

Investigating Acceleration of LLaMA Inference by Enabling Intermediate Layer Decoding via Instruction Tuning with ‘LITE’

Neeraj Varshney Agneet Chatterjee Mihir Parmar Chitta Baral
Arizona State University

Abstract

Large Language Models (LLMs) have achieved remarkable performance across a wide variety of tasks; however, their large size makes their inference slow and computationally expensive. Focusing on this problem, we study instruction tuning LLMs with additional explicit Losses from the InT_{er}mediate layErs (LITE) and show that it enables these layers to acquire ‘good’ generation ability without affecting the generation ability of the final layer. We then perform ‘dynamic confidence-based early exiting’ at token level from the intermediate layers which improves the computational efficiency of text generation without sacrificing the quality of the generation. We conduct comprehensive experiments by instruction tuning LLaMA-2 models on the Alpaca dataset and evaluate on four different instruction test sets. We show that dynamic early exiting achieves consistent and considerable inference cost improvements (37.86% for 7B and 46.35% for 13B model) while maintaining the generation quality. We further conduct a thorough analysis of the results and dissect the efficiency improvements which reveals several important findings.

1 Introduction

Recently developed LLMs (Touvron et al., 2023; OpenAI, 2023; Chowdhery et al., 2022; Rae et al., 2021; Smith et al., 2022) have revolutionized the field of natural language processing and achieved remarkable performance across a wide variety of tasks. ‘Instruction Tuning’ further teaches these models to follow the user’s instruction provided in natural language (Wei et al., 2022; Mishra et al., 2022; Sanh et al., 2022; Wang et al., 2022; Chung et al., 2022). Despite all the notable abilities of these models, their large size (number of parameters) makes their inference slow and computationally expensive which poses a practical challenge limiting their widespread adoption in resource constrained applications. Focusing on the above problem, in this work, we investigate instruction tuning

LLMs in a way that enables intermediate layer decoding for efficiently generating text without compromising the quality of the generation.

We first show that in standard instruction tuning, only the final layer of the model acquires the ability to generate ‘quality’ text while the representations of the intermediate layers (when passed through the language modeling head) fail to do so. This restricts decoding from these intermediate layers without degrading the generation quality. Addressing this point, we instruction tune LLMs with additional explicit Losses from the InT_{er}mediate layErs (LITE) and show that it enables these layers to acquire ‘good’ generation ability. Importantly, we show that these layers acquire this ability without affecting the generation ability of the final layer; however, as expected, their generation ability still remains slightly inferior to the generation ability of the final layer. Thus, decoding the complete response from intermediate layers improves the efficiency of inference but still results in degradation in the quality of the response.

Addressing the above limitation, we show that (a) LITE greatly aligns the intermediate layers’ token prediction with that of the final layer and (b) the intermediate layers’ token prediction probabilities provide a strong signal of this alignment. Building on these findings, we perform ‘dynamic confidence-based early exiting’ at token level from the intermediate layers which improves the efficiency of inference while maintaining the generation quality.

We conduct comprehensive experiments by instruction tuning LLaMA-2 models (Touvron et al., 2023) on the widely used Alpaca dataset (Taori et al., 2023) and holistically evaluate on four different human-instruction test sets including Vicuna, WizardLM, Koala, and Self-Instruct. Figure 1 compares the quality of responses (evaluated using the Claude model as detailed in Section 5) and the inference cost (measured in FLOPs) of the (i) standard generation method from the final layer with



Figure 1: Comparing the quality of the responses and the inference cost of (i) the standard generation from the final layer (orange) and (ii) the dynamic early exiting method (blue) on model tuned with LITE. The top and the bottom rows show the effectiveness on four different test sets for the LLaMA-2 7B and 13B models, respectively.

(ii) the dynamic early exiting method. It shows that dynamic early exiting achieves consistent and considerable inference cost improvements (37.86% for 7B and 46.35% for 13B model on average) while maintaining the generation quality.

We further perform a thorough analysis of the results over several important aspects, such as, comparing the semantic similarity between the responses generated from the final layer and the early exiting method, and dissecting the efficiency improvements by comparing the number of tokens generated in the outputs. We also discuss the potential of intermediate layer decoding in ‘*speculative sampling*’ and ‘*hallucination detection*’.

In summary, our work contributes to improving the efficiency of LLM inference while maintaining the generation quality, a crucial step en route to enabling their widespread adoption.

2 Related Work

Improving the inference efficiency of LLMs is an important research direction and is receiving considerable attention from the NLP community. In this section, we review some of the existing methods and differentiate our work from them.

Reducing model size: Since model size plays a crucial role in increasing the inference cost and latency, techniques like **quantization** (Dettmers et al., 2022; Yao et al., 2022; Xiao et al., 2023; Frantar et al., 2023), **knowledge distillation** (Hsieh et al., 2023; Jiao et al., 2020; Li et al., 2022), **model compression and network pruning** (Wang et al.,

2020; Guo et al., 2021) have been shown to be effective in improving the inference efficiency.

Furthermore, during sampling, a cache of the keys and values can be maintained for every attention layer which reduces the computations at inference time (**KV caching**). However, it increases the GPU VRAM memory requirement of inference.

Another technique **speculative sampling** (Leviathan et al., 2023; Chen et al., 2023) first generates a draft of K tokens from a smaller auto-regressive model and then scores the draft using the target model. This results in generation of more than one token (on average) from the target model in a single pass.

Early exiting and cascading based inference techniques have been shown to be effective for classification tasks with BERT-style models, such as DeeBERT (Xin et al., 2020) that speeds up BERT inference by inserting extra classification layers between each encoder layer, POWER-BERT (Goyal et al., 2020) that focuses on progressive word-vector elimination (based on significance computed using self-attention) along the encoder pipeline, DynaBERT (Hou et al., 2020) that adjusts the size of the model by selecting adaptive width and depth, and cascading (Varshney and Baral, 2022; Li et al., 2021; Varshney et al., 2022; Yue et al., 2023; Cheng et al., 2023; Varshney and Baral, 2023) in which sequential inference is done through models of bigger and bigger size with conditional exiting to output predictions efficiently. Our work is also related to Confident Adaptive Language Modeling (CALM)

(Schuster et al., 2022) and Depth-Adaptive Transformers (Elbayad et al., 2020) in which early exiting is performed by learning additional classifiers attached to the decoder layers.

Din et al. (2023) proposed to short-cut away transformer inference in between certain layers by learning linear transformations across layers in the network, i.e., **casting internal representations**. O’Brien and Lewis (2023); Gera et al. (2023) explore leveraging the intermediate layers for contrastive decoding to improve reasoning. We discuss other related and concurrent work in Appendix H.

Our work differs in the following aspects:

(1) Firstly, most of the existing work in early exiting focuses on improving the efficiency of encoder-only models (like BERT) or encoder-decoder models (like T5); our work focuses on the current state-of-the-art decoder-only LLMs (LLaMA-2). Furthermore, we focus on the instruction tuning setting with text generation, unlike prior work that focused on solving simpler tasks like GLUE classification or QA. (2) Early exiting methods typically require training additional classifiers for the intermediate layers, however, in this work, we use the same shared language modeling head at all the layers; thus, we do not introduce new model weights. (3) For leveraging the intermediate layers for decoding, we enable them to acquire generation ability by instruction tuning with LITE, unlike other methods that use a pre finetuned model in which these layers have poor generation ability as we show in Section 6.1. (4) Existing methods typically require complex architectural modifications, pruning, saliency quantification, or training new parameters. In contrast, our method (both for tuning and inference) is simple and easy to implement and yet achieves considerable benefits. (5) Existing methods typically require training a separate model for each computation budget; however, in our method, the same model can be adapted to meet all the computation constraints (by varying the exiting confidence thresholds). (6) The computational efficiency often comes with a compromise in performance. However, our method maintains the generation quality while providing efficiency benefits.

3 Instruction Tuning with LITE

Instruction Tuning (IT): One of the major reasons that necessitate instruction tuning of LLMs is the mismatch between their pre-training objective and the users’ objective, i.e., LLMs are typically

trained on minimizing the word prediction error while users want the model to follow their instructions. To this end, an instruction tuning dataset is collected and a pre-trained model is fine-tuned in a supervised manner (Mishra et al., 2022; Chung et al., 2022; Wei et al., 2021). Loss calculation during instruction tuning of a typical decoder-only LLM (LLaMA in this case) is shown in Figure 2 (left). The model consists of a stack of decoder layers followed by a language modeling head which outputs the probability distribution over the vocabulary tokens as its prediction. During the supervised fine-tuning, the loss over the output tokens is back-propagated from the final layer of the model:

$$Loss(y_{1:M}) = - \sum_{t=1}^M \log p(y_t | y_{<t})$$

IT with LITE: We show that in standard instruction tuning, only the final layer of the model acquires the ability to generate ‘quality’ text while the representations of the intermediate layers (when passed through the language modeling head) fail to do so (Section 6.1). In other words, it does not explicitly teach the intermediate layers of the tuned LLM to generate tokens. This restricts decoding from these intermediate layers without degrading the generation quality.

