi Jin, Tianqi Chen, and Zhihao Jia. 2023. Towards efficient generative large language model serving: A survey from algorithms to systems. *ArXiv*, abs/2312.15234.

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. 2019. Improved knowledge distillation via teacher assistant. In *AAAI Conference on Artificial Intelligence*.

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

Yifei Pu, Chi Wang, Xiaofeng Hou, Cheng Xu, Jiacheng Liu, Jing Wang, Minyi Guo, and Chao Li. 2024. M2sn : Adaptive and dynamic multi-modal shortcut network architecture for latency-aware applications. In *IEEE International Conference on Multimedia and Expo*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Daniel Rotem, Michael Hassid, Jonathan Mamou, and Roy Schwartz. 2023. Finding the sweet spot: Analysis and improvement of adaptive inference in low resource settings. In *Annual Meeting of the Association for Computational Linguistics*.

Ying Sheng, Shiyi Cao, Dacheng Li, Coleman Hooper, Nicholas Lee, Shuo Yang, Christopher Chou, Banghua Zhu, Lianmin Zheng, Kurt Keutzer, Joseph E. Gonzalez, and Ion Stoica. 2023. S-lora:

Serving thousands of concurrent lora adapters. *ArXiv*, abs/2311.03285.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. 2016. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2464–2469. IEEE.

Marcos Treviso, Tianchu Ji, Ji-Ung Lee, Betty van Aken, Qingqing Cao, Manuel R. Ciosici, Michael Hassid, Kenneth Heafield, Sara Hooker, Pedro H. Martins, André F. T. Martins, Peter Milder, Colin Raffel, Jessica Forde, Edwin Simpson, Noam Slonim, Jesse Dodge, Emma Stubell, Niranjan Balasubramanian, Leon Derczynski, Iryna Gurevych, and Roy Schwartz. 2023. Efficient methods for natural language processing: A survey. *TACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. of NeurIPS*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *CoRR*, abs/1804.07461.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

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*, pages 2246–2251, Online. Association for Computational Linguistics.

Ji Xin, Raphael Tang, Yaoliang Yu, and Jimmy J. Lin. 2021. Berxit: Early exiting for bert with better fine-tuning and extension to regression. In *Conference of the European Chapter of the Association for Computational Linguistics*.

Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. 2023. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *ArXiv*, abs/2312.12148.

Zhen Zhang, Wei Zhu, Jinfan Zhang, Peng Wang, Rize Jin, and Tae-Sun Chung. 2022. Pcee-bert: Accelerating bert inference via patient and confident early exiting. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 327–338.

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

Wei Zhu, Peifeng Wang, Yuan Ni, Guo Tong Xie, and Xiaoling Wang. 2023. Badge: Speeding up bert inference after deployment via block-wise bypasses and divergence-based early exiting. In *Annual Meeting of the Association for Computational Linguistics*.

A Appendix

A.1 Dataset information

The General Language Understanding Evaluation (GLUE) benchmark is a collection of nine natural language understanding (NLU) tasks. As shown in Table 8, it includes question answering (Rajpurkar et al., 2016), linguistic acceptability (Warstadt et al., 2018), sentiment analysis (Socher et al., 2013), text similarity (Cer et al., 2017), paraphrase detection (Dolan and Brockett, 2005), and natural language inference (NLI) (Dagan et al., 2006; Bar-Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009; Levesque et al., 2012; Williams et al., 2018). The diversity of the tasks makes GLUE very suitable for evaluating the generalization and robustness of NLU models. Numerous Early Exit-related studies, such as SWEET(Rotem et al., 2023), PABEE (Zhou et al., 2020), PALBERT (Balagansky and Gavrilov, 2023), and DREE (Ji et al., 2023b), utilize GLUE to evaluate their performance.

A.2 Experiment settings

A.2.1 Training

We train different backbones with different settings for a fair comparison with previous methods.

