

DeepSeek-V3 技术报告

DeepSeek-AI

research@deepseek.com

摘要

我们提出了 DeepSeek-V3，这是一种强大的混合专家（MoE）语言模型，总参数量为 671B，每个词元激活 37B 参数。为实现高效推理和经济训练，DeepSeek-V3 采用了多头潜在注意力（MLA）和 DeepSeekMoE 架构，这些架构已在 DeepSeek-V2 中得到充分验证。此外，DeepSeek-V3 首创了无辅助损失的负载均衡策略，并设定了多词元预测训练目标，以获得更强性能。我们使用 14.8 万亿个多样且高质量的词元对 DeepSeek-V3 进行预训练，随后进行监督微调和强化学习阶段，以充分发挥其能力。全面评估表明，DeepSeek-V3 超越了其他开源模型，并达到与领先闭源模型相当的性能。尽管性能卓越，DeepSeek-V3 的完整训练仅需 2.788M 个 H800 GPU 小时。此外，其训练过程非常稳定。在整个训练过程中，我们没有经历任何无法恢复的损失尖峰，也没有执行任何回滚。模型检查点可在 <https://github.com/deepseek-ai/DeepSeek-V3> 获得。

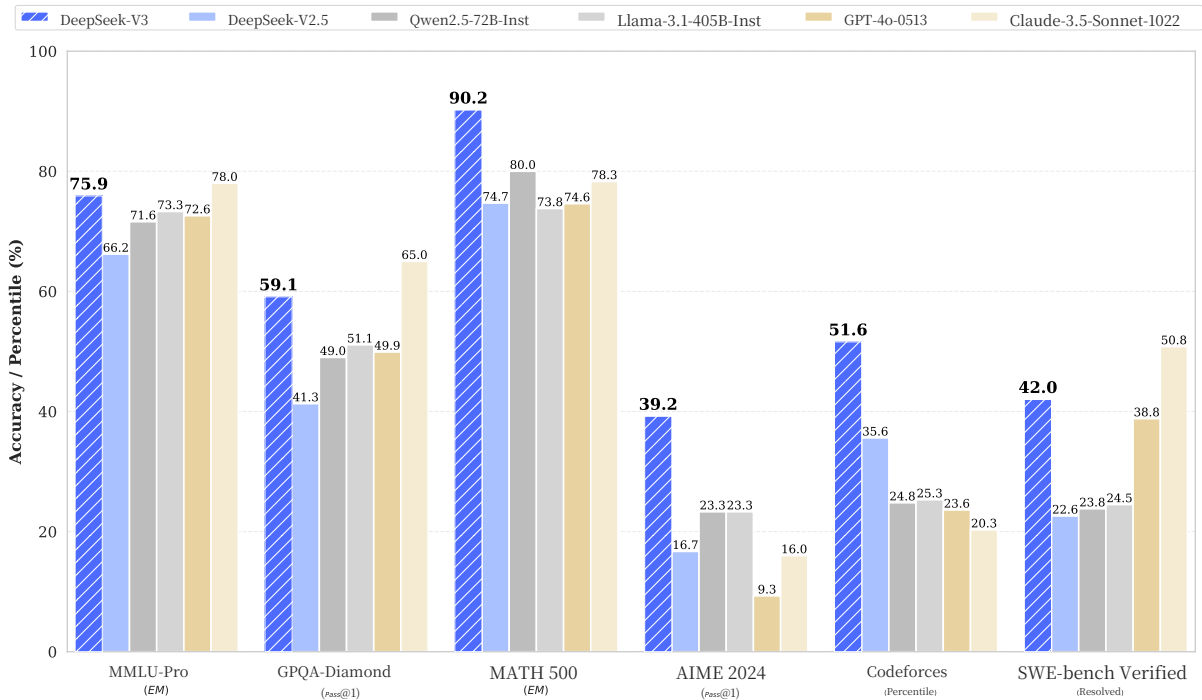


图 1 | DeepSeek-V3 及其同类模型的基准性能。

目录

1 引言	4
2 架构	6
2.1 基础架构.6 2.1.1 多头潜在注意力.7 2.1.2 采用无辅助损失负载均衡的 DeepSeekMoE.8 2.2 多词元预测.10	
3 基础设施	11
3.1 计算集群.....11 3.2 训练框架.....12	
3.2.1 DualPipe 与计算-通信重叠.....12 3.2.2 跨节点全对全通信的高效实现.....13 3.2.3 以最小开销实现极致节省内存.....14 3.3 FP8 训练.....14 3.3.1 混合精度框架.....15 3.3.2 通过量化与乘法提高精度.....16 3.3.3 低精度存储与通信.....18 3.4 推理与部署.....18 3.4.1 预填充.....19 3.4.2 解码.....19 3.5 硬件设计建议.....20 3.5.1 通信硬件.....20 3.5.2 计算硬件.....20	
4 预训练	1
4.1 数据构建.21 4.2 超参数.22 4.3 长上下文扩展.23 4.4 评估.24 4.4.1 评估基准.24 4.4.2 评估结果.24 4.5 讨论.26 4.5.1 多词元预测的消融研究.26 4.5.2 无辅助损失平衡策略的消融研究.26	

4.5.3 按批次负载均衡 VS. 按序列负载均衡.27

5 训练后处理

28

5.1 监督微调.28 5.2 强化学习.29 5.2.1 奖励模型.29 5.2.2 组相对策略优化.30

5.3 评估.30

5.3.1 评估设置.30 5.3.2 标准评估.31 5.3.3 开放式评估.33 5.3.4 作为生成式奖励模型的 DeepSeek-V3.33

5.4 讨论.34 5.4.1 从 DeepSeek-R1 蒸馏.34 5.4.2 自奖励.34 5.4.3 多词元预测评估.35 6 结论、局限性与未来方向 35 A 贡献与致谢 45 B 低精度训练的消融研究 47 B.1 FP8 与 BF16 训练. 47 B.2 关于按块量化的讨论.47 C 16B 基于辅助损失和无辅助损失模型的专家专门化模式 48

1. 引言

近年来，大语言模型（LLM）经历了快速迭代与演化（Anthropic, 2024; Google, 2024; OpenAI, 2024a），逐渐缩小了与通用人工智能（AGI）的差距。除了闭源模型外，开源模型也取得了显著进展，包括 DeepSeek 系列（DeepSeek-AI, 2024a,b,c; Guo et al., 2024）、LLaMA 系列（AI@Meta, 2024a,b; Touvron et al., 2023a,b）、Qwen 系列（Qwen, 2023, 2024a,b）和 Mistral 系列（Jiang et al., 2023; Mistral, 2024），努力缩小与其闭源对应模型之间的差距。为进一步拓展开源模型能力的边界，我们扩大模型规模并推出 DeepSeek-V3，这是一种拥有 671B 参数的大型混合专家（MoE）模型，其中每个词元激活 37B 参数。

