Qwen2.5-1M Technical Report

An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoyan Huang, Jiandong Jiang, Jianhong Tu, Jianwei Zhang, Jingren Zhou, Junyang Lin, Kai Dang, Kexin Yang, Le Yu, Mei Li, Minmin Sun, Qin Zhu, Rui Men, Tao He, Weijia Xu, Wenbiao Yin, Wenyuan Yu, Xiafei Qiu, Xingzhang Ren, Xinlong Yang, Yong Li, Zhiying Xu, Zipeng Zhang*

Qwen Team, Alibaba Group

Abstract

In this report, we introduce Qwen2.5-1M, a series of models that extend the context length to 1 million tokens. Compared to the previous 128K version, the Qwen2.5-1M series have significantly enhanced long-context capabilities through long-context pre-training and post-training. Key techniques such as long data synthesis, progressive pre-training, and multi-stage supervised fine-tuning are employed to effectively enhance long-context performance while reducing training costs.

To promote the use of long-context models among a broader user base, we present and open-source our inference framework. This framework includes a length extrapolation method that can expand the model context lengths by at least four times, or even more, without additional training. To reduce inference costs, we implement a sparse attention method along with chunked prefill optimization for deployment scenarios and a sparsity refinement method to improve precision. Additionally, we detail our optimizations in the inference engine, including kernel optimization, pipeline parallelism, and scheduling optimization, which significantly enhance overall inference performance. By leveraging our inference framework, the Qwen2.5-1M models achieve a remarkable 3x to 7x prefill speedup in scenarios with 1 million tokens of context. This framework provides an efficient and powerful solution for developing applications that require long-context processing using open-source models.

The Qwen2.5-1M series currently includes the open-source models Qwen2.5-7B-Instruct-1M and Qwen2.5-14B-Instruct-1M, as well as the API-accessed model Qwen2.5-Turbo. Evaluations show that Qwen2.5-1M models have been greatly improved in long-context tasks without compromising performance in short-context scenarios. Specifically, the Qwen2.5-14B-Instruct-1M model significantly outperforms GPT-4o-mini in long-context tasks and supports contexts eight times longer.

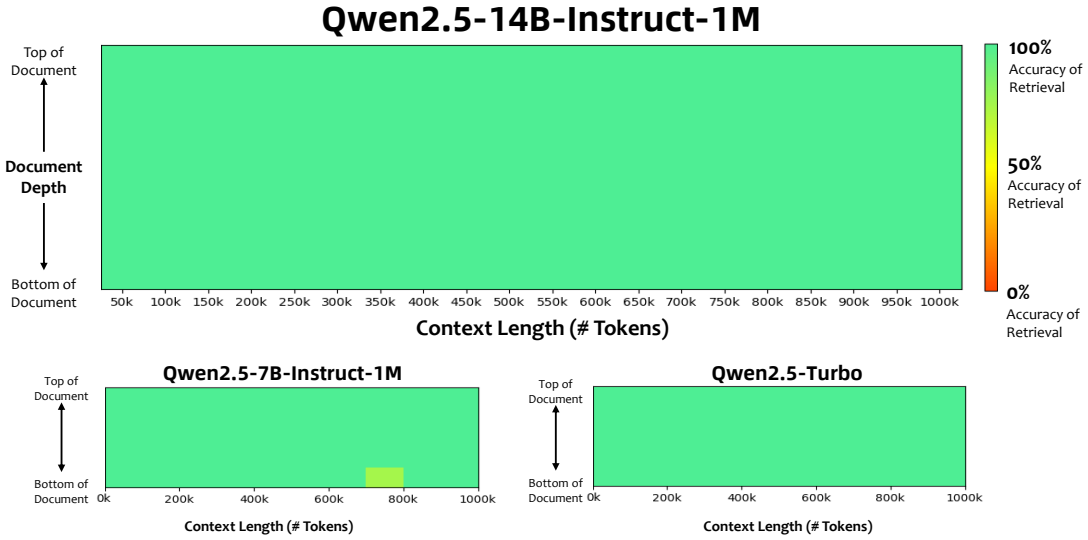


Figure 1: **Passkey Retrieval Test on Qwen2.5-1M Models with documents up to 1 Million Tokens.** This test evaluates the model's ability to retrieve a hidden number from ultra-long documents filled with irrelevant content. The results show that the Qwen2.5-1M models can accurately retrieve hidden numbers from documents containing up to 1M tokens, with only minor errors observed in the 7B model.

*Authors are ordered alphabetically by the last name.

1 Introduction

Large Language Models (LLMs) have revolutionized natural language processing by demonstrating remarkable capabilities in understanding, generating, and interacting with human language (Brown et al., 2020; OpenAI, 2023; 2024; Gemini Team, 2024; Anthropic, 2023a;b; 2024; Bai et al., 2023; Yang et al., 2024a; 2025; Touvron et al., 2023a;b; Dubey et al., 2024; Jiang et al., 2023a; 2024a). However, the limited context length restricts the amount of text that they can process at once, confining their capabilities to simpler, single tasks and preventing them from tackling complex real-world scenarios that require extensive information processing or generation. For example, LLMs struggle with performing code generation and debugging that rely on repository-level context or conducting in-depth research based on large volumes of documents.

To address this, increasing the context window of LLMs has become a significant trend. Models like GPT series (Brown et al., 2020; OpenAI, 2023; 2024), LLama series (Touvron et al., 2023a;b; Dubey et al., 2024), and our Qwen series (Bai et al., 2023; Yang et al., 2024a; Qwen Team, 2024a; Hui et al., 2024; Qwen Team, 2024c; Yang et al., 2024b) have rapidly expanded from initial context windows of 4k or 8k tokens to the current 128k tokens. There are also explorations to extend the context length of LLMs to 1M tokens or even longer, such as Gemini (Gemini Team, 2024), GLM-9B-Chat-1M (Zeng et al., 2024), and Llama-3-1M models from Gradient AI (Pekelis et al., 2024). This growth has enabled more sophisticated applications, allowing both users and developers to leverage these models' enhanced context capabilities for innovative research and development.

In this report, we will introduce the 1M context length version of Qwen2.5, namely the Qwen2.5-1M series. In terms of open-source weights, we release two instruction-tuned models: Qwen2.5-7B-Instruct-1M and Qwen2.5-14B-Instruct-1M. Compared to the 128K versions, these models exhibit significantly enhanced long-context capabilities. Additionally, we provide an API-accessible model based on Mixture of Experts (MoE), called Qwen2.5-Turbo, which offers performance comparable to GPT-4o-mini but with longer context, stronger capabilities, and more competitive pricing. Beyond the models themselves, we also open-source our inference framework optimized for long-context processing, enabling developers to deploy the Qwen2.5-1M models more cost-effectively.

This report outlines the key methodologies behind Qwen2.5-1M, focusing on two main aspects:

- **Efficient Long-Context Training.** The pre-training of Qwen2.5-1M incorporates synthetic data emphasizing long-range dependencies, with a progressive length extension strategy to reduce costs and enhance efficiency. Post-training addresses the scarcity of long-instruction datasets using agent-generated large-scale instruction data. A multi-stage Supervised Fine-Tuning (SFT) and Reinforcement Learning (RL) ensures balanced performance across short and long sequences, optimizing alignment with human preferences.
- **Efficient Inference and Deployment.** Our inference framework encompasses three key components: (1) a training-free length extrapolation method that allows models trained on 256k context lengths to seamlessly scale up to 1M contexts without requiring additional training; (2) a sparse attention mechanism aimed at reducing inference costs, with further optimizations to enhance GPU memory efficiency, integration with the length extrapolation method, and refined sparsity configurations to boost accuracy; and (3) engine-level optimizations such as kernel improvements, pipeline parallelism, and enhanced scheduling. By leveraging these advancements, our inference framework boost prefill speeds by 3 to 7 times in 1M-context scenarios.

2 Architecture

Qwen2.5-1M series are developed based on Qwen2.5 models (Yang et al., 2025) and support context length up to 1M tokens. It currently includes two dense models for opensource, namely Qwen2.5-7B-1M, Qwen2.5-14B-1M, and a MOE model for API service, namely Qwen2.5-Turbo.

The Qwen2.5-1M models retain the same Transformer-based architecture as Qwen2.5, ensuring compatibility in inference. Specifically, the architecture incorporates Grouped Query Attention (GQA, Ainslie et al., 2023) for efficient KV cache utilization, the SwiGLU activation function (Dauphin et al., 2017) for non-linear transformations, Rotary Positional Embeddings (RoPE, Su et al., 2024) to encode positional information, QKV bias (Su, 2023) in the attention mechanism, and RMSNorm (Jiang et al., 2023b) with pre-normalization to ensure stable training.

