Accelerating Large Language Model Inference with Self-Supervised Early Exits

Florian Valade

Accelerating Large Language Model Inference with Self-Supervised Early Exits

Florian Valade
Gustave Eiffel University
Fujitsu

July 29, 2024

Abstract

This paper presents a novel technique for accelerating inference in large, pre-trained language models (LLMs) by introducing early exits during inference. The computational demands of these models, used across a wide range of applications, can be substantial. By capitalizing on the inherent variability in token complexity, our approach enables selective acceleration of the inference process. Specifically, we propose the integration of early exit "heads" atop existing transformer layers, which facilitate conditional terminations based on a confidence metric. These heads are trained in a self-supervised manner using the model's own predictions as training data, thereby eliminating the need for additional annotated data. The confidence metric, established using a calibration set, ensures a desired level of accuracy while enabling early termination when confidence exceeds a predetermined threshold. Notably, our method preserves the original accuracy and reduces computational time on certain tasks, leveraging the existing knowledge of pre-trained LLMs without requiring extensive retraining. This lightweight, modular modification has the potential to greatly enhance the practical usability of LLMs, particularly in applications like real-time language processing in resource-constrained environments.

1 Introduction

Large language models (LLMs) have become central to advancing capabilities in natural language processing (NLP), delivering remarkable performance across a range of tasks. The trend towards scaling up these models correlates strongly with improved performance, understanding, and generality. However, the computational cost associated with these larger models is substantial, often necessitating the use of powerful server infrastructure Samsi et al. [2023]. This not only limits local usability but also raises significant privacy concerns and requires considerable investment to scale in response to user demand. Solutions exist to reduce the computational demands of these models, but they often impact the model's performance by reducing its accuracy Zhu et al. [2023].

Despite their effectiveness, these models often operate inefficiently. The nature of language itself contributes to this inefficiency; namely, not all tokens generated during the inference process contribute equally to the overall meaning or require the same level of computational resources. Some tokens are inherently simpler and can be predicted with high confidence early in the computation process, while others, contributing more significantly to the context or meaning, may require deeper processing.

In response to these challenges, our objective is to develop a method that can be easily integrated into existing pre-trained models to enhance their inference speed without necessitating extensive retraining or modification. Our proposed solution focuses on the strategic placement of early exit "heads" Scardapane et al. [2020], Teerapittayanon et al. [2016] within the transformer layers of an LLM. These heads are designed to terminate the inference process prematurely when a certain confidence threshold is met, based on the complexity and predictability of the token being processed.

Our contribution is twofold:

1. We introduce a lightweight, modular enhancement for pre-trained LLMs that significantly accelerates inference by incorporating early exits in the model architecture. These exits are strategically placed and are trained in a self-supervised manner, leveraging the outputs from the model itself as training targets.

2. We d evise a novel method for determining when to apply these early exits during inference. This involves generating a calibration set and establishing confidence thresholds in a self-supervised manner. These

thresholds allow the model to make informed decisions about whether to continue processing based on the predicted confidence level of the output at each step.

This method not only improves the efficiency of LLMs but also maintains the integrity and accuracy of the model's outputs, making it particularly useful for applications requiring real-time processing capabilities in resource-constrained environments.

2 Related Work

The Transformer architecture, introduced by Vaswani et al. [2023], has revolutionized the field of natural language processing (NLP). Large Language Models (LLMs) have become increasingly popular due to their ability to process and generate human-like text with remarkable accuracy. As LLMs continued to grow in size and complexity, they began to expand beyond NLP, demonstrating their potential for applications in other domains such as computer vision Dosovitskiy et al. [2020], speech recognition Radford et al. [2022] and other. The pursuit of larger and more accurate models has led to the development of numerous LLM architectures Devlin et al. [2019], Thoppilan et al. [2022], Brown et al. [2020], Radford et al., Touvron et al. [2023], Chowdhery et al. [2022], Zhang et al. [2022]. However, this growth in size and complexity has come at a cost: modern LLMs are often computationally expensive to run, making them less accessible for widespread adoption and hindering their potential for real-world applications.