着眼于未来，我们始终致力于实现强大的模型性能和经济的成本。因此，在架构方面，DeepSeek-V3 仍采用多头潜在注意力（MLA）（DeepSeek-AI, 2024c）以实现高效推理，并采用 DeepSeekMoE（Dai et al., 2024）以实现高性价比训练。这两种架构已在 DeepSeek-V2（DeepSeek-AI, 2024c）中得到验证，证明它们能够在保持强大模型性能的同时实现高效训练和推理。除基础架构外，我们还实施了两种额外策略，以进一步增强模型能力。首先，DeepSeek-V3 首创了无辅助损失策略（Wang et al., 2024a）来实现负载均衡，旨在最大限度降低为促进负载均衡而对模型性能造成的不利影响。其次，DeepSeek-V3 采用多 token 预测训练目标，我们观察到该目标能够提升模型在评估基准上的整体性能。

为实现高效训练，我们支持 FP8 混合精度训练，并对训练框架实施全面优化。低精度训练已成为高效训练的一种有前景的解决方案（Dettmers et al., 2022; Kalamkar et al., 2019; Narang et al., 2017; Peng et al., 2023b），其发展与硬件能力的进步密切相关（Luo et al., 2024; Micikevicius et al., 2022; Rouhani et al., 2023a）。在本工作中，我们引入了 FP8 混合精度训练框架，并首次在超大规模模型上验证其有效性。通过支持 FP8 计算与存储，我们同时实现了加速训练和降低 GPU 内存使用。在训练框架方面，我们设计了 DualPipe 算法以实现高效流水线并行，该算法具有更少的流水线气泡，并通过计算-通信重叠隐藏训练期间的大部分通信。这种重叠确保随着模型进一步扩展，只要保持恒定的计算-通信比，我们仍可在节点间使用细粒度专家，同时实现近乎为零的全对全通信开销。此外，我们还开发了高效的跨节点全对全通信内核，以充分利用 InfiniBand (IB) 和 NVLink 带宽。进一步地，我们精心优化内存占用，使得无需使用昂贵的张量并行即可训练 DeepSeek-V3。综合这些工作，我们实现了高训练效率。

在预训练期间，我们使用 14.8T 个高质量且多样化的词元训练 DeepSeek-V3。预训练过程非常稳定。在整个训练过程中，我们没有遇到任何无法恢复的损失尖峰，也不必回滚。接下来，我们对 DeepSeek-V3 进行两阶段上下文长度扩展。第一阶段将最大上下文长度扩展至 32K，第二阶段进一步扩展至 128K。随后，我们在 DeepSeek-V3 的基础模型上进行包括监督微调（SFT）和强化学习（RL）在内的后训练，使其与人类偏好对齐并进一步释放其潜力。在后训练阶段，我们从 DeepSeek-R1 系列模型中蒸馏推理能力，同时谨慎地保持模型准确率

训练成本	预训练	上下文扩展	后训练	总计
以 H800 GPU 小时计	2664K	119K	5K	2788K
以美元计	\$5.328M	\$0.238M	\$0.01M	\$5.576M

表 1 | DeepSeek-V3 的训练成本，假设 H800 的租赁价格为每 GPU 小时 \$2 r.

和生成长度之间的平衡。

我们在一系列全面的基准测试上评估了 DeepSeek-V3。尽管训练成本经济，全面评估显示，DeepSeek-V3-Base 已成为目前可用的最强开源基础模型，尤其是在代码和数学方面。其聊天版本也优于其他开源模型，并在一系列标准及开放式基准测试中达到与 GPT-4o 和 Claude-3.5-Sonnet 等领先闭源模型相当的性能。

最后，我们再次强调 DeepSeek-V3 的经济训练成本（总结见表 1），这是通过对算法、框架和硬件进行优化协同设计实现的。在预训练阶段，每训练一万亿个令牌，DeepSeek-V3 仅需 180K H800 GPU 小时，即在配备 2048 个 H800 GPU 的集群上耗时 3.7 天。因此，预训练阶段在不到两个月内完成，耗费 2664K GPU 小时。加上上下文长度扩展所需的 119K GPU 小时和后训练所需的 5K GPU 小时，DeepSeek-V3 的完整训练仅耗费 2.788M GPU 小时。假设 H800 GPU 的租赁价格为每 GPU 小时 \$2，我们的总训练成本仅为 \$5.576M。请注意，上述成本仅包括 DeepSeek-V3 的正式训练，不包括此前针对架构、算法或数据开展的研究与消融实验成本。

我们的主要贡献包括：

架构：创新的负载均衡策略与训练目标

- 在DeepSeek-V2高效架构的基础上，我们率先提出一种无辅助损失的负载均衡策略，从而最大限度地减少因鼓励负载均衡而产生的性能下降。
- 我们研究了多token预测（Multi-Token Prediction, MTP）目标，并证明其有利于模型性能。它还可用于推测解码，以加速推理。

预训练：迈向极致训练效率

- 我们设计了一种FP8混合精度训练框架，并首次验证了在超大规模模型上进行FP8训练的可行性和有效性。
- 通过算法、框架与硬件的协同设计，我们克服了跨节点MoE训练中的通信瓶颈，实现了近乎完全的计算-通信重叠。这显著提升了训练效率并降低了训练成本，使我们能够在不增加额外开销的情况下进一步扩大模型规模。
- 仅以266.4万H800 GPU小时的经济成本，我们便在14.8T个token上完成了DeepSeek-V3的预训练，打造出当前最强的开源基座模型。预训练之后的后续训练阶段仅需10万GPU小时。

后训练：从DeepSeek-R1进行知识蒸馏

- 我们提出了一种创新方法，将长思维链（CoT）模型（具体而言是 DeepSeek R1 系列模型之一）的推理能力蒸馏到标准 LLM，尤其是 DeepSeek-V3 中。我们的流程优雅地融合了

R1 的验证与反思模式，并显著提升其推理性能。同时，我们还保持了对 DeepSeek-V3 输出风格和长度的控制。

核心评估结果总结

- **Knowledge:** (1) 在 MMLU、MMLU-Pro 和 GPQA 等教育基准上，DeepSeek-V3 超越了所有其他开源模型，在 MMLU、MMLU-Pro 和 GPQA 上分别取得 88.5、75.9 和 59.1 的成绩。其性能可与 GPT-4o 和 Claude-Sonnet-3.5 等领先闭源模型相媲美，缩小了该领域开源模型与闭源模型之间的差距。(2) 在事实性基准上，DeepSeek-V3 在 SimpleQA 和 Chinese SimpleQA 上均展现出开源模型中的优越性能。尽管其英语事实知识 (SimpleQA) 落后于 GPT-4o 和 Claude-Sonnet-3.5，但在中文事实知识 (Chinese SimpleQA) 上超越了这些模型，凸显了其在中文事实知识方面的优势。
- **Code, Math, and Reasoning:** (1) 在所有非长 CoT 的开源和闭源模型中，DeepSeek-V3 在数学相关基准上达到了最先进的性能。值得注意的是，在 MATH-500 等特定基准上，它甚至超越了 o1-preview，展现出强大的数学推理能力。(2) 在编码相关任务上，DeepSeek-V3 成为 LiveCodeBench 等编码竞赛基准中表现最佳的模型，巩固了其在该领域的领先地位。对于工程相关任务，尽管 DeepSeek-V3 的表现略低于 Claude-Sonnet-3.5，但仍以显著优势超过所有其他模型，展现出其在多种技术基准上的竞争力。

