FFN-SkipLLM: A Hidden Gem for Autoregressive Decoding with Adaptive Feed Forward Skipping

Anonymous ACL submission

Abstract

Autoregressive Large Language Models (*e.g.,* LLaMa, GPTs) are omnipresent achieving remarkable success in language understanding and generation. However, such impressive capability typically comes with a substantial model size, which presents significant challenges for autoregressive token-by-token generation. To mitigate computation overload incurred during generation, several *early-exit* and *layer-dropping* strategies have been proposed. Despite some promising success due to the redundancy across LLMs layers on metrics like Rough-L/BLUE, our careful knowledge-intensive evaluation unveils issues such as generation collapse, hallucination, and noticeable performance drop even at the trivial exit ratio of $\sim$ 10-15% of layers. We attribute these errors primarily to ineffective handling of the KV cache through state copying during early exit. In this work, we observe the *saturation of computationally expensive feed-forward blocks* of LLM layers and propose **FFN-SkipLLM**, which is a novel fine-grained skip strategy for autoregressive LLMs. FFN-SkipLLM leverages an *input-adaptive feed-forward skipping approach* that can skip $\sim$ 25-30% of FFN blocks of LLMs with marginal change in performance on knowledge-intensive generation tasks without any requirement to handle the KV cache. Our extensive experiments and ablation studies across benchmarks like MT-Bench, Factoid-QA, and variable-length text summarization illustrate how our simple and easy-to-use method can facilitate faster autoregressive decoding. Related codes will be open-sourced.

1 Introduction

stealers, profoundly influencing not only the landscape of NLP (Ram et al., 2023; Liu et al., 2023a; Sawada et al., 2023; Jaiswal et al., 2021; Qin et al., 2023; Zhuo, 2023; Lee et al., 2023), but also recently buttressing numerous computer vision (Lian et al., 2023; Wang et al., 2023; Lai et al., 2023; Lu et al., 2023; Li et al., 2024) and graph neural networks (Ye et al., 2023; Chen et al., 2023c; Qian et al., 2023; Duan et al., 2023; Chen et al., 2024) algorithms, achieving stellar performance across various task benchmarks. However, their widespread adoption is hindered by their massive scale, characterized by billions of parameters, which demand exceedingly high computational resources and memory capacities. For instance, the GPT-175B model necessitates 325 GB of GPU memory for loading its weights and relies on a minimum of five A100 (80GB) GPUs employing sophisticated parallelism techniques (Sheng et al., 2023). This imposing computational and memory requirement presents a challenge to the broader accessibility of these models.

To alleviate the demanding hardware requirements for deploying massive trained models, considerable efforts have been devoted to mitigating their high computational inference cost resulting from token-by-token generation. Among several model compression techniques such as quantization (Liu et al., 2023c; Kim et al., 2023; Dettmers et al., 2023a; Frantar et al., 2022; Lin et al., 2023; Dettmers et al., 2023b), and sparse neural networks (Frankle and Carbin, 2019; Chen et al., 2020; Jaiswal et al., 2022; Lee et al., 2019; Zhangheng et al., 2023; Jaiswal et al., 2023b,a; Liu et al., 2023b; Yin et al., 2023a,b) which require additional hardware support for speedup, *token-level early exit or layer-skip* has emerged as a promising technique to alleviate these limitations by allowing tokens to cease computation as soon as their hidden states reach saturation (Sun et al., 2022; Del Corro et al., 2023; Schuster et al., 2022; Men et al., 2024). These methods exploit existing redundancy across LLMs' layers which can be ignored during token-by-token generation significantly saving massive computation involved within a layer (*e.g.,* $\sim$ 200-300 million parameters in a single LLaMa layer).

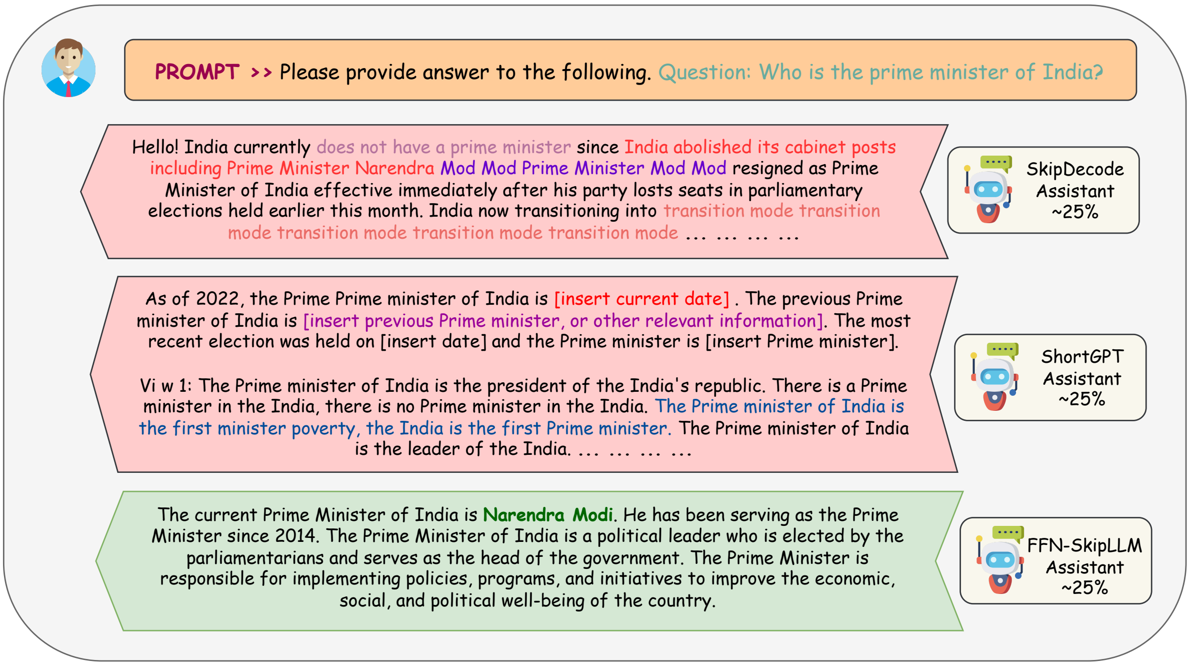

Figure 1: **Merits of Autoregressive Decoding with Layer Skipping:** Comparison of the responses generated by two recent Layer Skipping methods, namely SkipDecode (Del Corro et al., 2023) and ShortGPT (Men et al., 2024) for a knowledge-intensive QA example. It can be observed that both LLaMa-chat-13B model with $\sim 25\%$ layers skipped per token using SkipDecode and ShortGPT suffers from *hallucination* and *token collapse* (repetitive generation) while FFN-SkipLLM can still retrieve the correct response.