We note that during tuning, the same language modeling head (that is used with the final layer) can also be used with the intermediate layers to obtain the losses of those layers. Thus, this does not impact the number of parameters of the model. To this end, we calculate a weighted aggregation of the losses from the intermediate layers (including the final) to calculate the overall loss value:

$$Loss = \frac{\sum_{i=1}^N w_i Loss_i}{\sum_{i=1}^N w_i}$$

where N is the number of layers, w_i is the weight of the i^{th} layer, and $Loss_i$ is the cross entropy loss of the i^{th} layer as shown in Figure 2.

During training, we use the representations of the intermediate layers and calculate the loss from these layers at the end. We note that this is a general formulation as it captures a variety of scenarios including the standard fine-tuning in which the loss is calculated only from the last layer (i.e., $w_{1:N-1} = 0$ and $w_N = 1$). Furthermore, this formulation also allows aggregating losses from only the selected intermediate layers instead of all the layers

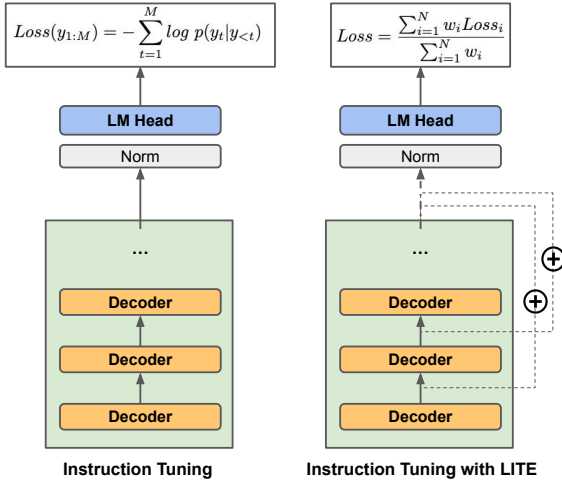


Figure 2: Loss calculation for standard instruction tuning (left) and instruction tuning with additional explicit losses from the intermediate layers LITE (right).

by accordingly defining the LM head pathways and the w_i values. In Section 6.2, we will show that this formulation while enabling the intermediate layers with ‘good’ generation ability does not adversely affect the final layer’s generation ability. Furthermore, as expected, the quality of generation typically improves with the layer number as the later layers have more capacity to learn.

4 Making Inference Efficient

In this section, we first detail auto-regressive inference and then describe early exiting techniques.

Auto-Regressive Inference: It refers to the process of generating a sequence of tokens where each token is generated based on the preceding tokens in the sequence. For generating a token, the model takes the input (including the previously generated tokens) and runs a forward pass in which the input is fed to the model and passed sequentially along its layers until the probabilities for the next token are predicted (called as **logits**). Chaining model forward passes with next token selection iteratively leads to the generation of text. In greedy decoding, the token with the highest probability is selected as the next word prediction at each timestep.

4.1 Fixed Early Exiting

Since instruction tuning with LITE enables the intermediate layers to acquire ‘good’ generation ability, the computations during inference can be terminated at a pre-specified intermediate layer (referred to as **exiting layer**) and the language modeling head can be used to predict the next token. This

saves the computations of the remaining layers that follow the specified exiting layer and thus it improves the efficiency of inference.

Though this method of fixed early exiting leads to improvement in the efficiency of inference, it is bound to result in some degradation in the quality of the generation as the generation ability of an intermediate layer still remains inferior to the generation ability of the final layer. However, the quality of generation typically improves with the layer number as the later layers have more capacity.

4.2 Dynamic Confidence-Based Early Exiting

Addressing the limitation of the fixed early exiting method, we study a dynamic early exiting method that decides the exiting layer for a token prediction based on the intermediate layer’s probability of the prediction (softmax over the logit values).

This is motivated by our following two findings: (a) Instruction Tuning with LITE greatly aligns the intermediate layers’ token prediction with that of the final layer (Section 6.3) and (b) The intermediate layers’ token prediction probabilities (referred to as **confidence**) provide a strong signal of this alignment (Section 6.4).

Building on these two findings, we perform ‘dynamic confidence-based early exiting’ at token level from the intermediate layers which improves the efficiency of inference while maintaining the generation quality. Specifically, a set of intermediate layers with their corresponding confidence thresholds are defined and at inference time, the exiting decision for a prediction is taken by comparing the intermediate layer’s prediction confidence against its corresponding threshold. This enables the model to perform efficient inference without degrading the generation quality. Note that this method does not introduce new parameters and uses the softmax probability to make the exiting decision. We study this exiting method for inference without KV caching.

5 Experimental Setup

Instruction Tuning: We instruction tune the LLaMA-2 models (Touvron et al., 2023) (7B and 13B) with the widely used Alpaca dataset (Taori et al., 2023). Alpaca consists of 52K instruction-following demonstrations generated using the self-instruct (Wang et al., 2023b) technique. In IT with LITE for 7B model (32 total layers), we aggregate losses from the following selected intermediate lay-

ers: (8, 12, 16, 20, 24, 28) along with the final layer and use equal weights in loss calculation. Similarly, for the 13B model (40 total layers), we use (8, 12, 16, 20, 24, 28, 32, 36) layers. We perform full parameter fine-tuning on 4 A100 GPUs.

We skip selecting the initial layers because they have a limited capacity to learn and thus can not give good token predictions. Furthermore, we select layers at an interval of 4 so that at inference time, the model can do enough reasoning/interactions between two consecutive checkpoints. Otherwise, checking at every layer can result in computational overhead. We train this model for 5 epochs so that it achieves training loss comparable to standard tuning.

We present all the results corresponding to this tuning configuration in the main paper and present the study corresponding to weighted LITE in the Appendix C.

Evaluation Datasets: To perform holistic evaluation, we experiment with four different human-instruction test sets including Vicuna (Chiang et al., 2023), Self-Instruct (Wang et al., 2023b), Koala (Geng et al., 2023), and WizardLM (Xu et al., 2023). We select these evaluation test sets as they can together cover a large number and types of instructions thus resulting in a comprehensive evaluation. Table 4 shows the statistics of the datasets.

Evaluation Methodology: The evaluation of the instruction-following ability of LLMs is challenging due to the existence of multiple correct responses to an input and the infeasibility of reproducing human evaluations. Addressing this problem, recent works have started to rely on automatic evaluations using LLMs (Zheng et al., 2023; Chiang et al., 2023). Specifically, we use Claude LLM (Bai et al., 2022) as a judge to compare the quality of responses of two models on a given instruction. We note that these LLMs have been shown to be vulnerable to position bias in their judgment (Wang et al., 2023a). To circumvent this bias, we evaluate a response pair with both orderings of the responses and then aggregate the judgment scores. We provide the prompt for comparing the quality of the responses of two models in Appendix A.

6 Results and Analysis

In this section, we first demonstrate the inability of the intermediate layers of the model tuned with standard IT to generate ‘quality’ text (6.1). Then,

we show the impact of IT with LITE: it does not adversely affect the generation quality of the final layer (6.2), it aligns the intermediate layers’ token predictions with the final layer (6.3), and the corresponding prediction confidence values provide a strong signal of the alignment (6.4). These findings motivate dynamic confidence-based early exiting. Finally, we show the effectiveness of the method in improving the efficiency of inference while maintaining the generation quality (6.5). To avoid repetition, we present results for the 7B model in the main paper and for the 13B model in B.8.

6.1 Generation Ability of Intermediate Layers

In order to obtain the text (sequence of tokens) generated via fixed exiting from an intermediate layer, we apply the normalization (RMSNorm) followed by the language modeling head to the representations of that intermediate layer and skip the computations of the layers following the exiting layer (as detailed in Section 4.1). For the model tuned with the standard instruction tuning, we compare the quality of the text (as detailed in Section 5) generated from different intermediate layers against the final layer’s generation in Figure 3. As expected, the intermediate layers generate text of considerably degraded quality and this quality drops as the layer number decreases.

This demonstrates that with standard instruction tuning, only the later layers (primarily the final layer) of the model acquire the ability to generate ‘quality’ text while the representations of the intermediate layers (when passed through the language modeling head) fail to do so. Thus, for such a model, the early exiting method saves the inference computation cost but considerably degrades the generation quality. This restricts employing such early exiting techniques for the model tuned with standard instruction tuning. We show examples of responses obtained via fixed early exiting from different intermediate layers in Appendix B.1.

We perform instruction tuning with LITE to enable the intermediate layers to acquire ‘good’ generation ability. Importantly, we show that these layers acquire this ability without affecting the generation ability of the final layer (Section 6.2).

6.2 Impact of LITE on the Final Layer

In Figure 4, we compare the quality of responses of (a) the model tuned using standard instruction tuning (IT) and (b) the model tuned using IT with LITE. Note that the responses for both these models

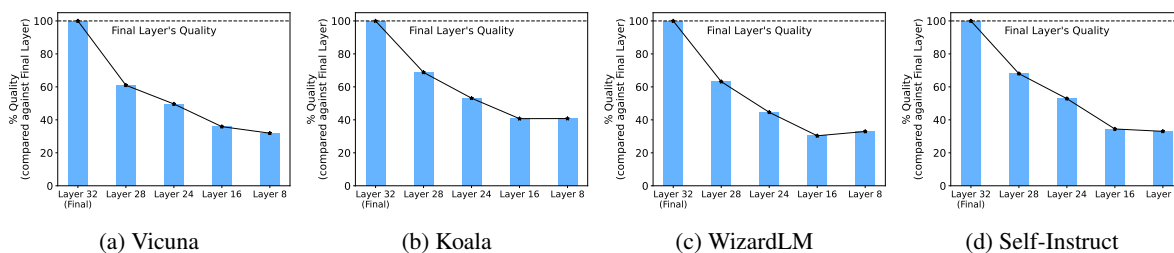


Figure 3: Demonstrating quality comparison of the output of intermediate layers (generated via fixed exiting) against the final layer’s generation of the model tuned with standard instruction tuning.

correspond to their respective final layer’s output. From the figure, it can be observed that for all the datasets, the outputs of both models are of comparable quality which shows that tuning with LITE does not adversely affect the generation ability of the final layer of the model.