在本文余下部分，我们首先详细介绍 DeepSeek-V3 的模型架构（第 2 节）。随后，我们介绍基础设施，包括计算集群、训练框架、对 FP8 训练的支持、推理部署策略以及对未来硬件设计的建议。接下来，我们描述预训练过程，包括训练数据构建、超参数设置、长上下文扩展技术、相关评估以及一些讨论（第 4 节）。此后，我们讨论后训练工作，包括监督微调（SFT）、强化学习（RL）、相应的评估和讨论（第 5 节）。最后，我们总结本工作，讨论 DeepSeek-V3 现有的局限性，并提出未来研究的潜在方向（第 6 节）。

2. 架构

我们首先介绍 DeepSeek-V3 的基本架构，其特点是采用多头潜在注意力（MLA）（DeepSeek-AI, 2024c）实现高效推理，并采用 DeepSeekMoE（Dai et al., 2024）实现经济训练。随后，我们提出多令牌预测（MTP）训练目标，并观察到它能够提升评估基准上的整体性能。对于未明确提及的其他细节，DeepSeek-V3 遵循 DeepSeek-V2（DeepSeek-AI, 2024c）的设置。

2.1. 基本架构

DeepSeek-V3 的基本架构仍处于 Transformer（Vaswani et al., 2017）框架内。为实现高效推理和经济训练，DeepSeek-V3 还采用了 MLA 和 DeepSeekMoE，这些技术已由 DeepSeek-V2 充分验证。与 DeepSeek-V2 相比，不同之处在于我们额外引入了无辅助损失的负载均衡

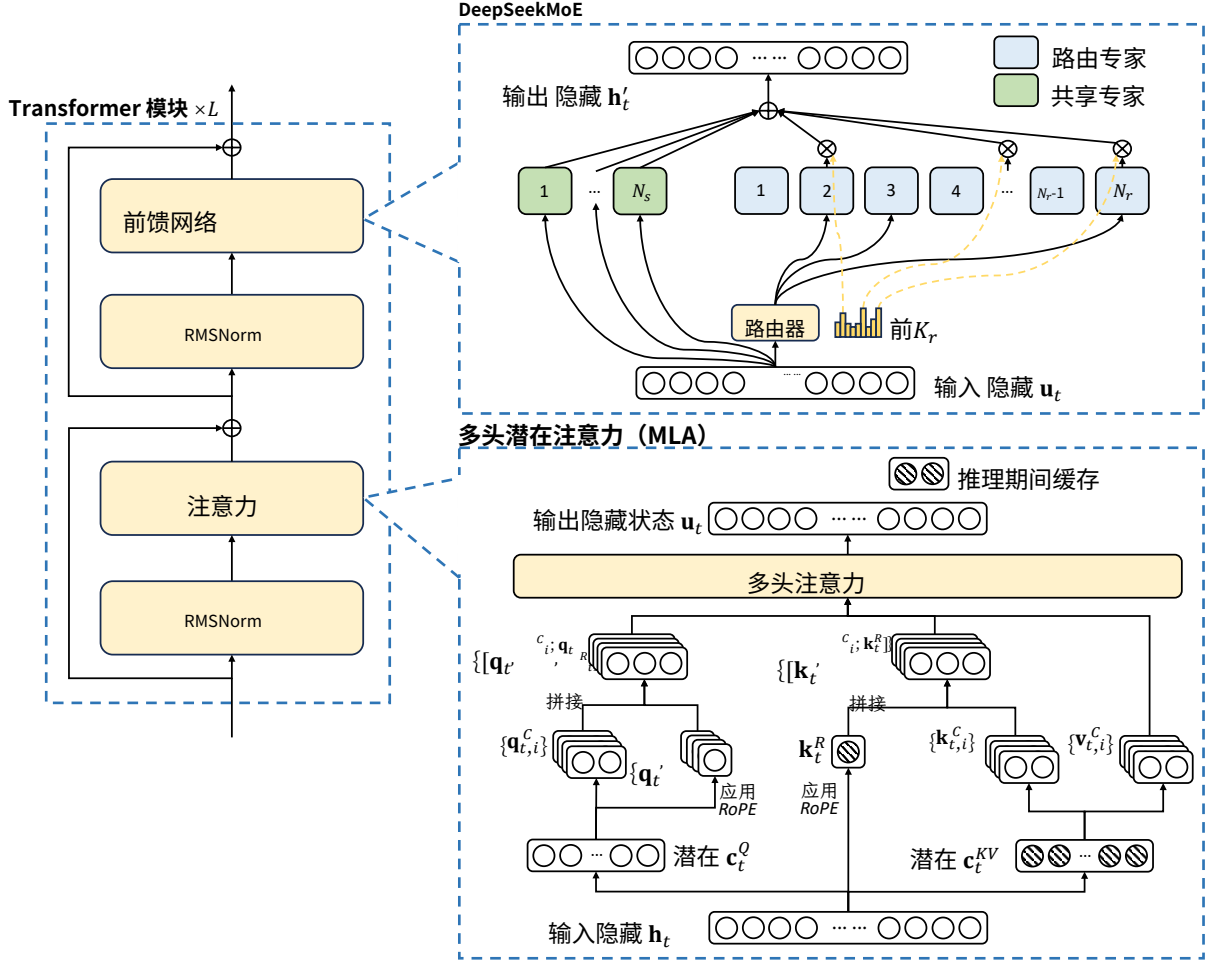


图 2 | DeepSeek-V3 的基本架构示意图。继承 DeepSeek-V2，我们采用 MLA 和 DeepSeekMoE，以实现高效推理和经济的训练。

策略（Wang et al., 2024a），用于缓解为确保负载均衡所付出努力导致的性能下降。图 2 展示了 DeepSeek-V3 的基本架构，本节将简要回顾 MLA 和 DeepSeekMoE 的细节。

2.1.1. 多头潜在注意力

对于注意力，DeepSeek-V3 采用 MLA 架构。令 d 表示嵌入维度， n_h 表示注意力头的数量， d_h 表示每个头的维度，并令 $\mathbf{h}_t \in \mathbb{R}^d$ 表示给定注意力层中第 t 个词元的注意力输入。MLA 的核心是对注意力键和值进行低秩联合压缩，以减少推理期间的键值（KV）缓存：

$$\mathbf{c}_t^{KV} = W^{DKV} \mathbf{h}_t, \quad (1)$$

$$[\mathbf{k}_{t,1}^C; \mathbf{k}_{t,2}^C; \dots; \mathbf{k}_{t,n_h}^C] = \mathbf{k}_t^C = W^{UK} \mathbf{c}_t^{KV}, \quad (2)$$

$$\mathbf{k}_t^R = \text{RoPE}(W^{KR} \mathbf{h}_t), \quad (3)$$

$$\mathbf{k}_{t,i} = [\mathbf{k}_{t,i}^C; \mathbf{k}_t^R], \quad (4)$$

$$[\mathbf{v}_{t,1}^C; \mathbf{v}_{t,2}^C; \dots; \mathbf{v}_{t,n_h}^C] = \mathbf{v}_t^C = W^{UV} \mathbf{c}_t^{KV}, \quad (5)$$

