Speculative Prefill: Turbocharging TTFT with Lightweight and Training-Free Token Importance Estimation

Jingyu Liu [1] **Beidi Chen** [2] **Ce Zhang** [1]

Abstract

Improving time-to-first-token (TTFT) is an essentially important objective in modern large language model (LLM) inference engines. Optimizing TTFT directly results in higher maximal QPS and meets the requirements of many critical applications. However, boosting TTFT is notoriously challenging since it is compute-bounded and the performance bottleneck shifts from the self-attention to the MLP part. We present SPECPREFILL[1], a training free framework that accelerates the inference TTFT for both long and medium context queries based on the following insight: LLMs are generalized enough to preserve the quality given only a *carefully chosen* subset of prompt tokens. At its core, SPECPREFILL leverages a lightweight model to speculate locally important tokens based on the context. These tokens, along with the necessary positional information, are then sent to the main model for processing. We evaluate SPECPREFILL with a diverse set of tasks, followed by a comprehensive benchmarking of performance improvement both in a real end-to-end setting and ablation studies. SPECPREFILL manages to serve `Llama-3.1-405B-Instruct-FP8` with up to 7× maximal end-to-end QPS on real downstream tasks and 7.66× TTFT improvement.

1. Introduction

Large Language Models (LLMs) represent a transformative innovation in artificial intelligence, enabling machines to understand and generate human-like languages (Bubeck et al., 2023; Wei et al., 2022; Feng et al., 2024). Many SOTA models have been developed, such as GPT-4 (OpenAI et al., 2024), the Llama family (Grattafiori et al., 2024), DeepSeek R1 (DeepSeek-AI et al., 2025), Mistral (Jiang et al., 2023a), Gemini (Team et al., 2024), and Qwen2 (Yang et al., 2024), to meet the increasing expectations of users. In order to broaden their real-world applications, one essential requirement is to build an efficient serving engine that can satisfy various requirements (Miao et al., 2023; Kwon et al., 2023; Zheng et al., 2024; Shoeybi et al., 2020).

There are several fundamental reasons why TTFT stands so pivotal: 1) many applications require a fast response time that directly influences how users perceive the responsiveness of the system and 2) more importantly, TTFT determines the scaling of maximal QPS an inference engine can support as shown in Figure 1. However, optimizing TTFT is an arduous task mostly because the prefill stage is largely compute-bounded and the computational bottleneck can change depending on the prompt length and batch size. For example, many works focus on improving the self-attention speed (Dao et al., 2022; Jiang et al., 2024a), but in reality, there is still a huge traffic of large-batch short to medium context queries where it is the MLP part that clogs the whole system. Despite achieving impressive results, prior works that target the prefill phase either require a post-training adaptation (Qiao et al., 2024; Horton et al., 2024) or scale less efficiently (Shi et al., 2024).

Inspired by those work, we found a key insight that LLMs can retain most of its performance when given only *a carefully chosen subset* of tokens from the prompt, and the model is able to adapt to that in a zero-shot manner. SPECPREFILL optimizes the TTFT by leveraging a secondary lightweight model to speculate locally important tokens. Only these tokens are sent later to the base model. It reduces the total FLOPS by a factor proportional to the percentage of token drop. SPECPREFILL does not require any fine-tuning and is ready to be deployed and scaled to larger models. We summarize our key contributions:

- We present a conceptually simple, effective, and noval framework called SPECPREFILL that significantly

[1]Department of Computer Science, The University of Chicago, Chicago, IL, USA [2]Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA. Correspondence to: Jingyu Liu <jingyu6@uchicago.edu>, Ce Zhang <cez@uchicago.edu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

[1]The code with experiment reproduction is available at https://github.com/anonymous/speculative_prefill.

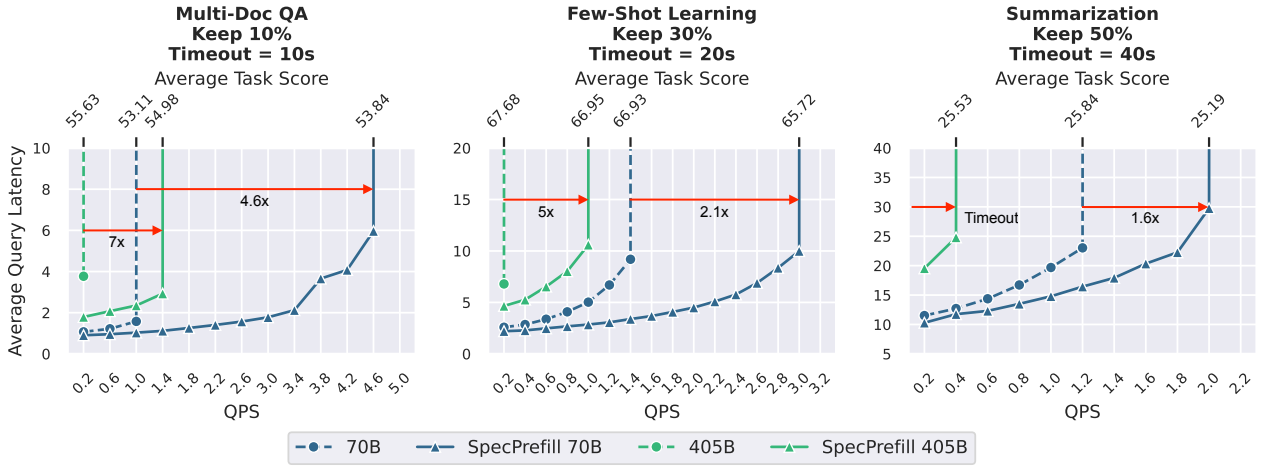

Figure 1. **Speculative Prefill QPS Improvement:** In an end-to-end server-client setting with real world datasets, we benchmark the average query latency under a given fixed timeout when sending queries at a constant QPS. SPECPREFILL significantly improves the maximum QPS supported by the vLLM server as well as the latency compared to not using it. When we reach low keep rate, we can even serve the 405B model with SPECPREFILL to run more efficiently than the 70B model. As the base model size increases and keep rate drops, we can get 7× end-to-end QPS boost while only occurring $< 5\%$ accuracy.

improve the prefill phase, hence the maximal QPS, of LLM inference without any fine-tuning or adaptation.

- We conducted comprehensive evaluations both on real and synthetic datasets to demonstrate its effectiveness and limitations, giving a full picture of the expected benefits when deployed to productions.
- We implemented our method on industry standard serving engines, and benchmark its performance in both an end-to-end fashion and ablation experiments. The end result is a system that can serve `Llama-3.1-405B-Instruct-FP8` with up to 7× maximal QPS under the same system specification and 7.66× reduced TTFT while maintaining decent accuracy.
- Our method can be easily combined with techniques from quantization, KV eviction, and speculative decoding, making it an ideal add-on to existing engines.

2. Background

2.1. Inference Bottlenecks

Improving LLM inference efficiency has been extensively studied in prior work (Miao et al., 2023; Yuan et al., 2024). We review works that focus on different aspects when dealing with real serving systems where the bottlenecks are quite different under various serving requirements (e.g. long context domains, latency sensitive applications, etc) (Kwon et al., 2023; Zheng et al., 2024).

LLM inference can be roughly divided into two major procedures, namely the prefill phase where the model computes the KV necessary for producing the output based on the query and the decoding phase where the model predicts new token auto-regressively.

2.2. Decoding Acceleration

The decoding phase is mostly memory-bounded, and therefore reducing the amount of data to move around will effectively help improve the latency. As a result, explicitly manipulating the KV cache has been extremely successful with many strategies: H2O (Zhang et al., 2023) and StreamingLLM (Xiao et al., 2024b) idenfitied key insights to KV dynamics which is used to evict less essential KV caches during decoding. CacheGen (Liu et al., 2024), Q-Hitter (Zhang et al., 2024b), and ShadowKV (Sun et al., 2024) apply efficient techniques to compress/quantize, store, and transmit KV caches to reduce memory overhead. Speculative decoding relies on the insight that hides the memory latency by concurrently verifying several speculated tokens from either a draft model or itself (Leviathan et al., 2023; Zhang et al., 2024a; Xia et al., 2024).

Despite being crucially important, decoding speed is not the only factor that influences the overall inference pipeline and we will review why *sometimes the prefill time optimization is even more essential* in many cases.

2.3. Prefill Acceleration

The time-to-first-token (TTFT) is crucially important both from a user experience but also the system serving perspective (Sec 4.7). In many critical applications, the input token length can often eclipse that of the generation tokens (e.g. 10:1 ratio) in real traffic (Qiao et al., 2024). Unlike

the decoding phase, however, the prefill phase is usually compute-bounded and the cost for the MLP calculation and the communication of tensor parallelism quickly becomes a bottleneck.

Many prior works have explored ways to make self-attention faster: Flash-attention series (Dao et al., 2022; Dao, 2023) compute the exact attention using carefully designed hardware-aware algorithms. Special (both static and dynamic) attention masks are designed for sparse calculation such as LongFormer (Beltagy et al., 2020), MInference (Jiang et al., 2024a), FlexPrefill (Lai et al., 2025), Hip Attention (Lee et al., 2025), Sample Attention (Zhu et al., 2024), and Duo Attention (Xiao et al., 2024a). However, none of these directly make the MLP part faster like SPECPREFILL. SPECPREFILL achieves consistent efficiency improvements in various regimes because it skips parts of the attention + MLP calculation and the all-reduce overhead, which proves to be effective especially when the ratio of $\frac{\text{batch size}}{\text{sequence length}}$ is large (Xiong et al., 2023).

Orthogonal to techniques such as prompt compression/rewrite (Jiang et al., 2023b; 2024b; Li et al., 2023), layer dropping (Elhoushi et al., 2024), and weight quantization methods (Lin et al., 2024), we explore selecting *important* prompt tokens to skip the full forward computation. GemFilter (Shi et al., 2024) uses an extra pass to get a model's own middle layer attention information that decides on what tokens to keep for the real forward. Contrast to this, we apply a separate and cheaper model to speculate locally important tokens via token transferability, which can scale more efficiently than GemFilter. Concurrent to ours, SwiftKV (Qiao et al., 2024) learns to skip later layers by reusing the past layers' KV, which achieves up to 50% TTFT reduction (SPECPREFILL can reach up to 87% TTFT reduction). Unlike our zero-shot requirement, they require extra light-weight fine-tuning due to modified model behavior. It is worth noting that our method approaches the problem in a different way, which makes them complimentary to each other. Finally, akin to our motivation, KV Prediction (Horton et al., 2024) proposes to adapt a cheaper system (i.e. a learned auxiliary network and a KV predictor) to predict the KV cache of the base model, thus bypassing the original KV computation. We show that SPECPREFILL can accomplish better TTFT reduction than theirs, without introducing extra overhead when coupled with speculative decoding (Leviathan et al., 2023) while maintaining competitive quality.

3. Speculative Prefill

In this section, we present SPECPREFILL by first describing its high-level algorithm, followed by several design choices that mitigate various biases, and a detailed implementation account. Finally, we touch a bit on how to integrate SPECPREFILL to speculative decoding, forming a full small-model-assisted inference paradigm.

3.1. Overall Architecture

SPECPREFILL follows a conceptually simple architecture where a usually less expensive model is chosen as the speculator model that predicts contextually important tokens given a prompt. The speculated tokens, alone with the original position information, are then fed to the main model for processing. In the following section, we will discuss two central design choices in more details, namely the token estimation algorithm and the selection strategy. Note that SPECPREFILL can be seamlessly integrated with speculative decoding in which the small model can work both in the prefill stage for token selection and the decoding stage for drafting proposals, making our approach almost free to integrate and deploy.

3.2. Token Importance Speculation

The goal here is to select which tokens are contextually important for a given query and send those along with necessary positional information for the main model. The procedure starts with calculating the attention scores from the speculator, which uses the last token's attention score w.r.t. the context as the surrogate for measuring token importance:

$$a_{ij} := \text{Softmax}(Q_{M+j}K^T))_i, \forall 0 \leq i < M, 0 \leq j < N$$

where M is the context length, N is the number of look-ahead steps, and a_{ij} is the attention score for the ith token in the prompt w.r.t. the jth decoded token, assuming we're looking at a particular layer.

We build on top of this by aggregating the scores over the whole speculator model (Section 3.2.2) with potential look-ahead (Section 3.2.1) and select tokens based on chunks (Section 3.2.3). The subset of chosen tokens with their original positional information (Section 3.2.4) will then be used for the main model's inference.

3.2.1. MITIGATE POSITION BIAS VIA LOOK-AHEAD

Prior works have shown that there are many biases for attention scores, such as the sink phenomenon (Xiao et al., 2024b) (the first couple of tokens tend to have higher weights) and the proximity bias (tokens closer to the output tend to have higher weights (Lv et al., 2024)). To mitigate these issues, instead of relying on the attention score of the last token alone, we further decode the speculator by N steps and obtain the attention information from the new N tokens (Wan et al., 2024). N here serves as a trade-off between bias and budget, which can substantially increase the performance for shorter context queries.

3.2.2. Aggregated Attention Score as Token Importance

Given the full attention scores of the speculator, we decide to use a max-mean aggregation strategy to map to scalar token importance. Formally, given an attention score tensor of shape $[N, L, S, H]$ where N is the number of look-ahead tokens, L is the number of layers, S is the sequence length, and H is the number of heads, we take the maximum over H and L dimension to make salient tokens stand out, and average over N to account for fair token contribution.

3.2.3. Denoise Attention Scores by Chunk Selection and Pooling

It has also been observed in concurrent works (Lv et al., 2024) that tokens that are positioned nearby share similarity in importance. We take this insights to select tokens by chunks in order to reduce the variance of our token importance estimation. Specifically, we chunk the context contiguously and average the token score within each block, and then we select the Top-K blocks. In order to eliminate the artifacts of chunkation, we apply a 1D average pooling before this to smooth the cross block scores.