To mitigate the computational costs associated with large language models, several techniques have been employed to improve their efficiency. Some of these approaches include quantization Sun et al. [2020], Shen et al. [2019], Yao et al. [2022], pruning Fan et al. [2019] or knowledge distillation Bai et al. [2021], Sun et al. [2019]. Early exit, has been explored in various machine learning domains as a means to reduce computational costs while maintaining acceptable accuracy. In the context of neural networks, early exit refers to the ability of a model to terminate computation at intermediate layers, allowing for faster inference and reduced computational overhead. Research in this field mainly goes along two axes: designing efficient early exit networks Teerapittayanon et al. [2017], Huang et al. [2018] and improving the exit rule to find the best trade off between accuracy and computation Liu et al. [2020], Xin et al. [2020], Zhou et al. [2020], Valade et al. [2024]. Most

of these methods rely on diverse confidence metric to determine a threshold of early exiting at inference time.

Our contribution lies in the versatility and adaptability of our early exit strategy, which can be integrated into any recent LLM architecture. Unlike many existing methods, our approach leverages the generative capacity of pretrained models by training our early exit mechanism without the need for additional data. We set our thresholds for exiting using a calibration set, which allows us to finely control the balance between computational efficiency and accuracy. This method ensures that our model can dynamically adjust to varying complexities in data while maintaining a strict budget on computational resources, making it highly suitable for real-world applications where processing speed and model responsiveness are crucial.

Our proposal shares some features with *conformal predictions* Vovk et al. [1999]. There, the goal is to build, given a trained model, a calibration set and a conformity measure, a prediction interval (or set) valid with high probability for a new instance. Our calibration of the thresholds for exiting borrows some idea from conformal prediction has the huge difference that we tackle a self-supervised problem. Moreover, our goal is not to build a set of outputs but only the next token for a sequence.

3 Methodology

This section details our methodology for integrating and utilizing early exits within large language models (LLMs) to enhance computational efficiency during inference. The approach is designed to be generalizable and, while we demonstrate its application using the Phi-2 Model, it is applicable to any multi-layered transformer model. This adaptability ensures that our methodology can be leveraged across a broad spectrum of modern LLMs, enhancing their usability without requiring significant modifications to their underlying architectures.

3.1 Definitions and Preliminaries

In this section, we establish the foundational definitions and notations used throughout our study on early exit strategies in LLMs.

Dataset: Let $\mathcal{T} = \{0, 1, \ldots, V-1\}$ represent the token space, where V is the size of the vocabulary and each element corresponds to a unique token

within the model's vocabulary.

We define two specific datasets used in our experiments: the calibration dataset and the training dataset. Let $\mathcal{D}_t = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{N_t}\}$ and $\mathcal{D}_c = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{N_c}\}$ be the calibration and training datasets, respectively. Each sample $\mathbf{x}_i$ in these datasets is a sequence of tokens: $\mathbf{x}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,L_i})$, where L_i denotes the length of the i^{th} sample and $x_{i,j} \in \mathcal{T}$ represents the j^{th} token in the i^{th} sample.

Classifiers: Define $f_\theta : \mathcal{T}^* \to \Delta^V$ as the LLM, where θ denotes the parameters of the main model, $\mathcal{T}^*$ signifies the set of all possible token sequences, and Δ^V represents the V-dimensional probability simplex, indicating the distribution over the vocabulary. The token that is assigned with the highest probability is then the next token in the sequence. The output of the main model given an input token list $\mathbf{x}$ is denoted as $\mathbf{p}_\theta = f_\theta(\mathbf{x})$ to simplify notation.

We incorporate K early exit heads into the LLM to facilitate efficient inference. Each head $h_k : \mathcal{T}^* \to \Delta^V$ for $k \in \{1, 2, ..., K\}$ is a classifier that outputs a probability distribution over the vocabulary. The output of the k^{th} head given an input token list $\mathbf{x}$ is denoted by $\mathbf{p}_k$, where $\mathbf{p}_k$ is a vector in Δ^V.

We define the confidence metric $c : \Delta^V \to \mathbb{R}$, which assesses the reliability of predictions by converting a probability vector $\mathbf{p}_k$ into a scalar confidence value. This metric plays a critical role in determining whether the processing at an early exit head is sufficient or if further computation is necessary. As an example, a simple max function can be used as confidence metric. With these definitions established, we proceed to describe the implementation details of each component, their interaction within the system, and the methodologies used for training and inference.

3.2 Implementation and Training

To enhance the inference efficiency of large language models, we incorporate early exit "heads" into a pre-existing model, in this instance, the Phi-2 model Gunasekar et al. [2023]. These heads are implemented at regular intervals along the network. Structurally, each head is a simple multi-layer perceptron (MLP), identical to the final classification head of the model. each of these head takes as input hidden features from a transformer block inside the model.

