RAEE: A Robust Retrieval-Augmented Early Exiting Framework for Efficient Inference

Lianming Huang[*]
City University of Hong Kong

Shangyu Wu[*†]
City University of Hong Kong, MBZUAI

Yufei Cui
MILA, McGill University

Ying Xiong
MBZUAI

Xue Liu
MILA, McGill University

Tei-Wei Kuo
National Taiwan University

Nan Guan
City University of Hong Kong

Chun Jason Xue
MBZUAI

Abstract

Deploying large language model inference remains challenging due to their high computational overhead. Early exiting optimizes model inference by adaptively reducing the number of inference layers. Existing methods typically train internal classifiers to determine whether to exit at intermediate layers. However, such classifier-based early exiting frameworks require significant effort to train the classifiers while can only achieve comparable performance at best. To address these limitations, this paper proposes RAEE, a robust **R**etrieval-**A**ugmented **E**arly **E**xiting framework for efficient inference. First, this paper demonstrates that the early exiting problem can be modeled as a distribution prediction problem, where the distribution is approximated using similar data's exiting information. Then, this paper details the process of collecting exiting information to build the retrieval database. Finally, based on the pre-built retrieval database, RAEE leverages the retrieved similar data's exiting information to guide the backbone model to exit at the layer, which is predicted by the approximated distribution. Experimental results demonstrate that the proposed RAEE can significantly accelerate inference. More importantly, RAEE can also achieve a robust zero-shot performance on 8 downstream tasks.

1 Introduction

Large language models have been widely used in various application scenarios due to their excellent performance (Thoppilan et al., 2022; Touvron et al., 2023; Scao et al., 2022). However, deploying large language models on resource-constrained devices is still challenging due to the high computational overheads of performing model inference (Dao et al., 2022; Liu et al., 2023). Model pruning, as an advanced technique, provides a new direction for efficient inference (Valicenti et al., 2023; Ma et al., 2023). It selectively removes less important weights or connections from the neural network to reduce complexity and computational requirements without significantly degrading performance. One popular model pruning method is the early exiting technique, which speeds up inference by adaptively reducing the number of inference layers.

Most early exiting frameworks (Liu et al., 2020; Zhu, 2021; Fan et al., 2024) leverage classifiers to predict the exiting layer and then stop the inference at the predicted exiting layer. Those classifier-

[*]Authors contributed equally to this research.
[†]Corresponding author.

4th NeurIPS Efficient Natural Language and Speech Processing Workshop (ENLSP-IV 2024).

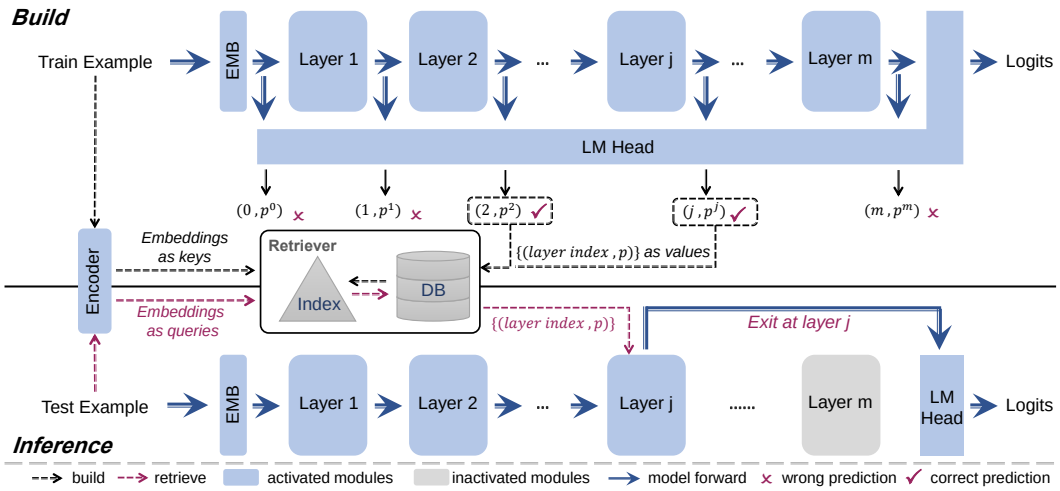


Figure 1: The overview of retrieval-augmented early exiting framework.

based early exiting frameworks can be categorized into three types according to the training strategies. One type is the training-based early exiting (Zhu, 2021; Zhou et al., 2020; Zhu et al., 2023), which requires training the classifiers along with the inference model. These methods introduce significant training and fine-tuning overheads, particularly when applied to large language models. Another branch of classifier-based early exiting can be concluded as semi-training-based early exiting (Fan et al., 2024). The backbone models in this type of early exiting framework would not be updated, and only classifiers would be fitted to predict the exiting layer. These methods may not capture the patterns between inputs and exiting layers well, requiring significant human effort in feature engineering. The last one is training-free early exiting (Sun et al., 2022), which requires no parameter updates and uses heuristics to determine the exiting layer. These methods lack the generalization ability in predicting the exiting layer and often fail to achieve optimal or sub-optimal solutions. Moreover, most existing early exiting frameworks sacrifice the model performance for acceleration (Fan et al., 2024; Sun et al., 2022; Schuster et al., 2022; Bae et al., 2023).

To address the above limitations, this paper first shows that the exiting layer predictions can be solved by predicting from an exiting distribution. Then, this paper presents the observations that similar data's exiting information can be used to approximate the exiting distribution. Based on these observations, this paper proposes the RAEE, a robust retrieval-augmented early exiting framework for efficient inference. RAEE collects exiting information from training data and builds the indexing to retrieve similar data's exiting information. During the inference, RAEE predicts the exiting layer based on the top-k nearest neighbors' exiting information and stops the model forwarding at the predicted exiting layer.

We conduct comprehensive experiments to evaluate the proposed RAEE and various comparison methods on 8 downstream tasks. Experimental results demonstrate that RAEE can accelerate the model inference while achieving robust model performance. Codes are available at [3].

The main contributions of this paper are:

- We model the early exiting problem as a distribution prediction problem and demonstrate that the exiting distribution can be approximated by the exiting information of similar data.
- We propose a robust retrieval-augmented early exiting framework, named RAEE, which leverages the external database to guide the early exiting.
- Experimental results show that the proposed RAEE can not only accelerate the model inference but also achieve a robust performance.

[3]https://anonymous.4open.science/r/RAEE-D724

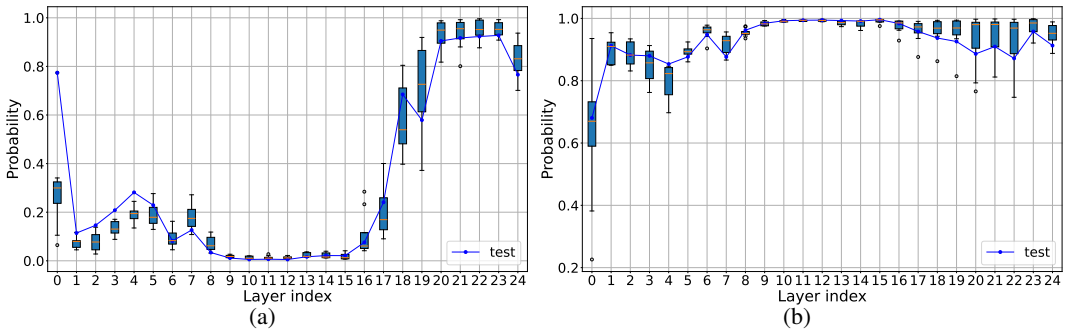


Figure 2: The probability of different exiting layers on SST-2 test data and corresponding top-8 nearest neighbor SST-2 training data.