3.2.4. Restoration of Position IDs

Finally, when we select the subset of tokens based on our compute budget and query compressibility, we also need to restore the position information which are also sent to the main model. Basically, instead of using a contiguous position ids as before, we send a potentially non-continuously increasing position ids which are obtained from tokens' positions in the original context. In addition to that, we also need to explicitly set the decoding token position to the context length in case we dropped tokens before the first decoding token. An example is shown below with ten prompt tokens and three decoding tokens (bold):

$$\begin{aligned}\text{Original Pos Ids: } &[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]\\ \text{Speculated Pos Ids: } &[0, 1, 3, 6, 7]\\ \text{Decoding Pos Ids: } &[0, 1, 3, 6, 7, \mathbf{10, 11, 12, ...}]\end{aligned}$$

where the **bold indices** are the decoding positions which are offset based on the original position information. We found this design choice to be crucially essential, especially for position-sensitive tasks such as synthetic tasks involving retrieval and counting.

3.3. Implementation Details

We describe both the high-level procedure and the implementation details of SpecPrefill in this section. In Algorithm 1, we list the high-level steps of conducting SpecPrefill. Our implementation is based on creating a monkey patch on top of vLLM (Kwon et al., 2023) which only needs a few line of code along with a configuration file to enable SpecPrefill. The KV cache is not necessary if we do not need to look-ahead for our speculator, which can save lots of memory allocation. However, we do need to explicitly store the queries of the decoded tokens (including the last token of the input query) which we later retrieve to compute the attention score. Note that a specific mapping (e.g. slot mapping in vLLM) might be kept track of to retrieve the right data. For batched look-ahead, we only consider tokens that are valid by checking afterwards whether they are equal to the EOS tokens. Finally, we want to mention that despite being being a sequential implementation, we can actually split the process of speculation into a separate procedure and decouple from the inference of the main model by adding a new layer of scheduling, which we leave as a future work.

3.4. Relation to Speculative Decoding

Speculative decoding has been proven to be extremely successful at accelerating the decoding TPS. SpecPrefill can be seamlessly combined with speculative decoding by sharing the same draft model. Since speculative decoding itself requires a full forward pass of the context, SpecPrefill will provide the necessary KV information required for subsequent decoding speculation, hence amortizing the overhead. This will open-up a huge space of possibilities, and lead to the first paradigm of an inference system that is fully aided by smaller speculators.

Algorithm 1 Speculative Prefill

Require: Base model M, speculator S, look-ahead steps N, batch of mixed requests B, base model QKV cache C_b, speculator KV cache C_s

$B_p, B_d \leftarrow$ *split_prefill_decode_requests* (B)
{Section 3.2.1}
for $i = 1$ to N **do**
  $B'_p \leftarrow$ *model_forward* $(S, B_p, C_s,$ *store_q*=True$)$
  $B_p \leftarrow$ *update_requests* (B_p, B'_p)
  $B_p \leftarrow$ *check_for_eos* (B_p)
end for
if *is_tensor_paralleled* () **then**
  tp_gather_qk (C_s)
end if
$Q, K \leftarrow$ *retrieve_qk* (B_p, C_s)
$A \leftarrow$ *compute_attention_score* (Q, K)
{Section 3.2.2}
$A \leftarrow$ *aggregate_attention_score* (A)
{Section 3.2.3}
$T \leftarrow$ *chunk_select_from_smoothed_attention* (A)
{Section 3.2.4}
$P \leftarrow$ *restore_pos_ids* (T, B_p)
$B \leftarrow$ *merge_requests* (T, P, B_p, B_d)
Return *model_forward* (M, B, C_b)

4. Experiments

In this section, we start with our experiment setup for reproducibility, followed by categorizing prompt compressibility of different queries. We evaluate SPECPREFILL on downstream long context, synthetic context probing, and standard short tasks. Finally, we conclude with a comprehensive efficiency measurement of our system under the real end-to-end setting.

4.1. Setup

We implement SPECPREFILL in vLLM that supports tensor parallelism with the same degree as the main model[2]. Due to its token dropping nature, we focus on evaluating *generative* tasks in this section and include a comprehensive range of benchmarks to fully present its applicability and potential pitfalls. We run all of experiments using a tensor parallelism of 8 for both the speculator and the base model across either 8 NVIDIA H100s or H200s (full system specification in Appendix D and guidance on reproducing results in Appendix C). We choose LLAMA-3.1-8B-INSTRUCT (Grattafiori et al., 2024) with BF16 precision as our speculator for a balance of efficiency and attention transferability and couple it with either `Llama-3.1-70B-Instruct` (BF16) or `Llama-3.1-405B-Instruct-FP8`[3] (fully quantized FP8) as the base model. In terms of token keep rate, we use a fixed percentage (i.e. the ratio of chunks when we do chunk selection) for a given task. In practice, we might devise more adaptive strategy for how many tokens to keep based on the query compressibility discussed next, or delegate the decision to users based on their needs. We leave all these possibilities for prospective applications.

4.2. Query Context Compressibility

We empirically found three types of queries during our evaluations based on the quality difference before and after applying SPECPREFILL:

1. *Information-dense queries*: These queries usually are short and information dense, which naturally makes token dropping less effective because there is no redundancy in the prompt.
2. *Compressible queries*: These queries are those that do not get degradation after removing a significant amount of tokens, often seen in long context tasks.
3. *Noisy queries*: These queries, perhaps surprisingly, get better results after dropping some "noisy" tokens. We hypothesize the reason behind the improvement might be that SPECPREFILL helps remove noisy and distracting tokens in the prompt, hence projecting the prompt to the space where the main model performs better.

We will see examples in each categories in the following evaluations. It is worth noting that we used a fixed keep percentage for our evaluation and it can be tremendously helpful to automatically decide on the percentage based on the query, pushing the limit of SPECPREFILL, which we leave as a future work.

4.3. Baselines and SPECPREFILL Variants

We aim to showcase both the quality and efficiency of SPECPREFILL under a comprehensive set of applications. To do so, we compare *three* variants of SPECPREFILL against *four* baselines:

- *Baselines*: We compare SPECPREFILL against four different baselines: 1) **Base Llama instruct model**. 2) **Sentence RAG**: SPECPREFILL can be framed as a special case of retrieval-augmented (RAG) LLM with the granularity of tokens or blocks and the relevance metric controlled by the speculator's internal knowledge (Li et al., 2024; Gao et al., 2024b; Lewis et al., 2021). Therefore, we implemented two simple sentence-level RAG baselines and report the better one as RAG-LLAMA. 3) **LLMLingua** (Jiang et al., 2023b): SPECPREFILL can also be seen as a context compression technique, and hence we test SPECPREFILL against a text-level compression method. 4) **MInference** (Jiang et al., 2024a): To understand the benefits of skipping the MLP part, we include a sparse attention optimization approach for completeness.
- SPECPREFILL: SPECPREFILL with raw attention scores and ignoring the techniques we discussed in Section 3.2.1 and 3.2.3.
- SPECPREFILL *Full*: SPECPREFILL with all techniques but no look-ahead.
- SPECPREFILL *Full LAH*: SPECPREFILL with all techniques with 8-step look-ahead[4].

4.4. Real Long Context Tasks: LongBench

We start with long context tasks using LongBench (Bai et al., 2024), which consists of six different categories focusing on various aspects of long context modeling abilities.

[2]We expose the API so that it only takes a few line of code to apply SPECPREFILL before initializing vLLM engines.

[3]https://huggingface.co/neuralmagic/Meta-Llama-3.1-405B-Instruct-FP8.

[4]We empirically found that going beyond 16 look-ahead gives minimal performance gain.

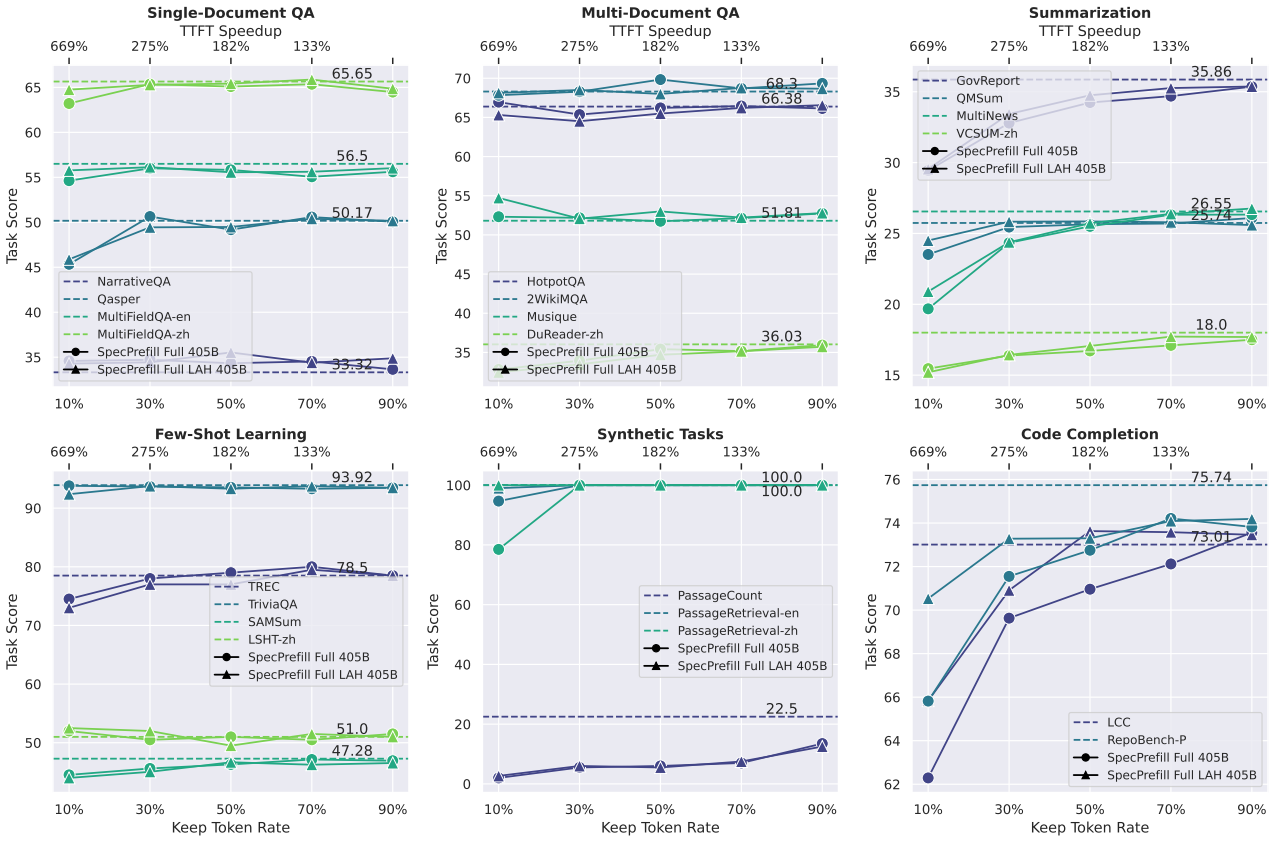

Figure 2. **LongBench Main Result on Llama 405B:** In this figure, we showcase the effectiveness of SPECPREFILL on LongBench, which consists of six categories of long context downstream tasks. In each plot, the dash lines are the results of baseline `Llama-3.1-405B-Instruct-FP8` for each subtask and we benchmark SPECPREFILL with increasing token keep rates. We observe different behaviors such as quality preservation, degradation, and improvement based on the task type.

In Figure 2, we report the main results on LongBench for `Llama-3.1-405B-Instruct-FP8`, whose length information is visualized in Appendix Figure 10. To compliment it, we also include the performance of `Llama-3.1-70B-Instruct` in Appendix Figure 8. We vary the token keep percentage starting from 10% to 90% and draw the baseline model quality using the dash lines for each subtask. To ablate the effect of our design choices, we compare SPECPREFILL and SPECPREFILL *Full LAH* with the 70B models in Appendix Figure 8, and SPECPREFILL *Full* and SPECPREFILL *Full LAH* with the 405B model in Figure 2.

As we can observe, for categories such as Single-Document QA, Multi-Document QA, Few-Shot Learning, SPECPREFILL can preserve most of the quality up to keeping only 10% tokens. For Summarization, we expect to see some degradation in performance as we drop more. Perhaps surprisingly, for the smaller 70B model, we can achieve better quality after we remove some tokens on tasks like Code Completion. As the model size increases, the quality gap between applying SPECPREFILL or not becomes smaller, which indicates that bigger models adapt better with our speculated subset of tokens.

To ablate the effectiveness of techniques discussed in Section 3.2, we compare them separately in Figure 2 and 8 to avoid crowdedness. In both cases, we can see consistent improvement and the benefits of look-ahead are more consistent in shorter context tasks (more details in Sec 4.6).

Finally, we demonstrate the superiority of SPECPREFILL over three different baselines in terms of preserving the quality of the inference for the 70B model. In Table 1, we group tasks in LongBench into categories and list the results with varying degrees of compression rates. For RAG-LLAMA, we use the question to retrieve the relative information from the context (more detailed descriptions are given in Appendix B). For LLMLingua, we follow their official examples and only compress the context, leaving the question and template intact. With the prior knowledge of separated context and question, these two methods are eclipsed by

Table 1. **LongBench 70B Model Comparison:** We compare different methods based on `Llama-70B-Inst` with varying compression rates on LongBench (grouped by task types). * denotes models that not only require context-question separation but also have the true compression rates distinctive from the predefined ones. Among all tested methods, SPECPREFILL achieves superior average performance (underlined scores are the best under the comparable rate).