Layer Name	# Parameters
attention.wq.weight	$\sim$ 16.77M
attention.wk.weight	$\sim$ 16.77M
attention.wv.weight	$\sim$ 16.77M
attention.wo.weight	$\sim$ 16.77M
feed_forward.w1.weight	$\sim$ 45.08M
feed_forward.w2.weight	$\sim$ 45.08M
feed_forward.w3.weight	$\sim$ 45.08M

Table 1: Parameter count of Attention and FFN layers of a transformer block in LLaMa-7B.

Although the proposed methods have shown some promising success, their performance is widely restricted by the issue of inappropriately handling KV caching. KV caching saves keys and values of all attention layers for previously generated tokens and accelerates sequence generation by reducing redundant computation (though at the cost of higher memory usage). Given a token generated via early exiting, its KV caches in subsequent layers are incomplete which impedes the generation of future tokens beyond the exiting layer of the current token.

For handling the KV cache issue, some recent works (Elbayad et al., 2019; Schuster et al., 2022; Li et al., 2021b; Chen et al., 2023a; Del Corro et al., 2023) propose three main solutions: copying hidden states, pre-fixed token-level skip pattern, and KV recomputation. Despite these mitigation methods, our careful knowledge-intensive investigation reveals that layer-skipping induces permanent damage due to deviation from the inference process that the model is trained to excel at, leading to significant hallucination of wrong facts and token generation collapse. Figure 1 shows the comparison of the responses generated by two recent Layer Skipping methods, namely SkipDecode (Del Corro et al., 2023) and ShortGPT (Men et al., 2024) for a knowledge-intensive QA example. In their response, both ShortGPT and SkipDeocde fail to generate the correct answer “Narendra Modi", suffer from token collapse, and hallucinate misinformation.

In this work, we ask an interesting unexplored question: *Instead of attempting to fix the KV cache, can we completely circumvent the KV cache bottleneck of layer-skipping and still avoid unnecessary computational expenses while mitigating hallucination and token generation collapse?* To this end, our work is the **first** attempt to investigate a fine-grained layer-skipping strategy that focuses on computationally expensive feed-forward network (FFN) blocks in LLMs. Table 1 presents the parameter counts of individual components of LLaMa-7B layer and it can be observed that FFN blocks hold approximately *two-thirds* of the parameter budget of the layer, marking them as favorable candidates for skipping during token-by-token generation. Our work derives its motivation from two primary observations: ① we find a *monotonically increasing cosine similarity* between the tensors generated before and after the FFN blocks across layers in LLMs which indicates unnecessary computation performed by these blocks, ② due to the observed phenomenon of attention sink (Xiao et al., 2023), we find that allowing a *small fraction of first-few token (∼ 5-10% of maximum sequence length) decoding using the full strength* (no-skip) of LLMs can significantly help in stabilizing the KV cache, paving way for skipping FFN blocks without significant performance degradation for later tokens. We propose **FFN-SkipLLM**, a novel fine-grained skip strategy of autoregressive LLMs which is an input-adaptive feed-forward skipping strategy that can skip ∼ 25-30% of FFN blocks of LLMs with marginal change in performance on knowledge-intensive tasks. Note that because we only skip FFN blocks, we in turn can fully circumvent the KV cache issue associated with layer-skipping. Our primary contributions can be summarized as:

- Unlike prior layer-skipping methods, we focus on only skipping computationally expensive FFN blocks based on our observation of their monotonically increasing saturation within the middle layers of LLMs.

- Our proposed FFN-SkipLLM uses a simple *cosine similarity* metric across tensors to capture the trend of FFN saturation and decide an input-adaptive skipping of FFN blocks. More specifically, once a similarity threshold is reached, given the monotonically increasing saturation, we **greedily** select the next k layers whose FFN blocks can be ignored depending on the desired skipping requirement.

- Our extensive knowledge-intensive experiments such as Factoid-QA, Multi-turn conversations, and Variable-length in-context text summarization, reveal that FFN-SkipLLM can skip ∼ 25-30% of FFN blocks of LLMs with a marginal change in performance and reduce hallucination and token collapse.

2 Layer-skipping: An Knowledge-Intensive Evaluation

Recent advancements in autoregressive models (Touvron et al., 2023; Qin et al., 2023; Zhang et al., 2022) have revolutionized the quality of language generation in various generative tasks, including question answering (Rajpurkar et al., 2016), summarization (Fabbri et al., 2019; Nallapati et al., 2016), and machine translation (Bahdanau et al., 2014). However, these large transformer models face challenges in terms of high inference latency attributed to their numerous layers and the autoregressive decoding process. The sequential computation of multiple stacks of transformer layers for each token during the inference stage imposes significant computational overheads, thus limiting their real-time adaptability.

To counter the computational cost of token-by-token generation with modern gigantic LLMs, several works (Chen et al., 2023b; Men et al., 2024; Del Corro et al., 2023; Kim et al., 2024; Bae et al., 2023a) have been recently exploring token-level early exit and layer-skipping (depth-pruning) strategies. The primary challenge associated with these approaches is that if the current token exits at a higher layer, there arises a need to recalculate the Key-Value (KV) caches for preceding tokens. To this end, three major approaches have been explored: (1) copy the hidden states of the current token at the exiting layer to all later layers, which will be used to compute the keys and values at later attention layers, (2) pre-specify the exiting layer for each token, while ensuring that KV missing in previous tokens will not hinder the generation of later tokens; with this approach, the ability of token-wise adaptive selection of exits is inevitably lost, (3) KV recomputation which is a variant of synchronized parallel decoding and adds additional computational and memory overhead.

Despite some notable performance gains over some metrics (*e.g.*, perplexity, Rough-L, BLUE), our careful knowledge-intensive investigation reveals that the KV cache problem during layer-skip is not effectively addressed. Figure 1 illustrates the responses generated by two recent layer-skipping methods SkipDecode (Del Corro et al., 2023) and (Men et al., 2024) for a given factoid-based QA task which requires answering using relevant entities and attributes ingested within LLMs during pre-training. Interestingly, answers generated by the SkipDecode agent hallucinate misinformation claiming *'... does not have a prime minister ... India abolished its cabinet posts ...* ' while the ShortGPT agent fails to generate any factoid to answer the question. Note that both agents suffer from token collapse and start generating repetitive content after some time. To quantitatively estimate the damage of layer-skipping, Table 3 presents the

Method (∼ 20% Skip)	Factoid-QA	Multi-turn Conversation	In-context Summarization
Full Model	79.02	7.61	8.15
SkipDecode (Del Corro et al., 2023)	73.33	6.53	7.47
ShortGPT (Men et al., 2024)	70.49	6.17	6.33
Ours (FFN-SkipLLM)	78.89	7.55	8.11

Table 2: Performance comparison of Autoregressive Decoding with $\sim 20\%$ layers skipped using SoTA methods (SkipDecode, ShortGPT) wrt. our proposed input-adaptive FFN-SkipLLM on knowledge-intensive tasks.

