Not All Layers of LLMs Are Necessary During Inference

Siqi Fan[1], Xin Jiang[2], Xiang Li[2], Xuying Meng[3], Peng Han[1], Shuo Shang[1*], Aixin Sun[4], Yequan Wang[2*]

[1]University of Electronic Science and Technology of China, Chengdu, China
[2]Beijing Academy of Artificial Intelligence, Beijing, China
[3]Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[4]School of Computer Science and Engineering, Nanyang Technological University, Singapore
{sqfann, tshwangyequan, jedi.shang}@gmail.com

Abstract

Due to the large number of parameters, the inference phase of Large Language Models (LLMs) is resource-intensive. However, not all requests posed to LLMs are equally difficult to handle. Through analysis, we show that for some tasks, LLMs can achieve results comparable to the final output at some intermediate layers. That is, *not all layers of LLMs are necessary during inference*. If we can predict at which layer the inferred results match the final results (produced by evaluating all layers), we could significantly reduce the inference cost. To this end, we propose a simple yet effective algorithm named **AdaInfer** to adaptively terminate the inference process for an input instance. AdaInfer relies on easily obtainable statistical features and classic classifiers like SVM. Experiments on well-known LLMs like the Llama2 series and OPT, show that AdaInfer can achieve an average of 17.8% pruning ratio, and up to 43% on sentiment tasks, with nearly no performance drop ($<$1%). Because AdaInfer does not alter LLM parameters, the LLMs incorporated with AdaInfer maintain generalizability across tasks.

1 Introduction

LLMs have demonstrated impressive performance on various downstream tasks using evaluation protocols such as zero-shot, few-shot, and fine-tuning. Example applications include text generation, question answering, and sentiment analysis. Notably, the in-context learning ability allows LLMs to adapt to various different tasks using input-output examples without parameter updates. However, the inference phases of LLMs are typically very expensive due to their large number of parameters [Pope *et al.*, 2023; Liu *et al.*, 2023]. Specifically, the inference time complexity for typical large models with a Transformer structure is $LSd(d + S)$ per single inference, where L, S, and d represent the number of layers, sequence length, and hidden size, respectively [Touvron *et al.*, 2023].

Existing solutions to achieve more efficient inference in LLMs include model pruning [Ma *et al.*, 2023; Kim *et al.*, 2024] and sparse models [LeCun *et al.*, 1989; Liu *et al.*, 2023]. Both solutions alter LLM parameters and may risk compromising generalization ability. Additionally, different LLM designs pose compatibility challenges with other acceleration methods. Hence, an ideal efficient LLM inference should use fewer computational resources while maintaining generalization and in-context learning abilities [Liu *et al.*, 2023].

If we draw an analogy between LLM inference and the human thinking process [Salthouse, 1996; Deary *et al.*, 2001], where simple questions can be answered quickly and complex questions require more time for reasoning, we may expect LLMs not to use the same inference power to handle all tasks. Previous work [Teerapittayanon *et al.*, 2016; Huang *et al.*, 2017] show that "easy" tasks activate at shallower layers while "hard" ones activate at deeper layers. For LLM training, a growth strategy [Li *et al.*, 2023] adds parameters in stages to reduce the overall training cost, *i.e.,* not all training instances use the same set of parameters. Hence, we consider that adjusting the parameters during inference based on the difficulty level of a task may be an effective way for efficient inference.

To this end, we conduct a statistical analysis to examine the correlation between the results obtained in intermediate layers and those in the final layers across various tasks. We made two observations: (i) not all layers of LLMs are necessary during inference, *i.e.,* early stopping works, and (ii) simpler tasks require fewer layers, while more complex tasks require more layers of inference. The key to achieving efficient LLM inference then becomes *when to stop the inference process adaptively based on the input instance*. Interestingly, exploring adaptive inference may bridge LLMs with the brain's information processing [Hubel and Wiesel, 1962; Murata *et al.*, 2000], aiding in the analysis of activated network modules during sample processing [Han *et al.*, 2021] and identifying crucial input components that affect the final prediction.

In this paper, we present AdaInfer, a simple yet effective algorithm for instance-aware adaptive inference. The core of AdaInfer lies in data-driven decision-making. We begin by performing a statistical analysis on each block feature of LLMs, such as logits, hidden states, mlp, and attention activation values. Consequently, we choose logits to construct

[*] Corresponding authors.

features and employ classical statistical classifiers, SVM and CRF, to predict the optimal layer at which to stop the inference. Experiments on well-known LLMs (*i.e.,* Llama2 series and OPT) show that AdaInfer can achieve an average of 17.8% pruning ratio, and up to 43% on sentiment tasks, with nearly no performance drop (<1%). The cost of collecting the small set of statistical features and running AdaInfer is negligible compared to the cost of LLM inference.

AdaInfer is an early stop strategy that optimizes efficiency without altering the model's parameters. Therefore, AdaInfer does not affect the model's generalization and in-context learning abilities. Furthermore, being orthogonal to other model acceleration techniques, AdaInfer offers the potential for further enhancing inference efficiency.

2 Related Work

We study how to leverage the representation of LLM to achieve adaptive inference. Our problem setting and methods are closely connected to many existing research areas.