2 Motivations

In this section, we demonstrate why using retrieval-based techniques is a simple yet effective way to augment the early exiting framework during the inference stage.

Problem Statement. Formally, early exiting can be defined as follows: Given a backbone model $\mathcal{M}$ with m layers and an input x, The early exiting framework aims to design an exiting function or classifier $l = f(x)$ to determine whether to exit at the layer l. The final prediction y is then transformed from the intermediate output states h_l of the l-th layer. And the final prediction probability can be formulated as,

$$P(y \mid x) = P(y \mid h_{f(x)}). \tag{1}$$

, where $f(x)$ is trained or built on the downstream tasks' training data $\mathcal{D}$.

Limitations of Existing Early Exiting. Existing early exiting frameworks can be categorized into training-based, semi-training-based, and training-free exiting methods. For training-based exiting, most works jointly optimize the exiting classifiers and the backbone model (Zhu et al., 2023, 2021; Zhu, 2021). These works fine-tune the pre-trained backbone model with exiting mechanisms over downstream tasks' training data and update both the backbone model and classifier parameters. Although these approaches could adapt the model with the early exiting framework and accelerate the inference while maintaining performance, the fine-tuning overhead is still a non-negligible amount, especially when fine-tuning modern large language models.

Instead of training both classifiers and the backbone model, other works (Fan et al., 2024) only fit the classifiers based on features extracted from the backbone model, with parameters in the backbone model fixed. These works can significantly reduce the training overhead of early exiting framework, as they aim to address the early exiting problem using efficient traditional machine learning approaches for classifier fitting, e.g., techniques for fitting support vector machine. However, such methods that only train classifiers may not effectively capture the patterns between inputs and exiting layers. The reasons may lie in that 1) the classifiers only focus on distinguishing the extract features at each layer, rather than recognize the patterns between input embeddings and exiting layers; 2) the feature extraction requires human effort to design, where classifiers only indirectly learn where to exit.

Training-free early exiting frameworks refer to computing the exiting layer with heuristics, e.g., hashing functions (Sun et al., 2022). These techniques basically introduce no or few training overhead and are easy to set up. Although these works can significantly accelerate the building process as well as the classifying process, they lack generalization in exiting layer predictions. The predictions made by these heuristics may also be sensitive to different inputs, leading to unstable performance of the early exiting framework.

Motivations of Retrieval-Augmented Early Exiting. To address the above limitations, this paper aims to leverage retrieval-based techniques to guide the early exiting, which requires no parameters to update and has a generalization capability. Existing retrieval databases basically use clustering and product quantization to build the retrieval indexing over millions of embeddings, which can perform efficient approximate nearest neighbor searching. The building process of the indexing is not

resource-constrained, which can run on either GPUs or CPUs. Besides, since the retrieval database stores the original data, every retrieval in the retrieval-augmented early exiting can be regarded as a generalization over several semantically similar data, while those original data may be used to train classifiers or build hashing functions in other early exiting frameworks. The retrieval-augmented techniques also exhibit strong adaptability to new data. By simply adding new data to the retrieval database, the index can retrieve up-to-date information.

To further demonstrate the efficacy of retrieval-augmented early exiting, this paper conducts some analysis experiments in Figure 2 to show the probability of different exiting layers on two SST-2 test data and corresponding top-8 nearest neighbor SST-2 training data. The probability in Figure 2 is calculated as the normalized logits of the answer label token. Note that it is impossible to obtain the probability of each exiting layer during the real inference as no labels are provided. The blue line with dots in Figure 2 shows the probability of correct predictions exiting at each layer. For example, in Figure 2 (a), when exiting at layer 18, the probability of correct predictions is about 0.68, while exiting at layer 9 only has a probability of 0.01. The box plot in Figure 2 presents the probability statistics of the top-8 nearest neighbor training data, indicating the probability of exiting at each layer. For instance, when exiting at layer 18, the maximum exit probability is approximately 0.81, the minimum is around 0.39, and the box plot indicates that half of the neighbors exhibit a probability larger than 0.54.

The blue line in Figure 2 represents the target distribution for the classifier to learn. The experimental results in Figure 2 (a) and (b) reveal that the exiting probabilities exhibit a similar pattern to those of the nearest neighbors. Additionally, the exiting layer varies across different inputs. Consequently, the experimental results lead to two key conclusions:

1. The exiting probability can be approximated by the probability of the top-k nearest neighbors.
2. Different inputs exhibit distinct probabilities for exiting layers.

These observations motivate us to propose a retrieval-augmented early exiting framework.

3 Methodology

In this section, this paper describes the proposed retrieval-augmented early exiting framework in detail. First, this paper introduces the details of how to build the retrieval database for the early exiting. Then, this paper presents the retrieval-augmented early exiting baseline called **RAEE**.

3.1 Building the Retrieval Database for Early Exiting

The critical factor in constructing the retrieval database is selecting appropriate keys and values. To avoid introducing too much retrieving overheads, this paper only retrieves once at the beginning of the backbone model. Consider the training data $\mathcal{D} = \{(x_1^{train}, y_1^{train}), \ldots, (x_{|\mathcal{D}|}^{train}, y_{|\mathcal{D}|}^{train})\}$ and a backbone model $\mathcal{M}$ with m layers $\{\mathcal{L}_1, \ldots, \mathcal{L}_m\}$. In this context, as shown in the top part of Figure 1, the keys $\mathcal{K}$ are input embeddings of training data, which can be obtained from an extra encoder model $\mathcal{E}$, such as BERT (Devlin et al., 2019), or the outputs of embedding layers in the backbone model $\mathcal{M}_{emb}$,

$$\mathcal{K} = \{e_i\}_{i=1}^{|\mathcal{D}|} = \{\mathcal{E}(x_i^{train})\}_{i=1}^{|\mathcal{D}|}. \tag{2}$$

For the values, this paper collects a set of possible exiting layers l_i and corresponding probabilities p_i for each embedding e_i, i.e., $v_i = \{(l_i^j, p_i^j)\}_{j=1}^{m_i}$, where m_i indicates the number of possible exiting layers for the embedding e_i. The layer l chosen as the exiting layer is determined by whether the outputs of this layer h_l can be used to make the right predictions $\hat{y}$ compared to the training labels y^{train}. Then, the values $\mathcal{V}$ are all sets of possible exiting layers,

$$\mathcal{V} = \{v_i\}_{i=1}^{|D|} = \left\{\{(l_i^j, p_i^j)\}_{j=1}^{m_i}\right\}_{i=1}^{|D|}. \tag{3}$$

We follow the same dataset splitting used in the LM-BFF (Gao et al., 2021), the collecting process requires no parameters to update, only model inference is performed. After collecting keys and values for the retrieval databases, this paper uses state-of-the-art approximate nearest neighbor search indexing, such as FAISS (Johnson et al., 2019), and efficient key-value stores to build the retrieval database. More details can be found in Algorithm B.1 of Appendix B.

3.2 RAEE

In this section, this paper then presents a retrieval-augmented early exiting framework named RAEE to optimize the model inference. RAEE regards the exiting layer as a random variable z, taking values in the set of $\{1, \ldots, m\}$, where m is the total number of layers in the backbone model $\mathcal{M}$. The probability mass function $P(z = l)$ represents the probability of the case that the backbone model exits at the layer l. With the gold label, we can observe that the random variable z follows an unknown discrete distribution F. Then, this paper shows how to leverage the retrieval database to approximate the distribution F.