For the implementation of these heads, we experimented with two initialization strategies: initializing the heads from scratch and copying the final

classification head in order to fine-tune it. The difference between the two are analyzed in section 4.

Training batch	**Calibration batch**
This is an example script:	A cat is
My name is	In Python, a list is
I am a	In C, we can define a function
Welcome to	The capital of France is
what is a	The derivative of x^2 is

Table 1: Starting batch used to have a diverse set of tokens generated.

The training of these heads is conducted in a self-supervised manner. We construct batches from various starting points (see Table 1) in conversations to ensure coverage of diverse linguistic structures and token complexities. For each input, we infer the next token based on the model f_θ and we compute a custom loss that compares the output from each early exit head to the output from the main model. The loss function for each head is defined as follows:

$$loss = \sum_{k=1}^{K} loss_k = \sum_{k=1}^{K} \left\{ (1-\lambda) \cdot CE\left(\mathbf{p}_k, \mathbf{p}_\theta\right) - \lambda \cdot Entropy\left(\mathbf{p}_k\right) \right\}, \quad (1)$$

where K represents the total number of classifiers[1], $\mathbf{p}_k$ denotes the output of the k_{th} classifier with $\mathbf{p}_\theta$ being the output of the underlying model, CE is the cross-entropy loss, computed as:

$$CE(\mathbf{x}, \mathbf{y}) = -\sum_i y_i \log(softmax(x_i)).$$

Moreover, the entropy penalty, defined as

$$Entropy(p) = -\sum_i p_i \log(p_i).$$

is added to encourage the model to be less confident (closer to the uniform probability distribution) in the case when the certainty of the classification is low.

In the loss function (1), the tuning parameter λ is a penalty weight that balances the cross-entropy loss with the entropy penalty, promoting

[1]The total number of classifiers is just the number of early exit heads within the model.

confidence in the early exits where appropriate, and allowing for uncertainty where necessary. This mechanism ensures that the early exits do not overly commit to predictions without sufficient confidence, and it plays a crucial role in the calibration of thresholds for early termination in subsequent stages. We study the impact of λ in section 4.

3.3 Calibration and Inference

After training the early exit heads, the next crucial step involves calibrating and using these heads during model inference. This process is divided into two main stages: calibration of the confidence thresholds and the application of these thresholds during inference.

3.3.1 Calibration of Confidence Thresholds

The calibration starts with the selection of an appropriate confidence metric. Our experimental setup tested several metrics: the maximum probability value, the entropy of the probability distribution, and the difference between the two highest probability scores from the classification vector. The latter, known as the "breaking ties" metric, was selected due to its superior empirical performance. Mathematically, the breaking ties metric for a probability vector $\mathbf{p}$ is defined as:

$$c(\mathbf{p}) = p_{(1)} - p_{(2)}$$

where $p_{(1)}$ and $p_{(2)}$ are the highest and second-highest entries in $\mathbf{p}$, respectively.

To calibrate the confidence thresholds, we use a calibration batch provided in Table 1. This batch is composed of diverse starting points designed to cover different linguistic contexts.

For each starting point, the model generates tokens, and the selected confidence metric is applied to each output from the early exits. We record both the metric value and a Boolean indicating whether the early exit's prediction matches the prediction of the underlying model f_θ. After generating a sufficient number of tokens, we obtain a dataset of metric values paired with correctness indicators.

Upon completing the calibration process, we derive two distinct vectors for each head k in the model. Let

$$\mathbf{c}_k = (c(h_k(\mathbf{x}_1)), c(h_k(\mathbf{x}_2)), \ldots, c(h_k(\mathbf{x}_{N_c}))) \in \mathbb{R}^{N_c} \tag{2}$$

be the vector containing the metric values obtained at head k during the calibration phase, where $\mathbf{x}_i$ is an input from the calibration dataset $\mathcal{D}_c$ and $c(h_k(\mathbf{x}_i))$ represents the confidence metric applied to the output of the classifier h_k for the input x_i.