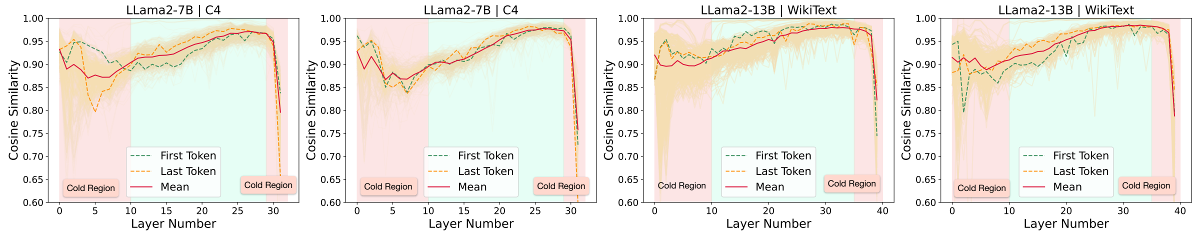

Figure 2: Cosine similarity across embedding dimension of a token tensor entering before and after the FFN block of different layers in LLaMa-2 7B and 13B model. Inputs are sampled at random from Wikitext ad C4 datasets and the mean curve indicates the average cosine similarity across 128 generated tokens. Red regions are termed *cold regions* in our work and skipping FFN blocks within this region significantly hurt LLMs performance.

performance of SkipDecode and ShortGPT with respect to the full model on three knowledge-rich tasks (Section 4.1, 4.2, 4.3) that closely resemble the daily use-cases for LLMs. It can be observed that despite impressive results reported on traditional metrics, we find the performance significantly suffers when compared to the full model. To this end, in this work, we attempt to explore an orthogonal direction that diverges from conventional layer-skipping and investigate the potential of skipping computationally heavy FFN blocks across layers which accounts for approximately two-thirds of the parameter count.

3 FFN-SkipLLM: A Fine-grained Input-adaptive FFN Skipping

3.1 Preliminaries and Motivation

Given a autoregressive large language model (LLaMa-2 in our case) $\mathbf{M_L}$ with T layers, each layer $l_i \in$ L consists of two major computational blocks: Multihead-Attention block (W_q, W_k, W_v, W_o) and FFN block $(FF_{W1}, FF_{W2}, FF_{W3})$. Table 1 presents the approximate parameter counts occupied by these components in layer l_i indicating FFN blocks occupying around two-thirds of the total parameter counts. In pursuit of avoiding the KV issue incurred due to entire layer-skipping, we explored the redundant computation done by FFN blocks during token-by-token generation. More specifically, given a layer l_i, we calculated the cosine similarity across the embedding dimension of the tensor entering a given FFN block and exiting the block.

Algorithm 1: Pseudocode for our Input-Adaptive FFN-SkipLLM

```
Input: warm_up_index: int; input_state:
       tensor; cold_s: int; cold_e: int;
       token_index: int
if token_index ≤ warm_up_index then
    generate_with_full_model(token_index,
     input_state)
else
    generate_with_skip_model(token_index,
     input_state, cold_s, cold_e)

def
 generate_with_skip_model(token_index,
 input_state, cold_s, cold_e):
    past_state ← input_state
    for <0 ... cold_s> do
        h ← past_start +
         attention(past_state)
        past_state ← h +
         feed_forward(h)
    skip_state ← False
    for <cold_s ... cold_e> do
        h ← past_start +
         attention(past_state)
        if skip_state == False then
            temp ← h + feed_forward(h)
            sim_score ← cosine (h,
             temp)
            if sim_score ≥ sim_threshold
             then
                skip_state ← True
            past_state ← temp
        else
            past_state ← h
    for <cold_e ... num_layers> do
        h ← past_state +
         attention(past_state)
        past_state ← h +
         feed_forward(h)
```

Figure 2 presents the layerwise mean cosine similarity of 128 generated tokens across different layers in LlaMa-2 7B and 13B models where the initial input prompt was sampled from the wikitext and C4 datasets. We are motivated by the following three observations: ① surprisingly **high cosine similarity** across the embedding dimension of the tensor entering a given FFN block and exiting it indicates the existence of redundant computation; ② **monotonically increasing cosine similarity** across middle layers (yellow region) indicating redundant computation is concentrated around middle layers in the model $\mathbf{M_L}$; ③ **existence of two cold segments** (red region) where there exists a decreasing trend of cosine similarity indicating they significantly influence the input tensor and should be left intact during our FFN blocks skipping goal. In addition, a recent work (Xiao et al., 2023) identified the emergence of *attention sink* attributed to the strong attention scores towards initial tokens in autoregressive token-by-token generation. Our experiments found this observation is highly effective in stabilizing the generated tokens with FFN block-skipping and reducing repetitive tokens. FFN-SkipLLM incorporates this with a hyperparameter `warm_up_index` to develop a high-quality KV cache for initial few token generations before adopting the FFN skipping policy.

3.2 Methodology

In this section, we will discuss our proposed methodology for input-adaptive FFN-SkipLLM. As discussed earlier, FFN-SkipLLM capitalizes on the redundant computational cost of FFN blocks across deep autoregressive LLMs for token generation. As shown in Figure 2, given the model $\mathbf{M_L}$, its layers can be categorized into two regions: *cold regions* (FFNs are non-redundant) and *non-cold regions* (FFNs tend to be redundant). Cold regions (red) encompass the first few layers (`cold_s`) and the last few layers (`cold_e`) and they can be identified using a small calibration set from Wikitext/C4. FFN-SkipLLM uses an extra hyperparameter `warm_up_index`[1] which represents how many initial first tokens will not undergo any layer-skipping to capitalize on attention sink observation.

Algorithm 1 illustrates the pseudocode for FFN-SkipLLM. A typical transformer layer performs two heavy operations: attention calculation and feed-forward transformation. Our proposed method allows both operations in cold regions but facilitates skipping feed-forward transformation within the non-cold regions. Our input adaptivity comes from tracking the cosine similarity of the token features before and after the FFN blocks and deciding when to start skipping given a `sim_thresold`. More specifically, based on our *monotonically increasing* cosine similarity in non-cold regions, we **greedily** skip k FFN blocks from the subsequent layers.

[1]Necessary ablation is provided in Section 5.1.