Next, we demonstrate two important characteristics of instruction tuning with LITE (in 6.3 and 6.4) that motivate us to study dynamic confidence-based early exiting from the intermediate layers.

6.3 ‘Alignment’ of Intermediate Layers

We define percentage ‘alignment’ of a layer as the measure of how often the token predictions of that layer match with that of the final layer (given same input prefixes). For this study, we do not do early exiting, instead we just use the representation of each intermediate layer and pass it through the LM head to obtain the corresponding token prediction of each layer. Note that for generating the next token, we follow the standard generation methodology and append the predicted token of the last layer to the input to obtain the token prediction of all the layers given the same input prefixes.

In Figure 5, we plot the percentage alignment of token predictions of all intermediate layers with the token predictions of the final layer. The figure shows the percentage alignment of (i) the model tuned using standard IT (orange) and (ii) the model tuned using IT with LITE (blue). We show this result aggregated over all the output token predictions for all the inputs of the corresponding dataset.

We draw the following inferences:

(a) The predictions of the intermediate layers of the model tuned with LITE align well with the final layer, i.e., given a prefix, the intermediate layers’ token predictions match quite well with the final layer’s token prediction. In contrast for the model tuned using IT, the token predictions of the intermediate layers do not align well with the final layer’s token predictions.

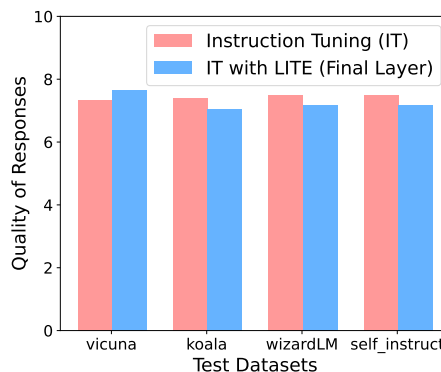


Figure 4: Comparing quality of responses of (a) model tuned using IT and (b) model tuned using IT with LITE. The outputs of the models are of comparable quality.

(b) As the layer number increases, the % alignment also increases, i.e., given a prefix, the predicted token of the later layers shows higher alignment (with the final layer) than the initial layers.

(c) There are some peaks in the curve for IT with LITE which correspond to the selected layers from which the loss is aggregated during tuning, i.e., these layers show higher alignment as expected.

In summary, this study demonstrates that IT with LITE greatly aligns the token predictions of intermediate layers with that of the final layer.

6.4 Token Probability and Alignment

We plot the relationship between the token prediction confidence (softmax over the logits of the LM head) of the intermediate layers and the percentage alignment with the token prediction of the final layer. Figure 6 shows this plot for the model tuned with LITE. The figure shows that in IT with LITE, the intermediate layers’ token prediction probabilities provide a strong signal of alignment, i.e., a high token prediction confidence implies a higher likelihood of its alignment with the token prediction of the final layer. It also shows that with the increase in the layer number, the percentage align-

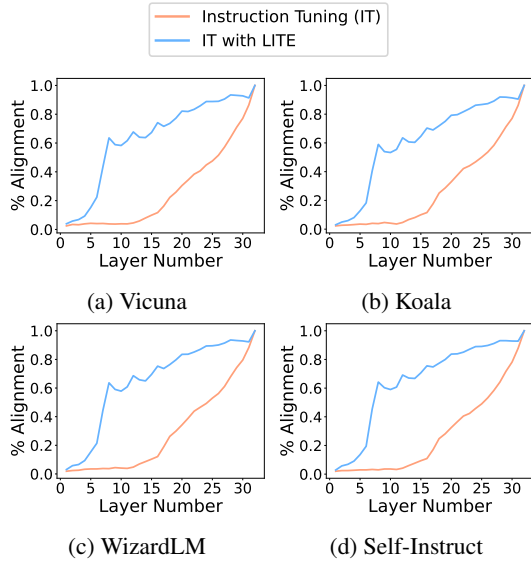


Figure 5: Comparing percentage ‘alignment’ of intermediate layer token predictions with the token predictions of the final layer for the model tuned using IT (orange) and the model tuned using IT with LITE (blue).

ment typically increases at the same confidence values. In contrast, in standard instruction tuning (IT), the confidence is not well correlated with the percentage alignment as we show in Appendix B.6.

6.5 Effectiveness of Dynamic Early Exiting

Motivated by the findings of the previous two subsections (6.3 and 6.4), we perform dynamic confidence-based early exiting at token-level, i.e., we exit when the token prediction confidence of the intermediate layer is sufficiently high (thus it is likely to align with the final layer’s prediction).

To this end, from the confidence vs percentage alignment curve, we identify a confidence threshold for each layer where the alignment is $> 95\%$. Specifically, we use the following thresholds: Layer 8: 0.95, Layer 12: 0.95, Layer 16: 0.9, Layer 20: 0.9, Layer 24: 0.8, and Layer 28: 0.7.

In the main paper, we present the results and analysis for the aforementioned configuration. However, we note that a different threshold configuration can also be used for inference. For instance, a more aggressive configuration with lower thresholds (shown in Appendix B.7) leads to even more cost improvements (49.92%); though it slightly drops the quality of generation (5.34%). The trade-off between quality and cost can be balanced depending on the application requirements. For example, applications with quality tolerance or resource limitations can keep low threshold to

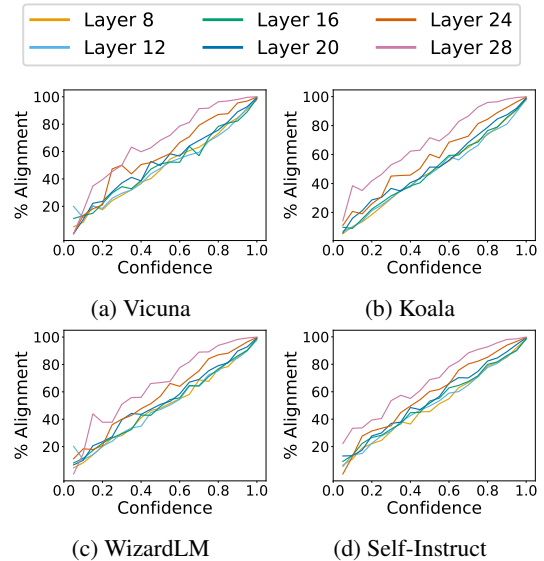


Figure 6: Demonstrating trend of token prediction confidence of the intermediate layers and the % alignment with the final layer for model tuned with LITE.

Test Dataset	Inference Cost Improvement (%)
Vicuna	33.39 %
Koala	35.40 %
WizardLM	36.12 %
Self Instruct	46.54 %

Table 1: Percentage improvements in the inference cost (measured in FLOPs) with dynamic early exiting.

achieve higher cost improvements.

Dynamic confidence-based early exiting: At a selected layer, we pass its representations through the LM head, calculate the softmax logit value, and compare it with the corresponding confidence threshold of the layer. If it surpasses the threshold value then we exit from that layer and proceed to generate the next token, otherwise, we repeat this process at the next selected layer.

Figure 1 (in Section 1) compares the quality of responses and the inference cost (measured in FLOPs) of the standard generation method (final layer) with the dynamic early exiting method. It shows that the dynamic early exiting method achieves consistent and considerable cost improvements (37.86% for 7B and 46.35% for 13B model on average) while maintaining the generation quality. Table 1 shows the percentage improvements in inference cost for each test set individually. We note that we use FLOPs as the metric of showcasing inference efficiency improvements because it is hardware independent, unlike latency.

6.5.1 Semantic Similarity of the Responses

In addition to comparing the quality, we also compare the semantic similarity between the responses of the final layer and the dynamic early exiting. Table 2 shows the semantic similarity (calculated using the ‘en_core_web_sm’ spacy model) for the four datasets. It shows that there is a large semantic similarity between the responses as the values are closer to 1. This implies that dynamic early exiting maintains the semantics of the responses while providing efficiency benefits. Appendix B.4 shows examples of responses from both the last layer and the dynamic early exiting method.

6.5.2 Dissecting the Cost Improvements

In Figure 7, we compare the average number of tokens generated in the final layer’s responses and the dynamic early exiting responses. It shows that both the methods generate a comparable number of tokens in their respective outputs. This asserts that the cost improvement resulting in dynamic early exiting is because of the reduced computations and not due to generating a lesser number of tokens.

6.5.3 Contribution of Different Exiting Layers

Figure 8 shows the percentage of token outputs from different exit layers. Note that this is aggregated across all the token positions. This shows that the model exits a considerable percentage of times from the intermediate layers (while maintaining the generation quality) which further justifies the improvement in inference efficiency. We further conduct several interesting studies and analyses of the results and present them in Appendix B.