Given an input x, RAEE first retrieves top-k nearest neighbors $\{v_1, \ldots, v_k\}$, where each neighbor v_i has m_i possible exiting layers. Naturally, we can approximate the distribution F by estimating the probability function $P(z = l)$,

$$P(z = l \mid x) = \sum_{i=1}^{k} P(v_i \mid x) \cdot \sum_{j}^{m_i} \mathbb{1}\left(any(l_i^j = l \;\&\&\; p_i^j \geq \tau)\right) \cdot p_i^j \tag{4}$$

, where $\mathbb{1}$ is the indicator function that returns 1 if the condition is true and 0 otherwise, $any(\cdot)$ is the function that returns true if one condition is true and false otherwise, τ is the threshold for filtering the layers with extremely low probability, the inner loop only count once since there is at most one possible exiting layer of neighbor i that is equalt to l. Since different neighbors should have different contributions to the probability function $P(z = l)$, RAEE uses the reciprocal of the scaled distance between each neighbor and the query to estimate the contribution,

$$P(v_i \mid x) = \frac{\min\left(\{distance(v_j, x)\}_{j=1}^{k}\right)}{distance(v_i, x)} \tag{5}$$

Then, RAEE designs a function $f(x)$ to determine the exiting layer, which selects the layer that maximizes the probability function $P(z = l)$,

$$f(x) = \arg\max_{l} P(z = l \mid x) \tag{6}$$

Notably, when there are multiple exiting layers with the same maximal probability, RAEE chooses the earliest exiting layer.

The bottom part of Figure 1 shows the inference workflow of RAEE. Specifically, RAEE first simultaneously feeds the inputs into both the backbone model for the label predictions and the same encoder used in the building process for the query embeddings. Then, the retriever in RAEE retrieves the top-k nearest neighbors in the retrieval databases based on the query embeddings. After obtaining all possible exiting layers of k nearest neighbors, RAEE computes the exiting layers based on the Equations 4-6. Finally, RAEE stops the forwarding at the calculated exiting layer, and passes the intermediate outputs of the exiting layer to the final prediction layer, e.g., LM Head in language models, to obtain the final predictions (Equation 1). More details can be found in Algorithm B.2 of Appendix B.

4 Experiments

In this section, this paper first introduces the dataset and the experimental setting. Then, this paper presents the main results of 8 downstream tasks. This paper also conducts an ablation study and analysis of RAEE to show the impact of these factors on model performance.

4.1 Dataset and Experimental Setup

Datasets We conduct comprehensive experiments across 8 downstream tasks from GLUE benchmark (Wang et al., 2019). These tasks cover sentiment analysis, opinion polarity analysis, grammatical judgment, natural language inference, paraphrasing, etc.

Experimental Settings The proposed RAEE was implemented using the PyTorch framework and Transformer. We evaluated RoBERTa, ElasticBERT (Liu et al., 2022), D-BERT (Sanh et al., 2019), D-RoBERTa (Sanh et al., 2019), HashEE (Sun et al., 2022), AdaInfer (Fan et al., 2024), and RAEE-RoBERTa on one NVIDIA Quadro RTX5000 GPU with 16GB GPU memory, while others on

Table 1: Zero-shot performance of different methods on 8 downstream tasks. Methods with an asterisk superscript '*' are evaluated on NVIDIA A100 GPU. The results in bold are the best results on the task. 'D-BERT' and 'D-RoBERTa' refer to DistilBERT and DistilRoBERTa, respectively.

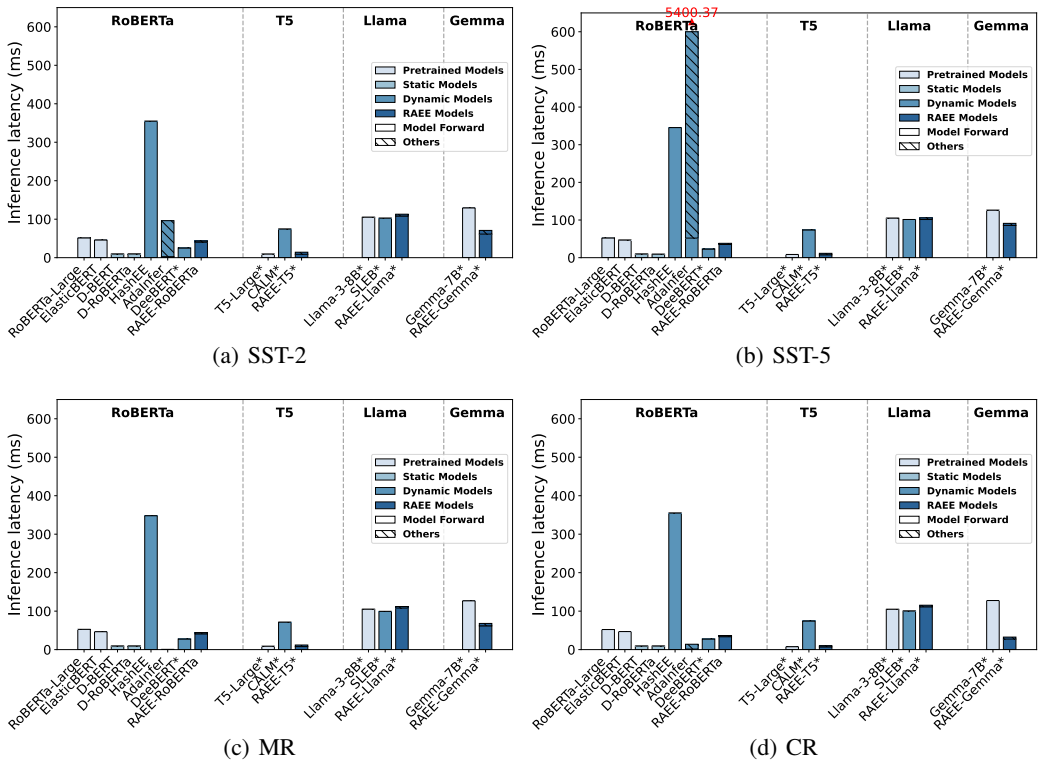
Methods	**SST-2**	**SST-5**	**MR**	**CR**	**MPQA**	**SUBJ**	**TREC**	**CoLA**	**Avg**
Pretrained Models									
RoBERTa-Large	83.60	**34.98**	80.80	79.55	67.60	51.45	32.40	2.03	54.05
ElasticBERT	51.15	27.78	50.10	50.00	47.05	50.00	18.60	6.02	37.59
T5-Large*	49.31	23.12	50.40	50.90	45.40	52.75	27.60	-4.64	36.86
Llama-3-8B*	62.84	26.06	59.65	72.90	51.75	52.80	8.40	0.00	41.80
Gemma-7B*	49.08	28.64	50.05	50.10	50.00	48.05	18.00	-0.79	36.64
Static Models									
D-BERT	71.90	25.61	67.65	77.10	64.85	70.35	28.40	-0.70	50.65
D-RoBERTa	81.77	28.37	77.95	**84.20**	69.50	54.35	32.00	-1.18	53.37
Dynamic Models									
HashEE	52.75	18.64	48.95	52.50	55.85	45.35	23.80	-3.86	36.75
AdaInfer	50.92	**34.98**	50.45	49.60	60.90	50.65	32.40	-1.62	41.04
DeeBERT*	52.29	18.05	50.60	50.00	75.95	80.85	16.20	0.00	42.99
CALM*	51.72	23.17	49.25	50.55	49.80	49.90	18.00	0.00	36.55
SLEB*	51.38	16.61	50.05	50.65	50.10	47.55	25.40	0.00	36.47
RAEE-RoBERTa	**84.06**	33.44	**81.75**	67.90	**79.25**	83.60	**60.60**	**15.55**	**63.27**
RAEE-T5*	53.21	27.19	50.60	51.55	55.90	49.90	39.80	12.55	42.59
RAEE-Llama*	75.57	32.04	67.90	72.75	76.10	**89.95**	46.00	7.84	58.52
RAEE-Gemma*	73.17	32.35	67.35	56.85	75.80	89.90	28.40	11.43	54.41

