本文由 AINLP 公众号整理翻译，更多 LLM 资源请扫码关注!

LongCat-Flash Technical Report

Meituan LongCat Team
longcat-team@meituan.com

ABSTRACT

We introduce LongCat-Flash, a 560-billion-parameter Mixture-of-Experts (MoE) language model designed for both computational efficiency and advanced agentic capabilities. Stemming from the need for scalable efficiency, LongCat-Flash adopts two novel designs: (a) *Zero-computation Experts*, which enables dynamic computational budget allocation and activates 18.6B–31.3B (27B on average) per token depending on contextual demands, optimizing resource usage. (b) *Shortcut-connected MoE*, which enlarges the computation-communication overlap window, demonstrating notable gains in inference efficiency and throughput compared to models of a comparable scale. We develop a comprehensive scaling framework for large models that combines hyperparameter transfer, model-growth initialization, a multi-pronged stability suite, and deterministic computation to achieve stable and reproducible training. Notably, leveraging the synergy among scalable architectural design and infrastructure efforts, we complete model training on more than 20 trillion tokens within 30 days, while achieving over 100 tokens per second (TPS) for inference at a cost of $0.70 per million output tokens. To cultivate LongCat-Flash towards agentic intelligence, we conduct a large-scale pre-training on optimized mixtures, followed by targeted mid- and post-training on reasoning, code, and instructions, with further augmentation from synthetic data and tool use tasks. Comprehensive evaluations demonstrate that, as a non-thinking foundation model, LongCat-Flash delivers highly competitive performance among other leading models, with exceptional strengths in agentic tasks. The model checkpoint of LongCat-Flash is open-sourced to foster community research.

LongCat Chat: https://longcat.ai
Hugging Face: https://huggingface.co/meituan-longcat
Github: https://github.com/meituan-longcat

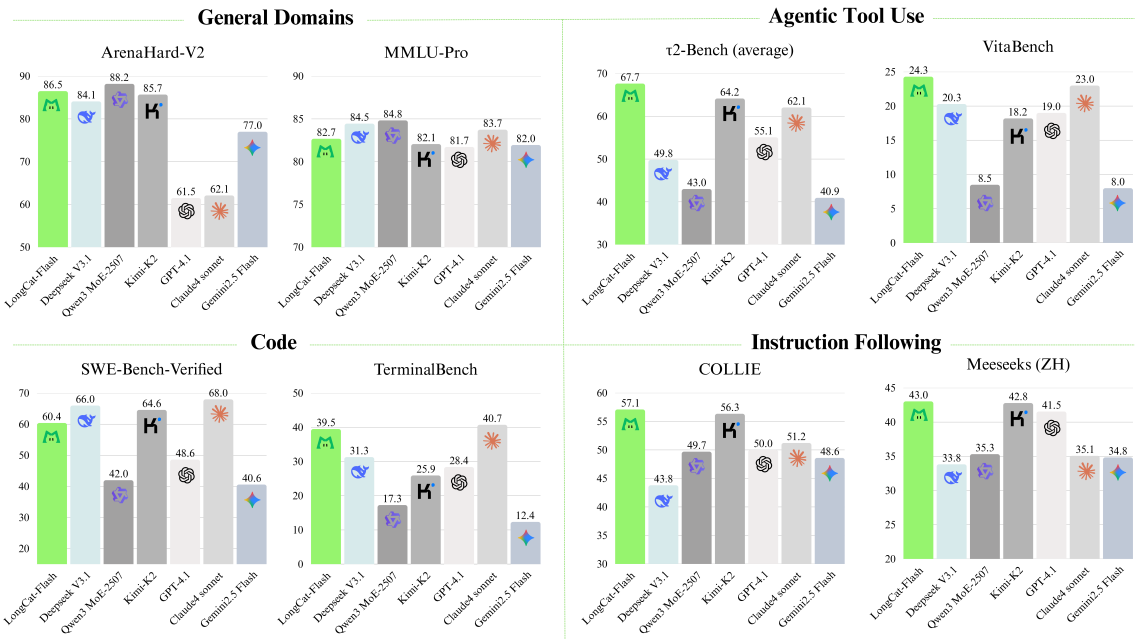


Figure 1: Benchmark performance of LongCat-Flash.

LongCat-Flash 技术报告

美团 LongCat 团队 longcat
-team@meituan.com

摘要

我们推出 LongCat-Flash，一种拥有五千六百亿参数的专家混合（MoE）语言模型，旨在实现计算效率与先进的代理能力。出于对可扩展性效率的需求，LongCat-Flash 采用两种新颖设计： (a) *Zero-computation Experts*，它使动态计算预算分配成为可能，并根据上下文需求在每个标记上激活 18.6B–31.3B（平均 27B），以优化资源使用。 (b) *Shortcut-connected MoE*，它扩大了计算-通信重叠窗口，与同等规模的模型相比，在推理效率和吞吐量方面显示出显著提升。我们开发了一个面向大模型的综合扩展框架，结合超参数迁移、模型增长初始化、多方位的稳定性套件，以及确定性计算，以实现稳定且可重复的训练。值得注意的是，利用可扩展架构设计与基础设施工作的协同作用，我们在 30 天内完成了超过 20 万亿个标记的模型训练，同时推理达到每秒超过 100 个标记（TPS），输出成本为每输出一百万标记 0.70 美元。为了将 LongCat-Flash 培养成为具备代理性智能的模型，我们对经过优化的混合进行大规模预训练，随后对推理、代码和指令进行定向的中期和后期训练，并通过合成数据和工具使用任务进一步增强。全面评估表明，作为一个非思考型基础模型，LongCat-Flash 在与其他领先模型的比较中具有高度竞争力，在代理任务方面具有卓越优势。LongCat-Flash 的模型检查点已开源，以促进社区研究。

LongCat 聊天： https://longcat.ai Hugging Face： htt
ps://huggingface.co/meituan-longcat GitHub： https://
github.com/meituan-longcat

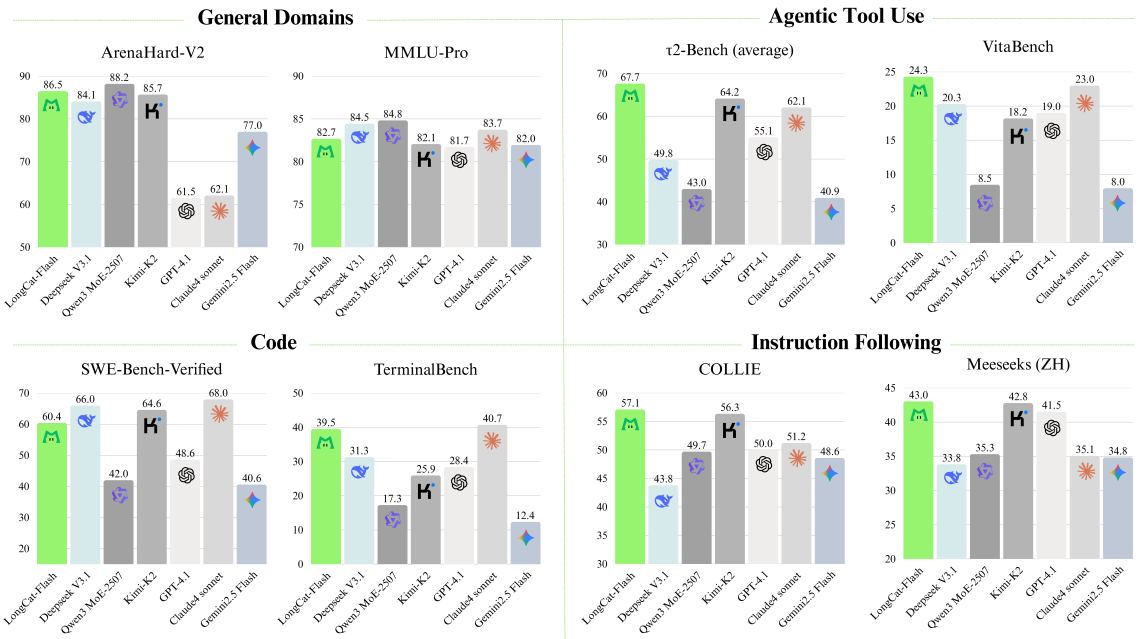


图1：LongCat-Flash 的基准性能。

Contents

目录

1 Introduction

The rapid advancement of large language models (LLMs) such as DeepSeek-V3 [DeepSeek-AI et al., 2025], Qwen 3 [Yang et al., 2025], and Kimi-K2 [Team et al., 2025] has demonstrated the effectiveness of scaling model size and computational resources. While some recent progress raises concerns about potential scaling slowdowns, we believe that algorithmic design, underlying system optimizations, and data strategy all play equally critical roles in further pushing the frontier of scalable intelligence. This requires innovations in both model architecture and training strategies to improve the cost-effectiveness of scaling, as well as a systematic data strategy to enhance the model's capability for solving real-world tasks.

In this work, we introduce LongCat-Flash, an efficient yet powerful Mixture-of-Experts (MoE) language model designed to advance the frontier of language model along two synergistic directions: *computational efficiency* and *agentic capability*. Trained on tens of thousands of accelerators, LongCat-Flash combines architectural innovations with a sophisticated, multi-stage training methodology for scalable and intelligent models. Our contributions span both efficiency and agentic intelligence:

- **Scalable Architectural Design for Computational Efficiency** LongCat-Flash is designed and optimized under two key principles: efficient computation utilization, as well as efficient training and inference. Specifically, (1) As *not all tokens are equal*, we introduce the *zero-computation experts* mechanism in MoE blocks to allocate a dynamic computation budget to important tokens based on their significance, i.e., activating 18.6 to 31.3 billion parameters (out of 560 billion total) based on contextual demands. To ensure consistent computation load, we employ expert bias adjusted by a PID-controller, maintaining an average of $\sim$27 billion activated parameters per token. (2) As communication overhead becomes a bottleneck during MoE model scaling, we incorporate the *Shortcut-connected MoE (ScMoE)* [Cai et al., 2024] design to expand the computation-communication overlap window. Combined with customized infrastructure optimizations, this design enables training at a massive scale of over tens of thousands accelerators and inference with high throughput and low latency.
- **Effective Model Scaling Strategy** Effectively and efficiently scaling model size remains a key challenge in strategy design. To this end, we develop a comprehensive stability-and-scaling framework for robustly training large-scale models: (1) We successfully apply a hyperparameter transfer strategy to such a large model, predicting optimal hyperparameter configurations by leveraging results from smaller proxy models with theoretical guarantees. (2) We initialize the model using a model-growth mechanism based on a refined half-scale checkpoint, achieving improved performance compared to conventional initialization methods. (3) A multi-pronged stability suite incorporates principled router-gradient balancing, a hidden z-loss to suppress massive activations, and fine-tuned optimizer configurations. (4) To enhance the reliability of large-scale cluster training, we introduce deterministic computation. This guarantees the exact reproducibility of experiments and enables the detection of SDC (Silent Data Corruption) during the training process. These interventions ensure that LongCat-Flash 's training remains stable, with no irrecoverable loss spikes.
- **Multi-Stage Training Pipeline for Agentic Capability** Through a meticulously designed pipeline, LongCat-Flash is endowed with advanced agentic behaviors. Initial efforts focus on constructing a more suitable base model for agentic post-training, where we design a two-stage pretraining data fusion strategy to concentrate reasoning-intensive domain data. During mid-training, we enhance reasoning and coding capabilities while extending the context length to 128k to meet agentic post-training requirements. Building on this advanced base model, we proceed with a multi-stage post-training. Recognizing the scarcity of high-quality, high-difficulty training problems for agentic tasks, we design a multi-agent synthesis framework that defines task difficulty across three axes, i.e., information processing, tool-set complexity, and user interaction—using specialized controllers to generate complex tasks requiring iterative reasoning and environmental interaction.

Overall, benefiting from our synergy among scalable architectural design, training strategies, and infrastructure efforts, LongCat-Flash achieves both high training throughput and low inference latency. Notably, we complete the pre-training of our 560B model over 20T tokens within 30 days and achieve 98.48% time availability without manual intervention for fault resolution. During inference, large-scale deployment efficiency exceeds 100 tokens per second (TPS) on H800, with a cost of \$0.7 per million output tokens, demonstrating remarkable performance compared to models with similar size.

We evaluate the base and instruction-tuned versions of LongCat-Flash across diverse benchmarks, with an overview summarized in Figure 1. As a non-thinking model, LongCat-Flash achieves performance comparable to state-of-the-art non-thinking models, including DeepSeek-V3.1 [DeepSeek-AI et al., 2025] and Kimi-K2 [Team et al., 2025], while using fewer parameters and offering faster inference speed. Specifically, LongCat-Flash scores 86.5 on ArenaHard-V2, 39.5 on TerminalBench, and 67.7 on τ^2-Bench, demonstrating robust capabilities in general domains, coding, and agentic tool use. To mitigate potential contamination from existing open-source benchmarks and enhance evaluation confidence,

1 引言

大规模语言模型（LLMs）如 DeepSeek-V3 [DeepSeek-AI et al., 2025]、Qwen 3 [Yang et al., 2025] 和 Kimi-K2 [Team et al., 2025] 的迅速发展已经证明了扩大模型规模和计算资源的有效性。尽管一些近期的进展引发了对潜在扩展放缓的担忧，但我们相信，算法设计、底层系统优化以及数据策略在进一步推动可扩展智能的前沿中同样扮演着至关重要的角色。这需要在模型架构和训练策略方面进行创新，以提高扩展的成本效益，以及一个系统化的数据策略，以提升模型解决现实世界任务的能力。

本工作提出 LongCat-Flash，一种高效而强大的专家混合（MoE）语言模型，旨在沿着两个协同方向推动语言模型的前沿：*computational efficiency* 和 *agentic capability*。在数万台加速器上进行训练，LongCat-Flash 将结构创新与复杂的多阶段训练方法相结合，实现可扩展且智能的模型。我们的贡献覆盖效率与具备代理性智能的能力：

- 可扩展的计算效率架构设计。LongCat-Flash 的设计与优化基于两个关键原则：高效的计算利用，以及高效的训练与推理。具体而言，正如 *not all tokens are equal*，我们在 MoE 块中引入 *zero-computation experts* 机制，以根据重要性为重要令牌分配动态计算预算，即基于上下文需求激活约 186 亿至 313 亿参数（占总计 5600 亿参数中的一部分）。为确保计算负载的一致性，我们采用经由 PID 控制器调整的专家偏置，平均每个令牌激活的参数数量为 ∼27 十亿。 （2）随着 MoE 模型规模扩展，通信开销成为瓶颈，我们引入 *Shortcut-connected MoE (ScMoE)* [Cai et al., 2024] 的设计来扩大计算-通信的重叠窗口。结合定制化的基础设施优化，这一设计使训练在数万颗加速器规模下成为可能，并实现高吞吐量、低延迟的推理。

- 有效且高效地扩展模型规模仍是策略设计中的一项关键挑战。为此，我们开发了一套全面的稳定性与扩展性框架，用以稳健地训练大规模模型： (1) 我们成功地将超参数迁移策略应用于如此大规模的模型，通过利用较小代理模型的结果并具备理论保证，预测出最佳超参数配置。 (2) 我们基于改进的半尺度检查点的模型增长机制对模型进行初始化，与传统初始化方法相比，取得了更好的性能。 (3) 这一多方面的稳定性套件包含基于原理的路由梯度平衡、用于抑制大规模激活的隐藏 z-loss，以及经过微调的优化器配置。 (4) 为了提高大规模集群训练的可靠性，我们引入确定性计算。这保证了实验的精确可重复性，并使在训练过程中能够检测到 SDC（静默数据损坏）。这些干预措施确保 LongCat-Flash 的训练保持稳定，不会出现不可恢复的损失尖峰。

- 用于具备代理能力的多阶段训练管线　通过精心设计的流程LongCat-Flash 获得了先进的代理行为。初期工作聚焦于构建更适合进行代理后训练的基础模型，在此我们设计了一个两阶段的预训练数据融合策略，以聚集推理密集型的领域数据。在中期训练阶段，我们在提升推理和编码能力的同时，将上下文长度扩展到 128k，以满足代理后训练的要求。在这一先进的基础模型之上，我们进行多阶段的后训练。鉴于用于代理任务的高质量、高难度训练问题稀缺，我们设计了一个多智能体综合框架，在三个维度上定义任务难度，即信息处理、工具集复杂性和用户交互——使用专门的控制器来生成需要迭代推理和环境交互的复杂任务。

总体而言，得益于我们在可扩展的体系架构设计、训练策略与基础设施建设方面的协同作用，LongCat-Flash 实现了高训练吞吐量与低推理延迟的双重目标。值得注意的是，我们在 30 天内完成了 560B 模型的预训练，总计处理了 20T 令牌，并实现了 98.48% 的可用性且无需人工干预来解决故障。在推理阶段，H800 上的大规模部署效率超过每秒 100 令牌（TPS），输出令牌成本为每百万令牌 0.7 美元，相较于同等规模的模型，展现出显著的性能。

我们在多样化的基准测试中评估 LongCat-Flash 的基础版本和指令微调版本，概览如图1所示。作为一个非思考型模型，LongCat-Flash 的性能可与最先进的非思考型模型相媲美，包括 DeepSeek-V3.1 [DeepSeek-AI 等，2025] 和 Kimi-K2 [Team 等，2025]，同时使用更少的参数并提供更快的推断速度。具体而言，LongCat-Flash 在 ArenaHard-V2 上得分 86.5，在 TerminalBench 上得分 39.5，在 τ^2-Bench 上得分 67.7，展现了在通用领域、编码和代理工具使用方面的稳健能力。为降低来自现有开源基准测试的潜在污染并提升评估的可信度，

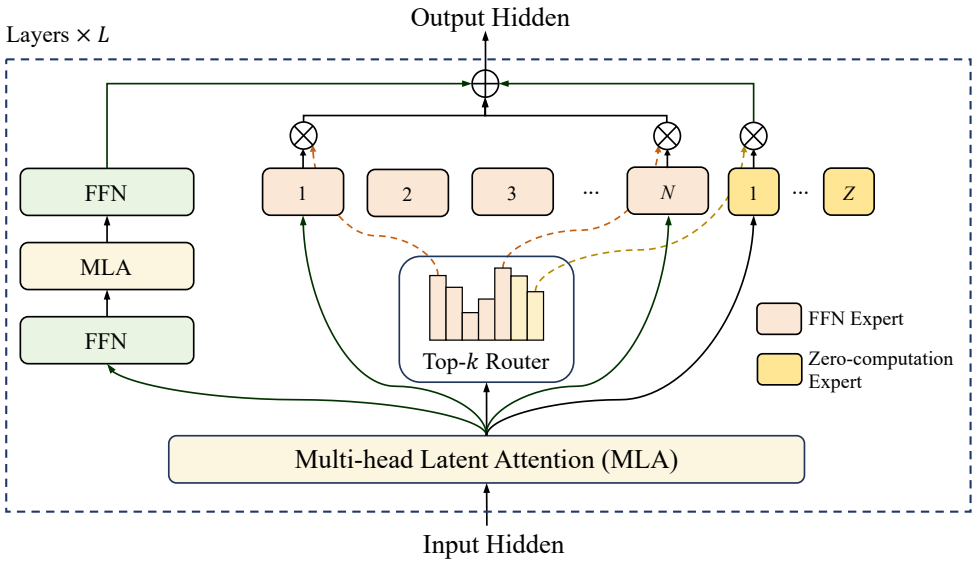


Figure 2: The architecture adopted in LongCat-Flash. Each layer employs Shortcut-connected Mixture of Experts (ScMoE) with zero-computation experts. ScMoE significantly expands the computation-communication window to boost training and inference efficiency. The zero-computation experts enable dynamic computation based on contextual importance, improving the efficiency of computational resource utilization.

we meticulously constructed two new benchmarks: Meeseeks [Wang et al., 2025a] and VitaBench. Meeseeks simulates realistic human-LLM interactions through an iterative feedback framework to evaluate multi-turn instruction-following ability, where LongCat-Flash achieves scores on par with frontier LLMs. VitaBench leverages real-world business scenarios to access models' proficiency in addressing complex real-world tasks, where LongCat-Flash delivers superior performance than other LLMs.

In the remainder of this report, we first detail the architecture and innovations in LongCat-Flash. Then, we describe the pre-training and post-training processes, including our training strategies, data construction methods, and evaluation results. Finally, we discuss the challenges and solutions in training LongCat-Flash, along with optimized inference and deployment methods that leverage its unique architecture.

2 Architecture

LongCat-Flash adopts a novel MoE architecture with two key innovations (Figure 2): (1) The MoE block incorporates zero-computation experts [Jin et al., 2024] to enable dynamic computation, allowing tokens to consume variable computational resources based on their contextual significance. Furthermore, the average computational load is regulated through an adaptive expert bias. (2) Each layer integrates two Multi-head Latent Attention (MLA) block [Liu et al., 2024a] and multiple heterogeneous Feed-Forward Network (FFN) blocks. A *shortcut* connection from the first MLA output directly to the MoE block [Cai et al., 2024] is employed. To further enhance performance, we refine both the MLA and fine-grained FFN experts via variance alignment. The following subsections will detail each of these components.

2.1 Zero-Computation Experts

Next-token prediction exhibits inherent computational heterogeneity. Difficult tokens may demand more resources for accurate prediction, while easy tokens require negligible computation. This phenomenon is also empirically evidenced by speculative decoding, where small draft models reliably predict the outputs of large models for most easy tokens [Leviathan et al., 2023].

Motivated by this, LongCat-Flash presents a dynamical computational resource allocation mechanism by activating a variable number of FFN experts per token through zero-computation experts [Jin et al., 2024, Zeng et al., 2024], enabling a more reasonable allocation of computations according to contextual significance. Specifically, LongCat-Flash expands its expert pool with Z zero-computation experts in addition to N standard FFN experts. Zero-computation

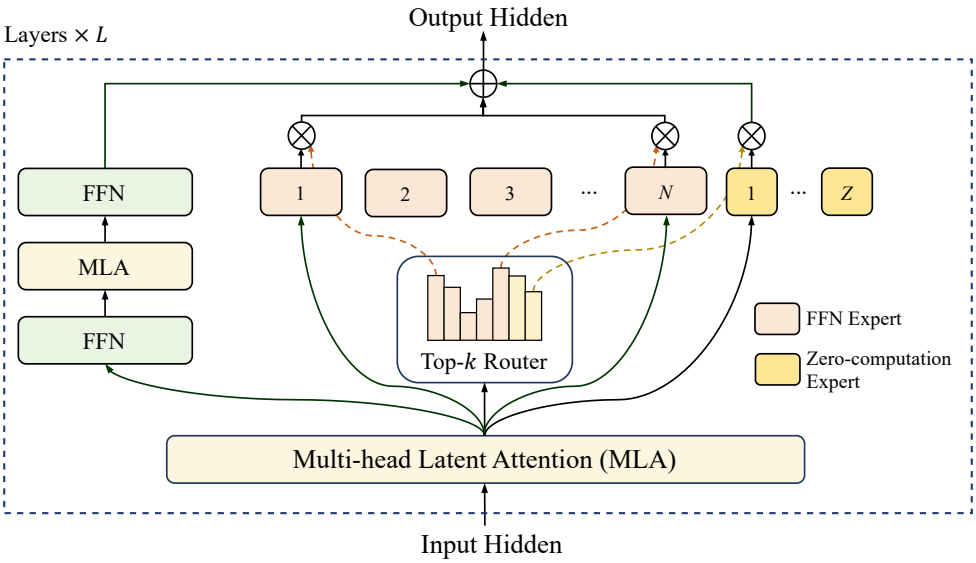


图2：LongCat-Flash中采用的架构。每一层采用带跳跃连接的专家混合模型（ScMoE），并配备零计算专家。ScMoE显著扩展了计算-通信窗口，以提升训练和推理效率。零计算专家使基于上下文重要性的动态计算成为可能，从而提高计算资源利用效率。

我们精心构建了两个新的基准测试：Meeseeks [Wang et al., 2025a] 与 VitaBench。Meeseeks 通过一个迭代反馈框架，模拟现实的人类与大语言模型的互动，以评估多轮指令遵循能力，其中 LongCat-Flash 的分数与前沿的大语言模型相当。VitaBench 利用现实世界的商业场景来评估模型在解决复杂现实任务方面的能力，其中 LongCat-Flash 的表现优于其他大语言模型。

在本报告的其余部分，我们首先详细介绍 LongCat-Flash 的架构与创新。然后，我们描述预训练和后训练过程，包括我们的训练策略、数据构建方法以及评估结果。最后，我们讨论在训练 LongCat-Flash 过程中所面临的挑战与解决方案，以及利用其独特架构的优化推理和部署方法。

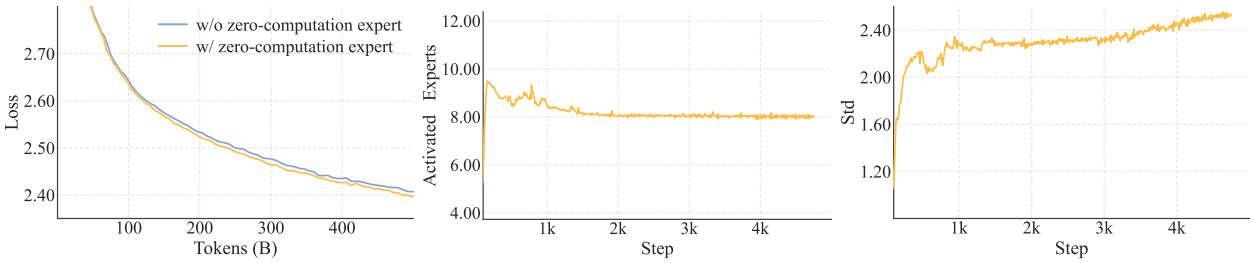
2 架构

LongCat-Flash 采用一种新颖的 MoE 架构，具有两个关键创新（图2）： (1) MoE 块引入零计算专家 [Jin et al., 2024] 以实现动态计算，允许令牌根据其上下文的重要性消耗可变的计算资源。此外，平均计算负载通过自适应专家偏置进行调节。 (2) 每一层集成两个多头潜在注意力（MLA）块 [Liu et al., 2024a] 和多个异构前馈网络（FFN）块。一个从第一块 MLA 的输出直接连接到 MoE 块的 *shortcut* 连接被采用。为了进一步提升性能，我们通过方差对齐来优化 MLA 和细粒度 FFN 专家。 以下各小节将详细介绍这些组件。

2.1 零计算专家

下一个标记的预测具有固有的计算异质性。困难的标记在预测准确性方面可能需要更多资源，而容易的标记则需要的计算量可以忽略不计。这一现象也通过投机解码得到经验性证据，其中小型草拟模型可以在大多数易于预测的标记上可靠地预测大型模型的输出 [Leviathan et al., 2023]。

受到此启发，LongCat-Flash 提出了一种通过为每个标记激活可变数量的 FFN 专家来实现动态计算资源分配的机制，该机制通过零计算专家 [Jin 等, 2024, Zeng 等, 2024] 实现，使计算资源的分配根据上下文的重要性更加合理。具体而言，LongCat-Flash 在其专家池中再增加 Z 个零计算专家，作为对 N 个标准 FFN 专家的补充。零计算



(a) (b) (c)

Figure 3: **(a)** Validation loss curve comparing models with/without zero-computation experts under matched computation budgets. The baseline (top-k=8, blue) activates fixed 6B parameters per token, while the zero-expert variant (top-k=12, orange) dynamically activates 4.2B-7.0B parameters but maintains 8 FFN experts expectation (with fluctuation less than 1%). The consistent loss reduction demonstrates the efficacy of zero-computation experts. **(b)** The average number of activated FFN experts during LongCat-Flash training. The average number remains closely around 8, corresponding to expected 27B activated parameters. **(c)** The standard deviation of activated FFN experts grows to 3, indicating substantial variability in activated parameters across different tokens.

experts simply return the input x_t as their output, thereby introducing no additional computational cost. Let x_t be the t-th token of the input sequence, the MoE module in LongCat-Flash can be formulated as follows:

$$\begin{aligned}
\text{MoE}(x_t) &= \sum_{i=1}^{N+Z} g_i \, E_i(x_t), \\
g_i &= \begin{cases} R(x_t)_i, & \text{if } R(x_t)_i \in \text{TopK}\big(R(x_t)_i + b_i \;\big|\; 1 \le i \le N+Z, K\big), \\ 0, & \text{otherwise}, \end{cases} \\
E_i(x_t) &= \begin{cases} \text{FFN}_i(x_t), & \text{if } 1 \le i \le N, \\ x_t, & \text{if } N < i \le N+Z, \end{cases}
\end{aligned} \tag{1}$$

where R denotes the softmax router, b_i is the expert bias corresponding to the i-th expert, and K denotes the number of experts selected per token.

The router assigns each token to K experts, where the number of activated FFN experts varies per token based on contextual importance. Through this adaptive allocation mechanism, the model learns to dynamically allocate more computational resources to tokens with higher contextual importance, thus achieving superior performance under the same computational capacity as illustrated in Figure 3a.

2.1.1 Computational Budget Control

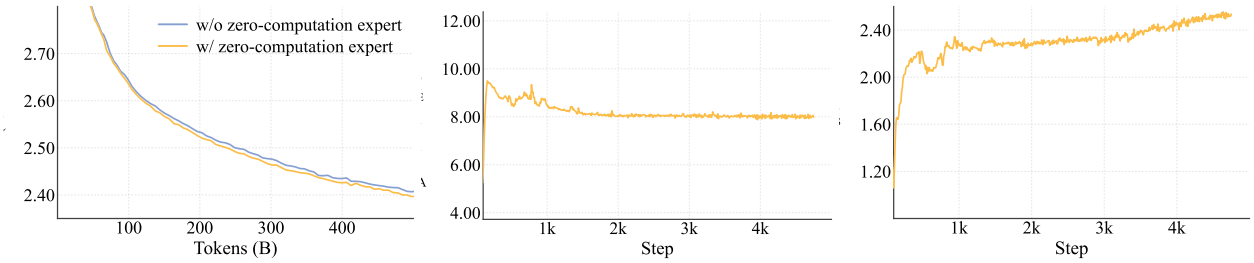
To incentivize the model to learn context-dependent computation allocation, fine-grained control over the average selection ratio of zero-computation experts is essential. Without explicit constraints, the model tends to under-utilize zero-computation experts, leading to inefficient resource usage.

We accomplish this by refining the expert bias mechanism from the aux-loss-free strategy [Wang et al., 2024a], introducing an expert-specific bias term that dynamically adjusts routing scores based on recent expert utilization, while remaining decoupled from the language model (LM) training objective. For the expert bias b_i corresponding to the i-th expert, it is updated each step with the incremental computed as:

$$\Delta b_i = \begin{cases} \mu \left(\dfrac{K_e}{K} \cdot \dfrac{1}{N} - \dfrac{T_i}{T_{\text{all}}} \right), & \text{if } 1 \le i \le N, \\ 0, & \text{if } N < i \le N+Z, \end{cases} \tag{2}$$

where μ denotes the bias adaptation rate, T_{all} denotes the number of tokens in a global batch, T_i denotes the number of tokens routed to the i-th expert, K_e denotes the expected number of activated FFN experts, which is smaller than K.