6.5.4 Effectiveness at Category Level

Vicuna and WizardLM datasets also provide the category corresponding to different test instances. To this end, we present category-level quality and inference cost results for these datasets in Figure 9 and 10 (Appendix). On average, the method results in cost improvement of 33.39% on Vicuna and 36.12% on WizardLM.

7 Conclusion and Discussion

In this work, we proposed instruction tuning with additional explicit losses from the intermediate layers and showed that it enables these layers to acquire ‘good’ generation ability without affecting the final layer’s generation ability. We performed ‘dynamic confidence-based early exiting’ at token level from the intermediate layers and showed that

Test Dataset	Semantic Similarity
Vicuna	0.9135
Koala	0.8940
WizardLM	0.9020
Self Instruct	0.9001

Table 2: Semantic similarity between the final layer’s and the dynamic early exiting responses on test sets.

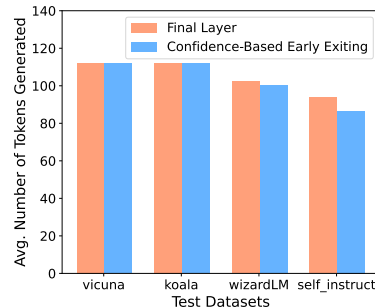


Figure 7: Comparing the average number of tokens generated in the final layer’s responses and the dynamic early exiting responses for the four datasets.

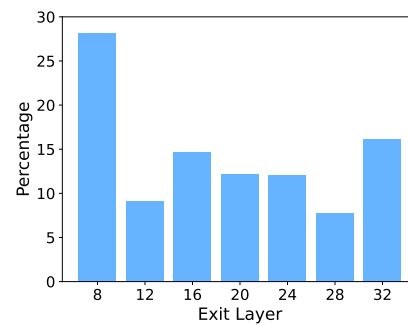


Figure 8: Percentage of token outputs from different exiting layers in the proposed method.

it improves the efficiency of inference while maintaining the generation quality. We further conducted a thorough analysis that resulted in several important findings. Overall, our work contributes to improving the efficiency of LLM inference while maintaining the generation quality, a crucial step en route to enabling their widespread adoption.

Looking forward, our work additionally opens up several other avenues for new research, such as speculative sampling from the intermediate layers to improve the inference efficiency and checking information consistency from the output of intermediate layers to detect hallucinations (discussed in Appendix I). Furthermore, this approach is complementary to some existing efficiency methods described in Section 2 and H, i.e., they can be used in conjunction to achieve even more efficiency gains.

Limitations

We have shown the efficacy of intermediate layer decoding to achieve inference efficiency benefits. However, utilizing the intermediate layers requires instruction tuning with LITE. This is because the standard instruction tuning does not teach the intermediate layers to generate quality text. Thus, for utilizing the intermediate layers, the model needs to be trained using instruction tuning with LITE and an already instruction-tuned model may not provide the same efficiency benefits while maintaining the generation ability. Though this method is generally applicable, we investigate it with LLaMA-2 models as also explicitly specified in the paper title. This is because the LLaMA-2 models are one of the best performing publicly available models and they are widely being used in ongoing NLP research. We also clearly note that this research is limited to the English language.

Ethics Statement

We have used AI assistants (Grammarly and ChatGPT) to address the grammatical errors and rephrase the sentences.

Acknowledgement

We thank the anonymous reviewers for their insightful feedback. This research was supported by the 2023 Spring Amazon Research Award (ARA).

References

- Amos Azaria and Tom Mitchell. 2023. [The internal state of an LLM knows when it’s lying](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 967–976, Singapore. Association for Computational Linguistics.
- Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. 2022. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.
- Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*.
- Zhoujun Cheng, Jungo Kasai, and Tao Yu. 2023. [Batch prompting: Efficient inference with large language model APIs](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 792–810, Singapore. Association for Computational Linguistics.
- Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. [Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality](#).
- Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.
- Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.
- Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. [GPT3.int8\(\): 8-bit matrix multiplication for transformers at scale](#). In *Advances in Neural Information Processing Systems*.
- Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2023. Chain-of-verification reduces hallucination in large language models. *arXiv preprint arXiv:2309.11495*.
- Alexander Yom Din, Taelin Karidi, Leshem Choshen, and Mor Geva. 2023. Jump to conclusions: Short-cutting transformers with linear transformations. *arXiv preprint arXiv:2303.09435*.
- Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. 2020. [Depth-adaptive transformer](#). In *International Conference on Learning Representations*.
- Angela Fan, Edouard Grave, and Armand Joulin. 2020. [Reducing transformer depth on demand with structured dropout](#). In *International Conference on Learning Representations*.
- Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. [OPTQ: Accurate quantization for generative pre-trained transformers](#). In *The Eleventh International Conference on Learning Representations*.
- Xinyang Geng, Arnav Gudibande, Hao Liu, Eric Wallace, Pieter Abbeel, Sergey Levine, and Dawn Song. 2023. [Koala: A dialogue model for academic research](#). Blog post.
- Ariel Gera, Roni Friedman, Ofir Arviv, Chulaka Gunasekara, Benjamin Sznajder, Noam Slonim, and Eyal Shnarch. 2023. [The benefits of bad advice: Autocontrastive decoding across model layers](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1)*:

- Long Papers*), pages 10406–10420, Toronto, Canada. Association for Computational Linguistics.
- Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2023. Critic: Large language models can self-correct with tool-interactive critiquing. *arXiv preprint arXiv:2305.11738*.
- Saurabh Goyal, Anamitra Roy Choudhury, Saurabh Rajee, Venkatesan Chakaravarthy, Yogish Sabharwal, and Ashish Verma. 2020. Power-bert: Accelerating bert inference via progressive word-vector elimination. In *International Conference on Machine Learning*, pages 3690–3699. PMLR.
- Demi Guo, Alexander Rush, and Yoon Kim. 2021. [Parameter-efficient transfer learning with diff pruning](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4884–4896, Online. Association for Computational Linguistics.
- Zhenyu He, Zexuan Zhong, Tianle Cai, Jason D Lee, and Di He. 2023. Rest: Retrieval-based speculative decoding. *arXiv preprint arXiv:2311.08252*.
- Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. 2020. [Dynabert: Dynamic bert with adaptive width and depth](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 9782–9793. Curran Associates, Inc.
- Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. [Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes](#). In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8003–8017, Toronto, Canada. Association for Computational Linguistics.
- Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. [Llmlingua: Compressing prompts for accelerated inference of large language models](#). *arXiv preprint arXiv:2310.05736*.
- Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. [TinyBERT: Distilling BERT for natural language understanding](#). In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.
- Parsa Kavehzadeh, Mojtaba Valipour, Marzieh Tahaei, Ali Ghodsi, Boxing Chen, and Mehdi Rezagholizadeh. 2023. Sorted llama: Unlocking the potential of intermediate layers of large language models for dynamic inference using sorted fine-tuning (soft). *arXiv preprint arXiv:2309.08968*.
- Sehoon Kim, Karttikeya Mangalam, Jitendra Malik, Michael W Mahoney, Amir Gholami, and Kurt Keutzer. 2023. Big little transformer decoder. *arXiv preprint arXiv:2302.07863*.
- Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.
- Lei Li, Yankai Lin, Deli Chen, Shuhuai Ren, Peng Li, Jie Zhou, and Xu Sun. 2021. [CascadeBERT: Accelerating inference of pre-trained language models via calibrated complete models cascade](#). In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 475–486, Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Zheng Li, Zijian Wang, Ming Tan, Ramesh Nallapati, Parminder Bhatia, Andrew Arnold, Bing Xiang, and Dan Roth. 2022. [DQ-BART: Efficient sequence-to-sequence model via joint distillation and quantization](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 203–211, Dublin, Ireland. Association for Computational Linguistics.
- Potsawee Manakul, Adian Liusie, and Mark Gales. 2023. [SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9004–9017, Singapore. Association for Computational Linguistics.
- Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. [Cross-task generalization via natural language crowdsourcing instructions](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3470–3487, Dublin, Ireland. Association for Computational Linguistics.
- Xuefei Ning, Zinan Lin, Zixuan Zhou, Huazhong Yang, and Yu Wang. 2023. [Skeleton-of-thought: Large language models can do parallel decoding](#). *arXiv preprint arXiv:2307.15337*.
- Sean O’Brien and Mike Lewis. 2023. Contrastive decoding improves reasoning in large language models. *arXiv preprint arXiv:2309.09117*.
- OpenAI. 2023. [Gpt-4 technical report](#). *ArXiv*, abs/2303.08774.
- Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.
- Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey,