Dynamic depth. This involves two methods: *Early Exit (EE)* and *Skip Layer*. *EE* first appeared in CNN/DNN networks for visual tasks [Bolukbasi *et al.*, 2017; Huang *et al.*, 2017; Teerapittayanon *et al.*, 2016]. Subsequently, it was utilized to accelerate the inference of encoder-only architectures in BERT [Li *et al.*, 2020; Liu *et al.*, 2020; Li *et al.*, 2021]. Recently, [Schuster *et al.*, 2022; Varshney *et al.*, 2023] discussed confidence-based *EE* for LM adaptive inference. Our proposed AdaInfer closely aligns with the *EE* concept. We apply *EE* to mainstream decoder-only LLMs, which adhere to the scaling law but suffer from high inference costs due to their large parameter count. *Skip Layer* dynamically omits the execution of middle layers (or modules) for an input token, facilitated by a binary router [Zeng *et al.*, 2023; Raposo *et al.*, 2024], or layer pruning [Kim *et al.*, 2024; Men *et al.*, 2024; Ma *et al.*, 2023]. The main difference between our method and theirs is that we achieve instance-wise inference (*i.e.,* dynamic pruning ratio tailored to specific tasks) without altering the model parameters, which is crucial for current LLMs. To the best of our knowledge, this is the first attempt to discover that each block's logits are crucial elements for *EE* classifiers in LLMs, and we incorporate this insight as a fundamental design choice in AdaInfer.

Speculative decoding. Unlike traditional autoregressive decoding, which generates tokens one by one, speculative decoding uses a smaller, faster "draft" model to predict multiple tokens at once [Leviathan *et al.*, 2023; Chen *et al.*, 2023]. In contrast, our approach only requires one target model. Self-speculative decoding [Zhang *et al.*, 2024; Elhoushi *et al.*, 2024] employs a pruned target model as the draft model. Our approach differs in layer selection strategy (probability features vs. multiple Bayesian searches) and focus (emphasizing the initial generation stage, while they concentrate on the generate stage). This distinction is further highlighted in our experiments.

Knowledge elicitation and honesty. The goal of it is to elicit latent knowledge from a superhuman machine learning model even under worst case assumptions [Christiano *et al.*, 2022]. Techniques like the logit lens and tuned lens [Belrose *et al.*, 2023; Nostalgebraist, 2020] trace the *prediction trajectory* of intermediate layers by converting hidden states into distributions over the vocabulary. Our work builds on these lens techniques and can be seen as their practical application—adaptive inference.

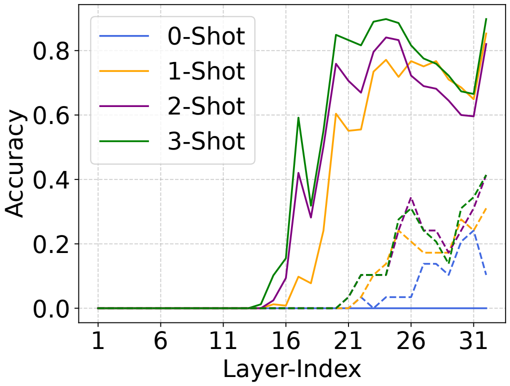


Figure 1: Accuracies obtained by inference at each decoder layer with the Llama2-7B model (32 layers). The *solid line* represents the sentiment analysis task, and the *dashed line* represents the MMLU task.

3 Efficiency Analysis of LLM Inference

Before presenting the statistical observations and insights on LLM inference, we first briefly review LLM's critical components.

3.1 Preliminary: LLM Building Blocks

Modern LLMs, rooted in the Transformer architecture [Vaswani *et al.*, 2017], can be trained with various unsupervised training objectives. In this paper, we focus on mainstream LLMs like GPT and the Llama series. These models are built with a decoder-only structure and are pre-trained with a full language modeling objective, computing loss on all tokens. Their key components can be broken down into the following blocks: *Tokenizer and Embedding Layer*, *Decoder Block*, and *Classification Layer*. The tokenization and embedding layer converts input text into numerical vectors h_0, enabling effective processing and analysis of textual data. The decoder block F processes h_0 through self-attention and feedforward neural networks, allowing the model to focus on the most relevant parts of the input. Lastly, the classification layer, or the LM head layer, maps decoder output h_L into a vocabulary-wide probability distribution $\mathbf{p}$ to facilitate word prediction. These blocks facilitate LLMs in efficiently handling NLP downstream tasks, with a primary emphasis on the decoder block.

During inference, each input instance passes through the decoder block F, layer by layer, until the last layer:

$$\mathbf{h}_{\ell+1} = \mathbf{h}_\ell + F_\ell(\mathbf{h}_\ell), \ell \in \{1, \ldots, L\} \quad (1)$$

where $h_\ell \in \mathbb{R}^d$ is the hidden state at block ℓ. Hence, the inference complexity is linearly related to the number of decoder layers L in the decoder block. The decoder block of earlier models typically comprised 6 layers, whereas current open-source models have many more. For example, Llama2-7B has 32 layers and Llama2-13B features 40 decoder layers [Touvron *et al.*, 2023].

3.2 Not all Layers are Necessary

To explore the possibility of skipping some intermediate layers during inference, we conduct experiments on two tasks: sentiment analysis [Socher *et al.*, 2013] and MMLU [Hendrycks *et al.*, 2021]. We examine the accuracies obtained by stopping inference at each decoding layer and compare them with the final results, *i.e.,* without stopping inference. The experiments were conducted on both Llama2-7B (32 layers) and Llama2-13B (40 layers), and the same observations hold.

Observation 1. *Not all layers of LLMs are necessary during inference: Early Stopping works.*

Using the SST-2 dataset [Socher *et al.*, 2013], we conduct sentiment classification experiments on the Llama2-13B (40 layers) model. We perform inference at each layer with a batch size of 1 and record the results. On average, an early exit at layer 21 (with a variance of 5.1) achieves comparable accuracy to the final layer output. Interestingly, simpler inputs like ‘I like Camera A’ activate only 18 layers, while more complex inputs like ‘Camera A is better than Camera B in picture quality’ activate about 24 layers. Early stop works on the Llama2-7B model as well.

