# AdaSkip: Adaptive Sublayer Skipping for Accelerating Long-Context LLM Inference

**Zhuomin He[1*†], Yizhen Yao[1*†], Pengfei Zuo[2*], Bin Gao[3†], Qinya Li[1‡], Zhenzhe Zheng[1], Fan Wu[1]**

[1]Shanghai Key Laboratory of Scalable Computing and Systems, Shanghai Jiao Tong University
[2]Huawei Cloud
[3]School of Computing, National University of Singapore
{dean_hzm, 1975275148}@sjtu.edu.cn, pengfei.zuo@huawei.com, bingao@comp.nus.edu.sg
{qinyali, zhengzhenzhe}@sjtu.edu.cn, fwu@cs.sjtu.edu.cn

## Abstract

Long-context large language models (LLMs) inference is increasingly critical, motivating a number of studies devoted to alleviating the substantial storage and computational costs in such scenarios. Layer-wise skipping methods are promising optimizations but rarely explored in long-context inference. We observe that existing layer-wise skipping strategies have several limitations when applied in long-context inference, including the inability to adapt to model and context variability, disregard for sublayer significance, and inapplicability for the prefilling phase. This paper proposes AdaSkip, an adaptive sublayer skipping method specifically designed for long-context inference. AdaSkip adaptively identifies less important layers by leveraging on-the-fly similarity information, enables sublayer-wise skipping, and accelerates both the prefilling and decoding phases. The effectiveness of AdaSkip is demonstrated through extensive experiments on various long-context benchmarks and models, showcasing its superior inference performance over existing baselines.

## Introduction

Recently, large language models (LLMs) evolve to support long-context inference (Xiao et al. 2024; Srivatsa et al. 2024; DeepSeek-AI et al. 2024) up to 1M (Liu et al. 2024; AI et al. 2024), unlocking more complex real-world applications such as personal agent (Park et al. 2023; Wang et al. 2024), document summarization (Wu et al. 2023), and coding assistance (Liu, Xu, and McAuley 2023; Bairi et al. 2023; Jimenez et al. 2024). Long-context inference introduces more computational and storage demands. It is crucial to reduce the inference cost for long sequences.

Layer-wise skipping strategies, as an emerging technology, show great promise to reduce the LLM inference cost and latency by omitting the execution of transformer layers at specific positions, e.g., early skipping (Del Corro et al. 2023; Zhu et al. 2024), periodic skipping (Liu, Meng, and Zhou 2024), and early exit (Varshney et al. 2023; Fan et al. 2024; Chen et al. 2024).

However, we observe that these layer-wise skipping strategies all have their limitations in taking effect in long-context inference due to the following reasons. First, existing layer-wise skipping strategies lead to a significant degradation in the generation quality due to predetermined fixed layers being skipped regardless of model and context variance. We observe that the importance distributions of transformer layers are different across models and contexts, and none of these strategies can perform consistently best across all models and contexts. Second, existing skipping strategies perform skipping at monolithic transformer layers which leads to suboptimal performance. We observe that the importance distributions of sublayers, i.e., attention and FFN modules, are independent. Moreover, in long-context inference, attention sublayers contribute significantly to inference latency (Tang et al. 2024; Jiang et al. 2024), highlighting the importance of prioritizing the skipping of more attention sublayers. Third, existing layer-wise skipping strategies are limited to the decoding phase, neglecting optimization of the prefilling phase in long-context inference, where the latency of the prefilling phase, i.e., time to first token (TTFT), imposes a significant burden on long-context inference latency.

To address the above limitations, we propose *AdaSkip*, an auto-adaptive, sublayer-wise skipping strategy tailored for long-context inference, which can benefit both the prefilling and decoding phases. Firstly, AdaSkip exploits on-the-fly similarity information during execution to adaptively identify the least important layers in different models, thereby improving the generation quality. Secondly, AdaSkip independently determines the importance distribution residing within sublayer modules like attention and FFN, enabling the sublayer-wise skipping. Finally, AdaSkip identifies the least important sublayers during both prefilling and decoding phases, significantly reducing the time and memory overhead of long-context scenarios.

In summary, our contributions are as follows:

1. We perform a comprehensive analysis of the importance distributions of various components including layer and sublayer modules across a range of different models. Based on the analysis, we present the limitations of the existing layer-wise skipping strategies in accelerating long-context inference.

*These authors contributed equally.
†Work done during their internship at Huawei Cloud.
‡Corresponding author.

2. We propose an auto-adaptive, sublayer-wise skipping strategy that works for both the prefilling and decoding phases in long-context scenarios.
3. We conduct extensive experiments on various long-context benchmarks and models, demonstrating AdaSkip exhibits favorable inference performance over baselines.

## Background and Motivation

In this section, we first perform a comprehensive exploration of the importance metric of the layer and sublayer-wise modules, then present observations on the characteristics of the importance distribution and motivate our design principles.

### IO Similarity and Transformer Module Importance

We first define the metric, *similarity*, to evaluate the importance of transformer layers and sublayer modules. Given two $n$-dimensional vectors, $\vec{a}$ and $\vec{b}$, we characterize the cosine similarity between these vectors as their similarity, defined as follows:

$$\textit{Similarity}(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|\|\vec{b}\|} = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}} \quad (1)$$

Following the existing works (Liu et al. 2023b; Jaiswal et al. 2024; Fan et al. 2024), the similarity between the input and output (IO) vectors of the transformer module, i.e., IO similarity, can be used to evaluate the importance of a transformer module. Specifically, following the forwarding of each module, if the input vector of the module closely resembles the output vector, it indicates that the module contributes minimally to the forward propagation process. In other words, the current module contributes less *importance* in terms of execution. Conversely, the current module possesses higher *importance* in terms of execution if the IO similarity is low.

We further empirically validate the correlation between the IO similarity and the importance of a transformer module. Given an inference task, we conduct a first-round inference process to profile the IO similarity of each transformer layer. Subsequently, we execute a second-round inference process that selectively skips the layers based on varying degrees of the profiled IO similarity. Then we assess the quality of generated output by evaluating its GPT score (Varshney et al. 2023; Jaiswal et al. 2024). The LeastSkip strategy, which skips the layers exhibiting the lowest IO similarity, experiences a substantial degradation in the GPT score (dropping below 1.0 even with one skipped layer), compared to the MostSkip strategy, which skips the layers with the highest IO similarity and yields GPT scores of 8.9, 6.1, and 4.2 when skipping 1, 3, and 5 layers, respectively.