Table 1: **Model architecture and license of Qwen2.5-1M open-weight models.**

Models	Layers	Heads (Q / KV)	Tie Embedding	Context / Generation Length	License
7B	28	28 / 4	No	1M / 8K	Apache 2.0
14B	48	40 / 8	No	1M / 8K	Apache 2.0

3 Pre-training

Long-context pre-training is computationally intensive and can be expensive. To enhance training efficiency and reduce costs, we focus on optimizing data efficiency and refining training strategies during the pre-training process of Qwen2.5-1M models. Specifically, our improvements come from the following aspects:

Natural and Synthetic Data. During the pre-training phase, we assemble an extensive and diverse corpus of natural long-text data to ensure that our Qwen2.5-1M models are exposed to a wide array of linguistic patterns and contexts. This corpus encompasses various domains, including but not limited to Common Crawl, arXiv, books, and code repositories.

Despite the richness, natural corpus often exhibits weak long-distance associations, making it challenging for models to learn the connections between distant tokens effectively. This limitation arises because natural texts typically prioritize local coherence over global structure, where the model can effortlessly predict the next token without relying on long-range dependencies.

To address these challenges, we augmented the natural corpus with synthetic data designed to enhance the model's capacity to understand and generate long-range dependencies. The synthetic data generation process involved several sophisticated tasks aimed at improving the model's comprehension of sequential relationships and contextual understanding:

- **Fill in the Middle** (FIM, Bavarian et al., 2022): FIM tasks require the model to predict missing segments within a given text sequence. By inserting gaps at various positions and lengths, FIM encourages the model to focus on integrating distant contextual information surrounding the gap.
- **Keyword-Based and Position-Based Retrieval**: This task involves retrieving relevant paragraphs based on specific keywords or recalling paragraphs that appear before or after a specified position. This task enables the model to enhance its ability to identify and connect relevant information across different parts of a text while improving its understanding of positional relationships within sequences.
- **Paragraph Reordering**: In this task, paragraphs are shuffled, and the model must reorder them to restore the original sequence. This task strengthens the model's ability to recognize logical flows and structural coherence, essential for generating well-organized and coherent text.

By integrating these synthetic data tasks into the pre-training process, we significantly improved the model's ability to capture long-range information. This approach not only enhances data efficiency but also reduces the overall computational cost by accelerating the learning process and requiring fewer iterations to achieve high performance.

Training Strategy. Training with long contexts requires substantial GPU memory, thus posing a severe challenge to both training costs and time. To improve training efficiency, the Qwen2.5-1M models adopted a progressive context length expansion strategy, which includes five stages.

The first two stages are similar to those of other Qwen2.5 models, where we directly use an intermediate version from Qwen2.5 Base models for subsequent long-context training. Specifically, the model is initially trained with a context length of 4096 tokens, and then the training is transferred to a context length of 32768 tokens. During this process, we employ the Adaptive Base Frequency (ABF) technique (Xiong et al., 2023), adjusting the base frequency of the Rotary Position Embedding (RoPE, Su et al., 2024) from 10,000 to 1,000,000.

In the subsequent three stages, the context lengths are expanded to 65,536 tokens, 131,072 tokens, and 262,144 tokens, with the RoPE base frequencies set to 1,000,000, 5,000,000, and 10,000,000, respectively. During these stages, the training data is curated to include 75% sequences at the current maximum length and 25% shorter sequences. This approach ensures that the model can effectively adapt to longer contexts while preserving its capability to process and generalize across sequences of different lengths.

Table 2: **Performance of Qwen2.5-14B-1M on RULER at each pre-training stage.**

Training Length	RULER						
	Avg.	4K	8K	16K	32K	64K	128K
32,768 Tokens	82.3	96.8	94.7	95.9	92.2	76.4	37.6
65,536 Tokens	86.8	96.5	95.5	93.6	92.5	86.7	56.0
131,072 Tokens	92.5	96.5	95.9	93.0	92.6	93.0	83.8
262,144 Tokens	92.7	95.6	93.8	93.1	94.1	91.9	87.6

To monitor the performance changes of the progress training, we evaluate Qwen2.5-14B-1M on the RULER (Hsieh et al., 2024) benchmark at the end of each training phase. As illustrated in Table 2, training with progressively longer sequences consistently enhances the model's comprehension capabilities for the corresponding sequence lengths. Notably, even the final pre-training stage, which uses sequences of 262,144 tokens, significantly improves performance on the 128K samples. This finding aligns with previous research (An et al., 2024b), suggesting that models benefit significantly from training on longer sequences to fully realize their potential on relatively shorter tasks.

4 Post-Training

The aim of post-training is to effectively enhance the model's performance on long-context tasks while ensuring that performance on short tasks does not decline. We highlight the following efforts in building the Qwen2.5-1M models during post-training:

Synthesizing Long Instruction Data. In long-context tasks, human annotation can be expensive and unreliable. To address this issue, our training data includes a substantial portion of synthetic long-context question-answer pairs. Specifically, inspired by Dubey et al. (2024); Bai et al. (2024), we select long documents from the pre-training corpus and prompt Qwen2.5 to generate queries based on a randomly extracted segment of each document. These queries encompass a variety of tasks, including summarization, information retrieval, multi-hop question answering, reasoning, coding, and others. We then leverage the Qwen-Agent framework (Qwen Team, 2024b) to generate high-quality responses based on the full documents. This framework employs advanced techniques such as retrieval-augmented generation, chunk-by-chunk reading, and step-by-step reasoning, enabling it to integrate the overall content of the documents into its responses comprehensively. Finally, we utilize the full documents, the model-generated queries, and the agent-based generated responses to constitute synthetic training data.

Two-stage Supervised Fine-tuning. To enhance the model's performance on long-context tasks without compromising its performance on shorter tasks, we utilize a two-stage training scheme. In the first stage, similar to the Qwen2.5 models, we trained the model exclusively on short instruction data, each containing up to 32,768 tokens, and maintained the same number of training steps. This stage ensures that the model retained its proficiency in handling short tasks. In the second stage, we introduce a mixed dataset comprising both short and long sequences, with lengths ranging from up to 32,768 tokens to up to 262,144 tokens. We carefully balance the ratio of short to long data to prevent the model from forgetting the skills it has acquired during the first stage.

Reinforcement Learning. We employ offline reinforcement learning, similar to Direct Preference Optimization (DPO, Rafailov et al., 2023), to enhance the model's alignment with human preferences. Specifically, we utilize the training pairs from the offline RL phase of other Qwen2.5 models, which consisted solely of short samples up to 8,192 tokens. We find that training on these short samples is sufficient to significantly improve the model's alignment with human preferences and to generalize effectively to long-context tasks. To substantiate this claim, we evaluated the model before and after RL using the longbench-chat benchmark. As shown in Table 3, the RL stage lead to significant improvements across all models, demonstrating the effective generalization of RL from short-context to long-context tasks.

Table 3: **Performance on Longbench-Chat before and after RL stage.**

Model	Before RL	After RL
Qwen2.5-7B-Instruct-1M	7.32	8.08 (+0.75)
Qwen2.5-14B-Instruct-1M	8.56	8.76 (+0.20)
Qwen2.5-Turbo	7.60	8.34 (+0.74)

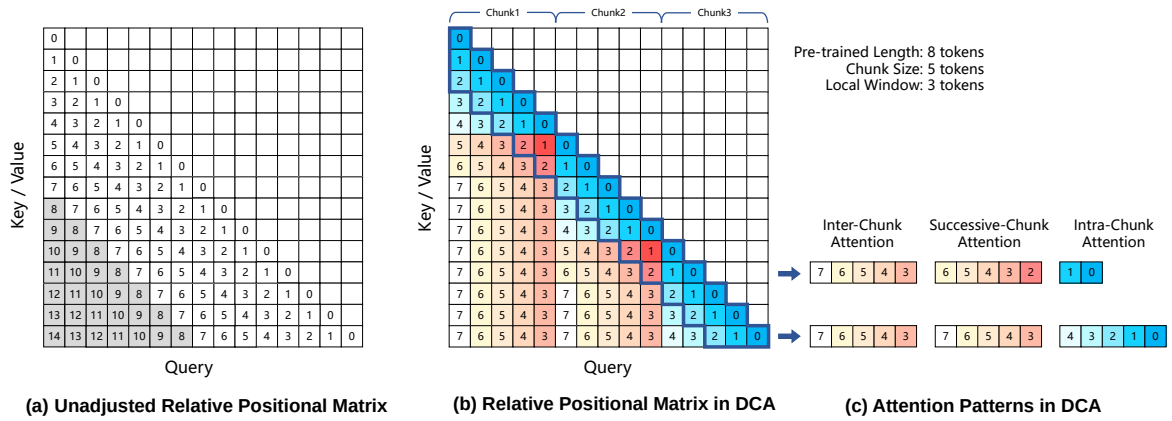


Figure 2: **An illustration of Dual Chunk Attention (DCA).** DCA remaps the relative positions to smaller numbers, thereby avoiding large relative positions that were not encountered during training (the gray areas of Figure (a)).