Model	Compression Rate	Single-Doc QA	Multi-Doc QA	Sum	Few-shot Learning	Code	Synthetic	Avg
Baseline	N/A	50.57	53.11	25.84	66.93	52.33	72.50	53.55
RAG*	10.38%	32.32	41.17	18.86	45.40	44.76	30.42	35.49
	27.68%	38.43	47.41	21.42	50.53	45.80	35.50	39.85
	45.64%	40.53	46.64	22.45	49.52	46.00	43.15	41.38
	63.42%	41.40	47.43	23.30	52.21	46.19	47.22	42.96
	82.22%	43.25	48.16	23.56	51.44	45.92	53.04	44.23
LLMLingua*	∼10%	26.50	32.94	20.95	37.40	45.00	16.33	29.85
	∼30%	38.83	44.02	23.37	42.23	47.27	37.00	38.79
	∼50%	43.64	50.67	24.77	50.96	49.05	60.33	46.57
	∼70%	45.90	52.88	25.44	59.77	51.48	68.50	50.66
	∼90%	45.94	53.91	25.87	60.46	54.06	72.00	52.04
MInference	N/A (Section 4.7.2)	50.46	53.23	25.83	66.36	52.48	69.00	52.89
SPECPREFILL	10%	47.64	52.96	21.74	64.52	63.33	66.25	52.74
	30%	49.47	53.39	24.41	65.83	62.62	67.83	53.92
	50%	50.18	52.56	25.10	65.60	59.91	68.17	53.59
	70%	50.06	52.44	25.51	65.77	58.08	68.67	53.42
	90%	50.26	53.25	25.65	66.35	53.47	70.67	53.27

SPECPREFILL by a large margin under the same rate. For MInference, we use the official searched optimal pattern, and SPECPREFILL can reach 99.7% average score with only 10% keep rate and outperform it with larger keep rate. Since the exact token-level "keep rate" of a sparse attention kernel is not defined, we defer to Section 4.7.2 for a more fair comparison between these two approaches. Overall, SPECPREFILL achieves impressive performance without any fine-tuning or input assumption, further supporting its effectiveness and flexibility.

4.5. Synthetic Context Probing: RULER

In addition to LongBench, we also evaluate SPECPREFILL on a synthetic context probing task to see if SPECPREFILL can preserve effective context lengths. RULER (Hsieh et al., 2024) is a suite of synthetically created tasks with controllable lengths, which ranges from retrieval, multi-hop tracking, real QA datasets, and context aggregation tasks. In Table 2, we include the results for the 70B model with SPECPREFILL that keeps 10% context. As we can observe, SPECPREFILL preserves the quality despite only using one tenth of the tokens except for aggregation tasks, which we believe to fall into the category of information-dense queries that are not our main target application. Take CWE from aggregation category for example: CWE asks for the common words presented in the prompt, which becomes challenging to answer by token dropping. We hope to explore in the future ways of potentially rewriting the queries instead of directly dropping the tokens to mitigate this type of limitation (Jiang et al., 2023b; 2024b). Averaging scores without the aggregation task, we can see that in most context lengths, SPECPREFILL even helps improve the quality[5], suggesting 1) that SPECPREFILL provides both efficiency and performance gains at the same time, and 2) the fact that there are lots of potential redundancy and noise in these synthetic tasks.

4.6. Standard Short Tasks

Unlike prior works on prefill token dropping techniques (Lv et al., 2024; Shi et al., 2024) that do not include regular short context task evaluation, we present a wide range of standard tasks to show the full spectrum of SPECPREFILL's performance and potential caveats. We select tasks spanning general knowledge (Generative MMLU (Hendrycks et al., 2021) and Instruction Following Evaluation (Zhou et al., 2023)), math (GSM8K 8 Shots (Cobbe et al., 2021)), coding (HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021)), and reasoning abilities (Arc Challenge (Clark et al., 2018) and GPQA 8 Shots (Rein et al., 2023)).

In Apendix Figure 6, we showcase the performance of `Llama-3.1-70B-Instruct` on these tasks. Non-surprisingly, prompts from standard tasks without few shot examples are very information dense, making SPECPREFILL less effective with low token keep rate. However, for certain tasks (e.g. MBPP and GPQA), we do observe improved performance when dropping certain tokens. On average, SPECPREFILL can maintain and even surpass the baseline when choosing the right token keep rate.

[5]For 4k Multi-hop Tracking, since we only keep around 400 tokens, we might unintentionally ignore some essential information. But to keep experiment setup more consistent, we list the results here for clarity.

Table 2. **RULER Results on Llama 70B:** We present results of SPECPREFILL with 10% token keep rate on the effective context probing suite RULER with varying context length. SPECPREFILL can preserve the performance of all except for aggregation tasks, which are *less compressible* due to the problem nature as each word in the prompt is important to reason about word frequency and commonality.

Model Name	Task Length	Retrieval Niah Variants	Multi-hop Tracking Variable Checking	QA SQuAD & HotpotQA	Aggregation CWE & FWE	Average w/o Aggregation
Llama-70B-Inst	4k	100.0	100.0	76.9	99.7	92.3
	8K	99.9	100.0	74.7	98.0	91.5
	16K	99.8	100.0	72.0	97.8	90.6
	32K	99.6	100.0	69.8	96.9	89.8
	64K	98.5	99.9	65.1	65.6	87.9
	128K	76.5	56.1	48.2	41.3	60.3
SPECPREFILL with 10% Keep Rate	4K	99.7	89.6	75.2	77.9	88.2
	8K	99.6	100.0	75.6	79.7	91.7
	16K	99.5	99.1	75.3	78.5	91.3
	32K	99.7	100.0	72.6	70.0	90.8
	64K	99.5	99.8	71.9	54.9	90.4
	128K	85.8	55.6	55.3	48.3	65.6

4.7. Efficiency Benchmarking

SPECPREFILL offers great improvement to TTFT, a speedup almost proportional to the percentage of tokens we drop from the speculator, with almost ignorable overhead as we increase the base model size. In this section, we benchmark both the 70B and 405B models under two settings: 1) understanding the average query latency and QPS dynamics with real downstream datasets, and 2) evaluating TTFT with varying sequence lengths on synthetic data. We used one node consisting of eight NVIDIA H200s for all experiments unless separately specified (full system specification is listed in Table 4 from the Appendix D).

4.7.1. AVERAGE QUERY LATENCY UNDER DIFFERENT QPS WITH REAL DOWNSTREAM DATASETS

We want to measure the *real* performance gain we can create in an end-to-end fashion. To do this, we launch a vLLM server with a given model, and an OpenAI API client [6] that sends asynchronous requests at a constant QPS with queries from datasets in LongBench. We measure the client-side per query latency that consists of the prefill stage with several decoding steps based on the maximum budget defined by the task. In Figure 1, we increase the QPS of our client and calculate the average query latency with a given fixed timeout to simulate real-world user needs. For each task category, we draw samples randomly from each subtask and shuffle them before starting the querying, making sure the same set of queries is used over all QPS. As we can observe, all models will follow a standard three-stage pattern: 1) the initial constant stage where latency remains almost unchanged as all queries can be finished before receiving new ones, 2) the middle linear stage where TTFT is small enough but the decoding step might not finish fast enough, and 3) the final timeout stage where the server can not even finish the prefill stage before new requests and all subsequent queries are jammed thereafter. Since the maximum QPS a system can support is $\mathcal{O}(1/TTFT)$[7] given finite timeout, the acceleration from SPECPREFILL will drastically increases the maximal QPS under a fixed timeout, which pushes the transition from stage 2 to stage 3 further later.

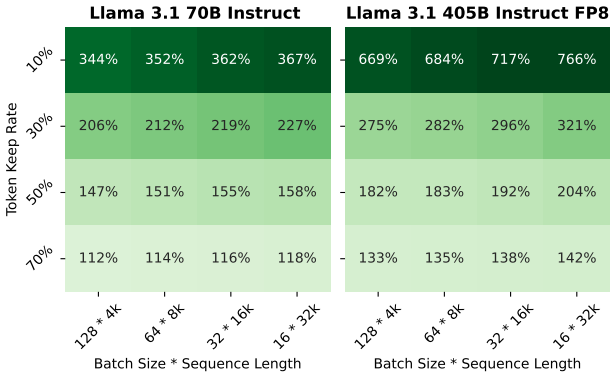

Figure 3. **SPECPREFILL TTFT Improvement:** We present prefill TTFT speed-up using SPECPREFILL under different settings over Llama-3.1-70B-Instruct and Llama-3.1-405B-Instruct-FP8 (achieving up to 7.66x faster TTFT when keeping 10% tokens for the 405B model).

We show that with the help of SPECPREFILL, the 405B model can convert to 7× QPS improvement on a Multi-Doc QA suite from LongBench while maintaining $> 95\%$ accuracy. The results vary as we change the model FLOPS ratio and drop rate. We believe that this type of analysis provides invaluable insights and tangible benefits of SPECPREFILL when deployed to real world systems.

4.7.2. TTFT IMPROVEMENT OVER DIFFERENT BATCH SIZE × SEQUENCE LENGTH PRODUCTS

We try to understand the dynamics of SPECPREFILL under different batch-size-sequence-length products and keep per-

[6] https://github.com/openai/openai-python

[7] If we have finitely large timeout, we would expect SPECPREFILL to support around N times larger maximal QPS if we reduce TTFT by N times.

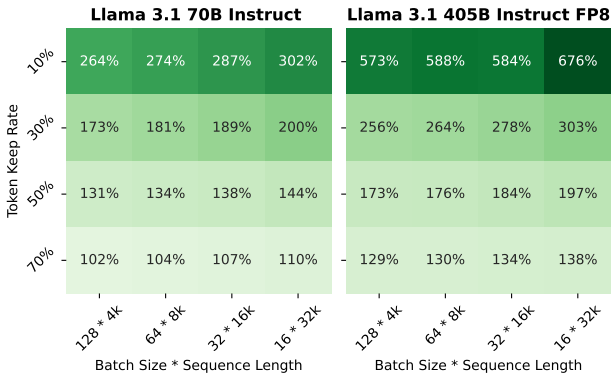

Figure 4. **SPECPREFILL with look-ahead TTFT Improvement:** Complimentary to Figure 3, we also show the relative speedup when using a look-ahead = 8 steps for both the 70B and 405B model.

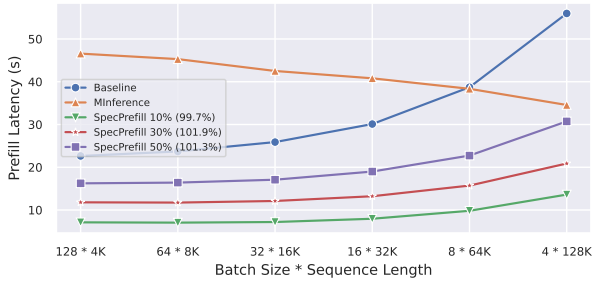

Figure 5. **SPECPREFILL v.s. MInference TTFT on 70B Models:** The superiority of SPECPREFILL becomes more clear as we increase the batch size under 128K context lengths and MInference gradually improves as the context length increases with smaller batch size due to less overhead. Percentages in parenthesis are the relative average scores to that of MInference on LongBench.

centage while isolating the advantage of TTFT. We use the official script from vLLM for latency benchmarking. In Figure 3 and 4.7.2 (with look-ahead), we highlight the TTFT speedup against the vanilla base model without SPECPREFILL, which we produce by setting the maximum decoding step to be 1. As we can see, for both the 70B and 405B models, not only do we see more direct effects of SPECPREFILL but also imply the increasing scaling along the sequence dimension. As the relative FLOPS ratio between the speculator and main model becomes larger, the overhead of speculation starts to become more negligible, which leads to more substantial improvement. In order to better understand the whole system, we include theoretical analysis of the overhead and the performance gains in Appendix E.

One salient feature of SPECPREFILL is the ability to skip some MLP calculations, one of the major benefits of which is its strong performance under large batch size. To further understand the trade-offs, we compare the TTFT of SPECPREFILL and MInference using the 70B model (one node of 8xH100s with TP=8). In Figure 4.7.2, we can see that SPECPREFILL outperforms both the dense model and MInference when using large batch size and short to medium length prompts (i.e. less than 128K tokens). The main limitation for MInference under this scenario is the overhead introduced due to additional approximations, which gets amortized only when the ratio $\frac{\text{sequence length}}{\text{batch size}}$ becomes large enough (Jiang et al., 2024a). Overall, we found SPECPREFILL to be able to achieve $2.54\times$ to $6.54\times$ relative speedup over MInference with 99.5% quality performance. We believe that this experiment provides a more comprehensive guide to practitioners on which method to choose for specific application needs (i.e. large batches plus less than 128K or ultra long prompts).

5. Limitation

The main limitation of SPECPREFILL is akin to all token-dropping based method: 1) they do not support explicit logit outputs for all input tokens, and hence we focus on generative evaluation, 2) without explicit context recomputation, multi-turn conversation can potentially fail (Li et al., 2025), which poses a trade-off of whether we should compute and store all KV caches during the prefill. Given that we have the full knowledge of the speculator, we advertise for more principled method to estimate token importance beyond attention scores and efficient methods for dynamic KV recomputation. Finally, we believe that a robust algorithm that determines how many tokens are required for a given prompt will be of great use for SPECPREFILL.

6. Conclusion

In this work, we introduce SPECPREFILL, a training-free framework for accelerating the LLM inference by speculating what tokens to drop with the help of a smaller speculator model. Leveraging the insight that models of different sizes within the same family can usually transfer token importance, SPECPREFILL not only achieves substantial improvement on TTFT, which leads to $7\times$ maximal supported QPS of an inference system, but also reduces the memory required. SPECPREFILL can also be readily combined with other techniques such as speculative decoding, the combination of which could result in the first unified small-model-assisted inference pipeline. With extensive evaluations, we believe that SPECPREFILL will be one of the practical answers to large-scale LLM inference systems.

Impact Statement

This paper presents work whose goal is to accelerate large language model inference procedures. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q., and Sutton, C. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

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. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172.

Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer, 2020. URL https://arxiv.org/abs/2004.05150.

Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S., Nori, H., Palangi, H., Ribeiro, M. T., and Zhang, Y. Sparks of artificial general intelligence: Early experiments with gpt-4, 2023. URL https://arxiv.org/abs/2303.12712.

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457, 2018.

Cobbe, K., Kosaraju, V., Bavarian, M., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems, 2021.

Dao, T. Flashattention-2: Faster attention with better parallelism and work partitioning, 2023. URL https://arxiv.org/abs/2307.08691.

Dao, T., Fu, D. Y., Ermon, S., Rudra, A., and Ré, C. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022. URL https://arxiv.org/abs/2205.14135.

DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., Xue, B., Wang, B., Wu, B., Feng, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., Dai, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Bao, H., Xu, H., Wang, H., Ding, H., Xin, H., Gao, H., Qu, H., Li, H., Guo, J., Li, J., Wang, J., Chen, J., Yuan, J., Qiu, J., Li, J., Cai, J. L., Ni, J., Liang, J., Chen, J., Dong, K., Hu, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Zhao, L., Wang, L., Zhang, L., Xu, L., Xia, L., Zhang, M., Zhang, M., Tang, M., Li, M., Wang, M., Li, M., Tian, N., Huang, P., Zhang, P., Wang, Q., Chen, Q., Du, Q., Ge, R., Zhang, R., Pan, R., Wang, R., Chen, R. J., Jin, R. L., Chen, R., Lu, S., Zhou, S., Chen, S., Ye, S., Wang, S., Yu, S., Zhou, S., Pan, S., Li, S. S., Zhou, S., Wu, S., Ye, S., Yun, T., Pei, T., Sun, T., Wang, T., Zeng, W., Zhao, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Xiao, W. L., An, W., Liu, X., Wang, X., Chen, X., Nie, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yang, X., Li, X., Su, X., Lin, X., Li, X. Q., Jin, X., Shen, X., Chen, X., Sun, X., Wang, X., Song, X., Zhou, X., Wang, X., Shan, X., Li, Y. K., Wang, Y. Q., Wei, Y. X., Zhang, Y., Xu, Y., Li, Y., Zhao, Y., Sun, Y., Wang, Y., Yu, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Ou, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Xiong, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Zhu, Y. X., Xu, Y., Huang, Y., Li, Y., Zheng, Y., Zhu, Y., Ma, Y., Tang, Y., Zha, Y., Yan, Y., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Xie, Z., Zhang, Z., Hao, Z., Ma, Z., Yan, Z., Wu, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Pan, Z., Huang, Z., Xu, Z., Zhang, Z., and Zhang, Z. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Elhoushi, M., Shrivastava, A., Liskovich, D., Hosmer, B., Wasti, B., Lai, L., Mahmoud, A., Acun, B., Agarwal, S., Roman, A., Aly, A., Chen, B., and Wu, C.-J. Layerskip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12622–12642. Association for Computational Linguistics, 2024. doi: 10.18653/v1/

2024.acl-long.681. URL http://dx.doi.org/10.18653/v1/2024.acl-long.681.

Feng, T., Jin, C., Liu, J., Zhu, K., Tu, H., Cheng, Z., Lin, G., and You, J. How far are we from agi: Are llms all we need?, 2024. URL https://arxiv.org/abs/2405.10313.

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac'h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. A framework for few-shot language model evaluation, 07 2024a. URL https://zenodo.org/records/12608602.

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., and Wang, H. Retrieval-augmented generation for large language models: A survey, 2024b. URL https://arxiv.org/abs/2312.10997.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Roziere, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C. C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D., Livshits, D., Wyatt, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E., Lobanova, E., Dinan, E., Smith, E. M., Radenovic, F., Guzmán, F., Zhang, F., Synnaeve, G., Lee, G., Anderson, G. L., Thattai, G., Nail, G., Mialon, G., Pang, G., Cucurell, G., Nguyen, H., Korevaar, H., Xu, H., Touvron, H., Zarov, I., Ibarra, I. A., Kloumann, I., Misra, I., Evtimov, I., Zhang, J., Copet, J., Lee, J., Geffert, J., Vranes, J., Park, J., Mahadeokar, J., Shah, J., van der Linde, J., Billock, J., Hong, J., Lee, J., Fu, J., Chi, J., Huang, J., Liu, J., Wang, J., Yu, J., Bitton, J., Spisak, J., Park, J., Rocca, J., Johnstun, J., Saxe, J., Jia, J., Alwala, K. V., Prasad, K., Upasani, K., Plawiak, K., Li, K., Heafield, K., Stone, K., El-Arini, K., Iyer, K., Malik, K., Chiu, K., Bhalla, K., Lakhotia, K., Rantala-Yeary, L., van der Maaten, L., Chen, L., Tan, L., Jenkins, L., Martin, L., Madaan, L., Malo, L., Blecher, L., Landzaat, L., de Oliveira, L., Muzzi, M., Pasupuleti, M., Singh, M., Paluri, M., Kardas, M., Tsimpoukelli, M., Oldham, M., Rita, M., Pavlova, M., Kambadur, M., Lewis, M., Si, M., Singh, M. K., Hassan, M., Goyal, N., Torabi, N., Bashlykov, N., Bogoychev, N., Chatterji, N., Zhang, N., Duchenne, O., Çelebi, O., Alrassy, P., Zhang, P., Li, P., Vasic, P., Weng, P., Bhargava, P., Dubal, P., Krishnan, P., Koura, P. S., Xu, P., He, Q., Dong, Q., Srinivasan, R., Ganapathy, R., Calderer, R., Cabral, R. S., Stojnic, R., Raileanu, R., Maheswari, R., Girdhar, R., Patel, R., Sauvestre, R., Polidoro, R., Sumbaly, R., Taylor, R., Silva, R., Hou, R., Wang, R., Hosseini, S., Chennabasappa, S., Singh, S., Bell, S., Kim, S. S., Edunov, S., Nie, S., Narang, S., Raparthy, S., Shen, S., Wan, S., Bhosale, S., Zhang, S., Vandenhende, S., Batra, S., Whitman, S., Sootla, S., Collot, S., Gururangan, S., Borodinsky, S., Herman, T., Fowler, T., Sheasha, T., Georgiou, T., Scialom, T., Speckbacher, T., Mihaylov, T., Xiao, T., Karn, U., Goswami, V., Gupta, V., Ramanathan, V., Kerkez, V., Gonguet, V., Do, V., Vogeti, V., Albiero, V., Petrovic, V., Chu, W., Xiong, W., Fu, W., Meers, W., Martinet, X., Wang, X., Wang, X., Tan, X. E., Xia, X., Xie, X., Jia, X., Wang, X., Goldschlag, Y., Gaur, Y., Babaei, Y., Wen, Y., Song, Y., Zhang, Y., Li, Y., Mao, Y., Coudert, Z. D., Yan, Z., Chen, Z., Papakipos, Z., Singh, A., Srivastava, A., Jain, A., Kelsey, A., Shajnfeld, A., Gangidi, A., Victoria, A., Goldstand, A., Menon, A., Sharma, A., Boesenberg, A., Baevski, A., Feinstein, A., Kallet, A., Sangani, A., Teo, A., Yunus, A., Lupu, A., Alvarado, A., Caples, A., Gu, A., Ho, A., Poulton, A., Ryan, A., Ramchandani, A., Dong, A., Franco, A., Goyal, A., Saraf, A., Chowdhury, A., Gabriel, A., Bharambe, A., Eisenman, A., Yazdan, A., James, B., Maurer, B., Leonhardi, B., Huang, B., Loyd, B., Paola, B. D., Paranjape, B., Liu, B., Wu, B., Ni, B., Hancock, B., Wasti, B., Spence, B., Stojkovic, B., Gamido, B., Montalvo, B., Parker, C., Burton, C., Mejia, C., Liu, C., Wang, C., Kim, C., Zhou, C., Hu, C., Chu, C.-H., Cai, C., Tindal, C., Feichtenhofer, C., Gao, C., Civin, D., Beaty, D., Kreymer, D., Li, D., Adkins, D., Xu, D., Testuggine, D., David, D., Parikh, D., Liskovich, D., Foss, D., Wang, D., Le, D., Holland, D., Dowling, E., Jamil, E., Montgomery, E., Presani, E., Hahn, E., Wood, E., Le, E.-T., Brinkman, E., Arcaute, E., Dunbar, E., Smothers, E., Sun, F., Kreuk, F., Tian, F., Kokkinos, F., Ozgenel, F., Caggioni, F., Kanayet, F., Seide, F., Florez, G. M., Schwarz, G., Badeer, G., Swee, G., Halpern, G., Herman, G., Sizov, G., Guangyi, Zhang, Lakshminarayanan, G., Inan, H., Shojanazeri, H., Zou, H., Wang, H., Zha, H., Habeeb, H., Rudolph, H., Suk, H., Aspegren, H., Goldman, H., Zhan, H., Damlaj, I., Molybog, I., Tufanov, I., Leontiadis, I., Veliche, I.-E., Gat, I., Weissman, J., Geboski, J., Kohli, J., Lam, J., Asher, J., Gaya, J.-B., Marcus, J., Tang, J., Chan, J., Zhen, J., Reizenstein, J., Teboul, J., Zhong, J., Jin, J., Yang, J., Cummings, J., Carvill, J., Shepard, J., McPhie, J., Torres, J., Ginsburg, J., Wang, J., Wu, K., U, K. H., Saxena, K., Khandelwal, K., Zand, K., Matosich, K., Veeraraghavan, K., Michelena, K., Li, K., Jagadeesh, K., Huang, K., Chawla, K., Huang, K., Chen, L., Garg, L., A, L., Silva, L., Bell, L., Zhang, L., Guo, L., Yu, L., Moshkovich, L., Wehrstedt, L., Khabsa, M., Avalani, M., Bhatt, M., Mankus, M., Hasson, M., Lennie, M., Reso, M., Groshev, M., Naumov, M., Lathi, M., Keneally, M.,

Liu, M., Seltzer, M. L., Valko, M., Restrepo, M., Patel, M., Vyatskov, M., Samvelyan, M., Clark, M., Macey, M., Wang, M., Hermoso, M. J., Metanat, M., Rastegari, M., Bansal, M., Santhanam, N., Parks, N., White, N., Bawa, N., Singhal, N., Egebo, N., Usunier, N., Mehta, N., Laptev, N. P., Dong, N., Cheng, N., Chernoguz, O., Hart, O., Salpekar, O., Kalinli, O., Kent, P., Parekh, P., Saab, P., Balaji, P., Rittner, P., Bontrager, P., Roux, P., Dollar, P., Zvyagina, P., Ratanchandani, P., Yuvraj, P., Liang, Q., Alao, R., Rodriguez, R., Ayub, R., Murthy, R., Nayani, R., Mitra, R., Parthasarathy, R., Li, R., Hogan, R., Battey, R., Wang, R., Howes, R., Rinott, R., Mehta, S., Siby, S., Bondu, S. J., Datta, S., Chugh, S., Hunt, S., Dhillon, S., Sidorov, S., Pan, S., Mahajan, S., Verma, S., Yamamoto, S., Ramaswamy, S., Lindsay, S., Lindsay, S., Feng, S., Lin, S., Zha, S. C., Patil, S., Shankar, S., Zhang, S., Zhang, S., Wang, S., Agarwal, S., Sajuyigbe, S., Chintala, S., Max, S., Chen, S., Kehoe, S., Satterfield, S., Govindaprasad, S., Gupta, S., Deng, S., Cho, S., Virk, S., Subramanian, S., Choudhury, S., Goldman, S., Remez, T., Glaser, T., Best, T., Koehler, T., Robinson, T., Li, T., Zhang, T., Matthews, T., Chou, T., Shaked, T., Vontimitta, V., Ajayi, V., Montanez, V., Mohan, V., Kumar, V. S., Mangla, V., Ionescu, V., Poenaru, V., Mihailescu, V. T., Ivanov, V., Li, W., Wang, W., Jiang, W., Bouaziz, W., Constable, W., Tang, X., Wu, X., Wang, X., Wu, X., Gao, X., Kleinman, Y., Chen, Y., Hu, Y., Jia, Y., Qi, Y., Li, Y., Zhang, Y., Zhang, Y., Adi, Y., Nam, Y., Yu, Wang, Zhao, Y., Hao, Y., Qian, Y., Li, Y., He, Y., Rait, Z., DeVito, Z., Rosnbrick, Z., Wen, Z., Yang, Z., Zhao, Z., and Ma, Z. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

Horton, M., Cao, Q., Sun, C., Jin, Y., Mehta, S., Rastegari, M., and Nabi, M. Kv prediction for improved time to first token, 2024. URL https://arxiv.org/abs/2410.08391.

Hsieh, C.-P., Sun, S., Kriman, S., Acharya, S., Rekesh, D., Jia, F., Zhang, Y., and Ginsburg, B. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mistral 7b, 2023a. URL https://arxiv.org/abs/2310.06825.

Jiang, H., Wu, Q., Lin, C.-Y., Yang, Y., and Qiu, L. Llmlingua: Compressing prompts for accelerated inference of large language models, 2023b. URL https://arxiv.org/abs/2310.05736.

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. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention, 2024a. URL https://arxiv.org/abs/2407.02490.

Jiang, H., Wu, Q., Luo, X., Li, D., Lin, C.-Y., Yang, Y., and Qiu, L. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression, 2024b. URL https://arxiv.org/abs/2310.06839.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Lai, X., Lu, J., Luo, Y., Ma, Y., and Zhou, X. Flexprefill: A context-aware sparse attention mechanism for efficient long-sequence inference, 2025. URL https://arxiv.org/abs/2502.20766.

Lee, H., Park, G., Suh, J., and Hwang, S. J. Infinitehip: Extending language model context up to 3 million tokens on a single gpu, 2025. URL https://arxiv.org/abs/2502.08910.