### Existing Layer-wise Skipping Strategies

Existing layer-wise skipping strategies propose skipping fixed layers with certain preferences to reduce inference execution time. As shown in Figure 1, according to the strategies to skip layers, existing layer-wise skipping strategies can be broadly categorized into three types: early skipping (Del Corro et al. 2023), periodic skipping (Liu, Meng, and Zhou 2024), and early exit (Schuster et al. 2022; Varshney et al. 2023; Fan et al. 2024; Bae et al. 2023). Early skipping (Del Corro et al. 2023) always skips the first few layers that are predetermined. Early skipping can support batching operations but may skip the important layers. Periodic skipping (Liu, Meng, and Zhou 2024) periodically skips a few middle layers. It follows a predetermined frequency to skip one layer every several layers. Periodic skipping supports batching operations but cannot capture the varying importance of different layers. Early exit (Varshney et al. 2023; Fan et al. 2024) always skips the last few layers. It evaluates whether the conditions (e.g., confidence level) are met after finishing the computation of each layer and the execution immediately exits upon condition fulfillment. Early exit may overlook the important layers that come later. Moreover, existing early exit strategies need to pay additional efforts and costs to either train classifier (Del Corro et al. 2023) or fine-tune the model to counterbalance the information loss resulting from imperfect layer skipping (Liu, Meng, and Zhou 2024; Varshney et al. 2023; Fan et al. 2024).

### Motivation

This subsection analyzes the limitations of existing LLM acceleration strategies for long-context inference.

**Observation 1:** ***The layer importance distribution exhibits significant variation across diverse models.*** We follow the same way used in the previous section to investigate the IO similarities of different layers on various models, in both prefilling and decoding phases. Figure 2 shows significant variation in the IO similarities of transformer layers for different models in three long-context datasets. Taking InternLM-7B-8k and LLaMA3.1-8B-128k as examples, layers with high IO similarity in InternLM-7B-8k appear in the middle, such as layers 12,13,14, and the curve is more irregular. Whereas layers with high IO similarity in LLaMA3.1-8B-128k, appear towards the end, with layers 27, 25, 28, 29, and 26 being the top 5 layers, and the curve is approximately monotonically ascending. This suggests that layer importance distributions vary among different models. Existing layer-wise skipping strategies tend to consistently skip fixed layers, overlooking the differences in importance distribution across models, which restricts their adaptability to various models. Adaptive skipping strategies matching various models are required.

**Observation 2:** ***The importance distributions of attention and FFN modules are different.*** We study the IO similarities of the sublayer-wise modules, i.e., attention and FFN. As shown in Figure 3, the sublayer-wise modules show diverse IO similarity distributions. Taking LLaMA3.1-8B-128k as an example, in the last 11 layers, the average IO similarity of attention is consistently around 0.97, indicating a high IO similarity. However, the highest average IO similarity of FFN in the last 11 layers is only 0.95, and it is relatively scattered. Furthermore, compared to FFN, attention modules demonstrate higher and more concentrated similarity, implying that a greater number of attention modules can be skipped, with the potential to save more KV cache in long-context inference. The different characteristics in IO similarity distributions of attention and FFN suggest that the existing layer-wise skipping methodologies that monolithically skip

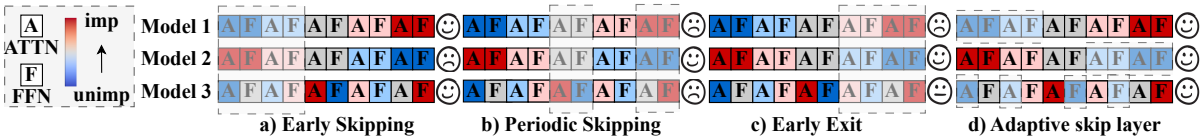
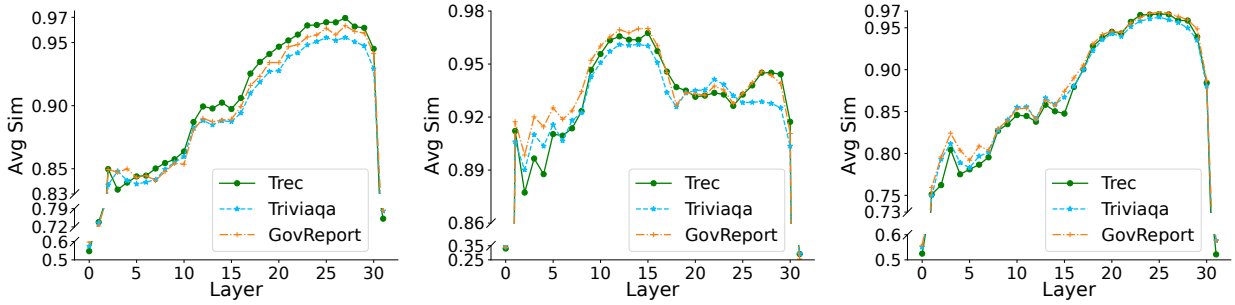



Figure 1: The comparisons of different skipping strategies. The dashed box indicates the layer to be skipped.




(a) Vicuna-7B-16k (b) InternLM-7B-8k (c) LLaMA3.1-8B-128k

Figure 2: IO similarities of different layers in various transformer models.

entire transformer layers are sub-optimal. Consequently, the attention sublayer and FFN sublayer within one transformer layer should be considered separately.

**Observation 3:** ***The importance distribution of sublayers in the prefilling and decoding phases have similar trends but different fluctuation degrees.*** We further investigate the IO similarities of sublayer modules in the prefilling and decoding phases respectively. As shown in Figure 4, both attention and FFN sublayers display a consistent IO similarity trend between the prefilling and decoding phases, indicating that similar skipping strategies can be shared between the two phases. What's more, we found a phenomenon that among all three models, each FFN sublayer has a higher IO similarity in the decoding phase than in the prefilling phase, which is different from that of attention sublayers. This suggests that we have the opportunity to skip more FFN sublayers in the decoding phase without affecting the model performance.

**Challenges.** Based on the above observations, an efficient skipping strategy for long-context inference should have the following capabilities: (1) adaptability to various models, (2) independent decision-making for sublayer-wise skipping, and (3) the ability to skip the most unimportant layers in both the prefilling and decoding phases.