- M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. 2022. **Multi-task prompted training enables zero-shot task generalization**. In *International Conference on Learning Representations*.
- Andrea Santilli, Silvio Severino, Emilian Postolache, Valentino Maiorca, Michele Mancusi, Riccardo Marin, and Emanuele Rodolà. 2023. Accelerating transformer inference for translation via parallel decoding. *arXiv preprint arXiv:2305.10427*.
- Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. 2022. Confident adaptive language modeling. *Advances in Neural Information Processing Systems*, 35:17456–17472.
- Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, et al. 2022. Using deep-speed and megatron to train megatron-turing nlG 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990*.
- Benjamin Spector and Chris Re. 2023. Accelerating llm inference with staged speculative decoding. *arXiv preprint arXiv:2308.04623*.
- Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajwal Bhargava, Shruiti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.
- Mojtaba Valipour, Mehdi Rezagholizadeh, Hossein Rajabzadeh, Marzieh Tahaei, Boxing Chen, and Ali Ghodsi. 2023. Sortednet, a place for every network and every network in its place: Towards a generalized solution for training many-in-one neural networks. *arXiv preprint arXiv:2309.00255*.
- Neeraj Varshney and Chitta Baral. 2022. **Model cascading: Towards jointly improving efficiency and accuracy of NLP systems**. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11007–11021, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Neeraj Varshney and Chitta Baral. 2023. **Post-abstention: Towards reliably re-attempting the abstained instances in QA**. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 967–982, Toronto, Canada. Association for Computational Linguistics.
- Neeraj Varshney, Man Luo, and Chitta Baral. 2022. Can open-domain qa reader utilize external knowledge efficiently like humans? *arXiv preprint arXiv:2211.12707*.
- Neeraj Varshney, Wenlin Yao, Hongming Zhang, Jian-shu Chen, and Dong Yu. 2023. A stitch in time saves nine: Detecting and mitigating hallucinations of llms by validating low-confidence generation. *arXiv preprint arXiv:2307.03987*.
- Peiyi Wang, Lei Li, Liang Chen, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023a. Large language models are not fair evaluators. *arXiv preprint arXiv:2305.17926*.
- Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023b. **Self-instruct: Aligning language models with self-generated instructions**. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.
- Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krma Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. 2022. **Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks**. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5085–5109, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Ziheng Wang, Jeremy Wohlwend, and Tao Lei. 2020. **Structured pruning of large language models**. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6151–6162, Online. Association for Computational Linguistics.
- Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022. **Finetuned language models are zero-shot learners**. In *International Conference on Learning Representations*.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. [SmoothQuant: Accurate and efficient post-training quantization for large language models](#). In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 38087–38099. PMLR.

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

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.

Nan Yang, Tao Ge, Liang Wang, Binxing Jiao, Daxin Jiang, Linjun Yang, Rangan Majumder, and Furu Wei. 2023a. Inference with reference: Lossless acceleration of large language models. *arXiv preprint arXiv:2304.04487*.

Seongjun Yang, Gibbeum Lee, Jaewoong Cho, Dimitris Papailiopoulos, and Kangwook Lee. 2023b. Predictive pipelined decoding: A compute-latency trade-off for exact llm decoding. *arXiv preprint arXiv:2307.05908*.

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. 2022. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35:27168–27183.

Murong Yue, Jie Zhao, Min Zhang, Liang Du, and Ziyu Yao. 2023. Large language model cascades with mixture of thoughts representations for cost-efficient reasoning. *arXiv preprint arXiv:2310.03094*.

Tianhua Zhang, Hongyin Luo, Yung-Sung Chuang, Wei Fang, Luc Gaiatskell, Thomas Hartvigsen, Xixin Wu, Danny Fox, Helen Meng, and James Glass. 2023. Interpretable unified language checking. *arXiv preprint arXiv:2304.03728*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*.

Appendix

A Evaluation Methodology

We use the following prompt with Claude model for comparing the quality of responses:

Human: You are a helpful and precise assistant for checking the quality of the answer.

[Question]
{question}

[The Start of Assistant 1's Answer]
{answer_1}

[The End of Assistant 1's Answer]

[The Start of Assistant 2's Answer]
{answer_2}

[The End of Assistant 2's Answer]

We would like to request your feedback on the performance of two AI assistants in response to the user question displayed above. Please rate the helpfulness, relevance, accuracy, level of details of their responses. Each assistant receives an overall score on a scale of 1 to 10, where a higher score indicates better overall performance. Please output a single line containing only two values indicating the scores for Assistant 1 and 2, respectively. The two scores are separated by a space.

Assistant:

To circumvent the position bias, we evaluate a response pair with both orderings of the responses and then aggregate the judgment scores. We use the Claude API with temperature set to 0 and modelId to ‘anthropic.claude-v1’.

B Additional Results

B.1 Inability of the Intermediate Layers of the Model Tuned with Standard Instruction Tuning to Generate ‘High-Quality’ Text

In this Subsection, we show examples of responses of intermediate layers (fixed exiting) of the model tuned with standard instruction tuning (IT).

Question: What are the differences between plant-based and animal-based protein sources?

Layer 32 (Final): The main difference between plant-based and animal-based protein sources is that plant-based proteins are found in plant-based foods, such as legumes, nuts, and seeds, while animal-based proteins are found in animal-based foods, such as meat, eggs, and fish. Plant-based proteins typically contain fewer calories, fat, and cholesterol than animal-based proteins. Additionally, ...

Layer 28: plant-based proteins sources such as legumes, legumes, and nuts contain essential amino achren needed for growth and repair of cellular structures. animal based proteins sources such as meat, eggs, and fish contain additional nutrients such as vitamin B Complex vitamin D vitamin Iron and calcium needed for optimal health.

Layer 24: plant based prote source protein source diversity, quality proteinsrichness quantityquantityquantityquantity quantity quantity quantity quantity quantity q ...

Layer 20: difference between Plant Based vs animal based protein sourceage oldenhamptonensis proportionalityuniciclesqueque ...

Question: Imagine you are a time traveler from the year 3000. What technological advancements would you tell people about?

Layer 32 (Final): From the year 3000, I would tell people about the advancements in technology that have made life easier and more efficient. I would talk about the autonomous cars, the implanted chips that allow people to access the internet and doctors to monitor patients remotely, the 3D-printed homes and the list goes on. I would also tell people about the advancements in renewable energy and how we are able to generate electricity from the sun and the wind.

Layer 28: From the year 3000 perspective, some technological advances might include flying cars, robotic vac vacuifiers, implanted implants used for implantable devices such as pacemakers, implanted cameras implanted microscopes implanted anything else you can think of implanted implanted implanted ...

Layer 24: techn adv advances techn advances techn advances techn ...

Layer 20:
futureproofing@gmailmail
domaininsights
intoxicatednessipag
eantryonswardwardsonswardsonswardso
...

The above examples and the results in Section 6.1 show that in standard instruction tuning, the intermediate layers do not acquire good generation ability as the generated responses are of poor quality.

In the next subsection, we illustrate examples that show that IT with LITE enables the intermediate layers to acquire good generation ability.

B.2 Intermediate Layers Acquire Good Generation Ability in IT with LITE

Question: What are the differences between plant-based and animal-based protein sources?

Layer 32 (Final): Plant-based protein sources are mainly found in foods such as legumes, nuts, and seeds, while animal-based protein sources are found in foods such as meat, fish, eggs, and dairy products. Plant-based proteins are generally lower in calories and fat, and are rich in vitamins, minerals, and fiber. Animal-based proteins are typically higher in calories and fat, and ...

Layer 28: Plant-based protein sources are mainly found in foods such as legumes, nuts, and seeds, while animal-based protein sources are found in foods such as meat, fish, eggs, and dairy products. Plant-based proteins are generally lower in calories and fat, and are rich in vitamins, minerals, and fiber. Animal-based proteins are typically higher in calories ...

Layer 24: Plant-based protein sources are mainly found in foods such as legumes, nuts, and seeds, and are typically lower in calories and fat than animal-based protein sources. Plant-based proteins are also rich in vitamins, minerals, and antioxidants, and are a good source of ...

Layer 20: Plant-based protein sources are found in many different foods, including legumes, nuts, seeds, and seeds. They are usually lower in calories and fat than animal-based protein sources. Plant-based proteins are also rich in fiber and contain essential vitamins ...

Question: Imagine you are a time traveler from the year 3000. What technological advancements would you tell people about?

Layer 32 (Final): From the year 3000, I would tell people about the incredible advancements in technology that have been made over the centuries. We take many things for granted now, such as the internet and autonomous vehicles, but these technologies were first developed in the ...

Layer 28: From the year 3000, I would tell people about the advancements in artificial intelligence, robotics, and genetic engineering. In particular, I would tell people about the advancements made in AI, such as the development of autonomous cars, virtual assistants, and smart homes. I would ...

Layer 24: From the year 3000, I would tell people about the incredible advancements in technology that have been made over the centuries. We are now able to control our environment by using thought-controlled prostheses, use genetic ...

Layer 20: In the 3000s, we have developed advanced artificial intelligence, self-driving cars, and teleportation of particles. We have also made significant progress in quantum computing, which has allowed us to ...

The above examples illustrate that IT with LITE enables the intermediate layers to acquire good generation ability. This is because the losses are aggregated from these intermediate layers during instruction tuning.

B.3 Poor Quality of Responses With Dynamic Early Exiting from Model Tuned with Standard Instruction Tuning (IT)

Here, we show examples of the responses generated via dynamic confidence-based early exiting from the model tuned with standard instruction tuning (IT). Note that this is without LITE.

It can be observed from these examples that the early exiting responses are of poor quality. They suffer from a number of issues including degeneration such as unwanted token repetitions, and lack of coherence and structure. We note that this can be attributed to the poor generation ability and alignment of the intermediate layers.