The proposed update rule employs a PID controller (proportional-integral-derivative) from control theory [Bennett, 1993], ensuring that the token allocation for the i-th expert converges to its target proportion. Compared to a fixed bias



(a) (b) (c)

图3：(a) 在匹配的计算预算下，对比有/无零计算专家的模型的验证损失曲线。基线（top-k=8，蓝色）在每个标记处激活固定的6B参数，而无专家变体（top-k=12，橙色）动态激活4.2B-7.0B参数，但维持8个FFN专家的期望值（波动小于1%）。持续的损失下降证明了零计算专家的有效性。 (b) LongCat-Flash 训练过程中激活的FFN专家的平均数量。该平均值保持在接近8的水平，与预期的27B激活参数相对应。 (c) 激活的FFN专家的标准差增至3，表明不同标记之间激活参数存在显著变动。

专家们只是将输入 x_t 作为它们的输出，从而不会引入额外的计算成本。设 x_t 为输入序列的第 t 个标记，LongCat-Flash 中的 MoE 模块可以表述如下：

$$
\begin{aligned}
\mathrm{MoE}(x_t) &= \sum_{i=1}^{N+Z} g_i \, E_i(x_t), \\
g_i &= \begin{cases} R(x_t)_i, & \text{if } R(x_t)_i \in \mathrm{TopK}\big(R(x_t)_i + b_i \;\big|\; 1 \le i \le N+Z, K\big), \\ 0, & \text{otherwise}, \end{cases} \\
E_i(x_t) &= \begin{cases} \mathrm{FFN}_i(x_t), & \text{if } 1 \le i \le N, \\ x_t, & \text{if } N < i \le N+Z, \end{cases}
\end{aligned}
\tag{1}
$$

其中，R 表示 softmax 路由器，b_i 是对应于第i个专家的专家偏置，K 表示每个词元所选的专家数量。

路由器将每个令牌分配给 K 位专家，其中激活的前馈神经网络（FFN）专家数量根据上下文重要性在每个令牌上变化。通过这种自适应分配机制，模型学习在具有更高上下文重要性的令牌上动态分配更多计算资源，从而在相同计算容量下实现更优的性能，如图3a所示。

2.1.1 计算预算控制

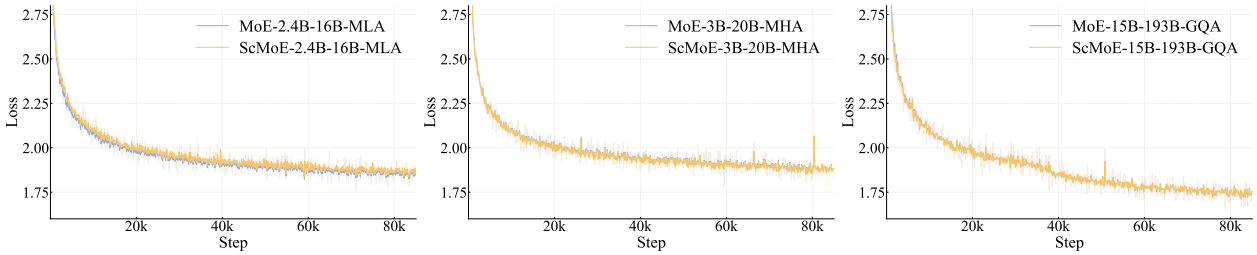
为促使模型学习上下文相关的计算分配，对零计算专家的平均选择比例进行细粒度控制是必不可少的。若没有明确的约束，模型往往无法充分利用零计算专家，从而导致资源使用效率低下。

我们通过改进来自 aux-loss-free 策略 [Wang et al., 2024a] 的专家偏置机制来实现这一点，引入一个面向特定专家的偏置项，该项基于最近的专家使用情况动态调整路由分数，同时与语言模型（LM）训练目标解耦。对于对应于 i-th 专家的专家偏置 b_i，它在每一步更新，增量的计算如下：

$$
\Delta b_i = \begin{cases} \mu \left(\dfrac{K_e}{K} \cdot \dfrac{1}{N} - \dfrac{T_i}{T_{\text{all}}} \right), & \text{if } 1 \le i \le N, \\ 0, & \text{if } N < i \le N+Z, \end{cases}
\tag{2}
$$

其中 μ 表示偏置自适应率，T_{all} 表示全局批次中的标记数量，T_i 表示路由到第 i 个专家的标记数量，K_e 表示被激活的 FFN 专家的期望数量，该数量小于 K。

所提出的更新规则采用来自控制理论的 PID 控制器（比例-积分-微分）[Bennett, 1993]，确保对第 i-th 专家的令牌分配收敛到其目标比例。相比固定偏置



(a) (b) (c)

Figure 4: Training loss curves comparing baseline models (without ScMoE) against their ScMoE-enhanced counterparts across four different model configurations. In all experiments—**(a)** 2.4B-16B with MLA, **(b)** 3B-20B with MHA, and **(c)** 15B-193B with GQA—the loss curves are virtually indistinguishable. This provides robust evidence that the ScMoE optimization is quality-neutral, and its benefits are orthogonal to both model scale and the specific attention architecture used.

increment [Wang et al., 2024a], this mechanism improves the robustness of the softmax router's probability distribution as the number of experts scales. Notably, we exclude zero-computation experts from bias updates, as their identity nature only requires a global constraint, which is automatically satisfied when all FFN experts achieve their expected token proportions. Empirically, large batch sizes and a decay schedule for μ improve the stability of budget control, while small batch sizes may require reduced update frequency.

During pre-training, we tracked the average number and standard deviation of activated experts (Figure 3b and 3c). The results show that after approximate 20B tokens of adjustment, the average expert number in all layers converged to the expected value, with fluctuations less than 1%. However, the standard deviation persisted at a relatively high level, indicating that the model allocates substantially divergent computational resources across tokens.

For detailed statistics and case studies of dynamic routing, please refer to Appendix A.1.

2.1.2 Load Balance Control

Efficient MoE training requires robust load balancing among FFN experts. While Eq. (2) enforces balance at the corpus level, a device-level load balance loss [DeepSeek-AI et al., 2025] to further prevent extreme sequence-level imbalance among EP groups is introduced. We make necessary efforts to accommodate the zero-computation experts. Specifically, assuming that all N FFN experts are divided into D groups, each group containing $G = \frac{N}{D}$ experts, the loss can be expressed as:

$$\mathcal{L}_{\text{LB}} = \alpha \sum_{j=1}^{D+1} f_j P_j, \tag{3}$$

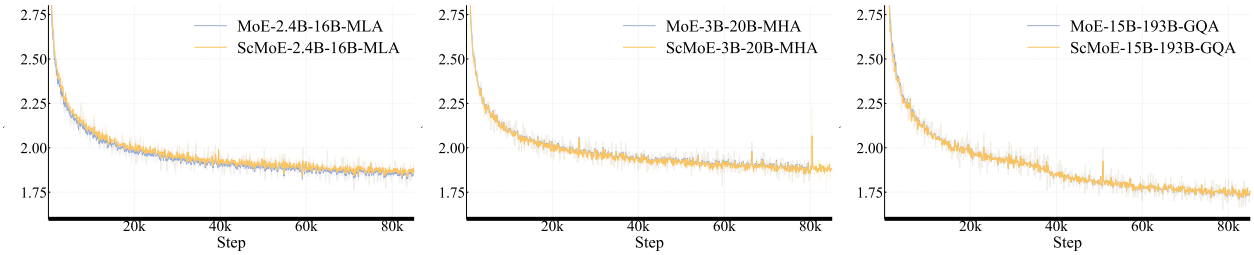
$$P_j = \frac{1}{T} \sum_{i \in \text{Group}_j} \sum_{t=1}^{T} R(x_t)_i, \tag{4}$$

$$f_j = \begin{cases} \dfrac{D}{K_e T} \displaystyle\sum_{t=1}^{T} \mathbb{I}(\text{token } t \text{ selects Group}_j), & \text{if } 1 \le j \le D, \\ \dfrac{1}{(K - K_e)T} \displaystyle\sum_{t=1}^{T} \mathbb{I}(\text{token } t \text{ selects zero-computation experts}), & \text{if } j = D+1, \end{cases} \tag{5}$$

where α is the balance factor, T is the number of tokens in a micro batch, and $\mathbb{I}$ denotes the indicator function. In the loss, we assign all zero-computation experts to an additional group and average the frequency in each group. By adjusting the coefficient of f_j, we ensure that the ratio of FFN experts to zero-computation experts would approach $\frac{K_e}{K-K_e}$ when the loss converges.

2.2 Shortcut-Connected MoE

Our initial architecture employs an interleaved topology of MoE and dense FFN blocks. This design has been extensively validated through empirical studies, demonstrating performance comparable to leading shared-expert



(a) (b) (c)

图4：对比在四种不同模型配置下的基线模型（不含 ScMoE）与其经 ScMoE 增强的对应版本的训练损失曲线。在所有实验中——(a) 2.4B-16B，采用 MLA；(b) 3B-20B，采用 MHA；(c) 15B-193B，采用 GQA——损失曲线几乎无法区分。这为 ScMoE 的优化提供了有力证据，表明其对质量是中性的，且其收益与模型规模以及所使用的具体注意力架构无关。

增量 [Wang et al., 2024a]，这个机制随着专家数量的增加而提高 softmax 路由器的概率分布的鲁棒性。值得注意的是，我们将零计算的专家从偏置更新中排除，因为它们的身份性质只需要一个全局约束，当所有 FFN 专家达到它们期望的令牌比例时，这一全局约束会自动满足。经验上，大批量大小和对 μ 的衰减调度可以提高预算控制的稳定性，而小批量大小可能需要降低更新频率。

在预训练期间，我们跟踪了激活专家的平均数量及其标准差（图 3b 和 3c）。结果表明，在经过大约 200 亿个 token 的处理后，所有层的平均专家数量收敛到预期值，波动幅度小于 1%。然而，标准差仍然处于相对较高的水平，表明模型在不同 token 之间分配了显著不同的计算资源。

有关动态路由的详细统计数据和案例研究，请参阅附录 A.1。

2.1.2 负载均衡控制

高效的 MoE 训练需要在 FFN 专家之间实现稳健的负载平衡。虽然 Eq. (2) 在语料级别强制保持平衡，但引入了一种设备级负载平衡损失 [DeepSeek-AI et al., 2025]，以进一步防止 EP 组之间的极端序列级不平衡。我们将采取必要措施来兼顾零计算专家。具体而言，假设所有 N FFN 专家被分成 D 个组，每组包含 $G = \frac{N}{D}$ 名专家，则损失可以表示为：

$$\mathcal{L}_{\text{LB}} = \alpha \sum_{j=1}^{D+1} f_j P_j, \tag{3}$$

$$P_j = \frac{1}{T} \sum_{i \in \text{Group}_j} \sum_{t=1}^{T} R(x_t)_i, \tag{4}$$

$$f_j = \begin{cases} \dfrac{D}{K_e T} \displaystyle\sum_{t=1}^{T} \mathbb{I}(\text{token } t \text{ selects Group}_j), & \text{if } 1 \le j \le D, \\ \dfrac{1}{(K - K_e)T} \displaystyle\sum_{t=1}^{T} \mathbb{I}(\text{token } t \text{ selects zero-computation experts}), & \text{if } j = D+1, \end{cases} \tag{5}$$

其中 α 是平衡因子，T 是一个微批次中的 token 数量，$\mathbb{I}$ 表示指示函数。在损失中，我们将所有零计算专家分配到一个额外的组，并对每个组中的频率进行平均。通过调整 f_j 的系数，我们确保在损失收敛时，FFN 专家与零计算专家的比例将趋近于 $\frac{K_e}{K-K_e}$。

2.2 带捷径连接的专家混合模型

我们初始的体系结构采用 MoE 与密集前馈网络（FFN）块的交错拓扑。这一设计经过大量实证研究的充分验证，表明其性能可与领先的共享专家模型相当。

models [Rajbhandari et al., 2022, Liu et al., 2024a]. However, the efficiency of large-scale MoE models is largely constrained by communication overhead. In the conventional execution paradigm, expert parallelism imposes a sequential workflow: an collective operation must first route tokens to their designated experts before computation can begin. This communication latency becomes a bottleneck, leading to device underutilization and limiting overall system throughput.

While shared-expert architectures attempt to mitigate this by overlapping communication with a single expert's computation, their efficiency is limited by the small computational window of that one expert. We overcome this limitation by employing the Shortcut-connected MoE (ScMoE) architecture [Cai et al., 2024]. ScMoE introduces a cross-layer shortcut that reorders the execution pipeline. This key innovation allows the dense FFN from the preceding block to execute in parallel with the dispatch/combine communication of the current MoE layer, creating a more substantial overlap window than shared-expert designs. Furthermore, the architecture design choice is verified by the following key findings.

First, ScMoE structure does not compromise model quality. As shown in Figure 4, the training loss curves of our architecture and the baseline without ScMoE are nearly identical, confirming this reordered execution does not impair model performance. Consistent results are observed across multiple settings, including a 2.4B-16B MoE model with MLA, a 3B-20B model with MHA [Vaswani et al., 2017], and 15B-193B models with GQA [Ainslie et al., 2023]. Importantly, these findings demonstrate that the stability and benefits of ScMoE are orthogonal to the choice of attention mechanism.

Second, the ScMoE architecture delivers substantial system-level efficiency gains for both training and inference.

For Large-Scale Training: The expanded overlap window allows the computation of the preceding block to be fully parallel with its dispatch and combine communication phases in the MoE layer, achieved by partitioning operations into fine-grained chunks along the token dimension.

For Efficient Inference: ScMoE enables a *Single Batch Overlap* pipeline, reducing the theoretical Time-Per-Output-Token (TPOT) by nearly 50% compared to leading models such as DeepSeek-V3. Moreover, it allows for the concurrent execution of distinct communication patterns: intra-node Tensor Parallelism communication (via NVLink) on the dense FFN can be fully overlapped with inter-node Expert Parallelism communication (via RDMA), thereby maximizing total network utilization.

In summary, ScMoE delivers substantial performance gains without sacrificing model quality. These efficiency gains are not achieved through trade-offs but are the direct outcome of a rigorously validated, quality-neutral architectural innovation.

2.3 Variance Alignment Design for Scalability

Architectural designs that excel at small scales may become suboptimal as models are scaled up, and vice versa, rendering initial design choices unreliable. Through extensive experimentation and theoretical analysis, we identify *variance misalignment* in specific modules as a key factor contributing to this discrepancy, which can lead to instability and degraded performance during scaling. To address this challenge, we propose variance alignment techniques for both MLA and MoE blocks.

2.3.1 Scale-Correction for MLA

LongCat-Flash employs a modified Multi-head Latent Attention (MLA) mechanism [Liu et al., 2024a], which incorporates scale-correction factors α_q and α_{kv} to address the variance imbalances inherent in asymmetric low-rank factorization. Our full mathematical formulation, which integrates these correction factors, is given as follows:

$$
\begin{aligned}
c_t^Q &= \boxed{\alpha_q}\, W^{DQ} h_t \in \mathbb{R}^{d_q}, & c_t^{KV} &= \boxed{\alpha_{kv}}\, W^{DKV} h_t \in \mathbb{R}^{d_{kv}}, & & \\
q_{t,i}^C &= W^{UQ} c_t^Q, & k_{t,i}^C &= W^{UK} c_t^{KV}, & v_{t,i} &= W^{UV} c_t^{KV}, \\
q_{t,i}^R &= \mathrm{RoPE}(W^{QR} c_t^Q), & k_t^R &= \mathrm{RoPE}(W^{KR} h_t), & & \\
q_{t,i} &= \left[\, q_{t,i}^C \,;\, q_{t,i}^R \,\right], & k_{t,i} &= \left[\, k_{t,i}^C \,;\, k_t^R \,\right], & & \\
o_{t,i} &= \mathrm{Attention}(q_{t,i},\, k_{1:t,i},\, v_{1:t,i}), & u_t &= W^O \left[\, o_{t,1} \,;\, o_{t,2} \,;\, \ldots \,;\, o_{t,n_h} \,\right], & &
\end{aligned}
\tag{6}
$$

where $h_t \in \mathbb{R}^{d_{\text{model}}}$ is the input hidden state, and the final query and key for each head i are formed by concatenating a content part (C) and a rotary part (R).

模型 [Rajbhandari 等人，2022，Liu 等人，2024a]。然而，大规模 MoE 模型的效率在很大程度上受通信开销的制约。在传统的执行范式中，专家并行性带来一个顺序化的工作流程：在计算开始之前，必须先通过一个聚集操作将令牌路由到它们指定的专家。这段通信时延成为瓶颈，导致设备利用率下降并限制整个系统的吞吐量。

尽管共享专家架构试图通过将通信与单个专家的计算重叠来缓解这一点，但它们的效率受到该单个专家的较小计算窗口的限制。我们通过采用 Shortcut-connected MoE (ScMoE) 架构 [Cai 等人，2024] 来克服这一限制。ScMoE 引入一个跨层捷径，重新排序执行流水线。这一关键创新使来自前一块的密集前馈网络（FFN）能够与当前 MoE 层的分发/合并通信并行执行，从而比共享专家设计创造出更大重叠窗口。此外，以下关键发现验证了这一架构设计选择。

首先，ScMoE 结构并不降低模型质量。正如图4所示，我们的架构与不使用 ScMoE 的基线在训练损失曲线方面几乎完全相同，这证实了这种重新排序的执行不会削弱模型性能。在多种设置下也观察到一致的结果，包括一个带 MLA 的 2.4B-16B MoE 模型、一个带 MHA [Vaswani et al., 2017] 的 3B-20B 模型，以及一个带 GQA [Ainslie et al., 2023] 的 15B-193B 模型。重要的是，这些发现表明 ScMoE 的稳定性和收益与注意力机制的选择无关。

其次，ScMoE 架构在训练和推理两个方面都带来显著的系统级效率提升。

用于大规模训练：扩展的重叠窗口使前一个块的计算能够与其在 MoE 层中的派发和合并通信阶段完全并行，这通过将操作沿令牌维度划分为细粒度的分块来实现。

为了高效推理：ScMoE 启用一个 *Single Batch Overlap* 流水线，相较于诸如 DeepSeek-V3 这样的领先模型，理论的 Time-Per-Output-Token (TPOT) 将降低近 50%。此外，它还允许并发执行不同的通信模式：在同一节点内的张量并行通信（通过 NVLink）与跨节点的专家并行通信（通过 RDMA）可以完全重叠，从而最大化整个网络的利用率。

总之，ScMoE 在不牺牲模型质量的前提下，带来显著的性能提升。这些效率提升不是通过权衡实现的，而是经过严格验证、质量中立的架构创新直接带来的结果。

2.3 用于可扩展性的方差对齐设计

在小尺度上表现出色的架构设计，随着模型规模的扩大，可能会变得次优，反之亦然，从而使初始设计选择不再可靠。通过大量的实验和理论分析，我们在某些模块中发现*variance misalignment*是导致这一差异的关键因素之一，这可能在扩展过程中引发不稳定并降低性能。为应对这一挑战，我们提出针对 MLA 与 MoE 块的方差对齐技术。

2.3.1 MLA 的尺度校正

LongCat-Flash 采用经修改的多头潜在注意力（MLA）机制 [Liu et al., 2024a]，该机制将尺度校正因子 α_q 和 α_{kv} 纳入其中，以解决不对称低秩分解固有的方差不平衡。我们将这些校正因子整合在内的完整数学表述如下：

$$
\begin{aligned}
c_t^Q &= \boxed{\alpha_q} W^{DQ} h_t \in \mathbb{R}^{d_q}, & c_t^{KV} &= \boxed{\alpha_{kv}} W^{DKV} h_t \in \mathbb{R}^{d_{kv}}, & & \\
q_{t,i}^C &= W^{UQ} c_t^Q, & k_{t,i}^C &= W^{UK} c_t^{KV}, & v_{t,i} &= W^{UV} c_t^{KV}, \\
q_{t,i}^R &= \mathrm{RoPE}(W^{QR} c_t^Q), & k_t^R &= \mathrm{RoPE}(W^{KR} h_t), & & \\
q_{t,i} &= \left[q_{t,i}^C ; q_{t,i}^R \right], & k_{t,i} &= \left[k_{t,i}^C ; k_t^R \right], & & \\
o_{t,i} &= \mathrm{Attention}(q_{t,i}, k_{1:t,i}, v_{1:t,i}), & u_t &= W^O \left[o_{t,1} ; o_{t,2} ; \ldots ; o_{t,n_h} \right], & &
\end{aligned}
\tag{6}
$$

其中 $h_t \in \mathbb{R}^{d_{\text{model}}}$ 是输入隐藏状态，对于每个头而言，最终的查询和键 i 是通过将内容部分 (C) 与旋转部分 (R) 拼接而形成。

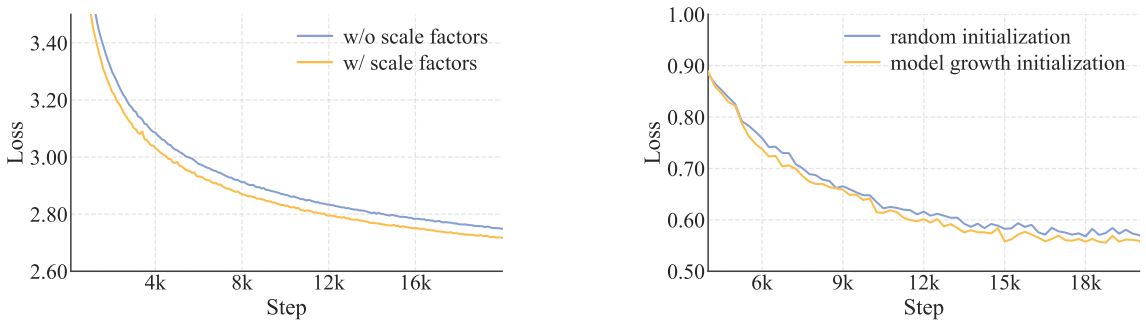


Figure 5: **(a)** Incorporating the scale-correction factor on MLA showing improved convergence (lower loss) on a 1B activated MOE model. **(b)** Validataion loss curve of a 6B activated MoE model in model growth experiments.

The introduction of α_q and α_{kv} address a fundamental variance mismatch among query/key vector components. At initialization, their variances are proportional to their source dimensions: $\sigma^2(q_t^C), \sigma^2(q_t^R) \propto d_q$ and $\sigma^2(k_t^C) \propto d_{kv}$. In contrast, the rotary key component k_t^R has a variance proportional to the full model dimension: $\sigma^2(k_t^R) \propto d_{\text{model}}$. This dimensional disparity causes unstable attention scores at initialization when d_q, d_{kv}, and d_{model} are varied, resulting in degraded and unpredictable performance during model scaling.

Our solution is to rescale the low-rank path components to align their final variance with a reference scale, and we use the full model dimension as a reference. This is achieved by defining the scaling factors as:

$$\alpha_q = \left(\frac{d_{\text{model}}}{d_q}\right)^{0.5} \quad \text{and} \quad \alpha_{kv} = \left(\frac{d_{\text{model}}}{d_{kv}}\right)^{0.5}. \tag{7}$$

This scale-invariant correction neutralizes the variance mismatch, ensuring they are well-conditioned for the attention computation. Our experiments reveal that this method improves the model performance, as shown in Figure 5a.

2.3.2 Variance Compensation for Experts Initialization

LongCat-Flash adopts the fine-grained expert strategy from DeepSeek-MoE [Liu et al., 2024a], which segments each expert into m finer-grained ones to enhance combinatorial flexibility and knowledge specialization. However, we observe that the performance of this design is sensitive to other architectural choices (e.g., expert numbers, top-k, m). To address this, we propose a variance compensation mechanism that counteracts the initialization variance reduction caused by expert segmentation. The mechanism applies a scaling factor γ to the aggregated output of the experts, formulated as:

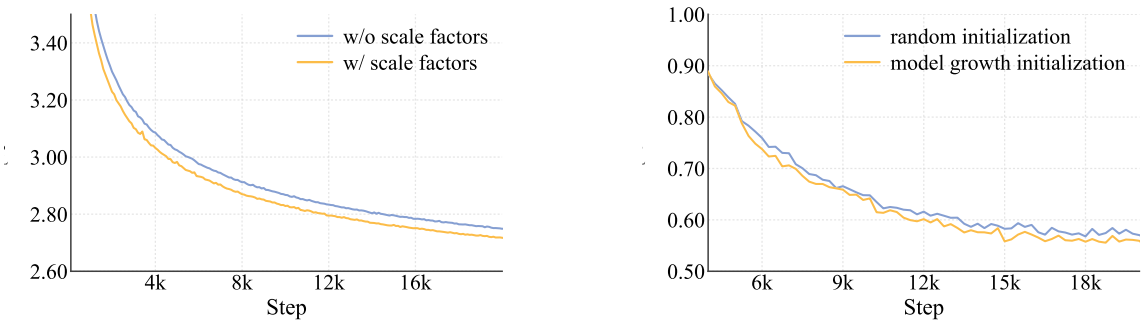
$$\text{MoE}(x_t) = \gamma \left(\sum_{i=1}^{mN} g_i \cdot E_i(x_t) \right), \tag{8}$$

where g_i is the router output over mN fine-grained experts and N represents the total number of experts before segmentation.

The scaling factor γ in Eq. (8) is derived by quantifying two primary sources of variance reduction:

1. Gating Dilution: Decomposing each original N experts into m finer-grained experts expands the total expert counts to mN. This expansion forces the softmax gate to distribute its probability mass across a larger expert pool, proportionally reducing the magnitude of individual gating values g_i. Consequently, the output variance is reduced approximately by a factor of m.
2. Dimensional Reduction: The intermediate hidden dimension of each fine-grained expert ($d_{\text{expert\_inter}}$) is reduced by a factor of m. Assuming uniform parameter initialization, the output variance of a single expert also decreases by a factor of m.

To preserve the MoE layer's output variance at initialization (matching the pre-segmentation baseline), γ must compensate for both effects. The combined variance compensation factor is thus $\gamma = \sqrt{m \cdot m} = m$.



(a) (b)

图 5：（a）在 MLA 上引入尺度修正因子，显示在一个 1B 已激活的 MOE 模型上具有更好的收敛性（损失更低）。/（b）在模型增长实验中的一个 6B 已激活的 MoE 模型的验证损失曲线。

引入 α_q 和 α_{kv} 解决了查询向量与键向量分量之间的一个基本方差不匹配。在初始化时，它们的方差与它们的源维度成正比：$\sigma^2(q_t^C), \sigma^2(q_t^R) \propto d_q$ 和 $\sigma^2(k_t^C) \propto d_{kv}$。相较之下，旋转键分量 k_t^R 的方差与整个模型维度成正比：$\sigma^2(k_t^R) \propto d_{\text{model}}$。这种维度差异会在初始化时导致不稳定的注意力分数，当 d_q、d_{kv} 与 d_{model} 发生变化时，进而在模型缩放过程中造成性能下降且不可预测。

我们的解决方案是对低秩路径分量进行重新缩放，使它们的最终方差与参考尺度对齐，我们以完整的模型维度作为参考。这是通过将缩放因子定义为：

$$\alpha_q = \left(\frac{d_{\text{model}}}{d_q}\right)^{0.5} \quad \text{and} \quad \alpha_{kv} = \left(\frac{d_{\text{model}}}{d_{kv}}\right)^{0.5}. \tag{7}$$

这种尺度不变的修正消除了方差不匹配，确保它们在进行注意力计算时具备良好的数值条件。我们的实验表明，该方法能够提升模型的性能，如图5a所示。

2.3.2 对专家初始化的方差补偿

LongCat-Flash 采用来自 DeepSeek-MoE [Liu et al., 2024a] 的细粒度专家策略，该策略将每个专家划分为 m 个更细粒度的子专家，以提升组合灵活性和知识专业化。 然而，我们发现该设计的性能对其他架构选项（例如专家数量、top-k、m）较为敏感。 为了解决这个问题，我们提出了一种方差补偿机制，用以抵消由专家分割引起的初始化方差降低。 该机制对专家聚合输出应用一个缩放系数，其公式定义如下：

$$\text{MoE}(x_t) = \gamma \left(\sum_{i=1}^{mN} g_i \cdot E_i(x_t) \right), \tag{8}$$

其中 g_i 是对 mN 个细粒度专家的路由输出，N 表示分割前的专家总数。

式(8)中的缩放因子 γ 是通过量化两种主要的方差降低来源来推导的：

1. 门控稀释：将每个原始的 N 专家分解为 m 个更细粒度的专家，使总专家数扩展到 mN。这一扩展迫使 softmax 门控将概率质量分布到更大的专家池中，从而按比例降低单个门控值 g_i 的幅度。因此，输出方差大约降低了 m 倍。

2. 维度降维：每个细粒度专家的中间隐藏维度($d_{\text{expert\_inter}}$)被降低了一个因子m。在假设参数初始化均匀的前提下，单个专家的输出方差也降低一个因子m。

为了在初始化时保持 MoE 层的输出方差与分段前的基线保持一致，γ 必须同时对这两种效应进行补偿。因此，合并的方差补偿系数是 $\gamma = \sqrt{m \cdot m} = m$。

2.4 Model Information

Tokenizer LongCat-Flash employs byte-pair encoding (BPE) [Shibata et al., 1999, Sennrich et al., 2015] for tokenization. Our tokenizer is trained on a comprehensive multilingual corpus spanning web pages, books, source code, etc, ensuring robust cross-domain performance. While inheriting GPT-4's pre-tokenization framework, we introduce the following modifications: (1) Enhanced CJK character segmentation for improved Chinese text handling, and (2) Independent digit tokenization to boost mathematical capabilities. The vocabulary size is optimized at 131,072 tokens, striking an effective balance between computational efficiency and linguistic coverage.

Multi-Token Prediction To enhance inference efficiency, we integrate Multi-Token Prediction (MTP) [Gloeckle et al., 2024, DeepSeek-AI et al., 2025] as an auxiliary training objective. For optimal inference performance, we employ a single dense layer rather than a MoE layer as the MTP head. Empirical observations reveal rapid convergence of MTP loss, prompting us to strategically introduce MTP training in the middle of the training pipeline to balance model performance with prediction accuracy. The MTP head achieves >90% acceptance rate in evaluations (Table 5).

Model Configurations LongCat-Flash consists of 28 layers (excluding the MTP layer) with a 6144-dimensional hidden state. Each MLA block uses 64 attention heads with per-head dimension 128 for balanced performance-efficiency tradeoff. Following DeepSeek-V3 [Liu et al., 2024a], we set the KV compression dimension to 512, and the query compression dimension to 1536. The FFNs in the dense path employ 12288 intermediate dimensions, while each FFN expert uses 2048 dimensions. The scaling factors in MLA blocks and FFN blocks follow the methodology in Section 2.3.1. Each layer contains 512 FFN experts and 256 zero-computation experts, with exactly 12 experts activated per token (selected from both types). LongCat-Flash has 560B total parameters, activating between 18.6B and 31.3B parameters per token depending on context, with an average activation of approximately 27B parameters.

3 Pre-Training

The pre-training of LongCat-Flash follows a three-stage curriculum: (1) We train the model on approximately 20 trillion tokens with 8192 sequence length to establish a robust base model. (2) Reasoning and coding capabilities are further enhanced using trillions of data. (3) The context length is extended to 128k through training on long context corpora. Each stage implements tailored data strategies accompanied by rigorous decontamination procedures to prevent test set leakage.

To optimize scalability, we introduce hyperparameter transfer and model growth strategies, significantly improving performance as model size increases. Given the inherent instability challenges in large-scale training, we identify and implement multiple effective techniques to enhance training stability.

3.1 Training Strategy

3.1.1 Hyperparameter Transfer

LongCat-Flash employs a hyperparameter transfer strategy based on width scaling [Everett et al., 2024] to efficiently train large-scale models. The methodology involves: (1) identifying optimal hyperparameters on a smaller proxy model, and (2) transferring these configurations to the target model through theoretically-motivated scaling rules.

The transfer mechanism centers on the width scaling factor $s = n_{\text{target}}/n_{\text{proxy}}$, where n is the model's hidden dimension. We specifically adopt the "Adam LR Full Align" rules for Standard Parameterization. These rules specify how to adapt the proxy model's optimal initialization variance (σ^2) and learning rate (η) for the target architecture. The practical transfer rules are summarized in Table 1.

Table 1: Practical hyperparameter transfer rules and their underlying scaling exponents, derived from the Adam LR Full Align principle for Standard Parameterization [Everett et al., 2024]. Here, s is the width scaling factor $n_{\text{target}}/n_{\text{proxy}}$.

Layer & Parameter	Target Model Setting
Embedding (Init Var, σ^2)	$\sigma^2_{\text{target}} = \sigma^2_{\text{proxy}}$
Embedding (Learning Rate, η)	$\eta_{\text{target}} = \eta_{\text{proxy}}$
Hidden/Unembedding (Init Var, σ^2)	$\sigma^2_{\text{target}} = \sigma^2_{\text{proxy}}/s$
Hidden/Unembedding (Learning Rate, η)	$\eta_{\text{target}} = \eta_{\text{proxy}}/s$

Following this methodology, our training involves the following steps:

2.4 模型信息

Tokenizer LongCat-Flash 使用字节对编码（BPE）[Shibata 等, 1999, Sennrich 等, 2015] 进行分词。我们的分词器是在覆盖网页、书籍、源代码等的全面多语言语料库上进行训练，确保跨领域的稳健表现。在继承 GPT-4 的预分词框架的同时，我们引入以下修改：（1）提升对 CJK 字符的分词以改善中文文本处理能力；（2）独立的数字分词以提升数学处理能力。词汇表大小优化为 131,072 个词元，在计算效率与语言覆盖之间取得有效平衡。

为提升推理效率，我们将 Multi-Token Prediction (MTP) [Gloeckle 等人，2024，DeepSeek-AI 等人，2025] 作为一个辅助训练目标整合进来。为了获得最佳推理性能，我们使用一个单一的全连接层作为 MTP 头，而不是 MoE 层。经验观察表明 MTP 损失迅速收敛，促使我们在训练流程的中段有策略地引入 MTP 训练，以在模型性能与预测准确性之间取得平衡。MTP 头在评估中实现了 >90% 的接受率（表 5）。

模型配置 LongCat-Flash 由 28 层组成（不包含 MTP 层），隐藏状态为 6144 维。 每个 MLA 模块使用 64 个注意力头，单头维度为 128，以实现性能与效率之间的平衡。 仿照 DeepSeek-V3 [Liu et al., 2024a]，我们将 KV 压缩维度设为 512，查询压缩维度设为 1536。 密集路径中的前馈网络（FFN）使用 12288 的中间维度，而每个 FFN 专家使用 2048 维。MLA 块和 FFN 块中的缩放因子遵循第 2.3.1 节的方法。 每层包含 512 个 FFN 专家和 256 个零计算专家，每个标记恰好激活 12 个专家（从两种类型中选取）。LongCat-Flash 总参数量为 560B，根据上下文不同，每个标记激活的参数在 18.6B 到 31.3B 之间，平均激活约 27B 参数。

3 预训练

LongCat-Flash 的预训练遵循一个三阶段课程：(1) 我们在大约 20万亿个令牌、序列长度为 8192 的条件下对模型进行训练，以建立一个稳健的基础模型。(2) 通过使用数以万亿计的数据，进一步提升推理和编码能力。(3) 通过在长上下文语料库上的训练，将上下文长度扩展至 128k。 每个阶段都实施定制的数据策略，并伴随严格的去污染程序，以防止测试集泄漏。

为优化可扩展性，我们引入超参数迁移和模型增长策略，随着模型规模的增大，性能显著提升。鉴于大规模训练中的固有不稳定性挑战，我们识别并实施多种有效技术以提升训练的稳定性。

3.1 训练策略

3.1.1 超参数迁移

LongCat-Flash 采用基于宽度缩放的超参数转移策略 [Everett 等人，2024] 以高效训练大规模模型。该方法包括：（1）在较小的代理模型上识别出最佳超参数；（2）通过理论驱动的缩放规则将这些配置转移到目标模型。

迁移机制聚焦于宽度缩放因子 $s = n_{\text{target}}/n_{\text{proxy}}$，其中 n 是模型的隐藏维度。我们专门采用用于标准参数化的“Adam LR Full Align”规则。这些规则规定如何将代理模型的最优初始化方差 (σ^2) 和学习率 (η) 适配到目标架构。实际的迁移规则在表1中总结。

表 1：基于标准参数化的 Adam LR 全对齐原理推导出的实用超参数传递规则及其底层缩放指数 [Everett 等人，2024]。其中，s 是宽度缩放因子 $n_{\text{target}}/n_{\text{proxy}}$。

Layer & Parameter	**Target Model Setting**
Embedding (Init Var, σ^2)	$\sigma^2_{\text{target}} = \sigma^2_{\text{proxy}}$
Embedding (Learning Rate, η)	$\eta_{\text{target}} = \eta_{\text{proxy}}$
Hidden/Unembedding (Init Var, σ^2)	$\sigma^2_{\text{target}} = \sigma^2_{\text{proxy}}/s$
Hidden/Unembedding (Learning Rate, η)	$\eta_{\text{target}} = \eta_{\text{proxy}}/s$

遵循这种方法，我们的训练包括以下步骤： 翻译文本：

1. We set the width scaling factor s to 8 based on a trade-off analysis between computational efficiency and transfer performance. The proxy model is configured with a width of 768.
2. We then perform a comprehensive hyperparameter search on the proxy model to identify the optimal layer-specific initializaton variances (σ^2_{proxy}) and learning rates (η_{proxy}).
3. The optimal hyperparameters from the proxy model were transferred to the target model following the rules detailed in Table 1. All other architectural attributes (depth, sparsity, and batch size) remain invariant during this transfer process.