Correspondingly, let $\mathbf{t}_k$ be a binary vector of length N_c where each element corresponds to the correctness of the prediction associated with the respective element in $\mathbf{c}_k$. Specifically, $t_{k,i}$ is 1 if the i^{th} prediction at head k is the same as the i^{th} prediction associated to of the underlying model $f_\theta(\mathbf{x}_i)$, otherwise 0. More formally, we can write:

$$t_{k,i} = \mathbb{1}_{\{\arg\max h_k(\mathbf{x}_i) = \arg\max f_\theta(\mathbf{x}_i)\}} = \begin{cases} 1 & \text{if } \arg\max(h_k(\mathbf{x}_i)) = \arg\max(f_\theta(\mathbf{x}_i)) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where the maxima are taken over the coordinates of the probability score vectors associated to the token sequence $\mathbf{x}_i$.

The calibration set is subsequently used to establish a confidence threshold for each early exit head. We begin by sorting the metric values obtained during calibration. Then, let $\epsilon \in [0, 1]$ be a confidence level selected to control the adherence to the model's output, specifying the minimum proportion of correct decisions required above the threshold. This parameter is crucial as it determines the trade-off between accuracy and efficiency in the inference process. The threshold for each head k is set to the lowest metric value where the percentage of correct predictions is at least ϵ, ensuring that decisions made past this threshold maintain a high level of precision. Formally, the threshold for each head is defined as follows.

Recall that $\mathbf{c}_k$ and $\mathbf{t}_k$ are respectively the vector of scores and the vector correctness in prediction given by (2) and (3). We can sort these two vectors *w.r.t.* the ascending order according to the values of $\mathbf{c}_k$. That is, we can rearrange terms $\mathbf{c}_k = (c_{k,1}, c_{k,2}, \ldots, c_{k,N_c})$ and $\mathbf{t}_k = (t_{k,1}, t_{k,2}, \ldots, t_{k,N_c})$ in a such way that $c_{k,1} \leq c_{k,2} \leq \ldots \leq c_{k,N_c}$. Moreover, let $\hat{j}$ be the first index such that:

$$\frac{\sum_{i=\hat{j}}^{N_c} t_{k,i}}{N_c - \hat{j} + 1} \geq \epsilon.$$

Then, let the threshold τ_k be defined as:

$$\tau_k = c_{k,\hat{j}}.$$

In other words, τ_k is the value of the confidence metric at the index $\hat{j}$, where

$\hat{j}$ is the smallest index such that the proportion of correct prediction in the remaining samples is at least ϵ.

3.3.2 Inference Process

After establishing the confidence thresholds for each early exit head through the calibration process, the model is then ready to utilize these thresholds during the inference phase to efficiently process new inputs.

During inference, each input $\mathbf{x}$ is sequentially processed through the model's layers, with the possibility of early termination at any of the early exit heads h_k. For each head $k \in \{1, 2, \ldots, K\}$, we compute:

$$\mathbf{p}_k = h_k(\mathbf{x}),$$
$$c_k = c(\mathbf{p}_k).$$

We then return $\mathbf{p}_k$ as output from the model if:

$$c_k \geq \tau_k.$$

If no head satisfies this inequality, we return $\mathbf{p}_\theta$.

This calibrated and threshold-driven early exit mechanism allows the model to balance efficiency with accuracy, ensuring that resource-intensive computations are only performed when necessary.

4 Experiments and Results

In this section, we present the results of our experiments[2] on training and evaluating the Phi-2 model with early exits placed at regular interval along the network, specifically after layer 6, 12, 18, 24. The underlying model has a total of 32 layers. We first describe the training process, including the impact of the hyperparameter λ on the penalty term and the effect of copying the last language model (LM) head versus initializing the weights for the early exit classifiers. We then present the results of our training experiments, highlighting the trade-offs between accuracy and entropy, which is used to improve our confidence metric.

[2]All computations are run on a server with an Intel(R) Xeon(R) Gold 5120 CPU and a Tesla V100 GPU with 32GB of Vram and 64GB of RAM. Code used in experiments to train and evaluate can be found under : https://anonymous.4open.science/r/BranchyLLM-B870

In the second part of this section, we focus on the inference phase, where we investigate the performance degradation of the model as we increase the speedup by exiting earlier in the network. We provide a comprehensive evaluation of our approach, including various evaluation scores and speedup metrics, to demonstrate the effectiveness of our method in achieving significant speedup while maintaining acceptable performance. Throughout our experiments, we use the Phi-2 model as a representative example, due to its relatively small size, which allows for rapid iteration and experimentation. However, our approach is theoretically applicable to any language model architecture.

4.1 Training

During training, we employed a modified loss function that deviates from the traditional cross-entropy loss. Our goal is to encourage the model to produce outputs with low confidence scores, allowing our confidence metric to effectively sort the most confident outputs. Specifically, our loss function incorporates a penalty term, weighted by the hyperparameter λ, to discourage the model from producing overly confident outputs.