4 Experimental Results

Baseline Details: To empirically evaluate the performance gains enabled by our proposed FFN-SkipLLM across multiple knowledge-intensive tasks. We aim to investigate how well FNN block skipping can retain the ability to access factoid answers ingested during pretraining, perform multi-turn instruction following, and in-context summarization. Our baselines are: ① *full model* which indicate the maximum capability of LLM under consideration; ② *random skip* where FFN-blocks are dropped at random without giving careful consideration of cold and non-cold regions; ③ *no input adaptive* where we do not track the cosine similarity per token and FFN-blocks are dropped at random from the non-cold region. Our baselines are constructed to carefully validate the effect of our observations in FFN-SkipLLM.

4.1 Factoid-based Question Answering

Task Definition and Rationale. Factoid-based Question Answering (Factoid-QA) (Iyyer et al., 2014), which asks precise facts about entities, is a long-standing problem in NLP. A typical Factoid-QA task aims to search for entities or entity attributes from a knowledge graph, and it is widely used as a tool in academia, commercial search engines, and conversational assistants. Modern LLMs are trained on gigantic text corpora ingesting a large amount of world knowledge about entities and their relationships during pre-training, and have unique abilities to generate factually correct responses to user queries. In this task setting, we aim to investigate *how our input-adaptive FFN block skipping impacts LLMs' ability to answer natural language questions using facts, i.e., entities or attributes knowledge ingested within them during pre-training?*

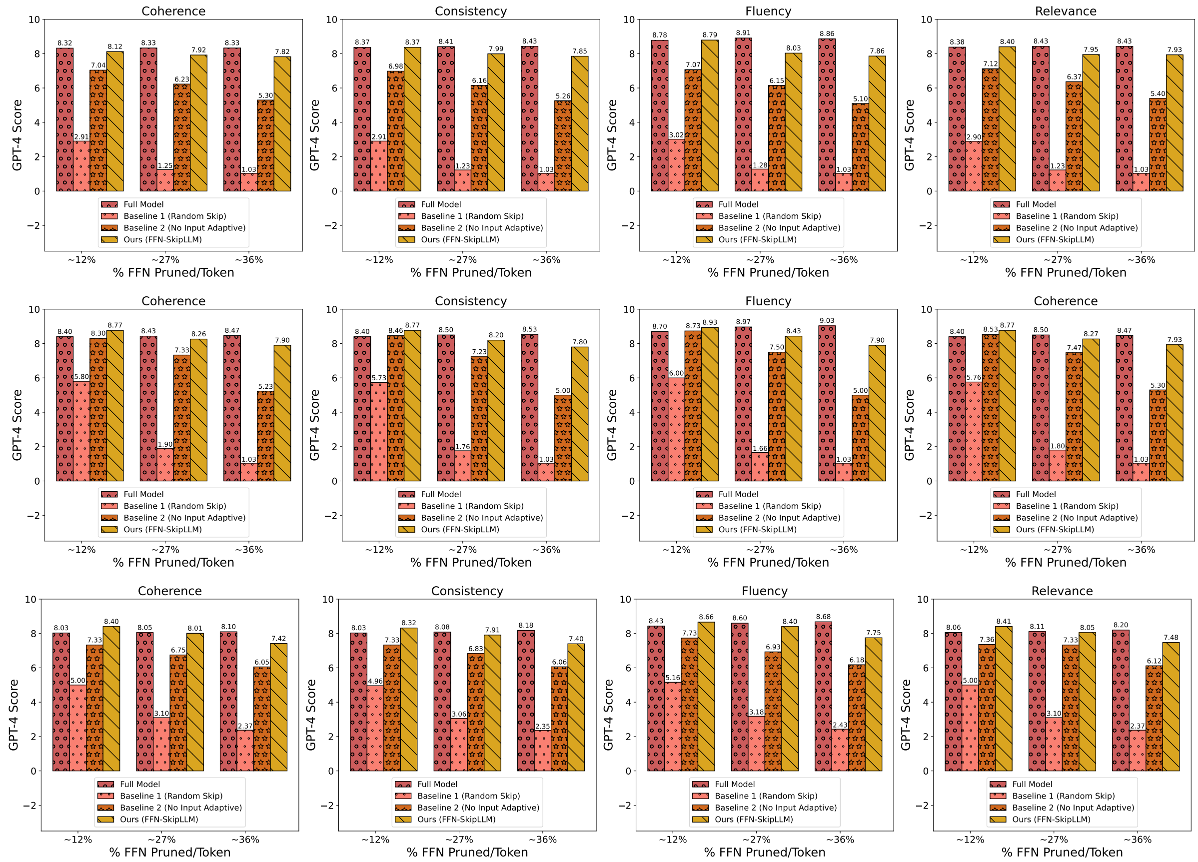

Figure 3: Performance comparison of our baselines wrt. FFN-SkipLLM for in-context summarization of small (row 1), medium (row 2), and large (row 3) stories while preserving coherence, consistency, fluency, and relevance.

Method	∼5%	∼15%	∼25%	∼35%
Full Model	79.02%			
Baseline 1 (Random Skip)	77.32%	72.96%	49.22%	31.07%
Baseline 2 (No input adaptive)	78.92%	77.71%	74.13%	69.93%
Ours (FFN-SkipLLM)	80.05%	78.42%	78.09%	75.61%

Table 3: Performance comparison of our baselines with varying layer skip ratios wrt. proposed input-adaptive FFN-SkipLLM on Factoid-based QA.

Prompt Design: Please give answer to this question: <QUESTION> The answer is

Example: Please give answer to this question: `Who is the prime minister of India?`

Model Response: Please give answer to this question: `Who is the prime minister of India?` The current Prime Minister of India is `Narendra Modi`. He has been serving as the Prime Minister since 2014. The Prime Minister of India is a political leader who is elected by the parliamentarians and serves as the head of the government. The Prime Minister is responsible for implementing policies, programs, and initiatives to improve the economic, social, and political well-being of the country.

Dataset Details and Results. We use FreebaseQA (Jiang et al., 2019) which is a dataset for open-domain QA over the Freebase knowledge graph. The QA pairs are collected from various sources, including the TriviaQA dataset (Joshi et al., 2017) and other trivia websites (QuizBalls, QuizZone, KnowQuiz), and are matched against Freebase to generate relevant subject-predicate-object triples that were further verified by human annotators. TriviaQA dataset shows rich linguistic variation and complexity, making it a good testbed for evaluating knowledge ingested within LLMs.

The results of various baseline methods and FFN-SkipLLM are demonstrated in Table 3. It is interesting to observe that FFN-SkipLLM with ∼5% skip ratio per token can outperform the full model performance. A careful study of Baselines 1 and 2 indicates the effectiveness of our observation of cold vs non-cold regions for FFN-block skipping. Note that at a high skip ratio, the performance of the random baseline is significantly worse with ≥50% performance drop. On the other hand, we can also note that our input-adaptive FFN-SkipLLM is highly robust in retaining a large fraction of full model performance in comparison to Baseline 2.