其中 $\mathbf{c}_t^{KV} \in \mathbb{R}^{d_c}$ 是键和值的压缩潜在向量； $d_c (\ll d_h n_h)$ 表示 KV 压缩维度； $W^{DKV} \in \mathbb{R}^{d_c \times d}$ 表示下投影矩阵； $W^{UK}, W^{UV} \in \mathbb{R}^{d_h n_h \times d_c}$ 分别是键和值的上投影矩阵； $W^{KR} \in \mathbb{R}^{d_h \times d}$ 是用于生成携带旋转位置嵌入（RoPE）（Su et al., 2024）的解耦键的矩阵； $\text{RoPE}(\cdot)$ 表示应用 RoPE 矩阵的操作；并且 $[\cdot; \cdot]$ 表示拼接。注意，对于 MLA，在生成期间仅需缓存蓝框向量（即 $\mathbf{c}_t^{KV}$ 和 $\mathbf{k}_t^R$ ），这在保持与标准多头注意力（MHA）（Vaswani et al., 2017）相当性能的同时，显著减少了 KV 缓存。

对于注意力查询，我们同样执行低秩压缩，这可以减少训练期间的激活内存：

$$\mathbf{c}_t^Q = W^{DQ} \mathbf{h}_t, \quad (6)$$

$$[\mathbf{q}_{t,1}^C; \mathbf{q}_{t,2}^C; \dots; \mathbf{q}_{t,n_h}^C] = \mathbf{q}_t^C = W^{UQ} \mathbf{c}_t^Q, \quad (7)$$

$$[\mathbf{q}_{t,1}^R; \mathbf{q}_{t,2}^R; \dots; \mathbf{q}_{t,n_h}^R] = \mathbf{q}_t^R = \text{RoPE}(W^{QR} \mathbf{c}_t^Q), \quad (8)$$

$$\mathbf{q}_{t,i} = [\mathbf{q}_{t,i}^C; \mathbf{q}_{t,i}^R], \quad (9)$$

其中 $\mathbf{c}_t^Q \in \mathbb{R}^{d_c'}$ 是查询的压缩潜在向量； $d_c' (\ll d_h n_h)$ 表示查询压缩维度； $W^{DQ} \in \mathbb{R}^{d_c' \times d}$, $W^{UQ} \in \mathbb{R}^{d_h n_h \times d_c'}$ 分别是查询的下投影矩阵和上投影矩阵； $W^{QR} \in \mathbb{R}^{d_h n_h \times d_c'}$ 是用于生成携带 RoPE 的解耦查询的矩阵。

最终，注意力查询（ $\mathbf{q}_{t,i}$ ）、键（ $\mathbf{k}_{j,i}$ ）和值（ $\mathbf{v}_{j,i}^C$ ）组合后得到最终的注意力输出 $\mathbf{u}_t$ ：

$$\mathbf{o}_{t,i} = \sum_{j=1}^t \text{Softmax}_j \left(\frac{\mathbf{q}_{t,i}^T \mathbf{k}_{j,i}}{\sqrt{d_h + d_h^R}} \right) \mathbf{v}_{j,i}^C, \quad (10)$$

$$\mathbf{u}_t = W^O [\mathbf{o}_{t,1}; \mathbf{o}_{t,2}; \dots; \mathbf{o}_{t,n_h}], \quad (11)$$

其中， $W^O \in \mathbb{R}^{d \times d_h n_h}$ 表示输出投影矩阵。

2.1.2. 采用无辅助损失负载均衡的 **DeepSeekMoE**

DeepSeekMoE 的基本架构。对于前馈网络（FFN），DeepSeek-V3 采用 DeepSeekMoE 架构（Dai et al., 2024）。与 GShard（Lepikhin et al., 2021）等传统 MoE 架构相比，DeepSeekMoE 使用更细粒度的专家，并将部分专家隔离为共享专家。令 $\mathbf{u}_t$ 表示第 t 个词元的 FFN 输入，我们按如下方式计算 FFN 输出 $\mathbf{h}_t'$ ：

$$\mathbf{h}_t' = \mathbf{u}_t + \sum_{i=1}^{N_s} \text{FFN}_i^{(s)}(\mathbf{u}_t) + \sum_{i=1}^{N_r} g_{i,t} \text{FFN}_i^{(r)}(\mathbf{u}_t), \quad (12)$$

$$g_{i,t} = \frac{g'_{i,t}}{\sum_{j=1}^{N_r} g'_{j,t}}, \quad (13)$$

$$g'_{i,t} = \begin{cases} s_{i,t}, & s_{i,t} \in \text{Topk}(\{s_{j,t} | 1 \leq j \leq N_r\}, K_r), \\ 0, & \text{otherwise}, \end{cases} \quad (14)$$

$$s_{i,t} = \text{Sigmoid}(\mathbf{u}_t^T \mathbf{e}_i), \quad (15)$$

其中, N_s 和 N_r 分别表示共享专家和路由专家的数量; $\text{FFN}_i^{(s)}(\cdot)$ 和 $\text{FFN}_i^{(r)}(\cdot)$ 分别表示第 i 个共享专家和第 i 个路由专家; K_r 表示激活的路由专家数量; $g_{i,t}$ 是第 i 个专家的门控值; $s_{i,t}$ 是词元与专家之间的亲和度; $\mathbf{e}_i$ 是第 i 个路由专家的质心向量; $\text{Topk}(\cdot, K)$ 表示一个集合, 其中包含针对第 t 个词元与所有路由专家计算得到的亲和度分数中最高的 K 个分数。与 DeepSeek-V2 略有不同, DeepSeek-V3 使用 sigmoid 函数计算亲和度分数, 并在所有选定的亲和度分数之间进行归一化, 以生成门控值。

Auxiliary-Loss-Free Load Balancing. 对于 MoE 模型, 不均衡的专家负载会导致路由崩溃 (Shazeer et al., 2017), 并在专家并行场景下降低计算效率。传统解决方案通常依赖辅助损失 (Fedus et al., 2021; Lepikhin et al., 2021) 来避免负载不均衡。然而, 过大的辅助损失会损害模型性能 (Wang et al., 2024a)。为了在负载均衡与模型性能之间实现更好的权衡, 我们首创了无辅助损失的负载均衡策略 (Wang et al., 2024a) 以确保负载均衡。具体而言, 我们为每个专家引入偏置项 b_i , 并将其加到相应的亲和度分数 $s_{i,t}$ 上, 以确定 top-K 路由:

$$g'_{i,t} = \begin{cases} s_{i,t}, & s_{i,t} + b_i \in \text{Topk}(\{s_{j,t} + b_j | 1 \leq j \leq N_r\}, K_r), \\ 0, & \text{otherwise.} \end{cases} \quad (16)$$