Leviathan, Y., Kalman, M., and Matias, Y. Fast inference from transformers via speculative decoding, 2023. URL https://arxiv.org/abs/2211.17192.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., tau Yih, W., Rocktäschel, T., Riedel, S., and Kiela, D. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021. URL https://arxiv.org/abs/2005.11401.

Li, Y., Dong, B., Lin, C., and Guerin, F. Compressing context to enhance inference efficiency of large language models, 2023. URL https://arxiv.org/abs/2310.06201.

Li, Y., Jiang, H., Wu, Q., Luo, X., Ahn, S., Zhang, C., Abdi, A. H., Li, D., Gao, J., Yang, Y., and Qiu, L. Scbench: A kv cache-centric analysis of long-context methods, 2025. URL https://arxiv.org/abs/2412.10319.

Li, Z., Li, C., Zhang, M., Mei, Q., and Bendersky, M. Retrieval augmented generation or long-context llms? a comprehensive study and hybrid approach, 2024. URL https://arxiv.org/abs/2407.16833.

Lin, J., Tang, J., Tang, H., Yang, S., Chen, W.-M., Wang, W.-C., Xiao, G., Dang, X., Gan, C., and Han, S. Awq: Activation-aware weight quantization for llm compression and acceleration, 2024. URL https://arxiv.org/abs/2306.00978.

Liu, J., Xia, C. S., Wang, Y., and Zhang, L. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=1qvx610Cu7.

Liu, Y., Li, H., Cheng, Y., Ray, S., Huang, Y., Zhang, Q., Du, K., Yao, J., Lu, S., Ananthanarayanan, G., Maire, M., Hoffmann, H., Holtzman, A., and Jiang, J. Cachegen: Kv cache compression and streaming for fast large language model serving, 2024. URL https://arxiv.org/abs/2310.07240.

Loper, E. and Bird, S. Nltk: The natural language toolkit, 2002. URL https://arxiv.org/abs/cs/0205028.

Lv, J., Feng, Y., Xie, X., Jia, X., Peng, Q., and Xie, G. Critiprefill: A segment-wise criticality-based approach for prefilling acceleration in llms, 2024. URL https://arxiv.org/abs/2409.12490.

Miao, X., Oliaro, G., Zhang, Z., Cheng, X., Jin, H., Chen, T., and Jia, Z. Towards efficient generative large language model serving: A survey from algorithms to systems, 2023. URL https://arxiv.org/abs/2312.15234.

OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., Bello, I., Berdine, J., Bernadett-Shapiro, G., Berner, C., Bogdonoff, L., Boiko, O., Boyd, M., Brakman, A.-L., Brockman, G., Brooks, T., Brundage, M., Button, K., Cai, T., Campbell, R., Cann, A., Carey, B., Carlson, C., Carmichael, R., Chan, B., Chang, C., Chantzis, F., Chen, D., Chen, S., Chen, R., Chen, J., Chen, M., Chess, B., Cho, C., Chu, C., Chung, H. W., Cummings, D., Currier, J., Dai, Y., Decareaux, C., Degry, T., Deutsch, N., Deville, D., Dhar, A., Dohan, D., Dowling, S., Dunning, S., Ecoffet, A., Eleti, A., Eloundou, T., Farhi, D., Fedus, L., Felix, N., Fishman, S. P., Forte, J., Fulford, I., Gao, L., Georges, E., Gibson, C., Goel, V., Gogineni, T., Goh, G., Gontijo-Lopes, R., Gordon, J., Grafstein, M., Gray, S., Greene, R., Gross, J., Gu, S. S., Guo, Y., Hallacy, C., Han, J., Harris, J., He, Y., Heaton, M., Heidecke, J., Hesse, C., Hickey, A., Hickey, W., Hoeschele, P., Houghton, B., Hsu, K., Hu, S., Hu, X., Huizinga, J., Jain, S., Jain, S., Jang, J., Jiang, A., Jiang, R., Jin, H., Jin, D., Jomoto, S., Jonn, B., Jun, H., Kaftan, T., Łukasz Kaiser, Kamali, A., Kanitscheider, I., Keskar, N. S., Khan, T., Kilpatrick, L., Kim, J. W., Kim, C., Kim, Y., Kirchner, J. H., Kiros, J., Knight, M., Kokotajlo, D., Łukasz Kondraciuk, Kondrich, A., Konstantinidis, A., Kosic, K., Krueger, G., Kuo, V., Lampe, M., Lan, I., Lee, T., Leike, J., Leung, J., Levy, D., Li, C. M., Lim, R., Lin, M., Lin, S., Litwin, M., Lopez, T., Lowe, R., Lue, P., Makanju, A., Malfacini, K., Manning, S., Markov, T., Markovski, Y., Martin, B., Mayer, K., Mayne, A., McGrew, B., McKinney, S. M., McLeavey, C., McMillan, P., McNeil, J., Medina, D., Mehta, A., Menick, J., Metz, L., Mishchenko, A., Mishkin, P., Monaco, V., Morikawa, E., Mossing, D., Mu, T., Murati, M., Murk, O., Mély, D., Nair, A., Nakano, R., Nayak, R., Neelakantan, A., Ngo, R., Noh, H., Ouyang, L., O'Keefe, C., Pachocki, J., Paino, A., Palermo, J., Pantuliano, A., Parascandolo, G., Parish, J., Parparita, E., Passos, A., Pavlov, M., Peng, A., Perelman, A., de Avila Belbute Peres, F., Petrov, M., de Oliveira Pinto, H. P., Michael, Pokorny, Pokrass, M., Pong, V. H., Powell, T., Power, A., Power, B., Proehl, E., Puri, R., Radford, A., Rae, J., Ramesh, A., Raymond, C., Real, F., Rimbach, K., Ross, C., Rotsted, B., Roussez, H., Ryder, N., Saltarelli, M., Sanders, T., Santurkar, S., Sastry, G., Schmidt, H., Schnurr, D., Schulman, J., Selsam, D., Sheppard, K., Sherbakov, T., Shieh, J., Shoker, S., Shyam, P., Sidor, S., Sigler, E., Simens, M., Sitkin, J., Slama, K., Sohl, I., Sokolowsky, B., Song, Y., Staudacher, N., Such, F. P., Summers, N., Sutskever, I., Tang, J., Tezak, N., Thompson, M. B., Tillet, P., Tootoonchian, A., Tseng, E., Tuggle, P., Turley, N., Tworek, J., Uribe, J. F. C., Vallone, A., Vijayvergiya, A., Voss, C., Wainwright, C., Wang, J. J., Wang, A., Wang, B., Ward, J., Wei, J., Weinmann, C., Welihinda, A., Welinder, P., Weng, J., Weng, L., Wiethoff, M., Willner, D., Winter, C., Wolrich, S., Wong, H., Workman, L., Wu, S., Wu, J., Wu, M., Xiao, K., Xu, T., Yoo, S., Yu, K., Yuan, Q., Zaremba, W., Zellers, R., Zhang, C., Zhang, M., Zhao, S., Zheng, T., Zhuang, J., Zhuk, W., and Zoph, B. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Qiao, A., Yao, Z., Rajbhandari, S., and He, Y. Swiftkv: Fast prefill-optimized inference with knowledge-preserving model transformation, 2024. URL https://arxiv.org/abs/2410.03960.

Reimers, N. and Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL https://arxiv.org/abs/1908.10084.

Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y.,

Dirani, J., Michael, J., and Bowman, S. R. Gpqa: A graduate-level google-proof q&a benchmark, 2023.

Shi, Z., Ming, Y., Nguyen, X.-P., Liang, Y., and Joty, S. Discovering the gems in early layers: Accelerating long-context llms with 1000x input token reduction, 2024. URL https://arxiv.org/abs/2409.17422.

Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., and Catanzaro, B. Megatron-lm: Training multi-billion parameter language models using model parallelism, 2020. URL https://arxiv.org/abs/1909.08053.

Sun, H., Chang, L.-W., Bao, W., Zheng, S., Zheng, N., Liu, X., Dong, H., Chi, Y., and Chen, B. Shadowkv: Kv cache in shadows for high-throughput long-context llm inference, 2024. URL https://arxiv.org/abs/2410.21465.

Team, G., Anil, R., Borgeaud, S., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., Millican, K., Silver, D., Johnson, M., Antonoglou, I., Schrittwieser, J., Glaese, A., Chen, J., Pitler, E., Lillicrap, T., Lazaridou, A., Firat, O., Molloy, J., Isard, M., Barham, P. R., Hennigan, T., Lee, B., Viola, F., Reynolds, M., Xu, Y., Doherty, R., Collins, E., Meyer, C., Rutherford, E., Moreira, E., Ayoub, K., Goel, M., Krawczyk, J., Du, C., Chi, E., Cheng, H.-T., Ni, E., Shah, P., Kane, P., Chan, B., Faruqui, M., Severyn, A., Lin, H., Li, Y., Cheng, Y., Ittycheriah, A., Mahdieh, M., Chen, M., Sun, P., Tran, D., Bagri, S., Lakshminarayanan, B., Liu, J., Orban, A., Güra, F., Zhou, H., Song, X., Boffy, A., Ganapathy, H., Zheng, S., Choe, H., Ágoston Weisz, Zhu, T., Lu, Y., Gopal, S., Kahn, J., Kula, M., Pitman, J., Shah, R., Taropa, E., Merey, M. A., Baeuml, M., Chen, Z., Shafey, L. E., Zhang, Y., Sercinoglu, O., Tucker, G., Piqueras, E., Krikun, M., Barr, I., Savinov, N., Danihelka, I., Roelofs, B., White, A., Andreassen, A., von Glehn, T., Yagati, L., Kazemi, M., Gonzalez, L., Khalman, M., Sygnowski, J., Frechette, A., Smith, C., Culp, L., Proleev, L., Luan, Y., Chen, X., Lottes, J., Schucher, N., Lebron, F., Rrustemi, A., Clay, N., Crone, P., Kocisky, T., Zhao, J., Perz, B., Yu, D., Howard, H., Bloniarz, A., Rae, J. W., Lu, H., Sifre, L., Maggioni, M., Alcober, F., Garrette, D., Barnes, M., Thakoor, S., Austin, J., Barth-Maron, G., Wong, W., Joshi, R., Chaabouni, R., Fatiha, D., Ahuja, A., Tomar, G. S., Senter, E., Chadwick, M., Kornakov, I., Attaluri, N., Iturrate, I., Liu, R., Li, Y., Cogan, S., Chen, J., Jia, C., Gu, C., Zhang, Q., Grimstad, J., Hartman, A. J., Garcia, X., Pillai, T. S., Devlin, J., Laskin, M., de Las Casas, D., Valter, D., Tao, C., Blanco, L., Badia, A. P., Reitter, D., Chen, M., Brennan, J., Rivera, C., Brin, S., Iqbal, S., Surita, G., Labanowski, J., Rao, A., Winkler, S., Parisotto, E., Gu, Y., Olszewska, K., Addanki, R., Miech, A., Louis, A., Teplyashin, D., Brown, G., Catt, E., Balaguer, J., Xiang, J., Wang, P., Ashwood, Z., Briukhov, A., Webson, A., Ganapathy, S., Sanghavi, S., Kannan, A., Chang, M.-W., Stjerngren, A., Djolonga, J., Sun, Y., Bapna, A., Aitchison, M., Pejman, P., Michalewski, H., Yu, T., Wang, C., Love, J., Ahn, J., Bloxwich, D., Han, K., Humphreys, P., Sellam, T., Bradbury, J., Godbole, V., Samangooei, S., Damoc, B., Kaskasoli, A., Arnold, S. M. R., Vasudevan, V., Agrawal, S., Riesa, J., Lepikhin, D., Tanburn, R., Srinivasan, S., Lim, H., Hodkinson, S., Shyam, P., Ferret, J., Hand, S., Garg, A., Paine, T. L., Li, J., Li, Y., Giang, M., Neitz, A., Abbas, Z., York, S., Reid, M., Cole, E., Chowdhery, A., Das, D., Rogozińska, D., Nikolaev, V., Sprechmann, P., Nado, Z., Zilka, L., Prost, F., He, L., Monteiro, M., Mishra, G., Welty, C., Newlan, J., Jia, D., Allamanis, M., Hu, C. H., de Liedekerke, R., Gilmer, J., Saroufim, C., Rijhwani, S., Hou, S., Shrivastava, D., Baddepudi, A., Goldin, A., Ozturel, A., Cassirer, A., Xu, Y., Sohn, D., Sachan, D., Amplayo, R. K., Swanson, C., Petrova, D., Narayan, S., Guez, A., Brahma, S., Landon, J., Patel, M., Zhao, R., Villela, K., Wang, L., Jia, W., Rahtz, M., Giménez, M., Yeung, L., Keeling, J., Georgiev, P., Mincu, D., Wu, B., Haykal, S., Saputro, R., Vodrahalli, K., Qin, J., Cankara, Z., Sharma, A., Fernando, N., Hawkins, W., Neyshabur, B., Kim, S., Hutter, A., Agrawal, P., Castro-Ros, A., van den Driessche, G., Wang, T., Yang, F., yiin Chang, S., Komarek, P., McIlroy, R., Lučić, M., Zhang, G., Farhan, W., Sharman, M., Natsev, P., Michel, P., Bansal, Y., Qiao, S., Cao, K., Shakeri, S., Butterfield, C., Chung, J., Rubenstein, P. K., Agrawal, S., Mensch, A., Soparkar, K., Lenc, K., Chung, T., Pope, A., Maggiore, L., Kay, J., Jhakra, P., Wang, S., Maynez, J., Phuong, M., Tobin, T., Tacchetti, A., Trebacz, M., Robinson, K., Katariya, Y., Riedel, S., Bailey, P., Xiao, K., Ghelani, N., Aroyo, L., Slone, A., Houlsby, N., Xiong, X., Yang, Z., Gribovskaya, E., Adler, J., Wirth, M., Lee, L., Li, M., Kagohara, T., Pavagadhi, J., Bridgers, S., Bortsova, A., Ghemawat, S., Ahmed, Z., Liu, T., Powell, R., Bolina, V., Iinuma, M., Zablotskaia, P., Besley, J., Chung, D.-W., Dozat, T., Comanescu, R., Si, X., Greer, J., Su, G., Polacek, M., Kaufman, R. L., Tokumine, S., Hu, H., Buchatskaya, E., Miao, Y., Elhawaty, M., Siddhant, A., Tomasev, N., Xing, J., Greer, C., Miller, H., Ashraf, S., Roy, A., Zhang, Z., Ma, A., Filos, A., Besta, M., Blevins, R., Klimenko, T., Yeh, C.-K., Changpinyo, S., Mu, J., Chang, O., Pajarskas, M., Muir, C., Cohen, V., Lan, C. L., Haridasan, K., Marathe, A., Hansen, S., Douglas, S., Samuel, R., Wang, M., Austin, S., Lan, C., Jiang, J., Chiu, J., Lorenzo, J. A., Sjösund, L. L., Cevey, S., Gleicher, Z., Avrahami, T., Boral, A., Srinivasan, H., Selo, V., May, R., Aisopos, K., Hussenot, L., Soares, L. B., Baumli, K., Chang, M. B., Recasens, A., Caine, B., Pritzel, A., Pavetic, F., Pardo, F., Gergely, A., Frye, J., Ramasesh, V., Horgan, D., Badola, K., Kassner, N., Roy, S., Dyer, E., Campos, V. C., Tomala, A., Tang, Y., Badawy, D. E.,