4.2 In-context Variable Length Text Summarization

Task Formulation and Details. Modern LLMs have shown astonishing success in summarizing long-context documents in both abstractive and extractive settings. However, it is **yet not explored**

how FFN block skipping impacts LLMs' capability for summarization. In this task setting, we aim to investigate *how well autoregressive decoding with FFN block skipping hold onto consistency, coherence, fluency, and relevance when prompted to summarize textual information of varying length (small, medium, and large) in abstractive setting* (Jain et al., 2023). For evaluation, similar to (Zheng et al., 2023), we propose to use GPT-4 as a judge, which compares the compressed LLM generated summaries wrt. GPT-3.5 (text-davinci-003) generated summaries.

> **Prompt Design:** A chat between a curious user and an artificial intelligence assistant. The assistant gives helpful, detailed, and polite answers to the user's questions. USER: `Summarize the given story in less than 150 words while preserving high coherence, consistency, fluency, and relevance.\n\n <STORY>.`
> ASSISTANT:
> **Example:** A chat between a curious user and an artificial intelligence assistant. The assistant gives helpful, detailed, and polite answers to the user's questions. USER: `Summarize the given story in less than 150 words while preserving high coherence, consistency, fluency, and relevance.\n\nLibyan and U.S. officials say the two governments held face-to-face talks in Tunisia ...have denied previous reports of talks with the government.`
> ASSISTANT:

Dataset Details and Results We use a popular summarization dataset CNN/DailyMail (Chen et al., 2016) for evaluation, which is an English-language dataset containing just over 300k unique news articles written by journalists at CNN and DailyMail. We created 3 subset categories {small ($\leq$470 words), medium ($\geq$470 and $\leq$ 790 words), and large ($\geq$ 790 words)} of stories, each with 100 articles reflecting word distribution of CNN/DailyMail to minimize OpenAI API costs.

Figure 3 summarizes the result of the variable length text summarization task. One interesting observation we find is that with increasing in-context stories for summarization, we found that the performance of random baseline improves. Upon digging we found that it start copying random text snippets from the in-context story directly into the summary which led to a comparatively better GPT-4 evaluation score. With an increasing skip ratio, we found that the performance gap between FFN-SkipLLM and our baselines increases. Moreover, at $\sim$10-12% skip ratio we found that GPT-4 consistently ranks our summary better than the full model across coherence, consistency, fluency, and relevance.

4.3 Multi-turn Conversation and Instruction Following

Task Formulation and Rationale. In this task setting, we investigate *how FFN block skipping impacts the LLMs' ability to answer open-ended questions and evaluate their multi-turn conversational and instruction-following ability – two critical elements for human preference*. Evaluating AI chatbots is a challenging task, as it requires examining language understanding, reasoning, and context awareness. To compare the performance of compressed LLMs' responses, we closely follow the prompt design setting in MT-Bench (Zheng et al., 2023) using GPT-4 as a judge. We prompt GPT-4 to rate the answers generated by compressed LLMs wrt. GPT-3.5 (text-davinci-003) model based on varying metrics (*e.g.*, correctness, helpfulness, logic, accuracy, *etc.*) on a scale of `[0-10]` with detailed explanations.

Dataset Details and Results. We rely on the 80 high-quality multi-turn questions identified in MT-Bench (Zheng et al., 2023). This setting covers common-use human-centric interaction with LLMs, and focuses on challenging questions to differentiate models. We used 8 common categories of user prompts to guide the prompt construction to interact with compressed LLMs: writing, roleplay, extraction, reasoning, math, coding, *etc*. For each category, we adopted manually designed 10 multi-turn questions from MT-Bench to evaluate our compressed models.

> **Prompt Design:** A chat between a curious user and an artificial intelligence assistant. The assistant gives helpful, detailed, and polite answers to the user's questions. USER: `<QUESTION>`
> ASSISTANT:
> **Example:** A chat between a curious user and an artificial intelligence assistant. The assistant gives helpful, detailed, and polite answers to the user's questions. USER: `How can I improve my time management skills?`
> ASSISTANT:

Figure 4 presents the performance comparison of our baseline models across 8 different categories. It is surprising to observe that across some categories such as coding, fermi, and commonsense; FFN-SkipLLM perform quite match the performance of the full model comfortably up to $\sim$25% skip ratio per token. Unlike identified by (Men et al., 2024) that layer dropping fails on generative tasks, it is important to acknowledge our careful FFN block dropping can significantly reduce hallucination across knowledge-intensive tasks. Note that our random skip baseline observes a terminal decline in performance even with a slop rato of 10-

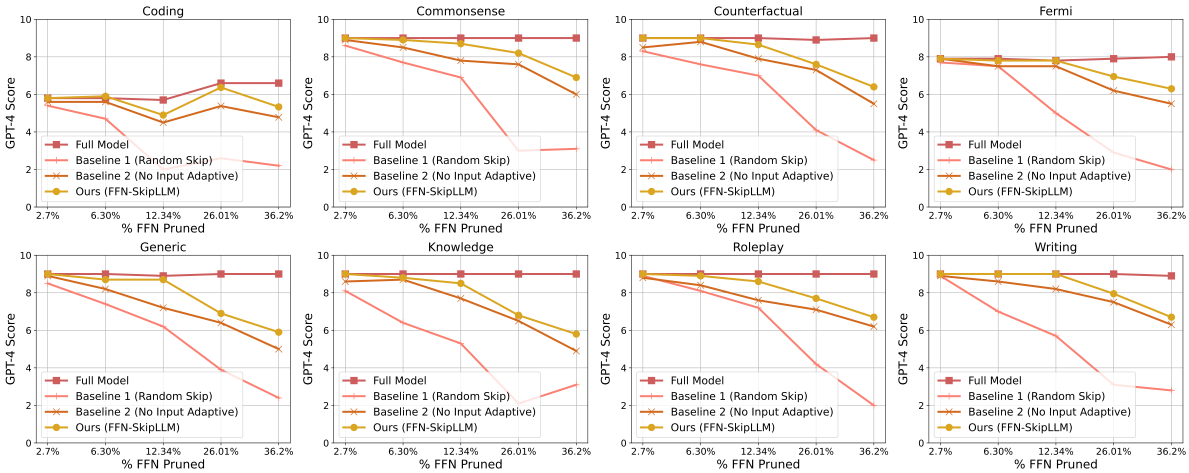

Figure 4: Performance comparison of our baselines with varying layer skip ratios wrt. FFN-SkipLLM on multi-turn conversation across 8 different categories.