one NVIDIA A100 GPU with 80GB GPU memory. The experiments were conducted in the zero-shot setting. For dynamic models with classifiers, we only train the classifiers on downstream tasks. The evaluation metric is accuracy, except for CoLA, which is measured by the Matthew correlation coefficient. We evaluate RAEE on various backbone models, such as RoBERTa-large, T5-Large (Raffel et al., 2020), Llama-3-8B (Dubey et al., 2024), and Gemma-7B (Team et al., 2024). The number of retrieved nearest neighbors is set to 12 in this paper.

To validate the effectiveness, we compared RAEE with three types of methods. **Pretrained Models:** 1) RoBERTa-Large (Gao et al., 2021), a state-of-the-art encoder model, where the prompt-based version is used; 2) ElasticBERT (Liu et al., 2022), a pre-trained multi-exit transformer model, where the large version is used in this paper; 3) T5-Large (Raffel et al., 2020), a versatile transformer-based model for various NLP tasks; 4) Llama-3-8B (Dubey et al., 2024), a pre-trained model with strength in specific language scenarios; 5) Gemma-7B (Team et al., 2024), a model with potential for outstanding performance in specific settings. **Static Models:** 1) DistilBERT (Sanh et al., 2019), a distilled version of BERT-base model; 2) DistilRoBERTa (Sanh et al., 2019), a distilled version of the RoBERTa-base model. **Dynamic Models:** 1) HashEE (Sun et al., 2022), a hash-based early exiting approach with ElasticBERT-large as its backbone model; 2) AdaInfer (Fan et al., 2024), an SVM-based early exiting method with our reproduced version on RoBERTa-large; 3) DeeBERT (Xin et al., 2020), a classical entropy-thresholding-based early exiting method with RoBERTa-Large as its backbone model; 4) CALM (Schuster et al., 2022), a classical entropy-thresholding-based early exiting method with T5-Large (Raffel et al., 2020) as its backbone model; 5) SLEB (Song et al., 2024), a method that tackles the limitation of early exiting methods by eliminating redundant transformer blocks with Llama-3-8b (Dubey et al., 2024) as its backbone model. The templates are listed in the Appendix A. More details about the experimental setup can be found in Appendix E.

4.2 Main Results

Table 1 presents the main results, comparing the performance of the RAEE method against three other method types across eight downstream tasks. From the experimental results, although the Pretrained



(a) SST-2 (b) SST-5

(c) MR (d) CR

Figure 3: Inference latency of different methods on 4 downstream tasks.

Models on some tasks, such as CR, can achieve good performance, we value the performance of the model on the overall tasks, and the proposed RAEE can achieve the best zero-shot performance on average. Compared to the backbone model, such as Roberta-Large, T5-Large, Llama-3-8B, and Gemma-7B without early exiting, RAEE outperforms them on most tasks. Besides, RAEE also significantly outperforms existing early exiting frameworks on all tasks. The above results demonstrate the robustness of the proposed RAEE by consistently achieving strong performance across various tasks and backbone models.

Figure 3 shows the inference latency of RAEE and comparisons. Due to space limits, we only show the inference latency on the first four tasks, and more results can be found in Appendix C. Experimental results show that the proposed RAEE can consistently accelerate the model inference over various backbone models. Although static models can achieve a fast inference speed due to the small model architecture, those models require great efforts to fine-tune or distill from backbone models. The exceptionally long inference latency of current dynamic models is primarily due to the internal classifiers' poor performance in zero-shot scenarios. The inadequate performance of these classifiers leads to late exits, which in turn causes high inference overheads. In addition, HashEE and AdaInfer show particularly poor inference latency. We used the original code for HashEE from its paper's repository, while AdaInfer, being closed-source, was reproduced based on its published paper version. Future work will focus on optimizing and standardizing these frameworks to improve efficiency.

4.3 Reasons for Significant Performance Improvement

The main results demonstrate that the proposed RAEE can significantly outperform backbone models, which is not an easy-understanding and intuitive phenomenon compared to previous early exiting methods. The reason of such phenomenon lies in that the retrieval database in RAEE also acts as an error corrector, which contains the exiting information of examples that are correctly predicted by intermediate layers, but backbone models without early exiting fail to predict.

Table 2: Model performance and inference latency of different methods with RoBERTa-Large.

Models	SST-2	SST-5	MR	CR	MPQA	SubJ	TREC	CoLA	Avg
Performance ↑									
RoBERTa-Large	83.60	**34.98**	80.80	**79.55**	67.60	51.45	32.40	2.03	54.05
RAEE w/o	**85.32**	34.75	81.05	71.55	75.80	63.90	42.60	-2.10	56.61
RAEE	85.21	33.44	**81.75**	67.70	**78.90**	**83.80**	**62.60**	**13.75**	**63.39**
Latency (ms) ↓									
RoBERTa-Large	49.40	49.63	49.63	49.39	49.02	49.44	49.65	49.11	49.41
RAEE w/o	48.18	52.07	47.87	41.73	45.97	42.66	43.50	46.28	46.03
RAEE	**47.27**	**36.10**	**45.82**	**38.49**	**41.84**	**33.76**	**34.47**	**31.74**	**38.69**

Table 3: The cost of building the retrieval database.

Model	SST-2	SST-5	MR	CR	MPQA	Subj	TREC	CoLA	Avg
Building Time (second)	430.81	514.46	545.90	151.97	531.17	495.42	349.59	526.55	443.23
# Entries	6920	8544	8662	1775	8606	8000	5452	8551	7063.75
Index Size (MB)	3.5	3.9	3.9	2.0	3.9	3.8	3.2	3.9	3.5
Database Size (MB)	2.6	2.1	3.1	0.8	2.6	2.7	1.0	2.7	2.2

To better support the above claims, we also conducted an analysis experiment using the retrieval database, which only contained exit information based on examples that backbone models correctly predicted without early exiting. As shown in Table 2, RAEE w/o refers to the method built on only correctly predicted examples. As expected, RAEE w/o achieves comparable performance to baselines but accelerates the inference process. This is because the test data that is correctly predicted by RAEE w/o can also be correctly predicted by backbone models. However, due to a lack of exiting information on examples where backbone models fail to predict, RAEE w/o also fails to predict on the test data where backbone models fail. Therefore, when providing the exiting information based on examples where backbone models fail to predict but intermediate outputs succeed in predicting, RAEE can make correct predictions and exit earlier. It is noted that the slight discrepancies observed in the performance of RAEE here, as compared to the main results, can be attributed to random factors in the construction of the faiss database. We will solve this in the future by fixing the database.

4.4 Cost of Building the Retrieval Database

We collect the statistics of building the retrieval database in Table 3, such as database size, index size, and time cost for the retrieval database for RoBERTa-Large. The average of building the retrieval database is less than 8 minutes on RTX 5000, an acceptable overhead compared to the time cost of fine-tuning. The index size and database size are quite small, which can be ignored compared to the backbone model size.