White, E., Mustafa, B., Lang, O., Jindal, A., Vikram, S., Gong, Z., Caelles, S., Hemsley, R., Thornton, G., Feng, F., Stokowiec, W., Zheng, C., Thacker, P., Çağlar Ünlü, Zhang, Z., Saleh, M., Svensson, J., Bileschi, M., Patil, P., Anand, A., Ring, R., Tsihlas, K., Vezer, A., Selvi, M., Shevlane, T., Rodriguez, M., Kwiatkowski, T., Daruki, S., Rong, K., Dafoe, A., FitzGerald, N., Gu-Lemberg, K., Khan, M., Hendricks, L. A., Pellat, M., Feinberg, V., Cobon-Kerr, J., Sainath, T., Rauh, M., Hashemi, S. H., Ives, R., Hasson, Y., Noland, E., Cao, Y., Byrd, N., Hou, L., Wang, Q., Sottiaux, T., Paganini, M., Lespiau, J.-B., Moufarek, A., Hassan, S., Shivakumar, K., van Amersfoort, J., Mandhane, A., Joshi, P., Goyal, A., Tung, M., Brock, A., Sheahan, H., Misra, V., Li, C., Rakićević, N., Dehghani, M., Liu, F., Mittal, S., Oh, J., Noury, S., Sezener, E., Huot, F., Lamm, M., Cao, N. D., Chen, C., Mudgal, S., Stella, R., Brooks, K., Vasudevan, G., Liu, C., Chain, M., Melinkeri, N., Cohen, A., Wang, V., Seymore, K., Zubkov, S., Goel, R., Yue, S., Krishnakumaran, S., Albert, B., Hurley, N., Sano, M., Mohananey, A., Joughin, J., Filonov, E., Kepa, T., Eldawy, Y., Lim, J., Rishi, R., Badiezadegan, S., Bos, T., Chang, J., Jain, S., Padmanabhan, S. G. S., Puttagunta, S., Krishna, K., Baker, L., Kalb, N., Bedapudi, V., Kurzrok, A., Lei, S., Yu, A., Litvin, O., Zhou, X., Wu, Z., Sobell, S., Siciliano, A., Papir, A., Neale, R., Bragagnolo, J., Toor, T., Chen, T., Anklin, V., Wang, F., Feng, R., Gholami, M., Ling, K., Liu, L., Walter, J., Moghaddam, H., Kishore, A., Adamek, J., Mercado, T., Mallinson, J., Wandekar, S., Cagle, S., Ofek, E., Garrido, G., Lombriser, C., Mukha, M., Sun, B., Mohammad, H. R., Matak, J., Qian, Y., Peswani, V., Janus, P., Yuan, Q., Schelin, L., David, O., Garg, A., He, Y., Duzhyi, O., Älgmyr, A., Lottaz, T., Li, Q., Yadav, V., Xu, L., Chinien, A., Shivanna, R., Chuklin, A., Li, J., Spadine, C., Wolfe, T., Mohamed, K., Das, S., Dai, Z., He, K., von Dincklage, D., Upadhyay, S., Maurya, A., Chi, L., Krause, S., Salama, K., Rabinovitch, P. G., M, P. K. R., Selvan, A., Dektiarev, M., Ghiasi, G., Guven, E., Gupta, H., Liu, B., Sharma, D., Shtacher, I. H., Paul, S., Akerlund, O., Aubet, F.-X., Huang, T., Zhu, C., Zhu, E., Teixeira, E., Fritze, M., Bertolini, F., Marinescu, L.-E., Bölle, M., Paulus, D., Gupta, K., Latkar, T., Chang, M., Sanders, J., Wilson, R., Wu, X., Tan, Y.-X., Thiet, L. N., Doshi, T., Lall, S., Mishra, S., Chen, W., Luong, T., Benjamin, S., Lee, J., Andrejczuk, E., Rabiej, D., Ranjan, V., Styrc, K., Yin, P., Simon, J., Harriott, M. R., Bansal, M., Robsky, A., Bacon, G., Greene, D., Mirylenka, D., Zhou, C., Sarvana, O., Goyal, A., Andermatt, S., Siegler, P., Horn, B., Israel, A., Pongetti, F., Chen, C.-W. L., Selvatici, M., Silva, P., Wang, K., Tolins, J., Guu, K., Yogev, R., Cai, X., Agostini, A., Shah, M., Nguyen, H., Donnaile, N. O., Pereira, S., Friso, L., Stambler, A., Kurzrok, A., Kuang, C., Romanikhin, Y., Geller, M., Yan, Z., Jang, K., Lee, C.-C., Fica, W., Malmi, E., Tan, Q., Banica, D., Balle, D., Pham, R., Huang, Y., Avram, D., Shi, H., Singh, J., Hidey, C., Ahuja, N., Saxena, P., Dooley, D., Potharaju, S. P., O'Neill, E., Gokulchandran, A., Foley, R., Zhao, K., Dusenberry, M., Liu, Y., Mehta, P., Kotikalapudi, R., Safranek-Shrader, C., Goodman, A., Kessinger, J., Globen, E., Kolhar, P., Gorgolewski, C., Ibrahim, A., Song, Y., Eichenbaum, A., Brovelli, T., Potluri, S., Lahoti, P., Baetu, C., Ghorbani, A., Chen, C., Crawford, A., Pal, S., Sridhar, M., Gurita, P., Mujika, A., Petrovski, I., Cedoz, P.-L., Li, C., Chen, S., Santo, N. D., Goyal, S., Punjabi, J., Kappaganthu, K., Kwak, C., LV, P., Velury, S., Choudhury, H., Hall, J., Shah, P., Figueira, R., Thomas, M., Lu, M., Zhou, T., Kumar, C., Jurdi, T., Chikkerur, S., Ma, Y., Yu, A., Kwak, S., Ähdel, V., Rajayogam, S., Choma, T., Liu, F., Barua, A., Ji, C., Park, J. H., Hellendoorn, V., Bailey, A., Bilal, T., Zhou, H., Khatir, M., Sutton, C., Rzadkowski, W., Macintosh, F., Shagin, K., Medina, P., Liang, C., Zhou, J., Shah, P., Bi, Y., Dankovics, A., Banga, S., Lehmann, S., Bredesen, M., Lin, Z., Hoffmann, J. E., Lai, J., Chung, R., Yang, K., Balani, N., Bražinskas, A., Sozanschi, A., Hayes, M., Alcalde, H. F., Makarov, P., Chen, W., Stella, A., Snijders, L., Mandl, M., Kärrman, A., Nowak, P., Wu, X., Dyck, A., Vaidyanathan, K., R, R., Mallet, J., Rudominer, M., Johnston, E., Mittal, S., Udathu, A., Christensen, J., Verma, V., Irving, Z., Santucci, A., Elsayed, G., Davoodi, E., Georgiev, M., Tenney, I., Hua, N., Cideron, G., Leurent, E., Alnahlawi, M., Georgescu, I., Wei, N., Zheng, I., Scandinaro, D., Jiang, H., Snoek, J., Sundararajan, M., Wang, X., Ontiveros, Z., Karo, I., Cole, J., Rajashekhar, V., Tumeh, L., Ben-David, E., Jain, R., Uesato, J., Datta, R., Bunyan, O., Wu, S., Zhang, J., Stanczyk, P., Zhang, Y., Steiner, D., Naskar, S., Azzam, M., Johnson, M., Paszke, A., Chiu, C.-C., Elias, J. S., Mohiuddin, A., Muhammad, F., Miao, J., Lee, A., Vieillard, N., Park, J., Zhang, J., Stanway, J., Garmon, D., Karmarkar, A., Dong, Z., Lee, J., Kumar, A., Zhou, L., Evens, J., Isaac, W., Irving, G., Loper, E., Fink, M., Arkatkar, I., Chen, N., Shafran, I., Petrychenko, I., Chen, Z., Jia, J., Levskaya, A., Zhu, Z., Grabowski, P., Mao, Y., Magni, A., Yao, K., Snaider, J., Casagrande, N., Palmer, E., Suganthan, P., Castaño, A., Giannoumis, I., Kim, W., Rybiński, M., Sreevatsa, A., Prendki, J., Soergel, D., Goedeckemeyer, A., Gierke, W., Jafari, M., Gaba, M., Wiesner, J., Wright, D. G., Wei, Y., Vashisht, H., Kulizhskaya, Y., Hoover, J., Le, M., Li, L., Iwuanyanwu, C., Liu, L., Ramirez, K., Khorlin, A., Cui, A., LIN, T., Wu, M., Aguilar, R., Pallo, K., Chakladar, A., Perng, G., Abellan, E. A., Zhang, M., Dasgupta, I., Kushman, N., Penchev, I., Repina, A., Wu, X., van der Weide, T., Ponnapalli, P., Kaplan, C., Simsa, J., Li, S., Dousse, O., Yang, F., Piper, J., Ie, N., Pasumarthi, R., Lintz, N., Vijayakumar, A., Andor, D., Valenzuela, P., Lui, M., Paduraru, C., Peng, D., Lee, K., Zhang, S., Greene, S., Nguyen, D. D., Kurylowicz, P., Hardin, C., Dixon, L.,