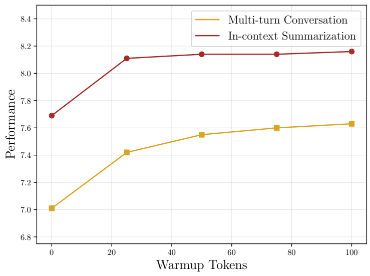

Figure 5: Ablation for the role of `warm_up_index` hyperparamter of FFN-SkipLLM on the performance.

	Dense	5%	10%	20%	30%	50%
FLOPs Reduction	0	∼0.3B	∼0.8B	∼1.7B	∼2.5B	∼4.3B
Throughput	9.08	8.35	9.17	10.22	10.89	12.45

Table 4: End-to-end decoding FLOPs reduction of LLaMa-2 7B model using FFN-SkipLLM.

15% which suggests the importance of cold regions and input-adaptivity.

5 Additional Results and Ablation

5.1 Influence of Warm-up Index on Performance

As discussed in Section 3, developing a high-quality KV during the initial pahse of decoing can significantly help in reducing hallucination and generation of repetitive tokens. FFN-SkipLLM incorporates this observation by incorporating an hyperparameter `warm_up_index`. We conducted a ablation study to understand the role of warmup tokens generation with full model capacity on the final performance on two evaluation tasks (in-context summarization and multi-turn converation) as presented in Figure 5. It can be clearly observed that the FFN-SkipLLM enjoys a significant benefit in performance with merely 25-30 warmup tokens which start saturating with further increase.

5.2 Inference Speedup Analysis

In this section, we analysze the speedup acheived by FFN-SkipLLM which attempts to skip redundant feed-forward computation, as presented in Table 4. The reported speedups correspond to end-to-end decode throughput of LLaMA-V2-7B model on MT-Bench dataset on an Nvidia RTX A6000 GPU using HuggingFace Accelerate. We also reported the total approximate FLOPs reduction acheived due to skipping computationally heavy feed-forward blocks of transformer layer. It is evident that FFN-SkipLLM can delivers a significant inference speedup compared to the dense model which becomes more evident with growing skipping ratio. Due to additional computational overhead of cosine similarity monitoring, we find that noticeable FLOPs reduction couldn't reflect in throughput at 5% skip ratio but becomes visible with increase in skip ratio.

6 Conclusion

In this paper, we explore an orthogonal dimension for layer-skipping and early-exit strategies that suffer from KV cache issues leading to the hallucination of misinformation and token collapse. We propose **FFN-SkipLLM**, a novel fine-grained skip strategy of autoregressive LLMs which is an input-adaptive feed-forward skipping strategy that can skip $\sim$ 25-30% of FFN blocks of LLMs with marginal change in performance on knowledge-intensive tasks. FNN-Skip LLM is built on the core observation of monotonically increasing redundancy within the FFN blocks of LLMs. Our future work includes exploring parameter-efficient continual fine-tuning techniques to push the performance of FFN-SkipLLM for high skip ratios.

7 Limitations

Our work has limitations. Firstly, all our experiments are conducted using LLaMa-v2 7B model and we plan to extend our work to other large models where we expect the performance benefits to be more noticeable given their ability to be compressed to high degree with marginal performance drop (Frantar and Alistarh, 2023). Secondly, due to the novelty of our approach in exploring FFN block skipping instead of conventional layer dropping, our baselines are self-curated along with SoTA layer-dropping baselines. Thirdly, one major limitation of our work is scaling FFN-SkipLLM for non-trivial skipping ratios ($\geq 35\%$) without a significant performance drop. Despite the acknowledged limitations, we beleive that our proposed framework and the unique insights will inspire future work focusing on efficient and compute-constrained LLM inference pipelines.

References

Sangmin Bae, Jongwoo Ko, Hwanjun Song, and Se-Young Yun. 2023a. Fast and robust early-exiting framework for autoregressive language models with synchronized parallel decoding. *arXiv preprint arXiv:2310.05424*.

Sangmin Bae, Jongwoo Ko, Hwanjun Song, and Se-Young Yun. 2023b. Fast and robust early-exiting framework for autoregressive language models with synchronized parallel decoding. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5910–5924, Singapore. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. *arXiv preprint arXiv:1606.02858*.

Runjin Chen, Tong Zhao, Ajay Jaiswal, Neil Shah, and Zhangyang Wang. 2024. Llaga: Large language and graph assistant. *arXiv preprint arXiv:2402.08170*.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. 2020. The lottery ticket hypothesis for pre-trained bert networks. *Advances in neural information processing systems*, 33:15834–15846.

Yanxi Chen, Xuchen Pan, Yaliang Li, Bolin Ding, and Jingren Zhou. 2023a. Ee-llm: Large-scale training and inference of early-exit large language models with 3d parallelism. *arXiv preprint arXiv:2312.04916*.

Yanxi Chen, Xuchen Pan, Yaliang Li, Bolin Ding, and Jingren Zhou. 2023b. Ee-llm: Large-scale training and inference of early-exit large language models with 3d parallelism. *ArXiv*, abs/2312.04916.

Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, et al. 2023c. Exploring the potential of large language models (llms) in learning on graphs. *arXiv preprint arXiv:2307.03393*.

Luciano Del Corro, Allie Del Giorno, Sahaj Agarwal, Bin Yu, Ahmed Awadallah, and Subhabrata Mukherjee. 2023. Skipdecode: Autoregressive skip decoding with batching and caching for efficient llm inference. *arXiv preprint arXiv:2307.02628*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023a. Qlora: Efficient finetuning of quantized llms. *ArXiv*, abs/2305.14314.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. 2023b. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *ArXiv*, abs/2306.03078.

Keyu Duan, Qian Liu, Tat-Seng Chua, Shuicheng Yan, Wei Tsang Ooi, Qizhe Xie, and Junxian He. 2023. Simteg: A frustratingly simple approach improves textual graph learning. *arXiv preprint arXiv:2308.02565*.

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

Alexander R Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir R Radev. 2019. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. *arXiv preprint arXiv:1906.01749*.

Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *ArXiv*, abs/2210.17323.

Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. 2020. Dynabert: dynamic bert with adaptive width and depth. In *Proceedings of the 34th International Conference on Neural Information*

Processing Systems, NIPS'20, Red Hook, NY, USA. Curran Associates Inc.

Mohit Iyyer, Jordan L. Boyd-Graber, Leonardo Max Batista Claudino, Richard Socher, and Hal Daumé. 2014. A neural network for factoid question answering over paragraphs. In *Conference on Empirical Methods in Natural Language Processing*.