However, implementing such a skipping strategy encounters several challenges. First, limited prior information is available to guide the skipping decisions throughout the prefilling phase. Second, distinguishing the unique information corresponding to specific models and contexts, required for making adaptive choices, is far from straightforward.

# Methodology

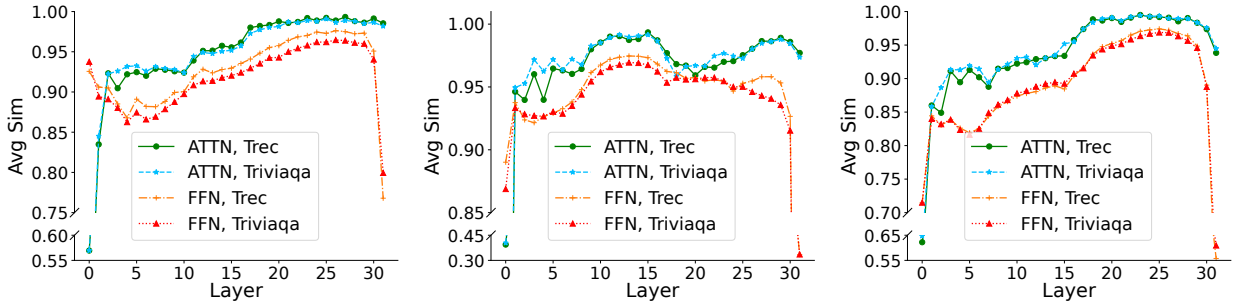
## Overview

To tackle the above challenges, we propose a novel skipping strategy for long-context inference, called AdaSkip, which adaptively selects sublayer-wise modules to skip considering the characteristics of models and inference context. Specifically, AdaSkip efficiently learns the importance distributions from the past inference execution to construct the skipping strategy for the prefilling phase. It further improves the skipping decision by online importance learning from on-the-fly intermediate data during the decoding phase. By integrating the above techniques, AdaSkip can accurately skip the least important sublayer-wise modules, avoiding the mismatch of layer importance and layer skipping decisions in fixed layer-wise skipping strategies.

## Sublayer Skipping during Prefilling with Offline Importance Learning

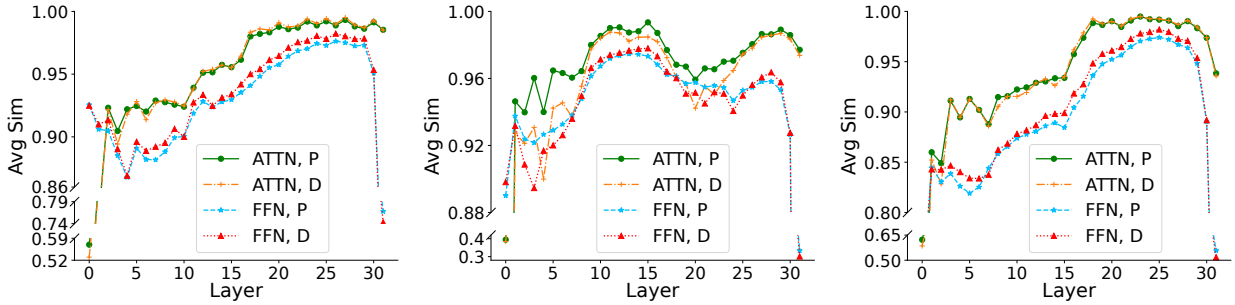
It is necessary to skip layers in prefilling phases during long-context inference, since the prefilling phase results in unacceptably high TTFT and substantial KV cache demands. However, existing layer-skipping strategies rarely consider skipping strategies in such phases. Moreover, since different models exhibit various similarity distributions, current fixed layer-skipping strategies cannot achieve optimal results. The primary obstacle in devising an adaptive sublayer-wise skipping approach for the prefilling phase lies in the absence of prior knowledge before execution. To address this challenge, we propose an offline importance learning method that leverages the high correlation between historical prefilling features and new prefilling features.

**Insight.** *Using sublayer-wise IO similarity feature from historical tasks can precisely predict the sublayer-wise skipping behavior for prefilling new inference tasks.* We perform the IO similarity analysis study of running inference tasks of multiple datasets (Taori et al. 2023) including 2WikiMQA, MultiFieldQA-en, and TriviaQA using LLaMA3.1-8B-128k and quantify the average hit rate of unimportant layers in the prefilling phase. We record the average IO similarity on the Src dataset in prefilling phases and test the hit rate on




(a) Vicuna-7B-16k (b) InternLM-7B-8k (c) LLaMA3.1-8B-128k

Figure 3: IO similarities of attention (ATTN) and FFN modules in different layers.




(a) Vicuna-7B-16k (b) InternLM-7B-8k (c) LLaMA3.1-8B-128k

Figure 4: IO similarities of sublayer modules in prefilling (P) and decoding (D) phases.

| Type | Src | Dest | Layer Hit Rate |
|---|---|---|---|
| ATTN | TriviaQA | MFieldQA | 3.76/4, 4.86/6, 9.31/10 |
| ATTN | MFieldQA | Wiki | 3.80/4, 5.54/6, 9.90/10 |
| ATTN | TriviaQA | Wiki | 3.79/4, 5.50/6, 9.68/10 |
| FFN | TriviaQA | MFieldQA | 3.66/4, 5.69/6, 9.56/10 |
| FFN | MFieldQA | Wiki | 3.77/4, 5.97/6, 9.38/10 |
| FFN | TriviaQA | Wiki | 3.75/4, 5.96/6, 9.64/10 |

Table 1: Average hit rate of unimportant layers using historical features across datasets in prefilling phases.

the Dest dataset. The results shown in Table 1 reveal that historical IO similarity in prefilling phases gains a high hit rate for subsequent tasks, suggesting that this feature can be used in prediction and shared across different datasets.