To monitor the training process, we tracked three key metrics: (1) accuracy of each head, which measures the proportion of instances where the argmax of the head's output matches the argmax of the main model's output; (2) entropy, which indicates the level of uncertainty in the head's output; and (3) the modified loss function. These metrics provide insights into the model's ability to produce accurate and uncertain outputs, which are essential for effective early exiting.

Figure 1 illustrates the training dynamics for four different scenarios: (a) low penalty ($\lambda = 0.1$), (b) high penalty ($\lambda = 0.95$), and (c) using a pre-trained classification head from the main model, with and without penalty.

The results demonstrate several key findings. Firstly, the copied head without penalty shows better accuracy but insufficient entropy, making our confidence metric less meaningful and thus impairing the early exit performance. Secondly, adding a penalty to a pre-trained head does not achieve the desired balance; accuracy fails to reach potential levels without the penalty, and entropy does not increase significantly. Finally, for initialized heads, a strong penalty does not adversely affect accuracy but significantly increases entropy, which is beneficial for our early exit strategy.

Our analysis suggests that while pre-trained heads excel in accuracy, they

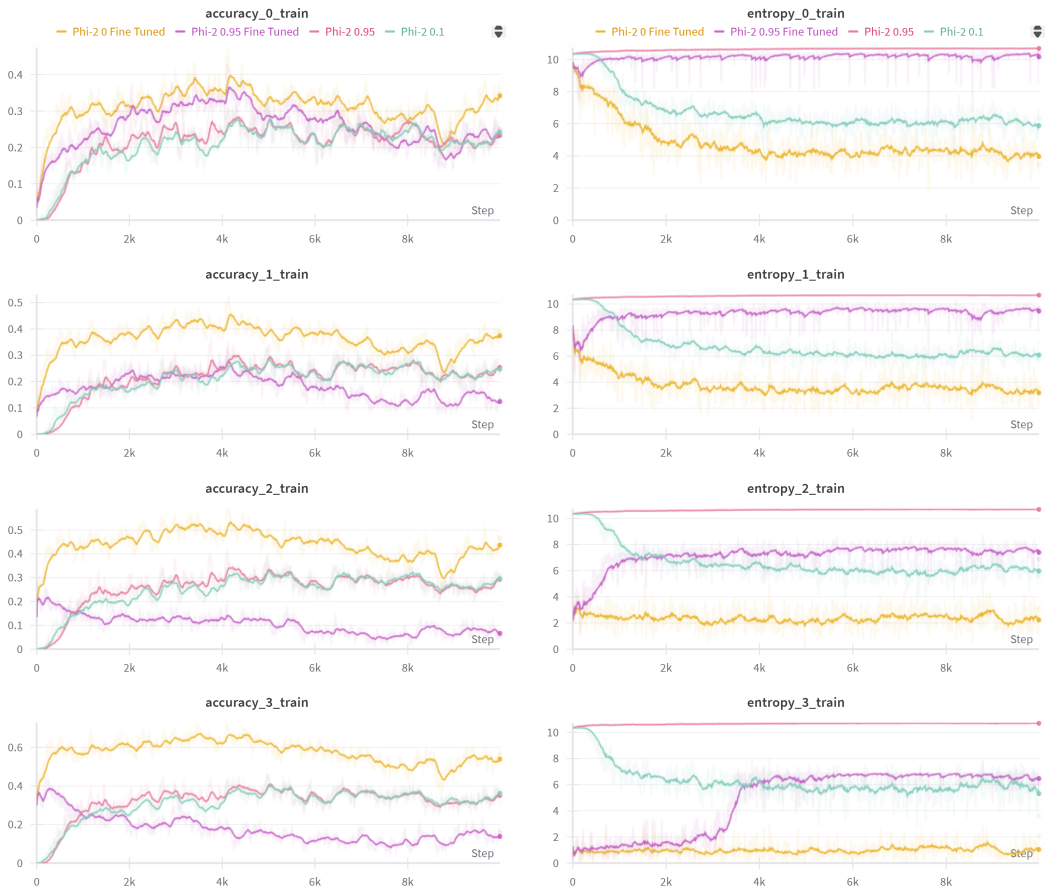

Figure 1: Accuracy and entropy metrics during training for different configurations of the early exit model. The first column shows the accuracy of each exit head, while the second column plots the corresponding entropy values. Four settings are compared: low entropy penalty ($\lambda = 0.1$) (**Phi-2 0.1**), high entropy penalty ($\lambda = 0.95$) (**Phi-2 0.95**), and heads initialized from the main model's classification head, with (**Phi-2 0 Fine Tuned**) and without penalty (**Phi-2 0.95 Fine Tuned**).

