D-LLM: A Token Adaptive Computing Resource Allocation Strategy for Large Language Models

Yikun Jiang[1*], **Huanyu Wang**[1*], **Lei Xie**[1], **Hanbin Zhao**[2†], **Chao Zhang**[2], **Hui Qian**[2], **John C.S. Lui**[3]
[1]FSI Lab, Huawei Technologies Co., Ltd.
[2]Zhejiang University [3]Chinese University of Hong Kong
`{jiangyikun5,xielei70}@huawei.com`
`{huanyuhello,zhaohanbin,zczju,qianhui}@zju.edu.cn`
`cslui@cse.cuhk.edu.hk`

Abstract

Large language models have shown an impressive societal impact owing to their excellent understanding and logical reasoning skills. However, such strong ability relies on a huge amount of computing resources, which makes it difficult to deploy LLMs on computing resource-constrained platforms. Currently, LLMs process each token equivalently, but we argue that not every word is equally important. Some words should not be allocated excessive computing resources, particularly for dispensable terms in simple questions. In this paper, we propose a novel dynamic inference paradigm for LLMs, namely D-LLMs, which adaptively allocate computing resources in token processing. We design a dynamic decision module for each transformer layer that decides whether a network unit should be executed or skipped. Moreover, we tackle the issue of adapting D-LLMs to real-world applications, specifically concerning the missing KV-cache when layers are skipped. To overcome this, we propose a simple yet effective eviction policy to exclude the skipped layers from subsequent attention calculations. The eviction policy not only enables D-LLMs to be compatible with prevalent applications but also reduces considerable storage resources. Experimentally, D-LLMs show superior performance, in terms of computational cost and KV storage utilization. It can reduce up to 45% computational cost and KV storage on Q&A, summarization, and math solving tasks, 50% on commonsense reasoning tasks.

1 Introduction

Large language models(LLMs), have demonstrated amazing capability on generation tasks. Benefiting from existing parameter-efficient finetuning strategies [30, 44, 54, 66, 84], LLMs can also be adapted to specific domains with extra training costs. However, such models containing billions of parameters require expensive computation costs and memory overhead at the inference stage. Thus, deploying LLMs on resource-constrained platforms becomes challenging especially considering the ever-growing requirements for offline using, model customizing, and data privacy [10, 14, 16, 50, 61, 63, 83]. To alleviate this problem, various inference acceleration techniques have been proposed, *e.g.*, quantization [4, 42, 57, 76, 79], distillation [24, 39, 40, 64, 65], and pruning [51, 52, 77, 81, 82]. Among these methods, pruning is a typical structural compression method to reduce the participating network layers. These methods usually remove redundant neurons based on hand-crafted metrics [37, 78], when optimizing or finetuning large language models.

*Equal contribution. †Correspondence to Hanbin Zhao. Our code is available at `https://github.com/Jyk-122/D-LLM`.

Although previous methods can decrease computational cost, many essential characteristics of natural languages are ignored. *First, tasks of different difficulties should not be allocated with the same computing resource.* Obviously, a response to "how are you" is much easier than "explaining a math theorem". *More importantly, not every word in a sentence is equally important.* Thus, it is not necessary to allocate too much computing resources to non-critical tokens, *i.e.*, articles and punctuation marks. In this paper, we introduce a novel *dynamic inference mechanism, namely D-LLMs,* which can boost inference speeds by dynamically skipping redundant network components. Specifically, we assign an execution decision module for each transformer layer. Before going through a transformer layer, tokens need to pass through the decision module to obtain an execution decision. Based on the execution decision, D-LLMs would adaptively decide whether the following layer should be executed or skipped. Thus, dispensable tokens and simple tasks would utilize much fewer layers than relevant tokens and difficult tasks. Another critical issue that should be emphasized here is how to apply such dynamic inference mechanism on LLMs, *e.g.*, LLaMA [69], GPT [55, 58], *etc.* Note that it is unacceptable to retrain or finetune entire models due to the high computational overhead and massive training data. Therefore, we combine our strategy with finetuning methods by inserting low-rank adaptors, *LoRA* [30], for well pre-trained large language models.

With the aforementioned method, each token execution by a specific network topology can successfully reduce the computational cost at inference time. However, to deploy such dynamic inference networks for realistic scenarios, making this dynamic inference compatible with KV-cache strategies [13, 21, 29] is a crucial challenge. Compared with static inference networks which cache all KV embeddings of previous tokens, dynamic inference networks selectively skip layers would lead to KV-cache misses. To tackle this issue, we design a simple yet effective eviction strategy on KV-cache in our proposed D-LLMs framework. Specifically, we evict the KV embeddings of uncalculated transformer layers in previous tokens and simplify the regular causal masks to sparse ones. For example, if a given token skips a transformer layer at the inference time, it would not be included in self-attention calculation of subsequent tokens. This way, the attention mask in each layer is also dynamically changed according to the execution decisions of preceding tokens.

To summarize, the main contributions of our paper are concluded as follows:

- We propose a novel inference framework for LLMs, termed D-LLMs, which can significantly reduce computational resource requirements with an adaptive allocation scheme. We design decision modules in LLMs, which dynamically decide to execute or skip transformer layers at the inference time.
- We propose a simple yet effective eviction policy by transforming the causal self-attention masks. As a result, it not only makes the proposed D-LLMs compatible with KV-cache methods but also saves the storage overhead at inference time, which is particularly important for resource-constrained platforms.
- We conduct extensive experiments on main-stream LLMs. The results demonstrate that D-LLMs reduce up to 45% computational cost and KV-cache storage on Q&A and math solving tasks, and 50% on commonsense reasoning tasks without performance degradation.

2 Related Works.

LLMs Finetuning Methods. Due to the large amount of computational cost of training LLMs, various parameter efficient finetuning methods have been explored, including prompt-based learning methods [7, 19, 25, 41, 80], adapter-based learning methods[27, 34, 67, 74] and reparametrization-based learning methods [11, 15, 30, 45]. Prompt-based learning methods propose to insert and infer a collection of trainable embeddings to existing tokens. For example, P-tuning [47] adds a prefix to input, and P-tuning v2 [46] applies prompts to intermediate layers. Adapter-based learning methods propose to insert specially designed adaptors to pre-trained LLMs. How to insert adaptors has also been explored, *i.e.*, Adamix [74] and Compacter [34] serially connect adaptors and [27, 67] parallelly connect adaptors with transformer. Reparametrization-based learning methods exploit finetuning models with low intrinsic dimension [1]. Specifically, LoRA [30] decomposes parameters into two low-rank trainable matrices before computation.

Different from these methods which focus on parameter-efficient finetuning, D-LLMs pay more attention to achieving high-speed inference via adaptive computing resource allocation.

LLMs Acceleration Methods. Extensive research works have been proposed to compress LLMs and reduce the network parameters. Among these methods, quantization [4, 36, 42, 57, 62], distillation [24, 39, 40, 64, 65] and pruning [38, 43, 48, 51, 82] are mainstream approaches. Quantization methods propose to reduce the bits of each parameter by converting floating-point (FP32 / FP16) parameters to integers (INT8 / INT4) or other discrete forms. This line of methods preserves the structural characteristics of the network and compresses models in exchange for lowering accuracy. Distillation methods [24, 39, 40] attempt to transfer knowledge from LLMs to a lightweight student model. However, such methods usually require retraining models in an end-to-end manner and may take up large computing resources. Model pruning methods propose to reduce the complexity of models by removing redundant parameters. These methods have two forms, static pruning [3, 18, 23] and dynamic pruning [2, 12, 21, 29, 49, 81]. Static pruning methods usually remove redundant layers based on various relevant metrics. For example, Shortened-LLaMA [35] measures the importance of layers and prunes the unnecessary ones. However, with the increase of pruned parameters, static pruning poses inevitable performance degradation. Dynamic pruning methods tend to prune unimportance layers based on inputs. For example, Ada-Infer [17] introduces an early-exit mechanism to stop the inference at intermediate layers. Mixture-of-Depths [59] measures the importance of each token and only calculates the top-k tokens at different layers.

In essence, the proposed D-LLM belongs to dynamic pruning methods, which dynamically decide which layers should be executed for each input token.