请注意, 偏置项仅用于路由。将与 FFN 输出相乘的门控值仍然来自原始亲和度分数 $s_{i,t}$ 。在训练期间, 我们持续监控每个训练步骤整个批次上的专家负载。在每一步结束时, 如果对应专家过载, 我们将偏置项减小 γ ; 如果对应专家负载不足, 则将其增加 γ , 其中 γ 是称为偏置更新速度的超参数。通过动态调整, DeepSeek-V3 在训练期间保持了均衡的专家负载, 并且性能优于通过纯辅助损失促进负载均衡的模型。

序列级互补辅助损失。 尽管 DeepSeek-V3 主要依赖无辅助损失策略来实现负载均衡, 但为防止任一序列内部出现极端不平衡, 我们还采用了互补的序列级均衡损失:

$$\mathcal{L}_{\text{Bal}} = \alpha \sum_{i=1}^{N_r} f_i P_i, \quad (17)$$

$$f_i = \frac{N_r}{K_r T} \sum_{t=1}^T \mathbb{1}(s_{i,t} \in \text{Topk}(\{s_{j,t} | 1 \leq j \leq N_r\}, K_r)), \quad (18)$$

$$s'_{i,t} = \frac{s_{i,t}}{\sum_{j=1}^{N_r} s_{j,t}}, \quad (19)$$

$$P_i = \frac{1}{T} \sum_{t=1}^T s'_{i,t}, \quad (20)$$

其中, 平衡因子 α 是一个超参数, 对于 DeepSeek-V3 将被赋予极小的值; $\mathbb{1}(\cdot)$ 表示指示函数; T 表示序列中的词元数量。按序列的平衡损失促使每个序列上的专家负载保持均衡。

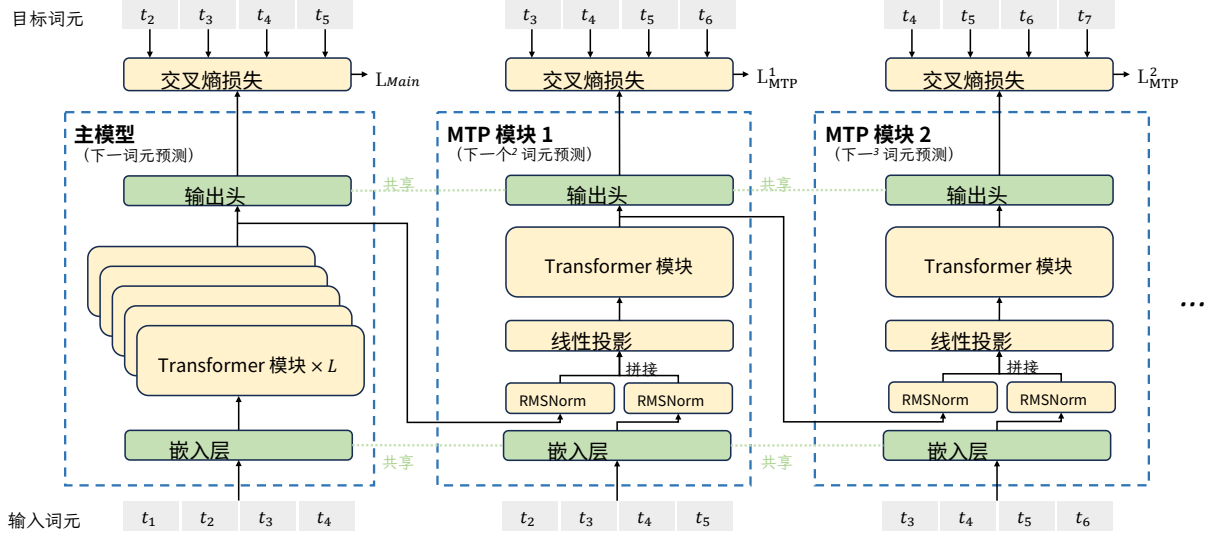


图 3 | 多token预测（MTP）实现示意图。我们保留每个深度对每个token进行预测的完整因果链。

节点受限路由。类似于 DeepSeek-V2 使用的设备受限路由，DeepSeek-V3 也采用受限路由机制，以限制训练期间的通信成本。简而言之，我们确保每个词元最多发送到 M 个节点，这些节点根据每个节点上分布的专家的最高 K_M 个亲和度分数之和进行选择。在此约束下，我们的 MoE 训练框架几乎可以实现计算与通信的完全重叠。

不丢弃token。得益于有效的负载均衡策略，DeepSeek-V3在完整训练过程中保持了良好的负载均衡。因此，DeepSeek-V3在训练期间不会丢弃任何token。此外，我们还实施了特定的部署策略以确保推理负载均衡，因此DeepSeek-V3在推理期间同样不会丢弃token。

2.2. 多token预测

受 Gloeckle et al. (2024) 启发，我们为 DeepSeek-V3 研究并设定了多词元预测（MTP）目标，将每个位置的预测范围扩展到多个未来词元。一方面，MTP 目标使训练信号更加密集，并可能提高数据效率。另一方面，MTP 可能使模型能够预先规划其表示，从而更好地预测未来词元。图 3 展示了我们的 MTP 实现。不同于 Gloeckle et al. (2024) 使用独立输出头并行预测 D 个附加词元，我们按顺序预测附加词元，并在每个预测深度保留完整的因果链。本节将介绍 MTP 实现的详细信息。

MTP 模块。具体而言，我们的 MTP 实现使用 D 个顺序模块来预测 D 个额外词元。第 k 个 MTP 模块由共享嵌入层 $\text{Emb}(\cdot)$ 、共享输出头 $\text{OutHead}(\cdot)$ 、Transformer 块 $\text{TRM}_k(\cdot)$ 和投影矩阵 $M_k \in \mathbb{R}^{d \times 2d}$ 组成。对于第 i 个输入词元 t_i ，在第 k 个预测深度，我们首先将第 i 个词元在第 $(k-1)$ 个深度的表示 $\mathbf{h}_i^{k-1} \in \mathbb{R}^d$ 与第 $(i+k)$ 个词元的嵌入 $\text{Emb}(t_{i+k}) \in \mathbb{R}^d$ 结合

并进行线性投影：

$$\mathbf{h}_i'^k = M_k[\text{RMSNorm}(\mathbf{h}_i^{k-1}); \text{RMSNorm}(\text{Emb}(t_{i+k}))], \quad (21)$$