Observation 2. *Simpler tasks require fewer layers for inference, while complex tasks go deeper.*

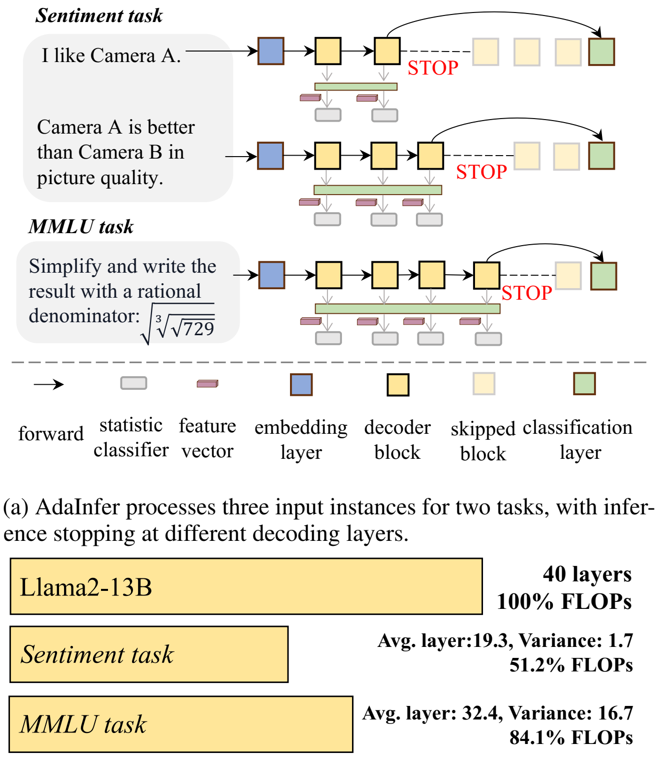
Figure 1 plots the accuracies by stopping inference at different decoding layers on a Llama2-7B. For the task of sentiment analysis, the accuracy matches that of the final layer by the 24th layer, represented by solid lines in the figure. For MMLU, a complex task, accuracy tends to improve with deeper layers. A similar trend holds across all four tested settings, from 0-shot to 3-shot learning.

Insight. Both observations are intuitive and, in fact, not new. Similar findings have been made in visual tasks with convolutional neural networks [Teerapittayanon *et al.*, 2016; Huang *et al.*, 2017] and sentence classification with BERT [Liu *et al.*, 2020]. We extend these observations to decoder-only LLM inferences.

Based on the two observations, we understand that (i) early stopping works, allowing us to reduce inference costs by stopping at certain decoding layers without compromising model accuracy, and (ii) the number of optimal decoding layers for inference is instance-dependent. The number of optimal decoding layers varies across tasks and even across instances of the same task. Recall the two example sentences for sentiment analysis discussed in Observatoin 1. This means that the layer at which inference stops must be dynamically determined (or predicted) for each input instance.

4 AdaInfer: Adaptive Inferences

Modifying LLM parameters may require additional training and pose a potential risk of compromising the model’s generalization capabilities. In designing AdaInfer, we embrace a cost-effective approach that preserves the model’s innate abilities without altering its parameters. The main idea is to capture signals at each decoding layer and make predictions on whether to stop the inference at the current layer.



(a) AdaInfer processes three input instances for two tasks, with inference stopping at different decoding layers.

(b) Effectiveness in reducing computational costs with early stopping during inference.

Figure 2: An illustration of AdaInfer’s processing and computational savings.

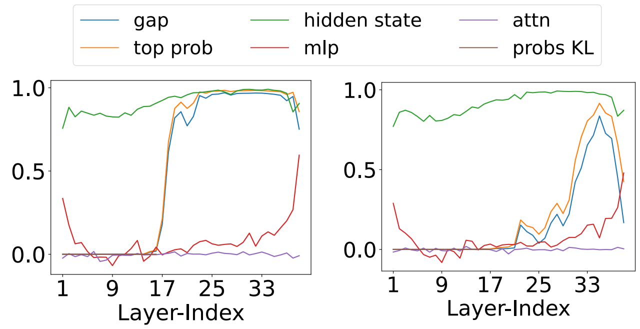
The workflow of AdaInfer is depicted in Figure 2a with three example input instances. At each decoding layer, a *Feature Selection* component crafts a feature vector for the current input instance. A binary *Classifier* then predicts whether to stop the inference, *i.e.,* bypass subsequent decoder layers.

4.1 Feature Selection

LLMs capture coarse-grained features in their initial layers and develop more detailed, fine-grained representations in deeper layers [Nostalgebraist, 2020; Belrose *et al.*, 2023]. This process is facilitated by the repeated application of multi-head attention mechanisms and the use of residual connections. However, there is a lack of features to demonstrate at which stage the representation is sufficient for the current task. Furthermore, these features need to be inherently universal to ensure compatibility across various LLMs.

As a part of feature engineering, we conduct a visual analysis of diverse features from each decoding layer (or decoding block illustrated in Figure 2a) of LLMs. Our examination focused specifically on:

- **Gap**: Measures the prediction confidence of the current block ℓ for the next token. It is defined as $P_\ell(\text{top token}) - P_\ell(\text{second token})$, where P_ℓ represents



(a) Sentiment Task

(b) MMLU Task

Figure 3: Changes of feature values along the 40 decoding layers in Llama2-13B model.

the probability distribution produced by block ℓ: $P_\ell = \text{softmax}(\text{LM head}(h_\ell))$, $P_\ell \in \mathbb{R}^{|V|}$, with $|V|$ denoting the vocabulary size.

- **Top Prob**: The probability estimate $P_\ell(\text{top token})$ for the most likely next token according to the current block.
- **Cosine Similarities**: Cosine similarity scores between the current and previous blocks, calculated for attention values (attn), multi-layer perceptron outputs (mlp), and hidden states.
- **Probs KL**: The KL divergence between the probability distributions P_ℓ and $P_{\ell-1}$.