Dynamic Inference in Computer Vision. Dynamic inference is a promising technique to skip layers at inference time to achieve acceleration [70, 71, 72, 73], which was initially proposed in computer visions. Research on dynamic inference mechanisms has two different approaches. The first one is layer skipping. This line of methods mainly focus on how to design a specific dynamic decision module or metric to reduce computational cost. Specifically, ConvNet-AIG [70] uses a convolutional module to define the inference graph conditioned on inputs. It proposes a router to make the execution decision for each model layer. SkipNet [73] utilizes LSTM as the decision module to determine whether a layer should be executed or not. Besides, CoDiNet [72] attempts to model the relationship between the inputs and their executing decisions to enhance the optimization. The second line of methods is early prediction. While a dynamic inference network has only one exit, an early prediction network is characterized by multiple exits. In early prediction networks, the execution would stop once a criterion for a given sample is satisfied. MSDN [31] is a typical example, which introduces multiple early-exits, according to the allowed time budget. Instead of bypassing all units, DCP [20] generated decisions to reduce the computational cost on channels.

Inspired by dynamic inference in the CV task, we rethink the characteristics of NLP and raise the following question. *Should models allocate the same computing resource on tasks of different difficulties or tokens of different importance?* Motivated by this, we introduce the D-LLMs framework, which will dynamically allocate fewer computing resources for unimportant tokens.

3 Methods

3.1 Preliminary: LLMs Architecture

As a natural language processing task, a sequence of input words is first embedded into tokens $\{x^1, x^2, \cdots, x^N\}$ by a tokenizer. Each token would then go through a pre-defined large language model sequentially in an autoregressive manner. That is to say the target of $\{x^1, x^2, \cdots, x^n\}$ is x^{n+1}. In this way, the inference paradigm of a large language model is formulated as

$$x^{n+1} = f_{\text{output}} \odot f_L \odot f_{L-1} \odot \cdots \odot f_1 \odot f_{\text{head}}(x^1, x^2, \cdots, x^n), \tag{1}$$

where L is the total number of transformer layers, f_{head} is the word embedding and f_{output} is the word classifier. For convenience, we use the superscript to represent the token index and the subscript to represent the layer index of token embeddings in the following sections.

Transformer Layer. Dominant architectures of LLMs are decoder-only transformer networks, which consist of L transformer layers. Each transformer layer contains a multi-head self-attention (MHA) and a feed-forward network (FFN). The inference of the l-th transformer layer can be defined as,

$$h_l^n = \text{MHA}(x_l^n) + x_l^n, \tag{2}$$

$$x_{l+1}^n = \text{FFN}(h_l^n) + h_l^n, \tag{3}$$

where x_l^n is the n-th input token embedding of the l-th transformer layer and h_l^n is the output of MHA. Specifically, a MHA divides token embeddings into multiple heads and perform self-attention operations in parallel.

Self Attention. Since large language models mainly process inputs sequentially, the calculation of former tokens should not be affected by latter ones. To tackle this problem, the self-attention module (SA) uses a causal attention mask $\mathbf{M}$ on attention scores as

$$\mathbf{M}_{i,j} = \begin{cases} 0, & j \leq i, \\ -\infty, & j > i. \end{cases} \tag{4}$$

With the causal attention mask $\mathbf{M}$, the features of the n-th token are processed without subsequent tokens. Under such framework, the formulation of MHA$(\cdot)$ is defined as

$$\text{MHA}(x_l^n) := \text{concat}(\text{SA}(x_{l,1}^n), \text{SA}(x_{l,2}^n), \cdots, \text{SA}(x_{l,h}^n)) \circ W_O, \tag{5}$$

where W_O is a learnable parameter and $\circ$ is matrix multiplication. MHA$(\cdot)$ is essentially concatenation of self-attention operated features divided into h heads $\{x_{l,i}^n\}_{i=1}^h$. Based on learnable weights W_Q, W_K, W_V, a self attention is formulated as

$$\text{SA}(x_{l,i}^n) := \text{softmax}(\frac{x_{l,i}^n \circ W_Q \circ (X \circ W_K)^T}{\sqrt{d}} + \text{M}) \circ X \circ W_V, \tag{6}$$

where $X = \{x_{l,i}^1, x_{l,i}^2, \cdots, x_{l,i}^N\}$ is input features of the sentence, d is the dimension of each head and $(\cdot)^T$ is transpose. In addition, FFN consists of several linear layers with activation functions following the MHA.

KV-Cache. The KV-Cache method [56] has been a standard acceleration technique for large language model inference. As shown in Eq. 6, the query embedding of a given token only calculates the self-attention with previous keys and values. Since decoder-only LLMs process input tokens in an autoregressive way, the key and value embeddings can be reused to boost inference speed by storing calculated key and value embeddings. The KV-cache method reduces the computation complexity from $O(n^2)$ to $O(n)$, but brings extra memory overheads to store the KV-cache when generating long texts [13, 21, 29, 75].

3.2 D-LLMs: Dynamic Inference LLMs

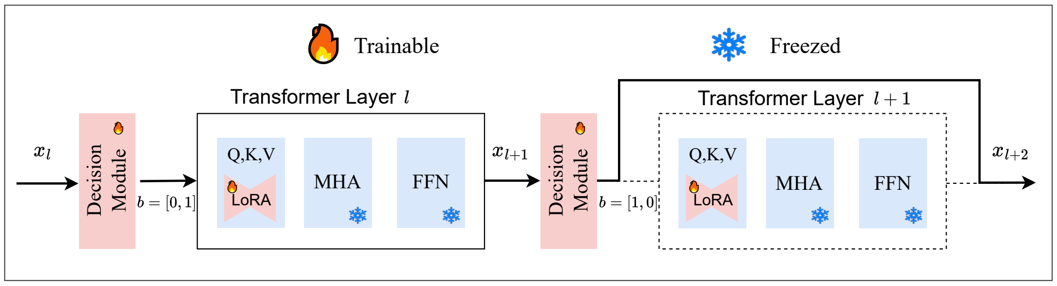
In this section, we illustrate the proposed dynamic inference mechanism in large language models. We assign a dynamic decision module before each single transformer layer. At the inference time, the dynamic decision module makes an execution decision whether the related transformer layer should be executed or not as shown in Fig. 1a.

Given a token x in an input sequence, it would go through a dynamic decision module g_l, prior to the l-th transformer layer f_l in large language models. As shown in Fig. 1b, a dynamic decision module is composed of two linear layers separated by an activation function. At the inference time, the dynamic decision module outputs $g_l(x_l)$, the parameters of a categorical distribution that represent the probability of skipping and executing based on the input feature x_l to the l-th transformer layer. The execution decision for the l-th layer is defined as

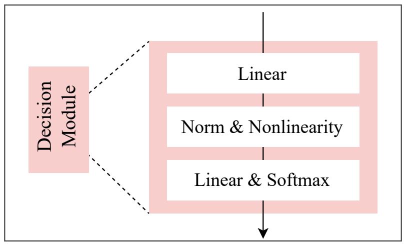
$$b_l := \mathbb{1}(\arg\max(g_l(x_l))), \tag{7}$$

where $\mathbb{1}$ is one-hot operation and b_l is a two-dimensional vector. This way, the result of the execution decision tuple for each token is either $[0, 1]$ or $[1, 0]$. However, $\arg\max$ operation deterministically outputs the max argument without exploration and is not differentiable during training. To solve this problem, we utilize the Gumbel-Softmax reparametrization technique [32] and straight-through estimator [5] to ensure the transformer networks can be optimized in an end-to-end fashion during training. Thus, the execution decisions are organized as hard forms when forward computing:

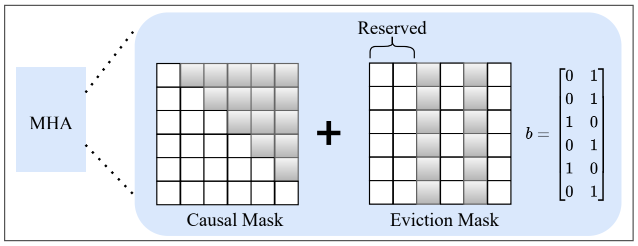
$$\hat{b}_l := \mathbb{1}(\arg\max(\log(g_l(x_l)) + \pi)), \quad \pi \sim G(0, 1), \tag{8}$$



(a) The framework of D-LLMs.



(b) The Dynamic Decision Module.


MHA
Reserved
Causal Mask
Eviction Mask
$b = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$

(c) The KV-cache Eviction Mask.

Figure 1: The framework the proposed D-LLMs. The inference paradigm of dynamic decisions for transformer layers is shown in Fig. 1a. The design of dynamic execution decision modules is shown in Fig. 1b. The mask in multi-head self-attention with eviction strategy is shown in Fig. 1c.

where $G(0,1)$is a Gumbel distribution. With sampled noise π, the decisions $\hat{b}_l$ satisfy the categorical distribution $g_l(x_l)$. The decisions are calculated in soft forms when back-propagating:

$$\tilde{b}_{l,i} = \frac{\exp((\log(g_l(x_l))_i + \pi_i)/\tau)}{\sum_i \exp((\log(g_l(x_l))_i + \pi_i)/\tau)}, i \in \{0,1\}. \tag{9}$$

The temperature τ controls the sharpness of softmax. As $\tau \to 0$, the softmax result converges to the discrete form from the categorical distribution. Finally, the output of layer l is defined as

$$x_{l+1} = \hat{b}_{l,0} \cdot x_l + \hat{b}_{l,1} \cdot f_l(x_l), \tag{10}$$

where $\hat{b}_{l,0}, \hat{b}_{l,1} \in \{0,1\}$ are the decisions of skipping or executing the l-th transformer layer.