4.5 Ablation Study of Top-*k*

Table 4 illustrates the impact of varying the number of retrievals on the distribution approximation. As k increases, the proposed RAEE-RoBERTa improves the overall performance from 61.22 to 63.27. This suggests that more retrieved exiting information can help enhance the approximation performance. However, when k beyond 12, the overall performance degrades from 63.27 to 62.83. The reasons may lie in that providing the exiting information of the retrievals that are not quite related to the query would introduce noise, thus misleading the final predictions. This also implies that only a few exiting information is enough to approximate the exiting distribution, and the time cost of the retrieving process can be saved.

4.6 Ablation Study on the Retrieval Database Size

Table 5 shows the performance of RAEE-RoBERTa with different sizes of retrieval databases. The size of the retrieval databases implies how similar embeddings would be retrieved, thus impacting

Table 4: The impact of retrieval number k on the distribution approximation.

k	SST-2	SST-5	MR	CR	MPQA	SUBJ	TREC	CoLA	Avg
2	78.21	32.94	77.35	68.50	78.25	80.05	60.20	14.26	61.22
4	81.08	33.48	79.40	67.90	79.45	82.95	**61.20**	14.78	62.53
8	83.60	**34.12**	81.00	68.05	**79.75**	**83.65**	**61.20**	12.28	62.96
12	**84.06**	33.44	81.75	67.90	79.25	83.60	60.60	**15.55**	**63.27**
16	83.72	32.71	**82.05**	69.10	78.70	83.35	**61.20**	15.24	63.26
20	83.14	33.53	81.85	**69.70**	78.45	83.05	61.00	11.94	62.83

Table 5: The impact of retrieval database size on the distribution approximation. The percentage refers to the amount of training data that is used to build the retrieval database.

Database Size	SST-2	SST-5	MR	CR	MPQA	SUBJ	TREC	CoLA	Avg
20%	83.14	32.35	81.40	67.80	75.55	77.85	57.20	11.47	60.85
50%	82.45	32.31	81.30	65.35	76.65	82.65	58.20	**16.83**	61.97
100%	**84.06**	**33.44**	**81.75**	**67.90**	**79.25**	**83.60**	**60.60**	15.55	**63.27**

the confidence of the provided exiting information. As the database size increases, the performance of RAEE-RoBERTa increases significantly from 60.85 to 63.27 on average. This demonstrates that collecting more data can improve the generalization of RAEE, thus approximating the exiting distribution more accurately.

5 Related Work

5.1 Early Exiting Framework

Model inference with early exit has been a popular pruning method to reduce both computation and memory overhead on text classification or generation tasks. Most current works (Bae et al., 2023; Kong et al., 2022; Ji et al., 2023; Wolczyk et al., 2021; Hooper et al., 2023) introduce classifiers in each layer to determine whether the inference should continue. Different from those works, our method does not require training the classifier. Our method predicts exit layers using a pre-built database, resulting in better generalization.

5.2 Retrieval-based Augmentations

Retrieval-based augmentations (Li et al., 2022; Wang et al., 2023; Xiong et al., 2023; Cui et al., 2023; Wu et al., 2024a,b) have been widely used in various natural language processing (NLP) tasks and achieved remarkable performance. Current works mostly leverage external knowledge databases to augment generator models on various text-generation tasks. Those works focus on improving the model's generation quality, while our work aims to use the retrieval knowledge to accelerate the model's inference. Additionally, other works have improved model generation efficiency using external retrieval databases. These works are out of the scope of the research problems in this paper.

6 Conclusion

This paper models the early exiting problem as a distribution approximation problem and observes that similar data's exiting information can be used to approximate. Then, this paper proposes a retrieval-augmented early exiting framework named RAEE. Experimental results show that RAEE can accelerate the model inference while significantly improving the model performance. For future work, exploring the performance of applying the retrieval-augmented early exiting framework in fine-tuning scenarios is worthwhile.

References

Sangmin Bae, Jongwoo Ko, Hwanjun Song, and Se-Young Yun. Fast and robust early-exiting framework for autoregressive language models with synchronized parallel decoding. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 5910–5924, 2023.

Yufei Cui, Ziquan Liu, Yixin Chen, Yuchen Lu, Xinyue Yu, Xue (Steve) Liu, Tei-Wei Kuo, Miguel Rodrigues, Chun Jason Xue, and Antoni B. Chan. Retrieval-augmented multiple instance learning. In *Proceedings of the Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023 (NeurIPS)*, 2023.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Proceedings of the Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina 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)*, pp. 4171–4186. Association for Computational Linguistics, 2019.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, and et al. The llama 3 herd of models, 2024. URL `https://arxiv.org/abs/2407.21783`.

Siqi Fan, Xin Jiang, Xiang Li, Xuying Meng, Peng Han, Shuo Shang, Aixin Sun, Yequan Wang, and Zhongyuan Wang. Not all layers of llms are necessary during inference. *CoRR*, abs/2403.02181, 2024.

Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL/IJCNLP)*, pp. 3816–3830. Association for Computational Linguistics, 2021.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Hasan Genc, Kurt Keutzer, Amir Gholami, and Yakun Sophia Shao. SPEED: speculative pipelined execution for efficient decoding. *CoRR*, abs/2310.12072, 2023.

Yixin Ji, Jikai Wang, Juntao Li, Qiang Chen, Wenliang Chen, and Min Zhang. Early exit with disentangled representation and equiangular tight frame. In *Findings of the Association for Computational Linguistics (ACL)*, pp. 14128–14142, 2023.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

Jun Kong, Jin Wang, Liang-Chih Yu, and Xuejie Zhang. Accelerating inference for pretrained language models by unified multi-perspective early exiting. In *Proceedings of the 29th International Conference on Computational Linguistics (COLING)*, pp. 4677–4686, 2022.

Zonglin Li, Ruiqi Guo, and Sanjiv Kumar. Decoupled context processing for context augmented language modeling. In *Proceedings of the Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

Weijie Liu, Peng Zhou, Zhiruo Wang, Zhe Zhao, Haotang Deng, and Qi Ju. Fastbert: a self-distilling BERT with adaptive inference time. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 6035–6044, 2020.

Xiangyang Liu, Tianxiang Sun, Junliang He, Jiawen Wu, Lingling Wu, Xinyu Zhang, Hao Jiang, Zhao Cao, Xuanjing Huang, and Xipeng Qiu. Towards efficient NLP: A standard evaluation and A strong baseline. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL): Human Language Technologies*, pp. 3288–3303, 2022.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Ré, and Beidi Chen. Deja vu: Contextual sparsity for efficient llms at inference time. In *Proceedings of the 2023 International Conference on Machine Learning (ICML)*, pp. 22137–22176, 2023.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Proceedings of the Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020. URL https://jmlr.org/papers/v21/20-074.html.

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

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilic, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurençon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris Emezue, Christopher Klamm, Colin Leong, Daniel van Strien, David Ifeoluwa Adelani, and et al. BLOOM: A 176b-parameter open-access multilingual language model. *CoRR*, abs/2211.05100, 2022.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/6fac9e316a4ae75ea244ddcef1982c71-Abstract-Conference.html.

Jiwon Song, Kyungseok Oh, Taesu Kim, Hyungjun Kim, Yulhwa Kim, and Jae-Joon Kim. SLEB: streamlining llms through redundancy verification and elimination of transformer blocks. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=fuX4hyLPm0.