lack the necessary uncertainty for effective early exits unless modified appropriately. The incorporation of a penalty term during training, particularly with newly initialized heads, ensures that the model maintains high accuracy while also producing outputs with higher entropy. This balance is crucial for the functionality of our early exit mechanism, as it allows for confident early terminations without compromising overall model performance.

4.2 Inference

During inference, our early exit mechanism dynamically decides when to stop processing further based on a confidence metric. This decision is controlled by the parameter ϵ, which determines the confidence threshold for each exit head. By adjusting ϵ, we can control the trade-off between speed and accuracy: a lower ϵ results in more data being processed through early exits, leading to faster inference but potentially lower accuracy, whereas a higher ϵ ensures more accurate predictions at the cost of increased computational time.

4.2.1 Experimental Setup

We evaluated the performance of our early exit strategy using the Phi-2 model across various values of ϵ. We compared the following metrics:

- **Benchmark Scores**: Performance on different benchmarks, such as MMLU, Winogrande, and Hellaswag.
- **Speedup**: The reduction in computational time compared to the full model inference.
- **Exit Distribution**: The percentage of tokens exiting at each head.

4.2.2 Results

Figure 2 shows the benchmark scores of the model as a function of ϵ. On the MMLU benchmark, the scores remain relatively stable as ϵ decrease, suggesting that the early exit heads can make confident and correct predictions without significantly sacrificing performance. This is encouraging as it indicates that we are not losing accuracy while speeding up our inference for this benchmark. However, on the Hellaswag and Winogrande benchmarks, the scores show a clear increase with higher values of ϵ. This indicates that while our method

maintains efficiency for the MMLU task, it is less effective for Hellaswag and Winogrande, where lower ϵ values lead to a noticeable drop in performance.

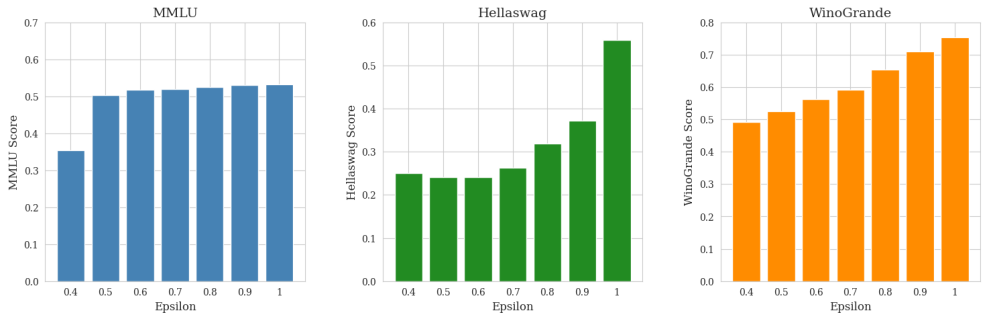

Figure 2: Model benchmark scores as a function of ϵ. Higher ϵ values correspond to higher scores.

Figure 3 illustrates the speedup achieved at different ϵ levels. A lower ϵ results in more tokens exiting early, leading to substantial computational savings. The speedup is most pronounced at lower ϵ values, demonstrating the efficiency gains from our early exit strategy.

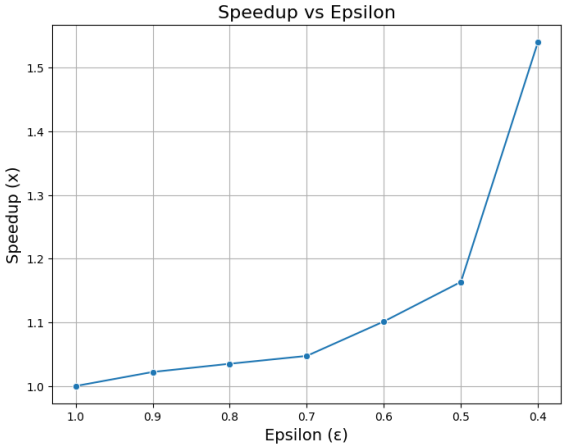

Figure 3: Inference speedup as a function of ϵ. Lower ϵ values lead to greater speedup.