3.3 Customizable Acceleration Rate

According to Eq. 10, when $\hat{b}_{l,1}$ is zero, $f_l(x_l)$ would not be involved in calculation. Thus, it achieves reducing the computational cost of f_l. The average acceleration rate ω^n for the token x^n can be defined as

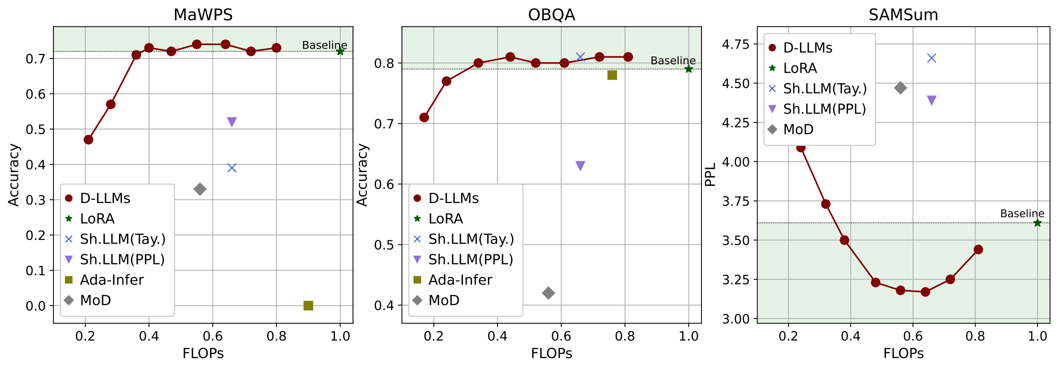
$$\omega^n = 1 - \frac{1}{L} \cdot \textstyle\sum_{l=1}^{L}(\hat{b}_{l,1}^n), \tag{11}$$

where L is the total number of layers, and $\hat{b}_{l,1}$ is the decision of executing the n-th token at the l-th layer. To make the acceleration rate adaptive to platforms of different computing capabilities, we design an acceleration ratio loss to customize the acceleration rate as

$$\mathcal{L}_{\text{rate}} = \frac{1}{N} \cdot \textstyle\sum_{n=1}^{N}(|\omega^n - \Omega|_1), \tag{12}$$

where $|\cdot|_1$ is l_1-norm and the Ω is the user-defined target acceleration rate.

Based on the aforementioned method, we achieve the optimization and inference for D-LLMs in end-to-end manners. Note that in addition to training the LLM model from scratch and obtaining generalized D-LLMs, our method can easily work on pre-trained LLMs by employing *LoRA* [30].



(a) Acc.(↑) v.s. FLOPs on MaWPS. (b) Acc.(↑) v.s. FLOPs on OBQA. (c) PPL(↓) v.s. FLOPs on SAMSum.

Figure 2: The performance against computational cost of D-LLMs on three datasets. The figures show that reducing around 40% to 60% computational cost achieves the best trade-off.

3.4 KV-Cache Eviction Strategy

In this section, we elaborate how our proposed dynamic inference mechanism can cooperate with the KV-cache strategy. Considering the inference of the n-th token of the l-th transformer layer in eq. (6), all previous tokens are involved in the calculation of x_l^n. It means that even the execution decision $\hat{b}_{l,1}^i, i \in \{1, \cdots, n\}$ on token embedding x_l^i is zero, we still need to calculate its keys and values for self-attention calculation of x_l^n. We conduct an aggressive strategy of self-attention, which bypasses the features of skipped tokens. We propose a KV-cache eviction policy by designing a mask on the attention matrix to ignore the uncalculated features. Thus, the corresponding KV-cache is unnecessary to keep, which reduces storage overhead during inference.

As shown in Fig. 1c, we design an eviction attention mask E at each layer together with causal mask in Eq. 6, which supports training in batch. In addition, recent research on attention maps and KV-cache [26, 75] shows that tokens at the beginning of a sentence are crucial to subsequent tokens. We also find that evicting the KV-cache of initial tokens at skipped layers would either leads to performance decreasing or causes unstable optimization. Therefore, we preserve the keys and values of the first m tokens at the beginning of texts. Hence, the eviction attention mask for the l-th layer is encoded as

$$\mathrm{E}_{i,j} = \begin{cases} 0, & j \leq m \text{ or } b_{l,1}^j = 1 \\ -\infty, & \text{otherwise}, \end{cases} \tag{13}$$

where $b_{l,1}^j = 1$ represents the j-th token executing the l-th transformer layer, and m is the number of preserved tokens at beginning. Thus, the KV-cache of m initial tokens are kept in the prefilling phase. In our method, we simply set the decisions of initial tokens to be executed in a deterministic way, which demonstrates equivalent performance and is easier to implement in practice.

Finally, we introduce the overall loss functions in D-LLMs. The objective function is composed of two parts: cross-entropy loss and acceleration ratio loss.

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \alpha \cdot \mathcal{L}_{\text{rate}}, \tag{14}$$

where $\mathcal{L}_{\text{CE}}$ is the cross-entropy loss and α is a hyper-parameter of the acceleration ratio loss $\mathcal{L}_{\text{rate}}$.

4 Experiments

4.1 Experimental Setup

Models and Benchmarks. We conduct experiments of the proposed D-LLMs on widely used LLMs, LLaMA2-7B and LLaMA3-8B. LLaMA2-7B and LLaMA3-8B both contain 32 transformer layers. We perform few-shot finetuning on nine different benchmarks to evaluate the effectiveness. The benchmarks are divided into three different tasks. The first task is Q&A and summarization, including Alpaca [68] and SAMSum [22], and perplexity (PPL) is used to measure their performance.

Table 1: Performance Comparison on different tasks under Few-shot Settings based on LLaMA2-7B. *Sh. Lla. PPL* refers to Shortened-LLaMA applying PPL metric. *Sh. Lla. Tay.* refers to Shortened-LLaMA applying Taylor metric. *Ada-Inf.* refers to Ada-Infer. For convenience, we mark the best performance in red and the lowest computational cost in blue.

Dataset	*MoD* [59]		*Sh. Lla. PPL* [35]		*Sh. Lla. Tay.* [35]		*Ada-Inf.* [17]		*D-LLM*	
Q&A	PPL(↓)	FLOPs(↓)	PPL(↓)	FLOPs(↓)	PPL(↓)	FLOPs(↓)	PPL(↓)	FLOPs(↓)	PPL(↓)	FLOPs(↓)
Alpaca	10.32	0.56	7.09	0.66	7.65	0.66	319	0.65	6.01	0.59
SAMSum	4.47	0.56	4.39	0.66	4.66	0.66	874	0.56	3.18	0.55
Math	Acc.(↑)	FLOPs(↓)	Acc.(↑)	FLOPs(↓)	Acc.(↑)	FLOPs(↓)	Acc.(↑)	FLOPs(↓)	Acc.(↑)	FLOPs(↓)
GSM8K	0.08	0.56	0.10	0.66	0.18	0.66	0.00	0.83	0.29	0.59
MaWPS	0.33	0.56	0.52	0.66	0.39	0.66	0.00	0.90	0.74	0.56
Com. Sen.	Acc.(↑)	FLOPs(↓)	Acc.(↑)	FLOPs(↓)	Acc.(↑)	FLOPs(↓)	Acc.(↑)	FLOPs(↓)	Acc.(↑)	FLOPs(↓)
BoolQ	0.64	0.56	0.67	0.66	0.73	0.66	0.71	0.61	0.73	0.52
PIQA	0.49	0.56	0.76	0.66	0.83	0.66	0.55	0.63	0.84	0.52
SIQA	0.58	0.56	0.75	0.66	0.81	0.66	0.80	0.64	0.82	0.54
OBQA	0.42	0.56	0.63	0.66	0.81	0.66	0.78	0.76	0.80	0.53
MMLU	0.28	0.56	0.47	0.66	0.53	0.66	0.41	0.60	0.53	0.55

Second, GSM8K [9], MaWPS [33] are about math problem solving. The answers are marked as correct only if the numerical difference is less than 10^{-5}. Finally, BoolQ [8], PIQA [6], SIQA [60], OBQA [53], MMLU [28] are datasets about common sense reasoning. The performance on these benchmarks is evaluated by accuracy.