Tianxiang Sun, Xiangyang Liu, Wei Zhu, Zhichao Geng, Lingling Wu, Yilong He, Yuan Ni, Guotong Xie, Xuanjing Huang, and Xipeng Qiu. A simple hash-based early exiting approach for language understanding and generation. In *Findings of the Association for Computational Linguistics (ACL)*, pp. 2409–2421, 2022.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, and et al. Gemma: Open models based on gemini research and technology, 2024. URL https://arxiv.org/abs/2403.08295.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, YaGuang Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Yanqi Zhou, Chung-Ching Chang, Igor Krivokon, Will Rusch, Marc Pickett, Kathleen S. Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Soraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Diaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravi Rajakumar, Alena Butryna, Matthew Lamm, Viktoriya Kuzmina, Joe Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Agüera y Arcas, Claire Cui, Marian Croak, Ed H. Chi, and Quoc Le. Lamda: Language models for dialog applications. *CoRR*, abs/2201.08239, 2022.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023.

Tim Valicenti, Justice Vidal, and Ritik Patnaik. Mini-gpts: Efficient large language models through contextual pruning. *CoRR*, abs/2312.12682, 2023.

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

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory. In *Proceedings of the Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

Maciej Wolczyk, Bartosz Wójcik, Klaudia Balazy, Igor T. Podolak, Jacek Tabor, Marek Smieja, and Tomasz Trzcinski. Zero time waste: Recycling predictions in early exit neural networks. In *Proceedings of the Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems (NeurIPS)*, pp. 2516–2528, 2021.

Shangyu Wu, Ying Xiong, Yufei Cui, Xue Liu, Buzhou Tang, Tei-Wei Kuo, and Chun Jason Xue. Improving natural language understanding with computation-efficient retrieval representation fusion. *CoRR*, abs/2401.02993, 2024a.

Shangyu Wu, Ying Xiong, Yufei Cui, Haolun Wu, Can Chen, Ye Yuan, Lianming Huang, Xue Liu, Tei-Wei Kuo, Nan Guan, and Chun Jason Xue. Retrieval-augmented generation for natural language processing: A survey, 2024b. URL `https://arxiv.org/abs/2407.13193`.

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

Ying Xiong, Xin Yang, Linjing Liu, Ka-Chun Wong, Qingcai Chen, Yang Xiang, and Buzhou Tang. EARA: improving biomedical semantic textual similarity with entity-aligned attention and retrieval augmentation. In *Findings of the Association for Computational Linguistics (EMNLP)*, pp. 8760–8771, 2023.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian J. McAuley, Ke Xu, and Furu Wei. BERT loses patience: Fast and robust inference with early exit. In *Proceedings of the Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

Wei Zhu. Leebert: Learned early exit for BERT with cross-level optimization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL/IJCNLP)*, pp. 2968–2980, 2021.

Wei Zhu, Xiaoling Wang, Yuan Ni, and Guotong Xie. GAML-BERT: improving BERT early exiting by gradient aligned mutual learning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3033–3044, 2021.

Wei Zhu, Peng Wang, Yuan Ni, Guotong Xie, and Xiaoling Wang. BADGE: speeding up BERT inference after deployment via block-wise bypasses and divergence-based early exiting. In *Proceedings of the The 61st Annual Meeting of the Association for Computational Linguistics (ACL): Industry Track*, pp. 500–509, 2023.

A Templates on All Tasks

Table 6 provides an overview of the manual templates and selected label words used for each dataset in this paper. These templates and label words were created following LM-BFF (Gao et al., 2021).

Table 6: Templates and label words used in this paper.

Task	Prompts	Label word
SST-2	`[CLS]` x It was `[MASK]`. `[SEP]`	"0":"terrible", "1":"great"
SST-5	`[CLS]` x It was `[MASK]`. `[SEP]`	"0":"terrible","1": "bad", "2": "okay","3": "good","4": "great"
MR	`[CLS]` x It was `[MASK]`. `[SEP]`	"0":"terrible", "1":"great"
CR	`[CLS]` x It was `[MASK]`. `[SEP]`	"0":"terrible", "1":"great"
MPQA	`[CLS]` x It was `[MASK]`. `[SEP]`	"0":"terrible", "1":"great"
SUBJ	`[CLS]` x This is `[MASK]`. `[SEP]`	"0":"subjective", "1":"objective"
TREC	`[CLS]` `[MASK]` x `[SEP]`	"0":"Description","1":"Entity","2":"Expression", "3":"Human","4":"Location","5":"Number"
CoLA	`[CLS]` x It was `[MASK]`. `[SEP]`	"0":"incorrect", "1":"correct"

B Detailed Algorithms of RAEE

Algorithm B.1 Collect the keys and values for building the retrieval database.

Input: Training data $\mathcal{D} = \{(x_1^{train}, y_1^{train}), \ldots, (x_{|\mathcal{D}|}^{train}, y_{|\mathcal{D}|}^{train})\}$, backbone model $\mathcal{M}$ with m layers $\{\mathcal{L}_1, \ldots, \mathcal{L}_m\}$, encoder $\mathcal{E}$ (None value means no encoder is provided).
Output: Keys $\mathcal{K}$ and values $\mathcal{V}$.

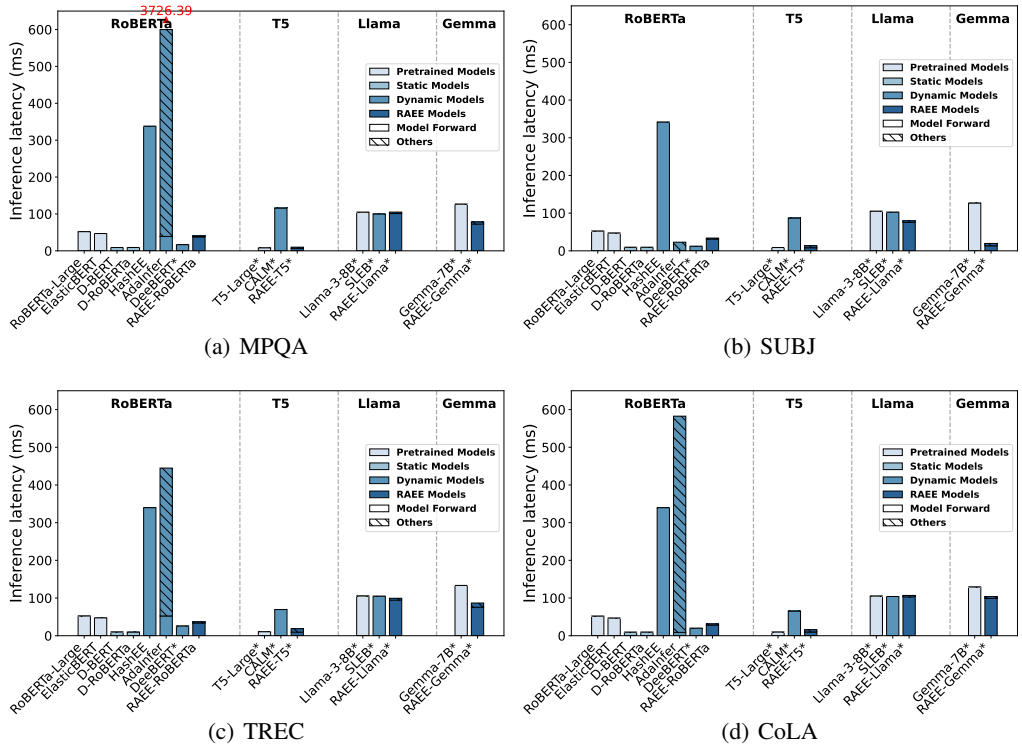
1: $\mathcal{K} = [], \mathcal{V} = []$
2: **for** $i = 1, \ldots, |\mathcal{D}|$ **do**
3: $v_i = []$;
4: $h_0 = \mathcal{M}_{emb}(x_i^{train})$;
5: **for** j=1, . . . , m **do**
6: $h_j = \mathcal{L}_j(h_{j-1})$; /* Compute the intermediate outputs of the layer j */
7: $logits = \mathcal{M}_{lm\_head}(h_j)$; /* Predict from the layer j */
8: $\hat{y} = \arg\max logits$;
9: $p_i^j = \max\{\text{softmax}(logits)\}$;
10: **if** $\hat{y}$ is equal to y_i^{train} **then**
11: Add (j, p_i^j) into v_i; /* Store the possible exiting layer */
12: **end if**
13: **end for**
14: Add v_i into $\mathcal{V}$;
15: **if** $\mathcal{E}$ is None **then**
16: Add h_0 into $\mathcal{K}$; /* Store the embeddings of backbone model when no encoder model */
17: **end if**
18: **end for**
19: **if** $\mathcal{E}$ is not None **then**
20: Add all $\mathcal{E}(x_i^{train})$ into $\mathcal{K}$; /* Store the embeddings of encoder */
21: **end if**
22: **return** $\mathcal{K}, \mathcal{V}$;