The exit distribution across different heads for various ϵ values is shown in Figure 4. With lower ϵ, a larger proportion of tokens exit at the earlier heads, while higher ϵ values push more tokens towards the final layers. This

distribution highlights the flexibility of our approach in balancing speed and accuracy based on application requirements.

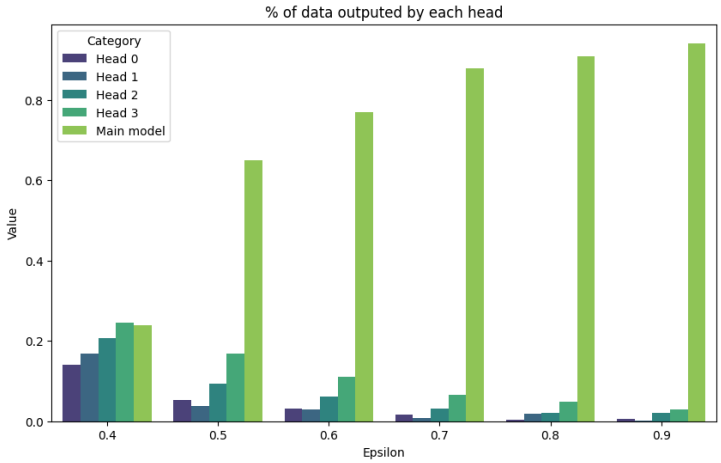

Figure 4: Distribution of token exits across different heads for various ϵ values.

4.2.3 Discussion

Our results demonstrate that the early exit strategy significantly enhances the efficiency of large language models. By carefully selecting ϵ, users can tailor the inference process to meet specific needs, achieving a desired balance between speed and accuracy. The ability to exit early without substantial loss in some benchmark performance makes this approach highly valuable for real-time applications on specific tasks.

The integration of early exits into the Phi-2 model showcases the potential for widespread adoption across other large language models.

5 Limitations and potential impacts

Our method demonstrates promising results with minimal degradation in performance on the MMLU benchmark, which suggests its potential effectiveness. However, we observed that other benchmarks experience a noticeable drop in performance. This indicates the necessity of evaluating our method on specific tasks to ensure performance retention. To maintain high accuracy across various tasks, a higher epsilon value should be used. However, this

approach reduces the usage of early exit heads, thereby limiting the overall usability and efficiency of the method.

Due to constraints in computational resources, we were unable to test our method on larger LLMs. Consequently, we selected a smaller LLM, phi-2, for our experiments. It is important to note that other, more substantial LLMs might exhibit different behaviors when subjected to our method. Therefore, further experimentation on a broader range of models is necessary to fully understand the generalizability and effectiveness of our approach across diverse architectures and scales.

About the impacts, our approach accelerates inference in large language models (LLMs), reducing computational resources and energy consumption significantly. This efficiency improvement enhances the practical usability of LLMs across various applications, resulting in lower operational costs and a smaller environmental footprint.

Careful task-specific evaluation is essential to avoid potential negative impacts, such as reduced accuracy and the risk of misinformation. While we observe minimal performance degradation on the MMLU benchmark, other tasks may experience accuracy loss. Such issues are inherent to all LLMs and underscore the need for rigorous validation to maintain output reliability.

6 Conclusion

In conclusion, we have presented a comprehensive framework for accelerating inference in large language models through the strategic integration of self-supervised early exits. By harnessing the inherent variability in token complexity, our method enables selective acceleration without sacrificing accuracy, thus addressing the pressing need for more efficient natural language processing solutions. The lightweight and modular nature of our approach not only facilitates seamless integration into existing pre-trained models but also enhances their practical usability across a wide array of real-world applications.

Throughout our experiments with the Phi-2 model, we have demonstrated significant computational savings while maintaining high accuracy on certain benchmarks. Our method offers a fine balance between speed and precision, allowing users to tailor the inference process according to specific requirements. Moreover, the adaptability of our approach highlights its potential for widespread adoption across diverse language model architectures, paving

the way for enhanced efficiency and scalability in natural language processing tasks.

Looking ahead, further research could explore optimizations and refinements to our methodology, such as fine-tuning the calibration process for different tasks or investigating alternative early exit strategies. Additionally, extending our experiments to larger and more complex language models could provide deeper insights into the scalability and generalizability of our approach. Ultimately, our work opens up exciting possibilities for accelerating inference in large language models, thereby advancing the frontier of natural language processing and enabling novel applications in various domains.