Implementation Details. To avoid finetuning the entire parameters in D-LLMs, we adopt the LoRA finetuning method as a baseline. We apply dynamic decision modules to transformer layers except the first two for training stability. The maximal context length of tokens is set to 1024. In the designing of the dynamic decision module, we use two linear layers with a hidden dimension of 512. For the KV-cache eviction policy, we reserve the first two tokens, *i.e.*, $m = 2$. Finally, α in the loss function is set to 5 for Q&A, summarization task, 1.0 for math problems tasks, and 0.1 for commonsense reasoning tasks. We finetune 20 epochs for GSM8K and 10 epochs for other datasets.

4.2 Performance Comparison

Inference Acceleration. We show the acceleration of our proposed D-LLMs. As defined in Eq. 11, D-LLMs can customize the acceleration rate by adjusting the target rate Ω. In Fig. 2, we show the performance under different FLOPs on three datasets. The performance on MaWPS and OBQA is evaluated by accuracy, while that on SAMSum is evaluated by PPL. With the increasing FLOPs on MaWPS and OBQA, the accuracy shows improvement. When only using 40% FLOPs and 30% FLOPs on MaWPS and OBQA, D-LLM exceeds the baseline with 100% FLOPs. In terms of SAMSum, which is a Q&A task, the performance gets better first and then gets worse. The reason that performance gets worse when the cost is over 60%, might be due to overfitting.

Comparison with state-of-the-art. We compare the proposed D-LLMs with state-of-the-art methods on nine benchmarks. These benchmarks are concluded in three tasks, *i.e.*, Q&A with summarization, Math Solving, and Common Sense Reasoning. The performance of D-LLMs exceed MoD [59], Shortened-LLaMA [35], and Ada-Infer [17] in a large margin. In terms of computational cost, we denoted the FLOPs of LLaMA2-7B with LoRA as 1.00, the cost of others are percentages based on it as shown in Tab. 1. It is worth noting that FLOPs are calculated on average cost over tokens. Specifically, with the same baseline, *i.e.*, LLaMA2-7B with LoRA finetuning, D-LLMs achieve better performance with only using about 50% computational cost. Shortened-LLaMA reports two different results in their paper, which utilize Taylor and PPL as metrics, respectively. D-LLMs surpass Shortened-LLaMA on all datasets, though adopting fewer computing resources. Besides, we also reproduce the MoD and Ada-Infer and show that D-LLMs perform better as well.

Table 2: Ablation on different tasks under Few-shot Settings based on LLaMA2-7B and LLaMA3-8B. *LoRA* refers to using LLaMA3-8B and LLaMA2-7B as pretrained model and finetuning with LoRA. *D-LLM w/o Evi. Str.* refers to the proposed method without eviction strategy.

Datasets	Alpaca	SAMSum	GSM8K	MaWPS	BoolQ	PIQA	SIQA	OBQA	MMLU
Metrics	PPL(↓) FLOPs(↓)	PPL(↓) FLOPs(↓)	Acc.(↑) FLOPs(↓)	Acc.(↑) FLOPs(↓)	Acc.(↑) FLOPs(↓)	Acc.(↑) FLOPs(↓)	Acc.(↑) FLOPs(↓)	Acc.(↑) FLOPs(↓)	Acc.(↑) FLOPs(↓)
Backbone	LLaMA2-7B								
LoRA finetune	4.69 1.00	3.61 1.00	0.27 1.00	0.72 1.00	0.73 1.00	0.84 1.00	0.81 1.00	0.79 1.00	0.54 1.00
D-LLM w/o. Evi. Str.	6.08 0.64	3.63 0.62	0.29 0.60	0.72 0.61	0.72 0.60	0.83 0.59	0.81 0.60	0.82 0.59	0.53 0.60
D-LLM	6.01 0.59	3.18 0.55	0.29 0.59	0.74 0.56	0.73 0.52	0.84 0.52	0.82 0.54	0.80 0.53	0.53 0.55
Backbone	LLaMA3-8B								
LoRA finetune	4.84 1.00	3.93 1.00	0.52 1.00	0.88 1.00	0.75 1.00	0.87 1.00	0.81 1.00	0.85 1.00	0.56 1.00
D-LLM	8.43 0.59	3.63 0.58	0.50 0.60	0.91 0.55	0.75 0.51	0.88 0.52	0.81 0.55	0.82 0.53	0.57 0.55

4.3 Quantitative Analysis

Ablation Study. We conduct ablation studies of our proposed method based on LLaMA2-7B and LLaMA3-8B and then perform the results in three different tasks including Q&A and Summary, Math Problem, and Common Sense Reasoning. As shown in Tab. 2, we illustrate the performance of dynamic inference large language models over the baseline. Based on LLaMA3-8B, with less than 55% computational cost, our proposed D-LLM achieves better performance than that of the baseline on five datasets. In comparison, obtaining a comparable performance, D-LLMs on LLaMA2-7B only take about 55% computational cost. On Q&A and Summary tasks, D-LLMs sacrifice some accuracy for lower computational cost. The performance of baseline methods, *i.e.*, LLaMA with LoRA finetuning, outperforms D-LLMs on Alpaca using 40% extra cost. Considering math-solving problems, D-LLMs spend more computing resources than common sense reasoning tasks, which also demonstrates that math-solving problems are more difficult.

KV-Cache Overhead. In this part, we discuss the KV-cache storage benefits of D-LLMs. With the KV-cache eviction strategy, some keys and values are not necessary, because these features are useless to subsequent tokens. So the memory overhead on KV-cache is reduced with the layers are skipped. The benefits of reducing KV-cache storage are actually the skipped ratios of transformer layers in D-LLMs. As a result, we achieve almost 45% storage overhead reduction than the baselines on both LLaMA3-8B and LLaMA2-7B.

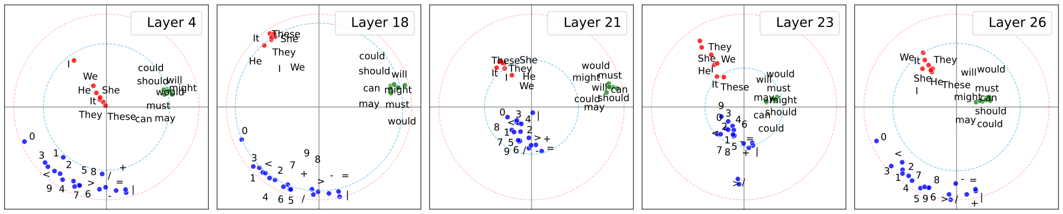
Table 3: The parameter analysis on numbers of reserved tokens not participate in dynamic inference.

Res. Tok.	$m = 0$		$m = 1$		$m = 2$		$m = 4$		$m = 8$	
Metrics	PPL(↓)	FLOPs(↓)	PPL(↓)	FLOPs(↓)	PPL(↓)	FLOPs(↓)	PPL(↓)	FLOPs(↓)	PPL(↓)	FLOPs(↓)
SAMSum	3.33	0.55	3.32	0.55	3.18	0.55	3.29	0.55	3.35	0.55
Metrics	Acc.(↑)	FLOPs(↓)	Acc.(↑)	FLOPs(↓)	Acc.(↑)	FLOPs(↓)	Acc.(↑)	FLOPs(↓)	Acc.(↑)	FLOPs(↓)
SIQA	0.80	0.53	0.81	0.55	0.82	0.53	0.81	0.55	0.80	0.55

Reserved Tokens (m). We conduct experiments to analyze the number m of initial tokens reserved in our proposed eviction policy. For a fair comparison, we set the computational cost of all experiments at the same level with $\Omega = 0.5$. We compare reserved tokens of 0, 1, 2, 4, and 8 on SAMSum and SIQA datasets. As shown in Tab. 3, using about 55% computational cost of entire LLMs, the perplexity decreases at first and then increases. The result demonstrates that keeping the first two tokens is the best setting. Similarly, it shows the same tendency on SIQA, and the inflection point appears at the same time. As a result, we keep the first two tokens without skipping over layers.

4.4 Qualitative Analysis.

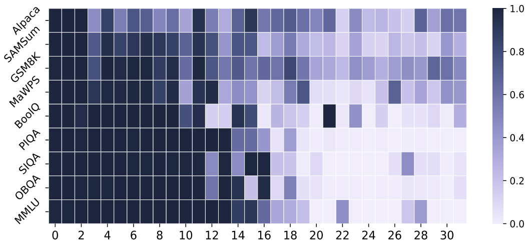
Execution Layers of Grammatical Terms. Here, we present a visualization of the execution ratios of different grammatical terms as shown in Fig. 3. From this figure, we observe that executing ratios of number and math symbols are much higher than others on layer #4 and layer #26. Meanwhile, layer #21 and layer #23 are utilized more by modal verbs and subject terms respectively. All terms show high execution ratios on the layer #18. Since different grammatical terms usually play the same role in sentences, thus, their execution ratios are similar. More importantly, different layers in D-LLMs are of different abilities, which are used by different grammatical terms.