BERT-base. For BERT-base backbone, we set the training batch size to 32 and determine the optimal learning rate from $\{1e-5, 2e-5, 3e-5, 5e-5\}$ for PABEE, PALBERT, and DREE, adhering to the training configurations specified in (Ji et al., 2023b). For *LoRAExit*, classifiers are deployed at layers $(3, 6, 9, 12)$, and LoRA Adapters are integrated into the query and value modules of each transformer layer. For training the 12-layer model, we explore the best learning rate within $\{5e-5, 4e-4, 8e-4, 1e-3\}$, maintaining a batch size of 32. During the distillation of other layers, we adjust the temperature T in Equation 5 among $\{2, 10, 20\}$, the balance weight λ in Equation 4 within $\{1, 0.95, 0.2\}$, and the learning rate between $\{1e-3, 4e-4\}$. The utilization of LoRA Adapters for training necessitates a learning rate adjustment distinct from other methods.

ALBERT-base. For the ALBERT-base backbone, we limit the training set size for each task in GLUE to $6K$ for faster training. If the training set size of a task is smaller than $6K$, we use the entire dataset. Other settings for PABEE, PALBERT, DREE and *LoRAExit* are the same as the BERT-base backbone.

DeBERTa-base. For the DeBERTa-base backbone, we adopt a similar approach to dataset size limitation as with the ALBERT-base, capping each task in GLUE at 6,000 instances to streamline training. For training, we adjust the batch size to 16 and explore the optimal learning rate within $\{1e-5, 2e-5, 3e-5, 4e-5, 5e-5\}$ specifically for SWEET and MM methods. Classifiers are strategically positioned at layers $(1, 4, 6, 12)$ for SWEET, MM, and *LoRAExit*. Additionally, LoRA Adapters are integrated into the *pos_proj* and *pos_q_proj* modules of each layer to accommodate DeBERTa's unique transformer layer structure, which lacks traditional query and value modules. The remaining configuration settings for our method align with those used for the BERT-base backbone.

We take the original weight from the pre-trained model and use the randomly initialized LoRA weight to tune it. We train all models with enough epochs and choose the best checkpoint for inference.

A.2.2 Inference

For all experiments, we standardize the batch size for inference at 1. For PABEE and DREE, we set the patience parameter to 6, and for PALBERT, we adjust the early exit threshold to 0.5, in line with the guidelines provided in (Ji et al., 2023b). For SWEET, MM, and *LoRAExit*, we apply task-specific confidence thresholds to optimize performance. Inference operations utilize the float16 data type to enhance processing efficiency and are conducted on the NVIDIA 40G A100 platform to ensure high computational power. The inference speed, determined by the time taken to process the development set for all tasks, serves as our primary performance metric. We calculate the average inference time from 10 separate runs for each task to ensure reliability, with the exception of the QQP task. Given the extensive size of the QQP evaluation datasets, we limit our measurements to a single run to maintain feasibility.

A.3 Effectiveness of batching

To evaluate the effectiveness of the batched CUDA GEMM operator used in the first part of Equation 6, we recorded the inference times of three base models across varying batch sizes, from 1 to 8. As depicted in Figure 4, the batch size exhibits minimal impact on the inference times across all three models, indicating that our approach remains efficient even with a batch size of 4 per sample.

Corpus	Task	#Train	#Dev	#Test	#Label	Metrics
General Language Understanding Evaluation (GLUE)						
CoLA	Acceptability	8.5k	1k	1k	2	Matthews corr
SST	Sentiment	67k	872	1.8k	2	Accuracy
MNLI	NLI	393k	20k	20k	3	Accuracy
RTE	NLI	2.5k	276	3k	2	Accuracy
WNLI	NLI	634	71	146	2	Accuracy
QQP	Paraphrase	364k	40k	391k	2	Accuracy/F1
MRPC	Paraphrase	3.7k	408	1.7k	2	Accuracy/F1
QNLI	QA/NLI	108k	5.7k	5.7k	2	Accuracy
STS-B	Similarity	7k	1.5k	1.4k	1	Pearson/Spearman corr

Table 8: Summary information of the NLP application benchmarks.