Question: What are the most effective ways to deal with stress?

Layer 32 (Final): The most effective ways to deal with stress include identifying sources of stress, developing coping strategies, maintaining a balanced diet, exercising regularly, getting enough sleep, and taking time to relax and do something enjoyable. Taking time to practice deep breathing and meditation, talking to a friend or therapist, and using humor are also helpful.

Dynamic Early Exiting with IT model: cop coping strategies include identifying sources of stress, developing cop coping skills such as cop coping and relaxation techniques such as meditation and med meditation med meditation med meditation med meditation med meditation med meditation med meditation med meditation med meditation ...

Question: Can you explain the basics of quantum computing?

Layer 32 (Final): Quantum computing is a form of computing that uses quantum-mechanical phenomena, such as superposition and entanglement, to perform operations on data. It enables computation at a much faster rate than traditional computers and can solve problems that are intractable on classical computers. Quantum computing uses qubits²⁰¹⁴ quantum bits²⁰¹⁴ which can be in a state of 0 or 1, instead of the traditional bit of 0 or 1. Qubits can be used to represent data and perform operations on it, such as addition, subtraction, and multiplication.

Dynamic Early Exiting with IT model: quantum computing basics quantum computing basics explain the basics of quantum computing including what it is, how it works, why it matter, how it can be used for computing, and what its future may hold. quantum computing basics quantum computing basics explain the basics of quantum computing including what it is, how it works, why it matter, how it can be used for computing ...

This further showcases the importance of LITE during instruction tuning as it allows the intermediate layers to acquire good generation ability, thus enabling opportunities for early exiting without compromising the quality of the generation.

B.4 Examples of Responses with Dynamic Early Exiting for Model instruction tuned with LITE

Here, we show examples of responses of the final layer and the dynamic early exiting method with the model instruction tuned using LITE.

Question: What are the main differences between Python and JavaScript programming languages?

Layer 32 (Final): The main differences between Python and JavaScript programming languages are their syntax, data types, data structures, and logic. Python is a high-level, interpreted language that is focused on readability and simplicity, while JavaScript is a low-level, interpreted language that is focused on adding interactive elements to web pages. Python has strong support for data typing, while JavaScript has weaker support for data types. Python is more suitable for creating scripts and applications, while JavaScript is more suitable for creating interactive web pages.

Dynamic Early Exiting: The main differences between Python and JavaScript programming languages are their syntax, data types, data structures, and logic. Python is a high-level, interpreted language that is focused on readability and simplicity, while JavaScript is a low-level, interpreted language that is focused on adding interactive features to web pages. Python has dynamic data types, while JavaScript has static data types. Python is more suitable for creating scripts and applications, while JavaScript is more suitable for creating interactive web pages.

Question: What if Isaac Newton had focused on biology instead of physics?

Layer 32 (Final): Isaac Newton had focused on biology instead of physics, he would have likely made similar discoveries in that field as he did in physics. His laws of motion and universal gravitation could have been formulated in a biological context, for example, describing the ways in which cells and organisms adhere to physical laws. He would have likely realized that biology is also an essential physical science, and that the laws of motion are just as universal in biology as in physics.

Dynamic Early Exiting: Isaac Newton had focused on biology instead of physics, he would have developed a different understanding of the natural world. He would have believed that biology was the foundation of all physics, and that it was the only way to understand the universe. He would have also believed that all living organisms had the same underlying laws of physics, rather than the laws of physics being created for humans to understand. He would have also believed that biology was the most important branch of science, and that it was the only way to understand the universe.

These examples show that the responses obtained via early exiting do not compromise the quality of the generation. Furthermore, responses have high syntactic and semantic overlap.

B.5 Quality and Inference Cost Analysis at Category Level

Vicuna and WizardLM test sets also provide the category corresponding to different test instances. To this end, we present category-level quality and inference cost results for these datasets.

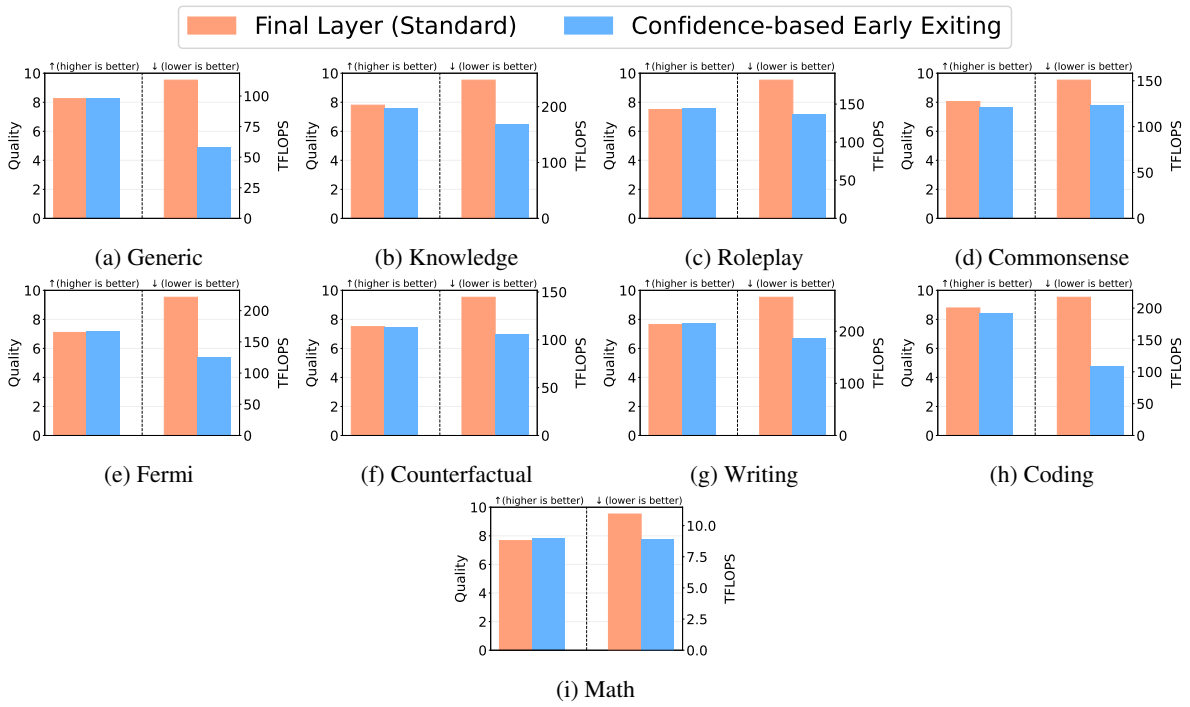


Figure 9: Comparing the quality of responses and the inference cost of the standard generation method with the dynamic early exiting method on different categories of the Vicuna Test set.

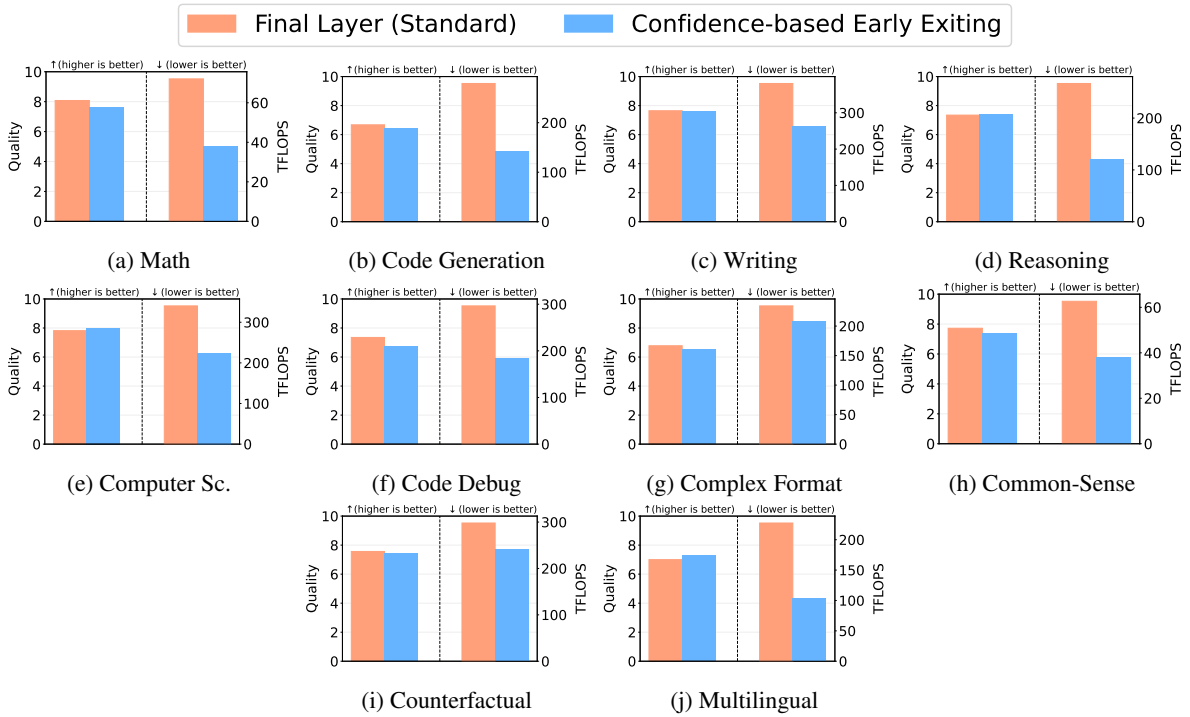


Figure 10: Comparing the quality of responses and the inference cost of the standard generation method with the dynamic early exiting method on different categories of the WizardLM Test set.