5 Inference and Deployment

Inference and deployment present significant challenges when LLMs process long-context tasks. Key issues include deploying models with longer sequences within the constraints of limited GPU memory, reducing computation to speed up processing, while maintaining accuracy during the optimization. In this section, we will introduce our approaches to addressing these challenges.

First, we present our length extrapolation method, which enables the model to support context lengths that are four times or even greater than the training length during inference. Next, we introduce a sparse attention mechanism that achieves more than a four-fold acceleration in the prefill stage. Finally, we delve into our optimizations at both the kernel and system levels, which further enhance overall inference performance.

Our inference and deployment solution detailed in this section has been open-sourced and integrated into vLLM (Kwon et al., 2023). It empowers users to deploy the Qwen-2.5 model on their own devices, leveraging our advanced length extrapolation methods and acceleration optimizations for enhanced performance.

5.1 Length Extrapolation

Length extrapolation is an inference technique designed to enhance model performance when processing long inputs that exceed the context length used during training. We employ the following two methods to achieve length extrapolation.

Dual Chunk Attention (DCA, An et al., 2024a). Modern LLMs based on RoPE experience performance degradation when processing sequences longer than the length in training, mainly due to encountering untrained, large relative positional distances between queries and keys in computing attention weights.

The DCA method addresses this issue by dividing the entire sequence into multiple chunks and remapping the relative positions into smaller numbers, ensuring that the distance between any two tokens does not exceed the pre-training length. An example of the remapped relative positional matrix is shown in Figure 2(b).

DCA employs three distinct attention patterns to efficiently manage token interactions at various distances:

- **Intra-Chunk Attention** handles the attention between tokens within the same chunk. Given that the distance between two tokens is relatively short, it preserves the original relative positions.
- **Inter-Chunk Attention** manages the attention between tokens that are not in the same chunk. To ensure that the maximum distance does not exceed the pre-training length, it uses a repeated sequences as relative positions among different chunks.
- **Successive-Chunk Attention** ensures the continuity of short-range relative positions by carefully managing the attention between two adjacent chunks. If the distance between a query and a key

is within the local window size, it retains the original relative positions. Otherwise, it adopts the approach used in Inter-Chunk Attention to handle longer distances.

By integrating these patterns, DCA enhances the model's capability to process context lengths that are four times longer or even more. Moreover, DCA can be seamlessly integrated with flash attention, and thus efficiently implemented in a production environment.

Attention Scaling in YaRN (Peng et al., 2023). When processing very long sequences, the attention mechanisms in LLMs can be distracted, leading to less focused on the key information. Peng et al. (2023) demonstrate that introducing a temperature parameter t to the attention logits can significantly enhance model performance in a simple yet effective manner. Specifically, the computation of attention weights are modified into

$$\text{softmax}\left(\frac{\mathbf{q}^{\mathrm{T}}\mathbf{k}}{t\sqrt{D}}\right), \quad \text{where} \quad \sqrt{\frac{1}{t}} = 0.1\ln(s) + 1. \tag{1}$$

$\mathbf{q}$ and $\mathbf{k}$ represent the query and key vectors, respectively. The scaling factor s is the ratio of the inference length to the training length, and D denotes the dimension of each attention head.

In the experiments in this report, we always use attention scaling in YaRN together with DCA. Note these two length extrapolation methods do not alter the model's behavior when processing short sequences, thus ensuring that performance on shorter tasks remains unaffected.

Effects of Length Extrapolation To demonstrate the effectiveness of the length extrapolation method, we evaluate the performance of the Qwen2.5-1M models and their 128k counterparts, both with and without DCA, under a context length of 1 million tokens. We select three tasks from RULER (Hsieh et al., 2024) for this evaluation: Passkey Retrieval, NIAH (Needle in a Haystack) with multiple queries, and NIAH with multiple values.

The results are shown in Figure 3. First, we find that DCA significantly enhances the performance of all instruction models when handling long-context tasks, particularly when the context length far exceeds the length in training. Second, for the relatively simple Passkey Retrieval task, DCA enable both the Qwen2.5-7B-Instruct and Qwen2.5-14B-Instruct models to achieve over 80% accuracy on sequences up to 1 million tokens, despite being trained only on sequences up to 32K tokens. This underscores the efficacy of DCA as a robust solution for length extrapolation. Finally, comparing the Qwen2.5-1M models with their 128k versions, we observed that training on longer sequences (up to 256k tokens) substantially improves the model's ability to extrapolate performance to even longer contexts.

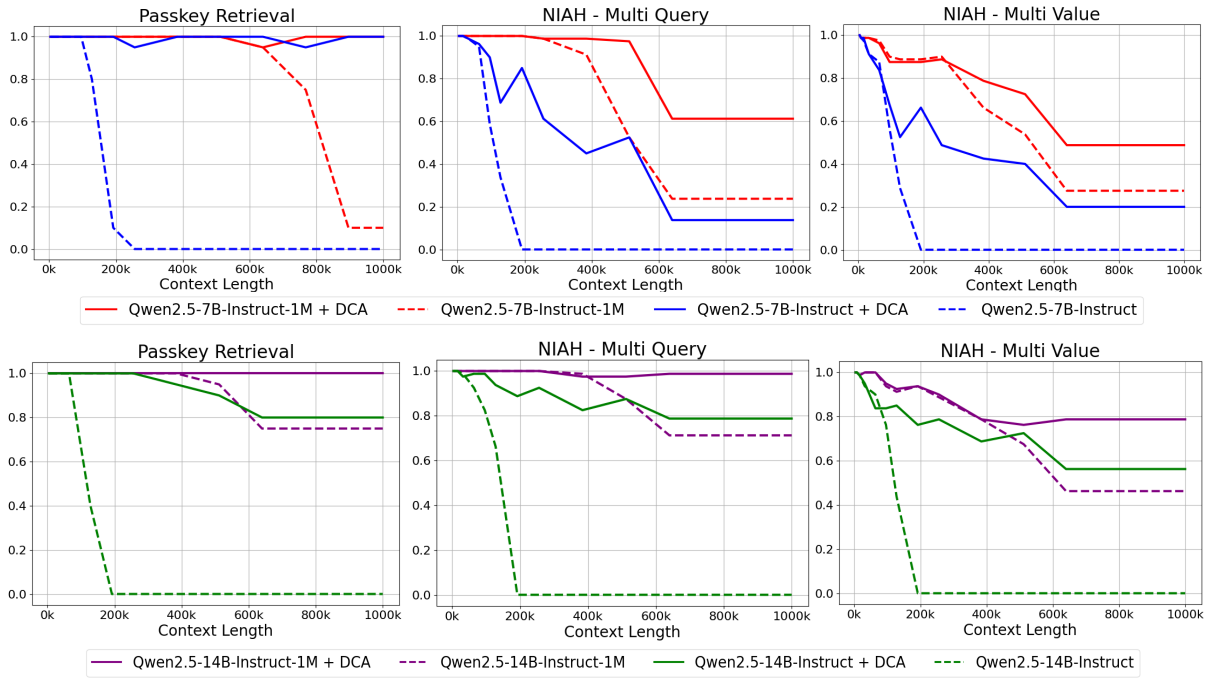
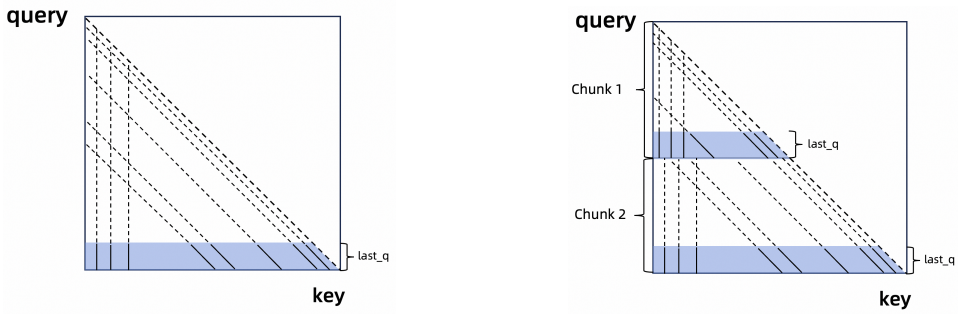


Figure 3: **The effects of length extrapolation on Long-Context Tasks.**



(a) Vertical-Slash pattern in MInference. (b) Combining MInference with chunked prefill.