Sameer Jain, Vaishakh Keshava, Swarnashree Mysore Sathyendra, Patrick Fernandes, Pengfei Liu, Graham Neubig, and Chunting Zhou. 2023. Multi-dimensional evaluation of text summarization with in-context learning. *arXiv preprint arXiv:2306.01200*.

Ajay Jaiswal, Shiwei Liu, Tianlong Chen, and Zhangyang Wang. 2023a. The emergence of essential sparsity in large pre-trained models: The weights that matter. *arXiv preprint arXiv:2306.03805*.

Ajay Jaiswal, Liyan Tang, Meheli Ghosh, Justin F Rousseau, Yifan Peng, and Ying Ding. 2021. Radbert-cl: Factually-aware contrastive learning for radiology report classification. In *Machine Learning for Health*, pages 196–208. PMLR.

Ajay Kumar Jaiswal, Shiwei Liu, Tianlong Chen, Ying Ding, and Zhangyang Wang. 2023b. Instant soup: Cheap pruning ensembles in a single pass can draw lottery tickets from large models. In *International Conference on Machine Learning*, pages 14691–14701. PMLR.

Ajay Kumar Jaiswal, Haoyu Ma, Tianlong Chen, Ying Ding, and Zhangyang Wang. 2022. Training your sparse neural network better with any mask. In *International Conference on Machine Learning*, pages 9833–9844. PMLR.

Kelvin Jiang, Dekun Wu, and Hui Jiang. 2019. Freebaseqa: A new factoid qa data set matching trivia-style question-answer pairs with freebase. In *North American Chapter of the Association for Computational Linguistics*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

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

Jeonghoon Kim, Jung Hyun Lee, Sungdong Kim, Joonsuk Park, Kang Min Yoo, Se Jung Kwon, and Dongsoo Lee. 2023. Memory-efficient fine-tuning of compressed large language models via sub-4-bit integer quantization. *ArXiv*, abs/2305.14152.

Xin Lai, Zhuotao Tian, Yukang Chen, Yanwei Li, Yuhui Yuan, Shu Liu, and Jiaya Jia. 2023. Lisa: Reasoning segmentation via large language model. *arXiv preprint arXiv:2308.00692*.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. 2019. Snip: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*.

Noah Lee, Na Min An, and James Thorne. 2023. Can large language models infer and disagree like humans? *ArXiv*, abs/2305.13788.

Lei Li, Yankai Lin, Deli Chen, Shuhuai Ren, Peng Li, Jie Zhou, and Xu Sun. 2021a. CascadeBERT: Accelerating inference of pre-trained language models via calibrated complete models cascade. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 475–486, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Tianhao Li, Sandesh Shetty, Advaith Kamath, Ajay Jaiswal, Xiaoqian Jiang, Ying Ding, and Yejin Kim. 2024. Cancergpt for few shot drug pair synergy prediction using large pretrained language models. *npj Digital Medicine*, 7(1):40.

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

Long Lian, Boyi Li, Adam Yala, and Trevor Darrell. 2023. Llm-grounded diffusion: Enhancing prompt understanding of text-to-image diffusion models with large language models. *arXiv preprint arXiv:2305.13655*.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. 2023. Awq: Activation-aware weight quantization for llm compression and acceleration. *ArXiv*, abs/2306.00978.

Junling Liu, Chao Liu, Peilin Zhou, Qichen Ye, Dading Chong, Kang Zhou, Yueqi Xie, Yuwei Cao, Shoujin Wang, Chenyu You, et al. 2023a. Llmrec: Benchmarking large language models on recommendation task. *arXiv preprint arXiv:2308.12241*.

Shiwei Liu, Tianlong Chen, Zhenyu Zhang, Xuxi Chen, Tianjin Huang, Ajay Jaiswal, and Zhangyang Wang. 2023b. Sparsity may cry: Let us fail (current) sparse neural networks together! *arXiv preprint arXiv:2303.02141*.

Weijie Liu, Peng Zhou, Zhiruo Wang, Zhe Zhao, Haotang Deng, and Qi Ju. 2020. FastBERT: a self-distilling BERT with adaptive inference time. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6035–6044, Online. Association for Computational Linguistics.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. 2023c. Llm-qat: Data-free quantization aware training for large language models. *arXiv preprint arXiv:2305.17888*.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023. Chameleon: Plug-and-play compositional reasoning with large language models. *arXiv preprint arXiv:2304.09842*.

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2024. Shortgpt: Layers in large language models are more redundant than you expect. *arXiv preprint arXiv:2403.03853*.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

Chen Qian, Huayi Tang, Zhirui Yang, Hong Liang, and Yong Liu. 2023. Can large language models empower molecular property prediction? *arXiv preprint arXiv:2307.07443*.

Chengwei Qin, Aston Zhang, Zhuosheng Zhang, Jiaao Chen, Michihiro Yasunaga, and Diyi Yang. 2023. Is chatgpt a general-purpose natural language processing task solver? *arXiv preprint arXiv:2302.06476*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *arXiv preprint arXiv:2302.00083*.

Tomohiro Sawada, Daniel Paleka, Alexander Havrilla, Pranav Tadepalli, Paula Vidas, Alexander Kranias, John J Nay, Kshitij Gupta, and Aran Komatsuzaki. 2023. Arb: Advanced reasoning benchmark for large language models. *arXiv preprint arXiv:2307.13692*.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. 2022. Confident adaptive language modeling.

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Daniel Y Fu, Zhiqiang Xie, Beidi Chen, Clark Barrett, Joseph E Gonzalez, et al. 2023. High-throughput generative inference of large language models with a single gpu. *arXiv preprint arXiv:2303.06865*.

Tianxiang Sun, Xiangyang Liu, Wei Zhu, Zhichao Geng, Lingling Wu, Yilong He, Yuan Ni, Guotong Xie, Xuanjing Huang, and Xipeng Qiu. 2022. A simple hash-based early exiting approach for language understanding and generation.

Peng Tang, Pengkai Zhu, Tian Li, Srikar Appalaraju, Vijay Mahadevan, and R. Manmatha. 2023. Deed: Dynamic early exit on decoder for accelerating encoder-decoder transformer models. *Preprint*, arXiv:2311.08623.

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

Wenhai Wang, Zhe Chen, Xiaokang Chen, Jiannan Wu, Xizhou Zhu, Gang Zeng, Ping Luo, Tong Lu, Jie Zhou, Yu Qiao, et al. 2023. Visionllm: Large language model is also an open-ended decoder for vision-centric tasks. *arXiv preprint arXiv:2305.11175*.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2023. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*.

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

Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. 2023. Natural language is all a graph needs. *arXiv preprint arXiv:2308.07134*.