**Method.** Based on the insight, the major workflow of offline importance learning consists of the similarity study and the corresponding deviation correction procedure. Specifically, suppose $N$ inference tasks (samples) are used in offline importance learning. As for the inference task $T_i$ with prompt length $|T_i|$. Suppose that the model has $M$ transformer layers with $M$ attention sublayers and $M$ FFN sublayers. We first take notes of average similarity $\bar{Similarity}$ in the prefilling phase. The average similarity of the $j_{th}$ sublayer, $\bar{Similarity}_j$, can be accumulated as:

$$\bar{Similarity}_j = \frac{\sum_{i=1}^{N}\sum_{t=1}^{|T_i|} Similarity(\vec{a}_{it}^j, \vec{b}_{it}^j)}{\sum_{i=1}^{N}|T_i|} \quad (2)$$

where $\vec{a}_{it}^j$ and $\vec{b}_{it}^j$ are the input and output vectors of the $t$-th token in task $i$. In addition, if the angle between vector $\vec{a}_{it}^j$ and $\vec{b}_{it}^j$ is not very large, the proportion of modulus of $\vec{a}_{it}^j$ and $\vec{b}_{it}^j$ relatively become prominent, suggesting some compensation needs to be applied.

However, due to the residual connections employed between each sublayer, the modulus of the input and output of one layer has minor variations, which implies that the average proportion of modulus can effectively compensate for the deviations. Hence, we use the average proportion of historical modulus of $\vec{a}_{it}^j$ and $\vec{b}_{it}^j$ in $j_{th}$ layer to scale $\vec{a}_{it}^j$ so that output vector $\hat{\vec{b}}_{it}^j$ is close to original $\vec{b}_{it}^j$. The average scale factor of $j$-th sublayer, $\bar{Scale}_j$, can be formulated as:

$$\bar{Scale}_j = \frac{\sum_{i=1}^{N}\sum_{t=1}^{|T_i|}\frac{\|\vec{b}_{it}^j\|}{\|\vec{a}_{it}^j\|}}{\sum_{i=1}^{N}|T_i|} \quad (3)$$

we use $\bar{Scale}_j$ to compensate the input $\vec{a}_{it}^j$, getting approximate output:

$$\hat{\vec{b}}_{it}^j = \bar{Scale}_j * \vec{a}_{it}^j \quad (4)$$

After obtaining $\bar{Similarity}$ and $\bar{Scale}$ of each sublayer module, we sort all sublayers in descending order based on their $\bar{Similarity}$, getting the sorted list *sorted* with $2M$ elements. Since there is a trade-off between the number of skipped layers and the generation quality, we introduce an acceleration

| Dataset | Size | Layer Hit Rate |
|---|---|---|
| TREC | 5 | 0.84/2, 2.67/4, 4.70/6 |
| TREC | 20 | 1.08/2, 3.04/4, 4.90/6 |
| TREC | 40 | 1.07/2, 3.09/4, 4.90/6 |
| GovReport | 5 | 1.01/2, 2.94/4, 4.97/6 |
| GovReport | 20 | 1.14/2, 3.01/4, 5.02/6 |
| GovReport | 40 | 1.19/2, 3.03/4, 5.03/6 |

Table 2: Average hit rate of unimportant layers identified through different window sizes in the decoding phase.

ratio, $\alpha$, as a knob to control this trade-off. Given the acceleration ratio $\alpha$, the number of sublayers to be skipped, $m$, can be calculated as $m = M - \frac{M}{\alpha}$, and the targeted skipping sublayer number is $2m$. The top $2m$ sublayers in the *sorted* list are selected, forming the *skipped* set.

## Extra FFN Sublayer Skipping during Decoding with Online Importance Learning

Based on Observation 3, we find that regardless of attention or FFN sublayer, IO similarity of unimportant sublayers is always similar in prefilling and decoding phases, which suggests that we can reuse the layers selected from prefilling phases when decoding. What's more, we observe that each FFN sublayer has higher IO similarity in decoding phases compared with the prefilling phase, which inspires us to explore more FFN skipping opportunities in decoding phases. In a nutshell, AdaSkip explores more potential of FFN skipping in decoding phases through online importance learning, hoping to obtain a larger speedup without losing performance.

**Insight.** *The IO similarity information of the current context can be used to explore extra FFN skipping opportunities in decoding phases.* We find that using the initial few tokens in the decoding phase can well hit the important layers of subsequent inference, and the hit rate gradually increases with the increase of the initial window. We test LLaMA3.1-8B-128k on TREC and GovReport datasets (Bai et al. 2023). For each context, we use the initial $n$ tokens in decoding phases to calculate the average IO similarity, and then observe the hit rate for subsequent decoding of the current sequence. The results are shown in Table 2. As the window size $n$ increases, the hit rate increases and gradually becomes constant, suggesting it is unnecessary to increase $n$ infinitely.

**Method.** Based on the above insight, the major workflow of online importance learning mainly consists of the similarity learned from the decoding phase of the new inference task. Specifically, for the new context, we define the first $P$ decoded tokens as *online learning windows*. These tokens are processed with unskipped layers in order to obtain current decoding features. We denote the set of FFN sublayers to be skipped in decoding phases by $skipped^P$. For $j$-th sublayer, given the input and output vectors of $t$-th decoded token as $\vec{a}_t^j$ and $\vec{b}_t^j$, $\bar{Similarity}_j$ of $j_{th}$ FFN sublayer for current context from the first decoded token to $P_{th}$ tokens for FFN sublayers can be formulated as:

$$\bar{Similarity}_j^P = \frac{\sum_{t=1}^{P} Similarity(\vec{a}_t^j, \vec{b}_t^j)}{P} \quad (5)$$

We get all indexes of FFN sublayers, i.e. *index*, and the indexes of all the layers skipped in the prefilling phase, i.e., *skipped*. To find out which layers in *index* set need to be skipped, we derive a threshold $\beta$ by observing the *skipped* set and then use this threshold to filter the additional skipped FFN sublayers. The threshold $\beta$ is the least *Similarity* value in *skipped*, i.e. $\beta = \min\{\bar{Similarity}_j \mid j \in skipped\}$.

We then traverse *index* to find the sublayers whose $\bar{Similarity}_j^P$ is above $\beta$, and these sublayers are the additional ones to be skipped in the new context. By combining the indexes of these sublayers with the indexes of the *skipped* set, we obtain the adaptive sublayer-wise skipping set, denoted as $skipped^P$. At last, similar to the last section, we use $\bar{Scale}_j$ to compensate for the potential deviation.

# Experiments

In this section, we thoroughly evaluate the performance of AdaSkip in long-context inference. We first show the experiment settings including the benchmarks, baselines, and setups. Then we show the experiment results and analysis.