Figure 4: **An illustration of MInference and our version integrated with chunked prefill.**

5.2 Efficient Inference with Sparse Attention

For long-context LLMs, inference speed is critical to user experience. The computational complexity of conventional attention mechanisms scales quadratically with the length of the input sequence. When the input length reaches one million tokens, the time spent on the attention mechanism can account for over 90% of the total forward pass time. Therefore, introducing sparse attention mechanisms is an essential step for the successful deployment of long-context models.

Specifically, we implemente a sparse attention mechanism based on MInference (Jiang et al., 2024b) to accelerate the prefill phase. Building on this foundation, we further optimize memory usage by integrating chunked prefill, combine these improvements with length extrapolation techniques, and introduce a sparse refinement method to address potential accuracy degradation in long sequences.

MInference (Jiang et al., 2024b). Attention computations in large language models (LLMs) exhibit sparsity for long context inputs. Jiang et al. (2024b) successfully identify and utilize only the critical tokens for attention computation, achieving results that are nearly identical to those obtained using full attention mechanisms. These critical tokens exhibit a distinct pattern across all samples, appearing as vertical and diagonal lines in the attention map. This pattern, referred to as the "Vertical-Slash" pattern, is illustrated in Figure 4(a).

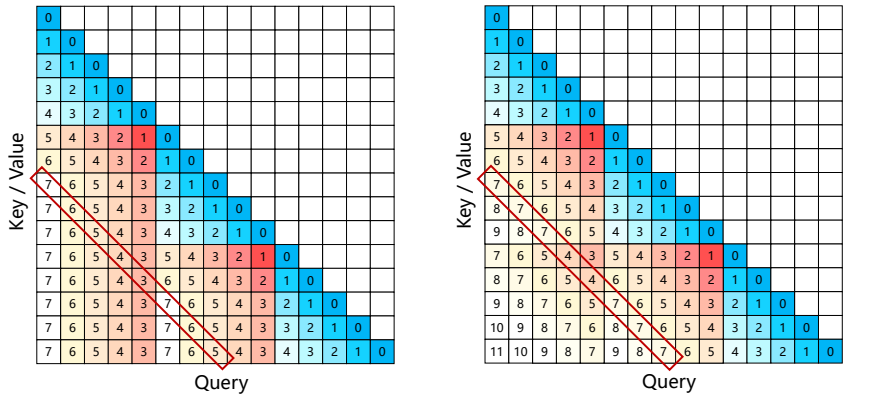
To leverage this sparsity, MInference first conducts an offline search to determine an optimal sparsification configuration. This configuration specifies how many vertical and diagonal lines each attention head should adopt. During inference, MInference initially computes the attention between the last query tokens (i.e., last_q) and all key tokens. Based on the partial attention results, it dynamically selects critical tokens following the "Vertical-Slash" pattern based on the pre-determined configuration, and finally computes attention only on these selected critical tokens. This approach significantly reduces computational and memory access costs by approximately 10 times while introducing only minimal accuracy loss.

Integrating with Chunked prefill. In MInference, the entire sequence is encoded simultaneously, leading to VRAM consumption by activation values that scales linearly with the input length. For instance, when the input reaches 1 million tokens, the VRAM consumption of activation values in a single MLP layer of Qwen2.5-7B can soar to 71GB, significantly exceeding the memory usage of model weights and key-value caches.

To address this challenge, chunked prefill can be employed during inference to reduce VRAM consumption. By using a chunk length of 32,768 tokens, chunked prefill can decrease activation VRAM usage by 96.7%. Additionally, when handling multiple requests, chunked prefill helps prevent decoding from being bottlenecked by lengthy prefill operations.

To integrate chunked prefill into MInference, we propose a strategy that selects critical tokens for each chunk, as illustrated in Figure 4(b). The input sequence is divided into multiple chunks, which are processed sequentially by the model. In the attention layer, rather than considering the last tokens of the entire input sequence that are not yet accessible, we leverage the last 64 tokens within each chunk to identify the critical tokens. This approach introduces distinct vertical and diagonal lines for each chunk during the token selection process, without significant loss in accuracy during our pilot experiments.

By integrating chunked prefill with MInference, our method significantly increases the maximum supported sequence length within limited VRAM resources.



(a) Non-continuous Relative Positional Matrix in DCA

(b) Continuous Relative Positional Matrix used in Selecting Critical Tokens

Figure 5: **Comparison of Relative Positional Matrices in DCA and in Selecting Critical Tokens.** The relative positions along the diagonal lines in (b) are more consistent than those in (a). For example, the diagonal line marked in red boxes contains $[5,6,7,3,4,5,6,7]$ in (a) and $[7,7,7,4,4,7,7,7]$ in (b).

Integrating with DCA. MInference can seamlessly integrate DCA into its implementation. However, we observe a performance drop in certain cases involving length extrapolation.

We hypothesize that the non-continuity of relative positions in DCA may disrupt the "slash" pattern, leading to decreased accuracy in selecting critical tokens. To address this issue, we propose to recover continuous relative positions when selecting critical tokens for both successive and inter-chunk attentions, ensuring that the relative positions along the diagonal lines are as consistent as possible, as illustrated in Figure 5. It is important to note that continuous relative positions are only introduced during the critical token selection phase, and the final computation of attention weights still uses the non-continuous position embeddings in DCA.

Sparsity refinement on 1M sequences. Before deployment, MInference requires an offline search to determine the optimal sparsification configuration for each attention head. This search process is conducted on short sequences due to the computational demand of full attention matrices, which scale quadratically with sequence length. Given the VRAM limitations of the devices used, the sequences in this search are typically kept below 32k tokens, leading to suboptimal performance on longer sequences, such as those with 1M tokens.

To address this limitation, we developed a method to refine the sparsification configuration specifically for sequences up to 1M tokens. Our approach leverages the efficient implementation of Flash Attention (Dao et al., 2022) to obtain *softmax_lse*, which is defined as:

$$\text{softmax\_lse}_{\text{full}} = \log \sum_{0 \le j \le i} \exp\left(\frac{\mathbf{q}^T k_j}{\sqrt{D}}\right). \tag{2}$$

The above *softmax_lse* represents the sum of unnormalized attention weights for the query $\mathbf{q}$. Similarly, we define the *softmax_lse* for sparse attention as:

$$\text{softmax\_lse}_{\text{sparse}} = \log \sum_{j \in \text{Critical}} \exp\left(\frac{\mathbf{q}^T k_j}{\sqrt{D}}\right), \tag{3}$$

indicating the sum of unnormalized attention weights for the query q_i only on critical tokens. Consequently, we can calculate the recall of attention weights as:

$$\text{Attention\_Recall} := \exp(\text{softmax\_lse}_{\text{sparse}} - \text{softmax\_lse}_{\text{full}}), \tag{4}$$

which is between 0 and 1, indicating how well the critical tokens are captured in the sparse attention computation.

Using this attention recall metric, we refine the sparsification configuration on a calibration set consisting of 1M-token sequences. The refinement process is detailed in Algorithm 1.

Algorithm 1 Sparsity Refinement

procedure
  for $l \leftarrow 1$ **to** *num_layers* **do**
    for $h \leftarrow 1$ **to** *num_heads* **do**
      $\mathbf{q} \leftarrow$ Queries of Layer l and Head h
      $\mathbf{k} \leftarrow$ Keys of Layer l and Head h
      $\mathbf{v} \leftarrow$ Values of Layer l and Head h
      $\mathbf{o}_{\text{full}}$, $\text{softmax\_lse}_{\text{full}} \leftarrow \text{full\_attention}(\mathbf{q}, \mathbf{k}, \mathbf{v})$

      $c \leftarrow$ Sparsity configs on Layer l and Head h
      $\mathbf{o}_{\text{sparse}}$, $\text{softmax\_lse}_{\text{sparse}} \leftarrow \text{sparse\_attention}(\mathbf{q}, \mathbf{k}, \mathbf{v}, c)$

      if $\exp(\text{softmax\_lse}_{\text{sparse}} - \text{softmax\_lse}_{\text{full}}) < \textit{Threshold}$ **then**
        Add Vertical & Slash Budgets to the config c.
      end if
    end for
  end for
end procedure

Impact of Sparse Attention on Accuracy To demonstrate the necessity of our method of integrating DCA and sparsity refinement, we evaluate Qwen2.5-7B-Instruct-1M on the Needle in a Haystack Test (Kamradt, 2023) with context lengths up to 1 million tokens. We choose this model because smaller models exhibit lower tolerance for information losses due to sparse attention, thereby better highlighting the value of our improvements.

As illustrated in Figure 6, Qwen2.5-7B-Instruct-1M with full attention successfully retrieves the majority of needles even in contexts of 1 million tokens. However, using the original MInference method results in a significant performance drop. For context lengths exceeding 400k tokens, the model's retrieval accuracy can fall to 60% or lower.

After incorporating continuous relative positions to select critical tokens and refining the sparsification configuration, as shown in Figure 6(c), the model recovers most of the performance and maintains about 4 times speedup during the prefilling stage.