Again, we use the sentiment and MMLU tasks on the Llama2-13B (40 layers) model for feature analysis, shown in Figure 3. Observe the following trends: (1) across tasks, both “gap” and “top prob” gradually increase alone the inference phase, stabilizing in the deeper layers. (2) The activation of “gap” and “top prob” varies across layers for different tasks. These phenomenons are also evident in the Llama2-7B, OPT-13B [Zhang *et al.*, 2022], and GPT-J [Wang and Komatsuzaki, 2021]. The feature analysis suggests that **“gap” and “top prob” can serve as universal features** for the inference-stopping signal. Notably, these two values remain consistent across two diverse tasks, indicating a versatile discriminating power applicable to various tasks. Factor studies in subsequent experiments also show that other features (*e.g.,* cosine similarities) exhibit subtle differences across layers.

4.2 Classifier

The classifier determines if the signal is compelling enough to warrant an early termination of the process. There are many choices for classifiers, ranging from rule-based classifiers [Huang *et al.*, 2017; Yang *et al.*, 2020; Wang *et al.*, 2022] to gating functions [Lin *et al.*, 2017; Bejnordi *et al.*, 2019]. In our context, classical statistical classification methods are a good option due to their efficiency and their ability to handle simple input features (*i.e.,* “gap” and “top prob”) for a binary classification task.

Given one instance, we obtain the feature vector x_d using the feature selection module. If the current layer's output $\hat{y}$ provides the correct answer y, the associated label y_c is a positive example; otherwise, it is a negative example. For LLMs trained to predict the next token, if the next token $\hat{y}$ predicted based on an intermediate decoding layer's output is the same as the token y predicted by the last decoding layer's output, then the layer's label $y_c = 1$.

Model	Params	Tokens	Layer Num.
EleutherAI/GPT-J	6B	0.4T	28
Meta/OPT	13B	0.18T	40
Meta/Llama 2	7B	2T	32
Meta/Llama 2	13B	2T	40
Meta/Llama 2	70B	2T	80

Table 1: Overview of the LLMs used in experiments with AdaInfer.

$$y_c = \begin{cases} 1 & \text{if } \hat{y} = y, \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

Thus, for an L-layer LLM, each input instance x yields L pairs of $\langle x^d, y_c \rangle$. The details of creating training data for the classifier are provided. In our implementation, we consider two types of classifiers: Support Vector Machines (SVM) [Hearst *et al.*, 1998] and Conditional Random Fields (CRF) [Lafferty *et al.*, 2001]. SVM does not rely on the context of sequences, while CRF incorporates sequence modeling along the decoding layers.

5 Experiments

We now conduct experiments with AdaInfer on well-known LLMs across various tasks. Specifically, we evaluate the zero/few-shot learning capabilities, with two primary types of tasks.

Question Answering Tasks. (1) MMLU [Hendrycks *et al.*, 2021] encompasses 57 tasks across humanities, social sciences, STEM, and more, requiring world knowledge and problem-solving capabilities. (2) CommonsenseQA [Talmor *et al.*,] tests for commonsense knowledge through multiple-choice questions. (3) SQuAD [Rajpurkar *et al.*, 2016] serves as a reading comprehension benchmark, with questions based on Wikipedia articles and answers either segments of passage or marked as unanswerable.

Text Classification Tasks. (1) SST-2 [Socher *et al.*, 2013] involves sentiment analysis of movie reviews with binary “positive” or “negative” labels. (2) AG News [Zhang *et al.*, 2015] classifies news headlines and article sentences into Business, Science/Technology, Sports, and World categories.

5.1 Experiment Settings

Large Language Models. We select well-established large language models as the backbone for AdaInfer. Specifically, we utilize OPT [Zhang *et al.*, 2022] and the Llama 2 series [Touvron *et al.*, 2023] from Meta, along with GPT-J [Wang and Komatsuzaki, 2021] from EleutherAI. Detailed information about these models is provided in Table 1. These models vary in terms of the number of parameters, ranging from 6B to 70B, and the number of layers, ranging from 28 layers to 80 layers. These models exhibit subtle differences in architectural design and training data volume.

Method	P. Ratio(↑)	MMLU			CommonsenseQA			SQuAD			Sentiment			AG News		
		Acc(↑)	#Avg.L(↓)	Var	Acc(↑)	#Avg.L(↓)	Var	Acc(↑)	#Avg.L(↓)	Var	Acc(↑)	#Avg.L(↓)	Var	Acc(↑)	#Avg.L(↓)	Var
Llama2 7B (32 Layers)																
Dense	0	43.05	32	–	**53.50**	32	–	**48.08**	32	–	95.20	32	–	79.65	32	–
ShortGPT$_p$	28.13	21.52	23	–	33.52	23	–	10.60	23	–	93.48	23	–	56.90	23	–
ShortGPT$_5$	15.63	29.95	27	–	41.90	27	–	12.97	27	–	90.40	27	–	53.25	27	–
ShortGPT$_3$	9.38	37.39	29	–	53.22	29	–	14.32	29	–	94.17	29	–	71.28	29	–
AdaInfer	9.66 → 35.71	**43.73**	28.91	4.97	53.00	27.90	5.93	45.82	26.77	11.88	**95.30**	20.57	5.10	**79.72**	29.20	2.70
Llama2 13B (40 Layers)																
Dense	0	**53.31**	40	–	64.92	40	–	**52.90**	40	–	**95.90**	40	–	**77.53**	40	–
ShortGPT$_p$	25	45.12	30	–	**65.00**	30	–	13.32	30	–	84.38	30	–	55.90	30	–
ShortGPT$_5$	12.50	46.64	35	–	64.45	35	–	16.35	35	–	89.80	35	–	70.17	35	–
ShortGPT$_3$	7.50	47.22	37	–	64.47	37	–	17.25	37	–	**95.90**	37	–	75.47	37	–
AdaInfer	9.13 → 43.33	52.44	36.35	8.15	62.48	34.60	10.20	48.35	31.18	31.75	92.65	22.67	8.10	76.43	34.02	24.18
OPT 13B (40 Layers)																
Dense	0	**23.60**	40	–	21.45	40	–	**26.12**	40	–	92.58	40	–	**72.83**	40	–
ShortGPT$_p$	25	10.17	30	–	11.50	30	–	0.65	30	–	14.72	30	–	2.27	30	–
ShortGPT$_5$	12.50	22.92	35	–	19.12	35	–	22.12	35	–	86.33	35	–	49.42	35	–
ShortGPT$_3$	7.50	23.05	37	–	19.68	37	–	24.65	37	–	91.35	37	–	66.62	37	–
AdaInfer	9.75 → 22.63	22.59	32.37	7.92	**21.62**	33.33	12.12	25.95	34.20	13.50	**92.97**	30.95	5.77	**72.83**	39.00	0.00