Model	Parameter (M)	Layer Number	Exit Layer
BASE	100	12	(1, 4, 6, 12)
LARGE	350	24	(1, 6, 12, 24)
XLARGE	700	48	(1, 12, 24, 48)

Table 9: The configurations and exit layer settings for different sizes of DeBERTa model.

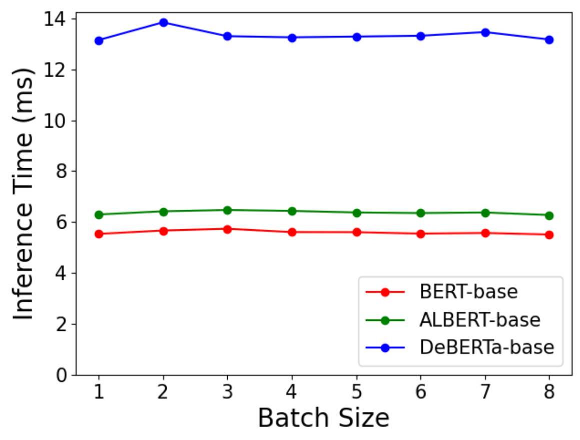


Figure 4: The inference time across different batch sizes of a single execution under three used models.

A.4 Memory Consumption Analysis

To evaluate the memory overhead of *LoRAExit* , we measure the memory consumption of SWEET and *LoRAExit* on four different sizes of the DeBERTa model during single-sample inference. As shown in Table 10, *LoRAExit* adds less than 8% memory overhead, and this overhead becomes relatively smaller with larger model sizes.

A.5 Error Analysis

Here we provide five failure cases from RTE Dataset on BERT-base model in Table 11. Specifically, in the first exit, the model provides the correct prediction, yet the confidence exceeds our predefined threshold. Conversely, at the second exit, the model delivers an incorrect prediction, with a confidence below the threshold. These discrepancies highlight instances of the model's "overthinking", leading to incorrect outcomes.

	SWEET	*LoRAExit*
deberta-base	280	303 (7.6%)
deberta-large	786	830 (5.3%)
deberta-v2-xlarge	1,717	1,809 (5.1%)
deberta-v2-xxlarge	3,023	3,158 (4.3%)

Table 10: The memory consumption (MB) of SWEET and LoRAExit on four different model sizes.

Sentence	Label	Exit 1 (label, confidence)	Exit 2 (label, confidence)
’sentence1’: ’Without a natural greenhouse effect, the temperature of the Earth would be about zero degrees F (-18C) instead of its present 57F (14C).’, ’sentence2’: ’Greenhouse effect changes global climate.’	0	(0, 0.0594177)	(1, 0.6367187)
’sentence1’: ’About 33.5 million people live in this massive conurbation. I would guess that 95% of the 5,000 officially foreign-capital firms in Japan are based in Tokyo.’, ’sentence2’: ’About 33.5 miilion people live in Tokyo.’	0	(0, 0.6640625)	(1, 0.0120086)
’sentence1’: ’Napkins, invitations and plain old paper cost more than they did a month ago.’, ’sentence2’: ’The cost of paper is rising.’	0	(0, 0.0000116)	(1, 0.1921386)
’sentence1’: ’NASA estimated, Monday, that it will cost 104 billion to return astronauts to the moon, by 2018, in a new rocket that combines the space shuttle with the capsule of an earlier NASA era.’, ’sentence2’: ’The new space vehicle design uses shuttle rocket parts and an Apollo-style capsule.’	1	(1, 0.01651)	(0, 0.0122222)
’sentence1’: ’California voters recall Gray Davis and elect Arnold Schwarzenegger as their governor.’, ’sentence2’: ’California voters dumped Gov. Gray Davis and replaced him with Arnold Schwarzenegger.’	0	(0, 0.0017108)	(1, 0.1357421)

Table 11: Five failure cases from RTE Dataset on BERT-base backbone.