Algorithm B.1 collects keys and values for building the retrieval database. For each sample in training data $\mathcal{D}$, the backbone model $\mathcal{M}$ is traversed layer by layer to compute the hidden state h_j and corresponding $logits$. If a prediction $\hat{y}$ at a certain layer j matches the sample's true label y_i^{train}, the exiting information, including the layer j and the probability p_i^j, is added to the sample's value list (Line 2-12). When the encoder $\mathcal{E}$ is unavailable (Line 16), RAEE utilizes the hidden states from the backbone model $\mathcal{M}$ as embeddings for indexing. The specific layer from which the hidden states are extracted is treated as a hyperparameter that the user can define.

Algorithm B.2 Model inference with the synchronized retrieval-augmented early exiting.

Input: Input x, backbone model $\mathcal{M}$ with m layers $\{\mathcal{L}_1, \ldots, \mathcal{L}_m\}$, encoder $\mathcal{E}$, indexing $\mathcal{I}$, top-k, the exiting layer determination function $f(\cdot)$.
Output: Final prediction $\hat{y}$.

1: $h_0 = \mathcal{M}_{emb}(x)$;
2: **if** $\mathcal{E}$ is not None **then**
3: $\quad e_{query} = \mathcal{E}(x)$; /* Encode the inputs when the encoder is available */
4: **else**
5: $\quad e_{query} = h_0$; /* Use the embeddings of backbone model */
6: **end if**
7: $\{(v_i, dis_i)\}_{i=1}^{k} = \mathcal{I}(e_{query}, k)$; /* Retrieve the possible exiting layers */
8: $l = f\left(\{(v_1, dis_1), \ldots, (v_k, dis_k)\}\right)$; /* Obtain the exiting layer */
9: **for** $i = 1, \ldots, l$ **do**
10: $\quad h_i = \mathcal{L}_i(h_{i-1})$; /* Perform model inference with early exiting */
11: **end for**
12: $logits = \mathcal{M}_{lm\_head}(h_l)$; /* Predict based on the layer h_l outputs */
13: $\hat{y} = \arg\max logits$;
14: **return** $\hat{y}$;



(a) MPQA (b) SUBJ

(c) TREC (d) CoLA

Figure 4: Inference latency of different methods on 4 downstream tasks (cond).

Algorithm B.2 performs model inference with retrieval-augmented early exiting. When the encoder $\mathcal{E}$ is unavailable (Line 5), RAEE utilizes the hidden states from the backbone model $\mathcal{M}$ as embeddings for querying. The specific layer from which the hidden states are extracted is treated as a hyperparameter that the user can define.

C Inference Latency

Figure 4 shows the inference latency of different methods of the last four downstream tasks. Experimental results demonstrate the consistent performance of the proposed RAEE compared to Figure 3.

Table 7: Exiting layers of different methods with RoBERTa-Large, T5-Large, Llama-3-8B, and Gemma-7B on 8 downstream tasks. The sum of the number of layers in the encoder and the decoder counts the number of layers for T5-large (Raffel et al., 2020). Due to the strong code coupling, it is hard to collect the exiting layer information of CALM(Schuster et al., 2022). Methods with an asterisk superscript '*' are evaluated on NVIDIA A100 GPU with 80G GPU memory, while others are evaluated on NVIDIA RTX 5000.

Model	**SST-2** ↓	**SST-5** ↓	**MR** ↓	**CR** ↓	**MPQA** ↓	**Subj** ↓	**TREC** ↓	**CoLA** ↓	**Avg** ↓
Pretrained Models									
RoBERTa-Large*	24.00	24.00	24.00	24.00	24.00	24.00	24.00	24.00	24.00
T5-Large*	48.00	48.00	48.00	48.00	48.00	48.00	48.00	48.00	48.00
Llama-3-8B*	32.00	32.00	32.00	32.00	32.00	32.00	32.00	32.00	32.00
Gemma-7B*	28.00	28.00	28.00	28.00	28.00	28.00	28.00	28.00	28.00
Dynamic Models									
AdaInfer	1.00	24.00	0.00	0.47	18.00	0.08	24.00	4.00	8.94
DeeBERT*	22.95	24.00	23.33	8.98	15.90	10.36	24.00	18.31	18.48
SLEB*	28.00	28.00	28.00	28.00	28.00	28.00	28.00	28.00	28.00
RAEE-RoBERTa	18.61	15.87	18.69	15.38	17.32	14.00	15.25	12.70	15.98
RAEE-T5*	22.05	18.44	21.74	26.91	17.77	18.71	27.34	18.72	21.46
RAEE-Llama*	29.86	27.08	29.72	31.16	28.82	21.45	24.24	30.97	27.91
RAEE-Gemma*	10.97	17.72	11.86	3.27	14.75	0.52	12.97	20.08	11.52

D Exiting Layers

Table 7 compares the average exiting layers of the RAEE method against two other method types across eight downstream tasks. Experimental results show that the RAEE method can exit earlier than the two other method types, thus reducing computational overhead during model inference for better model efficiency. This result also aligns with the expectations in the motivation example. This suggests that the RAEE method can accurately approximate the gold exiting layer distribution by using the retrieval database. Although AdaInfer exits earlier than the RAEE method, it exhibits quite poor performance, as shown in Table 1. The reason may be that during the zero-shot inference scenario, the collected features can only provide limited information for the SVM, thus resulting in unstable prediction performance.

E Implementation Details

This section lists the implementation details.

- For DeeBERT(Xin et al., 2020), we use RoBERTa-Large as its backbone model. Since DeeBERT(Xin et al., 2020) is a classical entropy-thresholding-based early-exit method, it requires first fine-tuning the backbone model on the downstream task and then updating all but the last off-ramp, for a fair comparison, we only update the off-ramp in DeeBERT on each downstream task. We also use RoBERTa-large as the backbone model and train all off-ramps for 50 epochs (much larger than the default setting of 10 epochs). Other experimental settings for DeeBERT(Xin et al., 2020) remain as default.

- For CALM (Schuster et al., 2022), we use T5-Large (Raffel et al., 2020) as its backbone model. CALM (Schuster et al., 2022) is also a classical entropy-thresholding-based early-exit method, and we evaluate it under the zero-shot setting.