Table 2: Performance and computational efficiency in multi-tasks. Accuracy (%) is denoted by 'Acc'. Results of few-shot learning with sample sizes of $\{5, 10, 15, 20\}$ are reported in average values. ShortGPT$_p$ follows the orignal paper's setting; ShortGPT$_5$ and ShortGPT$_3$ are to skip the last 5 and 3 decoding layers, respectively. The **best** and **second-best** performance in each section are highlighted.

Classifier Configuration. We utilized the sklearn library for training SVM[1] and CRF[2], adhering to their default configurations. SVM and CRF are computationally efficient, with low training and inference costs. In contrast, large models like LLaMA2 require much more computation due to deeper layers, higher dimensions, and longer sequences, making SVM and CRF lightweight by comparison.

In-Context Learning Setting. We evaluate AdaInfer under zero-shot and few-shot scenarios, using sample sizes of 5, 10, 15, and 20. For zero-shot, the input is the test set's x_q. For few-shot, training set examples are added to x_q. For in-context learning prompts, we use a default template: $\mathrm{Q} : \{x_k\}\backslash\mathrm{nA} : \{y_k\}\backslash\mathrm{n}\backslash\mathrm{n}$, concatenating random x_k and y_k samples from task-specific training sets.

Metrics. We report the top-1 accuracy on the test set following function vectors [Todd *et al.*,] (HELM implementation)[3]. For computational efficiency, we follow previous work [Ma *et al.*, 2023; Schuster *et al.*, 2022; Elbayad *et al.*, 2019] and report the pruning ratio (*P. Ratio*) and the average number of activated layers (*#Avg. L*) for each task, along with their variance (*Var*). These metrics directly measure complexity reduction, avoiding conflation with implementation or infrastructure-specific details [Dehghani *et al.*, 2021]. Considering the conditional checks and classifier computation involved in AdaInfer, we also compare the actual speed of AdaInfer in real-world scenarios with Dense implementation, reporting wall-clock time [Dehghani *et al.*, 2021].

Baseline Method: ShortGPT and Speculative Decoding. We compare AdaInfer with the structured pruning method ShortGPT [Men *et al.*, 2024], which prunes redundant layers in LLMs based on similarity scores. For the OPT model, we calculate redundant layers as outlined in the paper. For the LLama model, we use the same layers reported. Note that these model pruning methods apply a static pruning ratio across all tasks, whereas our AdaInfer adaptively performs model pruning based on input.

5.2 Main Results

The main results of AdaInfer are presented in Table 2. Conducted in few-shot settings, these experiments show the Top-1 accuracy, pruning ratios, average active layers for each task, and their variance. From the perspective of performance and computational efficiency, we draw the following experimental conclusions.

AdaInfer has minimum impact on performance (<1%). Table 2 shows that the Top-1 accuracy of AdaInfer remains within a very narrow margin of less than 1% for all tasks compared to dense models, *i.e.,* without early exit. In contrast, ShortGPT, following the paper's setting and denoted as ShortGPT$_p$, experiences a significant performance drop for almost all tasks [4]. Since AdaInfer adaptively skips decoding layers, the number of layers skipped varies for different instances and across different tasks. For a fair comparison, we have also evaluated ShortGPT$_5$ and ShortGPT$_3$, which skip the last 5 and 3 decoding layers, respectively. The numbers of skipped layers are chosen to match the overall range of layers skipped by AdaInfer. This allows for a more comprehensive comparison with methods that use a fixed pruning ratio [Yang *et al.*, 2024; Ma *et al.*, 2023; Men *et al.*, 2024]. The results in Table 2 demonstrate that AdaInfer surpasses both settings.

[1] https://scikit-learn.org/stable/modules/svm.html

[2] https://sklearn-crfsuite.readthedocs.io/en/latest/

[3] https://huggingface.co/blog/open-llm-leaderboard-mmlu

[4] We noted a decline in the performance of the reproduced ShortGPT on the SQuAD dataset when the prompts increased to 10, 15, 20 shots.

Task	Llama2 7B (FP32)			Llama2 13B (FP16)		
	Dense	AdaInfer	Speed up	Dense	AdaInfer	Speed up
MMLU	796.53	781.31	**1.02x**	339.19	320.46	**1.05x**
Sentiment	41.18	39.69	**1.04x**	28.18	21.76	**1.30x**

Table 3: Wall-clock time (s) and actual speedup for 358 test samples from MMLU and 245 test samples from Sentiment Tasks.