References

Haoli Bai, Wei Zhang, Lu Hou, Lifeng Shang, Jing Jin, Xin Jiang, Qun Liu, Michael Lyu, and Irwin King. BinaryBERT: Pushing the Limit of BERT Quantization, July 2021. URL `http://arxiv.org/abs/2012.15701`. arXiv:2012.15701 [cs].

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html`.

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. PaLM: Scaling Language Modeling with Pathways, October 2022. URL `http://arxiv.org/abs/2204.02311`. arXiv:2204.02311 [cs].

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL `https://aclanthology.org/N19-1423`.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv:2010.11929 [cs]*, October 2020. URL `http://arxiv.org/abs/2010.11929`. arXiv: 2010.11929.

Angela Fan, Edouard Grave, and Armand Joulin. Reducing Transformer Depth on Demand with Structured Dropout, September 2019. URL `http://arxiv.org/abs/1909.11556`. arXiv:1909.11556 [cs, stat].

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. Textbooks are all you need, 2023.

Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q. Weinberger. Multi-scale dense networks for resource efficient image classification, 2018.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Haotang Deng, and Qi Ju. Fastbert: a self-distilling bert with adaptive inference time, 2020.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust Speech Recognition via Large-Scale Weak Supervision, December 2022. URL `http://arxiv.org/abs/2212.04356`. arXiv:2212.04356 [cs, eess].

Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadepally. From Words to Watts: Benchmarking the Energy Costs of Large Language Model Inference, October 2023. URL `http://arxiv.org/abs/2310.03003`. arXiv:2310.03003 [cs].

Simone Scardapane, M. Scarpiniti, E. Baccarelli, and A. Uncini. Why should we add early exits to neural networks? *Cognitive Computation*, 12:954 – 966, 2020. doi: 10.1007/s12559-020-09734-4.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. Q-BERT: Hessian Based Ultra Low Precision Quantization of BERT, September 2019. URL `http://arxiv.org/abs/1909.05840`. arXiv:1909.05840 [cs].

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient Knowledge Distillation for BERT Model Compression, August 2019. URL `http://arxiv.org/abs/1908.09355`. arXiv:1908.09355 [cs].

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices, April 2020. URL `http://arxiv.org/abs/2004.02984`. arXiv:2004.02984 [cs].

Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2464–2469, 2016. doi: 10.1109/ICPR.2016.7900006.

Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. Branchynet: Fast inference via early exiting from deep neural networks, 2017.

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, Vincent Zhao, Yanqi Zhou, Chung-Ching Chang, Igor Krivokon, Will Rusch, Marc Pickett, Pranesh Srinivasan, Laichee Man, Kathleen 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 Aguera-Arcas, Claire Cui, Marian Croak, Ed Chi, and Quoc Le. LaMDA: Language Models for Dialog Applications, February 2022. URL `http://arxiv.org/abs/2201.08239`. arXiv:2201.08239 [cs].

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and Efficient Foundation Language Models, February 2023. URL `http://arxiv.org/abs/2302.13971`. arXiv:2302.13971 [cs].

Florian Valade, Mohamed Hebiri, and Paul Gay. Eero: Early exit with reject option for efficient classification with limited budget, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, August 2023. URL `http://arxiv.org/abs/1706.03762`. arXiv:1706.03762 [cs].

V. Vovk, A. Gammerman, and C. Saunders. Machine-learning applications of algorithmic randomness. In *In Proceedings of the Sixteenth International Conference on Machine Learning*, pages 444–453. Morgan Kaufmann, 1999.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. DeeBERT:

Dynamic Early Exiting for Accelerating BERT Inference, April 2020. URL http://arxiv.org/abs/2004.12993. arXiv:2004.12993 [cs].

Z. Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Neural Information Processing Systems*, 2022. doi: 10.48550/ARXIV.2206.01861.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. OPT: Open Pre-trained Transformer Language Models, June 2022. URL http://arxiv.org/abs/2205.01068. arXiv:2205.01068 [cs].

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. BERT Loses Patience: Fast and Robust Inference with Early Exit, October 2020. URL http://arxiv.org/abs/2006.04152. arXiv:2006.04152 [cs].

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. A Survey on Model Compression for Large Language Models, September 2023. URL http://arxiv.org/abs/2308.07633. arXiv:2308.07633 [cs].