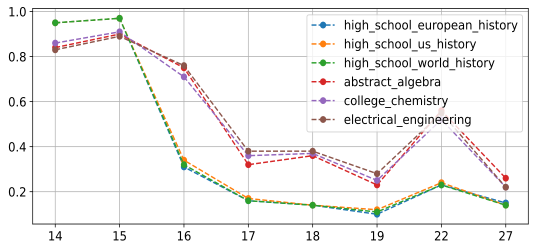
(a) Ratio on layer 4. (b) Ratio on layer 18. (c) Ratio on layer 21. (d) Ratio on layer 23. (e) Ratio on layer 26.

Figure 3: Execution ratios on different layers of three grammatical terms. Blue dots refer to number and math symbols, *e.g.*, '1, +, ×'. Red dots refer to subject terms, *e.g.*, 'She, He, They'. Green dots refer to modal verbs, *e.g.*, 'may, should, might'. The distance of a dot to the center represents executing ratio. The red circle is probability of 100% and the blue circle is the average ratios.

Execution Decisions of Similar Tasks. We count up and visualize the executing ratios of different layers to illustrate the difference on Benchmarks and Questions. As shown in Fig. 4a, the first few layers are utilized by most Benchmarks. Q&A and summarization tasks and math solving show larger diversity than common sense reasoning. Common sense reasoning tasks utilize more shallow layers and show smaller variances. In addition, we visualize several questions about high school history, science, and engineering knowledge from MMLU. As shown in Fig. 4b, the execution ratios of historical knowledge are similar, while that of science and engineering knowledge are similar. In conclusion, the execution behavior of utilizing similar knowledge tends to adopt similar ratios in D-LLM.



(a) Execution ratios of layers on Benchmarks.



(b) Execution ratios of layers on Questions.

Figure 4: The execution ratios of different layers visualizations on Benchmarks and Questions. Fig. 4a shows the execution ratios of different benchmarks on respective layers. A deeper color refers to a higher execution ratios. Fig. 4b shows six standard questions' execution ratios over layers from MMLU. The Y-axis is the execution ratios and X-axis is the layer index.

5 Conclusion

In this paper, we take the inference of large language models from a new perspective, that tasks of different difficulties should not require the same amount of computing resources and not every word is equally important in a sentence. Motivated by this, we propose a novel dynamic inference framework for LLMs, called D-LLMs, which achieves adaptive computing resource allocation for different tokens. We first propose a dynamic decision module to decide whether a layer is necessary for the given token. Then, we propose a KV-cache eviction strategy to enable D-LLMs' deployment in real-world applications. Finally, we conduct experiments on nine benchmarks and show D-LLMs performing comparable accuracy with only about 50% computational cost over baselines. More importantly, the D-LLMs can be deployed on almost all LLMs with negligible training cost.

Limitation. Our proposed D-LLMs make the first attempt to adaptively allocate computing resources at inference time for LLMs. Although experiments show surprising performance with lower cost, there are still some interesting issues worthy of future research. First, the generalizability of D-LLMs for broader tasks still needs verification. Second, it is worth exploring the dynamic mechanism on different granularities, *e.g.*, sentences and tasks.

References

[1] Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In arXiv:2012.13255, 2020.

[2] Sotiris Anagnostidis, Dario Pavllo, Luca Biggio, Lorenzo Noci, Aurelien Lucchi, and Thomas Hofmann. Dynamic context pruning for efficient and interpretable autoregressive transformers. In Adv. Neural Inform. Process. Syst., 2024.

[3] Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. Structured pruning of deep convolutional neural networks. In Emerg. Techn. Comput. Syst., 2017.

[4] Haoli Bai, Wei Zhang, Lu Hou, Lifeng Shang, Jing Jin, Xin Jiang, Qun Liu, Michael Lyu, and Irwin King. Binarybert: Pushing the limit of bert quantization. In arXiv:2012.15701, 2020.

[5] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv preprint arXiv:1308.3432, 2013.

[6] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In Proc. AAAI Conf. Artif. Intell., 2020.

[7] Lester Brian, Al-Rfou Rami, and Constant Noah. The power of scale for parameter-efficient prompt tuning. In arXiv:2104.08691, 2021.

[8] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In NAACL, 2019.

[9] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. In arXiv:2110.14168, 2021.

[10] Xiangxiang Dai, Jin Li, Xutong Liu, Anqi Yu, and John Lui. Cost-effective online multi-llm selection with versatile reward models. arXiv preprint arXiv:2405.16587, 2024.

[11] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. In Adv. Neural Inform. Process. Syst., 2023.

[12] Harry Dong, Xinyu Yang, Zhenyu Zhang, Zhangyang Wang, Yuejie Chi, and Beidi Chen. Get more with less: Synthesizing recurrence with kv cache compression for efficient llm inference. In arXiv:2402.09398, 2024.

[13] Shichen Dong, Wen Cheng, Jiayu Qin, and Wei Wang. Qaq: Quality adaptive quantization for llm kv cache. In arXiv:2403.04643, 2024.

[14] Xin Luna Dong, Seungwhan Moon, Yifan Ethan Xu, Kshitiz Malik, and Zhou Yu. Towards next-generation intelligent assistants leveraging llm techniques. In Proc. Knowl. Disc. Data Mining, 2023.

[15] Ali Edalati, Marzieh Tahaei, Ivan Kobyzev, Vahid Partovi Nia, James J Clark, and Mehdi Rezagholizadeh. Krόna: Parameter efficient tuning with kronecker adapter. In arXiv:2212.10650, 2022.

[16] Ahmad Faiz, Sotaro Kaneda, Ruhan Wang, Rita Chukwunyere Osi, Prateek Sharma, Fan Chen, and Lei Jiang. LLMCarbon: Modeling the end-to-end carbon footprint of large language models. In Int. Conf. Learn. Represent., 2024.

[17] Siqi Fan, Xin Jiang, Xiang Li, Xuying Meng, Peng Han, Shuo Shang, Aixin Sun, Yequan Wang, and Zhongyuan Wang. Not all layers of llms are necessary during inference. In arXiv:2403.02181, 2024.

[18] Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. Depgraph: Towards any structural pruning. In IEEE Conf. Comput. Vis. Pattern Recog., 2023.

[19] Peng Gao, Jiaming Han, Renrui Zhang, Ziyi Lin, Shijie Geng, Aojun Zhou, Wei Zhang, Pan Lu, Conghui He, Xiangyu Yue, Hongsheng Li, and Yu Qiao. Llama-adapter v2: Parameter-efficient visual instruction model. In arXiv:2304.15010, 2023.

[20] Xitong Gao, Yiren Zhao, ukasz Dudziak, Robert Mullins, and Cheng zhong Xu. Dynamic channel pruning: Feature boosting and suppression. In Int. Conf. Learn. Represent., 2019.

[21] Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive kv cache compression for LLMs. In Int. Conf. Learn. Represent., 2024.

[22] Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. SAMSum corpus: A human-annotated dialogue dataset for abstractive summarization. In Proc. Worksh. New Front. Summariz., 2019.

[23] Mitchell A Gordon, Kevin Duh, and Nicholas Andrews. Compressing bert: Studying the effects of weight pruning on transfer learning. In arXiv:2002.08307, 2020.

[24] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In Int. Conf. Learn. Represent., 2023.