We conducted comprehensive experiments to validate the effectiveness of this approach. The results demonstrate that this method significantly reduces computational costs when identifying optimal hyperparameters (initialization variance and learning rate) for large-scale model training, while establishing a robust, theoretically grounded framework for model scaling.

3.1.2 Model Growth Initialization

LongCat-Flash employs model growth as its initialization strategy, starting from a half-scale model pre-trained on tens of billions of tokens. Among existing model growth methods [Chen et al., 2015, Du et al., 2024, Wang et al., 2023a, Shen et al., 2022, Wang et al., 2023b, Gong et al., 2019], we adopt the layer stacking technique [Du et al., 2024, Kim et al., 2023] to expand parameters and enhance performance. Disregarding the embedding and unembedding processes temporarily, the whole procedure is formulated as:

$$
\begin{aligned}
L_{\text{small}} &= l_1 \circ l_2 \circ \cdots \circ l_n \\
L_{\text{target}} &= \underbrace{L_{\text{small}} \circ L_{\text{small}} \circ \cdots \circ L_{\text{small}}}_{r}
\end{aligned}
$$

where l_i denotes the transformation of the ith layer in the model, r denotes the expansion rate, L_{small} denotes the small model's transformation from token embeddings to final hidden states, and L_{target} represents the transformation of the target (large) model constructed by stacking r copies of the small model. We use $r = 2$ for our architecture.

Through extensive experiments, we consistently observed that models initialized via model growth exhibit a characteristic loss trajectory: an initial increase followed by accelerated convergence, ultimately outperforming randomly initialized baselines. Figure 5b presents a representative case from our 6B activated model experiments, demonstrating the advantage of model growth initialization.

We conjecture that this improvement arises from two synergistic factors: (1) the faster convergence of smaller models likely provides higher-quality parameter initializations for scaled training, and (2) growth operations potentially serve as implicit regularization against parameter collapse. Experimental evidence further suggests that over-optimizing predecessor models may negatively impact token efficiency in target models, indicating the need for judicious growth timing.

For LongCat-Flash initialization, we first train a 14-layer model with identical architecture to the target model, using random initialization on the initial data segment. The trained model is then stacked to create a 28-layer checkpoint, preserving all training states including sample counters and learning rate schedules from the predecessor.

3.1.3 Training Stability

We enhance the training stability of LongCat-Flash from three perspectives: router stability, activation stability, and optimizer stability.

Router Stability A fundamental challenge in training MoE models is router stability, which stems from the tension between two competing gradients:

- The language modeling (LM) loss, driving *expert specialization* (assigning tokens to the most suitable experts),
- The auxiliary load balancing (LB) loss, enforcing *routing uniformity* (distributing tokens evenly across experts).

When the LB gradient dominates, router parameters for all experts converge toward similarity, leading to uniform routing decisions regardless of input tokens. This nullifies the benefits of conditional computation and severely degrades model performance.

To diagnose and control this behavior, we propose a monitoring framework with two key metrics:

1. 我们基于计算效率与迁移性能之间的权衡分析，将宽度缩放因子 s 设置为 8。代理模型的宽度设为 768。

2. 接着我们在代理模型上进行全面的超参数搜索，以确定最优的逐层初始化方差（σ^2_{proxy}）和学习率（η_{proxy}）。
3. 根据表1所述的规则，将代理模型的最优超参数转移到目标模型。在此传输过程中，其他所有架构属性（深度、稀疏性和批大小）保持不变。

我们进行了全面的实验来验证该方法的有效性。结果表明，在为大规模模型训练确定最优超参数（初始化方差和学习率）时，该方法能显著降低计算成本，同时建立了一个稳健、理论基础扎实的模型扩展框架。

3.1.2 模型增长初始化

LongCat-Flash 采用模型增长作为初始化策略，从一个在数十亿个令牌上预训练的半规模模型开始。在现有的模型增长方法 [Chen et al., 2015, Du et al., 2024, Wang et al., 2023a, Shen et al., 2022, Wang et al., 2023b, Gong et al., 2019] 中，我们采用层叠技术 [Du et al., 2024, Kim et al., 2023] 来扩展参数并提升性能。暂时忽略嵌入与去嵌入过程，整个过程被表述为：

$$
\begin{aligned}
L_{\text{small}} &= l_1 \circ l_2 \circ \cdots \circ l_n \\
L_{\text{target}} &= \underbrace{L_{\text{small}} \circ L_{\text{small}} \circ \cdots \circ L_{\text{small}}}_{r}
\end{aligned}
$$

其中 l_i 表示模型中第 i 层的变换，r 表示扩展率，L_{small} 表示从令牌嵌入到最终隐藏状态的小模型的变换，L_{target} 表示通过堆叠 r 个小模型构建的目标（大型）模型的变换。我们在架构中使用 $r = 2$。

通过大量实验，我们始终观察到通过模型增长初始化的模型呈现出一种特征性的损失轨迹：初始上升，随后收敛加速，最终优于随机初始化的基线。图5b 展示了我们在 6B 激活模型实验中的一个代表性案例，展示了模型增长初始化的优势。

我们推测这一改进来自两个协同因素：（1）较小模型的更快收敛可能为规模化训练提供更高质量的参数初始化；以及（2）增长操作可能作为对参数崩溃的隐式正则化。实验证据进一步表明，对前身模型进行过度优化可能对目标模型的词元效率产生负面影响，表明需要谨慎的增长时机。

对于 LongCat-Flash 的初始化，我们首先训练一个与目标模型具有相同体系结构的 14 层模型，在初始数据段进行随机初始化。训练好的模型随后堆叠成一个 28 层的检查点，保留所有训练状态，包括样本计数器和来自前代模型的学习率调度。

3.1.3 训练稳定性

我们从三个方面增强 LongCat-Flash 的训练稳定性：路由稳定性、激活稳定性和优化器稳定性。

路由稳定性 在训练 MoE 模型时，一个根本性的挑战是路由稳定性，这源于两个互相竞争的梯度之间的张力：

- 语言模型（LM）损失，驱动 *expert specialization* (assigning tokens to the most suitable experts),
- 辅助负载均衡（LB）损失，强制 *routing uniformity* (distributing tokens evenly across experts)。

当 LB 梯度占主导地位时，所有专家的路由参数趋于相似，导致无论输入令牌如何，路由决策都趋于一致。这会抵消条件计算的优势，并严重降低模型的性能。

为了诊断和控制这种行为，我们提出一个包含两个关键指标的监控框架：

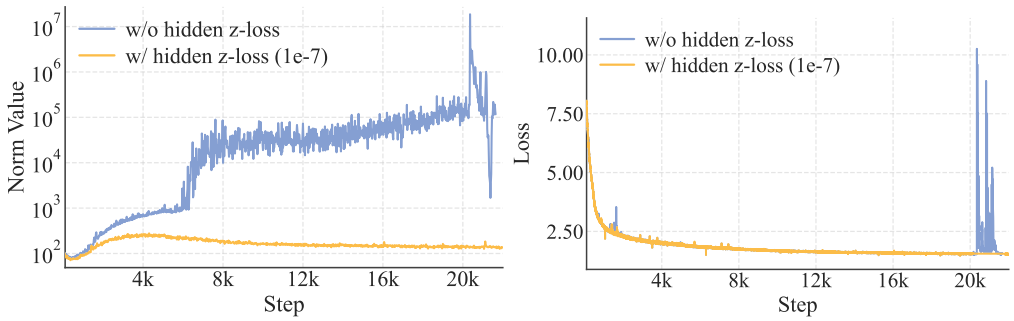


Figure 6: L2 norm of last layer's hidden states and the training loss for a small model with suboptimal hyper-parameters. The introduction of a negligible-coefficient hidden z-loss stabilizes the norm curve without degrading training loss.

- Router Weight Similarity: Measure the average pairwise cosine similarity between expert weight vectors $\{w_i\}$. A high similarity is a direct indicator that the load balancing loss is excessively dominant.
- Gradient Norm Ratio (R_g): Quantify the relative influence of the two losses on the batch-averaged expert probability vector $\vec{P}$:

$$R_g = \frac{\|\alpha \nabla_{\vec{P}} \mathcal{L}_{\text{LB}}\|_2}{\|\nabla_{\vec{P}} \mathcal{L}_{\text{LM}}\|_2}, \tag{9}$$

  where $\mathcal{L}_{\text{LB}}$ is the load balancing loss computed without the coefficient α.

Guided by this framework, we establish a practical guideline for setting the hyperparameter α. The principle is to ensure the load balancing term acts as a regularizer without overwhelming the LM loss. We therefore recommend choosing a coefficient that keeps the R_g below a small threshold (e.g., $R_g < 0.1$).

Activation Stability via Hidden z-loss Inspired by the router z-loss [Zoph et al., 2022], we design hidden z-loss to circumvent the widespread occurrence of massive activation [Sun et al., 2024] during LLM training. Through empirical observations, we find that such massive activations correlate with severe loss spikes during training, which are associated with optimization instability and potential performance degradation. Hidden z-loss is mainly used to suppress elements with extremely large magnitudes:

$$\mathcal{L}_{\text{Z}} = \frac{\lambda}{T} \sum_{t=1}^{T} (\log \sum_{i=1}^{|x_t|} \exp(\text{abs}(x_t^i)))^2, \tag{10}$$

where λ is the coefficient to weight this loss, $|x_t|$ is the hidden size and $\text{abs}(*)$ denotes absolute value function. As depicted in Figure 6, we found that a very small loss coefficient can significantly suppress the massive activation phenomenon without compromising training loss, thus reducing the risk of numerical errors during BF16 training.

On the Practical Configuration of Adam's Epsilon As model scale increases, the epsilon (ε) parameter in the Adam optimizer, traditionally treated as a minor constant for numerical stability, emerges as a critical hyperparameter. OLMo et al. [2024] demonstrated that setting it to 1e-8 yields superior results compared to the default value of 1e-5. This heightened sensitivity primarily stems from two factors: (1) large-scale models typically employ smaller parameter initializations, and (2) they utilize substantially larger batch sizes during training. When using default ε values, the parameter's magnitude may become comparable to or even exceed the typical scale of gradient second moments, thereby disrupting the optimizer's adaptive mechanism.

As illustrated in Figure 7, our empirical analysis tracking the Root Mean Square (RMS) norm [Zhang and Sennrich, 2019] of gradients reveals two key findings: (1) Threshold effect: Significant performance degradation occurs when ε approaches the observed gradient RMS norm; (2) Lower bound stability: Once ε is reduced below this critical threshold, further decreases have a negligible impact on model performance. Consequently, we recommend setting ε to a small value (several orders of magnitude smaller than the expected gradient RMS norm). In LongCat-Flash, we adopt ε=1e-16, a configuration that ensures numerical stability while preserving the optimizer's adaptive properties.

3.2 General Pre-Training

We first conduct a general pre-training stage to ensure overall model ability. A multi-phase pipeline is designed to ensure data quality and diversity. The main phases include:

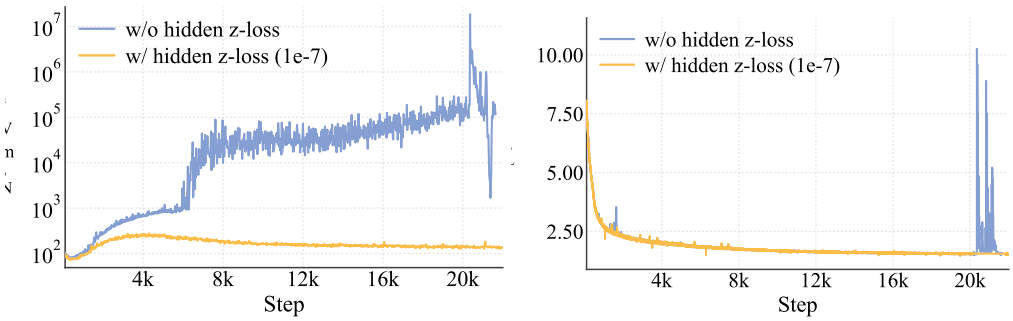


图6：一个具有次优超参数的小模型的最后一层隐藏状态的 L2 范数以及该模型的训练损失。引入一个系数极小的隐藏 z 损失可以稳定范数曲线，同时不降低训练损失。

- 路由权重相似性：测量专家权重向量 $\{w_i\}$ 之间的平均两两余弦相似度。高相似度是负载均衡损失过度占主导地位的直接指标。
- 梯度范数比（R_g）：量化两个损失对批量平均的专家概率向量$\bar{P}$的相对影响：

$$R_g = \frac{\|\alpha \nabla_{\bar{P}} \mathcal{L}_{\text{LB}}\|_2}{\|\nabla_{\bar{P}} \mathcal{L}_{\text{LM}}\|_2}, \tag{9}$$

其中 $\mathcal{L}_{\text{LB}}$ 是在不包含系数 α 的情况下计算的负载均衡损失。

在这个框架的指导下，我们为设置超参数 α 建立了一条实用准则。其原则是确保负载平衡项作为正则化项发挥作用，同时不过度压制语言模型损失。因此，我们建议选择一个系数，使 R_g 保持在一个较小的阈值以下（例如 $R_g < 0.1$）。

通过隐藏 z-loss 实现激活稳定性：受到 router z-loss [Zoph 等人，2022] 启发，我们设计隐藏 z-loss，以规避在大型语言模型（LLM）训练过程中普遍出现的巨量激活 [Sun 等人，2024]。通过经验观察，我们发现此类巨量激活与训练过程中的严重损失峰值相关，这些峰值与优化不稳定性以及潜在的性能下降相关。隐藏 z-loss 主要用于抑制幅度极大的元素：

$$\mathcal{L}_{\text{Z}} = \frac{\lambda}{T} \sum_{t=1}^{T} (\log \sum_{i=1}^{|x_t|} \exp(\text{abs}(x_t^i)))^2, \tag{10}$$

其中 λ 是对该损失进行加权的系数，$|x_t|$ 是隐藏维度，abs(∗) 表示绝对值函数。如图6所示，我们发现一个非常小的损失系数可以显著抑制大规模的激活现象，而不会影响训练损失，从而降低在 BF16 训练过程中的数值错误风险。

关于 Adam 的 Epsilon 的实际配置：随着模型规模的增大，Adam 优化器中的 epsilon（ε）参数，传统上被视为用于数值稳定性的次要常数，现已成为一个关键的超参数。OLMo 等人 [2024] 证明将其设置为 1e-8 相比默认值 1e-5 能获得更优的结果。这种敏感性主要源于两个因素：（1）大规模模型通常采用更小的参数初始化；（2）在训练中它们使用的批量大小显著增大。当使用默认的 ε 值时，参数的量级可能与梯度二阶矩的典型量级相当，甚至超过，从而破坏优化器的自适应机制。

如图7所示，我们对梯度的均方根（RMS）范数[Zhang and Sennrich, 2019]的经验分析揭示了两个关键发现：1) 阈值效应：当ε接近观测到的梯度 RMS 范数时，性能显著下降；2) 下限稳定性：一旦ε降低到该临界阈值以下，进一步下降对模型性能的影响可忽略。因此，我们建议将ε设置为一个较小的值（比预期的梯度 RMS 范数小几数量级）。在 LongCat-Flash 中，我们采用ε=1e-16，这是一种在确保数值稳定性的同时保持优化器自适应属性的配置。

3.2 通用预训练

我们首先进行一个通用的预训练阶段，以确保模型的整体能力。设计了一个多阶段的流水线，以确保数据质量和多样性。主要阶段包括：

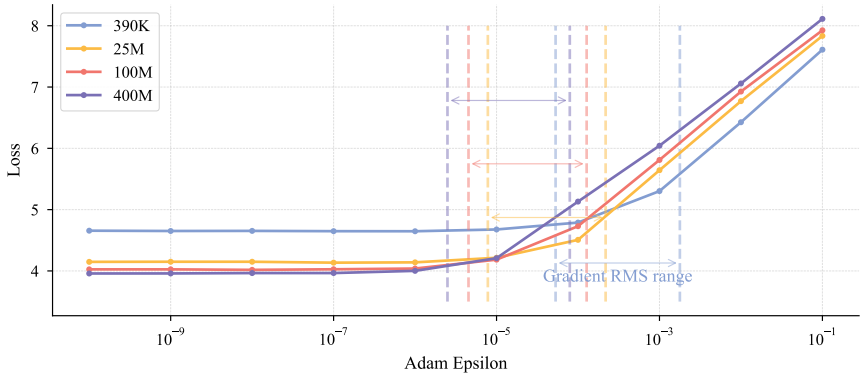


Figure 7: Exploring the impact of the Root Mean Square (RMS) norm of gradients and epsilon on loss across different model sizes. The "Gradient RMS range" denotes the span between the maximum and minimum gradient RMS values for different weights in the model. As the model size increases (ranging from 390K to 400M parameters), the gradient RMS becomes smaller. When epsilon approaches the range of the gradient RMS, a rapid deterioration in loss is observed.

Content Extraction We utilize a customized version of trafilatura [Barbaresi, 2021] for general web content and a dedicated process for STEM material to correctly parse complex elements like formulas, code, and tables.

Quality Filtering A two-step filtering approach is applied. An initial classifier removes clearly low-quality documents, followed by finer-grained screening based on metrics like fluency and content completeness.

Deduplication We apply an efficient MinHash implementation for large-scale deduplication, supplemented by a strategy to identify and handle repetitive web templates for more accurate document-level deduplication.

The final data mixture process adopts a two-stage schedule, progressively increasing the proportion of high-quality reasoning data (e.g., STEM and code).

- Stage 1: For general-purpose data, we employ an instance-level data mixing strategy that balances data quality and diversity described in SampleMix [Xi et al., 2025], where we compute an initial sampling distribution using quality and diversity scores, and further adjust the tendency of the distribution based on fine-grained domain and writing style labels. Redundant low-value domains (e.g., advertisement, sports, hiring) are downsampled, while reasoning-rich domains (e.g., science) are upsampled.
- Stage 2: We prioritize reasoning-intensive domains in this phase, with STEM and code comprising 70% of the final mixture. Preliminary experiments showed that abrupt reductions in general-domain data temporarily degrade model capabilities. Thus, we implement gradual code proportion increases, guided by continuous perplexity monitoring on external validation sets to ensure smooth transitions without compromising general performance.

3.3 Reasoning and Coding Enhancement

To further enhance the model's reasoning and coding capabilities and establish a robust base model with substantial potential for subsequent post-training, we conduct a mid-training stage utilizing high-quality relevant data generated through a combination of pretraining data retrieval and data synthesis.

The systematic synthetic data workflow is introduced to optimize data quality and diversity through three key mechanisms: (1) Knowledge graph traversal and node combination to ensure conceptual complexity and domain coverage; (2) Multi-stage iterative refinement to progressively improve difficulty levels and Chain-of-Thought (CoT) reasoning quality; (3) Dual-modality generation and verification (textual and computational) to guarantee mathematical accuracy and solution validity. Careful quality control is conducted combing both rule-based and model-based filters, and the final dataset comprises hundreds of billions of tokens.

3.4 Long Context Extension

We implement a two-stage context length extension strategy to meet the requirements for subsequent long-context reasoning and agentic training. In the first stage, the context window expands from 8k to 32k tokens using 80B training tokens, with RoPE's base [Su et al., 2024] frequency raised from 1,000,000 to 5,000,000. In the second stage, we further extend it to 128k tokens through an additional 20B tokens, increasing the base frequency to 10,000,000.

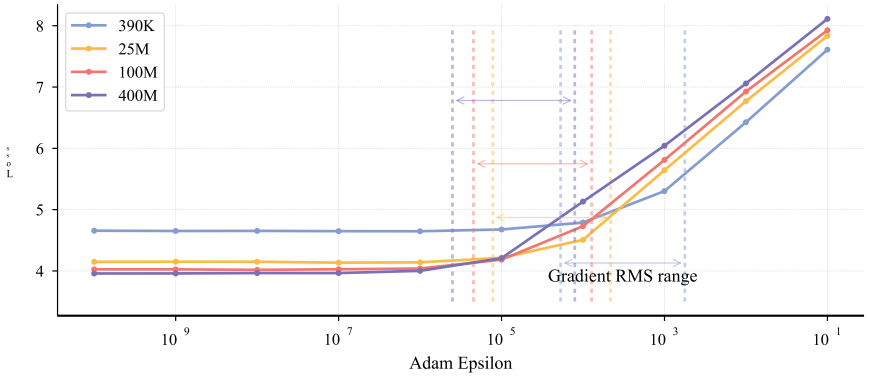


图 7：在不同模型规模下，探索梯度的均方根（RMS）范数和 epsilon 对损失的影响。Gradient RMS range 表示模型中不同权重的最大梯度 RMS 值与最小梯度 RMS 值之间的跨度。随着模型规模增加（参数数量从 39 万到 4 亿不等），梯度 RMS 变小。当 epsilon 接近梯度 RMS 的范围时，损失会迅速恶化。

内容提取 我们使用经过定制的 trafilatura [Barbaresi, 2021] 来处理通用网页内容，并为 STEM 材料设计了专用流程，以正确解析诸如公式、代码和表格等复杂元素。

质量筛选 采用两步筛选方法。初始分类器会移除显然低质量的文档，随后基于如流畅度和内容完整性等指标进行更细粒度的筛选。

去重 我们对大规模去重应用一种高效的MinHash 实现，并辅以一种策略，用于识别和处理重复的网页模板，以实现更准确的文档级去重。

最终数据混合过程采用两阶段计划，逐步提高高质量推理数据的比例（例如 STEM 和代码）。

- 阶段 1：对于通用数据，我们采用一个实例级别的数据混合策略，该策略在 SampleMix [Xi et al., 2025] 中描述，旨在平衡数据质量与多样性；我们使用质量和多样性分数来计算初始抽样分布，并基于细粒度的领域和写作风格标签进一步调整分布的倾向。冗余的低价值领域（例如广告、体育、招聘）将被下采样，而具备推理丰富性的领域（例如科学）将被上采样。
- 阶段二：在这一阶段，我们优先关注需要推理的领域，其中 STEM（科学、技术、工程、数学）和代码占最终混合的 70%。初步实验表明，对通用领域数据的突然减少会暂时削弱模型能力。因此，我们采取逐步增加代码所占比例的策略，并以外部验证集上的持续困惑度监测为引导，以确保在不损害通用性能的前提下实现平滑过渡。

3.3 推理与编码增强

为了进一步提升模型的推理能力与编程能力，并建立一个对后续训练具有巨大潜力的稳健基线模型，我们在中期训练阶段，利用通过预训练数据检索与数据合成相结合而生成的高质量相关数据进行训练。

该系统化的合成数据工作流程被引入，旨在通过三个关键机制来优化数据质量与多样性： 1）知识图谱遍历与节点组合，以确保概念复杂性和领域覆盖； 2）多阶段迭代式改进，以逐步提升难度水平和 Chain-of-Thought（CoT）推理质量； 3）双模态生成与验证（文本与计算）以保证数学准确性和解答的正确性。 并进行严格的质量控制，结合基于规则的过滤与基于模型的过滤，最终数据集包含数千亿个词元。

3.4 长文本上下文扩展

我们实现了一项两阶段的上下文长度扩展策略，以满足后续长上下文推理和智能体训练的需求。第一阶段，上下文窗口从8k扩展至32k标记，使用80B个训练标记，RoPE的基频[Su 等, 2024]从1,000,000提升至5,000,000。第二阶段，我们再通过额外的20B个训练标记将其扩展至128k标记，使基频提升至10,000,000。

The training corpus is built upon naturally occurring long-text data, such as high-quality books and novels. Additionally, we developed a systematic approach to organize repository-level source code to improve the model's long-context capabilities. We carefully selected high-quality repositories and applied a multi-stage filtering process to remove non-textual content, build artifacts, and auto-generated code, resulting in a curated 20B-token dataset for long-context pre-training.

To ensure that the model's general capabilities remain stable during the length extension, we adopt a data mixture strategy identical to that of our main pre-training phase and augment this mixture with an additional 25% of long-context data to enhance the model's long-context performance.

3.5 Decontamination

We perform rigorous decontamination on all training data to prevent data leakage from test sets of common benchmarks. For web and code data, we remove documents containing any 13-gram overlap with predefined test sets. For synthetic data and question-answering pairs, we employ a stricter strategy based on semantic similarity using BGE-m3 [Chen et al., 2024] embeddings. Documents are discarded if they meet either of the following criteria: (1) Semantic similarity score > 0.9 to any test case; (2) Lexical overlap (measured by sparse embeddings) combined with a similarity score between 0.7–0.9.

3.6 Evaluation

This section presents a comprehensive evaluation of the LongCat-Flash base model, including the methodology and results.

3.6.1 Evaluation Benchmarks and Configurations

The model evaluation covers four core capabilities: general tasks, general reasoning, mathematical reasoning, and coding. The benchmarks used for assessment include:

- **General Tasks:** MMLU [Hendrycks et al., 2021a], MMLU-Pro [Wang et al., 2024b], C-Eval [Huang et al., 2023], and CMMLU [Li et al., 2023a].
- **Reasoning Tasks:** GPQA [Rein et al., 2023], SuperGPQA [M-A-P Team, ByteDance., 2025], BBH [Suzgun et al., 2023], PIQA [Bisk et al., 2019], DROP [Dua et al., 2019], CLUEWSC [Xu et al., 2020], and WinoGrande [Sakaguchi et al., 2019].
- **Math Tasks:** GSM8K [Cobbe et al., 2021], MATH [Hendrycks et al., 2021b].
- **Coding Tasks:** MBPP+ [Liu et al., 2024b], HumanEval+ [Liu et al., 2024b], MultiPL-E [Cassano et al., 2022], and CRUXEval [Gu et al., 2024].

We compare the LongCat-Flash base model with state-of-the-art open-source base MoE models, including DeepSeek-V3.1 Base [DeepSeek-AI et al., 2025], Llama-4-Maverick Base [Meta AI, 2025], and Kimi-K2 Base [MoonshotAI, 2025].

To ensure fairness, all models are evaluated under identical pipelines and configurations. For minority results that cannot be reproduced, we directly adopt metrics from public reports and explicitly annotate them in Table 2. The evaluation settings are as follows:

- General/reasoning/math tasks: Use few-shot prompts to guide output format. Performance is measured via accuracy or F1 score.
- HumanEval+ and MBPP+: Follow OpenAI's recommended setting [Chen et al., 2021].
- MultiPL-E: Follow BigCode Evaluation Harness[Ben Allal et al., 2022].
- CRUXEval: Follow the official configuration[1], employing 2-shots examples.

3.6.2 Evaluation Results

Table 2 presents the evaluation results across diverse benchmarks. LongCat-Flash Base model achieves performance on par with state-of-the-art base models despite its compact active/total parameter size. Although Llama-4-Maverick has fewer activated and total parameters, LongCat-Flash Base surpasses both on nearly all benchmarks.

[1] https://github.com/facebookresearch/cruxeval

训练语料库建立在自然产生的长文本数据之上，例如高质量的书籍和小说。此外，我们开发了一套系统化的方法来组织仓库级别的源代码，以提升模型的长上下文能力。我们经过精心挑选高质量的代码仓库，并应用多阶段筛选流程，去除非文本内容、构建产物和自动生成的代码，从而获得一个经过精心整理的 20B-token 数据集，用于长上下文预训练。

为了在长度扩展期间保持模型的一般能力的稳定性，我们采用与主预训练阶段相同的数据混合策略，并在该混合中再增加25%的长上下文数据，以提升模型在长上下文方面的性能。

3.5 去污染

我们对所有训练数据进行严格的净化处理，以防止来自常见基准测试的测试集数据泄露。对于网络数据和代码数据，我们删除与预定义测试集存在任意 13-gram 重叠的文档。对于合成数据和问答对，我们采用基于语义相似性的更严格策略，使用 BGE-m3 [Chen et al., 2024] 的嵌入。文档如果符合以下任一条件，将被舍弃：（1）对任意测试用例的语义相似度分数 > 0.9；（2）词汇重叠（通过稀疏嵌入测量）结合一个在 0.7–0.9 之间的相似度分数。

3.6 评估

本节对 LongCat-Flash 基础模型进行了全面评估，包括方法学和结果。

3.6.1 评估基准与配置

模型评估涵盖四项核心能力：一般任务、通用推理、数学推理和编码。用于评估的基准包括：

- 常规任务：MMLU [Hendrycks 等人，2021a]、MMLU-Pro [Wang 等人，2024b]、C-Eval [Huang 等人，2023] 以及 CMMLU [Li 等人，2023a]。
- 推理任务：GPQA [Rein 等人，2023]、SuperGPQA [M-A-P Team，ByteDance., 2025]、BBH [Suzgun 等人，2023]、PIQA [Bisk 等人，2019]、DROP [Dua 等人，2019]、CLUEWSC [Xu 等人，2020]，以及 WinoGrand e [Sakaguchi 等人，2019]。
- 数学任务：GSM8K [Cobbe 等人，2021]，MATH [Hendrycks 等人，2021b]。
- 编码任务：MBPP+ [Liu 等, 2024b]、HumanEval+ [Liu 等, 2024b]、MultiPL-E [Cassano 等, 2022]，以及 CR UXEval [Gu 等, 2024]。

我们将 LongCat-Flash 基础模型与最先进的开源基础 MoE 模型进行比较，包括 DeepSeek- V3.1 Base [DeepSee k-AI 等人，2025]、Llama-4-Maverick Base [Meta AI，2025] 和 Kimi-K2 Base [MoonshotAI，2025]。

为了确保公平性，所有模型在相同的流水线与配置下进行评估。对于无法复现的少数结果，我们直接采用公开报告中的指标，并在表2中对它们进行明确标注。评估设置如下：

- 通用/推理/数学任务：使用少量示例提示来引导输出格式。性能通过准确率或 F1 分数来衡量。
- HumanEval+ 与 MBPP+：遵循 OpenAI 推荐的设置 [Chen 等, 2021]。
- MultiPL-E: 遵循 BigCode 评估框架[Ben Allal 等人, 2022].
- CRUXEval：遵循官方配置[1]，采用2-shot示例。

3.6.2 评估结果

表2展示了在不同基准测试中的评估结果。尽管 LongCat-Flash Base 模型的活跃参数量与总参数量都非常紧凑，其性能仍与最先进的基线模型相当。尽管 Llama-4-Maverick 的激活参数量和总参数量更少，LongCat-Fl ash Base 在几乎所有基准测试中都超越了两者。

[1] https://github.com/facebookresearch/cruxeval

A comparative analysis reveals that LongCat-Flash Base matches DeepSeek-V3.1 Base's performance across all domains despite containing fewer parameters. While the two models perform similarly in general tasks, LongCat-Flash Base demonstrates a notably advantage on the MMLU-Pro benchmark (featuring challenging questions). For reasoning tasks, LongCat-Flash Base attains a higher average score. In math and coding tasks, it outperforms DeepSeek-V3.1 Base on most benchmarks, with only marginal performance gaps observed on CRUXEval and MultiPL-E. Against Kimi K2 Base, LongCat-Flash Base shows modestly lower performance in general tasks but achieves parity or superiority in reasoning, math, and coding tasks.

These results collectively underscore LongCat-Flash Base's parameter efficiency, as it delivers competitive or superior performance to larger models across the majority of evaluated benchmarks.

Table 2: Comparison between LongCat-Flash and other base models. Values marked with * are sourced from public reports.