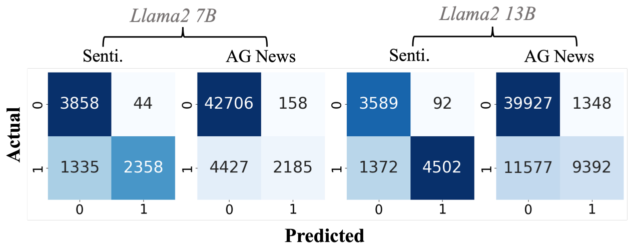


Figure 4: SVM confusion matrix.

In short, AdaInfer achieves adaptive inference while maintaining LLM capabilities and in-context learning abilities without modifying model parameters. This finding is promising, especially in light of our Observation 1, where we demonstrate the feasibility of implementing early exit strategies while preserving performance. As shown in Table 2, AdaInfer even surpasses the last layer accuracy for certain tasks. This suggests that deep layers may over-represent certain instances, potentially impeding performance during LLM inference.

Pruning ratio ranges from 9% to 43%, average 17.8%. We report the average and variance of the activated layers for each task and compute the pruning ratios in Table 2. The pruning ratios vary from task to task, ranging from 9% to 43%, a clear indication of AdaInfer assesseing different early exit layer configurations for different task inputs. More layers are skipped for simple tasks like sentiment analysis task, where a 43% reduction in computational cost can be achieved on Llama2-13B. For more complex question answering tasks, the savings range from 9% to 20%.

Wall-clock time. Next, we study the end-to-end runtime of AdaInfer. Table 3 compares the runtime of AdaInfer with a dense implementation on MMLU and Sentiment tasks (5-shot, batch size set to 1), using $6 \times V100$ (32GB). We observed a 1.03x speed up on MMLU and 1.17x speed up on Sentiment when applying AdaInfer. That is, despite AdaInfer converting hidden states to logits at each block through the LM head layer, it only utilizes the last token's hidden state, which is independent of the input sequence length. Consequently, this computation cost is minimal (0.03% of the total FLOPs for transformer inference). Meanwhile, statistical classifiers like SVM have much lower computational costs compared to LLM inference, highlighting the computational efficiency potential of AdaInfer.

5.3 Analysis of SVM Classifier

Analysis of AdaInfer's theoretical upper bound and feature engineering. To understand AdaInfer's performance limits, we first analyze the theoretical upper bound of early stopping. On the LLaMA-7B (32-layer) model, we record predictions at each layer for test samples. For each sample, we define its theoretical optimal activation layer as the earliest layer that outputs the correct token. As shown in Figure 5, AdaInfer's average activation layer is 3-5 layers behind the theoretical optimum. The high variance in theoretical optimal layers indicates that samples within the same task require different computational depths.

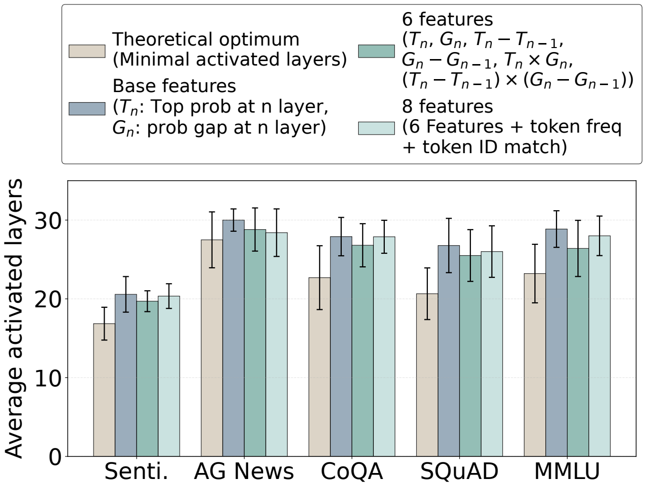


Figure 5: Theoretical upper bound of AdaInfer and feature engineering. "Senti." refers to the sentiment classification task; "CoQA" refers to CommonsenseQA.

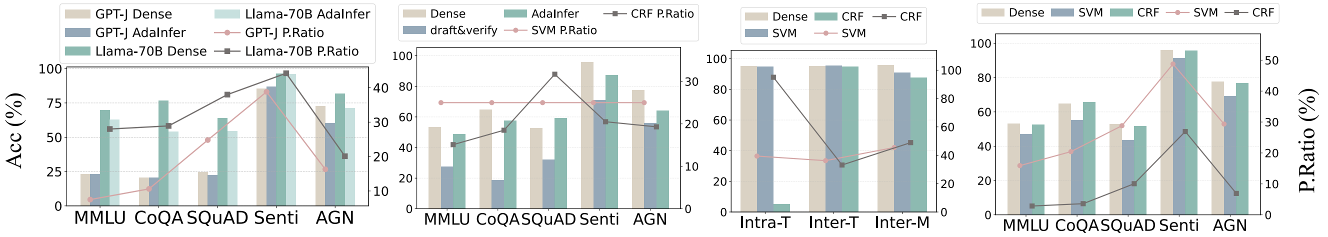
Feature	Sentiment	MMLU
Base Features (gap, top prob)	94.90	41.13
+attn	94.90	41.13
+hidden state	67.53	41.13
+mlp	67.88	41.93

Table 4: Comparative analysis of SVM performance with incremental feature addition in sentiment and MMLU/anatomy tasks.

For feature selection, we gradually expand from basic features. The main experiments only use two features from decoding layer n: the highest probability value T_n and probability gap G_n. We then extend to six features (T_n, G_n, $T_n - T_{n-1}$, $G_n - G_{n-1}$, $T_n \times G_n$, $(T_n - T_{n-1}) \times (G_n - G_{n-1})$). Finally, we add token frequency and token consistency to form an eight-feature combination. As shown in Figure 5, adding features improves performance. However, the gap to theoretical upper bound remains, suggesting room for further optimization.