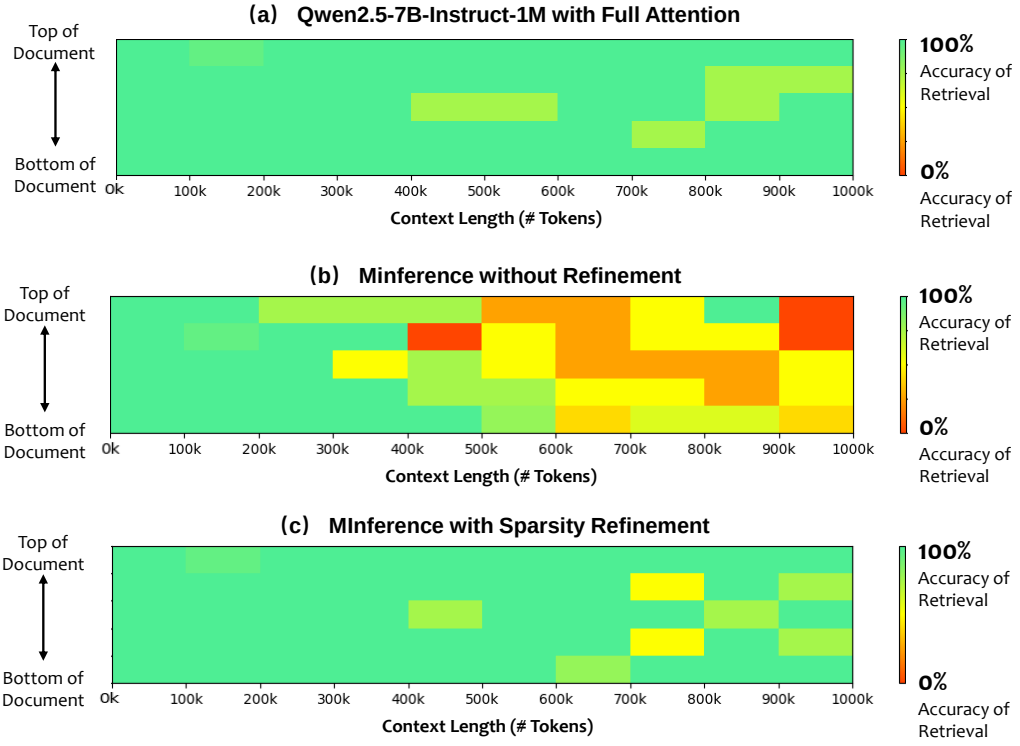


Figure 6: **Evaluate Qwen2.5-7B-Instruct-1M on Needle in A Haystack with Different Sparsification Configurations.**

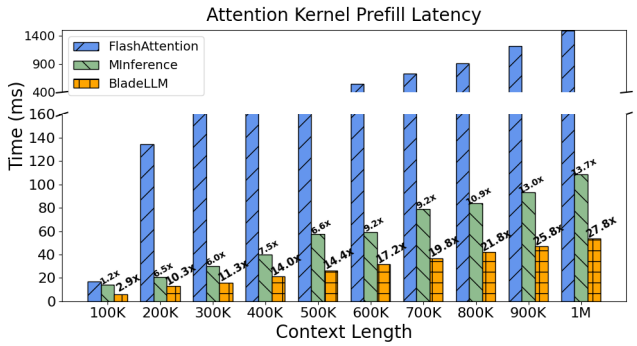


Figure 7: **Performance of Sparse Attention Kernels.**

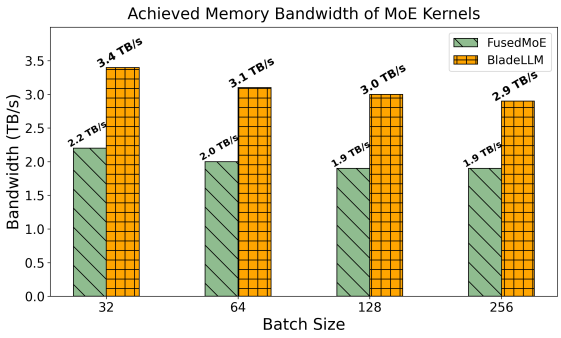


Figure 8: **Performance of MoE Kernels.**

5.3 Inference Engine

In addition to algorithmic advancements, optimizing the inference engine is essential for enabling LLMs to process long sequence effectively. The API services for Qwen2.5-1M Models are powered by BladeLLM, a high-performance inference engine developed by the Alibaba PAI Engine team. BladeLLM has been specifically optimized for long-sequence prefill and decoding through enhancements in kernel performance, pipeline parallelism, and scheduling algorithms. To assist the open-source community in efficiently deploying the Qwen models with extended context lengths, several of these optimizations have been open-sourced and will be integrated into vLLM (Kwon et al., 2023).

5.3.1 Kernel Optimization

Sparse Attention Kernel Optimization As the context length (L_c) increases, the computational complexity of attention ($O(L_c^2)$) and memory access ($O(L_c)$) grow significantly. To tackle this issue, the industry has explored optimization strategies to enhance efficiency through sparsity, such as the Vertical-Slash method by MInference (Jiang et al., 2024b). However, we observe that the efficiency of the attention kernel after sparsification remains low, still resulting in a significant proportion of the total inference time in end-to-end applications. Therefore, extreme optimization of the sparse attention kernel is even more crucial to fully unleashing the potential of sparsification.

To mitigate the overhead associated with sparse memory access, we implemente multi-stage pipeline parallelism and performed intensive instruction-level optimization when loading sparse KV pairs from global memory. Our optimized sparse attention kernel is engineered to leverage the capabilities of various GPU architectures, including NVIDIA's Ampere and Hopper series, AMD's MI300 series, and other hardware platforms.

Our experiments demonstrate that the optimized kernel in BladeLLM achieves remarkable high computational efficiency across multiple hardware platforms, with a peak FLOPs utilization rate of up to 90%. As shown in Figure 7, on the A100 GPU, under a 1 million token context, MInference exhibits 13.7x speedup compared to FlashAttention, while BladeLLM achieves 27.8x speedup under the same sparsity configuration.

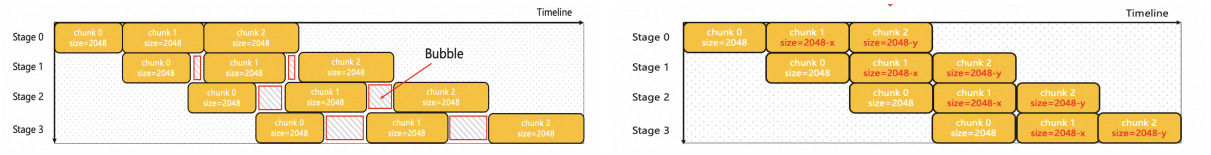
MoE Kernel Optimization Mixture-of-Experts (MoE) models, characterized by their sparse activation of parameters and outstanding accuracy, are particularly well-suited for large-scale deployment. In optimizing the performance of our MoE model, Qwen2.5-Turbo, we have identified that decoding performance is significantly influenced by memory access speed. Specifically, during the decoding phase, when handling batch sizes of 32 or greater, the access to large model parameters in each decoding iteration becomes a critical bottleneck for the overall efficiency of MOE layers. Therefore, enhancing memory access efficiency within the MoE kernel is crucial to achieving peak decoding performance.

BladeLLM enhances the efficiency of MoE kernels through a variety of optimization techniques, including improved Tensor Core utilization specifically tailored for memory-bound scenarios and fine-grained warp specialization. On the H20 GPU, these optimizations achieve peak memory access efficiency of 3.4 TB/s, representing a 55% improvement over the FusedMoE kernels in vLLM. Performance results across different batch sizes are illustrated in Figure 8.

5.3.2 Dynamic Chunked Pipeline Parallelism

Pipeline parallelism is a technique that divides the model into multiple segments, allowing different parts of the model to be processed concurrently. This approach significantly reduces communication volume compared to tensor parallelism, and its kernel execution efficiency is enhanced due to more intensive operations. In recent industry practices, segmented pipeline parallelism (Chunked Pipeline Parallelism) has been employed to accelerate the prefilling phase. Nevertheless, in scenarios involving extensive context lengths, we have identified an issue: variations in history length (i.e., the length of the past KV Cache) across different chunks lead to substantial disparities in attention computation time. This discrepancy results in numerous pipeline bubbles, as depicted in Figure 9(a).

BladeLLM employs Dynamic Chunked Pipeline Parallelism (DCPP) for long-context prefilling, dynamically adjusting the chunk size based on the computation complexity of the attention kernel to ensure that the execution time of each chunk is as equal as possible, thereby minimizing pipeline bubbles, as shown in Figure 9(b).