其中 $[\cdot; \cdot]$ 表示拼接。特别是，当 $k = 1$ 时， $\mathbf{h}_i^{k-1}$ 表示主模型给出的表示。请注意，对于每个 MTP 模块，其嵌入层与主模型共享。组合后的 $\mathbf{h}_i'^k$ 作为 Transformer 块在第 k 个深度处的输入，以生成当前深度的输出表示 $\mathbf{h}_i^k$ ：

$$\mathbf{h}_{1:T-k}^k = \text{TRM}_k(\mathbf{h}_{1:T-k}'), \quad (22)$$

其中 T 表示输入序列长度， $i:j$ 表示切片操作（包括左右边界）。最后，以 $\mathbf{h}_i^k$ 为输入，共享输出头将计算第 k 个额外预测令牌 $p_{i+1+k}^k \in \mathbb{R}^V$ 的概率分布，其中 V 为词汇表大小：

$$p_{i+k+1}^k = \text{OutHead}(\mathbf{h}_i^k). \quad (23)$$

输出头 $\text{OutHead}(\cdot)$ 将表示线性映射为 logits，随后应用 $\text{Softmax}(\cdot)$ 函数，计算第 k 个额外令牌的预测概率。此外，对于每个 MTP 模块，其输出头与主模型共享。我们维持预测因果链的原则与 EAGLE (Li et al., 2024b) 类似，但其主要目标是推测解码 (Leviathan et al., 2023; Xia et al., 2023)，而我们利用 MTP 来改进训练。

MTP 训练目标。 对于每个预测深度，我们计算交叉熵损失 $\mathcal{L}^k_{\text{MTP}}$ ：

$$\mathcal{L}_{\text{MTP}}^k = \text{CrossEntropy}(p_{2+k:T+1}^k, t_{2+k:T+1}) = -\frac{1}{T} \sum_{i=2+k}^{T+1} \log p_i^k[t_i], \quad (24)$$

其中， T 表示输入序列长度， t_i 表示第 i 个位置的真实词元， $p_i^k[t_i]$ 表示给定第 k 个 MTP 模块时 t_i 的相应预测概率。最后，我们计算所有深度上的 MTP 损失的平均值，并将其乘以权重因子 λ ，得到总体 MTP 损失 $\mathcal{L}_{\text{MTP}}$ ，该损失作为 DeepSeek-V3 的附加训练目标：

$$\mathcal{L}_{\text{MTP}} = \frac{\lambda}{D} \sum_{k=1}^D \mathcal{L}_{\text{MTP}}^k. \quad (25)$$

MTP 在推理中的应用。 我们的 MTP 策略主要旨在提升主模型的性能，因此在推理期间，我们可以直接丢弃 MTP 模块，主模型仍可独立且正常地运行。此外，我们还可以将这些 MTP 模块重新用于推测解码，以进一步降低生成延迟。

3. 基础设施

3.1. 计算集群

DeepSeek-V3 在配备 2048 个 NVIDIA H800 GPU 的集群上训练。H800 集群中的每个节点包含 8 个 GPU，节点内通过 NVLink 和 NVSwitch 连接。不同节点之间采用 InfiniBand (IB) 互连以促进通信。

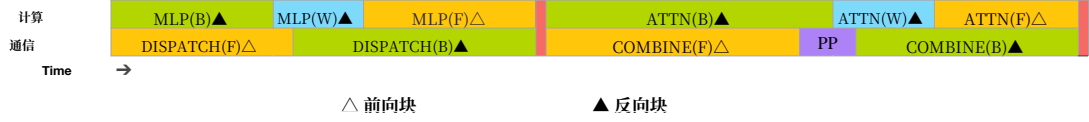


图 4 | 一对独立前向和反向块的重叠策略（Transformer 块的边界未对齐）。橙色表示前向，绿色表示“输入反向”，蓝色表示“权重反向”，紫色表示 PP 通信，红色表示屏障。all-to-all 和 PP 通信均可被完全隐藏。

3.2. 训练框架

DeepSeek-V3 的训练由 HAI-LLM 框架支持，这是我们的工程师从零打造的高效、轻量级训练框架。总体而言，DeepSeek-V3 采用 16 路流水线并行（PP）（Qi et al., 2023a）、跨越 8 个节点的 64 路专家并行（EP）（Lepikhin et al., 2021），以及 ZeRO-1 数据并行（DP）（Rajbhandari et al., 2020）。

为了促进 DeepSeek-V3 的高效训练，我们实施了精细的工程优化。首先，我们设计了 DualPipe 算法以实现高效的流水线并行。与现有 PP 方法相比，DualPipe 的流水线气泡更少。更重要的是，它在前向和反向过程中重叠计算与通信阶段，从而解决跨节点专家并行带来的高通信开销挑战。其次，我们开发了高效的跨节点 all-to-all 通信内核，以充分利用 IB 和 NVLink 带宽，并节省专用于通信的流式多处理器（SM）。最后，我们对训练期间的内存占用进行了精细优化，使我们能够在不使用昂贵张量并行（TP）的情况下训练 DeepSeek-V3。

3.2.1. DualPipe 与计算-通信重叠

对于 DeepSeek-V3，跨节点专家并行引入的通信开销导致计算与通信的比例低效，约为 1:1。为应对这一挑战，我们设计了一种名为 DualPipe 的创新流水线并行算法，它不仅通过有效重叠前向与反向计算-通信阶段来加速模型训练，还减少了流水线气泡。

DualPipe 的核心思想是将一对独立的前向和反向块内的计算与通信重叠。具体而言，我们将每个块划分为四个组件：attention、all-to-all dispatch、MLP 和 all-to-all combine。特别地，对于反向块，attention 和 MLP 进一步拆分为两部分，即 backward for input 和 backward for weights，如 ZeroBubble（Qi et al., 2023b）中所示。此外，我们还有一个 PP communication 组件。如图 4 所示，对于一对前向和反向块，我们重新排列这些组件，并手动调整专用于通信与计算的 GPU SM 比例。在这种重叠策略中，我们可以确保执行期间 all-to-all 和 PP 通信都被完全隐藏。得益于高效的重叠策略，完整的 DualPipe 调度如图 5 所示。它采用双向流水线调度，同时从流水线两端输入微批次，并且相当大一部分通信可以完全重叠。这种重叠还确保，随着模型进一步扩展，只要保持恒定的计算通信比，我们仍可在节点间采用细粒度专家，同时实现近乎为零的 all-to-all 通信开销。



图 5 | 8 个 PP rank 和两个方向上 20 个微批次的 DualPipe 调度示例。反向方向的微批次与前向方向对称，因此为简化图示省略其批次 ID。由共享黑色边框包围的两个单元格表示计算与通信相互重叠。

方法	气泡	参数	激活
1F1B	$(PP - 1)(F + B)$	$1\times$	PP
ZB1P	$(PP - 1)(F + B - 2W)$	$1\times$	PP
DualPipe (我们的方法)	$(\frac{PP}{2} - 1)(F+B + B - 3W)$	$2\times$	$PP + 1$

表 2 | 不同流水线并行方法的流水线气泡和内存使用对比。 F 表示前向块的执行时间， B 表示完整反向块的执行时间， W 表示“权重反向”块的执行时间， $F\&B$ 表示两个相互重叠的前向和反向块的执行时间。