## Experiment Settings

**Benchmarks** We select benchmarks based on representative long-context application scenarios (Bai et al. 2024), encompassing document QA, few-shot learning, and summarization. To better evaluate different skipping strategies in prefilling and decoding phases, we divide the benchmarks into *prefilling tasks* and *decoding tasks* by average output length. We select MultiFieldQA (Bai et al. 2023), TriviaQA (Joshi et al. 2017), and TREC (Li and Roth 2002) as prefilling tasks, with average input lengths of 6493, 8677, and 8208, and output lengths capped at 32. For decoding tasks, we choose GovReport (Huang et al. 2021) and MultiNews (Fabbri et al. 2019), with average input lengths of 9214 and 8265, and output lengths limited to 512. We evaluate the end-to-end performance of all layer-wise skipping strategies by skipping layers in both prefilling and decoding phases.

**Baselines and Setups.** Three layer-wise skipping strategies are considered as baselines: (1) SkipDecode (Del Corro et al. 2023) skips the initial layers except for the first one, representing early skipping; (2) Unified Skipping (Liu, Meng, and Zhou 2024) uniformly skips the intermediate layers except for the first and last layers, representing periodic skipping; and (3) Early Exit (Varshney et al. 2023; Fan et al. 2024) skips the last few layers. Note that layer-wise skipping skips two sublayers, i.e., attention and FFN, in a single layer-skip operation. Specifically, as these baselines were originally designed to skip layers only during the decoding phase, we limit layer skipping to the decoding phase in decoding tasks to ensure a fair comparison. Three of the latest and widely adopted long-context LLMs are tested: LLaMA3.1-8B-128k, InternLM-7B-8k, and Vicuna-v1.5-7B-16k. A single L20 GPU with CUDA version 12.1 is used as the testbed.

## Results of Prefilling Tasks

The middle of Table 3 presents the results of the prefilling tasks. Given the same number of target skip sublayers,

| # Target Skip Sublayer | Theo. Speedup (SU) | Model | Skipping Strategy | Prefilling Layer-Skipping | | | | Decoding Layer-Skipping | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Doc QA | Few-shot Learning | | Actual SU | Text Summarization | | Actual SU |
| | | | | MFieldQA (F1) | TriviaQA (F1) | TREC (ACC) | | GovReport (Rouge-L) | MultiNews (Rouge-L) | |
| 0 | 1.00 | LLaMA3.1-8B-128k | Full Model | 29.7 | 91.6 | 75.0 | 1.00 | 34.2 | 25.8 | 1.00 |
| | | InternLM-7B-8k | Full Model | 26.6 | 70.4 | 50.4 | 1.00 | 18.2 | 17.6 | 1.00 |
| | | Vicuna-v1.5-7B-16k | Full Model | 32.9 | 87.8 | 68.9 | 1.00 | 27.2 | 22.4 | 1.00 |
| 8 | 1.14 | LLaMA3.1-8B-128k | Early Exit | 13.1 | 18.5 | 28.3 | **1.10** | 16.8 | 14.5 | 1.11 |
| | | | SkipDecode | 0.4 | 0.0 | 0.0 | **1.10** | 19.3 | 16.3 | 1.07 |
| | | | Unifed Skipping | 3.4 | 4.7 | 2.2 | **1.10** | 28.2 | 22.8 | 1.11 |
| | | | AdaSkip | **23.4** | **86.6** | **72.8** | 1.09 | **30.9** | **24.0** | **1.15** |
| | | InternLM-7B-8k | Early Exit | 6.1 | 28.2 | 32.8 | 1.13 | 3.1 | 3.6 | 1.12 |
| | | | SkipDecode | 0.0 | 0.0 | 0.0 | 1.13 | 11.0 | 10.6 | 1.08 |
| | | | Unifed Skipping | 15.4 | 21.1 | 13.3 | 1.13 | 9.8 | 10.1 | 1.12 |
| | | | AdaSkip | **23.9** | **60.3** | **42.7** | **1.25** | **13.7** | **13.3** | **1.24** |
| | | Vicuna-v1.5-7B-16k | Early Exit | 18.5 | 73.7 | 29.4 | 1.11 | 12.2 | 13.3 | 1.13 |
| | | | SkipDecode | 0.0 | 0.0 | 0.0 | 1.11 | 4.1 | 4.5 | 1.07 |
| | | | Unifed Skipping | 0.0 | 0.0 | 0.0 | 1.09 | 2.6 | 2.3 | 1.12 |
| | | | AdaSkip | **29.6** | **82.4** | **66.1** | **1.15** | **23.6** | **20.2** | **1.20** |
| 16 | 1.33 | LLaMA3.1-8B-128k | Early Exit | 11.4 | 4.5 | 7.8 | **1.23** | 4.9 | 5.3 | 1.26 |
| | | | SkipDecode | 0.0 | 0.1 | 0.0 | **1.23** | 15.2 | 13.8 | 1.15 |
| | | | Unifed Skipping | 0.6 | 1.0 | 0.0 | 1.22 | 12.0 | 8.7 | 1.26 |
| | | | AdaSkip | **18.0** | **62.3** | **72.2** | 1.22 | **17.5** | **19.1** | **1.32** |
| | | InternLM-7B-8k | Early Exit | 0.7 | 0.5 | 6.1 | 1.31 | 0.6 | 0.4 | 1.28 |
| | | | SkipDecode | 0.0 | 0.0 | 0.0 | 1.31 | 9.2 | 9.8 | 1.16 |
| | | | Unifed Skipping | 5.1 | 0.4 | 5.0 | 1.31 | 5.7 | 6.4 | 1.28 |
| | | | AdaSkip | **17.2** | **38.7** | **29.4** | **1.51** | **9.4** | **9.8** | **1.47** |
| | | Vicuna-v1.5-7B-16k | Early Exit | 9.6 | **41.4** | 15.0 | 1.25 | 3.0 | 3.6 | 1.28 |
| | | | SkipDecode | 0.0 | 0.0 | 0.0 | 1.25 | 4.8 | 3.8 | 1.16 |
| | | | Unifed Skipping | 0.0 | 0.0 | 0.0 | 1.25 | 2.3 | 2.2 | 1.28 |
| | | | AdaSkip | **10.6** | 39.0 | **43.9** | **1.31** | **13.7** | **14.7** | **1.40** |

Table 3: Evaluation of different skipping strategies.