Benchmark	DeepSeek-V3.1 Base	Llama-4-Maverick Base	Kimi-K2 Base	LongCat-Flash Base
Architecture	MoE	MoE	MoE	MoE
# Total Params	671B	402B	1043B	560B
# Activated Params	37B	17B	32B	27B
General Domains				
MMLU (acc)	87.46	84.41	87.47	87.05
MMLU-Pro (acc)	59.29	63.90	68.36	70.32
CEval (acc)	89.33	81.93	91.24	87.73
CMMLU (acc)	88.21	80.71	90.35	87.19
General Reasoning				
GPQA (acc)	47.16	48.08	45.89	51.09
SuperGPQA (acc)	-	40.58*	44.70*	54.19
BBH (acc)	89.46	87.56	89.19	90.54
DROP (f1)	80.74	77.44	69.81	78.39
PIQA (acc)	93.00	90.59	95.10	92.33
WinoGrande (acc)	83.50	73.32	82.87	85.08
CLUEWSC (acc)	88.16	88.00	76.32	91.12
Mathematical Reasoning				
GSM8K (acc)	92.22	84.61	92.27	92.19
MATH (acc)	61.56	63.34	66.74	64.82
Coding				
MBPP+ (pass@1)	59.26	70.11	80.49	77.25
HumanEval+ (pass@1)	67.07	60.37	69.84	65.85
MultiPL-E (pass@1)	62.00	58.35	59.22	69.25
CRUXEval-I (pass@1)	65.87	62.00	65.87	71.63
CRUXEval-O (pass@1)	71.25	64.25	68.75	75.88

4 Post-Training

We implement a conventional multi-stage post-training framework to augment the base model's performance across diverse domains, ranging from sophisticated reasoning, coding and agentic tool use tasks to general-purpose capabilities. During this process, we observed that the limited availability of high-quality problem sets is a significant bottleneck across all domains. In the subsequent sections, we present key insights derived from our post-training methodology, organized into three distinct phases: (1) Reasoning and coding, (2) Agentic tool use, and (3) General capability.

4.1 Reasoning and Coding

Mathematics To generate high-quality and novel problems, we use a persona [Ge et al., 2024], self-instruct [Wang et al., 2022] paradigm. This process is guided by a comprehensive mathematical framework that spans topics from elementary to advanced levels. We leverage a diverse set of math-related "expert" personas to ask questions, steering

比较分析显示，LongCat-Flash Base 在各领域的表现与 DeepSeek-V3.1 Base 相当，尽管其参数量更少。尽管两种模型在一般任务上的表现相近，LongCat-Flash Base 在 MMLU-Pro 基准测试（包含挑战性的问题）上显示出显著的优势。对于推理任务，LongCat-Flash Base 取得更高的平均分。在数学与编程任务方面，它在大多数基准测试上优于 DeepSeek-V3.1 Base，只有在 CRUXEval 和 MultiPL-E 上观察到微小的性能差距。与 Kimi K2 Base 相比，LongCat-Flash Base 在一般任务中的表现略低，但在推理、数学和编程任务上达到同等或更优的水平。

这些结果共同凸显 LongCat-Flash Base 的参数效率，因为它在大多数经过评估的基准测试中，能够提供与更大模型相当甚至更优的性能。

表2：LongCat-Flash 与其他基线模型的比较。标有 * 的数值取自公开报告。

Benchmark	DeepSeek-V3.1 Base	Llama-4-Maverick Base	Kimi-K2 Base	LongCat-Flash Base
Architecture	MoE	MoE	MoE	MoE
# Total Params	671B	402B	1043B	560B
# Activated Params	37B	17B	32B	27B
General Domains				
MMLU (acc)	87.46	84.41	87.47	87.05
MMLU-Pro (acc)	59.29	63.90	68.36	70.32
CEval (acc)	89.33	81.93	91.24	87.73
CMMLU (acc)	88.21	80.71	90.35	87.19
General Reasoning				
GPQA (acc)	47.16	48.08	45.89	51.09
SuperGPQA (acc)	-	40.58*	44.70*	54.19
BBH (acc)	89.46	87.56	89.19	90.54
DROP (f1)	80.74	77.44	69.81	78.39
PIQA (acc)	93.00	90.59	95.10	92.33
WinoGrande (acc)	83.50	73.32	82.87	85.08
CLUEWSC (acc)	88.16	88.00	76.32	91.12
Mathematical Reasoning				
GSM8K (acc)	92.22	84.61	92.27	92.19
MATH (acc)	61.56	63.34	66.74	64.82
Coding				
MBPP+ (pass@1)	59.26	70.11	80.49	77.25
HumanEval+ (pass@1)	67.07	60.37	69.84	65.85
MultiPL-E (pass@1)	62.00	58.35	59.22	69.25
CRUXEval-I (pass@1)	65.87	62.00	65.87	71.63
CRUXEval-O (pass@1)	71.25	64.25	68.75	75.88

4 后训练阶段

我们实现了一种传统的多阶段后训练框架，以提升基础模型在跨越多样领域中的性能，覆盖从复杂推理、编码和代理性工具使用任务到通用能力等各类任务。在此过程中，我们发现高质量题集的数量有限是横跨所有领域的一个重要瓶颈。在随后的章节中，我们将给出从我们的后训练方法中得出的关键见解，分为三个不同阶段：（1）推理与编码，（2）代理性工具使用，以及（3）通用能力。

4.1 推理与编码

数学　为生成高质量且新颖的问题，我们采用一个结合 Ge 等人（2024）的角色设定与 Wang 等人（2022）的自我指令范式的范式。这一过程由一个涵盖从初等到高阶水平的综合数学框架来引导。我们利用一组多样化、与数学相关的“专家”身份来提问，从而引导

LLMs to synthesize queries that cover underrepresented subjects. Each query is structured to elicit Chain-of-Thought (CoT) reasoning, promoting step-by-step problem-solving in the generated answers. Details of persona curation and answer verification are as follows:

- Persona Curation: The personas are constructed from multiple sources: we generate them from our high-quality pretraining data, derive them from existing math queries, and incorporate relevant collections from Persona Hub. Each persona is systematically labeled by its STEM discipline. To ensure maximum diversity and alignment with our subject framework, we use the MinHash algorithm to select the final set of personas for query generation.
- Answer Verification: We employ a two-stage process to ensure the accuracy of the synthesized solutions: (1) We generate answers for each problem using several different LLMs and select the most consistent solution as the final answer. (2) We train a generative reward model, specifically enhanced with reasoning data, to automatically score and verify the logical soundness of the problem-solving steps.

Coding We assemble a diverse set of coding queries from multiple sources, including public datasets, queries generated from GitHub code snippets [Wei et al., 2024] and coding-related forums, as well as queries evolved using the Code Evol-Instruct method [Luo et al., 2024]. The data distribution is balanced according to topic diversity and difficulty. Specifically, we train a model to select queries that are clear, consistent, and correct, with sufficient explanatory detail, and implement a filtering pipeline to eliminate responses containing garbled content, repetitive patterns, or logical errors. For software engineering tasks, we curate and validate ten thousands of Docker images containing test cases. Each image is used to verify whether model-generated code can resolve specific issues in the corresponding repository. We develop an agent-based system that leverages various tools to autonomously analyze code structures, identify relevant files, fix bugs, and implement new features. This process yields thousands of successful trajectories that pass all test cases, thereby enhancing the model's ability to autonomously solve real-world software engineering problems.

Logical Reasoning We construct logical reasoning datasets covering deductive, hypothetical, and inductive reasoning, which include tasks such as LogicPro [Jiang et al., 2025], PODA [Wang et al., 2025b], and Zebra-style logic puzzles. To manage difficulty, we first use the Pass@k metric for an initial balance, then filter out intractable problems where advanced thinking models failed. We also convert multiple-choice questions to a fill-in-the-blank format to mitigate random guessing. The evaluation of responses focused on four key areas: (1) correctness of the final answer; (2) completeness and clarity of reasoning; (3) avoidance of excessive repetition; and (4) consistent use of language.

4.2 Agentic Tool Use

We define agentic tasks as complex problem resolution through systematic environment interaction. In this paradigm, models must iteratively analyze existing information and determine when environmental interaction is needed. Specifically, within the agentic tool utilization framework, the environment comprises user and tools with distinct characteristics. User functions as an autonomous information-providing entity without upstream or downstream dependencies, but exhibit reluctance to be disturbed and non-spontaneous information disclosure. Consequently, models must minimize user queries while employing strategic questioning techniques to elicit maximally precise information when interaction becomes necessary. Tools can be invoked extensively with high frequency, but exhibit intricate interdependencies. From this perspective, excluding domain-specific expertise such as advanced programming capabilities or mathematical computation, we attribute task difficulty escalation to three factors:

- **Information processing complexity** Models must engage in sophisticated reasoning processes to integrate and transform information into required components.
- **Tool set complexity** By modeling the tool set as a directed graph based on intertool dependencies, complexity can be quantitatively characterized by the graph's node cardinality and edge density.
- **User interaction complexity** Models must learn to engage in multi-round strategic questioning with minimal frequency, adapting to various conversational styles, levels of communication willingness and patterns of information disclosure, thus facilitating effective user interaction while ensuring adequate information acquisition.

Building on these insights, we construct a multi-agent data synthesis framework that generates high-quality challenging tasks by systematically addressing three complexity dimensions critical for agent training: (1) tool set complexity, (2) information processing complexity, and (3) user interaction complexity. The framework comprises the following specialized agents:

- **UserProfileAgent** Beyond generating fundamental user profiles encompassing personal information and preferences, we further implement controls over user conversational styles, communication willingness levels, and information

大型语言模型用于合成覆盖代表性不足主题的查询。每个查询的结构旨在诱导连锁思维（CoT）推理，促进生成答案中的逐步解题过程。关于人设策划和答案验证的细节如下： 翻译文本:

- 人设筛选：这些人设来自多个来源：我们从高质量的预训练数据中生成它们，从现有的数学查询中推导出它们，并将 Persona Hub 的相关收藏纳入其中。每个人设都按其 STEM 学科进行系统标注。为确保最大程度的多样性并与我们的学科框架保持一致，我们使用 MinHash 算法来选择用于查询生成的最终人设集合。
- 答案验证：我们采用两阶段流程以确保合成解的准确性：(1) 我们使用几种不同的大型语言模型为每道题目生成答案，并将最一致的解作为最终答案。(2) 我们训练一个生成式奖励模型，特别通过推理数据增强，以自动对解题步骤的逻辑健全性进行评分和验证。

编码 我们汇集来自多个来源的多样化编码查询，包括公开数据集、来自 GitHub 代码片段 [Wei et al., 2024] 的查询，以及与编码相关的论坛中的查询，以及通过 Code Evol-Instruct 方法 [Luo et al., 2024] 演化出的查询。数据分布根据主题多样性和难度进行平衡。具体而言，我们训练一个模型来选择清晰、连贯且正确、具有充分解释细节的查询，并实现一个筛选管道，以排除包含乱码内容、重复模式或逻辑错误的回答。对于软件工程任务，我们整理并验证包含测试用例的成千上万的 Docker 镜像。每个镜像用于验证模型生成的代码是否能够解决相应代码库中的特定问题。我们开发一个基于代理的系统，利用各种工具来自动分析代码结构、识别相关文件、修复错误并实现新功能。这一过程产生了成千上万条通过所有测试用例的成功轨迹，从而提升模型自动解决现实世界软件工程问题的能力。

逻辑推理，我们构建覆盖演绎、假设和归纳推理的逻辑推理数据集，其中包括 LogicPro [Jiang et al., 2025]、PODA [Wang et al., 2025b]，以及 Zebra 风格的逻辑谜题等任务。为了控制难度，我们首先使用 Pass@k 指标来实现初步平衡，然后筛除那些即使使用高级推理模型也难以处理的问题。我们还将选择题转换为填空题格式，以降低随机猜测的影响。对回答的评估聚焦于四个关键方面：(1) 最终答案的正确性；(2) 推理的完整性与清晰性；(3) 避免过度重复；(4) 语言的一致性。

4.2 代理性工具使用 翻译文本:

我们将具代理性的任务定义为通过系统化的环境交互来解决复杂问题。在这种范式下，模型必须对现有信息进行迭代分析，并判断何时需要与环境交互。具体来说，在具代理性工具利用框架内，环境由具有不同特征的用户和工具组成。用户作为一个自主的信息提供实体，既没有上游或下游的依赖，但不愿被打扰，信息披露也缺乏自发性。因此，模型必须在尽量减少对用户的查询的同时，采用策略性提问技巧，在需要互动时尽可能获取最精确的信息。工具可以被广泛且高频地调用，但存在错综复杂的相互依赖关系。从这个角度看，排除诸如高级编程能力或数学计算等领域特定专业知识，我们将任务难度的上升归因于三个因素:

- 信息处理的复杂性 模型必须进行复杂的推理过程，将信息整合并转化为所需的组件。
- 工具集复杂度：通过将工具集建模为基于工具间依赖关系的有向图，可以通过图的节点基数和边密度来对复杂性进行定量表征。
- 用户交互复杂性 模型必须学会以尽量低的频率进行多轮策略性提问，以适应各种对话风格、沟通意愿水平和信息披露模式，从而在确保获得充足信息的同时促进有效的用户互动。

在这些洞见的基础上，我们构建一个多智能体数据合成框架，通过系统性地解决对智能体训练至关重要的三个复杂性维度来生成高质量、具有挑战性的任务：（1）工具集复杂性，（2）信息处理复杂性，以及（3）用户交互复杂性。该框架由以下专门的智能体组成:

- UserProfileAgent 除了生成涵盖个人信息和偏好等基本用户画像之外，我们还进一步对用户对话风格、沟通意愿水平和信息进行控制。

disclosure patterns to more accurately simulate authentic user interaction scenarios while simultaneously enhancing task complexity.

- **ToolSetAgent** To maximize data diversity and prevent overfitting to specific scenarios, we adopt an approach analogous to Kimi-K2 [Team et al., 2025], enumerating 40 distinct domains and subsequently leveraging models to enumerate 1,600 applications. Based on these applications, we construct 80,000 mock tools, forming an extensive tool graph. Through random walk methodologies, we systematically sample subgraphs with predetermined node quantities from this comprehensive tool graph, and hence tool graph complexity is controlled via node quantity.
- **InstructionAgent** The difficulty of reasoning is quantified in the following dimensions: constraint complexity, quantity of reasoning points, and length of the reasoning chain. The model is required to generate instructions that comprehensively describe complete tasks based on the tool set extracted by the **ToolSetAgent**.
- **EnvironmentAgent** We augment environmental information including item details, location specifics, temporal parameters, and meteorological conditions based on content generated by the **UserProfileAgent** and **InstructionAgent**. Additionally, we introduce confounding elements for items and locations to further increase reasoning complexity.
- **RubricAgent** We construct a comprehensive series of specific checklists based on various task-related information. During final evaluation, considering the long-context characteristics inherent to agentic tasks, we employ a sliding window approach to assess the entire trajectory, continuously updating the completion status of checklist items.
- **ValidatorAgent** and **DeduplicatorAgent** We check the quality of our final tasks from several angles and remove any that are too similar. This process ensures we have a diverse and high-quality set of tasks.

With these high-quality challenging tasks, we further conduct rigorous response selection to construct our cold start training set with an appropriate quantity, revealing diverse patterns and preserving high exploration ability. We also carefully select a subset of these generated task for further post-training procedure, to make sure each task worth massive exploration.

4.3 General Capability

Instruction-following We curate both single-turn and multi-turn instruction-Following datasets, with varying levels of constraint complexity and quantity. For multiple-constraint queries, we adopt the insight from Ye et al. [2025] to filter queries with low semantic quality or constraint conflicts. For different query types, we employ verifiable rules, model-based verification, and customized strategies to ensure responses satisfy all constraints. Additionally, we compile critique datasets targeting challenging tasks to enhance the model's critical thinking abilities [Wang et al., 2025c]. We observe that certain constraint types are inherently difficult to follow, making direct generation of valid query-answer pairs unreliable. To address this, we propose a reverse prompt generation strategy: generating queries from predefined answers guaranteed to meet constraints.

Long Context To enable the model to identify and analyze relevant information within complex, lengthy contexts, we develop three types of long-sequence datasets: reading comprehension, table-based question answering, and custom-designed tasks. To facilitate the learning of salient information in long sequences, we aggregate topically related context segments for data construction. We specifically enhance the model's multi-hop reasoning, multi-turn dialogue, and complex calculation abilities. To mitigate hallucination when confronted with an incomplete context, we optimize the model's refusal capabilities, thereby improving its awareness of knowledge boundaries and limitations.

Safety Building on the framework of Mu et al. [2024] and aligned with our internal content guidelines, we develop a content safety policy that categorizes queries into more than 40 distinct safety categories across five response types: *comply, comply with guideline, soft refuse, soft refuse with guideline, or hard refuse*. Explicit criteria ensure consistent, safety standards-compliant responses for each response type. This system operates as a context-aware data synthesizer through two stages: (1) Query Classification: Queries from diverse sources (open-domain corpora, internal business risk reports, government Q&A, and adversarial LLM-synthesized red-team content) are classified into safety categories using automated labeling with human verification. (2) Response Mapping & Optimization: Classified queries are mapped to response types and generate optimized, type-specific responses that undergo human evaluation before serving as training targets.

4.4 Evaluation

We conduct a comprehensive and rigorous evaluation of LongCat-Flash after post-training. Specifically, we assess its capabilities across multiple dimensions, including general domains, instruction following, mathematical reasoning, general reasoning, and coding & agent tasks.

信息披露模式，以更准确地模拟真实的用户交互场景，同时提高任务的复杂性。

- ToolSetAgent 为了最大化数据多样性并防止对特定场景的过拟合，我们采用一种类似于 Kimi-K2 [Team et al., 2025] 的方法：枚举 40 个不同领域，随后利用模型对 1,600 个应用进行枚举。基于这些应用，我们构建 80,000 个模拟工具，形成一个庞大的工具图。通过随机游走的方法，我们从这个综合工具图中系统地抽取具有事先设定的节点数量的子图，因此通过节点数量来控制工具图的复杂度。
- 指令代理：推理的难度在以下维度中被量化：约束复杂度、推理点的数量，以及推理链的长度。该模型需要基于 ToolSetAgent 提取的工具集生成能够全面描述完整任务的指令。
- 环境代理 我们增强环境信息，其中包括物品细节、地点细节、时间参数和气象条件，这些信息基于由用户画像代理和指令代理生成的内容。此外，我们为物品和地点引入混淆元素，以进一步增加推理复杂性。翻译文本：
- RubricAgent 我们基于各种与任务相关的信息，构建了一系列全面的具体检查清单。在最终评估时，考虑到代理任务固有的长上下文特征，我们采用滑动窗口方法来评估整个轨迹，并持续更新检查清单项的完成状态。
- ValidatorAgent 和 DeduplicatorAgent，我们从多个角度检查最终任务的质量，并删除任何过于相似的任务。这一过程确保我们拥有多样且高质量的任务集合。

借助这些高质量、具有挑战性的任务，我们进一步进行严格的响应选择，以构建冷启动训练集的适量规模，揭示多样化的模式并保持较强的探索能力。我们还从这些生成的任务中仔细选择一个子集用于后续的训练后处理，以确保每个任务都值得进行大规模的探索。

4.3 通用能力

指令遵循：我们同时整理单轮与多轮的指令遵循数据集，具有不同层次的约束复杂度和数量。对于多约束的查询，我们采用 Ye 等人 [2025] 的洞见来筛选出语义质量较低或存在约束冲突的查询。对于不同类型的查询，我们采用可验证的规则、基于模型的验证，以及定制化策略，以确保回答符合所有约束。此外，我们还编制面向具有挑战性的任务的批判性数据集，以提升模型的批判性思维能力 [Wang et al., 2025c]。我们观察到某些约束类型天生就难以遵循，这使直接生成有效的查询-答案对变得不可靠。为了解决这个问题，我们提出一种反向提示生成策略：从预定义且确保满足约束的答案中生成查询。

长文本上下文 为了使模型能够在复杂且冗长的上下文中识别和分析相关信息，我们开发了三种长序列数据集：阅读理解、基于表格的问答任务，以及自定义设计的任务。为了促进在长序列中学习到重要信息，我们聚合主题相关的上下文片段以用于数据构建。我们专门增强模型的多跳推理、多轮对话以及复杂计算能力。为在面对不完整上下文时减少幻觉，我们优化模型的拒绝能力，从而提高其对知识边界和局限性的认识。

安全性 在 Mu 等人[2024] 的框架基础上，并与我们内部的内容指南保持一致，我们制定了一项内容安全政策，将查询分为超过40个不同的安全类别，覆盖五种响应类型：
comply, comply with guideline, soft refuse, soft refuse with guideline, or hard refuse。明确的标准确保每种响应类型都提供一致、符合安全标准的回答。该系统作为一个具情境感知的数据综合器，分两阶段运作：（1）查询分类：来自多源的查询（开放域语料库、内部业务风险报告、政府问答，以及对抗性大模型合成的红队内容）通过自动标注并经人工验证的方式被分类到安全类别中。（2）响应映射与优化：将分类后的查询映射到响应类型，并生成经过优化、针对性强的响应，随后经人工评估后再作为训练目标使用。

4.4 评估

我们在后训练阶段对 LongCat-Flash 进行全面而严格的评估。具体而言，我们评估其在多个维度上的能力，包括通用领域、指令遵循、数学推理、一般推理，以及编程与代理任务。

4.4.1 Evaluation Benchmarks and Configurations

The evaluation employs the following benchmarks:

- **General Domains:** MMLU [Li et al., 2023a], MMLU-Pro [Wang et al., 2024b], ArenaHard [Li et al., 2024a,b], CEval [Huang et al., 2023], and CMMLU [Li et al., 2023a].
- **Instruction Following:** IFEval [Zhou et al., 2023], COLLIE [Yao et al., 2024], and Meeseeks [Wang et al., 2025a], Meeseeks evaluates models' instruction-following capabilities in multi-turn scenarios through an iterative feedback framework that simulates realistic human-LLM interactions, enabling models to self-correct based on turn-specific failures and better reflect real-world usage patterns.
- **Mathematical Reasoning:** MATH500 [Lightman et al., 2023], AIME24 [MAA, 2024], AIME25 [MAA, 2025], and BeyondAIME [ByteDance-Seed, 2025].
- **General Reasoning:** GPQA-diamond [Rein et al., 2024], DROP [Dua et al., 2019], ZebraLogic [Lin et al., 2025], and GraphWalks [OpenAI, 2025a].
- **Coding:** Humaneval+, MBPP+ [Liu et al., 2023, 2024c], LiveCodeBench (2024.08-2025.05) [Jain et al., 2025], SWE-Bench-Verified [Jimenez et al., 2024], and TerminalBench [Team, 2025a].
- **Agentic Tool Use:** τ^2-Bench [Barres et al., 2025] and AceBench [Chen et al., 2025]. Furthermore, we develop a high-quality proprietary benchmark, VitaBench, leveraging Meituan's comprehensive real-world business scenarios to systematically evaluate models' capabilities in addressing complex real-world tasks. Within VitaBench, to comprehensively assess models' generalized agentic capabilities, we deliberately curate cross-domain quotidian scenarios and explicitly delineate inter-tool dependencies, eschewing the provision of extensive domain-specific policies. Our benchmark emphasizes three critical dimensions of complexity: tool set complexity (characterized by dense tool graphs averaging over 30 available tools per task), reasoning complexity, and user interaction complexity (featuring challenging user personas with an average exceeding 60 interaction rounds per task for evaluated models). The complete benchmark dataset, along with detailed construction methodologies and comprehensive result analysis, will be fully released in subsequent work.

We also evaluate the safety performance of LongCat-Flash. Specifically, we conduct evaluations on four major risk categories:

- **Harmful**: Violence, hate Speech, insulting, harassment and bullying, self-harm and suicide, adult content, etc.
- **Criminal**: Illegal activities, underage violations, extreme terrorism and violence, etc.
- **Misinformation**: misinformation and disinformation, unsafe practices, hallucination, etc.
- **Privacy**: privacy violation, infringement, etc.

Within each category, a sufficient number of private test queries are constructed, followed by a comprehensive manual review to ensure the accuracy of their classification and the reliability of their quality.

We compare the chat version of LongCat-Flash with several contemporary non-thinking chat models, including DeepSeek-V3.1 [DeepSeek-AI et al., 2025], Qwen3-235B-A22B (2507 version) [Yang et al., 2025], Kimi-K2 [MoonshotAI, 2025], GPT-4.1 [OpenAI, 2025b], Claude4-Sonnet [Anthropic, 2025], and Gemini2.5-Flash [Comanici et al., 2025]. For closed-source models, we conduct evaluations through their official APIs. For models supporting both thinking and non-thinking modes (Qwen3-235B-A22B, Gemini2.5-Flash, and Claude4-Sonnet), we explicitly configure these models to operate in non-thinking mode for a fair comparison.

For each benchmark category, we employ the following specialized metrics and settings:

- **General domain benchmarks:** We use accuracy as the evaluation metric. Unlike the original benchmarks that rely on exact-match (EM) for correctness judgment, we employ a scoring model to assess whether model responses align with reference answers. Since our scoring model recognizes semantically correct answers even without exact textual matches, reported values may be slightly higher than originally documented.
- **Instruction following benchmarks:** We design regular expressions based on instruction rules to verify compliance. Rule-based and model-based answer span extraction tools are additionally employed to support this evaluation.
- **Mathematical reasoning benchmarks:** We apply the aforementioned scoring model for MATH500, and the averaged EM scores over 10 runs for AIME-related benchmarks.
- **General reasoning benchmarks:** We apply the scoring model for GPQA-diamond, calculate the F1 score for DROP, adopt rule-based matching for ZebraLogic, and use the precision metric for GraphWalk following the official implementation on its 128k context length subset.

4.4.1 评估基准与配置

评估使用以下基准：

- 通用领域：MMLU [Li 等，2023a]、MMLU-Pro [Wang 等，2024b]、ArenaHard [Li 等，2024a,b]、CEval [Huang 等，2023] 以及 CMMLU [Li 等，2023a]。
- 指令遵循：IFEval [Zhou 等，2023]、COLLIE [Yao 等，2024]、以及 Meeseeks [Wang 等，2025a]，Meeseeks 通过一个迭代式反馈框架，在多轮场景中评估模型的指令遵循能力，该框架模拟真实的人与大语言模型（LLM）之间的互动，使模型能够基于轮次特定的失败进行自我纠正，并更好地反映现实世界的使用模式。
- 数学推理：MATH500 [Lightman 等，2023]、AIME24 [MAA，2024]、AIME25 [MAA，2025] 以及 Beyond AIME [ByteDance-Seed，2025]。
- 通用推理：GPQA-diamond [Rein et al., 2024]、DROP [Dua et al., 2019]、ZebraLogic [Lin et al., 2025] 以及 GraphWalks [OpenAI, 2025a]。
- 编码：Humaneval+、MBPP+［Liu 等，2023，2024c］、LiveCodeBench（2024.08-2025.05）［Jain 等，2025］、SWE-Bench-Verified［Jimenez 等，2024］，以及 TerminalBench［Team，2025a］。
- 具代理性的工具使用：τ^2-Bench [Barres et al., 2025] 与 AceBench [Chen et al., 2025]。此外，我们开发高质量的专有基准 VitaBench，借助美团全面的真实世界业务场景，系统地评估模型在解决复杂现实世界任务方面的能力。在 VitaBench 中，为全面评估模型的通用代理能力，我们刻意策划跨领域的日常场景，并明确界定工具之间的相互依赖，避免提供大量领域特定的政策。我们的基准强调三个关键的复杂性维度：工具集复杂性（以每个任务平均拥有超过30个可用工具的密集工具图来表征）、推理复杂性，以及用户交互复杂性（具有挑战性的用户画像，评估模型的每个任务平均需要超过60轮交互）。完整的基准数据集，以及详细的构建方法和全面的结果分析，将在后续工作中全部公开发布。

我们还评估 LongCat-Flash 的安全性能。具体来说，我们对四大主要风险类别进行评估：

- 有害内容：暴力、仇恨言论、侮辱、骚扰与霸凌、自我伤害与自杀、成人内容等。
- 犯罪：非法活动、未成年违规行为、极端恐怖主义与暴力等。
- 误导信息：误导信息与虚假信息、不安全的做法、幻觉等。
- 隐私：侵犯隐私、侵权等。

在每个类别中，构建了足够数量的私有测试查询，随后进行全面的人工评审，以确保它们分类的准确性和质量的可靠性。

我们将 LongCat-Flash 的聊天版本与若干当代非思考型聊天模型进行比较，包括 DeepSeek-V3.1 [DeepSeek-AI 等，2025]、Qwen3-235B-A22B (2507 版本) [Yang 等，2025]、Kimi-K2 [Moon- shotAI，2025]、GPT-4.1 [OpenAI，2025b]、Claude4-Sonnet [Anthropic，2025]，以及 Gemini2.5-Flash [Comanici 等，2025]。对于闭源模型，我们通过它们的官方 API 进行评估。对于同时支持思考和非思考模式的模型（Qwen3-235B-A22B、Gemini 2.5-Flash、以及 Claude4-Sonnet），为公平比较，我们明确将这些模型配置为在非思考模式下运行。

对于每个基准类别，我们采用以下专门的指标和设置：

- 通用领域基准：我们使用准确性作为评估指标。与依赖精确匹配（EM）来判断正确性的原始基准不同，我们使用一个评分模型来评估模型回答是否与参考答案一致。由于我们的评分模型即使在没有严格文本匹配的情况下也能识别语义正确的答案，所报告的数值可能会略高于原始记录的数值。
- 指令遵循基准测试：我们设计基于指令规则的正则表达式来验证合规性。还采用基于规则的和基于模型的答案区间提取工具来支持这一评估。
- 数学推理基准：我们对 MATH500 应用前述评分模型；对于与 AIME 相关的基准，取 10 次运行的平均 EM 分数。
- 通用推理基准：我们使用 GPQA-diamond 的评分模型，对 DROP 计算 F1 分数，采用基于规则的匹配来处理 ZebraLogic，并在 GraphWalk 上按照其官方实现的 128k 上下文长度子集使用精确度指标。

- **Coding benchmarks:** Each problem is scored 1 if the model's response passes all test cases in a sandbox environment or matches a specific state, otherwise 0. The final score is the average across all problems. We adopt the script provided by OpenAI[2] to evaluate Humaneval+ and MBPP+, and the official scripts for the others. Specifically, for SWE-Bench-Verified, we use R2E-Gym[3] (Openhands scraffold) with runs limited to 100 iterations for evaluation except DeepSeek V3.1 (using Openhands[4]). For Terminal-Bench, we use the Terminus framework with direct prompting for evaluation.

- **Agentic tool use benchmarks:** We utilize official benchmark frameworks to ensure fairness and reproducibility. For AceBench, we use direct prompting rather than function calling. For our proposed **VitaBench**, given the inherent long-context characteristics of agentic tasks, we employ a sliding window mechanism to systematically evaluate task completion status throughout the entire execution trajectory, facilitating continuous updates to the completion status of individual checklist components.

4.4.2 Evaluation Results

As detailed in Table 3, our comprehensive evaluation reveals that LongCat-Flash is a powerful and versatile model. It consistently demonstrates leading performance in different domains, often outperforming contemporary models across a wide array of challenging tasks with relatively fewer activated parameters. The following analysis provides a detailed breakdown of its impressive capabilities across different dimensions.

General Domains In general domain knowledge, LongCat-Flash demonstrates a strong and well-rounded performance. It achieves an excellent score of 86.50 on ArenaHard-V2, ranking second among all evaluated models and showcasing its robust capabilities in challenging head-to-head comparisons. On foundational benchmarks, it remains highly competitive, scoring 89.71 on MMLU and 90.44 on CEval. These results are comparable to leading models, and notably, are achieved with fewer parameters than competitors like DeepSeek-V3.1 and Kimi-K2, indicating high efficiency.

Instruction Following LongCat-Flash exhibits state-of-the-art instruction following capabilities. It achieves the highest score of 89.65 on IFEval, outperforming all other models and demonstrating superior reliability in adhering to complex and nuanced directives. Furthermore, it secures the best score on COLLIE (57.10) and Meeseeks-zh (43.03), underscoring its exceptional proficiency across diverse and challenging instruction sets in both English and Chinese.

Mathematical Reasoning In mathematical reasoning, LongCat-Flash shows powerful and advanced capabilities. While its score on MATH500 (96.40) is highly competent, its strength is particularly evident in more complex, competition-level benchmarks. It delivers excellent, top-tier results on AIME25 (61.25) and BeyondAIME (43.00), ranking among the best-performing models in these challenging domains. This highlights its advanced capacity for sophisticated, multi-step logical deduction and problem-solving.

General Reasoning For general reasoning tasks, LongCat-Flash's performance is also solid. It demonstrates exceptional strength in structured logical deduction, achieving a score of 89.30 on ZebraLogic, which is among the top competitors. It also obtains a competitive score of 79.06 on the reading comprehension benchmark DROP. Conversely, its results on GPQA-diamond (73.23) and GraphWalks (51.05) indicate an opportunity for further improvement, particularly in enhancing its capabilities for analyzing structured data within extremely long contexts.

Coding LongCat-Flash displays a promising and capable profile in the coding domain. Its standout performance is on TerminalBench, where it achieves a score of 39.51, ranking second and demonstrating excellent proficiency in practical, agentic command-line tasks. It is also competitive on the SWE-Bench-Verified benchmark with a score of 60.4. On foundational code generation tasks such as Humaneval+ and MBPP+, its performance is solid, yet there remains potential for future optimization to align with the leading models.

Agentic Tool Use LongCat-Flash demonstrates a clear advantage in using agentic tool use domain, notably outperforming other models on τ^2-Bench even when compared to models with more parameters. In highly complex scenarios, it achieves the highest score of 24.30 on VitaBench, demonstrated strong capability in complex scenarios.

Safety LongCat-Flash showed outstanding capability in identifying and mitigating risks on the whole, particularly in the domains of Harmful and Criminal compared to other models.