此外，即使在没有繁重通信负担的更一般场景中，DualPipe 仍表现出效率优势。表 2 总结了不同 PP 方法的流水线气泡和内存使用情况。如表所示，与 ZB1P (Qi et al., 2023b) 和 1F1B (Harlap et al., 2018) 相比，DualPipe 显著减少了流水线气泡，而峰值激活内存仅增加 $\frac{1}{PP}$ 倍。尽管 DualPipe 需要保留两份模型参数，但由于训练期间采用较大的 EP 大小，这不会显著增加内存消耗。与 Chimera (Li and Hoefler, 2021) 相比，DualPipe 只要求流水线阶段数和微批次次数可被 2 整除，而不要求微批次次数能被流水线阶段数整除。此外，对于 DualPipe，随着微批次次数增加，气泡和激活内存都不会增加。

3.2.2. 跨节点 All-to-All 通信的高效实现

为确保 DualPipe 具备充足的计算性能，我们定制了高效的跨节点 all-to-all 通信内核（包括分发和合并），以减少专用于通信的 SM 数量。内核的实现与 MoE 门控算法以及集群网络拓扑协同设计。具体而言，在我们的集群中，跨节点 GPU 通过 IB 完全互连，节点内通信则通过 NVLink 处理。NVLink 提供 160 GB/s 的带宽，约为 IB (50 GB/s) 的 3.2 倍。为有效利用 IB 和 NVLink 的不同带宽，我们将每个词元限制为最多分发到 4 个节点，从而减少 IB 流量。对于每个词元，在作出路由决策后，它首先通过 IB 传输到目标节点上具有相同节点内索引的 GPU。当到达目标节点后，我们将尽力确保其通过 NVLink 即时转发到承载目标专家的特定 GPU，而不会被随后到达的词元阻塞。这样，通过 IB 和 NVLink 的通信便可完全重叠，并且每个词元能够在不产生额外 NVLink 开销的情况下，高效地为每个节点平均选择 3.2 个专家。这意味着，尽管 DeepSeek-V3

实际上只选择 8 个路由专家，但它可以将这一数量扩展至最多 13 个专家（4 个节点 $\times$ 3.2 个专家/节点），同时保持相同的通信成本。总体而言，在这种通信策略下，仅需 20 个 SM 即可充分利用 IB 和 NVLink 的带宽。

具体而言，我们采用 warp 专门化技术（Bauer et al., 2014），并将 20 个 SM 划分为 10 个通信通道。在分发过程中，（1）IB 发送、（2）IB 到 NVLink 转发以及（3）NVLink 接收分别由相应的 warp 处理。根据所有 SM 上的实际工作负载，动态调整分配给每项通信任务的 warp 数量。同样，在合并过程中，（1）NVLink 发送、（2）NVLink 到 IB 转发与累加以及（3）IB 接收与累加也由动态调整的 warp 处理。此外，分发和合并内核都与计算流重叠，因此我们还考虑它们对其他 SM 计算内核的影响。具体而言，我们采用定制的 PTX（Parallel Thread Execution）指令，并对通信块大小进行自动调优，从而显著减少 L2 缓存的使用以及对其他 SM 的干扰。

3.2.3. 以最小开销实现极致内存节省

为了降低训练期间的内存占用，我们采用以下技术

s.

重新计算 RMSNorm 和 MLA 上投影。 我们在反向传播期间重新计算所有 RMSNorm 操作和 MLA 上投影，从而无需持续存储其输出激活。该策略仅带来轻微开销，却显著降低了存储激活所需的内存。

CPU 中的指数移动平均。 训练期间，我们保留模型参数的指数移动平均（EMA），以便在学习率衰减后及早估计模型性能。EMA 参数存储在 CPU 内存中，并在每个训练步骤后异步更新。该方法使我们能够维护 EMA 参数，而不会产生额外的内存或时间开销。

多 token 预测的共享嵌入和输出头。 借助 DualPipe 策略，我们将模型最浅层（包括嵌入层）和最深层（包括输出头）部署在同一 PP rank 上。这种安排使 MTP 模块与主模型之间的共享嵌入和输出头能够实现参数与梯度的物理共享。这种物理共享机制进一步提高了我们的内存效率。

3.3. FP8 训练

受近期低精度训练进展的启发（Dettmers et al., 2022; Noune et al., 2022; Peng et al., 2023b），我们提出了一种细粒度混合精度框架，利用 FP8 数据格式训练 DeepSeek-V3。尽管低精度训练前景广阔，但其通常受到激活值、权重和梯度中离群值存在的限制（Fishman et al., 2024; He et al.; Sun et al., 2024）。尽管推理量化已取得显著进展（Frantar et al., 2022; Xiao et al., 2023），但展示低精度技术成功应用于大规模语言模型的研究相对较少

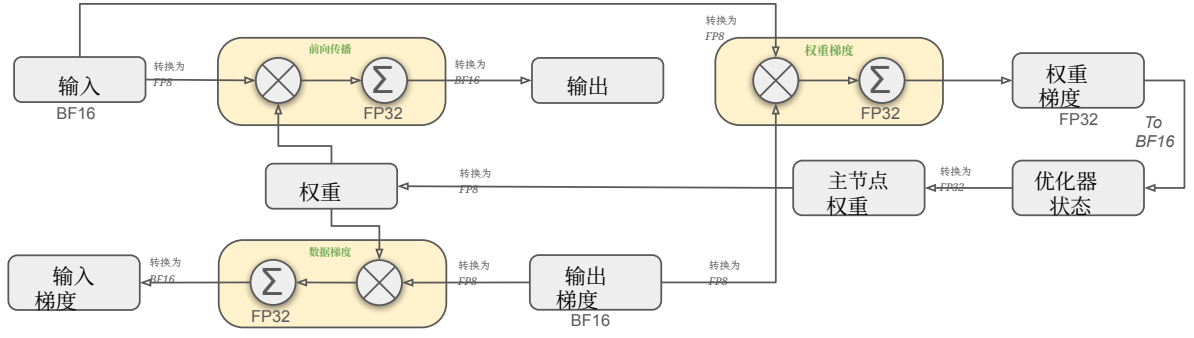


图 6 | 采用 FP8 数据格式的整体混合精度框架。为便于说明，仅展示 Linearoperator。

预训练 (Fishman et al., 2024)。为应对这一挑战并有效扩展 FP8 格式的动态范围，我们引入了一种细粒度量化策略：使用 $1 \times N_c$ 个元素进行逐 tile 分组，或使用 $N_c \times N_c$ 个元素进行逐块分组。在我们的高精度累加过程中，相关反量化开销得到大幅缓解，这是实现准确 FP8 通用矩阵乘法 (GEMM) 的关键。此外，为进一步降低 MoE 训练中的内存和通信开销，我们以 FP8 缓存和分发激活，同时以 BF16 存储低精度优化器状态。我们在两个规模类似于 DeepSeek-V2-Lite 和 DeepSeek-V2 的模型上验证了所提出的 FP8 混合精度框架，训练 token 数约为 1 万亿 (更多细节见附录 B.1)。值得注意的是，与 BF16 基线相比，我们的 FP8 训练模型的相对损失误差始终低于 0.25%，明显处于训练随机性可接受的范围内。