AdaSkip significantly outperforms the other baselines in both Doc QA and Few-shot Learning tasks. For example, on the LLaMA3.1-8B-128k model, with a target skip sublayer number of 8, AdaSkip achieves a classification accuracy of 72.8% on TREC and an F1 score of 86.6 on TriviaQA, closely approximating the performance of the full model. Even with up to 16 skipped sublayers, AdaSkip's accuracy on TREC remains at 72.2%. In contrast, the accuracy of the SkipDecode and Unified Skipping approaches decrease by more than 90% when skipping only 8 sublayers (4 whole layers).

In terms of speedup, the computational complexity of attention scales quadratically with sequence length, making attention computations more demanding than those of FFN in long-context scenarios. Due to AdaSkip skipping more attention sublayers, it achieves over a 10% speedup advantage on InternLM compared to the baseline. For the LLaMA model, the attention sequence parallelism and other optimization techniques are relatively mature, making the FFN execution time longer during the prefilling phase. As a result, our approach is slightly outperformed by the baseline.

## Results of Decoding Tasks

The right half of Table 3 presents the results of decoding tasks. Despite the baselines being specifically tailored for decoding tasks, our method consistently demonstrates superior performance. Even with the number of skipped sublayers reaching 16, we still maintain comparable performance. For instance, the LLaMA model achieves Rouge-L scores exceeding 17.5 on both datasets, comparable to the full InternLM model. It is noteworthy that the Early Exit method, which performs reasonably well in the prefilling tasks, fails to maintain generation quality during decoding. Its Vicuna Rouge-L scores for the two summarization tasks fall below 4.0, possibly due to the accumulation of errors in the autoregressive process. In contrast, AdaSkip accurately identifies the least significant sublayers, allowing LLM to maintain optimal performance even with additional skipping of FFNs.

In terms of execution speed, the inference time during the decoding phase is primarily dictated by HBM access. In long-context inference, attention operates slower than FFN due to the extensive KV cache access required. Our approach achieves a higher acceleration ratio by skipping more attention layers at the outset, thanks to the higher attention

| # Target Skip Sublayer | Skipping Strategy | LLaMA-3.1B-128k | | InternLM-7B-8k | | Vicuna-v1.5-7B-16k | |
|---|---|---|---|---|---|---|---|
| | | GovReport (Rouge-L) | MultiNews (Rouge-L) | GovReport (Rouge-L) | MultiNews (Rouge-L) | GovReport (Rouge-L) | MultiNews (Rouge-L) |
| 0 | Full Model | 34.2 | 25.8 | 18.2 | 17.6 | 27.2 | 22.4 |
| 8 | Early Exit | 15.3 | 12.4 | 2.7 | 3.4 | 12.2 | 13.3 |
| | SkipDecode | 1.0 | 1.1 | 0.0 | 0.0 | 0.0 | 0.0 |
| | Unifed Skipping | 1.6 | 1.1 | 8.9 | 9.9 | 0.0 | 0.0 |
| | AdaSkip | **30.5** | **24.1** | **12.9** | **12.7** | **23.0** | **21.1** |
| 16 | Early Exit | 4.3 | 4.4 | 0.5 | 0.3 | 1.9 | 1.8 |
| | SkipDecode | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| | Unifed Skipping | 0.0 | 0.1 | 0.4 | 1.0 | 0.0 | 0.0 |
| | AdaSkip | **18.9** | **17.8** | **7.7** | **7.1** | **13.6** | **14.1** |

Table 4: Evaluation on End-to-End Skipping Strategies.

similarity obtained during the offline learning phase. After online learning, we further enhance the acceleration by selectively skipping additional FFN layers. Overall, our method delivers up to a 17% acceleration improvement compared to the baseline. The Skip Decode approach achieves the lowest speedup because it employs a progressive layer skipping strategy, where the number of skipped layers gradually increases with the decoding steps until reaching the preset number.

### Results of End-to-End Testing

We evaluate the end-to-end performance of various layer skipping strategies, namely, implementing simultaneous skipping during both the prefilling and decoding phases. As demonstrated by Table 4, the performance of existing approaches significantly degrades when layer skipping is applied in both phases, compared to applying it solely during decoding. The SkipDecode approach causes Rouge-L scores to plummet to nearly zero across all three models. Similarly, Unified Skipping, which previously exhibited a modest difference from our approach on specific data points in decoding tasks, sees all its Rouge-L scores drop below 10.0 in this scenario. Additionally, when skipping 16 sublayers, the Early Exit approach yields scores below 5.0 across all models.

The results highlight the significant limitations of existing methods, which are unable to effectively apply layer skipping during both the prefilling and decoding phases in tasks with longer generation lengths. In contrast, our approach maintains nearly identical performance as when layer skipping is applied only during the decoding phase, demonstrating that our skipping strategy effectively adapts to both the prefilling and decoding phases. In real-world long-context tasks, our method exhibits exceptional practical value due to its ability to employ a complete layer-skipping strategy. It can markedly optimize the TTFT introduced during the prefilling phase and reduce the storage costs of the KV cache for long prompts.

## Related Work

**Long-context Model.** With the growing demand for long-context models, numerous studies have concentrated on expanding the context window of LLMs. Many models have fine-tuned LLaMA-2 by scaling Rotary Position Embeddings (RoPE) (Su et al. 2023), expanding its input window to 32k, as seen in LongChat (Li et al. 2023), and to 128k, as demonstrated in Yarn-LLaMA-2 (Peng et al. 2023). By leveraging length extrapolation, the context windows can extend beyond 1 million tokens (Liu et al. 2023a). However, these approaches do not alleviate the substantial inference costs associated with long-context processing.

**Long-context LLM Inference Optimization.** Given the substantial increase in KV cache size introduced by long sequences, many studies have concentrated their inference optimization efforts on compressing, evicting, and reusing KV cache. Heavy Hitter Oracle (H2O) (Zhang et al. 2024b) retains a limited budget of the important KV cache based on the sum of historical attention scores. SnapKV (Li et al. 2024) reduces memory access during decoding by observing the attention distribution of the prompt's tail over the prefix to selectively filter the corresponding KV cache, thereby achieving acceleration. PyramidKV (Zhang et al. 2024a) optimizes KV cache storage more flexibly by allocating different KV cache budgets to various layers and attention heads based on the observed information flow aggregation patterns. However, these approaches fail to address the substantial computational burden associated with generating extensive KV cache during the long sequence prefilling stage.