Janzer, L., Choo, K., Feng, Z., Zhang, B., Singhal, A., Du, D., McKinnon, D., Antropova, N., Bolukbasi, T., Keller, O., Reid, D., Finchelstein, D., Raad, M. A., Crocker, R., Hawkins, P., Dadashi, R., Gaffney, C., Franko, K., Bulanova, A., Leblond, R., Chung, S., Askham, H., Cobo, L. C., Xu, K., Fischer, F., Xu, J., Sorokin, C., Alberti, C., Lin, C.-C., Evans, C., Dimitriev, A., Forbes, H., Banarse, D., Tung, Z., Omernick, M., Bishop, C., Sterneck, R., Jain, R., Xia, J., Amid, E., Piccinno, F., Wang, X., Banzal, P., Mankowitz, D. J., Polozov, A., Krakovna, V., Brown, S., Bateni, M., Duan, D., Firoiu, V., Thotakuri, M., Natan, T., Geist, M., tan Girgin, S., Li, H., Ye, J., Roval, O., Tojo, R., Kwong, M., Lee-Thorp, J., Yew, C., Sinopalnikov, D., Ramos, S., Mellor, J., Sharma, A., Wu, K., Miller, D., Sonnerat, N., Vnukov, D., Greig, R., Beattie, J., Caveness, E., Bai, L., Eisenschlos, J., Korchemniy, A., Tsai, T., Jasarevic, M., Kong, W., Dao, P., Zheng, Z., Liu, F., Yang, F., Zhu, R., Teh, T. H., Sanmiya, J., Gladchenko, E., Trdin, N., Toyama, D., Rosen, E., Tavakkol, S., Xue, L., Elkind, C., Woodman, O., Carpenter, J., Papamakarios, G., Kemp, R., Kafle, S., Grunina, T., Sinha, R., Talbert, A., Wu, D., Owusu-Afriyie, D., Du, C., Thornton, C., Pont-Tuset, J., Narayana, P., Li, J., Fatehi, S., Wieting, J., Ajmeri, O., Uria, B., Ko, Y., Knight, L., Héliou, A., Niu, N., Gu, S., Pang, C., Li, Y., Levine, N., Stolovich, A., Santamaria-Fernandez, R., Goenka, S., Yustalim, W., Strudel, R., Elqursh, A., Deck, C., Lee, H., Li, Z., Levin, K., Hoffmann, R., Holtmann-Rice, D., Bachem, O., Arora, S., Koh, C., Yeganeh, S. H., Põder, S., Tariq, M., Sun, Y., Ionita, L., Seyedhosseini, M., Tafti, P., Liu, Z., Gulati, A., Liu, J., Ye, X., Chrzaszcz, B., Wang, L., Sethi, N., Li, T., Brown, B., Singh, S., Fan, W., Parisi, A., Stanton, J., Koverkathu, V., Choquette-Choo, C. A., Li, Y., Lu, T., Ittycheriah, A., Shroff, P., Varadarajan, M., Bahargam, S., Willoughby, R., Gaddy, D., Desjardins, G., Cornero, M., Robenek, B., Mittal, B., Albrecht, B., Shenoy, A., Moiseev, F., Jacobsson, H., Ghaffarkhah, A., Rivière, M., Walton, A., Crepy, C., Parrish, A., Zhou, Z., Farabet, C., Radebaugh, C., Srinivasan, P., van der Salm, C., Fidjeland, A., Scellato, S., Latorre-Chimoto, E., Klimczak-Plucińska, H., Bridson, D., de Cesare, D., Hudson, T., Mendolicchio, P., Walker, L., Morris, A., Mauger, M., Guseynov, A., Reid, A., Odoom, S., Loher, L., Cotruta, V., Yenugula, M., Grewe, D., Petrushkina, A., Duerig, T., Sanchez, A., Yadlowsky, S., Shen, A., Globerson, A., Webb, L., Dua, S., Li, D., Bhupatiraju, S., Hurt, D., Qureshi, H., Agarwal, A., Shani, T., Eyal, M., Khare, A., Belle, S. R., Wang, L., Tekur, C., Kale, M. S., Wei, J., Sang, R., Saeta, B., Liechty, T., Sun, Y., Zhao, Y., Lee, S., Nayak, P., Fritz, D., Vuyyuru, M. R., Aslanides, J., Vyas, N., Wicke, M., Ma, X., Eltyshev, E., Martin, N., Cate, H., Manyika, J., Amiri, K., Kim, Y., Xiong, X., Kang, K., Luisier, F., Tripuraneni, N., Madras, D., Guo, M., Waters, A., Wang, O., Ainslie, J., Baldridge, J., Zhang, H., Pruthi, G., Bauer, J., Yang, F., Mansour, R., Gelman, J., Xu, Y., Polovets, G., Liu, J., Cai, H., Chen, W., Sheng, X., Xue, E., Ozair, S., Angermueller, C., Li, X., Sinha, A., Wang, W., Wiesinger, J., Koukoumidis, E., Tian, Y., Iyer, A., Gurumurthy, M., Goldenson, M., Shah, P., Blake, M., Yu, H., Urbanowicz, A., Palomaki, J., Fernando, C., Durden, K., Mehta, H., Momchev, N., Rahimtoroghi, E., Georgaki, M., Raul, A., Ruder, S., Redshaw, M., Lee, J., Zhou, D., Jalan, K., Li, D., Hechtman, B., Schuh, P., Nasr, M., Milan, K., Mikulik, V., Franco, J., Green, T., Nguyen, N., Kelley, J., Mahendru, A., Hu, A., Howland, J., Vargas, B., Hui, J., Bansal, K., Rao, V., Ghiya, R., Wang, E., Ye, K., Sarr, J. M., Preston, M. M., Elish, M., Li, S., Kaku, A., Gupta, J., Pasupat, I., Juan, D.-C., Someswar, M., M., T., Chen, X., Amini, A., Fabrikant, A., Chu, E., Dong, X., Muthal, A., Buthpitiya, S., Jauhari, S., Hua, N., Khandelwal, U., Hitron, A., Ren, J., Rinaldi, L., Drath, S., Dabush, A., Jiang, N.-J., Godhia, H., Sachs, U., Chen, A., Fan, Y., Taitelbaum, H., Noga, H., Dai, Z., Wang, J., Liang, C., Hamer, J., Ferng, C.-S., Elkind, C., Atias, A., Lee, P., Listík, V., Carlen, M., van de Kerkhof, J., Pikus, M., Zaher, K., Müller, P., Zykova, S., Stefanec, R., Gatsko, V., Hirnschall, C., Sethi, A., Xu, X. F., Ahuja, C., Tsai, B., Stefanoiu, A., Feng, B., Dhandhania, K., Katyal, M., Gupta, A., Parulekar, A., Pitta, D., Zhao, J., Bhatia, V., Bhavnani, Y., Alhadlaq, O., Li, X., Danenberg, P., Tu, D., Pine, A., Filippova, V., Ghosh, A., Limonchik, B., Urala, B., Lanka, C. K., Clive, D., Sun, Y., Li, E., Wu, H., Hongtongsak, K., Li, I., Thakkar, K., Omarov, K., Majmundar, K., Alverson, M., Kucharski, M., Patel, M., Jain, M., Zabelin, M., Pelagatti, P., Kohli, R., Kumar, S., Kim, J., Sankar, S., Shah, V., Ramachandruni, L., Zeng, X., Bariach, B., Weidinger, L., Vu, T., Andreev, A., He, A., Hui, K., Kashem, S., Subramanya, A., Hsiao, S., Hassabis, D., Kavukcuoglu, K., Sadovsky, A., Le, Q., Strohman, T., Wu, Y., Petrov, S., Dean, J., and Vinyals, O. Gemini: A family of highly capable multimodal models, 2024. URL https://arxiv.org/abs/2312.11805.

Wan, Z., Wu, Z., Liu, C., Huang, J., Zhu, Z., Jin, P., Wang, L., and Yuan, L. Look-m: Look-once optimization in kv cache for efficient multimodal long-context inference, 2024. URL https://arxiv.org/abs/2406.18139.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., and Fedus, W. Emergent abilities of large language models, 2022. URL https://arxiv.org/abs/2206.07682.

Xia, H., Yang, Z., Dong, Q., Wang, P., Li, Y., Ge, T., Liu, T., Li, W., and Sui, Z. Unlocking efficiency in large language

model inference: A comprehensive survey of speculative decoding, 2024. URL https://arxiv.org/abs/2401.07851.

Xiao, G., Tang, J., Zuo, J., Guo, J., Yang, S., Tang, H., Fu, Y., and Han, S. Duoattention: Efficient long-context llm inference with retrieval and streaming heads, 2024a. URL https://arxiv.org/abs/2410.10819.

Xiao, G., Tian, Y., Chen, B., Han, S., and Lewis, M. Efficient streaming language models with attention sinks, 2024b. URL https://arxiv.org/abs/2309.17453.

Xiong, W., Liu, J., Molybog, I., Zhang, H., Bhargava, P., Hou, R., Martin, L., Rungta, R., Sankararaman, K. A., Oguz, B., Khabsa, M., Fang, H., Mehdad, Y., Narang, S., Malik, K., Fan, A., Bhosale, S., Edunov, S., Lewis, M., Wang, S., and Ma, H. Effective long-context scaling of foundation models, 2023. URL https://arxiv.org/abs/2309.16039.

Yang, A., Yang, B., Hui, B., Zheng, B., Yu, B., Zhou, C., Li, C., Li, C., Liu, D., Huang, F., Dong, G., Wei, H., Lin, H., Tang, J., Wang, J., Yang, J., Tu, J., Zhang, J., Ma, J., Yang, J., Xu, J., Zhou, J., Bai, J., He, J., Lin, J., Dang, K., Lu, K., Chen, K., Yang, K., Li, M., Xue, M., Ni, N., Zhang, P., Wang, P., Peng, R., Men, R., Gao, R., Lin, R., Wang, S., Bai, S., Tan, S., Zhu, T., Li, T., Liu, T., Ge, W., Deng, X., Zhou, X., Ren, X., Zhang, X., Wei, X., Ren, X., Liu, X., Fan, Y., Yao, Y., Zhang, Y., Wan, Y., Chu, Y., Liu, Y., Cui, Z., Zhang, Z., Guo, Z., and Fan, Z. Qwen2 technical report, 2024. URL https://arxiv.org/abs/2407.10671.

Yuan, Z., Shang, Y., Zhou, Y., Dong, Z., Zhou, Z., Xue, C., Wu, B., Li, Z., Gu, Q., Lee, Y. J., Yan, Y., Chen, B., Sun, G., and Keutzer, K. Llm inference unveiled: Survey and roofline model insights, 2024. URL https://arxiv.org/abs/2402.16363.

Zhang, J., Wang, J., Li, H., Shou, L., Chen, K., Chen, G., and Mehrotra, S. Draft & verify: Lossless large language model acceleration via self-speculative decoding, 2024a. URL https://arxiv.org/abs/2309.08168.

Zhang, Z., Sheng, Y., Zhou, T., Chen, T., Zheng, L., Cai, R., Song, Z., Tian, Y., Ré, C., Barrett, C., Wang, Z., and Chen, B. H_2o: Heavy-hitter oracle for efficient generative inference of large language models, 2023. URL https://arxiv.org/abs/2306.14048.

Zhang, Z., Liu, S., Chen, R., Kailkhura, B., Chen, B., and Wang, A. Q-hitter: A better token oracle for efficient llm inference via sparse-quantized kv cache. In Gibbons, P., Pekhimenko, G., and Sa, C. D. (eds.), *Proceedings of Machine Learning and Systems*, volume 6, pp. 381–394, 2024b.

Zheng, L., Yin, L., Xie, Z., Sun, C., Huang, J., Yu, C. H., Cao, S., Kozyrakis, C., Stoica, I., Gonzalez, J. E., Barrett, C., and Sheng, Y. Sglang: Efficient execution of structured language model programs, 2024. URL https://arxiv.org/abs/2312.07104.

Zhou, J., Lu, T., Mishra, S., Brahma, S., Basu, S., Luan, Y., Zhou, D., and Hou, L. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

Zhu, Q., Duan, J., Chen, C., Liu, S., Li, X., Feng, G., Lv, X., Cao, H., Chuanfu, X., Zhang, X., Lin, D., and Yang, C. Sampleattention: Near-lossless acceleration of long context llm inference with adaptive structured sparse attention, 2024. URL https://arxiv.org/abs/2406.15486.

A. Standard Short Task Performance of SPECPREFILL

In Figure 6, we report SPECPREFILL when applied to `Llama-3.1-70B-Instruct` on standard short tasks as discussed in Sec 4.6. It is worth noting that for shorter tasks, the queries are more likely to become information dense, rendering SPECPREFILL less effective especially for certain tasks.

B. Comparing SPECPREFILL with RAG Based Systems

In this section, we first detail the algorithm behind two of our RAG baselines and present results comparing SPECPREFILL against them. Both variants split the context of the prompt into sentences using *nltk* library (Loper & Bird, 2002). After splitting the context, all sentences are encoded using pretrained sentence embedding models (Reimers & Gurevych, 2019). A specially chosen query is used to select relevant sentences based on similarity scores without exceeding the predefined budget. Finally, the new context is re-assembled and fed to the main model. We highlight the key differences in various steps of the pipeline in the following table 3:

Model Name	Embedding Model	RAG Query	Reassemble Method
RAG-LLAMA LS	`gtr-t5-large`	Last sentence in full prompt	Original order
RAG-LLAMA EQ	`all-mpnet-base-v2`	Provided by the dataset	Ordered by relevance

Table 3. **RAG Baseline Specification:** We implemented two RAG baselines with different trade-offs to compare SPECPREFILL.

In Figure 7, we compare `Llama-3.1-70B-Instruct` with RAG-LLAMA-70B LS and RAG-LLAMA-70B EQ. Since both RAG variants are based on sentence chunkation, and hence we calculate the final real token keep percentage for visualization and a fair comparison.

It is worth noting that both two RAG variants can in principle fall short under given tasks due to the fact that their strategy for selecting the query for retrieval is not flexible enough. For example, RAG-LLAMA-70B LS will become less effective when the real query is not placed at the end of the prompt, and RAG-LLAMA-70B EQ not only assumes that the real query is separated from the context and given to the model but also needs special catering when it is not obvious how to design the query for certain tasks (e.g. summarization). Therefore, we consider RAG systems and SPECPREFILL to be useful for different cases with varying degrees of requirements for efficiency, cost, performance, and generality.

C. Experiment Details

There are some details for our experiments that we wish to give some accounts for:

1. For standard short task evaluation in Sec 4.6, we use LM-EVAL-HARNESS (Gao et al., 2024a) and EVAL-PLUS (Liu et al., 2023). For several tasks in LM-EVAL-HARNESS, we include task configuration files for `Llama-3.1` based on its templates in our code base, which should be placed in the right place for reproducing experimental results.
2. In the QPS experiment in Sec 4.7, we add an extra 5 seconds to the timeout in order to avoid potential system instability. The final results are reported with the original timeout. We set the number of samples for each category based on the maximal QPS we want to evaluate, which makes sure we have a constant QPS during the duration of querying.
3. When running all experiments in vLLM (0.6.3.post1), we set enforce_eager=True and enable_chunked_prefill=False to avoid any unexpected behaviors.

D. System Specification

In Table 4, we list the detailed specification of the system on which we test the efficiency of models.

E. Overhead Analysis

SPECPREFILL uses a smaller model as a speculator to help accelerate the larger model. Although proven to be effective, there is no free lunch. In this section, we analyze and quantify the overhead incurred by the speculator so that practitioners are more informed when they choose to use it.