[2] https://github.com/bigcode-project/bigcode-evaluation-harness
[3] https://github.com/R2E-Gym/R2E-Gym
[4] https://github.com/All-Hands-AI/OpenHands

- 编码基准测试：若模型的回答通过沙盒环境中的所有测试用例，或达到某个特定状态，则该题得分为1，否则为0。最终分数为所有题目的平均分。我们采用 OpenAI[2] 提供的脚本来评估 Humaneval+ 和 MBPP+，其余基准使用官方脚本。具体来说，对于 SWE-Bench-Verified，我们使用 R2E-Gym[3] (Openhands scraffold)，评估的迭代次数限制为 100 次，除了 DeepSeek V3.1（使用 Openhands[4]）。对于 Terminal-Bench，我们使用 Terminus 框架进行直接提示评估。

- 代理性工具使用基准：我们使用官方基准框架，以确保公平性和可重复性。对于 AceBench，我们使用直接提示而非函数调用。对于我们提出的 VitaBench，鉴于代理性任务固有的长上下文特征，我们采用滑动窗口机制在整个执行轨迹中系统地评估任务完成状态，便于对单个清单组件的完成状态进行持续更新。

4.4.2 评估结果

如表3所示，我们的综合评估表明，LongCat-Flash 是一个功能强大且多才多艺的模型。它在不同领域始终展现出领先的性能，往往在大量具有挑战性的任务中以相对较少的参数量超越同期的模型。以下分析对其在不同维度上的令人印象深刻的能力进行了详细分解。

通用领域 在通用领域知识方面，LongCat-Flash 展现出强劲且全面的性能。它在 ArenaHard-V2 上取得了 86.50 的优秀分数，在所有被评估的模型中排名第二，并在具有挑战性的直接对比中展示出其健壮的能力。在基础基准测试中，它仍然具有很强的竞争力，在 MMLU 上得分 89.71，在 CEval 上得分 90.44。这些结果可与领先模型相媲美，且显著地比 DeepSeek-V3.1 与 Kimi-K2 等竞争对手使用的参数更少，表明具有较高的效率。

Instruction Following LongCat-Flash 展示了最先进的指令遵循能力。它在 IFEval 上取得了最高分 89.65，超越所有其他模型，并在遵循复杂而细致的指令方面展现出卓越的可靠性。此外，它在 COLLIE（57.10）和 Meeseeks-zh（43.03）上也取得了最佳分数，凸显了其在英语和中文两种语言中对多样且具有挑战性的指令集的卓越能力。

在数学推理方面，LongCat-Flash 展现出强大且先进的能力。尽管在 MATH500（96.40）上的得分非常出色，其优势在更复杂、竞赛级基准测试中尤为明显。它在 AIME25（61.25）和 BeyondAIME（43.00）上交出出色、顶尖的结果，在这些具有挑战性的领域中跻身表现最佳的模型之列。这凸显了它在复杂、多步逻辑推理与问题解决方面的先进能力。翻译文本:

通用推理　对于一般推理任务，LongCat-Flash 的表现也相当稳健。它在结构化逻辑推理方面展现出卓越的实力，在 ZebraLogic 上取得了 89.30 分，位列顶尖竞争者之列。它在阅读理解基准 DROP 上也取得了具有竞争力的 79.06 分。相反，GPQA-diamond（73.23）和GraphWalks（51.05）上的结果表明还有进一步改进的空间，特别是在提升其分析极长上下文中的结构化数据的能力方面。

Coding LongCat-Flash 在编码领域展现出一个有前景且能力出众的形象。它在 TerminalBench 上的突出表现是取得了 39.51 的分数，排名第二，并在实际的、具自主性的命令行任务中展现出卓越的熟练度。它在 SWE-Bench-Verified 基准测试中也具备竞争力，得分为 60.4。在基础代码生成任务，如 Humaneval+ 和 MBPP+，其表现稳健，但未来仍有进一步优化的潜力，以达到与领先模型相一致的水平。

Agentic Tool Use LongCat-Flash 在代理性工具使用领域显示出显著优势，在 τ^2-Bench 上甚至明显优于其他模型，即便与参数量更大的模型相比亦然。在高度复杂的场景中，它在 VitaBench 上取得最高分 24.30，展现出在复杂场景中的强大能力。

总体上，Safety LongCat-Flash 在识别和缓解风险方面表现出色，尤其是在有害和犯罪领域，与其他模型相比。

[2] https://github.com/bigcode-project/bigcode-evaluation-harness
[3] https://github.com/R2E-Gym/R2E-Gym
[4] https://github.com/All-Hands-AI/OpenHands

Table 3: Evaluation results of frontier chat models. Values marked with * are sourced from other public reports. Note that DeepSeek-V3.1, Qwen3-235B-A22B, Gemini2.5-Flash, and Claude4-Sonnet are evaluated under their non-thinking mode.

Benchmark	DeepSeek V3.1	Qwen3 MoE-2507	Kimi-K2	GPT-4.1	Claude4 Sonnet	Gemini2.5 Flash	LongCat-Flash
Architecture	MoE	MoE	MoE	-	-	-	MoE
# Total Params	671B	235B	1043B	-	-	-	560B
# Activated Params	37B	22B	32B	-	-	-	27B
General Domains							
MMLU (acc)	90.96	90.23	89.86	89.64	91.75	86.33	89.71
MMLU-Pro (acc)	84.45	84.83	82.06	81.72	83.74	81.95	82.68
ArenaHard-V2 (acc)	84.10	88.20	85.70	61.50	62.10	77.00	86.50
CEval (acc)	89.21	92.70	91.26	79.53	86.63	78.78	90.44
CMMLU (acc)	88.04	88.14	89.66	77.65	86.51	78.30	84.34
Instruction Following							
IFEval (acc)	86.69	88.54	88.91	85.58	88.35	83.92	89.65
COLLIE (acc)	43.80	49.71	56.34	50.00	51.22	48.60	57.10
Meeseeks-zh (acc)	33.83	35.32	42.79	41.54	35.07	34.84	43.03
Mathematical Reasoning							
MATH500 (acc)	96.08	98.80	97.60	90.60	93.80	98.40	96.40
AIME24 (avg@10)	66.30*	81.67	69.60*	47.00	47.00	79.67	70.42
AIME25 (avg@10)	49.27	68.33	50.66	32.00	37.00	67.33	61.25
BeyondAIME (avg@10)	36.50	57.60	36.60	22.10	20.50	44.20	43.00
General Reasoning							
GPQA-diamond (acc)	74.90*	77.43	75.76	67.68	70.71	80.30	73.23
DROP (f1)	84.19	78.57	89.04	66.94	73.06	45.03	79.06
ZebraLogic (acc)	85.30	94.22	89.11	56.30*	75.85	51.78	89.30
GraphWalks-128k (precision)	73.54	80.72	47.50	85.02	80.57	64.83	51.05
Coding							
LiveCodeBench (pass@1)	56.40*	46.48	46.70	39.21	45.59	39.65	48.02
Humaneval+ (pass@1)	92.68	94.51	85.98	93.29	94.51	87.80	88.41
MBPP+ (pass@1)	79.89	79.89	81.75	79.37	80.16	76.19	79.63
SWE-Bench-Verified (acc)	66.00*	42.00	64.60	48.60	68.00*	40.60	60.40
TerminalBench (acc)	31.30*	17.28	25.93	28.40	40.74	12.35	39.51
Agentic Tool Use							
τ^2-Bench (telecom) (avg@4)	38.50	22.50	67.50	35.20	46.20	16.50	73.68
τ^2-Bench (airline) (avg@4)	46.00	36.00	54.20	56.00	60.00	41.50	58.00
τ^2-Bench (retail) (avg@4)	64.90	70.50	70.80	74.10	80.00	64.80	71.27
AceBench (acc)	69.70	71.10	82.20	80.10*	76.20*	74.50*	76.10
VitaBench (avg@4)	20.30	8.50	18.20	19.00	23.00	8.00	24.30
Safety							
Harmful	82.79	80.82	53.91	56.19	66.56	-	83.98
Criminal	87.83	89.13	77.19	81.58	87.58	-	91.24
Misinformation	83.17	77.76	42.68	45.49	54.91	-	81.72
Privacy	98.80	98.80	96.39	98.80	100.00	-	93.98

表3：前沿对话模型的评估结果。标注为 * 的数值来自其他公开报告。注意，DeepSeek-V3.1、Qwen3-235B-A22B、Gemini2.5-Flash 和 Claude4-Sonnet 在它们的非思考模式下进行评估。

Benchmark	DeepSeek V3.1	Qwen3 MoE-2507	Kimi-K2	GPT-4.1	Claude4 Sonnet	Gemini2.5 Flash	LongCat-Flash
Architecture	MoE	MoE	MoE	-	-	-	MoE
# Total Params	671B	235B	1043B	-	-	-	560B
# Activated Params	37B	22B	32B	-	-	-	27B
General Domains							
MMLU (acc)	90.96	90.23	89.86	89.64	91.75	86.33	89.71
MMLU-Pro (acc)	84.45	84.83	82.06	81.72	83.74	81.95	82.68
ArenaHard-V2 (acc)	84.10	88.20	85.70	61.50	62.10	77.00	86.50
CEval (acc)	89.21	92.70	91.26	79.53	86.63	78.78	90.44
CMMLU (acc)	88.04	88.14	89.66	77.65	86.51	78.30	84.34
Instruction Following							
IFEval (acc)	86.69	88.54	88.91	85.58	88.35	83.92	89.65
COLLIE (acc)	43.80	49.71	56.34	50.00	51.22	48.60	57.10
Meeseeks-zh (acc)	33.83	35.32	42.79	41.54	35.07	34.84	43.03
Mathematical Reasoning							
MATH500 (acc)	96.08	98.80	97.60	90.60	93.80	98.40	96.40
AIME24 (avg@10)	66.30*	81.67	69.60*	47.00	47.00	79.67	70.42
AIME25 (avg@10)	49.27	68.33	50.66	32.00	37.00	67.33	61.25
BeyondAIME (avg@10)	36.50	57.60	36.60	22.10	20.50	44.20	43.00
General Reasoning							
GPQA-diamond (acc)	74.90*	77.43	75.76	67.68	70.71	80.30	73.23
DROP (f1)	84.19	78.57	89.04	66.94	73.06	45.03	79.06
ZebraLogic (acc)	85.30	94.22	89.11	56.30*	75.85	51.78	89.30
GraphWalks-128k (precision)	73.54	80.72	47.50	85.02	80.57	64.83	51.05
Coding							
LiveCodeBench (pass@1)	56.40*	46.48	46.70	39.21	45.59	39.65	48.02
Humaneval+ (pass@1)	92.68	94.51	85.98	93.29	94.51	87.80	88.41
MBPP+ (pass@1)	79.89	79.89	81.75	79.37	80.16	76.19	79.63
SWE-Bench-Verified (acc)	66.00*	42.00	64.60	48.60	68.00*	40.60	60.40
TerminalBench (acc)	31.30*	17.28	25.93	28.40	40.74	12.35	39.51
Agentic Tool Use							
τ^2-Bench (telecom) (avg@4)	38.50	22.50	67.50	35.20	46.20	16.50	73.68
τ^2-Bench (airline) (avg@4)	46.00	36.00	54.20	56.00	60.00	41.50	58.00
τ^2-Bench (retail) (avg@4)	64.90	70.50	70.80	74.10	80.00	64.80	71.27
AceBench (acc)	69.70	71.10	82.20	80.10*	76.20*	74.50*	76.10
VitaBench (avg@4)	20.30	8.50	18.20	19.00	23.00	8.00	24.30
Safety							
Harmful	82.79	80.82	53.91	56.19	66.56	-	83.98
Criminal	87.83	89.13	77.19	81.58	87.58	-	91.24
Misinformation	83.17	77.76	42.68	45.49	54.91	-	81.72
Privacy	98.80	98.80	96.39	98.80	100.00	-	93.98

5 Training Infrastructures

The core design principle of our training infrastructure is scalability with precision. We developed a systematic method to verify operator precision and embedded online Silent Data Corruption (SDC) detection into idle computation phases to minimize numerical errors. To guarantee reproducibility and ensure consistent results between small-scale experiments and full-scale training, we enforced determinism across all computation and communication operators. This enabled bitwise-aligned loss values across multiple re-runs of any training step.

With correctness ensured, we focused on accelerating training efficiency. Wall-clock time is critical for rapid algorithm iteration, yet single accelerator provides limited capability. We therefore scaled training across tens of thousands of accelerators, confronting challenges in scalability and stability. Through model–system co-design, multi-dimensional parallelism, and fully automated fault detection and recovery, we achieved near-linear scaling and 98.48% availability, completing training within 30 days.

5.1 Numerical Precision Control and Fault Detection

ULP Evaluation Floating-point errors are influenced by multiple factors, even varying between accelerators of the same vendor across generations. To quantify and mitigate these errors, we adopt ULP (Unit in the Last Place) as a metric, where ULP error measures the deviation of accelerator BF16 results from CPU FP32 ground truth. A zero ULP error indicates perfect accuracy, while larger values imply worse precision. We collect all operator types and shapes used in training and compare their ULP errors. Table 4 shows the ULP error for GEMM between two solutions.

Table 4: GEMM Precision Comparison (ULP)

			Solution 1		Solution 2	
Case	Output Shape	Value Range	Max	Min	Max	Min
1	[1024,1536]	[-5,5]	2292	-568	112	-100
2	[1024,576]	[-5,5]	65362	-82046	6.5	-9
3	[1024,16384]	[-19,15]	544	-104	224	-112
4	[1024,12288]	[-4,4]	202	-88	72	-41
5	[1024,6144]	[-1,1]	5376	-1376	304	-224
6	[1024,24576]	[-5,5]	7200	-510	104	-294
7	[1024,131072]	[5,5]	8128	-6976	2528	-368
8	[1024,6144]	[-1,1]	5344	-8064	80	-258

SDC Detection Mechanism SDC faults are typically unavoidable in large-scale training and can severely degrade model performance by altering data without system warnings. To address this, we implement an efficient on-chip in-place operator recomputation mechanism. Specifically, we find that the backward computation for FlashAttention Gradients (FAG) is most sensitive to SDC because it simultaneously mixes tensor and vector computations. Bitwise differences between recomputed results indicate potential SDC risks. The detection computations are orchestrated within compute streams, and the recomputation interval is manually adjustable, enabling a flexible trade-off between detection coverage and computational cost.

Notably, operator precision control is necessary but insufficient for ensuring model accuracy. Experiments with different operator implementations may show training loss discrepancies within 1e-3$\sim$1e-4 yet exhibit larger than 5 pp variation on benchmarks. Cost-effectively evaluating the impact of operator precision errors on model performance remains an open challenge.

5.2 Kernel Optimization for Determinism and Performance

Determinism serves as the gold standard for computational correctness, eliminating floating-point errors as experimental variables. However, achieving determinism often incurs significant performance overhead. We address this through kernel redesigns, maintaining deterministic computation and communication throughout LongCat-Flash's training.

Deterministic FAG The default FAG implementation is non-deterministic because dQ, dK, and dV are reduced along different dimensions, where atomic addition lacks order preservation. We develop an efficient deterministic FAG kernel using limited extra workspace to accumulate tiles in a deterministic order. With co-optimizations including double-buffer pipelining, tuned tiling schedules, and load balancing, our implementation achieves 1.6x the performance of the original deterministic version and 0.95x that of the non-deterministic version, striking a balance between determinism and efficiency.

5 种训练基础设施

我们训练基础设施的核心设计原则是实现高精度的可扩展性。我们开发了一套系统化的方法来验证算子精度，并将在线静默数据损坏（SDC）检测嵌入到空闲计算阶段，以尽量减少数值误差。为了保证可重复性并确保小规模实验与全规模训练之间结果的一致性，我们对所有计算和通信算子强制实现确定性。这使得在任意训练步骤的多次重新运行中，损失值在位级上保持对齐。

在确保正确性的前提下，我们专注于提升训练效率。墙钟时间对于快速算法迭代至关重要，但单个加速器的能力有限。因此，我们将训练规模扩大到数万台加速器，面临可扩展性和稳定性方面的挑战。通过模型与系统协同设计、多维并行以及全自动故障检测与恢复，我们实现了接近线性扩展和98.48%的可用性，并在30天内完成训练。

5.1 数值精度控制与故障检测

ULP 评估　浮点误差受多种因素影响，甚至在同一厂商、跨代的加速器之间也会有所差异。为量化并减小这些误差，我们采用 ULP（单位在最后一位）作为度量指标，其中 ULP 误差衡量加速器 BF16 结果与 CPU FP32 基准真值之间的偏差。ULP 误差为零表示完全准确，而更大的数值则意味着精度更差。我们收集训练中使用的所有算子类型及形状，并比较它们的 ULP 误差。表 4 展示了两种解之间在 GEMM 运算上的 ULP 误差。

表 4：GEMM 精度比较（ULP）

			Solution 1		Solution 2	
Case	Output Shape	Value Range	Max	Min	Max	Min
1	[1024,1536]	[-5,5]	2292	-568	112	-100
2	[1024,576]	[-5,5]	65362	-82046	6.5	-9
3	[1024,16384]	[-19,15]	544	-104	224	-112
4	[1024,12288]	[-4,4]	202	-88	72	-41
5	[1024,6144]	[-1,1]	5376	-1376	304	-224
6	[1024,24576]	[-5,5]	7200	-510	104	-294
7	[1024,131072]	[5,5]	8128	-6976	2528	-368
8	[1024,6144]	[-1,1]	5344	-8064	80	-258

SDC 检测机制 SDC 故障在大规模训练中通常是不可避免的，并且可能通过在没有系统警告的情况下改变数据来严重降低模型性能。为了解决这一点，我们实现了一个高效的片上就地算子重新计算机制。具体而言，我们发现 FlashAttention 梯度（FAG）的反向计算对 SDC 最为敏感，因为它同时混合张量与向量运算。重新计算结果之间的按位差异表明潜在的 SDC 风险。检测计算在计算流中进行编排，重新计算的间隔可以手动调节，从而在检测覆盖率和计算成本之间实现灵活的权衡。

值得注意的是，算子精度控制是必要的，但不足以确保模型的准确性。对不同算子实现的实验可能在1e-3∼1e-4范围内显示训练损失差异，但在基准测试中却表现出超过5个百分点的变动。以成本效益的方式评估算子精度误差对模型性能的影响仍然是一个尚未解决的挑战。

5.2 确定性与性能的内核优化

确定性是计算正确性的金标准，将浮点误差排除在实验变量之外。然而，实现确定性往往会带来显著的性能开销。我们通过内核重新设计来解决这一问题，在 LongCat-Flash 的训练过程中保持确定性计算与通信。

确定性 FAG　默认的FAG 实现是非确定性的，因为 dQ、dK 和 dV 在不同维度上被归约，而原子加法不具备保持顺序的特性。我们开发了一个高效的确定性 FAG 内核，使用有限的额外工作区以确定的顺序累积瓦片。结合双缓冲流水线、调优的瓦片调度和负载均衡等协同优化，我们的实现将原始确定性版本的性能提升至 1.6 倍，且达到非确定性版本性能的 0.95 倍，在确定性与效率之间取得平衡。

Deterministic ScatterAdd ScatterAdd in backward passes is essential for gradient aggregation but suffers from input-output operand counts mismatches. The default implementation enforces sequential execution within a single compute unit, causing up to 50x slowdown. We propose a hierarchical reduction algorithm that parallelizes gradient aggregation across all available processors, achieving performance parity with the non-deterministic version.

Optimized Grouped GEMM Grouped GEMM's performance is critical given its high computational volume but low compute density versus dense GEMM. We optimize it via: (1) Double-buffer pipelining to overlap computation, memory I/O, and epilogue; (2) Diagonal tiling to mitigate L2 cache conflicts; (3) HBM bandwidth control via compute unit limits to overlap Grouped GEMM with dispatch/combine communication. These optimizations yield 5%–45% speedups over the default version.

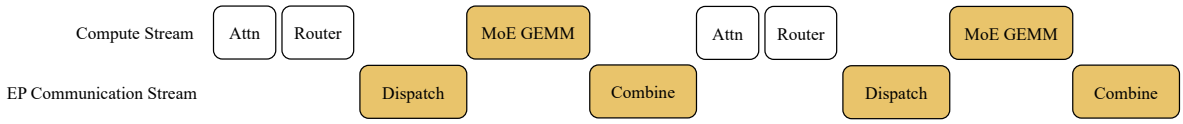
Fused GemmAdd The dw computation suffers bandwidth-bound bottlenecks during gradient accumulation. We fuse FP32 addition into the GEMM epilogue, avoiding intermediate write-backs and hiding addition within tile GEMM pipelines. This significantly reduces latency and eliminates the precision loss caused by the conversion of BF16 data to HBM, achieving a speedup of 3.12x to 3.86x on the fused GroupedGemmAdd benchmark.

Furthermore, we re-implement IO-bound kernels (e.g., MoE layer permute/unpermute) with integrated functionalities like drop-token and zero-computation experts handling, ensuring both determinism and performance.

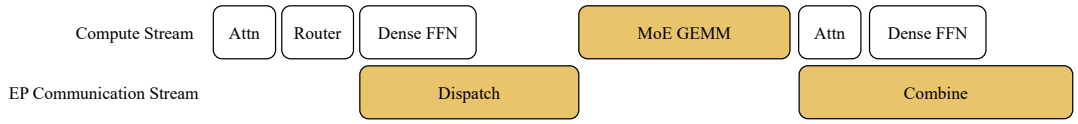
5.3 Distributed Strategy for Large-scale Training

The training architecture is centered on Expert Parallelism Groups (EP), each comprising 32 accelerators. Within an EP Group, the attention layer employs Context Parallelism (CP=8) instead of Tensor Parallelism (TP) to minimize communication overhead, and the FFN layer uses EP partitioning without TP. Multiple EP groups are scaled across Pipeline Parallelism (PP) and Data Parallelism (DP) dimensions.

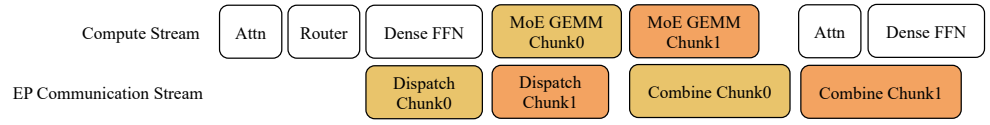
Expert parallelism (EP) is adopted to reduce static memory usage, including weights and optimizer states. However, EP inherently introduces costly dispatch and combine communication operations. To mitigate this, LongCat-Flash adopts the ScMoE structure, which enables dispatch/combine communication to overlap by more computation in a single batch. Furthermore, the MoE layer is divided into two chunks along the token dimension. These subchunks achieve two objectives: (1) Overlap with the dense FFN computation. (2) Overlap with each other (see Figure 8).



(a) Two Layers of Typical MoE



(b) ScMoE Layer



(c) ScMoE Layer with Chunk

Figure 8: These architectures have the same total and activated number of experts. ScMoE with chunk achieves the highest efficiency because more communication is overlapped by computation.

There are two optimized strategies for dispatch/combine communication: (1) All-gather/reduce-scatter kernel with pipeline in the intranode and the internode; (2) Optimized all-to-all kernel. The native all-to-all expands the local data size by top-k times, increasing traffic through the 200Gb/s per accelerator RDMA network. Additionally, all-to-all performance is unstable due to inadequate congestion control. We select pipelined all-gather/reduce-scatter with

确定性 ScatterAdd 在反向传播中对于梯度聚合至关重要，但存在输入输出操作数数量不匹配的问题。默认实现强制在单个计算单元内进行顺序执行，导致高达50倍的性能下降。我们提出一种分层规约算法，在所有可用处理器上对梯度聚合进行并行化，达到与非确定性版本相同的性能水平。

优化后的分组 GEMM 的性能至关重要，因为其计算量很大，但相较于密集 GEMM 的计算密度较低。我们通过以下方法对其进行优化： (1) 双缓冲流水线以实现计算、内存 I/O 与后处理阶段的重叠； (2) 对角线切块以降低 L2 缓存冲突； (3) 通过对计算单元的限制来控制 HBM 带宽，以使分组 GEMM 与派发/合并通信重叠。 这些优化相比默认版本带来 5%–45% 的加速。

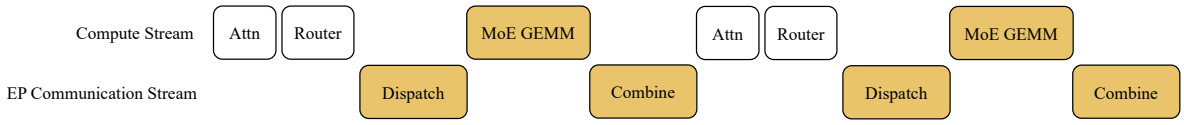
GemmAdd 融合dw 计算在梯度累积过程中遭遇带宽瓶颈。我们将 FP32 加法融入 GEMM 尾部计算，避免中间写回，并将加法隐藏在分块 GEMM 流水线中。这显著降低了延迟，消除了由于 BF16 数据转换为 HBM 而导致的精度损失，在融合的 GroupedGemmAdd 基准测试中实现了 3.12x 到 3.86x 的加速。

此外，我们重新实现 I/O 受限的内核（例如 MoE 层的置换/逆置换），并整合诸如丢弃标记和对零计算专家的处理等功能，确保确定性和性能。

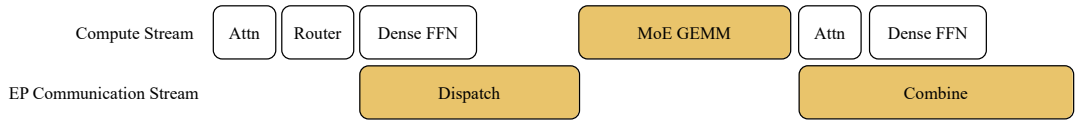
5.3 大规模训练的分布式策略

训练架构以 Expert Parallelism Groups (EP) 为核心，每组包含 32 个加速器。在一个 EP 组内，注意力层采用 Context Parallelism (CP=8) 而非 Tensor Parallelism (TP) 以最小化通信开销，且 FFN 层使用 EP 分区而不使用 TP。多个 EP 组在 Pipeline Parallelism (PP) 与 Data Parallelism (DP) 维度上进行扩展。

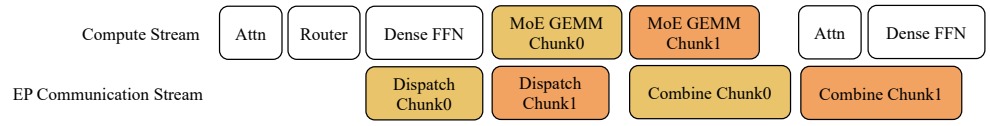
专家并行性（EP）被采用以降低包括权重和优化器状态在内的静态内存占用。然而，EP 本质上会带来成本高昂的派发与合并通信操作。为缓解这一点，LongCat-Flash 采用 ScMoE 结构，该结构通过在单个批次中进行更多计算来实现派发/合并通信的重叠。此外，MoE 层沿 token 维度被分成两个子块。这些子块实现了两个目标：(1) 与密集 FFN 计算重叠；(2) 彼此之间的重叠（见图 8）。



(a) Two Layers of Typical MoE



(b) ScMoE Layer



(c) ScMoE Layer with Chunk

图 8：这些架构在专家的总数和激活数量方面相同。带分块的 ScMoE 实现了最高效率，因为有更多的通信被计算所重叠。

有两种用于调度/聚合通信的优化策略： (1) 在节点内和节点间具流水线的 All-gather/reduce-scatter 内核； (2) 优化后的 all-to-all 内核。原生的 all-to-all 将本地数据量扩大到 top-k 倍，增加通过每个加速器 RDMA 网络（带宽为 200Gb/s）的流量。此外，由于拥塞控制不足，all-to-all 的性能不稳定。我们选择带流水线的 all-gather/reduce-scatter 与

deterministic as the primary solution, the proportion of time to non-overlapping dispatch/combine communication decreases from 25.3% to 8.4% with ScMoE architecture.

Existing pipeline strategies (e.g., 1F1B, interleaved-1F1B, Zero-bubble [Qi and Others, 2023]) suffer from imbalanced memory usage across pipeline stages. To this end, we adopt the V-ZB algorithm [Qi et al., 2024], which balances memory usage at all stages and reduces peak memory to less than 60GB in the training of LongCat-Flash. Additionally, we enable the post-validation strategy from zero bubble, achieving zero theoretical bubbles. A key refinement is replacing inverse operations with backup data from the previous step during optimizer state rollback, preserving numerical bitwise alignment.

5.4 Reliability and Observability

Reliability is measured by the proportion of time contributing to the final training trajectory (Availability), where unavailable time includes fault recovery and wasted time between the last checkpoint and fault occurrence. Asynchronous checkpointing reduces training stall to 2∼4 seconds, allowing higher frequency and minimizing fault-induced loss. Combined with online critical log filtering, optimized initialization, and full automation, recovery time is reduced to <10 minutes. These mechanisms achieve 98.48% availability, with all 20 faults handled automatically without manual intervention.

Observability combines fine- and coarse-grained profiling with a metric platform. Fine-grained PyTorch profiler timelines enable distributed, parallel-aware co-analysis to identify pipeline parallelism "bubbles" and inter-rank communication waits. Coarse-grained monitoring adds low-overhead runtime analysis of stragglers. The metric platform tracks loss, weights, gradients, and activations for rapid model state assessment.

6 Inference and Deployment

LongCat-Flash employs a model-system co-design, which significantly contributes to its high throughput and low latency. This section focuses on inference optimizations implemented in one of our deployment clusters, presenting methods to simultaneously boost system throughput and significantly reduce latency to 100 TPS on H800. We first present our parallel inference architecture co-designed with the model architecture. Following the inference architecture, optimization methods such as quantization and custom kernel are described. Finally, we present our deployment strategy and performance results.

6.1 Model-Specific Inference Optimization

To achieve an efficient inference system, two key challenges must be addressed: (1) Computation and communication orchestration, and (2) KV cache I/O and storage. For the first challenge, existing approaches typically exploit parallelism at three conventional granularities: operator-level overlap like NanoFlow [Zhu et al., 2025], expert-level overlap represented by EPS-MoE [Qian et al., 2025], and layer-level overlap demonstrated in DeepSeek-V3 TBO (Two Batch Overlap) [Team, 2025b]. LongCat-Flash's ScMoE architecture introduces a fourth dimension—module-level overlap—for which we designed the SBO (Single Batch Overlap) scheduling strategy to optimize both latency and throughput. For the second challenge—KV cache I/O and storage—LongCat-Flash addresses these issues through architectural innovations in its attention mechanism and MTP structure to reduce the effective I/O overhead.

6.1.1 Computation and Communication Orchestration

LongCat-Flash naturally exhibits computation-communication overlap properties in its structure, which is the key to achieving lower latency while maintaining generation throughput. We carefully design Single Batch Overlap (SBO), a four-stage pipeline execution that uses module-level overlap to fully unleash LongCat-Flash's potential as shown in Figure 9. SBO differs from TBO by hiding communication overhead within a single batch. In SBO, stage 1 requires separate execution because the MLA output serves as input for subsequent stages. In stage 2, we overlap all-to-all dispatch with Dense FFN and Attn 0 (QKV Projection). This overlap is crucial because communication overhead is excessive, prompting us to split the attention process. Stage 3 independently executes MoE GEMM. The latency of this stage will benefit from the wide EP deployment strategy. In stage 4, we overlap Attn 1 (Core Attention and Output Projection) and Dense FFN with the all-to-all combine. This orchestration effectively mitigates the communication overhead, ensuring efficient inference for LongCat-Flash.