We also examine intermediate features, including cosine similarities between blocks (attention values, MLP, and hidden states), as discussed in Section 4.1. Table 4 shows that attention values have no impact, while MLP and hidden states negatively affect results, consistent with Figure 3. We believe logits effectively measure forward progress, whereas changes in other features reflect additional factors.

Classifier performance. Figure 4 presents the confusion matrices of our SVM classifier on Sentiment Analysis and AG News tasks. The SVM was trained with only two basic features. For Llama2-7B, the classifier achieves high true



(a) Eval across scaling law (b) Alternative exit strategies (c) Generalization study (d) Compared with draft&verify

Figure 6: Fine-grained results. "Senti" refers to the sentiment classification task, "CoQA" refers to CommonsenseQA, and "AGN" refers to AG News task.

negatives (3,858 for Sentiment, 42,706 for AG News) and true positives (2,358 and 2,185). The performance improves on Llama2-13B. Here, we see higher true positives. The true negative rates remain high. These high true negative rates show that the classifier is conservative in making stopping decisions. This helps maintain output quality. The results confirm that our simple feature selection is effective.

5.4 Fine-grained Results

Evaluation across scaling law. Table 2 reports results on 7B/13B-sized Llama2 and OPT models. In experiments with the Llama2 70B version, we observe that in a zero-shot setting, AdaInfer matches or slightly exceeds the dense model while reducing computational costs by 10% to 50%. However, in the few-shot setting, despite similar reductions in computation, AdaInfer's accuracy shows a 1% to 25% drop across different tasks compared to the dense model, *i.e.,* without early exit. This calls for more feature engineering for larger models, such as the 70B or even larger scales. Improving AdaInfer to adapt to these larger models is a direction for our future research. The results of all LLMs are summarized in Figure 6(a).

Evaluation on Alternative Exit Strategies. In Main Table 2, we employ SVM as the classifier for AdaInfer. To explore the impact of alternative exit strategies, we implemented AdaInfer with a GAP threshold set at 0.8 (stopping inference when the current block's GAP feature exceeds 0.8) and AdaInfer with CRF as the classifier. Figure 6(b) shows that both GAP and CRF reduce computational costs by 3% to 50% while maintaining comparable LLM performance. Notably, in the zero-shot setting, GAP outperforms CRF, suggesting a relatively weak dependency between block features.

LLM Block Features and Sequential Processing. An intriguing finding emerges from our experiments. Despite CRF's specialization in sequence modeling, when modeling feature sequences block-by-block in zero-shot scenarios, it underperforms compared to a simpler SVM classifier. This reveals the block features extracted by LLMs appear to exhibit relative temporal independence. Second, rather than explicit temporal relationship modeling, LLMs may rely more heavily on distributed representations acquired during pretraining for processing sequential information. In other words, the features from individual LLM blocks may not necessarily maintain inherent temporal relationships. This suggests that LLM performance improvements are substantially driven by architectural design choices and large-scale pretraining approaches.

Classifier Generalization Study We train the AdaInfer classifier on 6 randomly selected datasets from a pool of 71 subdatasets. To evaluate generalization, we conduct three tests:

- **Intra-Task:** Testing sentiment task using sentiment-trained classifier.
- **Inter-Task:** Testing sentiment using classifier trained on knowledge QA.
- **Inter-Model:** Testing sentiment on Llama2-13B using Llama2-7B-trained classifier.

Results in Figure 6(c) show that SVM exhibits strong intra-task and inter-task generalization, aligning with our main findings. However, CRF suffers from premature termination in intra-task settings, likely due to overfitting to local features. Inter-model generalization shows moderate accuracy due to variations in logit distributions across models. Based on results from Tables 2, SVM remains the optimal classifier for AdaInfer.

Compared with Self-Speculative Decoding Our experiments focus on multiple-choice tasks, where we generate the first token for each input with dynamic depths. In contrast, the draft&verify method [Zhang *et al.*, 2024; Elhoushi *et al.*, 2024] *generates the first token for all inputs using full static depth*. For comparison, we also employ a 5-shot prompt (with fixed random seeds) and skip layers as described in their papers. This targeted approach, as shown in Figure 6(d), enables AdaInfer to achieve a dynamic pruning ratio and outperform draft&verify in first token generation settings.

6 Conclusion

In this paper, we show that not all layers are needed during LLM inference. We introduce AdaInfer, a simple yet effective algorithm that dynamically decides when to stop inference based on each input. The decision is made by a lightweight classifier using two intuitive features: the top token's probability and its gap with the second-ranked token. While not exhaustive, these features enable significant efficiency gains without altering model parameters. AdaInfer prunes an average of 17.8% of layers (up to 43%) with minimal performance loss (<1%). It is especially effective for workloads with many easy tasks and is compatible with other acceleration methods, offering a new direction for efficient inference.

Acknowledgments

This paper was supported by the National Key R&D Program of China (2024YFE0111800), and the National Science Foundation of China (NSFC No.62032001, and 62106249).

References

[Bejnordi *et al.*, 2019] Babak Ehteshami Bejnordi, Tijmen Blankevoort, and Max Welling. Batch-shaping for learning conditional channel gated networks. *arXiv preprint arXiv:1907.06627*, 2019.

[Belrose *et al.*, 2023] Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens. *CoRR*, abs/2303.08112, 2023.

[Bolukbasi *et al.*, 2017] Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. Adaptive neural networks for efficient inference. In *International Conference on Machine Learning*, 2017.

[Chen *et al.*, 2023] Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *CoRR*, 2023.