## Conclusion

In conclusion, this paper focuses on exploring the layer-wise skipping strategy in long-context inference. It first discusses the typical challenges in long-context inference and presents a detailed examination of the importance distribution of various components including layer and sublayer modules such as attention and FFN across a variety of different models. The analysis underlines the limitations of the current layer-wise skipping strategies in long-context inference. In response to these limitations, this paper proposes a novel, auto-adaptive, sublayer-wise skipping strategy that requires no training and is applicable to both the prefilling and decoding phases. Through rigorous testing across a diverse array of long-context datasets and models, we have demonstrated that our system, AdaSkip, significantly outperforms the baseline in both generation quality and inference speed.

## Acknowledgments

This work was supported in part by China NSF grant No. 62202297, Open Project Program of Laboratory of Pinghu, and Huawei Cloud. The opinions, findings, conclusions, and recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies or the government.

## References

AI, .; :; Young, A.; Chen, B.; Li, C.; Huang, C.; Zhang, G.; Zhang, G.; Li, H.; Zhu, J.; Chen, J.; Chang, J.; Yu, K.; Liu, P.; Liu, Q.; Yue, S.; Yang, S.; Yang, S.; Yu, T.; Xie, W.; Huang, W.; Hu, X.; Ren, X.; Niu, X.; Nie, P.; Xu, Y.; Liu, Y.; Wang, Y.; Cai, Y.; Gu, Z.; Liu, Z.; and Dai, Z. 2024. Yi: Open Foundation Models by 01.AI. arXiv:2403.04652.

Bae, S.; Ko, J.; Song, H.; and Yun, S.-Y. 2023. Fast and robust early-exiting framework for autoregressive language models with synchronized parallel decoding. *arXiv preprint arXiv:2310.05424*.

Bai, Y.; Lv, X.; Zhang, J.; Lyu, H.; Tang, J.; Huang, Z.; Du, Z.; Liu, X.; Zeng, A.; Hou, L.; Dong, Y.; Tang, J.; and Li, J. 2023. LongBench: A Bilingual, Multitask Benchmark for Long Context Understanding. arXiv:2308.14508.

Bai, Y.; Lv, X.; Zhang, J.; Lyu, H.; Tang, J.; Huang, Z.; Du, Z.; Liu, X.; Zeng, A.; Hou, L.; Dong, Y.; Tang, J.; and Li, J. 2024. LongBench: A Bilingual, Multitask Benchmark for Long Context Understanding. arXiv:2308.14508.

Bairi, R.; Sonwane, A.; Kanade, A.; C, V. D.; Iyer, A.; Parthasarathy, S.; Rajamani, S.; Ashok, B.; and Shet, S. 2023. CodePlan: Repository-level Coding using LLMs and Planning. arXiv:2309.12499.

Chen, Y.; Pan, X.; Li, Y.; Ding, B.; and Zhou, J. 2024. EE-LLM: Large-Scale Training and Inference of Early-Exit Large Language Models with 3D Parallelism. arXiv:2312.04916.

DeepSeek-AI; Liu, A.; Feng, B.; Wang, B.; Wang, B.; Liu, B.; Zhao, C.; Dengr, C.; Ruan, C.; Dai, D.; Guo, D.; Yang, D.; Chen, D.; Ji, D.; Li, E.; Lin, F.; Luo, F.; Hao, G.; Chen, G.; Li, G.; Zhang, H.; Xu, H.; Yang, H.; Zhang, H.; Ding, H.; Xin, H.; Gao, H.; Li, H.; Qu, H.; Cai, J. L.; Liang, J.; Guo, J.; Ni, J.; Li, J.; Chen, J.; Yuan, J.; Qiu, J.; Song, J.; Dong, K.; Gao, K.; Guan, K.; Wang, L.; Zhang, L.; Xu, L.; Xia, L.; Zhao, L.; Zhang, L.; Li, M.; Wang, M.; Zhang, M.; Zhang, M.; Tang, M.; Li, M.; Tian, N.; Huang, P.; Wang, P.; Zhang, P.; Zhu, Q.; Chen, Q.; Du, Q.; Chen, R. J.; Jin, R. L.; Ge, R.; Pan, R.; Xu, R.; Chen, R.; Li, S. S.; Lu, S.; Zhou, S.; Chen, S.; Wu, S.; Ye, S.; Ma, S.; Wang, S.; Zhou, S.; Yu, S.; Zhou, S.; Zheng, S.; Wang, T.; Pei, T.; Yuan, T.; Sun, T.; Xiao, W. L.; Zeng, W.; An, W.; Liu, W.; Liang, W.; Gao, W.; Zhang, W.; Li, X. Q.; Jin, X.; Wang, X.; Bi, X.; Liu, X.; Wang, X.; Shen, X.; Chen, X.; Chen, X.; Nie, X.; Sun, X.; Wang, X.; Liu, X.; Xie, X.; Yu, X.; Song, X.; Zhou, X.; Yang, X.; Lu, X.; Su, X.; Wu, Y.; Li, Y. K.; Wei, Y. X.; Zhu, Y. X.; Xu, Y.; Huang, Y.; Li, Y.; Zhao, Y.; Sun, Y.; Li, Y.; Wang, Y.; Zheng, Y.; Zhang, Y.; Xiong, Y.; Zhao, Y.; He, Y.; Tang, Y.; Piao, Y.; Dong, Y.; Tan, Y.; Liu, Y.; Wang, Y.; Guo, Y.; Zhu, Y.; Wang, Y.; Zou, Y.; Zha, Y.; Ma, Y.; Yan, Y.; You, Y.; Liu, Y.; Ren, Z. Z.; Ren, Z.; Sha, Z.; Fu, Z.; Huang, Z.; Zhang, Z.; Xie, Z.; Hao, Z.; Shao, Z.; Wen, Z.; Xu, Z.; Zhang, Z.; Li, Z.; Wang, Z.; Gu, Z.; Li, Z.; and Xie, Z. 2024. DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model. arXiv:2405.04434.

Del Corro, L.; Del Giorno, A.; Agarwal, S.; Yu, B.; Awadallah, A.; and Mukherjee, S. 2023. Skipdecode: Autoregressive skip decoding with batching and caching for efficient llm inference. *arXiv preprint arXiv:2307.02628*.

Fabbri, A. R.; Li, I.; She, T.; Li, S.; and Radev, D. R. 2019. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. *arXiv preprint arXiv:1906.01749*.