(a) Pipeline Bubbles in Chunked Prefilling

(b) Dynamic Chunked Pipeline Parallelism

Figure 9: **Optimization of Pipeline Parallelism in BladeLLM.**

5.3.3 Scheduling

A typical LLM inference engine can be divided into the following components:

- **API Server**: Responsible for receiving requests and sending responses.
- **Scheduler**: Responsible for request scheduling, KV Cache Block allocation, etc.
- **Model Runner**: Responsible for model computation and sampling.
- **Decoder**: Responsible for converting sampled Token IDs into texts for output.

In early mainstream inference engines, the Scheduler, Model Runner, and Decoder operated in a serial manner, as shown in Figure 10(a). In this setup, non-GPU operations such as the Scheduler and Decoder occupied a significant portion of the decode time, leading to lower GPU utilization.

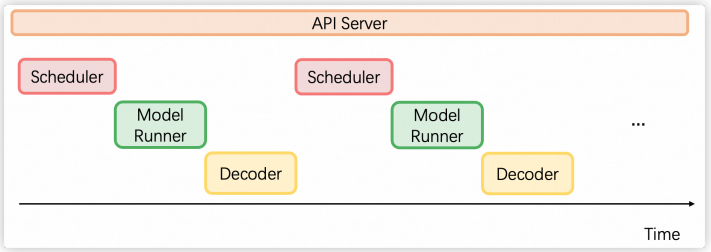
To address these issues, BladeLLM has implemented a fully asynchronous LLM inference architecture called Totally Asynchronous Generator (TAG), as shown in Figure 10(b). Specifically, the three components are handled by three separate processes, where no synchronization is required among them.

1. **Scheduler**: Allocates KV Cache for the next Model Runner step based on anticipated tokens (usually 1, but can be more in speculative sampling) without waiting for previous results of Model Runner.
2. **Model Runner**: Retrieves requests from the queue allocated by the Scheduler and processes them. After processing, it places the sampled token IDs directly into the Decoder's queue and continues with the next computation step.
3. **Decoder**: Asynchronously retrieves token IDs from the queue, converts them to text, and sends them to the API Server.

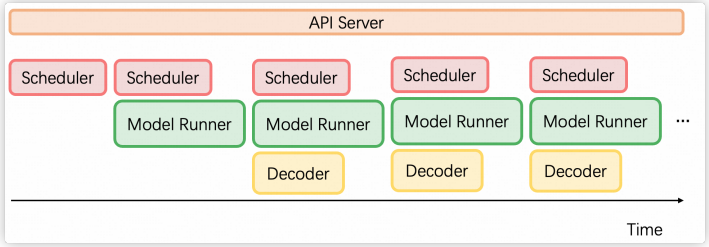
Furthermore, BladeLLM employs shared memory across its components to further reduce inter-process communication overhead. Through these methods, BladeLLM significantly reduces overhead in non-GPU stages of the inference engines, substantially enhancing decoding efficiency.

6 Evaluation

To comprehensively evaluate the performance of the Qwen2.5-1M series models, we will begin by assessing their capabilities in long-context tasks, highlighting the significant improvements achieved through our specialized long-context optimizations. Next, we will examine their performance in short-context tasks and compare these results with those of the 128k version. Finally, we will demonstrate the inference speed of the models.



(a) Serial Execution Pipeline of Early Mainstream Inference Engines



(b) Totally Asynchronous Generator (TAG)

Figure 10: **Scheduling Optimization in BladeLLM**.

6.1 Long Context Benchmarks

We first evaluate the Qwen2.5-1M series of models on the Passkey Retrieval task with a context length of 1 million tokens. As illustrated in Figure 1, both the Qwen2.5-14B-Instruct-1M and Qwen2.5-Turbo models achieved perfect accuracy, successfully identifying all hidden numbers within the contexts up to 1 million tokens. The smaller Qwen2.5-7B-Instruct-1M model also performed admirably, with only a few minor errors. These results highlight the robust retrieval capabilities of the Qwen2.5-1M models when processing extensive 1 million token contexts.

For more advanced tasks, we utilize three benchmarks to evaluate long-context capabilities:

- RULER (Hsieh et al., 2024): An extension of the needle-in-a-haystack task, RULER challenges models to find multiple "needles" or answer multiple questions within irrelevant contexts, or to identify the most or least frequent words in the text. The maximum data length is 128K tokens.
- LV-Eval (Yuan et al., 2024): This benchmark tests a model's ability to understand numerous evidence fragments simultaneously. We have refined the evaluation metrics from the original LV-Eval to avoid false negatives caused by overly strict matching rules. The maximum data length is 256K tokens.
- Longbench-Chat (Bai et al., 2024): A dataset for evaluating human preference alignment in long-context tasks. The maximum data length is 100K tokens.

For baselines, we choose GLM-9B-Chat-1M (Zeng et al., 2024), Llama-3-8B-Instruct-Gradient-1048k (Pekelis et al., 2024), Llama-3.1-70B-Instruct, GPT-4o-mini, and GPT-4o.

The results from the three benchmarks are detailed in Tables 4 and 5. It is evident that the Qwen2.5-1M series models significantly outperform their 128k counterparts in most long-context tasks, particularly for sequences exceeding 64k in length. On the RULER dataset, all models in the Qwen2.5-1M series even surpasses GPT-4, highlighting their exceptional capability in long-context retrieval tasks.

Notably, the Qwen2.5-14B-Instruct-1M model achieved an accuracy of 92.2 on 128k sequences, marking the first time any model in Qwen2.5 series surpassed the 90-point threshold. Additionally, it consistently outperforms GPT-4o-mini across multiple datasets, offering a robust open-source alternative for long-context tasks.

Qwen2.5-Turbo's performance is positioned between that of the Qwen2.5-7B-Instruct-1M and Qwen2.5-14B-Instruct-1M models in the long-context benchmarks. It delivers faster inference speeds and lower costs, making it a more cost-effective and efficient option.

Qwen2.5-72B-Instruct, despite being trained on sequences limited to 32k tokens, consistently outperformed the Qwen2.5-14B-Instruct-1M model across all sequence lengths in the LV-Eval benchmark when

Table 4: **Performance of Qwen2.5 Models on RULER.** *DCA+YaRN* does not change the model behavior within its training length.

Model		Claimed Length	RULER						
			Avg.	4K	8K	16K	32K	64K	128K
GLM4-9b-Chat-1M		1M	89.9	94.7	92.8	92.1	89.9	86.7	83.1
Llama-3-8B-Instruct-Gradient-1048k		1M	88.3	95.5	93.8	91.6	87.4	84.7	77.0
Llama-3.1-70B-Instruct		128K	89.6	96.5	95.8	95.4	94.8	88.4	66.6
GPT-4o-mini		128K	87.3	95.0	92.9	92.7	90.2	87.6	65.8
GPT-4		128K	91.6	96.6	96.3	95.2	93.2	87.0	81.2
Qwen2.5-32B-Instruct	RoPE	32K	88.0	96.9	97.1	95.5	95.5	85.3	57.7
	DCA+YaRN	128K	**92.9**					**90.3**	**82.0**
Qwen2.5-72B-Instruct	RoPE	32K	90.8	97.7	97.2	97.7	96.5	88.5	67.0
	DCA+YaRN	128K	**95.1**					**93.0**	**88.4**
Qwen2.5-7B-Instruct	RoPE	32K	80.1	96.7	95.1	**93.7**	89.4	74.5	31.4
	DCA+YaRN	128K	85.4					82.3	55.1
Qwen2.5-7B-Instruct-1M	RoPE / DCA+YaRN	1M	**91.8**	**96.8**	**95.3**	93.0	**91.1**	**90.4**	**84.4**
Qwen2.5-14B-Instruct	RoPE	32K	86.5	**97.7**	96.8	**95.9**	93.4	82.3	53.0
	DCA+YaRN	128K	91.4					86.7	78.1
Qwen2.5-14B-Instruct-1M	RoPE / DCA+YaRN	1M	**95.7**	97.5	**97.1**	94.6	**94.9**	**94.9**	**92.2**
Qwen2.5-Turbo	RoPE / DCA+YaRN	1M	93.1	97.5	95.7	95.5	94.8	90.8	84.5

Table 5: **Performance of Qwen2.5 Models on LV-Eval and LongBench-Chat.** *DCA+YaRN* does not change the model behavior within its training length.