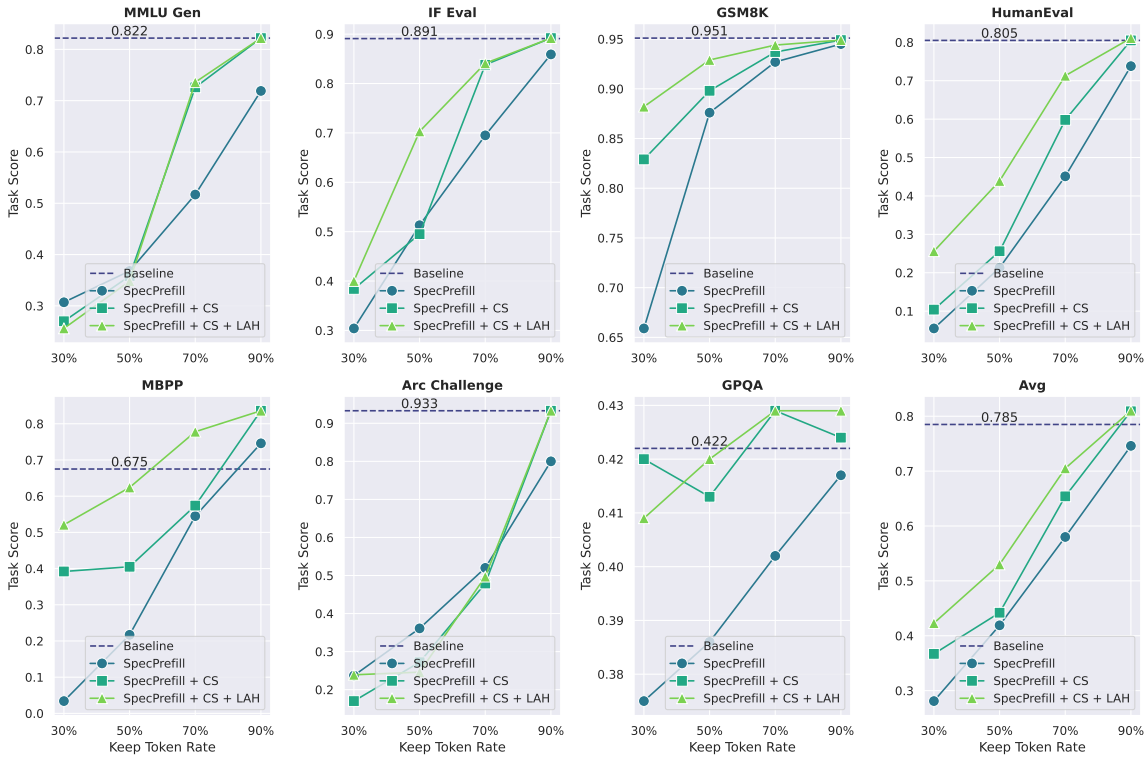

Figure 6. **Standard Short Tasks on Llama 70B:** We include results on popular regular context tasks ranging from common knowledge, math, reasoning, and coding ability. Unlike prior works on token eviction and prompt compression, we wish to give a comprehensive evaluation on domains where SPECPREFILL becomes less effective due to the fact that short and knowledge rich prompts are less compressible.

We start by calculating the FLOPS of a transformer model based on a standard implementation, and then compute the theoretical overhead when using a specific speculator for a specific main model with the official Llama model configuration. Since the prefill phase is mostly compute-bound, it is fair to use the FLOPS ratio as a decent approximation of latency improvement. We introduce several parameters for an Llama-like transformer architecture and we calculate the FLOPS of each module separately using these parameters. We ignore less essential computations (e.g. normalization, RoPE, etc) and the formula are shown in Table 5.

In Figure 9, we calculate the theoretical upper-bound of the TTFT speedup for SPECPREFILL with sequence length being $32K$ and batch size equal to 16. As we can see from the theoretical analysis, the real speedup we obtain from the implementation is very close to it, as shown in Figure 3, which suggests that our implementation is highly efficient (i.e. measured $7.66\times$ compared to analyzed $7.72\times$). On each bar, we annotate the overhead of the speculation process, which we define as:

$$overhead(\alpha) := \frac{FLOPS(spec)}{FLOPS(spec) + \alpha * FLOPS(base)}, \forall \alpha \in (0, 1]$$

Within our expectation, the overhead is higher when we have a lower keep rate and lower when we have a higher keep rate. Table E reports the theoretical relative FLOPS between the speculator and the base model.

F. LongBench Task Length Distribution

We visualize the LongBench suite's average length for each task, which, when coupled with the token keep rate, provides a more clear picture of the model's efficiency gain.

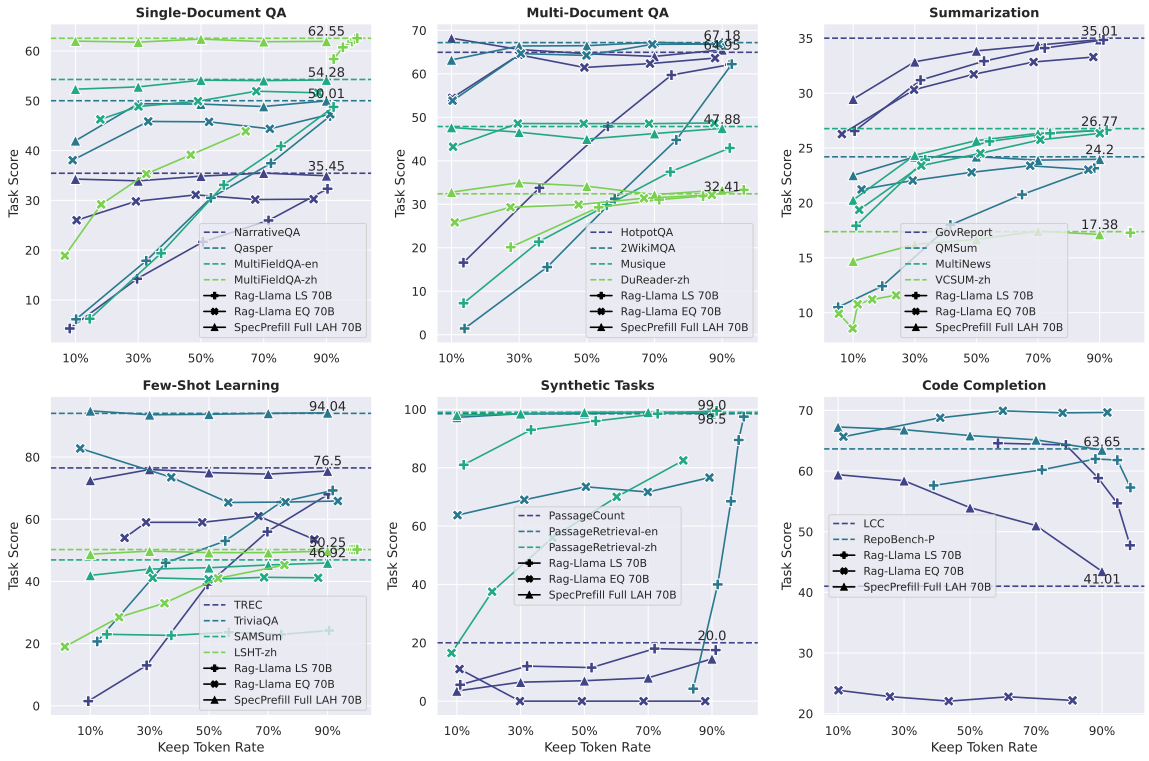

Figure 7. **LongBench Baseline Comparison:** We compare SPECPREFILL against the baseline and RAG-LLAMA in this separate figure for clarity. RAG-LLAMA could be effective on certain tasks but for the majority of the tasks, but SPECPREFILL does not require any prior knowledge about the prompt while still being accurate with a finer-control.

System Hardware/Software Name	Value
CUDA Version	12.7
vLLM Version	0.6.3.post1
GPU Type	8 × NVIDIA H200
Total GPU TFLOPS	428.2
Total RAM	1123.2 GB
Per GPU Memory Bandwidth	4052.8 GB/s
Per GPU NVLink Bandwidth	478.1 GB/s
Per GPU PCIe Bandwidth	52.8 GB/s
Per GPU PCIe Lanes	16 × PCIe 5.0
Disk Bandwidth	4730 MB/s
Internet Upload Speed	605.5 Mbps
Internet Download Speed	733.4 Mbps

Table 4. **System Specification for Efficiency Benchmarking:** Efficiency scores can vary when benchmarked on different platforms, and therefore, we list the detailed specification of the system we're using for better reproducibility and understanding.

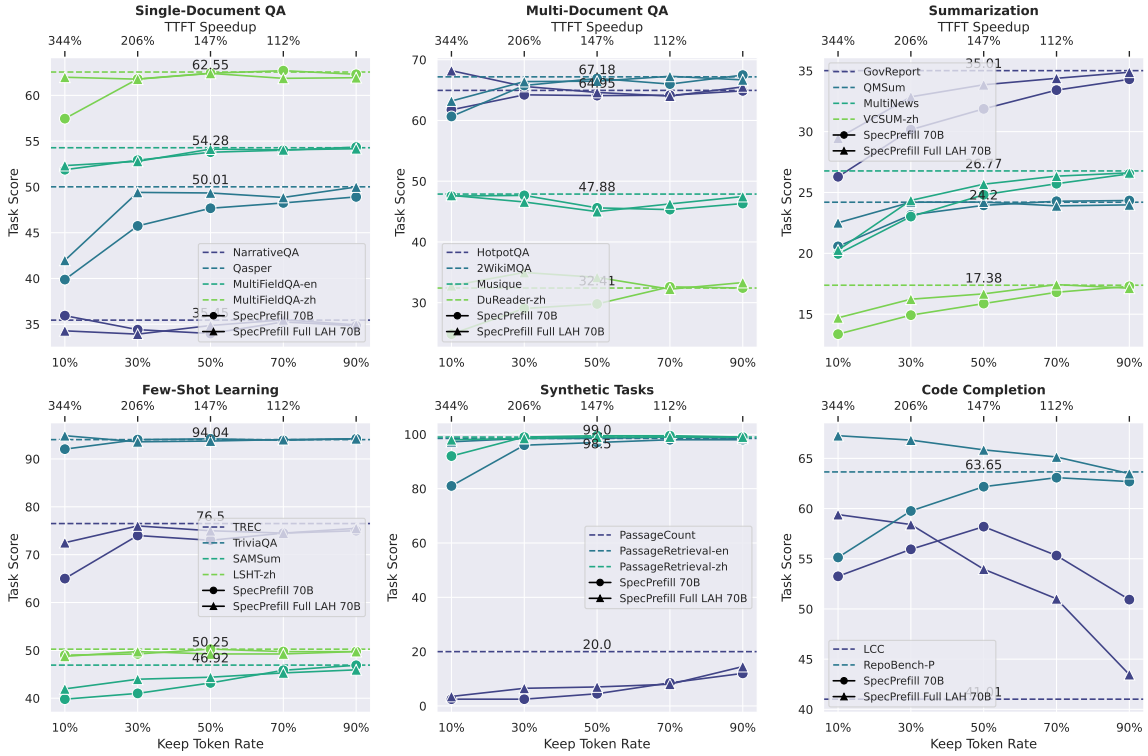

Figure 8. **LongBench Results on 70B Model:** We supplement the evaluation of `Llama-3.1-70B-Instruct`, under the same setup as in Figure 2 and 7.

Parameters	Value
Number of layers	L
Hidden Size	D
Intermediate Size	I
Number of Query Heads	H
Number of KV Heads	H'
Vocabulary Size	V
Sequence Length	S
Batch Size	B
MLP FLOPS	$3BSDI$
QKVO Projections FLOPS	$BSD^2(2+2H'/H)$
Self-Attention FLOPS	$2BS^2D$
LM-Head FLOPS	$BSDV$
Total FLOPS	$LBSD(3I+D(2+2H'/H)+2S)+BSDV$

Table 5. **FLOPS Estimation of Llama Models:** We estimate the FLOPS (MACS) for Llama model with a given configuration. We ignored lower order terms of computations such as vector addition, RMSNorm, RoPE, and treat all matrix operation as FMA for simplicity.

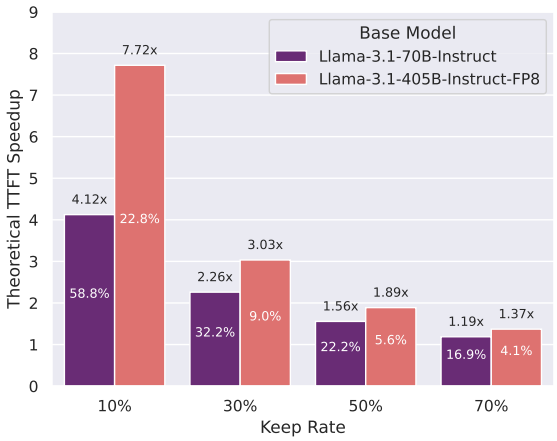

Figure 9. **Theoretical TTFT Speedup of SPECPREFILL:** We show the theoretical upper-bound of the TTFT speedup that SPECPREFILL can achieve assuming no other system overhead. The percentage of each bar is the percentage of FLOPS spent on speculation w.r.t. the total FLOPS. Comparing it with results in Figure 3, our real measurement has very little gap to the theoretical maximum, which suggests the efficiency of the implementation.

Base Model Size	$FLOPS_{spec}/FLOPS_{base}$
70B	14.24%
405B	2.96%

Table 6. **Relative FLOPS of SPECPREFILL:** We calculate the theoretical FLOP ratio between the speculator model of size 8B and the base model.

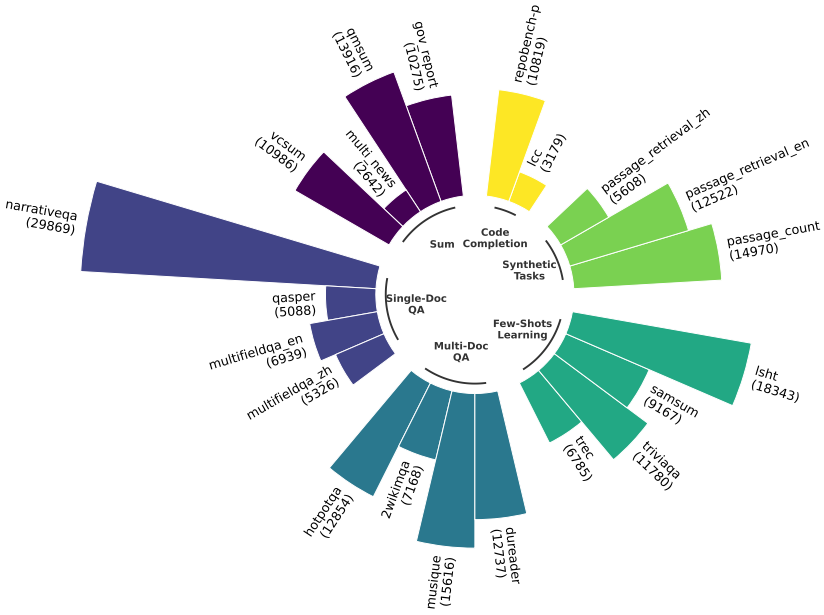

Figure 10. **LongBench Prompt Token Lengths:** We visualize the average token lengths of prompts for each task spanning the five major categories in LongBench.