[25] Qin Guanghui and Jason Eisner. Learning how to ask: Querying lms with mixtures of soft prompts. In arXiv:2104.06599, 2021.
[26] Chi Han, Qifan Wang, Hao Peng, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. Lm-infinite: Zero-shot extreme length generalization for large language models. In Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), 2024.
[27] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. In arXiv:2110.04366, 2021.
[28] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In Int. Conf. Learn. Represent., 2021.
[29] Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length llm inference with kv cache quantization. In arXiv:2401.18079, 2024.
[30] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In Int. Conf. Learn. Represent., 2022.
[31] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q Weinberger. Multiscale dense networks for resource efficient image classification. In Int. Conf. Learn. Represent., 2018.
[32] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In Int. Conf. Learn. Represent., 2017.
[33] Marek Kadlík, Michal tefánik, Ondej Sotolá, and Vlastimil Martinek. Calc-x and calcformers: Empowering arithmetical chain-of-thought through interaction with symbolic systems. In Proc. Conf. Empir. Meth. Natural Langu. Process., 2023.
[34] Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. In Adv. Neural Inform. Process. Syst., 2021.
[35] 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. In arXiv:2402.02834, 2024.
[36] Jeonghoon Kim, Jung Hyun Lee, Sungdong Kim, Joonsuk Park, Kang Min Yoo, Se Jung Kwon, and Dongsoo Lee. Memory-efficient fine-tuning of compressed large language models via sub-4-bit integer quantization. In Adv. Neural Inform. Process. Syst., 2023.
[37] Aaron Klein, Jacek Golebiowski, Xingchen Ma, Valerio Perrone, and Cedric Archambeau. Structural pruning of large language models via neural architecture search. In AutoML Conference 2023 (Workshop), 2023.
[38] Eldar Kurtic, Daniel Campos, Tuan Nguyen, Elias Frantar, Mark Kurtz, Benjamin Fineran, Michael Goin, and Dan Alistarh. The optimal bert surgeon: Scalable and accurate second-order pruning for large language models. In arXiv:2203.07259, 2022.
[39] Lei Li, Yankai Lin, Shuhuai Ren, Peng Li, Jie Zhou, and Xu Sun. Dynamic knowledge distillation for pre-trained language models. In arXiv:2109.11295, 2021.
[40] Shiyang Li, Jianshu Chen, Yelong Shen, Zhiyu Chen, Xinlu Zhang, Zekun Li, Hong Wang, Jing Qian, Baolin Peng, Yi Mao, et al. Explanations from large language models make small reasoners better. In arXiv:2210.06726, 2022.
[41] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In arXiv:2101.00190, 2021.
[42] Yanjing Li, Sheng Xu, Xianbin Cao, Xiao Sun, and Baochang Zhang. Q-DM: An efficient low-bit quantized diffusion model. In Adv. Neural Inform. Process. Syst., 2023.
[43] Yixiao Li, Yifan Yu, Qingru Zhang, Chen Liang, Pengcheng He, Weizhu Chen, and Tuo Zhao. LoSparse: Structured compression of large language models based on low-rank and sparse approximation. In Int. Conf. Mach. Learn., 2023.
[44] Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In Adv. Neural Inform. Process. Syst., 2022.
[45] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. In arXiv:2402.09353, 2024.
[46] Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. In arXiv:2110.07602, 2021.
[47] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. Gpt understands, too. In AI Open, 2023.
[48] Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, and Beidi Chen. Contextual sparsity for efficient LLMs at inference time. In Int. Conf. Mach. Learn., 2023.
[49] Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. In arXiv:2402.02750, 2024.

[50] Ruilong Ma, Jingyu Wang, Qi Qi, Xiang Yang, Haifeng Sun, Zirui Zhuang, and Jianxin Liao. Pipellm: Pipeline llm inference on heterogeneous devices with sequence slicing. In Proc. SIGCOMM Conf., 2023.

[51] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. In Adv. Neural Inform. Process. Syst., 2023.

[52] 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. In arXiv:2403.03853, 2024.

[53] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In Conf. Empir. Meth. Natur. Lang. Process., 2018.

[54] Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng Xin Yong, Hailey Schoelkopf, Xiangru Tang, Dragomir Radev, Alham Fikri Aji, Khalid Almubarak, Samuel Albanie, Zaid Alyafeai, Albert Webson, dward Raff, and Colin Raffel. Crosslingual generalization through multitask finetuning. In Proc. Assoc. Comput. Ling., 2023.

[55] openai. Chatgpt. In chat.openai.com, 2023.

[56] 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, 2023.

[57] Haotong Qin, Mingyuan Zhang, Yifu Ding, Aoyu Li, Zhongang Cai, Ziwei Liu, Fisher Yu, and Xianglong Liu. Bibench: benchmarking and analyzing network binarization. In Int. Conf. Mach. Learn., 2023.

[58] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. In GitHub.com, 2019.

[59] 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. In arXiv:2404.02258, 2024.

[60] Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. Socialiqa: Commonsense reasoning about social interactions. In arXiv: arXiv 1904.09728, 2019.

[61] Armin Sarabi, Tongxin Yin, and Mingyan Liu. An llm-based framework for fingerprinting internet-connected devices. In Proc. Intern. Measur. Conf., 2023.

[62] Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. Omniquant: Omnidirectionally calibrated quantization for large language models. In Int. Conf. Learn. Represent., 2024.

[63] Benjamin Frederick Spector and Christopher Re. Accelerating LLM inference with staged speculative decoding. In Int. Conf. Mach. Learn., 2023.

[64] Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for bert model compression. In arXiv:1908.09355, 2019.

[65] Siqi Sun, Zhe Gan, Yu Cheng, Yuwei Fang, Shuohang Wang, and Jingjing Liu. Contrastive distillation on intermediate representations for language model compression. In arXiv:2009.14167, 2020.

[66] Zhiqing Sun, Yikang Shen, Qinhong Zhou, Hongxin Zhang, Zhenfang Chen, David Cox, Yiming Yang, and Chuang Gan. Principle-driven self-alignment of language models from scratch with minimal human supervision. In Adv. Neural Inform. Process. Syst., 2023.

[67] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. Lst: Ladder side-tuning for parameter and memory efficient transfer learning. In Adv. Neural Inform. Process. Syst., 2022.

[68] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Stanford alpaca: An instruction-following llama model. In GitHub.com, 2023.

[69] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971, 2023.

[70] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In Eur. Conf. Comput. Vis., 2018.

[71] Huanyu Wang, Songyuan Li, Shihao Su, Zequn Qin, and Xi Li. Rdi-net: Relational dynamic inference networks. In Int. Conf. Comput. Vis., 2021.

[72] Huanyu Wang, Zequn Qin, Songyuan Li, and Xi Li. Codinet: Path distribution modeling with consistency and diversity for dynamic routing. In IEEE Trans. Pattern Anal. Mach. Intell., 2022.

[73] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E. Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In Eur. Conf. Comput. Vis., 2018.

[74] Yaqing Wang, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. Adamix: Mixture-of-adapter for parameter-efficient tuning of large language models. In arXiv:2205.12410, 2022.

[75] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In Int. Conf. Learn. Represent., 2023.

[76] Z Yao, RY Aminabadi, M Zhang, X Wu, C Li, and Y Zeroquant He. Efficient and affordable post-training quantization for large-scale transformers, 2022. In arXiv:2206.01861, 2022.

[77] Deming Ye, Yankai Lin, Yufei Huang, and Maosong Sun. Tr-bert: Dynamic token reduction for accelerating bert inference. In arXiv:2105.11618, 2021.

[78] Shuzhou Yuan, Ercong Nie, Bolei Ma, and Michael Färber. Why lift so heavy? slimming large language models by cutting off the layers. In arXiv:2402.11700, 2024.

[79] Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. Q8bert: Quantized 8bit bert. In Adv. Neural Inform. Process. Syst. Worksh., 2019.

[80] Renrui Zhang, Jiaming Han, Chris Liu, Peng Gao, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, and Yu Qiao. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. In Int. Conf. Learn. Represent., 2024.

[81] Tianyun Zhang, Shaokai Ye, Kaiqi Zhang, Jian Tang, Wujie Wen, Makan Fardad, and Yanzhi Wang. A systematic dnn weight pruning framework using alternating direction method of multipliers. In Eur. Conf. Comput. Vis., 2018.

[82] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Re, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In Adv. Neural Inform. Process. Syst., 2023.

[83] Juntao Zhao, Borui Wan, Chuan Wu, Yanghua Peng, and Haibin Lin. Llm-pq:serving llm on heterogeneous clusters with phase-aware partition and adaptive quantization. In Proc. Annu. Sympos. Princip. Pract. Program., 2024.

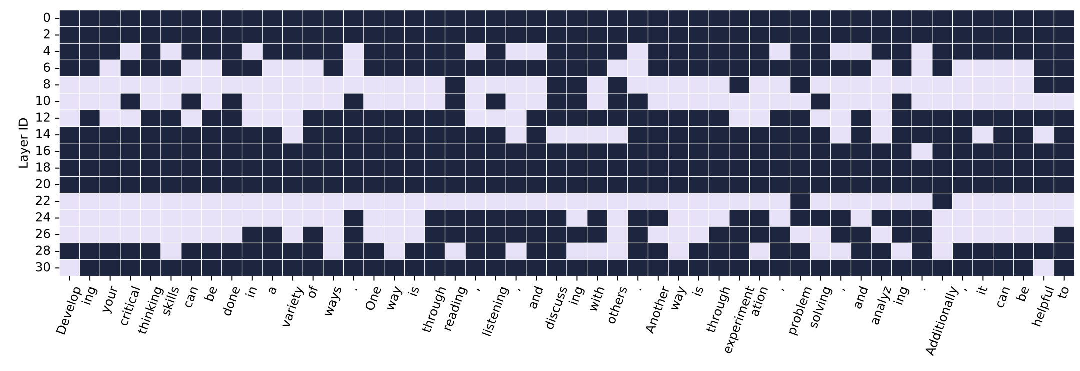
[84] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. MiniGPT-4: Enhancing vision-language understanding with advanced large language models. In Int. Conf. Learn. Represent., 2024.