Lu Yin, Ajay Jaiswal, Shiwei Liu, Souvik Kundu, and Zhangyang Wang. 2023a. Pruning small pre-trained weights irreversibly and monotonically impairs "difficult" downstream tasks in llms.

Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Mykola Pechenizkiy, Yi Liang, Zhangyang Wang, and Shiwei Liu. 2023b. Outlier weighed layerwise sparsity (owl): A missing secret sauce for pruning llms to high sparsity. *arXiv preprint arXiv:2310.05175*.

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

LI Zhangheng, Shiwei Liu, Tianlong Chen, AJAY KUMAR JAISWAL, Zhenyu Zhang, Dilin Wang, Raghuraman Krishnamoorthi, Shiyu Chang, and Zhangyang Wang. 2023. Sparse cocktail: Every sparse pattern every sparse ratio all at once.

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

Wei Zhu. 2021. LeeBERT: Learned early exit for BERT with cross-level optimization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2968–2980, Online. Association for Computational Linguistics.

Terry Yue Zhuo. 2023. Large language models are state-of-the-art evaluators of code generation. *arXiv preprint arXiv:2304.14317*.

A Appendix

A.1 Background Work

Recent advances in model compression (pruning, quantization, and distillation) have been very successful in democratizing LLMs, allowing them to perform inference on consumer-grade GPUs. In contrast to their static nature, input-dependent early-exit or layer-dropping strategies present a unique potential for faster inference for new gigantic auto-regressive models during token-by-token generation. The majority of existing approaches primarily has been around BERT-scale encoder models (Hou et al., 2020; Li et al., 2021a; Liu et al., 2020; Xin et al., 2020; Zhu, 2021).

A notable challenge in auto-regressive generation tasks is managing Key-Value (KV) caching, a process that stores the keys and values from attention layers corresponding to previously generated tokens to accelerate sequence generation. However, if a token is generated via early exiting, the KV caches for all subsequent layers are missing, complicating the generation of future tokens that exit at layers beyond the initial exiting layer. This challenge has been acknowledged in the literature, and various strategies have been proposed to address it. One method (Elbayad et al., 2019; Li et al., 2021b; Schuster et al., 2022) duplicates the hidden states from the current token's exiting layer to subsequent layers, which act as the KV cache for generating future tokens. Although being efficient, it causes deviation in the inference process and generates sub-optimal outputs.

Another approach (Del Corro et al., 2023) predetermines the exiting layers for all tokens, which guarantees later tokens always exits at earlier layers, thus ensuring KV caches are always present. However,this approach suffers from degrading performance for as token length increases, and pinpointing the optimal exiting parameters to balance model performance with inference efficiency is non-trivial. The third strategy (Bae et al., 2023b; Tang et al., 2023) stores the hidden states of previous tokens that early-exited. When a KV cache missing occurs, a batched forward pass wtih current and recent tokens is conducted, materializing the missing KV cache. In the worst-case scenario, this approach requires utilizing the full network, thus negating the intended efficiency benefits. In contrast to these work, our work explores an orthogonal direction to layer skipping and focuses on FFN-block skipping which circumvents the hassle and issues with KV caching and can effectively ignore two-thirds of parameter counts.

GENERAL QUESTION PROMPT >> You are a helpful and precise assistant for checking the quality of the answer.", "prompt_template": "

```
[Question]\n{question}\n\n[The Start of Assistant 1's Answer]\n{answer_1}\n\n[The End of Assistant 1's Answer]\n\n[The Start of
Assistant 2's Answer]\n{answer_2}\n\n[The End of Assistant 2's Answer]\n\n[System]\n{prompt}\n\n", "defaults": {"prompt": "We would
like to request your feedback on the performance of two AI assistants in response to the user question displayed above.\nPlease
rate the helpfulness, relevance, accuracy, level of details, factual information, and length of their responses. Each assistant
receives an overall score on a scale of 1 to 10, where a higher score indicates better overall performance.\nPlease first output a
single line containing only two values indicating the scores for Assistant 1 and 2, respectively. The two scores are separated by a
space. In the subsequent line, please provide a comprehensive explanation of your evaluation, avoiding any potential bias and
ensuring that the order in which the responses were presented does not affect your judgment."}
```

CODING QUESTION PROMPT >> You are a helpful and precise assistant for checking the quality of the answer.", "prompt_template": "

```
[Question]\n{question}\n\n[The
Start of Assistant 1's Answer]\n{answer_1}\n\n[The End of Assistant 1's Answer]\n\n[The Start of Assistant 2's
Answer]\n{answer_2}\n\n[The End of Assistant 2's Answer]\n\n[System]\n{prompt}\n\n", "defaults": {"prompt": "Your task is to
evaluate the coding abilities of the above two assistants. They have been asked to implement a program to solve a given problem.
Please review their code submissions, paying close attention to their problem-solving approach, code structure, readability, and
the inclusion of helpful comments.\n\nPlease ensure that the assistants' submissions:\n\n1. Correctly implement the given problem
statement.\n2. Contain accurate and efficient code.\n3. Include clear and concise comments that explain the code's logic and
functionality.\n4. Adhere to proper coding standards and best practices.\n\nOnce you have carefully reviewed both submissions,
provide detailed feedback on their strengths and weaknesses, along with any suggestions for improvement. You should first output a
single line containing two scores on the scale of 1-10 (1: no code/no sense; 10: perfect) for Assistant 1 and 2, respectively. Then
give extra comments starting from the next line."}
```

MATHS QUESTION PROMPT >> You are a helpful and precise assistant for checking the quality of the answer.", "prompt_template": "

```
[Question]\n{question}\n\n[The
Start of Assistant 1's Answer]\n{answer_1}\n\n[The End of Assistant 1's Answer]\n\n[The Start of Assistant 2's
Answer]\n{answer_2}\n\n[The End of Assistant 2's Answer]\n\n[System]\n{prompt}\n\n", "defaults": {"prompt": "We would like to
request your feedback on the mathematical proficiency of two AI assistants regarding the given user question displayed
above.\nFirst, please solve the problem independently, without referring to the answers provided by Assistant 1 and Assistant
2.\nAfterward, please examine the problem-solving process of Assistant 1 and Assistant 2 step-by-step to ensure their correctness,
identifying any incorrect steps if present. Your evaluation should take into account not only the answer but also the problem-
solving steps.\nFinally, please output a Python tuple containing two numerical scores for Assistant 1 and Assistant 2, ranging from
1 to 10, respectively. If applicable, explain the reasons for any variations in their scores and determine which assistant
performed better."}
```

Figure 6: Examples of prompts used for different categories to evaluate the compressed LLM ASSISTANT *wrt.* GPT-3.5 ASSISTANT using GPT-4 as a Judge.