Vicuna: Figure 9 compares the quality of responses and the inference cost of the standard generation method (final layer) with the dynamic early exiting method for different categories of Vicuna test set. On average, it results in cost improvement of 33.39%. It can be observed that the approach consistently achieves efficiency improvement in all the categories which demonstrates the generality of the approach.

WizardLM: Figure 10 compares the quality of responses and the inference cost of the standard generation method (final layer) with the dynamic early exiting method for different categories of WizardLM test set. On average, it results in cost improvement of 36.12%.

B.6 Relationship Between Token Prediction Confidence and Percentage Alignment of the Intermediate Layers for the Model Tuned with Instruction Tuning (IT)

Figure 11 shows the relationship between the token prediction confidence of the intermediate layers and the percentage alignment with the token prediction of the final layer for standard instruction tuning (IT). It shows that the confidence is not well correlated with the percentage alignment. However, in IT with LITE (Figure 6), the intermediate layers’ token prediction probabilities provide a strong signal of alignment.

B.7 Dynamic Confidence-Based Early Exiting with Aggressive Confidence Thresholds

We also experiment with aggressive confidence thresholds. Specifically, we use the following confidence thresholds: Layer 8: 0.85, Layer 12: 0.85, Layer 16: 0.8, Layer 20: 0.8, Layer 24: 0.7, and Layer 28: 0.6. These thresholds are lower than those used in the main paper. Figure 12 shows the quality and cost comparisons. It leads to larger cost improvements (of 49.92%) though it slightly drops the quality of generation (by 5.34%).

B.8 Results for 13B Model

For the 13B model, we use the following confidence thresholds: Layer 8: 0.95, Layer 12: 0.95, Layer 16: 0.9, Layer 20: 0.9, Layer 24: 0.8, Layer 28: 0.7, Layer 32: 0.7, and Layer 36: 0.65,

Table 3 shows the cost improvements resulting from dynamic early exiting from the 13B model on each test dataset. On average, it results in 46.35% cost improvement. This improvement is higher

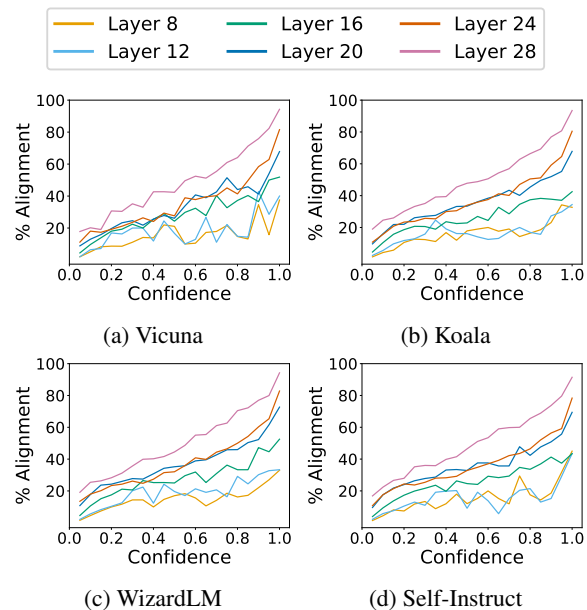


Figure 11: Demonstrating relationship between token prediction confidence of the intermediate layers and the percentage alignment with the token prediction of the final layer for model tuned with IT.

Test Dataset	Cost Improvement (%)
Vicuna	43.60 %
Koala	45.62 %
WizardLM	50.84 %
Self Instruct	45.35 %

Table 3: Percentage improvements in the inference cost (measured in FLOPs) with dynamic early exiting for the 13B model. On average, it achieves an improvement of 46.35%.

than the improvement achieved in the case of the 7B model (37.86%).

C Weighted LITE

We also experiment using increasing weights for different intermediate layers during the loss aggregation. We use increasing weights as the later layers have more capacity to learn. Specifically, for the 7B model where we select layer numbers 8, 12, ..., 28, and 32, we use the following weights: 1, 2, ..., 7.

In Figure 13, we plot the percentage alignment of token predictions of all intermediate layers with the token predictions of the final layer. The figure shows the percentage alignment of (i) the model tuned using standard IT (orange) and (ii) the model tuned using IT with weighted LITE (blue). In Fig-

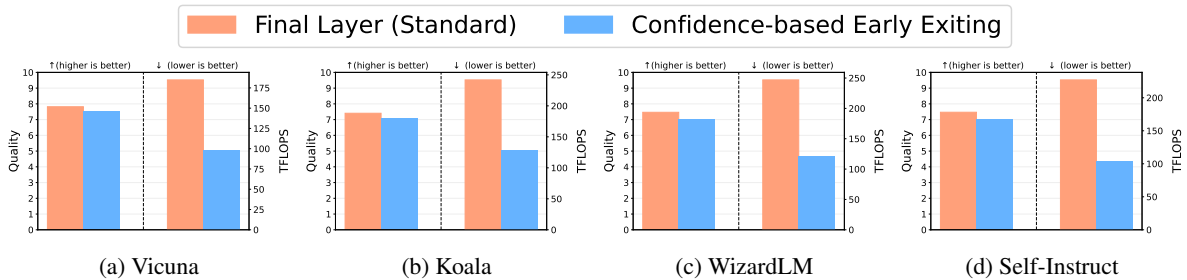


Figure 12: Comparing the quality of responses (evaluated using the Claude model) and the inference cost (measured in FLOPs) of the standard generation method from the final layer with the dynamic early exiting method. Confidence Thresholds: Layer 8: 0.85, Layer 12: 0.85, Layer 16: 0.8, Layer 20: 0.8, Layer 24: 0.7, and Layer 28: 0.6. This **aggressive configuration** results in larger cost improvements of 49.93% but results in a slight degradation in the generation quality.

Test Set	# Samples
Vicuna	80
Koala	180
WizardLM	218
Self Instruct	252

Table 4: Statistics of evaluation datasets.

Id.	Final Layer (Standard)	Confidence-based Early Exiting
1	7.25	7.2375
2	7.725	7.6125
3	7.5625	7.425
Avg.	7.51	7.425

Table 5: Human annotations for the quality of responses of the 7B model on the Vicuna test dataset.

ure 14, we compare the alignment for the model tuned using IT with LITE and the model tuned using IT with weighted LITE. It can be observed that assigning lower weight to the initial layers results in just a slight reduction in alignment percentage.

We also plot the relationship between the token prediction confidence (softmax over the logits of the LM head) of the intermediate layers and the percentage alignment with the token prediction of the final layer. Figure 15 shows this plot for the model tuned with weighted LITE.

D Additional Evaluations

In the paper, we have presented LLM-based evaluations. To further support these evaluations, in this section, we present human evaluations and GPT-4 based evaluations. In human evaluations, we asked the participants to annotate the quality of the responses. We conducted this study on the Vicuna test dataset and for the 7B model. Specifically, we asked 3 participants to rate the quality of responses on a scale of 1 to 10, where a higher score indicates better overall performance (similar to the LLM-based evaluations). All the participants were graduate students. Table 5 shows the ratings of the individual participants and the aggregated scores. Additionally, we also use GPT-4 for evaluations. The scores corresponding to ‘Final Layer

(Standard)’ and ‘Confidence-based Early Exiting’ are 6.3 and 6.17 respectively.

From both these studies, it can be inferred that the quality ratings are indeed very close and thus support our LLM-based evaluations.

E Comparison with Early Exiting Baselines

Motivated by the techniques presented in (Fan et al., 2020), we develop several early exiting baseline techniques. Specifically, we implement two variants of random early exiting technique and a sequential exiting technique. We note these are just other early exiting baseline techniques and experimenting with them can demonstrate the sheer effectiveness of our dynamic confidence based early exiting.

In the random early exiting technique, instead of looking at the confidence (as we do in the dynamic confidence based early exiting), we randomly make a decision to exit from a ‘check-point’ layer. In one variant of this technique, at each checkpoint layer, we make a binary random decision to exit or continue (therefore two choices). In another variant, we reduce this chance value to $(1/\#\text{checkpoint layers})$.

In the sequential early exiting technique, we

cyclically change the exiting layer for the next token. For example, for the first token, we use the first checkpoint layer (8th layer), for the second token, we use the second checkpoint layer (12th layer) and so on in a cyclic manner. We present the results of these baselines on the vicuna test dataset in Table 6.

From these results, it can be observed that the dynamic confidence based early exiting method results in just a minimal drop in quality of generation while the other baselines (despite using the intermediate layers of the model tuned with LITE) result in a considerable drop in quality of generation.

In addition, for standard instruction tuning (without LITE), we have already shown that performing early exiting results in very poor quality of generation (Appendix B.3).

F Comparing Latency

We study latency improvements on the Vicuna test dataset using 7B model. We find that the standard generation with the final layer requires 14912.12 ms on average while dynamic confidence based early exiting method takes 9873.20 ms on average. This implies that the early exiting method takes up just 66.2% latency of the standard generation process while maintaining the generation quality. This result supports the FLOP improvements.