Fan, S.; Jiang, X.; Li, X.; Meng, X.; Han, P.; Shang, S.; Sun, A.; Wang, Y.; and Wang, Z. 2024. Not all Layers of LLMs are Necessary during Inference. *arXiv preprint arXiv:2403.02181*.

Huang, L.; Cao, S.; Parulian, N.; Ji, H.; and Wang, L. 2021. Efficient attentions for long document summarization. *arXiv preprint arXiv:2104.02112*.

Jaiswal, A.; Hu, B.; Yin, L.; Ro, Y.; Liu, S.; Chen, T.; and Akella, A. 2024. FFN-SkipLLM: A Hidden Gem for Autoregressive Decoding with Adaptive Feed Forward Skipping. *arXiv preprint arXiv:2404.03865*.

Jiang, H.; Li, Y.; Zhang, C.; Wu, Q.; Luo, X.; Ahn, S.; Han, Z.; Abdi, A. H.; Li, D.; Lin, C.-Y.; Yang, Y.; and Qiu, L. 2024. MInference 1.0: Accelerating Pre-filling for Long-Context LLMs via Dynamic Sparse Attention. arXiv:2407.02490.

Jimenez, C. E.; Yang, J.; Wettig, A.; Yao, S.; Pei, K.; Press, O.; and Narasimhan, K. 2024. SWE-bench: Can Language Models Resolve Real-World GitHub Issues? arXiv:2310.06770.

Joshi, M.; Choi, E.; Weld, D. S.; and Zettlemoyer, L. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Li, D.; Shao, R.; Xie, A.; Sheng, Y.; Zheng, L.; Gonzalez, J.; Stoica, I.; Ma, X.; and Zhang, H. 2023. How Long Can Context Length of Open-Source LLMs truly Promise? In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.

Li, X.; and Roth, D. 2002. Learning Question Classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Li, Y.; Huang, Y.; Yang, B.; Venkitesh, B.; Locatelli, A.; Ye, H.; Cai, T.; Lewis, P.; and Chen, D. 2024. Snapkv: Llm knows what you are looking for before generation. *arXiv preprint arXiv:2404.14469*.

Liu, H.; Yan, W.; Zaharia, M.; and Abbeel, P. 2024. World Model on Million-Length Video And Language With Blockwise RingAttention. arXiv:2402.08268.

Liu, T.; Xu, C.; and McAuley, J. 2023. RepoBench: Benchmarking Repository-Level Code Auto-Completion Systems. arXiv:2306.03091.

Liu, X.; Yan, H.; Zhang, S.; An, C.; Qiu, X.; and Lin, D. 2023a. Scaling laws of rope-based extrapolation. *arXiv preprint arXiv:2310.05209*.

Liu, Y.; Meng, F.; and Zhou, J. 2024. Accelerating Inference in Large Language Models with a Unified Layer Skipping Strategy. *arXiv preprint arXiv:2404.06954*.

Liu, Z.; Wang, J.; Dao, T.; Zhou, T.; Yuan, B.; Song, Z.; Shrivastava, A.; Zhang, C.; Tian, Y.; Re, C.; et al. 2023b. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, 22137–22176. PMLR.

Park, J. S.; O'Brien, J. C.; Cai, C. J.; Morris, M. R.; Liang, P.; and Bernstein, M. S. 2023. Generative Agents: Interactive Simulacra of Human Behavior. arXiv:2304.03442.

Peng, B.; Quesnelle, J.; Fan, H.; and Shippole, E. 2023. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*.

Schuster, T.; Fisch, A.; Gupta, J.; Dehghani, M.; Bahri, D.; Tran, V.; Tay, Y.; and Metzler, D. 2022. Confident adaptive language modeling. *Advances in Neural Information Processing Systems*, 35: 17456–17472.

Srivatsa, V.; He, Z.; Abhyankar, R.; Li, D.; and Zhang, Y. 2024. Preble: Efficient Distributed Prompt Scheduling for LLM Serving. arXiv:2407.00023.

Su, J.; Lu, Y.; Pan, S.; Murtadha, A.; Wen, B.; and Roformer, Y. L. 2023. Enhanced transformer with rotary position embedding., 2021. *DOI: https://doi. org/10.1016/j. neucom*.

Tang, J.; Zhao, Y.; Zhu, K.; Xiao, G.; Kasikci, B.; and Han, S. 2024. Quest: Query-Aware Sparsity for Efficient Long-Context LLM Inference. arXiv:2406.10774.

Taori, R.; Gulrajani, I.; Zhang, T.; Dubois, Y.; Li, X.; Guestrin, C.; Liang, P.; and Hashimoto, T. B. 2023. Stanford alpaca: An instruction-following llama model.

Varshney, N.; Chatterjee, A.; Parmar, M.; and Baral, C. 2023. Accelerating llama inference by enabling intermediate layer decoding via instruction tuning with lite. *arXiv e-prints*, arXiv–2310.

Wang, L.; Ma, C.; Feng, X.; Zhang, Z.; Yang, H.; Zhang, J.; Chen, Z.; Tang, J.; Chen, X.; Lin, Y.; Zhao, W. X.; Wei, Z.; and Wen, J. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6).

Wu, H.; Zhan, M.; Tan, H.; Hou, Z.; Liang, D.; and Song, L. 2023. VCSUM: A Versatile Chinese Meeting Summarization Dataset. arXiv:2305.05280.

Xiao, C.; Zhang, P.; Han, X.; Xiao, G.; Lin, Y.; Zhang, Z.; Liu, Z.; and Sun, M. 2024. InfLLM: Training-Free Long-Context Extrapolation for LLMs with an Efficient Context Memory. arXiv:2402.04617.

Zhang, Y.; Gao, B.; Liu, T.; Lu, K.; Xiong, W.; Dong, Y.; Chang, B.; Hu, J.; Xiao, W.; et al. 2024a. PyramidKV: Dynamic KV Cache Compression based on Pyramidal Information Funneling. *arXiv preprint arXiv:2406.02069*.

Zhang, Z.; Sheng, Y.; Zhou, T.; Chen, T.; Zheng, L.; Cai, R.; Song, Z.; Tian, Y.; Ré, C.; Barrett, C.; et al. 2024b. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36.

Zhu, Y.; Yang, X.; Wu, Y.; and Zhang, W. 2024. Hierarchical Skip Decoding for Efficient Autoregressive Text Generation. arXiv:2403.14919.