Model		Claimed Length	LV-Eval					LongBench-Chat
			16K	32K	64K	128K	256K	
GLM4-9B-Chat-1M		1M	46.4	43.2	42.9	40.4	37.0	7.82
Llama-3-8B-Instruct-Gradient-1048k		1M	31.7	31.8	28.8	26.3	21.1	6.20
Llama-3.1-70B-Instruct		128K	48.6	47.4	42.9	26.2	N/A	6.80
GPT-4o-mini		128K	52.9	48.1	46.0	40.7	N/A	8.48
Qwen2.5-32B-Instruct	RoPE	32K	56.0	53.6	40.1	20.5	0.7	-
	DCA+YaRN	128K			**48.8**	**45.3**	**41.0**	8.70
Qwen2.5-72B-Instruct	RoPE	32K	60.4	57.5	47.4	27.0	2.4	-
	DCA+YaRN	128K			**53.9**	**50.9**	**45.2**	8.72
Qwen2.5-7B-Instruct	RoPE	32K	**55.9**	**49.7**	33.1	13.6	0.5	-
	DCA+YaRN	128K			48.0	41.1	36.9	7.42
Qwen2.5-7B-Instruct-1M	RoPE / DCA+YaRN	1M	52.5	49.4	**48.6**	**48.3**	**42.7**	**8.08**
Qwen2.5-14B-Instruct	RoPE	32K	53.0	50.8	37.0	18.4	0.8	-
	DCA+YaRN	128K			46.8	43.6	39.4	8.04
Qwen2.5-14B-Instruct-1M	RoPE / DCA+YaRN	1M	**54.5**	**53.5**	**50.1**	**47.6**	**43.3**	**8.76**
Qwen2.5-Turbo	RoPE / DCA+YaRN	1M	53.4	50.0	45.4	43.9	38.0	8.34

augmented with our length extrapolation method, DCA+YaRN. This result underscores the substantial value of the length extrapolation technique while also highlighting the inherent advantages of larger models in managing complex long-context tasks.

6.2 Short Context Benchmarks

In addition to performance improvements in long-context tasks, we conducted a comprehensive comparison between Qwen2.5-1M and its 128k counterpart on short-context tasks.

We selected widely used benchmarks targeting natural language understanding, coding, mathematics, and reasoning. For general evaluation, we utilized MMLU-Pro (Wang et al., 2024), MMLU-redux (Gema et al., 2024), and LiveBench 0831 (White et al., 2024). For science and mathematics, we evaluated the models on GPQA (Rein et al., 2023), GSM8K (Cobbe et al., 2021), and MATH (Hendrycks et al., 2021). In coding, we assessed performance using HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021), MultiPL-E (Cassano et al., 2023), and LiveCodeBench 2305-2409 (Jain et al., 2024). Additionally, we measured instruction-following capabilities using IFEval (Zhou et al., 2023), where we report results for the strict prompt-level accuracy. To further evaluate human preference alignment and instruction-

Table 6: **Performance of Qwen2.5-7/14B-Instruct with the 1M versions and Qwen2.5-Turbo.**

Datasets	GPT4o-mini	Qwen2.5-7B	Qwen2.5-7B-1M	Qwen2.5-14B	Qwen2.5-14B-1M	Qwen2.5-Turbo
General Tasks						
MMLU-Pro	63.1	56.3	54.3	63.7	63.3	**64.5**
MMLU-redux	81.5	75.4	74.8	80.0	80.7	**81.7**
LiveBench 0831	31.1	35.9	35.2	44.4	**44.6**	42.3
Mathematics & Science Tasks						
GPQA	40.2	36.4	41.4	**45.5**	39.9	42.3
MATH	70.2	75.5	72.9	80.0	79.5	**81.1**
GSM8K	93.2	91.6	91.7	**94.8**	**94.8**	93.8
Coding Tasks						
HumanEval	88.4	84.8	86.0	83.5	**88.4**	86.6
MBPP	**85.7**	79.2	75.9	82.0	80.2	82.8
MultiPL-E	75.0	70.4	72.4	72.8	**77.1**	73.7
LiveCodeBench	40.7	28.7	28.0	**42.6**	38.6	37.8
Alignment Tasks						
IFEval	80.4	71.2	73.0	81.0	**84.3**	76.3
Arena-Hard	**74.9**	52.0	48.1	68.3	70.2	67.1
MTbench	-	8.75	8.30	8.88	**8.89**	8.81

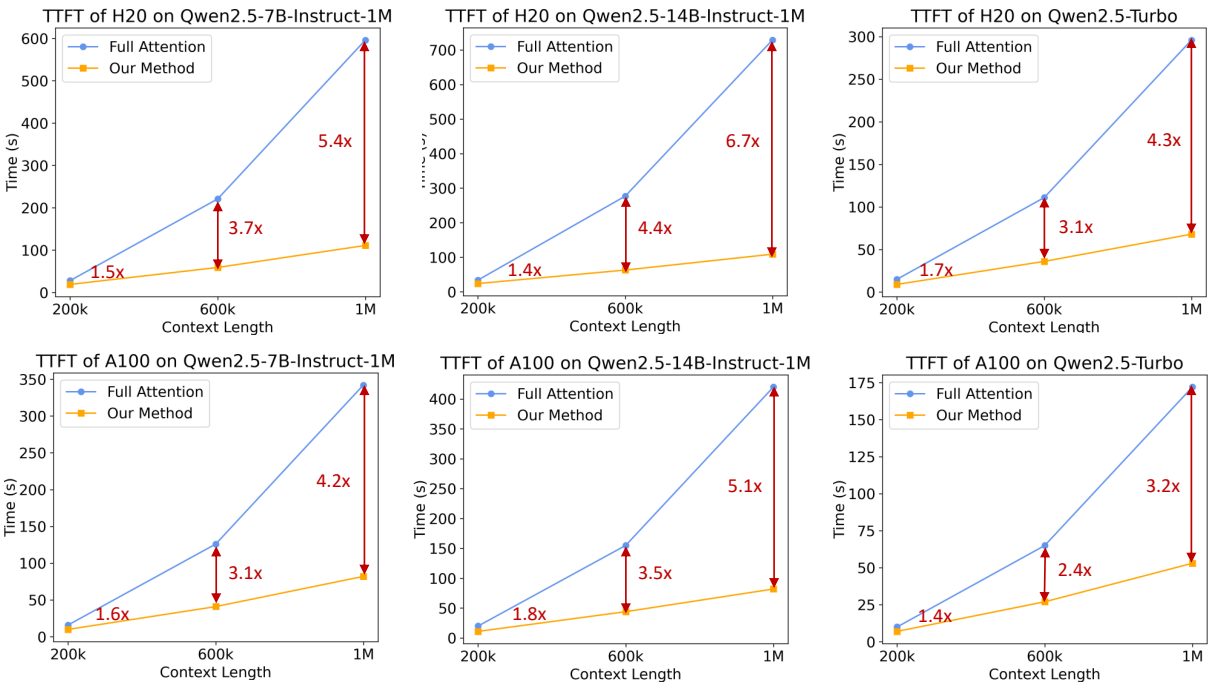


Figure 11: **TTFT (Time to First Token) of Qwen2.5-7B-Instruct-1M, Qwen2.5-14B-Instruct-1M, Qwen2.5-Turbo on H20 and A100 GPUs**.

following performance, we assessed the models on MT-Bench (Zheng et al., 2023) and Arena-Hard (Li et al., 2024).

As shown in Table 6, Qwen2.5-7B-Instruct-1M and Qwen2.5-14B-Instruct-1M maintain performance on short text tasks that is similar to the 128k versions, ensuring that their fundamental capabilities have not been compromised by the addition of long-sequence processing abilities. Compared to GPT-4o-mini, both Qwen2.5-14B-Instruct-1M and Qwen2.5-Turbo achieve similar performance on short text tasks while supporting a context length that is eight times longer.

6.3 Speed Comparison

To demonstrate the acceleration of our final solution on processing long sequences, we evaluate the Time to First Token (TTFT) for different context lengths on Nvidia H20 and A100 GPUs. The experimental configurations are as follows:

- For Qwen2.5-14B-Instruct-1M and Qwen2.5-Turbo, we employed tensor parallelism with 8-way partitioning.
- For Qwen2.5-7B-Instruct-1M, due to the constraints imposed by the Grouped Query Attention mechanism, we utilized tensor parallelism with 4-way partitioning.

In all experiments, we use a batch size of 1.

As illustrated in Figure 11, our method, enhanced with sparse attention and optimized inference engines, achieves a 3.2 to 6.7 times speedup when processing a context length of 1M across various model sizes and devices. For instance, on the H20 GPUs, the Qwen2.5-14B-Instruct-1M model reduces inference time from 12.2 minutes (with full attention) to just 109 seconds. Similarly, the Qwen2.5-Turbo model decreases its processing time from 4.9 minutes to only 68 seconds. These improvements significantly reduce user waiting times for long-sequence tasks.

Compared to the open-source Qwen2.5-1M models, Qwen2.5-Turbo excels in short tasks and achieves competitive results on long-context tasks, while delivering shorter processing times and lower costs. Consequently, it offers an excellent balance of performance and efficiency, making it highly cost-effective.