[Christiano *et al.*, 2022] Paul Christiano, Ajeya Cotra, and Mark Xu. Eliciting latent knowledge. Technical report, Alignment Research Center (ARC), 2022. Cited on page 5, 11, 17, 44.

[Deary *et al.*, 2001] Ian J Deary, Geoff Der, and Graeme Ford. Reaction times and intelligence differences: A population-based cohort study. *Intelligence*, 29(5):389–399, 2001.

[Dehghani *et al.*, 2021] Mostafa Dehghani, Anurag Arnab, Lucas Beyer, Ashish Vaswani, and Yi Tay. The efficiency misnomer. *arXiv preprint arXiv:2110.12894*, 2021.

[Elbayad *et al.*, 2019] Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. Depth-adaptive transformer. *arXiv preprint arXiv:1910.10073*, 2019.

[Elhoushi *et al.*, 2024] Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed A Aly, Beidi Chen, and Carole-Jean Wu. Layerskip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics, ACL*, 2024.

[Han *et al.*, 2021] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[Hearst *et al.*, 1998] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 1998.

[Hendrycks *et al.*, 2021] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[Huang *et al.*, 2017] 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. *arXiv preprint arXiv:1703.09844*, 2017.

[Hubel and Wiesel, 1962] David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106, 1962.

[Kim *et al.*, 2024] Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, and Hyoung-Kyu Song. Shortened llama: A simple depth pruning for large language models. *arXiv preprint arXiv:2402.02834*, 2024.

[Lafferty *et al.*, 2001] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[LeCun *et al.*, 1989] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

[Leviathan *et al.*, 2023] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning, ICML*, 2023.

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

[Li *et al.*, 2021] Xiaonan Li, Yunfan Shao, Tianxiang Sun, Hang Yan, Xipeng Qiu, and Xuanjing Huang. Accelerating bert inference for sequence labeling via early-exit. *arXiv preprint arXiv:2105.13878*, 2021.

[Li *et al.*, 2023] Xiang Li, Yiqun Yao, Xin Jiang, Xuezhi Fang, Xuying Meng, Siqi Fan, Peng Han, Jing Li, Li Du, Bowen Qin, Zheng Zhang, Aixin Sun, and Yequan Wang. FLM-101B: an open LLM and how to train it with $100k budget. *CoRR*, abs/2309.03852, 2023.

[Lin *et al.*, 2017] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. *Advances in neural information processing systems*, 30, 2017.

[Liu *et al.*, 2020] Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Haotang Deng, and Qi Ju. Fastbert: a self-distilling bert with adaptive inference time. *arXiv preprint arXiv:2004.02178*, 2020.

[Liu *et al.*, 2023] Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Ré, and Beidi Chen. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning, ICML*, 2023.

[Ma *et al.*, 2023] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 2023.

[Men *et al.*, 2024] Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect, 2024.

[Murata *et al.*, 2000] Akira Murata, Vittorio Gallese, Giuseppe Luppino, Masakazu Kaseda, and Hideo Sakata. Selectivity for the shape, size, and orientation of objects for grasping in neurons of monkey parietal area aip. *Journal of neurophysiology*, 83(5):2580–2601, 2000.

[Nostalgebraist, 2020] Nostalgebraist. Interpreting gpt: The logit lens. https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens, 2020. LessWrong.

[Pope *et al.*, 2023] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems*, 5, 2023.

[Rajpurkar *et al.*, 2016] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *arXiv e-prints*, page arXiv:1606.05250, 2016.

[Raposo *et al.*, 2024] David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models, 2024.

[Salthouse, 1996] Timothy A Salthouse. The processing-speed theory of adult age differences in cognition. *Psychological review*, 103(3):403, 1996.

[Schuster *et al.*, 2022] Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. *Advances in Neural Information Processing Systems*, 2022.

[Socher *et al.*, 2013] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013.

[Talmor *et al.*,] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*.

[Teerapittayanon *et al.*, 2016] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd international conference on pattern recognition (ICPR)*, 2016.

[Todd *et al.*,] Eric Todd, Millicent L. Li, Arnab Sen Sharma, Aaron Mueller, Byron C. Wallace, and David Bau. Function vectors in large language models. In *Proceedings of the 2024 International Conference on Learning Representations*.

[Touvron *et al.*, 2023] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[Varshney *et al.*, 2023] Neeraj Varshney, Agneet Chatterjee, Mihir Parmar, and Chitta Baral. Accelerating llama inference by enabling intermediate layer decoding via instruction tuning with lite. *arXiv e-prints*, 2023.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[Wang and Komatsuzaki, 2021] Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax, May 2021.

[Wang *et al.*, 2022] Yequan Wang, Hengran Zhang, Aixin Sun, and Xuying Meng. CORT: A new baseline for comparative opinion classification by dual prompts. In *Findings of the Association for Computational Linguistics: EMNLP*, 2022.

[Yang *et al.*, 2020] Le Yang, Yizeng Han, Xi Chen, Shiji Song, Jifeng Dai, and Gao Huang. Resolution adaptive networks for efficient inference. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020.

[Yang *et al.*, 2024] Yifei Yang, Zouying Cao, and Hai Zhao. Laco: Large language model pruning via layer collapse. *arXiv preprint arXiv:2402.11187*, 2024.

[Zeng *et al.*, 2023] Dewen Zeng, Nan Du, Tao Wang, Yuanzhong Xu, Tao Lei, Zhifeng Chen, and Claire Cui. Learning to skip for language modeling. *arXiv preprint arXiv:2311.15436*, 2023.

[Zhang *et al.*, 2015] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 2015.

[Zhang *et al.*, 2022] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[Zhang *et al.*, 2024] Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. Draft& verify: Lossless large language model acceleration via self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics, ACL*, 2024.