We have focused primarily on the FLOP values because the latency values vary (though marginally) on different executions, unlike the FLOP values. Furthermore, the follow-up works may not be able to precisely compare the latency values in an apples-to-apples comparison. Thus, FLOP values more clearly and reliably show the impact of the methodology and can also facilitate future work.

G Design Decisions

FLOPs for measuring Computational Cost: We note that we use FLOPs as the metric of showcasing inference efficiency improvements because it is hardware independent, unlike latency.

KV Caching: We explore the dynamic exiting method for inference without KV caching. This is because the representations of the layers after the exiting layer are not computed in this method and thus will not be available in the cache for the next token prediction if the model exits from a higher layer than the previous token prediction.

LLaMA-2 Models for Experiments: We experiment with LLaMA-2 models as they are publicly

available and widely used for LLM research.

Evaluation Datasets To perform holistic evaluation, we experiment with four different human-instruction test sets including Vicuna, Self-Instruct, Koala, and WizardLM. We select these evaluation test sets as they can together cover a large number and types of instructions thus resulting in a comprehensive evaluation. Table 4 shows the statistics of the datasets.

H Extended Related and Concurrent Work

Yang et al. (2023b) propose an approach called Predictive Pipelined Decoding (PPD) that focuses on lowering the latency by utilizing additional compute resources. Specifically, it accelerates the decoding by parallelizing processes, each of which starts to decode from the top-k predicted tokens of the specific transformer layer. Simultaneously, the main process continues to compute the output of the final layer and predicts the next token.

Yang et al. (2023a) propose an inference-with-reference decoding method that exploits the overlap between an LLM’s output and available reference. Specifically, it first selects a text span from the reference and copies its tokens to the LLM decoder, and then checks if they are acceptable based on the output token probabilities. He et al. (2023) also uses retriever to generate the draft tokens in speculative decoding.

Jiang et al. (2023) propose to compress the prompts to accelerate the inference. Santilli et al. (2023) propose a parallel decoding strategy for translation using Jacobi and Gauss-Seidel fixed-point iteration methods

The most similar yet non-trivially different from our work is a concurrent work (Kavehzadeh et al., 2023) which is motivated from SortedNet (Valipour et al., 2023) that focuses on leveraging the inherent modularity of models to construct sub-models with varying computational loads. In contrast, in our work, we instruction tune with LITE and then perform token-level dynamic confidence based early exiting to achieve efficiency benefits while maintaining the generation quality. Our approach is motivated from our two findings of the model tuned using IT with LITE which are (a) LITE greatly aligns the intermediate layers’ token prediction with that of the final layer and (b) the intermediate layers’ token prediction probabilities provide a strong signal of this alignment.

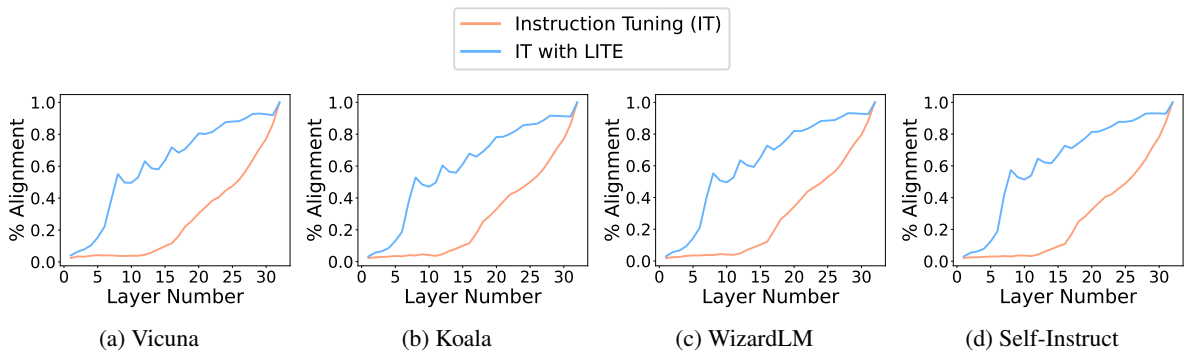


Figure 13: Comparing percentage ‘alignment’ of intermediate layer token predictions with the token predictions of the final layer for the model tuned using IT (orange) and the model tuned using IT with weighted LITE (blue).

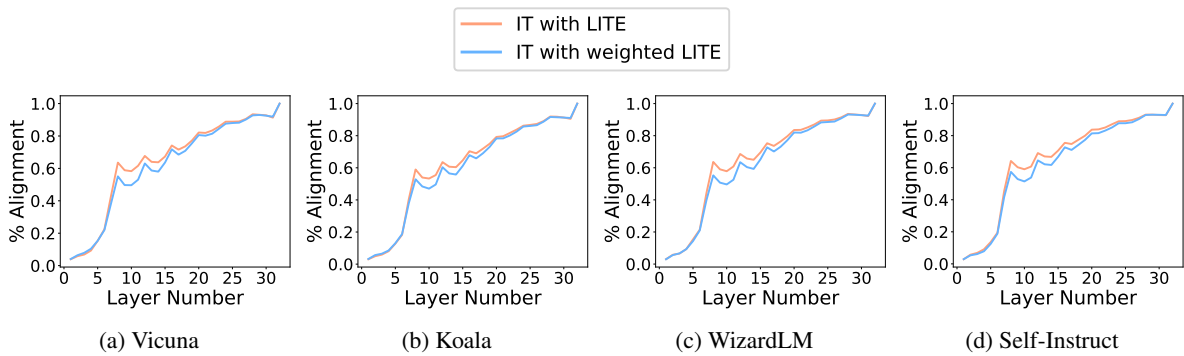


Figure 14: Comparing percentage ‘alignment’ of intermediate layer token predictions with the token predictions of the final layer for the model tuned using IT with LITE (orange) and the model tuned using IT with weighted LITE (blue).

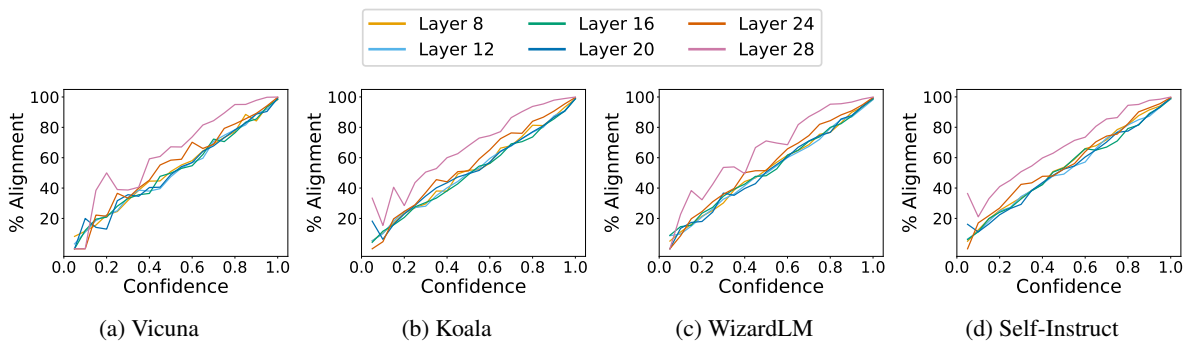


Figure 15: Demonstrating trend of token prediction confidence of the intermediate layers and the % alignment with the final layer for model tuned with weighted LITE.

Method	Quality	Cost (TFLOPs)
Standard (Final Layer)	7.78	184
Dynamic Confidence based Early Exiting	7.72	123
Random Baseline (first variant)	5.93	69
Random Baseline (second variant)	6.91	129
Sequential Baseline	6.97	124

Table 6: Performance comparison of dynamic confidence based early exiting with other early exiting baselines described in Section E.

I Discussion on Other Avenues of Research using Intermediate Layer Decoding

with the final layer or using the intermediate layers to generate the complete output and then checking the information consistency.

I.1 Speculative Sampling

In speculative sampling (Leviathan et al., 2023; Chen et al., 2023; Ning et al., 2023; Kim et al., 2023; Spector and Re, 2023), a short draft of K tokens is first generated from a smaller auto-regressive model and then the draft is scored using the target model. Using a rejection sampling scheme, a subset of the K draft tokens is accepted by sequentially checking from left to right, and thus in this process, the distribution of the target model is recovered for the accepted tokens. The efficiency in this technique comes from ‘producing’ more than one token (on average) from the target model in a single pass.

This technique requires an additional drafting model. However, we showed that instruction tuning with LITE enables the intermediate layers to acquire ‘good’ generation ability. Thus, the intermediate layer(s) of the same model can be used as the drafting model while the last layer remains to be the target model. This circumvents the requirement of maintaining a separate drafting model for speculative sampling.

I.2 Hallucination Detection

Addressing the hallucination problem of LLMs is an important research direction and a number of methods have been developed (Manakul et al., 2023; Azaria and Mitchell, 2023; Zhang et al., 2023; Dhuliawala et al., 2023; Varshney et al., 2023; Gou et al., 2023). Sampling based methods require generating multiple samples and then checking the information consistency between them. Enabling the intermediate layers with the generation ability equips us with multiple opportunities, such as checking the consistency based on the alignment percentage of the intermediate layers