A Appendix

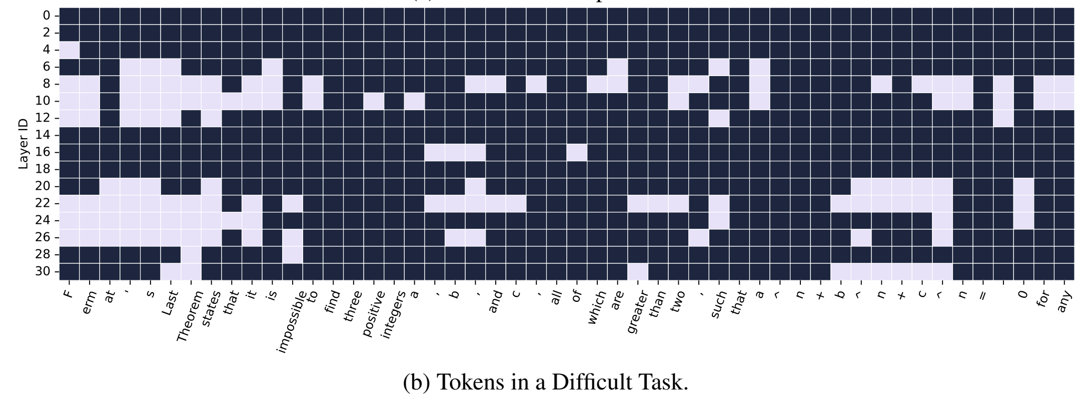
The Appendix is organized as follows. In Sec.A.1, we visualize the execution decisions over layers of tasks of different difficulties. In Sec.A.2, we show the execution ratios of each layer under different computational cost. In Sec.A.3, we present the concrete accuracy and perplexity under different cost. In Sec.A.4, we show the accuracy and computational cost under different hyper-parameter α. In Sec.A.5, we provide information about extra overhead taken by our proposed decision modules in D-LLMs. In Sec.A.6, we elaborate details on the reproduction of state-of-the-art methods and implementation of our D-LLMs. In Sec.A.7, we present the comparisons of generated results between D-LLMs and LLaMA with LoRA. In Sec.A.8, we list the licenses for all assets used in this paper.

A.1 Visualization on Tasks of Different Difficulties.

To demonstrate the key motivation *'models should not allocate the same computing resource on tasks of different difficulties or tokens of different importance.'*, we illustrate the transformer layers to be executed for tasks of different difficulties in D-LLMs as shown in Fig. 5. We compare the execution decisions over layers on two questions. The first question is *"How can I develop my critical thinking skills?"*, which is referred as a simple one. The second question is *"Can you explain Fermat's Last Theorem?"*, which is referred as a difficult one. From Fig. 5, we can observe that the simple question utilizes fewer tokens than the different question. Moreover, the skipping decisions usually happen on upper half and lower half, while the execution decisions happen on initial layers, middle layers, and last layers.



(a) Tokens in a Simple Task.

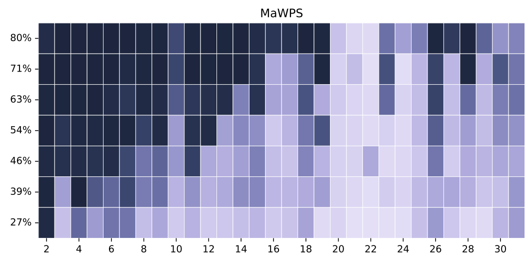


(b) Tokens in a Difficult Task.

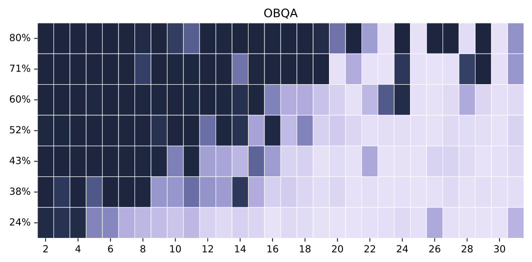
Figure 5: The transformer layers to be executed for tasks of different difficulties in D-LLMs. fig. 5a is the execution decisions, when D-LLM answers *"How can I develop my critical thinking skills?"*. fig. 5b is the execution decisions, when D-LLM answers *"Can you explain Fermat's Last Theorem?"*. The second question is more difficulty than the first one, therefore, utilizing more transformer layers. A filled block refers to a token executing the corresponding layer, while an empty block refers to a token skipping the corresponding layer.

A.2 Execution Ratios under Different Cost.

We visualize the executing ratios of layers under different computational cost in Fig. 6. As one can observe, with the reduced computational cost, the skipped layers are usually from the last layers. When we only utilize about 25% computational cost, most of the layers are skipped. Only the first three layers are executed by most tokens. This phenomenon illustrates that just a few layers are enough for most of the tokens' understanding and generation. Only a few tokens require more computational resources.



(a) Execution ratios under different cost on MaWPS.



(b) Execution ratios under different cost on OBQA.

Figure 6: Execution Decisions over Layers under different ratios on MaWPS and OBQA. The vertical coordinate is the computational cost. The horizontal coordinate is the indices of layers. A dark color refers to most tokens execute the layer, while a light color refers to most tokens skip the layer.

A.3 Performance and Cost Trade-off.

With the customizable acceleration rate loss, we adjust the user-defined target acceleration rate Ω to achieve performance under different FLOPs on three datasets. The concrete performance in terms of accuracy and computational cost is shown in Tab. 4. Under extreme conditions, our proposed D-LLMs obtain 57% and 77% accuracy with less than 25% percentage of computational cost on MaWPS and OBQA, respectively. Specifically, the performance ranges from 57% to 73% under computational cost from 0.28 to 0.80 on MaWPS. And it ranges from 24% to 81% under computational cost from 0.24 to 0.81. In terms of SAMSum, our method achieves at least 4.09 PPL with 76% computational cost reduction.

Table 4: The accuracy against computational cost on MaWPS, OBQA, and SAMSum datasets.

MaWPS		OBQA		SAMSum	
Acc.(↑)	FLOPs(↓)	Acc.(↑)	FLOPs(↓)	PPL(↓)	FLOPs(↓)
0.73	0.80	0.81	0.81	3.44	0.81
0.72	0.72	0.81	0.72	3.25	0.72
0.74	0.64	0.80	0.61	3.17	0.64
0.74	0.55	0.80	0.52	3.18	0.56
0.72	0.47	0.81	0.44	3.23	0.48
0.73	0.40	0.80	0.39	3.50	0.38
0.71	0.36	0.80	0.34	3.73	0.32
0.57	0.28	0.77	0.24	4.09	0.24

A.4 Hyper-parameter Analysis.

We use hyper-parameter α as the factor of $\mathcal{L}_{rate}$ in loss function Eq.14 to control the importance of acceleration ratio loss during training. α influences the performance on specific tasks and the ability to control the acceleration rate in D-LLMs. We perform parameter analysis on the MaWPS. In Tab. 5, the results show that larger α provides better control over the acceleration rate. For example, with the target acceleration ratio Ω set to 80%, the trained model with $\alpha = 0.1$ achieves only 47%, whereas

$\alpha = 5$, D-LLM achieves 72%. In addition, an excessively large α can decrease the performance of D-LLM, even when trained models share the same computation cost. For example, comparing cases on ($\Omega = 0.7, \alpha = 1$) with cases on ($\Omega = 0.6, \alpha = 5$), trained D-LLMs both achieve approximately 60% acceleration. However, the accuracy in $\alpha = 5$ decreases by 11% compared to the $\alpha = 1$ case.

Table 5: The accuracy and computational cost against hyper-parameter α on MaWPS dataset.

	$\alpha = 0.1$		$\alpha = 1$		$\alpha = 10$	
Ω	Acc.(↑)	FLOPs(↓)	Acc.(↑)	FLOPs(↓)	Acc.(↑)	FLOPs(↓)
0.5	0.75	0.63	0.74	0.56	0.62	0.52
0.6	0.73	0.57	0.72	0.47	0.62	0.42
0.7	0.75	0.56	0.73	0.40	0.61	0.34
0.8	0.73	0.53	0.71	0.36	0.57	0.28
0.9	0.72	0.49	0.70	0.33	0.50	0.22

A.5 Overhead of Decision Modules.