Additionally, the ScMoE architecture, under the wide EP deployment scheme, facilitates the overlap of intra-node NVLink bandwidth utilization and inter-node RDMA communication through GPUDirect RDMA [Choquette, 2022], thereby improving overall bandwidth efficiency. Dense FFN in ScMoE has a relatively large intermediate size, so TP

以确定性作为主要解决方案，在 ScMoE 架构下，用于非重叠派发/合并通信的时间比例从 25.3% 降低到 8.4%。

现有的流水线策略（例如 1F1B、交错-1F1B、Zero-bubble [Qi and Others, 2023]）在各流水线阶段的内存使用存在不平衡的问题。为此，我们采用 V-ZB 算法 [Qi et al., 2024]，该算法在所有阶段实现内存使用的平衡，并使 LongCat-Flash 的训练峰值内存降至低于 60GB。此外，我们启用来自 Zero-bubble 的后验证策略，实现理论上的零气泡。一个关键改进是在优化器状态回滚期间，用上一步的备份数据替代逆运算，以保持数值的位级对齐。

5.4 可靠性与可观测性

可靠性通过对最终训练轨迹贡献的时间比例来衡量（可用性），其中不可用时间包括故障恢复以及上一次检查点与故障发生之间的浪费时间。异步检查点将训练停滞时间降低到 2∼4 秒，从而实现更高的检查点频率并最小化故障引起的损失。结合在线关键日志过滤、优化的初始化和全自动化，恢复时间降低到 <10 分钟。这些机制实现了 98.48% 的可用性，所有 20 次故障均自动处理，无需人工干预。

可观测性将细粒度和粗粒度的分析与指标平台结合起来。细粒度的 PyTorch 性能分析时间线使系统能够进行分布式、具并行感知能力的协同分析，以识别管道并行中的“气泡”和秩间通信等待。粗粒度监控增加了对慢任务的低开销运行时分析。指标平台跟踪损失、权重、梯度和激活，以快速评估模型状态。

6 推断与部署

LongCat-Flash 采用模型-系统协同设计，这显著提升了其吞吐量并降低了延迟。本节聚焦于在我们其中一个部署集群中实现的推理优化，介绍在 H800 上能够同时提升系统吞吐量并显著降低延迟至 100 TPS 的方法。我们首先介绍与模型架构共同设计的并行推理架构。在推理架构之后，描述了诸如量化和自定义内核等优化方法。最后，我们介绍我们的部署策略和性能结果。

6.1 模型特定推理优化

为实现高效的推理系统，需要解决两个关键挑战：（1）计算与通信的编排，以及（2）KV 缓存的 I/O 与存储。对于第一个挑战，现有方法通常在三种常规粒度上利用并行性：操作级重叠，如 NanoFlow [Zhu et al., 2025]，专家级重叠由 EPS-MoE [Qian et al., 2025] 表示，以及层级重叠，在 DeepSeek-V3 TBO（Two Batch Overlap）[Team, 2025b] 中得到展示。LongCat-Flash 的 ScMoE 架构引入了第四个维度——模块级重叠（module-level overlap），为此我们设计了 SBO（Single Batch Overlap）调度策略，以同时优化延迟和吞吐量。对于第二个挑战——KV 缓存的 I/O 与存储——LongCat-Flash 通过在其注意力机制和 MTP 结构方面的架构创新来降低有效的 I/O 开销。

6.1.1 计算与通信编排

LongCat-Flash 自然地展示了其结构中的计算-通信重叠特性，这是在保持生成吞吐量的同时实现更低延迟的关键。我们仔细设计了单批次重叠（SBO，Single Batch Overlap），这是一个采用模块级重叠的四阶段流水线执行，如图9所示，充分释放 LongCat-Flash 的潜力。SBO 与 TBO 的区别在于将通信开销隐藏在单个批次中。在 SBO 的阶段1中由于 MLA 输出作为后续阶段的输入，因此需要单独执行。在阶段2中，我们将全对全调度与 Dense FFN 和 Attn 0（QKV 投影）重叠执行。这种重叠至关重要，因为通信开销过大，促使我们将注意力过程拆分。阶段3 独立执行 MoE GEMM。该阶段的延迟将受益于大规模 EP 部署策略。在阶段4中，我们将 Attn 1（核心注意力与输出投影）与 Dense FFN 与全对全合并进行重叠编排。这种编排有效缓解了通信开销，确保 LongCat-Flash 的高效推理。

此外，在广泛的 EP 部署方案下，ScMoE 架构促进了同节点 NVLink 带宽利用与跨节点 RDMA 通信通过 GPUDirect RDMA [Choquette, 2022] 实现的重叠，从而提高整体带宽利用效率。ScMoE 中的 Dense FFN 具有相对较大的中间大小，因此 TP

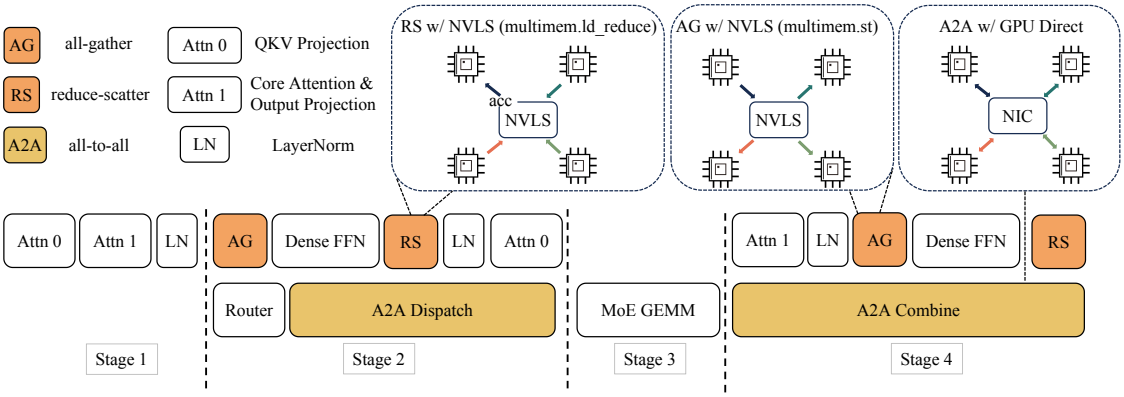


Figure 9: An overview of overlapping strategy.

deployment is employed to minimize memory footprint, necessitating all-gather and reduce-scatter communication before and after Dense FFN, respectively. To reduce this communication overhead, we develop custom kernels and adopt TP2 or TP4 instead of TP8.

6.1.2 Speculative Decoding

LongCat-Flash employs MTP as the draft model for speculative decoding. Our optimization framework originates from a systematic breakdown of Speculative Decoding's speedup formulation, as Sadhukhan et al. [2025] has mentioned:

$$\frac{T_{Avg}^{SD}}{T_T} = \frac{1}{\Omega(\gamma,\alpha)}\left(\frac{\gamma \cdot T_D}{T_T} + \frac{T_V(\gamma)}{T_T}\right),$$

where T_{Avg}^{SD}, T_T, T_D are expected latency per token for speculative decoding, target model and draft model. γ represents number of draft token in one decoding step. $\Omega(\gamma,\alpha)$ is expected accept length for a given step γ and acceptance rate α. And $T_V(\gamma)$ is expected latency for target verification. Our approach targets three key factors:

- Expected accept length $\Omega(\gamma,\alpha)$, which is positively correlated with the acceptance rate α of draft tokens. To maximize acceptance rate α, we employ MTP. Integrate a single MTP head during late-phase pre-training, achieving approximately 90% acceptance rate on test sets.
- Draft to target cost ratio $\gamma\frac{T_D}{T_T}$, which is dominated by the structures of both target model and draft model. As noted by Liu et al. [2024d], balancing draft quality and speed is critical. To minimize generation overhead while maintaining comparable acceptance rates, LongCat-Flash adopts a lightweight MTP architecture with reduced parameters. Our experiments (Table 5) show that a single dense layer for MTP heads optimizes this trade-off, outperforming ScMoE layers in latency.
- Target verification to decoding cost ratio $\frac{T_V(\gamma)}{T_T}$. In order to reduce this ratio, we adopt the C2T [Huo et al., 2025] method, using a classification model to filter out tokens that are unlikely to be accepted before verification.

Table 5: Draft token acceptance rate on MT-Bench of different MTP head structures with a 6B activated model. The ratio of MTP head parameters to main model parameters is also reported.

MTP layer	Activated parameters ratio	Acceptance rate α
Dense layer	1.41%	92.1%
ScMoE layer	4.17%	92.9%

6.1.3 Reducing KV Cache

To balance performance and efficiency, LongCat-Flash adopts MLA with 64 heads for its attention mechanism, which reduces the computational load of the attention component while achieves exceptional KV cache compression and thus reduces storage and bandwidth pressure. This is crucial for orchestrating LongCat-Flash's pipeline, as noted in Figure 9

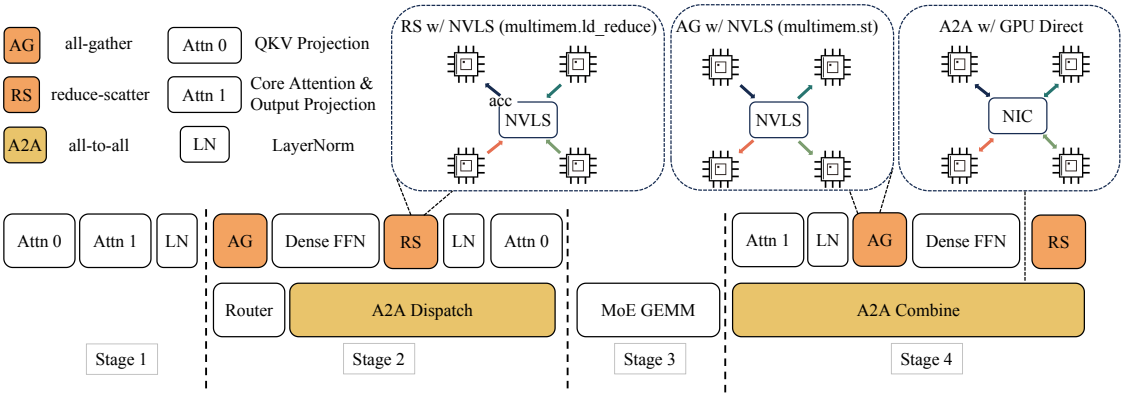


图9：重叠策略概览。

部署旨在最小化内存占用，分别在 Dense FFN 之前进行 all-gather 通信，在 Dense FFN 之后进行 reduce-scatter 通信。为降低这一通信开销，我们开发了自定义内核，并采用 TP2 或 TP4，而不是 TP8。

6.1.2 推测解码

LongCat-Flash 采用 MTP 作为推测解码的初始模型。我们的优化框架源自对推测解码加速公式的系统性分解，正如 Sadhukhan 等人 [2025] 所提及：

$$\frac{T_{Avg}^{SD}}{T_T} = \frac{1}{\Omega(\gamma,\alpha)}\left(\frac{\gamma \cdot T_D}{T_T} + \frac{T_V(\gamma)}{T_T}\right),$$

其中 T_{Avg}^{SD}、T_T、T_D 分别是推测解码、目标模型和草拟模型每个标记的期望延迟。γ 表示在一次解码步骤中的草拟标记数量。$\Omega(\gamma,\alpha)$ 表示给定步骤 γ 的期望接受长度以及接受率 α。而 $T_V(\gamma)$ 表示目标验证的期望延迟。我们的方法针对三个关键因素：

- 预期的接受长度 $\Omega(\gamma,\alpha)$，与草案标记的接受率 α 正相关。为了最大化接受率 α，我们采用 MTP。在后期阶段的预训练中整合一个单独的 MTP 头，在测试集上实现大约 90% 的接受率。
- 从草案到目标的成本比率 $\gamma\frac{T_D}{T_T}$，由目标模型和草案模型的结构共同主导。正如 Liu 等人 [2024d] 所指出，平衡草案质量与速度至关重要。为了在保持可比的接受率的同时尽量降低生成开销，LongCat-Flash 采用了一个参数更少的轻量级 MTP 架构。我们的实验（表5）表明，MTP 头的单个全连接层可以优化这一权衡，在延迟方面优于 ScMoE 层。
- 目标验证与解码成本之比 $\frac{T_V(\gamma)}{T_T}$。为了降低这一比值，我们采用 C2T [Huo 等人，2025] 方法，使用一个分类模型来过滤掉在进行验证之前不太可能被接受的词元。

表 5：在 MT-Bench 上，不同 MTP 头结构在一个 6B 已激活模型下的草案标记接受率。同时报告了 MTP 头参数与主模型参数之比。

MTP layer	Activated parameters ratio	Acceptance rate α
Dense layer	1.41%	92.1%
ScMoE layer	4.17%	92.9%

6.1.3 减少 KV 缓存

为了在性能与效率之间取得平衡，LongCat-Flash 为其注意力机制采用了 64 头的 MLA，这在降低注意力组件的计算负载的同时实现了卓越的 KV 缓存压缩，从而降低存储和带宽压力。这对于编排 LongCat-Flash 的流水线至关重要，如图 9 所示。

the model always features an attention computation that cannot be overlapped with communication. Specifically, the MQA-like structure of the MLA absorb method shares KV across the m-dimension (64 heads), aligning with the shape of the WGMMA instruction for maximal hardware utilization.

6.2 System-Wide Inference Techniques

6.2.1 Minimize Schedule Overhead

The decoding phase in LLM inference systems can become launch-bound due to kernel launch overhead. This issue is exacerbated when introducing speculative decoding—particularly with LongCat-Flash's lightweight MTP, where separate scheduling of verification kernels and draft forward passes introduces significant overhead. To mitigate this, a TVD fusing strategy is used to fuse Target forward, Verification, and Draft forward into a single CUDA graph. To further improve GPU utilization, we implement an overlapped scheduler. However, experimental results reveal that the low latency of LongCat-Flash's forward pass renders a single-step pre-schedule strategy insufficient to fully eliminate scheduling overhead. As shown in Figure 10, we introduce a multi-step overlapped scheduler to launch the kernel for multiple forward steps in a single schedule iteration. This approach effectively hides CPU scheduling and synchronization within the GPU forward process, ensuring continuous GPU occupancy.

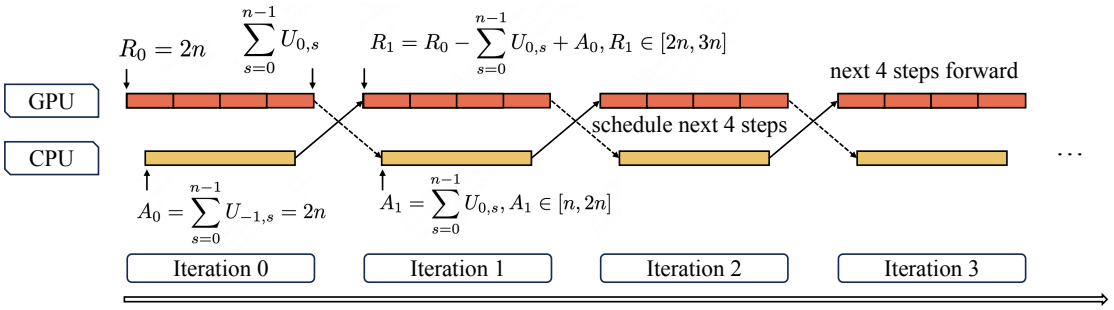


Figure 10: Multi-step overlapped scheduler (4 steps as a example here).

In a multi-step overlapped scheduler, we need to dynamically pre-allocate KV cache slots for multiple future steps without prior knowledge of the accept length of speculative decoding in previous iterations. An important issue is whether multi-step overlapped scheduling causes divergent KV cache allocation. We illustrate this with $MTP = 1$ and the number of steps, $n = 4$. Let R_i represents available KV entries during the GPU's i-th iteration forward pass, thus $R_0 = (MTP + 1) \times n = 2n$. $U_{i,s} \in [1, 2]$ represents the accept length in the i-th iteration for the s step, with the initial value $U_{-1,s} = 2$. Then, while the GPU is performing the i-th iteration of forward computation, the scheduler pre-allocates the KV cache slots needed for the $(i + 1)$-th forward iteration based on the accept length in the $(i - 1)$-th forward iteration, where A_i represents the KV cache slots that is allocated. Formally:

$$A_i = \sum_{s=0}^{n-1} U_{i-1,s},\ i \geq 0$$

$$R_i = R_{i-1} - \sum_{s=0}^{n-1} U_{i-1,s} + A_{i-1},\ i \geq 1$$

By induction, we obtain the closed-form expression:

$$R_i = 4n - \sum_{s=0}^{n-1} U_{i-1,s},\ i \geq 1$$

which means:

$$R_i \in [2n, 3n],\ i \geq 1$$

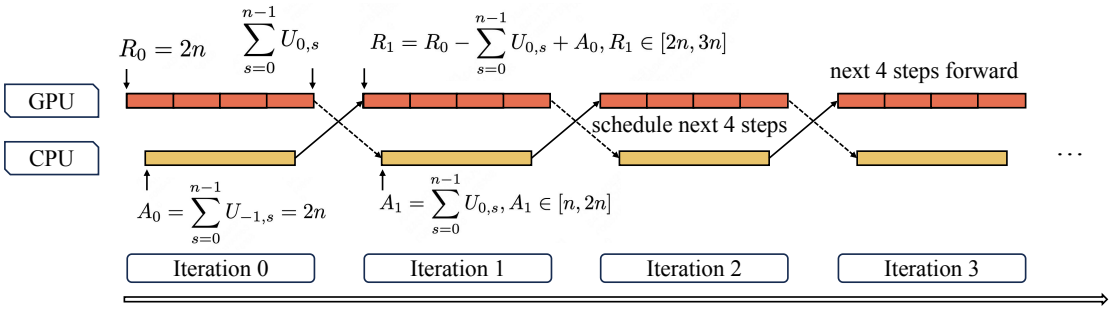
Through mathematical induction, this ensures safe KV cache allocation for the next iteration even without knowing the current iteration's accept length, while guaranteeing convergence in allocated KV cache size.

该模型始终具备一种注意力计算，无法与通信重叠。具体地，MLA absorb 方法的类似 MQA 的结构在 m 维度（64 个头）上共享 KV，与 WGMMA 指令的形状保持一致，以实现硬件利用率的最大化。

6.2 系统级推理技术

6.2.1 最小化调度开销

LLM 推理系统中的解码阶段可能因内核启动开销而成为启动瓶颈。此问题在引入推测解码时尤为突出——特别是在 LongCat-Flash 的轻量级 MTP 场景下，其中对验证内核和草拟前向传播的分离调度会带来显著的开销。为缓解此问题，采用 TVD 融合策略，将目标前向传播、验证和草拟前向传播融合成一个 CUDA 图。 为了进一步提升 GPU 的利用率，我们实现了一个重叠调度器。然而，实验结果表明，LongCat-Flash 的前向传播低延迟使得单步预调度策略不足以彻底消除调度开销。如图 10 所示，我们引入一个多步重叠调度器，在一个调度迭代中为多步前向计算启动内核。这种方法在 GPU 的前向处理中有效隐藏了 CPU 调度和同步，从而确保 GPU 的持续占用。



Fi图 10：多步重叠调度器（4 步作为一个示例 e 这里).

在一个多步重叠的调度器中，我们需要在不知道前几轮推断解码的可接受长度的情况下，动态地为多个未来步骤预分配 KV 缓存槽。一个重要的问题是多步重叠调度是否会导致 KV 缓存分配发散。我们用 $MTP = 1$ 与步骤数量 $n = 4$ 来说明这一点。设 R_i 表示在 GPU 的第 i 次前向传播迭代中可用的 KV 条目，因此 $R_0 = (MTP + 1) \times n = 2n$。$U_{i,s} \in [1, 2]$ 表示在第 i 次迭代中对第 s 步的可接受长度，初始值为 $U_{-1,s} = 2$。随后，在 GPU 执行前向计算的第 i 次迭代时，调度器基于在第 $(i-1)$-次前向迭代中的可接受长度，预分配第 $(i+1)$-次前向迭代所需的 KV 缓存槽，其中 A_i 表示已分配的 KV 缓存槽。形式上：

$$A_i = \sum_{s=0}^{n-1} U_{i-1,s},\ i \geq 0$$

$$R_i = R_{i-1} - \sum_{s=0}^{n-1} U_{i-1,s} + A_{i-1},\ i \geq 1$$

通过归纳法，我们得到闭式表达式：

$$R_i = 4n - \sum_{s=0}^{n-1} U_{i-1,s},\ i \geq 1$$

这意味着：

$$R_i \in [2n, 3n],\ i \geq 1$$

通过数学归纳法，这可以在不知道当前迭代的接受长度的情况下，为下一次迭代安全地分配 KV 缓存，同时保证所分配的 KV 缓存大小的收敛性。

6.2.2 Custom Kernel

The autoregressive nature of LLM inference creates distinct efficiency challenges. The prefilling phase is compute-bound, and methods like chunk-prefill [Agrawal et al., 2023] regularize data for optimal processing. In contrast, the decoding phase is often memory-bound due to small, irregular batch sizes from traffic patterns, which hurts kernel performance. Therefore, optimizing these specific cases is crucial for minimizing Time-Per-Output-Token (TPOT).

MoE GEMM Existing libraries like DeepGEMM [Zhao et al., 2025a] map model weights to right-hand matrices (B in A×B=C) aligned with k/n dimensions, while input activations become left-hand matrices mapped to m/k dimensions, where m represents token count. This conventional approach requires padding when token counts fall below m's 64-element minimum. To address this inefficiency, we leverage the SwapAB [Dege et al., 2025] technique: treating weights as left-hand matrices and activations as right-hand matrices. By exploiting the n-dimension's flexible 8-element granularity, SwapAB maximizes tensor core utilization.

Communication Kernels The inference system leverages NVLink Sharp's hardware-accelerated broadcast (multimem.st) and in-switch reduction (multimem.ld_reduce) to minimize data movement and SM occupancy, as shown in Figure 9. By using inline PTX assembly, the reduce-scatter and all-gather kernels enable high-efficiency data transmission. These kernels support both uniform and nonuniform token distributions across GPUs, and consistently outperform NCCL [NVIDIA] and MSCCL++ [Shah et al., 2025] across 4KB to 96MB message sizes, using only 4 thread blocks.

6.2.3 Quantization

LongCat-Flash employs the same quantization scheme as DeepSeek-V3, using fine-grained block-wise quantization: activations per [1,128] blocks and weights per [128,128] blocks. Besides, to achieve an optimal performance-accuracy trade-off, we applied layer-wise mixed-precision quantization based on two methodologies: The first scheme follows our approaches in FPTQ [Li et al., 2023b] and Super-Expert [Su et al., 2025], where we observed that certain linear layers (particularly Downproj) exhibited input activations with extreme magnitudes reaching 10^6. The second scheme involves computing block-wise FP8 quantization errors (both relative and absolute) layer by layer, which revealed significant quantization errors in specific expert layers. By taking the intersection of both schemes, we achieved substantial accuracy improvements.

6.3 Deployment and Performance

6.3.1 Measured Performance

Table 6: Performance of LongCat-Flash under different settings.

Model	Attention	Avg Context	#Hopper GPUs	TGS	TPS/u
DeepSeek-V3-profile	bf16	4096	128	2324	20
DeepSeek-V3-blog	bf16	4989	144	1850	20 ~22
LongCat-Flash	bf16	5000	128	3785	35
LongCat-Flash	bf16	5000	128	**2205**	**68.9**
LongCat-Flash	bf16	5000	128	**804**	**100.5**
LongCat-Flash	fp8	5000	128	4230	26.4
LongCat-Flash	fp8	8192	128	3240	33.8

To enable independent optimization of prefilling and decoding phases, PD-Disaggregated architecture is adopted. A key challenge in this design is the overhead of transmitting KV caches from prefilling to decoding nodes. To mitigate this, we implement layer-wise transmission, which significantly reduces Time-To-First-Token (TTFT) under high QPS workloads. For prefilling and decoding nodes, the minimum deployment unit consists of 2 nodes with 16 H800-80GB GPUs. Meanwhile, wide EP is deployed with DeepEP [Zhao et al., 2025b] to minimize communication overhead. Besides, we modify DeepEP and EPLB (Expert Parallelism Load Balancer) to support zero-computation experts, where the outputs of zero-computation experts can be obtained without communication.

Table 6 compares the throughput and latency of LongCat-Flash with DeepSeek-V3 (DeepSeek-V3-profile from DeepSeek [2025a], DeepSeek-V3-blog from DeepSeek [2025b]), where TGS (token per GPU per second) represents generation throughput per device (higher values indicate lower cost), and TPS/u (tokens per second per user) represents the generation speed for one user (higher values are better). During testing, the steady-state generation throughput

6.2.2 自定义核函数

LLM 推理的自回归特性带来了一系列显著的效率挑战。预填充阶段是计算密集型的，诸如 chunk-prefill [Agrawal 等, 2023] 的方法对数据进行正则化，以实现最优处理。相比之下，解码阶段常常受内存瓶颈制约，原因是来自流量模式的小而不规则的批量大小，这会降低内核性能。因此，针对这些特定场景的优化对于将 Time-Per-Output-Token (TPOT) 最小化至关重要。

MoE GEMM 现有的库，如 DeepGEMM [Zhao 等人，2025a] 将模型权重映射到右侧矩阵（在 A×B=C 中的 B），并与 k/n 维度对齐，而输入激活则映射为左侧矩阵，映射到 m/k 维度，其中 m 表示令牌数量。这种传统方法在令牌数量低于 m 的 64 元素下限时需要进行填充。为了解决这一低效问题，我们采用 SwapAB [Dege 等人，2025] 技术：将权重视为左侧矩阵，激活视为右侧矩阵。通过利用 n 维度的灵活 8 元素粒度，SwapAB 最大化张量核心利用率。

通信内核 推理系统利用 NVLink Sharp 的硬件加速广播（multi- mem.st）和交换内规约（multimem.ld_reduce）来尽量减少数据移动和 SM 占用率，如图 9 所示。通过使用内联 PTX 汇编，reduce-scatter 和 all-gather 内核实现高效的数据传输。这些内核支持跨 GPU 的均匀和非均匀 token 分布，并在 4KB 到 96MB 的消息大小范围内始终超过 NCCL [NVIDIA] 和 MSCCL++ [Shah 等，2025]，仅使用 4 个线程块。

6.2.3 量化

LongCat-Flash 采用与 DeepSeek-V3 相同的量化方案，使用细粒度的块级量化：激活值按 [1,128] 块量化，权重按 [128,128] 块量化。此外，为了实现最佳的性能与精度折衷，我们基于两种方法应用逐层混合精度量化：第一种方案遵循我们在 FPTQ [Li et al., 2023b] 与 Super-Expert [Su et al., 2025] 的方法，在这些方法中我们观察到某些线性层（特别是 Downproj）的输入激活值的幅度达到 10^6。第二种方案涉及逐层计算基于块的 FP8 量化误差（相对误差与绝对误差），这揭示了在某些专家层中存在显著的量化误差。通过取两种方案的交集，我们实现了显著的精度提升。

6.3 部署与性能

6.3.1 实测性能

表6：在不同设置下的 LongCat-Flash 性能。

Model	Attention	Avg Context	#Hopper GPUs	TGS	TPS/u
DeepSeek-V3-profile	bf16	4096	128	2324	20
DeepSeek-V3-blog	bf16	4989	144	1850	20 ~22
LongCat-Flash	bf16	5000	128	3785	35
LongCat-Flash	bf16	5000	128	**2205**	**68.9**
LongCat-Flash	bf16	5000	128	**804**	**100.5**
LongCat-Flash	fp8	5000	128	4230	26.4
LongCat-Flash	fp8	8192	128	3240	33.8

为实现预填充和解码阶段的独立优化，采用 PD-分离式架构。该设计的一个关键挑战在于将 KV 缓存从预填充节点传输到解码节点所产生的开销。为减轻这一点，我们实现逐层传输，在高 QPS 负载下可显著降低 Time-To-First-Token (TTFT)。对于预填充和解码节点，最小部署单元由 2 个节点组成，配备 16 个 H800-80GB GPU。与此同时，使用 DeepEP [Zhao 等人，2025b] 部署宽域 EP，以最小化通信开销。此外，我们修改 DeepEP 和 EPLB (Expert Parallelism Load Balancer，专家并行负载均衡器) 以支持零计算专家，其中零计算专家的输出无需通过通信即可获得。

表 6 比较 LongCat-Flash 与 DeepSeek-V3 的吞吐量和延迟（DeepSeek-V3-profile 来自 DeepSeek [2025a]，DeepSeek-V3-blog 来自 DeepSeek [2025b]），其中 TGS（每 GPU 每秒的令牌数）表示设备的生成吞吐量（数值越高表示成本越低），而 TPS/u（每个用户每秒的令牌数）表示单个用户的生成速度（数值越高越好）。在测试过程中，稳态生成吞吐量

under a given sequence length is used for calculation. LongCat-Flash achieves higher generation throughput and faster generation speed across different sequence lengths.

In Agent applications based on the ReACT [Yao et al., 2023] pattern, completing a single task requires multiple rounds of model interactions, where interaction latency directly impacts user experience. Analysis of typical Agent invocation patterns reveals differentiated speed requirements for model outputs:

- Reasoning content (user-visible): consisting of cognitive processes and explanations, must match human reading speed (20 tokens/s).
- Action commands (user-invisible): structured data such as function names and parameters, typically 30~100 tokens, yet directly affect tool invocation startup time—demanding the highest possible speed.

To address this scenario, LongCat-Flash achieves a generation speed of nearly 100 tokens/s for action commands. Under a cost assumption of $2 per hour for an H800 GPU, this translates to a price of $0.7 per million output tokens. This performance constrains the single-round tool-call latency to under one second, thereby significantly enhancing the interactivity of Agent applications.

6.3.2 Theoretical Performance

Figure 9 shows that LongCat-Flash's latency is primarily determined by three components:

- MLA: Its time consumption cannot be reduced by increasing the number of EP.
- All-to-all dispatch/combine: Both are constrained by single-device batch size and topk.
- MoE: Its time consumption in the memory-bound region decreases with increasing EP count.

Assuming the number of EP is 128, MLA uses DP for DeepSeek-V3 and LongCat-Flash, GQA uses TP4 for Qwen3-235B-A22B as it has 4 kv heads, and the batch size per device is 96. Actually, the GQA feature of Qwen-235B-A22B results in a relatively high memory footprint for its KV cache, making it difficult to achieve a per-GPU batch size of 96 in practice. The assumption that it can reach this value is made here solely for the purpose of theoretical analysis. As pointed out by [Jiashi Li, 2025], FlashMLA can achieve up to 660 TFlops on NVIDIA H800 SXM5 GPUs; Zhao et al. [2025b] indicates that DeepEP bandwidth can reach 40GB/s. Both of these metrics were utilized in our computations. Assuming the cost for per H800 is $2 per hour. Considering MTP=1 with an acceptance rate of 80%, we can calculate the theoretical time consumption and cost of each module in one layer of DeepSeek-V3, Qwen3-235B-A22B and LongCat-Flash, as listed in Table 7. For Qwen3-235B-A22B, which does not natively support MTP, we assume a speculative sampling strategy with a comparable acceptance rate.

Table 7: Theoretical decoding time and cost of different models.

	DeepSeek-V3	Qwen3-235B-A22B	LongCat-Flash
MTP	w/	w/o	w/
n_layer	61	94	28
batch per device	96	96	96
			Time cost of different modules in one layer
attention	471 us	314 us	264 us
all-to-all dispatch	275 us	157 us	236 us
MoE	77 us	29 us	60 us
all-to-all combine	551 us	315 us	472 us
			TPOT and Price
overlap strategy	TBO	TBO	SBO
TPOT (ms)	30	26.2	16
$/1M output token	0.17	0.15	0.09

Under this configuration, the theoretical extreme TPOT for LongCat-Flash with SBO can be expressed as:

$$\mathrm{TPL} = 264 + 236 + 60 + 472 = 1032\ us,$$

$$\mathrm{TPOT} = \frac{28 \times \mathrm{TPL}}{1000 \times 1.8} \approx 16\ \mathrm{ms},$$

在给定的序列长度下用于计算。LongCat-Flash 在不同序列长度下实现了更高的生成吞吐量和更快的生成速度。

在基于 ReACT [Yao et al., 2023] 模式的智能体应用中，完成一个任务需要多轮模型交互，其中交互延迟直接影响用户体验。对典型智能体调用模式的分析揭示了模型输出在速度方面的差异化要求：

- 推理内容（对用户可见）：由认知过程和解释组成，必须与人类阅读速度相匹配（每秒 20 个词元）。
- 操作命令（用户不可见）：结构化数据，如函数名和参数，通常为 30~100 个标记，但会直接影响工具调用的启动时间——需要达到尽可能高的速度。

为了解决这一场景，LongCat-Flash 实现了约每秒 100 个令牌的行动命令生成速度。在假设 H800 GPU 的每小时成本为 2 美元的前提下，这相当于每百万输出令牌 0.7 美元。这一性能将单轮工具调用延迟限制在一秒内，从而显著提升 Agent 应用的互动性。

6.3.2 理论性能

图9显示，LongCat-Flash 的延迟主要由三个组成部分决定：

- MLA：通过增加 EP 的数量，无法降低其时间消耗。
- 全对全分发/合并：两者都受限于单设备批量大小和 topk。
- MoE：在内存带宽受限区域的时间消耗会随着 EP 数量的增加而减少。

假设 EP 的数量为 128，MLA 对 DeepSeek-V3 和 LongCat-Flash 使用 DP，GQA 对 Qwen3-235B-A22B 使用 TP4，因为它有 4 个 KV 头，并且每个设备的批量大小为 96。实际上，Qwen3-235B-A22B 的 GQA 特性会导致其 KV 缓存占用较高的内存，在实际中很难实现每张 GPU 的批量大小为 96。这里仅为了理论分析的目的，才假设它可以达到该数值。正如 [Jiashi Li, 2025] 指出，FlashMLA 在 NVIDIA H800 SXM5 GPU 上的理论峰值可达到 660 TFlops；Zhao 等人 [2025b] 指出 DeepEP 带宽可达到 40GB/s。我们的计算中使用了这两个指标。假设每个 H800 的成本为每小时 2 美元。考虑 MTP=1 的接受率为 80%，我们可以计算 DeepSeek-V3、Qwen3-235B-A22B 和 LongCat-Flash 在一层中的各模块的理论时间消耗和成本，如表 7 所列。对于不原生支持 MTP 的 Qwen3-235B-A22B，我们假设一种具有可比接受率的推测性采样策略。

表 7：不同模型的理论解码时间和成本。

	DeepSeek-V3	Qwen3-235B-A22B	LongCat-Flash
MTP	w/	w/o	w/
n_layer	61	94	28
batch per device	96	96	96
			Time cost of different modules in one layer
attention	471 us	314 us	264 us
all-to-all dispatch	275 us	157 us	236 us
MoE	77 us	29 us	60 us
all-to-all combine	551 us	315 us	472 us
			TPOT and Price
overlap strategy	TBO	TBO	SBO
TPOT (ms)	30	26.2	16
$/1M output token	0.17	0.15	0.09

在此配置下，带有 SBO 的 LongCat-Flash 的理论极限 TPOT 可以表示为：

$$\mathrm{TPL} = 264 + 236 + 60 + 472 = 1032\ us,$$

$$\mathrm{TPOT} = \frac{28 \times \mathrm{TPL}}{1000 \times 1.8} \approx 16\ \mathrm{ms},$$

where TPL denotes the time cost per layer.

The measured value under batch size 96 is approximately TPOT = 26 ms, which is about 61.5% of the theoretical value and is on par with DeepSeek-V3 (~64%). The gap between measured value and theoretical speed mainly comes from the overhead of small operators and the loss in communication bandwidth.