7 Conclusion

In this technical report, we introduce the Qwen2.5-1M series of models, which includes the open-source models Qwen2.5-7B-Instruct-1M and Qwen2.5-14B-Instruct-1M, as well as the API-accessible model Qwen2.5-Turbo. We detail how these models were developed by extending the Qwen2.5 Base model through long-context pre-training and post-training. We introduce techniques such as data synthesis and progressive training to improve training effectiveness with lower costs.

Beyond training, efficient inference and deployment present significant challenges for long-context models. We have implemented several optimizations, including training-free length extrapolation methods, sparse attention mechanisms, and inference engine enhancements. These optimizations substantially improve the efficiency and reduce the operational costs of running long-sequence models, making practical applications more feasible. We have open-sourced several of these optimizations, and firmly believe that this is the most effective way to drive progress in the field.

We recognize that long-context models still have significant potential for improvement. Our focus is on developing models that excel in both short and long-context tasks, ensuring they deliver substantial value in real-world long-context scenarios. We will continue to explore more efficient training strategies, model architectures, and inference methods, making them deployable effectively and perform exceptionally well even in resource-constrained environments. We are confident that these efforts will expand the applicability of long-context models to a much broader range of use cases.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: Training generalized multi-query Transformer models from multi-head checkpoints. In *EMNLP*, pp. 4895–4901. Association for Computational Linguistics, 2023.

Chenxin An, Fei Huang, Jun Zhang, Shansan Gong, Xipeng Qiu, Chang Zhou, and Lingpeng Kong. Training-free long-context scaling of large language models. *CoRR*, abs/2402.17463, 2024a.

Chenxin An, Jun Zhang, Ming Zhong, Lei Li, Shansan Gong, Yao Luo, Jingjing Xu, and Lingpeng Kong. Why does the effective context length of llms fall short?, 2024b. URL https://arxiv.org/abs/2410.18745.

Anthropic. Introducing Claude, 2023a. URL https://www.anthropic.com/index/introducing-claude.

Anthropic. Claude 2. Technical report, Anthropic, 2023b. URL https://www-files.anthropic.com/production/images/Model-Card-Claude-2.pdf.

Anthropic. The Claude 3 model family: Opus, Sonnet, Haiku. Technical report, Anthropic, AI, 2024. URL https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf.

Jacob Austin, Augustus Odena, Maxwell I. Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. Program synthesis with large language models. *CoRR*, abs/2108.07732, 2021.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report. *CoRR*, abs/2309.16609, 2023.

Yushi Bai, Xin Lv, Jiajie Zhang, Yuze He, Ji Qi, Lei Hou, Jie Tang, Yuxiao Dong, and Juanzi Li. LongAlign: A recipe for long context alignment of large language models. In *EMNLP (Findings)*, pp. 1376–1395. Association for Computational Linguistics, 2024.

Mohammad Bavarian, Heewoo Jun, Nikolas Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. Efficient training of language models to fill in the middle. *CoRR*, abs/2207.14255, 2022. doi: 10.48550/ARXIV.2207.14255. URL https://doi.org/10.48550/arXiv.2207.14255.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *NeurIPS*, 2020.

Federico Cassano, John Gouwar, Daniel Nguyen, Sydney Nguyen, Luna Phipps-Costin, Donald Pinckney, Ming-Ho Yee, Yangtian Zi, Carolyn Jane Anderson, Molly Q. Feldman, Arjun Guha, Michael Greenberg, and Abhinav Jangda. MultiPL-E: A scalable and polyglot approach to benchmarking neural code generation. *IEEE Trans. Software Eng.*, 49(7):3675–3691, 2023.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with io-awareness. In *NeurIPS*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/67d57c32e20fd0a7a302cb81d36e40d5-Abstract-Conference.html.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pp. 933–941. PMLR, 2017.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov,

Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. The Llama 3 herd of models. *CoRR*, abs/2407.21783, 2024.

Aryo Pradipta Gema, Joshua Ong Jun Leang, Giwon Hong, Alessio Devoto, Alberto Carlo Maria Mancino, Rohit Saxena, Xuanli He, Yu Zhao, Xiaotang Du, Mohammad Reza Ghasemi Madani, et al. Are we done with mmlu? *CoRR*, abs/2406.04127, 2024.

Gemini Team. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. Technical report, Google, 2024. URL https://storage.googleapis.com/deepmind-media/gemini/gemini_v1_5_report.pdf.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *NeurIPS Datasets and Benchmarks*, 2021.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? *CoRR*, abs/2404.06654, 2024.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2.5-Coder technical report. *CoRR*, abs/2409.12186, 2024.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. *CoRR*, abs/2403.07974, 2024.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7B. *CoRR*, abs/2310.06825, 2023a.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts. *CoRR*, abs/2401.04088, 2024a.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. *arXiv preprint arXiv:2407.02490*, 2024b.

Zixuan Jiang, Jiaqi Gu, Hanqing Zhu, and David Z. Pan. Pre-RMSNorm and Pre-CRMSNorm Transformers: Equivalent and efficient pre-LN Transformers. *CoRR*, abs/2305.14858, 2023b.

Gregory Kamradt. Needle in a haystack - pressure testing LLMs, 2023. URL https://github.com/gkamradt/LLMTest_NeedleInAHaystack.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-Hard and BenchBuilder pipeline. *CoRR*, abs/2406.11939, 2024.

OpenAI. GPT4 technical report. *CoRR*, abs/2303.08774, 2023.

OpenAI. Hello GPT-4o, 2024. URL https://openai.com/index/hello-gpt-4o/.

Leonid Pekelis, Michael Feil, Forrest Moret, Mark Huang, and Tiffany Peng. Llama 3 gradient: A series of long context models, 2024. URL https://gradient.ai/blog/scaling-rotational-embeddings-for-long-context-language-models.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models. *CoRR*, abs/2309.00071, 2023.

Qwen Team. Code with CodeQwen1.5, 2024a. URL https://qwenlm.github.io/blog/codeqwen1.5/.

Qwen Team. Generalizing an llm from 8k to 1m context using qwen-agent, May 2024b. URL https://qwenlm.github.io/blog/qwen-agent-2405/.

Qwen Team. Introducing Qwen2-Math, 2024c. URL https://qwenlm.github.io/blog/qwen2-math/.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *NeurIPS*, 2023.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level Google-proof Q&A benchmark. *CoRR*, abs/2311.12022, 2023.

Jianlin Su. The magical effect of the Bias term: RoPE + Bias = better length extrapolation, 2023. URL https://spaces.ac.cn/archives/9577.

Jianlin Su, Murtadha H. M. Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced Transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023b.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. *CoRR*, abs/2406.01574, 2024.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Benjamin Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Siddartha Naidu, Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. LiveBench: A challenging, contamination-free LLM benchmark. *CoRR*, abs/2406.19314, 2024.

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, Madian Khabsa, Han Fang, Yashar Mehdad, Sharan Narang, Kshitiz Malik, Angela Fan, Shruti Bhosale, Sergey Edunov, Mike Lewis, Sinong Wang, and Hao Ma. Effective long-context scaling of foundation models. *CoRR*, abs/2309.16039, 2023.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 technical report. *CoRR*, abs/2407.10671, 2024a.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2.5-Math technical report: Toward mathematical expert model via self-improvement. *CoRR*, abs/2409.12122, 2024b.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *CoRR*, abs/2412.15115, 2025.

Tao Yuan, Xuefei Ning, Dong Zhou, Zhijie Yang, Shiyao Li, Minghui Zhuang, Zheyue Tan, Zhuyu Yao, Dahua Lin, Boxun Li, Guohao Dai, Shengen Yan, and Yu Wang. LV-Eval: A balanced long-context benchmark with 5 length levels up to 256K. *CoRR*, abs/2402.05136, 2024.

Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Juanzi Li, Lei Zhao, Lindong Wu, Lucen Zhong, Mingdao Liu, Minlie Huang, Peng Zhang, Qinkai Zheng, Rui Lu, Shuaiqi Duan, Shudan Zhang, Shulin Cao, Shuxun Yang, Weng Lam Tam, Wenyi Zhao, Xiao Liu, Xiao Xia, Xiaohan Zhang, Xiaotao Gu, Xin Lv, Xinghan Liu, Xinyi Liu, Xinyue Yang, Xixuan Song, Xunkai Zhang, Yifan An, Yifan Xu, Yilin Niu, Yuantao Yang, Yueyan Li, Yushi Bai, Yuxiao Dong, Zehan Qi, Zhaoyu Wang, Zhen Yang, Zhengxiao Du, Zhenyu Hou, and Zihan Wang. ChatGLM: A family of large language models from GLM-130B to GLM-4 all tools. *CoRR*, abs/2406.12793, 2024.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *NeurIPS*, 2023.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *CoRR*, abs/2311.07911, 2023.