We provide information about extra overhead taken by decision modules of D-LLM in Tab.6. The base LLM is LLaMA2-7B.

Params/FLOPs overheads. Decision modules are parameter-efficient, which only take 1.0% of the size of parameters of LLM and 0.9% FLOPs compared with the forward computation of LLM.

Training memory overheads. We store parameters to be updated in float32 and others in bfloat16 during training. The running GPU memory usage of training LoRA on LLM is about 26.2 GB and training D-LLM is about 32 GB, which means the extra memory cost taken by decision modules is only 5.8 GB.

Inference memory overheads. During inference, the additional GPU memory cost taken by decision modules is only 0.3 GB. Furthermore, D-LLM requires less memory when generating each token since layers decided to be skipped are unnecessary to load into GPU memory. For example, D-LLM with an acceleration rate 50% requires only 7.4 GB when inference, significantly less than the 13.9 GB needed by an origin LLM using all layers.

Latency. To the latency of inference, computing through one decision module costs only 0.1 ms, which is quite lightweight compared with computing through a transformer block, which costs about 1.2 ms. The gain of acceleration brought by layer skipping is significant.

Table 6: Overhead information of decision modules in D-LLMs.

	Params	FLOPs	Training(GB)	Inference(GB)	Latency/Block(ms)
Decision Modules	0.9%	0.8%	5.8	0.3	0.1
LLM	1.0	1.0	26.2(LoRA)	13.9	1.2

A.6 Experimental Details

Reproduce of State-of-the-art. *Shortened-LLaMA.* The pruning strategy on Shortened-LLaMA is based on Taylor and PPL metrics. According to authors' implementation, the number of pruning layers is set to 11. Taylor-based Shortened-LLaMA prunes layers [27,24,26,25,29,28,23,22,21,20,19] and PPL-based Shortened-LLaMA prunes layers [24,26,25,10,27,13,22,14,9,29,12].

Ada-Infer. Ada-Infer is trained on LoRA-based pretrained LLMs. We use Gap and Top Prob as the input feature and reproduce the classifier with two linear layers of 512 hidden dimension.

Mixture-of-depth. MoD selects top-k tokens for calculation at dynamic layers. According to their paper, we deploy dynamic selections on even layers and the number of k is set to 12.5% of the length of sentences. We reproduce the classifier with a linear layer and incorporate the same LoRA module for finetuning.

Computational resources. We conduct experiments on a server with 8 NVIDIA L20 GPUs. Models are implemented in Pytorch with a AdamW optimizer. It takes about 0.07s every step at the training phase. The learning rate is set to 9×10^{-3}. Furthermore, we warm up training in the first 2 epochs and then utilize cosine annealing schedule.

A.7 Comparison of Generated Results.

Instruction: Give students tips on how to keep their nerves under control during class presentations.

D-LLMs on LLaMA2-7B: *(58% FLOPs)*
1. Practice your presentation in front of a mirror or with a friend.
2. Take deep breaths and relax before the presentation.
3. Make sure to stay focused on the task and don't let your nerves get the better of you.
4. Make sure to smile and stay positive throughout the presentation.
5. Don't be afraid to make mistakes or stumble.
6. Make sure to thank your teacher and classmates for their support.
7. Ask questions and listen to your teachers feedback.
8. Don't be afraid to ask questions if you don't understand something.

LLaMA2-7B + LoRA: *(100% FLOPs)*
1. Practice presenting your material in front of a mirror or with a friend.
2. Make sure to breathe deeply and relax your body before the presentation.
3. Visualize success and remind yourself of your strengths.
4. Speak slowly and clearly, and make sure to pause for a few seconds between each point.
5. Smile and make eye contact with the audience.
6. Be confident and don't be afraid to ask for help if needed.

Instruction: What is Gremolata?

D-LLMs on LLaMA2-7B: *(61% FLOPs)*
Gremolata is a classic Italian condiment made from garlic, lemon, and parsley. It is traditionally served with grilled meats, such as steak, chicken, and fish, and is often added to pasta dishes, soups, and salads.

LLaMA2-7B + LoRA: *(100% FLOPs)*
Gremolata is a popular Italian condiment made from garlic, parsley, and lemon zest. It is often used as a topping for pasta dishes, but can also be used as a side dish or as a topping for grilled meats.

Instruction: Please, summarise the book "Harry Potter and the Deathly Hallows" in two paragraphs.

D-LLMs on LLaMA2-7B: *(60% FLOPs)*
Harry Potter and the Deathly Hallows is a story of a young wizard and his quest to defeat the evil Lord Voldemort. The book follows Harry Potter as he and his friends embark on a dangerous journey to find the Deathly Hallows, the legendary objects that will grant Harry the power of defeating the evil Lord Voldemort. Along the way, Harry and his friends face many challenges and make difficult choices, but ultimately, they are able to defeat the evil Lord Voldemort and save the wizarding world.

LLaMA2-7B + LoRA: *(100% FLOPs)*
Harry Potter and the Deathly Hallows is the seventh and final book in the Harry Potter series. It follows Harry Potter and his friends as they embark on a quest to find and destroy the Horcruxes, powerful objects that contain the essence of Lord Voldemort. Along the way, they face many challenges and dangers, including a powerful wizard who is determined to stop them. In the end, Harry and his friends succeed in their quest and destroy the Horcruxes, saving the wizarding world from Lord Voldemort.

Instruction: Cody eats three times as many cookies as Amir eats. If Amir eats 5 cookies,how many cookies do both of them eat together?

D-LLMs on LLaMA2-7B: *(55% FLOPs)*
Cody eats 3 * 5 = «3*5=15»15 cookies.
Together, they eat 5 + 15 = «5+15=20»20 cookies.

LLaMA2-7B + LoRA: *(100% FLOPs)*
If Amir eats 5 cookies, Cody eats 3*5 = «5*3=15»15 cookies
Together, they eat 15+5 = «15+5=20»20 cookies

Instruction: Mike needed a new pair of jeans. When he got to the mall he saw that his favorite jeans were advertised 25% off. The original price of the jeans was \$40. How much money will Mike have left over if he pays with a \$50.00 bill?

D-LLMs on LLaMA2-7B: *(62% FLOPs)*
The jeans are on sale for 25% off so they are on sale for 100% - 25% = 75%
The original price of the jeans was \$40.00 and they are on sale for 75% so he will pay 40*.75 = \$«40*.75=30.00»30.00
Mike has \$50.00 and he will pay \$30.00 for the jeans so that leaves him with 50-30 = \$«50-30=20.00»20.00

LLaMA2-7B + LoRA: *(100% FLOPs)*
The jeans are \$40.00 and he can get 25% off, so 40*.25 = \$«40*.25=10.00»10.00 discount
If the jeans are \$40.00 and he gets \$10.00 off then he will pay 40-10=\$«40-10=30.00»30.00 for the jeans.
If he pays \$30.00 and has \$50.00 then 50-30 = \$«50-30=20.00»20.00 left over.

A.8 License of Assets.

We do not utilize any close-source assets in this paper. The LLaMA2-7B and LLaMA3-8B can be downloaded from `https://llama.meta.com/llama-downloads` and the licenses are from `https://llama.meta.com/llama2/license/`, `https://llama.meta.com/llama3/license/`. All benchmarks are obtained from huggingface. The license of Alpaca, SAMSum is CC BY-NC 4.0. The license of for BoolQ is Creative Commons Share-Alike 3.0. The license of PIQA is Academic Free License v3. And license for MaWPS, MMLU is MIT acknowledge.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We clearly state the claims and accurately introduce the contributions and scope in abstract and introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations are discussed in Sec.5 of the manuscript.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: D-LLMs is a novel inference framework and do not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We have clearly elaborated the method in Sec.3 and specify the experimental details in Sec.4.1 of the manuscript.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We have provided our code at `https://github.com/Jyk-122/D-LLM`, and will release all experimental results.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so No is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All experimental details and settings are elaborated in Sec 4.1 in manuscript and Sec 1.4 in appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We have reported the statistical significance of performance against cost under different settings, all numerical results are provided in Sec 4 in the manuscript and Sec 1.2 in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We have presented the information of computing resources in Sec 1.4 of the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: The proposed D-LLMs is an inference framework for LLM, thus, has no negative societal impact.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The proposed method do not involve such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have stated the licenses of assets used in Sec 1.6 of the appendix.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We will release documentation along with code.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: We did not conduct crowdsourcing experiments and research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: We did not conduct crowdsourcing experiments and research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.