We apply the same method to calculate the theoretical limits of TPOT and generation cost for DeepSeek-V3 and Qwen3-235B-A22B under TBO scheduling. It can be observed from Table 7 that through model system co-design, LongCat-Flash achieves significant theoretical improvements in both throughput and latency.

Furthermore, we observed two key insights about LongCat-Flash: (1) LongCat-Flash exposes not only all-to-all communication and MoE computation, but also an MLA computation. As a result, at the same batch size, LongCat-Flash incurs slightly longer per-layer time than DeepSeek-V3. However, due to its significantly reduced layer count, LongCat-Flash achieves lower overall latency. (2) LongCat-Flash's second MLA is overlapped by the all-to-all combine. This means that in the decoding phase, LongCat-Flash can increase the sequence length to a certain extent without substantial latency increase.

7 Conclusion

We introduce LongCat-Flash, a 560B-parameter MoE model featuring three key innovations: (1) a context-aware dynamical computation mechanism and shortcut-connection MoE, enabling high efficiency in both training and inference, (2) integrated strategies that ensure stable large-scale training, (3) a multi-stage training pipeline that cultivates LongCat-Flash's agentic capabilities, allowing it to perform complex tasks requiring iterative reasoning and environmental interaction. By releasing LongCat-Flash as an open-source model, we aim to advance research in efficient MoE architectures, high-quality data strategies, and agentic model development, fostering community-driven innovation in large language models.

其中 TPL 表示每层的时间成本。

在批量大小为96时，测得的值大约为 TPOT = 26 ms，约等于理论值的 61.5%，并且与 DeepSeek-V3 (~64%) 相当。测得值与理论速度之间的差距主要来自小算子开销和通信带宽的损失。

我们采用相同的方法来计算在 TBO 调度下 DeepSeek-V3 与 Qwen3-235B-A22B 的 TPOT 理论极限和生成成本。可从表 7 看出，通过模型与系统的协同设计，LongCat-Flash 在吞吐量和延迟方面都实现了显著的理论改进。

此外，我们观察到关于 LongCat-Flash 的两个关键见解： (1) LongCat-Flash 不仅暴露 all-to-all 通信和 MoE 计算，而且还暴露 MLA 计算。因此，在相同的批量大小下，LongCat- Flash 的每层时间略长于 DeepSeek-V3。然而，由于其显著减少的层数，LongCat-Flash 实现了更低的总体延迟。 (2) LongCat-Flash 的第二个 MLA 与 all-to-all 的组合（combine）重叠。这意味着在解码阶段，LongCat-Flash 可以在不显著增加延迟的情况下，在一定程度上增加序列长度。

7 结论

我们推出 LongCat-Flash，一款拥有 560B 参数的 MoE 模型，具有三项关键创新： (1) 一种上下文感知的动态计算机制与捷径连接的 MoE，能够在训练和推理阶段实现高效； (2) 整合策略，确保大规模训练的稳定性； (3) 一套多阶段的训练流程，培养 LongCat-Flash 的自主性能力，使其能够执行需要迭代推理和环境交互的复杂任务。通过将 LongCat-Flash 作为开源模型发布，我们旨在推动高效 MoE 架构、高质量数据策略以及具自主性能力的模型开发方面的研究，促进在大型语言模型领域的社区驱动创新。

8 Contributions

The listing of authors is in alphabetical order. Names marked with an asterisk (*) indicate people who have left our team.

Bayan
Bei Li
Bingye Lei
Bo Wang
Bolin Rong
Chao Wang
Chao Zhang
Chen Gao
Chen Zhang
Cheng Sun
Chengcheng Han
Chenguang Xi
Chi Zhang
Chong Peng
Chuan Qin
Chuyu Zhang
Cong Chen
Congkui Wang
Dan Ma
Daoru Pan
Defei Bu
Dengchang Zhao
Deyang Kong
Dishan Liu
Feiye Huo
Fengcun Li
Fubao Zhang
Gan Dong
Gang Liu
Gang Xu
Ge Li
Guoqiang Tan
Guoyuan Lin
Haihang Jing
Haomin Fu
Haonan Yan
Haoxing Wen
Haozhe Zhao
Hong Liu
Hongmei Shi*
Hongyan Hao
Hongyin Tang
Huantian Lv
Hui Su
Jiacheng Li
Jiahao Liu
Jiahuan Li
Jiajun Yang
Jiaming Wang
Jian Yang
Jianchao Tan
Jiaqi Sun
Jiaqi Zhang
Jiawei Fu
Jiawei Yang
Jiaxi Hu
Jiayu Qin
Jingang Wang
Jiyuan He
Jun Kuang
Junhui Mei
Kai Liang
Ke He
Kefeng Zhang
Keheng Wang
Keqing He*
Liang Gao
Liang Shi
Lianhui Ma
Lin Qiu
Lingbin Kong
Lingtong Si
Linkun Lyu
Linsen Guo
Liqi Yang
Lizhi Yan
Mai Xia
Man Gao
Manyuan Zhang
Meng Zhou
Mengxia Shen
Mingxiang Tuo
Mingyang Zhu
Peiguang Li
Peng Pei
Peng Zhao
Pengcheng Jia
Pingwei Sun
Qi Gu
Qianyun Li
Qingyuan Li*
Qiong Huang
Qiyuan Duan
Ran Meng
Rongxiang Weng
Ruichen Shao
Rumei Li
Shizhe Wu
Shuai Liang
Shuo Wang
Suogui Dang
Tao Fang
Tao Li
Tefeng Chen
Tianhao Bai
Tianhao Zhou
Tingwen Xie
Wei He
Wei Huang
Wei Liu
Wei Shi
Wei Wang
Wei Wu
Weikang Zhao
Wen Zan
Wenjie Shi
Xi Nan
Xi Su
Xiang Li
Xiang Mei
Xiangyang Ji
Xiangyu Xi
Xiangzhou Huang
Xianpeng Li
Xiao Fu
Xiao Liu
Xiao Wei
Xiaodong Cai
Xiaolong Chen
Xiaoqing Liu
Xiaotong Li
Xiaowei Shi
Xiaoyu Li
Xili Wang
Xin Chen
Xing Hu
Xingyu Miao
Xinyan He
Xuemiao Zhang
Xueyuan Hao
Xuezhi Cao
Xunliang Cai
Xurui Yang
Yan Feng
Yang Bai
Yang Chen
Yang Yang
Yaqi Huo
Yerui Sun
Yifan Lu
Yifan Zhang
Yipeng Zang
Yitao Zhai
Yiyang Li
Yongjing Yin
Yongkang Lv
Yongwei Zhou
Yu Yang
Yuchen Xie
Yueqing Sun
Yuewen Zheng
Yuhua Wei
Yulei Qian
Yunfan Liang
Yunfang Tai
Yunke Zhao
Zeyang Yu
Zhao Zhang
Zhaohua Yang
Zhenchao Zhang
Zhikang Xia
Zhiye Zou
Zhizhao Zeng
Zhongda Su
Zhuofan Chen
Zijian Zhang
Ziwen Wang
Zixu Jiang
Zizhe Zhao
Zongyu Wang
Zunhai Su*
LongCat-Flash

References

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:arXiv preprint arXiv:2412.19437*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

8 项贡献

作者名单按字母顺序排列。 用星号*》标记的名字表示已离开我们团队的人员。

Bayan
Bei Li
Bingye Lei
Bo Wang
Bolin Rong
Chao Wang
Chao Zhang
Chen Gao
Chen Zhang
Cheng Sun
Chengcheng Han
Chenguang Xi
Chi Zhang
Chong Peng
Chuan Qin
Chuyu Zhang
Cong Chen
Congkui Wang
Dan Ma
Daoru Pan
Defei Bu
Dengchang Zhao
Deyang Kong
Dishan Liu
Feiye Huo
Fengcun Li
Fubao Zhang
Gan Dong
Gang Liu
Gang Xu
Ge Li
Guoqiang Tan
Guoyuan Lin
Haihang Jing
Haomin Fu
Haonan Yan
Haoxing Wen
Haozhe Zhao
Hong Liu
Hongmei Shi*
Hongyan Hao
Hongyin Tang
Huantian Lv
Hui Su
Jiacheng Li
Jiahao Liu
Jiahuan Li
Jiajun Yang
Jiaming Wang
Jian Yang
Jianchao Tan
Jiaqi Sun
Jiaqi Zhang
Jiawei Fu
Jiawei Yang
Jiaxi Hu
Jiayu Qin
Jingang Wang
Jiyuan He
Jun Kuang
Junhui Mei
Kai Liang
Ke He
Kefeng Zhang
Keheng Wang
Keqing He*
Liang Gao
Liang Shi
Lianhui Ma
Lin Qiu
Lingbin Kong
Lingtong Si
Linkun Lyu
Linsen Guo
Liqi Yang
Lizhi Yan
Mai Xia
Man Gao
Manyuan Zhang
Meng Zhou
Mengxia Shen
Mingxiang Tuo
Mingyang Zhu
Peiguang Li
Peng Pei
Peng Zhao
Pengcheng Jia
Pingwei Sun
Qi Gu
Qianyun Li
Qingyuan Li*
Qiong Huang
Qiyuan Duan
Ran Meng
Rongxiang Weng
Ruichen Shao
Rumei Li
Shizhe Wu
Shuai Liang
Shuo Wang
Suogui Dang
Tao Fang
Tao Li
Tefeng Chen
Tianhao Bai
Tianhao Zhou
Tingwen Xie
Wei He
Wei Huang
Wei Liu
Wei Shi
Wei Wang
Wei Wu
Weikang Zhao
Wen Zan
Wenjie Shi
Xi Nan
Xi Su
Xiang Li
Xiang Mei
Xiangyang Ji
Xiangyu Xi
Xiangzhou Huang
Xianpeng Li
Xiao Fu
Xiao Liu
Xiao Wei
Xiaodong Cai
Xiaolong Chen
Xiaoqing Liu
Xiaotong Li
Xiaowei Shi
Xiaoyu Li
Xili Wang
Xin Chen
Xing Hu
Xingyu Miao
Xinyan He
Xuemiao Zhang
Xueyuan Hao
Xuezhi Cao
Xunliang Cai
Xurui Yang
Yan Feng
Yang Bai
Yang Chen
Yang Yang
Yaqi Huo
Yerui Sun
Yifan Lu
Yifan Zhang
Yipeng Zang
Yitao Zhai
Yiyang Li
Yongjing Yin
Yongkang Lv
Yongwei Zhou
Yu Yang
Yuchen Xie
Yueqing Sun
Yuewen Zheng
Yuhua Wei
Yulei Qian
Yunfan Liang
Yunfang Tai
Yunke Zhao
Zeyang Yu
Zhao Zhang
Zhaohua Yang
Zhenchao Zhang
Zhikang Xia
Zhiye Zou
Zhizhao Zeng
Zhongda Su
Zhuofan Chen
Zijian Zhang
Ziwen Wang
Zixu Jiang
Zizhe Zhao
Zongyu Wang
Zunhai Su*
LongCat-Flash

参考文献

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao , Chengqi Deng, Chenyu Zhang, Chong Ruan 等。Deepseek-v3 技术报告。 *arXiv preprint arXiv:arXiv preprint arXiv:2412.19437*, 2025。

An Yang、Anfeng Li、Baosong Yang、Beichen Zhang、Binyuan Hui、Bo Zheng、Bowen Yu、Chang Gao、Ch engen Huang、Chenxu Lv 等。Qwen3 技术报告。*arXiv preprint arXiv:2505.09388*, 2025。

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

Weilin Cai, Juyong Jiang, Le Qin, Junwei Cui, Sunghun Kim, and Jiayi Huang. Shortcut-connected expert parallelism for accelerating mixture-of-experts. *arXiv preprint arXiv:2404.05019*, 2024.

Jiaming Wang, Yunke Zhao, Peng Ding, Jun Kuang, Zongyu Wang, Xuezhi Cao, and Xunliang Cai. Ask, fail, repeat: Meeseeks, an iterative feedback benchmark for llms' multi-turn instruction-following ability. *arXiv preprint arXiv:2504.21625*, 2025a.

Peng Jin, Bo Zhu, Li Yuan, and Shuicheng Yan. MoE++: Accelerating mixture-of-experts methods with zero-computation experts. *arXiv preprint arXiv:2410.07348*, 2024.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024a.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, 2023.

Zihao Zeng, Yibo Miao, Hongcheng Gao, Hao Zhang, and Zhijie Deng. AdaMoE: Token-adaptive routing with null experts for mixture-of-experts language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, 2024.

Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, and Damai Dai. Auxiliary-loss-free load balancing strategy for mixture-of-experts. *arXiv preprint arXiv:2408.15664*, 2024a.

Stuart Bennett. *A History of Control Engineering 1930-1955*. Peter Peregrinus, GBR, 1st edition, 1993. ISBN 0863412998.

Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation AI scale. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. 2017.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

Yusuxke Shibata, Takuya Kida, Shuichi Fukamachi, Masayuki Takeda, Ayumi Shinohara, Takeshi Shinohara, and Setsuo Arikawa. Byte pair encoding: A text compression scheme that accelerates pattern matching. 1999.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. *arXiv preprint arXiv:2404.19737*, 2024.

Katie Everett, Lechao Xiao, Mitchell Wortsman, Alexander A. Alemi, Roman Novak, Peter J. Liu, Izzeddin Gur, Jascha Sohl-Dickstein, Leslie Pack Kaelbling, Jaehoon Lee, and Jeffrey Pennington. Scaling exponents across parameterizations and optimizers. *arXiv preprint arXiv:2407.05872*, 2024.

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015.

Wenyu Du, Tongxu Luo, Zihan Qiu, Zeyu Huang, Yikang Shen, Reynold Cheng, Yike Guo, and Jie Fu. Stacking your transformers: A closer look at model growth for efficient LLM pre-training. *arXiv preprint arXiv:2405.15319*, 2024.

Peihao Wang, Rameswar Panda, Lucas Torroba Hennigen, Philip Greengard, Leonid Karlinsky, Rogerio Feris, David Daniel Cox, Zhangyang Wang, and Yoon Kim. Learning to grow pretrained models for efficient transformer training. *arXiv preprint arXiv:2303.00980*, 2023a.

Sheng Shen, Pete Walsh, Kurt Keutzer, Jesse Dodge, Matthew Peters, and Iz Beltagy. Staged training for transformer language models. In *International Conference on Machine Learning*, 2022.

Yite Wang, Jiahao Su, Hanlin Lu, Cong Xie, Tianyi Liu, Jianbo Yuan, Haibin Lin, Ruoyu Sun, and Hongxia Yang. Lemon: Lossless model expansion. *arXiv preprint arXiv:2310.07999*, 2023b.

Linyuan Gong, Di He, Zhuohan Li, Tao Qin, Liwei Wang, and Tieyan Liu. Efficient training of BERT by progressively stacking. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Kimi 团队、Yifan Bai、Yiping Bao、Guanduo Chen、Jiahao Chen、Ningxin Chen、Ruijue Chen、Yanru Chen、Yuankun Chen、Yutian Chen 等。Kimi k2：开放代理性智能。*arXiv preprint arXiv:2507.20534*，2025。Weilin Cai、Juyong Jiang、Le Qin、Junwei Cui、Sunghun Kim、Jiayi Huang。快捷连接的专家并行性以加速专家混合模型。*arXiv preprint arXiv:2404.05019*，2024。Jiaming Wang、Yunke Zhao、Peng Ding、Jun Kuang、Zongyu Wang、Xuezhi Cao、Xunliang Cai。Ask, fail, repeat: Meeseeks，一种用于大型语言模型多轮指令遵循能力的迭代反馈基准。*arXiv preprint arXiv:2504.21625*，2025a。Peng Jin、Bo Zhu、Li Yuan、Shuicheng Yan。MoE++：使用零计算专家加速专家混合方法。*arXiv preprint arXiv:2410.07348*，2024。Aixin Liu、Bei Feng、Bin Wang、Bingxuan Wang、Bo Liu、Chenggang Zhao、Chengqi Dengr、Chong Ruan、Damai Dai、Daya Guo 等。Deepseek-v2：一个强大、经济且高效的专家混合语言模型。*arXiv preprint arXiv:2405.04434*，2024a。Yaniv Leviathan、Matan Kalman、Yossi Matias。通过推测解码实现 Transformer 的快速推断。发表于 *International Conference on Machine Learning*，2023。Zihao Zeng、Yibo Miao、Hongcheng Gao、Hao Zhang、Zhijie Deng。AdaMoE：为混合专家语言模型提供带空专家的令牌自适应路由。发表于 *Findings of the Association for Computational Linguistics: EMNLP 2024*，2024。Lean Wang、Huazuo Gao、Chenggang Zhao、Xu Sun、Damai Dai。用于专家混合的无辅助损失负载均衡策略。*arXiv preprint arXiv:2408.15664*，2024a。Stuart Bennett。*A History of Control Engineering 1930-1955*。Peter Peregrinus，GBR，第一版，1993。ISBN 0863412998。Samyam Rajbhandari、Conglong Li、Zhewei Yao、Minjia Zhang、Reza Yazdani Aminabadi、Ammar Ahmad Awan、Jeff Rasley、Yuxiong He。Deepspeed-moe：推进混合专家推断和训练以支撑下一代 AI 规模。发表于 *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*，2022。Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N Gomez、 ukasz Kaiser、Illia Polosukhin。注意力是你所需要的一切。2017。Joshua Ainslie、James Lee-Thorp、Michiel de Jong、Yury Zemlyanskiy、Federico Lebron、Sumit Sanghai。Gqa：从多头检查点训练广义多查询 Transformer 模型。发表于 *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*，2023。Yusuke Shibata、Takuya Kida、Shuichi Fukamachi、Masayuki Takeda、Ayumi Shinohara、Takeshi Shinohara、Setsuo Arikawa。字节对编码（Byte Pair Encoding）：一种加速模式匹配的文本压缩方案。1999。Rico Sennrich、Barry Haddow、Alexandra Birch。使用子词单元的稀有词神经机器翻译。*arXiv preprint arXiv:1508.07909*，2015。Fabian Gloeckle、Badr Youbi Idrissi、Baptiste Rozière、David Lopez-Paz、Gabriel Synnaeve。通过多令牌预测实现更好更快的大型语言模型。*arXiv preprint arXiv:2404.19737*，2024。Katie Everett、Lechao Xiao、Mitchell Wortsman、Alexander A. Alemi、Roman Novak、Peter J. Liu、Izzeddin Gur、Jascha Sohl-Dickstein、Leslie Pack Kaelbling、Jaehoon Lee、Jeffrey Pennington。跨参数化与优化器的尺度指数。*arXiv preprint arXiv:2407.05872*，2024。Tianqi Chen、Ian Goodfellow、Jonathon Shlens。Net2Net：通过知识迁移加速学习。*arXiv preprint arXiv:1511.05641*，2015。Wenyu Du、Tongxu Luo、Zihan Qiu、Zeyu Huang、Yikang Shen、Reynold Cheng、Yike Guo、Jie Fu。堆叠你的 Transformer：对高效 LLM 预训练中的模型增长进行更近距离的观察。*arXiv preprint arXiv:2405.15319*，2024。Peihao Wang、Rameswar Panda、Lucas Torroba Hennigen、Philip Greengard、Leonid Karlinsky、Rogerio Feris、David Daniel Cox、Zhangyang Wang、Yoon Kim。学习以增长预训练模型以实现高效 Transformer 训练。*arXiv preprint arXiv:2303.00980*，2023a。Sheng Shen、Pete Walsh、Kurt Keutzer、Jesse Dodge、Matthew Peters、Iz Beltagy。Transformer 语言模型的分阶段训练。发表于 *International Conference on Machine Learning*，2022。Yite Wang、Jiahao Su、Hanlin Lu、Cong Xie、Tianyi Liu、Jianbo Yuan、Haibin Lin、Ruoyu Sun、Hongxia Yang。Lemon：无损模型扩展。*arXiv preprint arXiv:2310.07999*，2023b。Linyuan Gong、Di He、Zhuohan Li、Tao Qin、Liwei Wang、Tieyan Liu。通过逐步堆叠实现 BERT 的高效训练。发表于 *Proceedings of the 36th International Conference on Machine Learning*，2019。

Dahyun Kim, Chanjun Park, Sanghoon Kim, Wonsung Lee, Wonho Song, Yunsu Kim, Hyeonwoo Kim, Yungi Kim, Hyeonju Lee, Jihoo Kim, et al. Solar 10.7 b: Scaling large language models with simple yet effective depth up-scaling. *arXiv preprint arXiv:2312.15166*, 2023.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. ST-MoE: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.

Mingjie Sun, Xinlei Chen, J. Zico Kolter, and Zhuang Liu. Massive activations in large language models. *arXiv preprint arXiv:2402.17762*, 2024.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, et al. 2 olmo 2 furious. *arXiv preprint arXiv:2501.00656*, 2024.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. 2019.

Adrien Barbaresi. Trafilatura: A web scraping library and command-line tool for text discovery and extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, 2021.

Xiangyu Xi, Deyang Kong, Jian Yang, Jiawei Yang, Zhengyu Chen, Wei Wang, Jingang Wang, Xunliang Cai, Shikun Zhang, and Wei Ye. Samplemix: A sample-wise pre-training data mixing strategey by coordinating data quality and diversity. *arXiv preprint arXiv:2503.01506*, 2025.

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

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. BGE M3-Embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2021a.

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. *arXiv preprint arXiv:2406.01574*, 2024b.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Yao Fu, Maosong Sun, and Junxian He. C-Eval: A multi-level multi-discipline chinese evaluation suite for foundation models. In *Advances in Neural Information Processing Systems*, 2023.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. CMMLU: Measuring massive multitask language understanding in chinese. *arXiv preprint arXiv:2306.09212*, 2023a.

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 qa benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

M-A-P Team, ByteDance. SuperGPQA: Scaling LLM evaluation across 285 graduate disciplines. *arXiv preprint arXiv:2502.14739*, 2025.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, 2023.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. *arXiv preprint arXiv:1911.11641*, 2019.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv:1903.00161*, 2019.

Liang Xu, Hai Hu, Xuanwei Zhang, Lu Li, Chenjie Cao, Yudong Li, Yechen Xu, Kai Sun, Dian Yu, Cong Yu, Yin Tian, Qianqian Dong, Weitang Liu, Bo Shi, Yiming Cui, Junyi Li, Jun Zeng, Rongzhao Wang, Weijian Xie, Yanting Li, Yina Patterson, Zuoyu Tian, Yiwen Zhang, He Zhou, Shaoweihua Liu, Zhe Zhao, Qipeng Zhao, Cong Yue, Xinrui Zhang, Zhengliang Yang, Kyle Richardson, and Zhenzhong Lan. CLUE: A Chinese language understanding evaluation benchmark. In *Proceedings of the 28th International Conference on Computational Linguistics*, 2020.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*, 2019.

Dahyun Kim, Chanjun Park, Sanghoon Kim, Wonsung Lee, Wonho Song, Yunsu Kim, Hyeonwoo Kim, Yungi Kim, Hyeonju Lee, Jihoo Kim 等。Solar 10.7 b：通过简单而有效的深度提升来扩展大型语言模型。*arXiv preprint arXiv:2312.15166*, 2023年。

Barret Zoph、Irwan Bello、Sameer Kumar、Nan Du、Yanping Huang、Jeff Dean、Noam Shazeer 与 William Fedus。ST-MoE：设计稳定且可迁移的稀疏专家模型。*arXiv preprint arXiv:2202.08906*，2022。

Mingjie Sun, Xinlei Chen, J. Zico Kolter, and Zhuang Liu. 大型语言模型中的大规模激活。*arXiv preprint arXiv:2402.17762*，2024。

OLMo 团队，Pete Walsh，Luca Soldaini，Dirk Groeneveld，Kyle Lo，Shane Arora，Akshita Bhagia，Yuling Gu，Shengyi Huang，Matt Jordan 等人。2 olmo 2 furious。*arXiv preprint arXiv:2501.00656*，2024年。

标志 ang 和 Rico Sennrich。均方根层归一化 2019 年。

Adrien Barbaresi。Trafilatura：一个用于文本发现与提取的网页抓取库和命令行工具。在 *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*，2021 年。

Xiangyu Xi、Deyang Kong、Jian Yang、Jiawei Yang、Zhengyu Chen、Wei Wang、Jingang Wang、Xunliang Cai、Shikun Zhang 与 Wei Ye。Samplemix：通过协调数据质量与多样性实现的样本级预训练数据混合策略。*arXiv preprint arXiv:2503.01506*, 2025。

苏建林、穆塔达·艾哈迈德、陆宇、潘胜风、温博，以及刘云峰。Roformer：带旋转位置嵌入的增强型 Transformer。*Neurocomputing*，568:127063，2024。

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, 和 Zheng Liu。BGE M3-Embedding：多语言、多功能、多粒度文本嵌入通过自我知识蒸馏实现。*arXiv preprint arXiv:2402.03216*，2024。

Dan Hendrycks、Collin Burns、Steven Basart、Andy Zou、Mantas Mazeika、Dawn Song 和 Jacob Steinhardt。测量大规模多任务语言理解。*arXiv preprint arXiv:2009.03300*, 2021a.

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 和 Wenhu Chen。MMLU-Pro：一个更健壮且具有挑战性的多任务语言理解基准。*arXiv preprint arXiv:2406.01574*, 2024b。

Yuzhen Huang、Yuzhuo Bai、Zhihao Zhu、Junlei Zhang、Jinghan Zhang、Tangjun Su、Junteng Liu、Chuancheng Lv、Yikai Zhang、Jiayi Lei、Yao Fu、Maosong Sun和Junxian He。C-Eval：面向基础模型的多层级多学科中文评估套件。发表于 *Advances in Neural Information Processing Systems*，2023 年。

Haonan Li、Yixuan Zhang、Fajri Koto、Yifei Yang、Hai Zhao、Yeyun Gong、Nan Duan 与 Timothy Baldwin。CMMLU：在中文中的大规模多任务语言理解能力的衡量。*arXiv preprint arXiv:2306.09212*，2023a。

David Rein、Betty Li Hou、Asa Cooper Stickland、Jackson Petty、Richard Yuanzhe Pang、Julien Dirani、Julian Michael 与 Samuel R. Bowman。GPQA：一个研究生级别、无需谷歌检索即可完成的问答基准。*arXiv preprint arXiv:2311.12022*，2023。

字节跳动的 M-A-P 团队。SuperGPQA：将对大型语言模型的评估扩展至 285 个研究生学科。 *arXiv preprint arXiv:2502.14739*，2025 年。

Mirac Suzgun、Nathan Scales、Nathanael Schärli、Sebastian Gehrmann、Yi Tay、Hyung Won Chung、Aakanksha Chowdhery、Quoc Le、Ed Chi、Denny Zhou 和 Jason Wei。具有挑战性的 BIG-bench 任务，以及链式推理是否能解决它们。在 *Findings of the Association for Computational Linguistics: ACL 2023*，2023 年。

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao，以及 Yejin Choi。PIQA：在自然语言中对物理常识的推理。*arXiv preprint arXiv:1911.11641*，2019。

Dheeru Dua、Yizhong Wang、Pradeep Dasigi、Gabriel Stanovsky、Sameer Singh 和 Matt Gardner。DROP：一个需要对段落进行离散推理的阅读理解基准。*arXiv preprint arXiv:1903.00161*，2019。

Liang Xu, Hai Hu, Xuanwei Zhang, Lu Li, Chenjie Cao, Yudong Li, Yechen Xu, Kai Sun, Dian Yu, Cong Yu, Yin Tian, Qianqian Dong, Weitang Liu, Bo Shi, Yiming Cui, Junyi Li, Jun Zeng, Rongzhao Wang, Weijian Xie, Yanting Li, Yina Patterson, Zuoyu Tian, Yiwen Zhang, He Zhou, Shaoweihua Liu, Zhe Zhao, Qipeng Zhao, Cong Yue, Xinrui Zhang, Zhengliang Yang, Kyle Richardson, and Zhenzhong Lan. CLUE：一个中文语言理解评估基准。 在 *Proceedings of the 28th International Conference on Computational Linguistics*，2020 年。

Keisuke Sakaguchi、Ronan Le Bras、Chandra Bhagavatula 与 Yejin Choi。Winogrande：一种大规模的对抗性Winograd范式挑战。*arXiv preprint arXiv:1907.10641*，2019。

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. *arXiv preprint arXiv:2110.14168*, 2021.

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. *arXiv preprint arXiv:2103.03874*, 2021b.

Jiawei Liu, Songrun Xie, Junhao Wang, Yuxiang Wei, Yifeng Ding, and Lingming Zhang. Evaluating language models for efficient code generation. *arXiv preprint arXiv:2408.06450*, 2024b.

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 extensible approach to benchmarking neural code generation. *arXiv preprint arXiv:2208.08227*, 2022.

Alex Gu, Baptiste Rozière, Hugh Leather, Armando Solar-Lezama, Gabriel Synnaeve, and Sida I. Wang. Cruxeval: A benchmark for code reasoning, understanding and execution. *arXiv preprint arXiv:2401.03065*, 2024.

Meta AI. The llama 4 herd: The beginning of a new era of natively multimodal ai innovation, 2025. URL https://ai.meta.com/blog/llama-4-multimodal-intelligence/.

MoonshotAI. Kimi-K2 documentation, 2025. URL https://moonshotai.github.io/Kimi-K2/.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Loubna Ben Allal, Niklas Muennighoff, Logesh Kumar Umapathi, Ben Lipkin, and Leandro von Werra. A framework for the evaluation of code generation models. https://github.com/bigcode-project/bigcode-evaluation-harness, 2022.

Tao Ge, Xin Chan, Xiaoyang Wang, Dian Yu, Haitao Mi, and Dong Yu. Scaling synthetic data creation with 1,000,000,000 personas. *arXiv preprint arXiv:2406.20094*, 2024.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*, 2022.

Yuxiang Wei, Federico Cassano, Jiawei Liu, Yifeng Ding, Naman Jain, Zachary Mueller, Harm de Vries, Leandro von Werra, Arjun Guha, and Lingming Zhang. Selfcodealign: Self-alignment for code generation. In *Advances in Neural Information Processing Systems*, 2024.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. Wizardcoder: Empowering code large language models with evol-instruct. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*, 2024.

Jin Jiang, Yuchen Yan, Yang Liu, Jianing Wang, Shuai Peng, Xunliang Cai, Yixin Cao, Mengdi Zhang, and Liangcai Gao. LogicPro: Improving complex logical reasoning via program-guided learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025.

Chenxu Wang, Ping Jian, and Zhen Yang. Thought-path contrastive learning via premise-oriented data augmentation for logical reading comprehension. In *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, 2025b.

Junjie Ye, Caishuang Huang, Zhuohan Chen, Wenjie Fu, Chenyuan Yang, Leyi Yang, Yilong Wu, Peng Wang, Meng Zhou, Xiaolong Yang, et al. A multi-dimensional constraint framework for evaluating and improving instruction following in large language models. *arXiv preprint arXiv:2505.07591*, 2025.

Yubo Wang, Xiang Yue, and Wenhu Chen. Critique fine-tuning: Learning to critique is more effective than learning to imitate. *arXiv preprint arXiv:2501.17703*, 2025c.

Tong Mu, Alec Helyar, Johannes Heidecke, Joshua Achiam, Andrea Vallone, Ian Kivlichan, Molly Lin, Alex Beutel, John Schulman, and Lilian Weng. Rule based rewards for language model safety. In *Advances in Neural Information Processing Systems*, 2024.

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. *arXiv preprint arXiv:2406.11939*, 2024a.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. From live data to high-quality benchmarks: The arena-hard pipeline, April 2024b. URL https://lmsys.org/blog/2024-04-19-arena-hard/.