- For SLEB(Song et al., 2024), we use Llama-3-8b (Dubey et al., 2024) as its backbone model. SLEB(Song et al., 2024) tackles the limitation of early exit methods by eliminating redundant transformer blocks. Since the proposed RAEE exits at 27.91 layers, for a fair comparison, we also set the hyper-parameter `num_remove_blocks` of SLEB(Song et al., 2024) as 4 for comparable efficiency.

F Comparisons to Fine-tuning-based Methods

We have evaluated CALM(Schuster et al., 2022) and FREE (Bae et al., 2023) on the given 8 downstream tasks in table 8 and table 9. We finetune them with the default setting on each task for one epoch. Experiments are conducted on 1 NVIDIA A100 GPU. Since the experimental results of CALM(Schuster et al., 2022) and FREE(Bae et al., 2023) under the zero-shot setting are the same, we only report the zero-shot results of CALM(Schuster et al., 2022). Although the fine-tuned CALM(Schuster et al., 2022) and FREE(Bae et al., 2023) can achieve much better performance, they require large computational resources and time for the fine-tuning process, which is NOT the main research goal of this paper. Under the zero-shot setting, the proposed RAEE can still outperform CALM(Schuster et al., 2022) or FREE(Bae et al., 2023).

Table 8: Model performance of different methods with T5-Large under different settings on 8 downstream tasks.

Model	**SST-2** ↑	**SST-5** ↑	**MR** ↑	**CR** ↑	**MPQA** ↑	**Subj** ↑	**TREC** ↑	**CoLA** ↑	**Avg** ↑
Zero-Shot									
RAEE-T5	53.21	27.19	50.60	51.55	55.90	49.90	39.80	12.55	42.59
CALM	51.72	23.17	49.25	50.55	49.80	49.90	18.00	0.00	36.55
Fine-Tuning									
CALM	88.76	28.96	86.90	49.25	85.05	82.85	71.40	-2.61	61.32
FREE	87.96	42.81	85.30	7.40	85.80	78.25	64.80	0.04	56.55

Table 9: Inference latency(ms) of different methods with T5-Large under different settings on 8 downstream tasks.

Model	**SST-2** ↓	**SST-5** ↓	**MR** ↓	**CR** ↓	**MPQA** ↓	**Subj** ↓	**TREC** ↓	**CoLA** ↓	**Avg** ↓
Zero-Shot									
RAEE-T5	14.21	11.56	11.98	10.33	9.93	13.60	18.60	15.93	13.27
CALM	74.29	73.87	71.16	74.57	116.15	87.27	69.49	65.60	79.05
Fine-Tuning									
CALM	99.72	100.39	101.45	90.98	102.97	108.79	84.58	87.14	97.00
FREE	85.69	95.79	95.24	95.21	94.87	90.91	92.11	89.45	92.41

G Retrieved Examples of RAEE

We show two examples from the SST-2 task and their retrieved top-k data samples. As shown in Table 10 and Table 11, the retrieved samples are semantically similar to the query sentence, demonstrating the proposed RAEE's efficacy.

H Limitations

The limitations of RAEE may lie in the following aspects. **Building Overheads.** Although building the retrieval database is an offline process that can be completed in a few hours, there is still a trade-off between the generalization capability and the building overheads. Analysis experiments have demonstrated that more data can significantly improve the RAEE's generalization capability. **Retrieving Quality.** Various types of indexing can be chosen for building the retrieval databases. Different retrievers have different retrieving qualities. The indexing used in RAEE may not be the optimal choice across different tasks. Addressing these limitations is not the main focus of this paper, while future optimizations on those topics can be combined with the proposed RAEE.

Table 10: Examples of data and corresponding retrieved data.

Query/Top-K	Sentence	Label
Query	although laced with humor and a few fanciful touches, the film is a refreshingly serious look at young women.	1
Top-1	the film is hard to dismiss – moody, thoughtful, and lit by flashes of mordant humor.	1
Top-2	the movie enters a realm where few non-porn films venture, and comes across as darkly funny, energetic, and surprisingly gentle.	1
Top-3	the movie, despite its rough edges and a tendency to sag in certain places, is wry and engrossing.	1
Top-4	metaphors abound, but it is easy to take this film at face value and enjoy its slightly humorous and tender story.	1
Top-5	it may not be particularly innovative, but the film's crisp, unaffected style and air of gentle longing make it unexpectedly rewarding.	1
Top-6	it has its faults, but it is a kind, unapologetic, sweetheart of a movie, and mandy moore leaves a positive impression.	1
Top-7	although frailty fits into a classic genre, in its script and execution it is a remarkably original work.	1
Top-8	unlike lots of hollywood fluff, this has layered, well-developed characters and some surprises.	1
Top-9	as broad and cartoonish as the screenplay is, there is an accuracy of observation in the work of the director, frank novak, that keeps the film grounded in an undeniable social realism.	1
Top-10	though its rather routine script is loaded with familiar situations, the movie has a cinematic fluidity and sense of intelligence that makes it work more than it probably should.	1
Top-11	it tends to remind one of a really solid woody allen film, with its excellent use of new york locales and sharp writing.	1
Top-12	though a touch too arthouse 101 in its poetic symbolism, heaven proves to be a good match of the sensibilities of two directors.	1

I Future Work

In Algorithm B.2, which outlines the procedure for model inference using retrieval-augmented early exiting, several key improvements could be pursued to optimize efficiency in future work. The inference process (Lines 9-11) and the retrieval process (Lines 2-8) hold the potential for parallel execution. By implementing separate threads for each of these processes, one focusing on inference and the other on retrieval, efficiency could be substantially improved. Furthermore, during the inference process with asynchronized RAEE, the calculated intermediate results would be collected for early exiting, which not only speeds up the inference process but also optimizes resource usage by potentially minimizing unnecessary computations.

Table 11: Examples of data and corresponding retrieved data (Cond).

Query/Top-K	Sentence	Label
Query	... a boring parade of talking heads and technical gibberish that will do little to advance the linux cause.	0
Top-1	a vile, incoherent mess... a scummy ripoff of david cronenberg's brilliant 'videodrome.	0
Top-2	completely creatively stillborn and executed in a manner that i'm not sure could be a single iota worse... a soulless hunk of exploitative garbage.	0
Top-3	contrived, maudlin and cliche-ridden... if this sappy script was the best the contest received, those rejected must have been astronomically bad.	0
Top-4	could as easily have been called ' under siege 3: in alcatraz '... a cinematic corpse that never springs to life.	0
Top-5	little more than a stylish exercise in revisionism whose point...is no doubt true, but serves as a rather thin moral to such a knowing fable.	0
Top-6	a thoroughly awful movie – dumb, narratively chaotic, visually sloppy...a weird amalgam of 'the thing' and a geriatric scream.	0
Top-7	on a cutting room floor somewhere lies...footage that might have made no such thing a trenchant, ironic cultural satire instead of a frustrating misfire.	0
Top-8	...while certainly clever in spots, this too-long, spoofy update of shakespeare's macbeth does n't sustain a high enough level of invention.	0
Top-9	worthless, from its pseudo-rock-video opening to the idiocy of its last frames.	0
Top-10	comes across as a relic from a bygone era, and its convolutions... feel silly rather than plausible.	0
Top-11	a tired, unnecessary retread...a stale copy of a picture that was n't all that great to begin with.	0
Top-12	(less a movie than) an appalling, odoriferous thing...so rotten in almost every single facet of production that you'll want to crawl up your own in embarrassment.	0