Karl Cobbe、Vineet Kosaraju、Mohammad Bavarian、Mark Chen、Heewoo Jun、Lukasz Kaiser、Matthias Plappert、Jerry Tworek、Jacob Hilton、Reiichiro Nakano、Christopher Hesse 与 John Schulman。训练验证者解决数学文字题。*arXiv preprint arXiv:2110.14168*，2021。Dan Hendrycks、Collin Burns、Saurav Kadavath、Akul Arora、Steven Basart、Eric Tang、Dawn Song，以及 Jacob Steinhardt。使用数学数据集衡量数学问题求解能力。*arXiv preprint arXiv:2103.03874*，2021b。Jiawei Liu、Songrun Xie、Junhao Wang、Yuxiang Wei、Yifeng Ding，以及 Lingming Zhang。评估用于高效代码生成的语言模型。*arXiv preprint arXiv:2408.06450*，2024b。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 与 Abhinav Jangda。MultiPL-E：一种可扩展且易于扩展的用于对神经代码生成进行基准测试的方法。*arXiv preprint arXiv:2208.08227*，2022。Alex Gu、Baptiste Rozière、Hugh Leather、Armando Solar-Lezama、Gabriel Synnaeve、Sida I. Wang。Cruxeval：用于代码推理、理解与执行的基准测试。*arXiv preprint arXiv:2401.03065*，2024。Meta AI。The llama 4 herd：原生多模态 AI 创新新时代的开端，2025。URL https://ai.meta.com/blog/llama-4-multimodal-intelligence/。MoonshotAI。Kimi-K2 文档，2025。URL https://moonshotai.github.io/Kimi-K2/。Mark Chen、Jerry Tworek、Heewoo Jun、Qiming Yuan、Henrique Ponde De Oliveira Pinto、Jared Kaplan、Harri Edwards、Yuri Burda、Nicholas Joseph、Greg Brockman 等。评估在代码上训练的大型语言模型。*arXiv preprint arXiv:2107.03374*，2021。Loubna Ben Allal、Niklas Muennighoff、Logesh Kumar Umapathi、Ben Lipkin、Leandro von Werra。用于代码生成模型评估的框架。https://github.com/bigcode-project/ bigcode-evaluation-harness，2022。Tao Ge、Xin Chan、Xiaoyang Wang、Dian Yu、Haitao Mi、 Dong Yu。通过10亿个角色实现合成数据创建的规模化。*arXiv preprint arXiv:2406.20094*，2024。Yizhong Wang、Yeganeh Kordi、Swaroop Mishra、Alisa Liu、Noah A Smith、Daniel Khashabi、Hannaneh Hajishirzi。Self-instruct：将语言模型与自行生成的指令对齐。*arXiv preprint arXiv:2212.10560*，2022。Yuxiang Wei、Federico Cassano、Jiawei Liu、Yifeng Ding、Naman Jain、Zachary Mueller、Harm de Vries、Leandro von Werra、Arjun Guha、Lingming Zhang。Selfcodealign：用于代码生成的自我对齐。发表于 *Advances in Neural Information Processing Systems*，2024。Ziyang Luo、Can Xu、Pu Zhao、Qingfeng Sun、Xiubo Geng、Wenxiang Hu、Chongyang Tao、Jing Ma、Qingwei Lin、Daxin Jiang。Wizardcoder：通过 evol-instruct 提升代码大型语言模型的能力。发表于 *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*，2024。Jin Jiang、Yuchen Yan、Yang Liu、Jianing Wang、Shuai Peng、Xunliang Cai、Yixin Cao、Mengdi Zhang、Liangcai Gao。LogicPro：通过程序引导学习来提高复杂的逻辑推理。发表于 *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*，2025。Chenxu Wang、Ping Jian、Zhen Yang。思路路径对比学习：基于前提导向的数据增强用于逻辑阅读理解。发表于 *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*，2025b。Junjie Ye、Caishuang Huang、Zhuohan Chen、Wenjie Fu、Chenyuan Yang、Leyi Yang、Yilong Wu、Peng Wang、Meng Zhou、Xiaolong Yang 等。用于评估和改进大型语言模型指令遵循的多维约束框架。*arXiv preprint arXiv:2505.07591*，2025。Yubo Wang、Xiang Yue、Wenhu Chen。批判性微调：学习批评比学习模仿更有效。*arXiv preprint arXiv:2501.17703*，2025c。Tong Mu、Alec Helyar、Johannes Heidecke、Joshua Achiam、Andrea Vallone、Ian Kivlichan、Molly Lin、Alex Beutel、John Schulman、Lilian Weng。基于规则的奖励以提升语言模型的安全性。发表于 *Advances in Neural Information Processing Systems*，2024。Tianle Li、Wei-Lin Chiang、Evan Frick、Lisa Dunlap、Tianhao Wu、Banghua Zhu、Joseph E Gonzalez、Ion Stoica。从众包数据到高质量基准：Arena-hard 与 benchbuilder 流水线。*arXiv preprint arXiv:2406.11939*，2024a。Tianle Li、Wei-Lin Chiang、Evan Frick、Lisa Dunlap、Tianhao Wu、Banghua Zhu、Joseph E Gonzalez、Ion Stoica。从实时数据到高质量基准：Arena-hard 流水线，2024年4月b。URL https://lmsys.org/ blog/2024-04-19-arena-hard/。

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

Shunyu Yao, Howard Chen, Austin W. Hanjie, Runzhe Yang, and Karthik R Narasimhan. COLLIE: Systematic construction of constrained text generation tasks. In *The Twelfth International Conference on Learning Representations*, 2024.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

MAA. Aime 2024, 2024. URL https://maa.org/math-competitions/american-invitational-mathematics-examination-aime.

MAA. Aime 2025, 2025. URL https://artofproblemsolving.com/wiki/index.php/AIMEProblemsandSolutions.

ByteDance-Seed. Beyondaime: Advancing math reasoning evaluation beyond high school olympiads. https://huggingface.co/datasets/ByteDance-Seed/BeyondAIME, 2025.

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. In *First Conference on Language Modeling*, 2024.

Bill Yuchen Lin, Ronan Le Bras, Kyle Richardson, Ashish Sabharwal, Radha Poovendran, Peter Clark, and Yejin Choi. Zebralogic: On the scaling limits of LLMs for logical reasoning. In *Forty-second International Conference on Machine Learning*, 2025.

OpenAI. Graphwalks dataset, 2025a. URL https://huggingface.co/datasets/openai/graphwalks.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation. In *Advances in Neural Information Processing Systems*, 2023.

Jiawei Liu, Songrun Xie, Junhao Wang, Yuxiang Wei, Yifeng Ding, and Lingming Zhang. Evaluating language models for efficient code generation. In *First Conference on Language Modeling*, 2024c.

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. In *The Thirteenth International Conference on Learning Representations*, 2025.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024.

The Terminal-Bench Team. Terminal-bench: A benchmark for ai agents in terminal environments, Apr 2025a. URL https://github.com/laude-institute/terminal-bench.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*, 2025.

Chen Chen, Xinlong Hao, Weiwen Liu, Xu Huang, Xingshan Zeng, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Yuefeng Huang, et al. ACEBench: Who wins the match point in tool learning? *arXiv preprint arXiv:2501.12851*, pages arXiv–2501, 2025.

OpenAI. Introducing GPT-4.1 in the api, April 2025b. URL https://openai.com/index/gpt-4-1/.

Anthropic. Introducing claude 4, May 2025. URL https://www.anthropic.com/news/claude-4.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Author Qi and Others. Zero-bubble pipeline parallelism for large language models. *arXiv preprint arXiv:2301.12345*, 2023.

Penghui Qi, Xinyi Wan, Nyamdavaa Amar, and Min Lin. Pipeline parallelism with controllable memory. 2024.

Kan Zhu, Yufei Gao, Yilong Zhao, Liangyu Zhao, Gefei Zuo, Yile Gu, Dedong Xie, Tian Tang, Qinyu Xu, Zihao Ye, Keisuke Kamahori, Chien-Yu Lin, Ziren Wang, Stephanie Wang, Arvind Krishnamurthy, and Baris Kasikci. NanoFlow: Towards optimal large language model serving throughput. *arXiv preprint arXiv:2408.12757*, 2025.

Yulei Qian, Fengcun Li, Xiangyang Ji, Xiaoyu Zhao, Jianchao Tan, Kefeng Zhang, and Xunliang Cai. EPS-MoE: Expert pipeline scheduler for cost-efficient moe inference. *arXiv preprint arXiv:2410.12247*, 2025.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 大语言模型的指令遵循评估。 *arXiv preprint arXiv:2311.07911*，2023。Shunyu Yao, Howard Chen, Austin W. Hanjie, Runzhe Yang, and Karthik R Narasimhan. COLLIE：受限文本生成任务的系统性构建。发表于 *The Twelfth International Conference on Learning Representations*，2024。Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 让我们逐步验证。在 *The Twelfth International Conference on Learning Representations*，2023。MAA。AIME 2024，2024。 网址 https://maa.org/math-competitions/ american-invitational-mathematics-examination-aime。MAA。AIME 2025，2025。 网址 https://artofproblemsolving.com/wiki/index.php/ AIMEProblemsandSolutions。ByteDance-Seed。BeyondAIME：在高中奥林匹克之外推进数学推理评估。 https: //huggingface.co/datasets/ByteDance-Seed/BeyondAIME，2025。David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA：一个研究生水平的 Google-proof 问答基准。发表于 *First Conference on Language Modeling*，2024。Bill Yuchen Lin, Ronan Le Bras, Kyle Richardson, Ashish Sabharwal, Radha Poovendran, Peter Clark, and Yejin Choi. Zebralogic：关于大语言模型在逻辑推理方面的可扩展性极限。发表于 *Forty-second International Conference on Machine Learning*，2025。OpenAI。Graphwalks 数据集，2025a。 网址 https://huggingface.co/datasets/openai/graphwalks。Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct？rigorous evaluation of large language models for code generation。发表于 *Advances in Neural Information Processing Systems*，2023。Jiawei Liu, Songrun Xie, Junhao Wang, Yuxiang Wei, Yifeng Ding, and Lingming Zhang. Evaluating language models for efficient code generation。发表于 *First Conference on Language Modeling*，2024c。Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench：对代码的大语言模型进行全面且无污染的评估。发表于 *The Thirteenth International Conference on Learning Representations*，2025。Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. SWE-bench：语言模型能解决现实世界的 GitHub 问题吗？发表于 *The Twelfth International Conference on Learning Representations*，2024。The Terminal-Bench Team。Terminal-bench：用于终端环境中 AI 代理的基准，2025年4月a。 网址 https://github.com/laude-institute/terminal-bench。Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench：在双控制环境中评估对话代理。发表于 *arXiv preprint arXiv:2506.07982*，2025。Chen Chen, Xinlong Hao, Weiwen Liu, Xu Huang, Xingshan Zeng, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Yuefeng Huang, et al. ACEBench：在工具学习中，谁赢得赛点？*arXiv preprint arXiv:2501.12851*，页面 arXiv–2501，2025。OpenAI。API 中推出 GPT-4.1，2025年4月b。 网址 https://openai.com/index/gpt-4-1/。Anthropic。Introducing Claude 4，2025年5月。 网址 https://www.anthropic.com/news/claude-4。Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen，等。Gemini 2.5：以高级推理、多模态、长上下文和下一代代理能力推动前沿。*arXiv preprint arXiv:2507.06261*，2025。Author Qi and Others。Zero-bubble pipeline parallelism for large language models。*arXiv preprint arXiv:2301.12345*，2023。Penghui Qi, Xinyi Wan, Nyamdavaa Amar, and Min Lin。 Pipeline parallelism with controllable memory。2024。Kan Zhu, Yufei Gao, Yilong Zhao, Liangyu Zhao, Gefei Zuo, Yile Gu, Dedong Xie, Tian Tang, Qinyu Xu, Zihao Ye, Keisuke Kamahori, Chien-Yu Lin, Ziren Wang, Stephanie Wang, Arvind Krishnamurthy, and Baris Kasikci。NanoFlow：朝向大语言模型服务吞吐量的最优实现。*arXiv preprint arXiv:2408.12757*，2025。Yulei Qian, Fengcun Li, Xiangyang Ji, Xiaoyu Zhao, Jianchao Tan, Kefeng Zhang, and Xunliang Cai。 EPS-MoE：用于成本高效 MoE 推理的专家管道调度器。*arXiv preprint arXiv:2410.12247*，2025。

The SGLang Team. Deploying deepseek with pd disaggregation and large-scale expert parallelism on 96 h100 gpus. https://lmsys.org/blog/2025-05-05-large-scale-ep/, 2025b. Accessed: [May 2025].

Jack Choquette. Nvidia hopper gpu: Scaling performance. In *2022 IEEE Hot Chips 34 Symposium (HCS)*, 2022.

Ranajoy Sadhukhan, Jian Chen, Zhuoming Chen, Vashisth Tiwari, Ruihang Lai, Jinyuan Shi, Ian En-Hsu Yen, Avner May, Tianqi Chen, and Beidi Chen. MagicDec: Breaking the latency-throughput tradeoff for long context generation with speculative decoding. *arXiv preprint arXiv:2408.11049*, 2025.

Jiahao Liu, Qifan Wang, Jingang Wang, and Xunliang Cai. Speculative decoding via early-exiting for faster llm inference with thompson sampling control mechanism. *arXiv preprint arXiv:2406.03853*, 2024d.

Feiye Huo, Jianchao Tan, Kefeng Zhang, Xunliang Cai, and Shengli Sun. C2T: A classifier-based tree construction method in speculative decoding. *arXiv preprint arXiv:2502.13652*, 2025.

Amey Agrawal, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S Gulavani, and Ramachandran Ramjee. Sarathi: Efficient llm inference by piggybacking decodes with chunked prefills. *arXiv preprint arXiv:2308.16369*, 2023.

Chenggang Zhao, Liang Zhao, Jiashi Li, and Zhean Xu. DeepGEMM: clean and efficient fp8 gemm kernels with fine-grained scaling. https://github.com/deepseek-ai/DeepGEMM, 2025a.

Pengcuo Dege, Qiuming Luo, Rui Mao, and Chang Kong. FlashMLA-ETAP: Efficient transpose attention pipeline for accelerating mla inference on nvidia h20 gpus. *arXiv preprint arXiv:2506.01969*, 2025.

NVIDIA. NVIDIA Collective Communications Library (NCCL). https://github.com/NVIDIA/nccl. Version 2.21.5.

Aashaka Shah, Abhinav Jangda, Binyang Li, Caio Rocha, Changho Hwang, Jithin Jose, Madan Musuvathi, Olli Saarikivi, Peng Cheng, Qinghua Zhou, Roshan Dathathri, Saeed Maleki, and Ziyue Yang. MSCCL++: Rethinking gpu communication abstractions for cutting-edge ai applications. *arXiv preprint arXiv:2504.09014*, 2025.

Qingyuan Li, Yifan Zhang, Liang Li, Peng Yao, Bo Zhang, Xiangxiang Chu, Yerui Sun, Li Du, and Yuchen Xie. FPTQ: Fine-grained post-training quantization for large language models. *arXiv preprint arXiv:2308.15987*, 2023b.

Zunhai Su, Qingyuan Li, Hao Zhang, YuLei Qian, Yuchen Xie, and Kehong Yuan. Unveiling super experts in mixture-of-experts large language models. *arXiv preprint arXiv:2507.23279*, 2025.

Chenggang Zhao, Shangyan Zhou, Liyue Zhang, Chengqi Deng, Zhean Xu, Yuxuan Liu, Kuai Yu, Jiashi Li, and Liang Zhao. DeepEP: an efficient expert-parallel communication library. https://github.com/deepseek-ai/DeepEP, 2025b.

DeepSeek. Profiling data in deepseek infra. https://github.com/deepseek-ai/profile-data, 2025a. Accessed: [May 2025].

DeepSeek. Day 6: One more thing, deepseek-v3/r1 inference system overview. https://github.com/deepseek-ai/open-infra-index/blob/main/202502OpenSourceWeek/day_6_one_more_thing_deepseekV3R1_inference_system_overview.md, 2025b. Accessed: [May 2025].

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2023.

Shengyu Liu Jiashi Li. FlashMLA: Efficient mla decoding kernels. https://github.com/deepseek-ai/FlashMLA, 2025.

SGLang 团队。在96个 H100 GPU 上部署 deepseek，采用 pd 分解和大规模专家并行。https://lmsys.org/blog/2025-05-05-large-scale-ep/，2025b。访问时间：[2025年5月]。Jack Choquette。NVIDIA Hopper GPU：提升性能。发表于 *2022 IEEE Hot Chips 34 Symposium (HCS)*，2022 年。Ranajoy Sadhukhan、Jian Chen、Zhuoming Chen、Vashisth Tiwari、Ruihang Lai、Jinyuan Shi、Ian En-Hsu Yen、Avner May、Tianqi Chen、Beidi Chen。MagicDec：打破长上下文生成的延迟-吞吐量权衡，采用推测解码。*arXiv preprint arXiv:2408.11049*，2025。Jiahao Liu、Qifan Wang、Jingang Wang、Xunliang Cai。通过提前退出的推测解码以加速 llm 推断，采用汤普森采样控制机制。*arXiv preprint arXiv:2406.03853*，2024d。Feiye Huo、Jianchao Tan、Kefeng Zhang、Xunliang Cai、Shengli Sun。C2T：一种基于分类器的推测解码树构造方法。*arXiv preprint arXiv:2502.13652*，2025。Amey Agrawal、Ashish Panwar、Jayashree Mohan、Nipun Kwatra、Bhargav S Gulavani、Ramachandran Ramjee。Sarathi：通过搭便车解码与分块预填，提高 LLM 推断效率。*arXiv preprint arXiv:2308.16369*，2023。Chenggang Zhao、Liang Zhao、Jiashi Li、Zhean Xu。DeepGEMM：使用细粒度缩放的干净高效 FP8 GEMM 内核。https://github.com/deepseek-ai/DeepGEMM，2025a。Pengcuo Dege、Qiuming Luo、Rui Mao、Chang Kong。Flash MLA-ETAP：用于加速 NVIDIA H20 GPU 上 MLA 推断的高效转置注意力流水线。
arXiv preprint arXiv:2506.01969，2025。NVIDIA。NVIDIA Collective Communications Library（NCCL）。https://github.com/NVIDIA/nccl。版本 2.21.5。Aashaka Shah、Abhinav Jangda、Binyang Li、Caio Rocha、Changho Hwang、Jithin Jose、Madan Musuvathi、Olli Saarikivi、Peng Cheng、Qinghua Zhou、Roshan Dathathri、Saeed Maleki、Ziyue Yang。MSCCL++：重新思考面向前沿 AI 应用的 GPU 通信抽象。
arXiv preprint arXiv:2504.09014，2025。Qingyuan Li、Yifan Zhang、Liang Li、Peng Yao、Bo Zhang、Xiangxiang Chu、Yerui Sun、Li Du、Yuchen Xie。FPTQ：用于大语言模型的细粒度后训练量化。
arXiv preprint arXiv:2308.15987，2023b。Zunhai Su、Qingyuan Li、Hao Zhang、YuLei Qian、Yuchen Xie、Kehong Yuan。揭示混合专家大语言模型中的超级专家。*arXiv preprint arXiv:2507.23279*，2025。Chenggang Zhao、Shangyan Zhou、Liyue Zhang、Chengqi Deng、Zhean Xu、Yuxuan Liu、Kuai Yu、Jiashi Li、Liang Zhao。DeepEP：一个高效的专家并行通信库。https://github.com/deepseek-ai/DeepEP，2025b。DeepSeek。DeepSeek 基础设施中的性能分析数据。https://github.com/deepseek-ai/profile-data，2025a。访问时间：[2025年5月]。DeepSeek。Day 6：再来一件事，deepseek-v3/r1 推理系统概览。https://github.com/deepseek-ai/open-infra-index/blob/main/202502OpenSourceWeek/day_6_one_more_thing_deepseekV3R1_inference_system_overview.md，2025b。访问时间：[2025年5月]。Shunyu Yao、Jeffrey Zhao、Dian Yu、Nan Du、Izhak Shafran、Karthik Narasimhan、Yuan Cao。React：在语言模型中协同推理与行动。*arXiv preprint arXiv:2210.03629*，2023。Shengyu Liu、Jiashi Li。FlashMLA：高效的 MLA 解码内核。https://github.com/deepseek-ai/FlashMLA，2025。

A Appendix

A.1 Statistics and Case Studies of Dynamic Routing

Figure 11 shows the average activated FFN experts of LongCat-Flash base model across benchmarks. A consistent computational bias favors English tokens over Chinese and mathematical ones. We present a more detailed expert

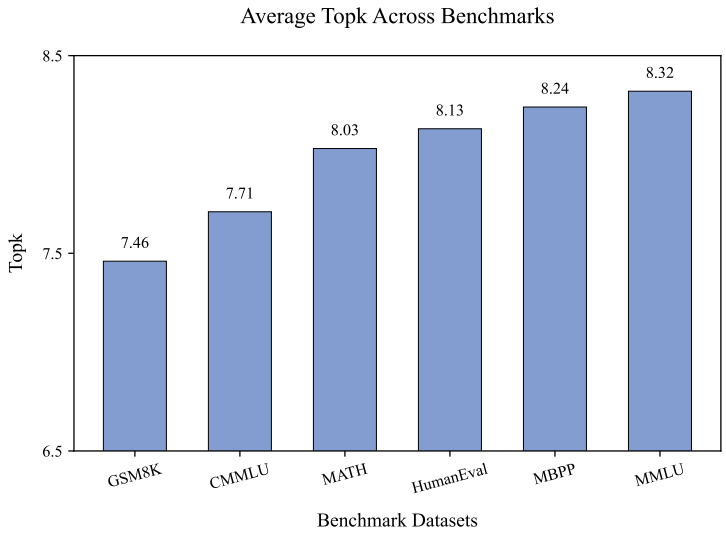


Figure 11: The average number of activated FFN experts across different benchmarks.

selection across different layers for several cases in Table 8. These cases reveal different patterns of expert selection across layers. In the first layer, function words (including articles, conjunctions, prepositions), numbers and punctuation marks consistently receive lower computational resources. In contrast, the final layer (Layer 28) exhibits less specialized feature allocation compared to Layer 1, though identifiable patterns still exist. For example, in the Chinese text case, tokens preceding punctuation marks tend to be assigned fewer computational resources. We hypothesize that shallow layers prioritize token-internal semantics for allocation, while deeper layers dynamically adjust resources based on predictive complexity, potentially reflecting a hierarchical transition from local feature processing to global prediction optimization.

附录 A

A.1 动态路由的统计与案例研究

图11显示了 LongCat-Flash 基线模型在各基准测试中的平均激活的 FFN 专家。一个持续的计算偏差倾向于英语词元，而不是中文词元和数学词元。我们提供一个更详细的专家分析。

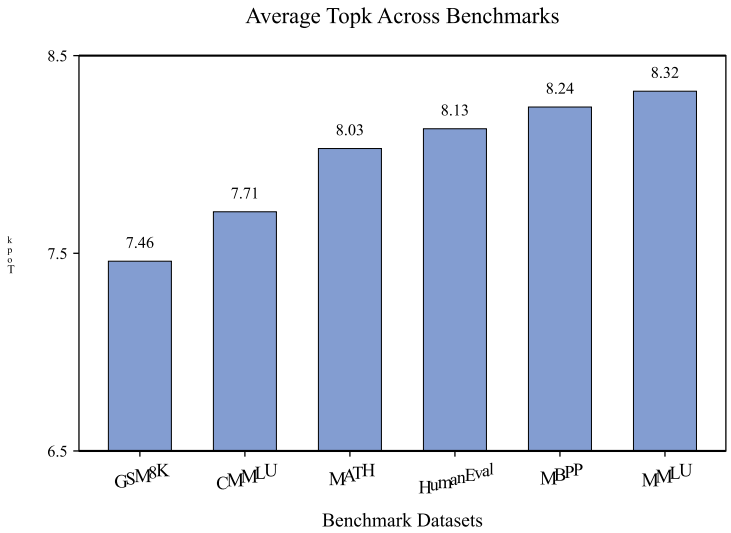


图 11：在不同基准上的被激活的 FFN 专家数量的平均值。

表 8 的若干情形中，在不同层之间的选择。这些情形揭示了跨层的专家选择的不同模式。 在第一层，功能词（包括冠词、连词、介词）、数字和标点符号始终获得较低的计算资源。 相比之下，最后一层（第 28 层）的特征分配不如第一层那样专门化，尽管仍存在可辨识的模式。 例如，在中文文本的情形中，标点符号之前的标记往往被分配到较少的计算资源。 我们假设浅层在资源分配时优先考虑标记内部的语义，而深层则基于预测复杂性动态调整资源，这可能反映了从局部特征处理到全局预测优化的分层转变。

Layer 1 - English

A large language model (LLM) is a language model trained with self-supervised machine learning on a vast amount of text, designed for natural language processing tasks, especially language generation. The largest and most capable LLMs are generative pretrained transformers (GPTs), based on a transformer architecture, which are largely used in generative chatbots such as ChatGPT, Gemini and Claude. LLMs can be fine-tuned for specific tasks or guided by prompt engineering. These models acquire predictive power regarding syntax, semantics, and ontologies inherent in human language corpora, but they also inherit inaccuracies and biases present in the data they are trained on.

Layer 1 - Math

Below is an instruction that describes a task. Write a response that appropriately completes the request.\n\nQuestion: A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take?\nLet's think step by step.\nAnswer: It takes 2/2=<<2/2=1>>1 bolt of white fiber So the total amount of fabric is 2+1=<<2+1=3>>3 bolts of fabric The answer is 3.\nQuestion: In a dance class of 20 students, 20% enrolled in contemporary dance, 25% of the remaining enrolled in jazz dance, and the rest enrolled in hip-hop dance. What percentage of the entire students enrolled in hip-hop dance?\nLet's think step by step.\n

Layer 1 - Code

```
def quick_sort(arr: list) -> list:\n    if len(arr) <= 1:\n        return arr\n    \n    pivot = arr[len(arr) // 2]\n    \n    left = [x for x in arr if x < pivot]\n    middle = [x for x in arr if x == pivot]\n    right = [x for x in arr if x > pivot]\n    \n    return quick_sort(left) + middle + quick_sort(right)\n\n\nif __name__ == "__main__":\n    unsorted_list = [3, 6, 8, 10, 1, 2, 1, 5, 7, 4]\n    sorted_list = quick_sort(unsorted_list)\n    print(sorted_list)\n\n
```

Layer 1 - Chinese

宇宙学中通常所说的“大爆炸”是指：宇宙是在过去有限的时间之前，由一个密度极大且温度极高的太初状态(Initial singularity)演变而来的。根据2015年普朗克卫星所得到的最佳观测结果，宇宙大爆炸距今137.99 ± 0.21亿年，并经过不断的膨胀到达今天的状态。大爆炸这一模型的框架基于爱因斯坦的广义相对论，又在场方程的求解上作出了一定的简化（例如宇宙学原理假设空间的均匀性和各向同性）。1922年，苏联物理学家亚历山大·弗里德曼用广义相对论描述了流体，从而给出了这一模型的场方程。1927年，比利时物理学家乔治·勒梅特通过求解弗里德曼方程已经在理论上提出了同样的观点，这个解后来被称作弗里德曼－勒梅特－罗伯逊－沃尔克度规。

Layer 28 - English

A large language model (LLM) is a language model trained with self-supervised machine learning on a vast amount of text, designed for natural language processing tasks, especially language generation. The largest and most capable LLMs are generative pretrained transformers (GPTs), based on a transformer architecture, which are largely used in generative chatbots such as ChatGPT, Gemini and Claude. LLMs can be fine-tuned for specific tasks or guided by prompt engineering. These models acquire predictive power regarding syntax, semantics, and ontologies inherent in human language corpora, but they also inherit inaccuracies and biases present in the data they are trained on.

Layer 28 - Math

Below is an instruction that describes a task. Write a response that appropriately completes the request.\n\nQuestion: A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take?\nLet's think step by step.\nAnswer: It takes 2/2=<<2/2=1>>1 bolt of white fiber So the total amount of fabric is 2+1=<<2+1=3>>3 bolts of fabric The answer is 3.\nQuestion: In a dance class of 20 students, 20% enrolled in contemporary dance, 25% of the remaining enrolled in jazz dance, and the rest enrolled in hip-hop dance. What percentage of the entire students enrolled in hip-hop dance?\nLet's think step by step.\n

Layer 28 - Code

```
def quick_sort(arr: list) -> list:\n    if len(arr) <= 1:\n        return arr\n    \n    pivot = arr[len(arr) // 2]\n    \n    left = [x for x in arr if x < pivot]\n    middle = [x for x in arr if x == pivot]\n    right = [x for x in arr if x > pivot]\n    \n    return quick_sort(left) + middle + quick_sort(right)\n\n\nif __name__ == "__main__":\n    unsorted_list = [3, 6, 8, 10, 1, 2, 1, 5, 7, 4]\n    sorted_list = quick_sort(unsorted_list)\n    print(sorted_list)\n\n
```

Layer 28 - Chinese

宇宙学中通常所说的“大爆炸”是指：宇宙是在过去有限的时间之前，由一个密度极大且温度极高的太初状态(Initial singularity)演变而来的。根据2015年普朗克卫星所得到的最佳观测结果，宇宙大爆炸距今137.99 ± 0.21亿年，并经过不断的膨胀到达今天的状态。大爆炸这一模型的框架基于爱因斯坦的广义相对论，又在场方程的求解上作出了一定的简化（例如宇宙学原理假设空间的均匀性和各向同性）。1922年，苏联物理学家亚历山大·弗里德曼用广义相对论描述了流体，从而给出了这一模型的场方程。1927年，比利时物理学家乔治·勒梅特通过求解弗里德曼方程已经在理论上提出了同样的观点，这个解后来被称作弗里德曼－勒梅特－罗伯逊－沃尔克度规。

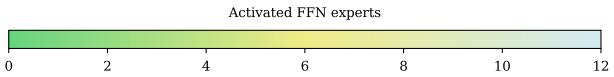


Table 8: The number of activated FFN experts per token across layers.

Layer 1 - English

A large language model (LLM) is a language model trained with self-supervised machine learning on a vast amount of text, designed for natural language processing tasks, especially language generation. The largest and most capable LLMs are generative pretrained transformers (GPTs), based on a transformer architecture, which are largely used in generative chatbots such as ChatGPT, Gemini and Claude. LLMs can be fine-tuned for specific tasks or guided by prompt engineering. These models acquire predictive power regarding syntax, semantics, and ontologies inherent in human language corpora, but they also inherit inaccuracies and biases present in the data they are trained on.

Layer 1 - Math

```
Below is an instruction that describes a task. Write a response that appropriately completes the request.\n\nQuestion: A robe takes 2 bolts of blue fiber and half that much white fiber.  How many bolts in total does it take?\nLet's think step by step.\nAnswer: It takes 2/2=<<2/2=1>>1 bolt of white fiber So the total amount of fabric is 2+1=<<2+1=3>>3 bolts of fabric The answer is 3.\nQuestion: In a dance class of 20 students, 20% enrolled in contemporary dance, 25% of the remaining enrolled in jazz dance, and the rest enrolled in hip-hop dance. What percentage of the entire students enrolled in hip-hop dance?\nLet's think step by step.\n
```

Layer 1 - Code

```
def quick_sort(arr: list) -> list:\n    if len(arr) <= 1:\n        return arr\n    \n    pivot = arr[len(arr) // 2]\n    \n    left = [x for x in arr if x < pivot]\n    middle = [x for x in arr if x == pivot]\n    right = [x for x in arr if x > pivot]\n    \n    return quick_sort(left) + middle + quick_sort(right)\n\n\nif __name__ == "__main__":\n    unsorted_list = [3, 6, 8, 10, 1, 2, 1, 5, 7, 4]\n    sorted_list = quick_sort(unsorted_list)\n    print(sorted_list)\n\n
```

Layer 1 - Chinese

宇宙学中通常所说的“大爆炸”是指：宇宙是在过去有限的时间之前，由一个密度极大且温度极高的太初状态(Initial singularity)演变而来的。根据2015年普朗克卫星所得到的最佳观测结果，宇宙大爆炸距今137.99 ± 0.21亿年，并经过不断的膨胀到达今天的状态。大爆炸这一模型的框架基于爱因斯坦的广义相对论，又在场方程的求解上作出了一定的简化（例如宇宙学原理假设空间的均匀性和各向同性）。1922年，苏联物理学家亚历山大·弗里德曼用广义相对论描述了流体，从而给出了这一模型的场方程。1927年，比利时物理学家乔治·勒梅特通过求解弗里德曼方程已经在理论上提出了同样的观点，这个解后来被称作弗里德曼－勒梅特－罗伯逊－沃尔克度规。

Layer 28 - English

A large language model (LLM) is a language model trained with self-supervised machine learning on a vast amount of text, designed for natural language processing tasks, especially language generation. The largest and most capable LLMs are generative pretrained transformers (GPTs), based on a transformer architecture, which are largely used in generative chatbots such as ChatGPT, Gemini and Claude. LLMs can be fine-tuned for specific tasks or guided by prompt engineering. These models acquire predictive power regarding syntax, semantics, and ontologies inherent in human language corpora, but they also inherit inaccuracies and biases present in the data they are trained on.

Layer 28 - Math

```
Below is an instruction that describes a task. Write a response that appropriately completes the request.\n\nQuestion: A robe takes 2 bolts of blue fiber and half that much white fiber.  How many bolts in total does it take?\nLet's think step by step.\nAnswer: It takes 2/2=<<2/2=1>>1 bolt of white fiber So the total amount of fabric is 2+1=<<2+1=3>>3 bolts of fabric The answer is 3.\nQuestion: In a dance class of 20 students, 20% enrolled in contemporary dance, 25% of the remaining enrolled in jazz dance, and the rest enrolled in hip-hop dance. What percentage of the entire students enrolled in hip-hop dance?\nLet's think step by step.\n
```

Layer 28 - Code

```
def quick_sort(arr: list) -> list:\n    if len(arr) <= 1:\n        return arr\n    \n    pivot = arr[len(arr) // 2]\n    \n    left = [x for x in arr if x < pivot]\n    middle = [x for x in arr if x == pivot]\n    right = [x for x in arr if x > pivot]\n    \n    return quick_sort(left) + middle + quick_sort(right)\n\n\nif __name__ == "__main__":\n    unsorted_list = [3, 6, 8, 10, 1, 2, 1, 5, 7, 4]\n    sorted_list = quick_sort(unsorted_list)\n    print(sorted_list)\n\n
```

Layer 28 - Chinese

宇宙学中通常所说的“大爆炸”是指：宇宙是在过去有限的时间之前，由一个密度极大且温度极高的太初状态(Initial singularity)演变而来的。根据2015年普朗克卫星所得到的最佳观测结果，宇宙大爆炸距今137.99 ± 0.21亿年，并经过不断的膨胀到达今天的状态。大爆炸这一模型的框架基于爱因斯坦的广义相对论，又在场方程的求解上作出了一定的简化（例如宇宙学原理假设空间的均匀性和各向同性）。1922年，苏联物理学家亚历山大·弗里德曼用广义相对论描述了流体，从而给出了这一模型的场方程。1927年，比利时物理学家乔治·勒梅特通过求解弗里德曼方程已经在理论上提出了同样的观点，这个解后来被称作弗里德曼－勒梅特－罗伯逊－沃尔克度规。

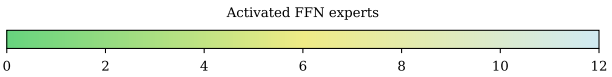


表 8：跨层每个标记激活的 FFN 专家数量。