3.3.1. 混合精度框架

基于低精度训练中广泛采用的技术 (Kalamkar et al., 2019; Narang et al., 2017)，我们提出了用于 FP8 训练的混合精度框架。在该框架中，大多数计算密集型操作以 FP8 执行，同时策略性地将少数关键操作保留为其原始数据格式，以平衡训练效率和数值稳定性。整体框架如图 6 所示。

首先，为加速模型训练，大多数核心计算内核，即 GEMM 操作，均以 FP8 精度实现。这些 GEMM 操作接收 FP8 张量作为输入，并以 BF16 或 FP32 生成输出。如图 6 所示，与 Linear operator 相关的三个 GEMM，即 Fprop (前向传播)、Dgrad (激活反向传播) 和 Wgrad (权重反向传播)，均以 FP8 执行。理论上，与原始 BF16 方法相比，这一设计可使计算速度翻倍。此外，FP8 WgradGEMM 允许将激活以 FP8 存储，以供反向传播使用，从而显著降低内存消耗。

尽管 FP8 格式具有效率优势，但某些算子由于对低精度计算敏感，仍需要更高精度。此外，一些低成本算子也可以使用更高精度，而对总体训练成本的开销可忽略不计。因此，经过仔细研究，我们对以下组件保持原始精度 (例如 BF16 或 FP32)：嵌入模块、输出头、MoE 门控模块、归一化算子和注意力算子。这些有针对性地保留高精度可确保 DeepSeek-V3 的训练动态稳定。为进一步保证数值稳定性，我们以更高精度存储主权重、权重梯度和优化器状态。尽管

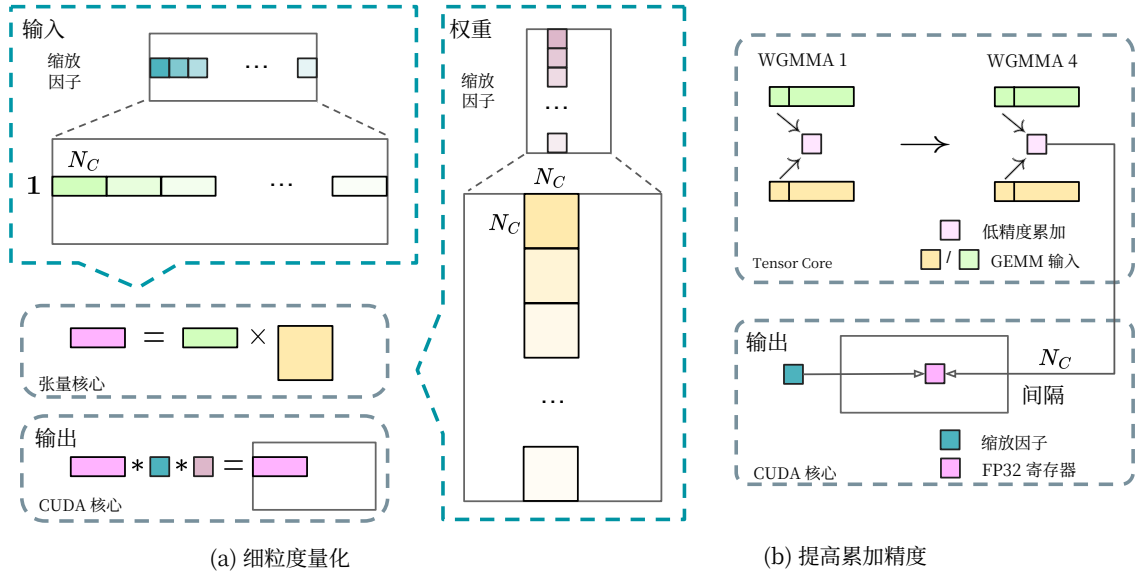


图 7 | (a) 我们提出一种细粒度量化方法，以减轻特征离群值导致的量化误差；为便于说明，仅展示 F_{prop} 。(b) 结合我们的量化策略，我们通过每隔 $N_C = 128$ 个元素 MMA 将高精度累加提升至 CUDA Cores，从而提高 FP8 GEMM 精度。

这些高精度组件会带来一定的内存开销，但在我们的分布式训练系统中，通过在多个 DP rank 之间进行高效分片，可以将其影响降至最低。

3.3.2. 通过量化与乘法提高精度

基于我们的混合精度 FP8 框架，我们引入了若干策略来增强低精度训练的准确性，重点关注量化方法和乘法过程。

Fine-Grained Quantization. 在低精度训练框架中，由于 FP8 格式的动态范围有限（受限于较少的指数位），上溢和下溢是常见挑战。标准做法是通过将输入张量的最大绝对值缩放到 FP8 的最大可表示值，使输入分布与 FP8 格式的可表示范围对齐（Narang et al., 2017）。这种方法使低精度训练对激活异常值高度敏感，而异常值会严重降低量化精度。为解决这一问题，我们提出了一种在更细粒度层级应用缩放的细粒度量化方法。如图 7 (a) 所示：（1）对于激活值，我们以 1×128 tile 为单位对元素进行分组和缩放（即每个 token 的每 128 个通道）；（2）对于权重，我们以 128×128 block 为单位对元素进行分组和缩放（即每 128 个输入通道、每 128 个输出通道）。这种方法通过根据更小的元素组调整缩放比例，使量化过程能够更好地适应异常值。在附录 B.2 中，我们进一步讨论了以与权重量化相同的 block 为单位对激活值进行分组和缩放时的训练不稳定性。

我们方法的一项关键修改是在 GEMM 操作的内维度上引入按组缩放因子。标准 FP8 GEMM 不直接支持此功能。然而，结合我们精确的 FP32 累加策略，它可以

得到高效实现。

值得注意的是，我们的细粒度量化策略与微缩放格式的理念高度一致（Rouhani et al., 2023b），而 NVIDIA 下一代 GPU（Blackwell 系列）的 Tensor Core 已宣布支持具有更小量化粒度的微缩放格式（NVIDIA, 2024a）。我们希望我们的设计能为未来工作提供参考，以跟上最新 GPU 架构的发展。

提高累加精度。低精度 GEMM 操作常遭受下溢问题，其精度很大程度上取决于高精度累加，而高精度累加通常以 FP32 精度执行（Kalamkar et al., 2019; Narang et al., 2017）。然而，我们观察到，NVIDIA H800 GPU 上 FP8 GEMM 的累加精度仅限于保留约 14 位，显著低于 FP32 累加精度。当内维度 K 较大时，这一问题会更加明显（Wortsman et al., 2023）；在大规模模型训练中，批量大小和模型宽度增大正是典型场景。以两个具有 $K = 4096$ 的随机矩阵的 GEMM 操作为例，在我们的初步测试中，Tensor Core 有限的累加精度导致最大相对误差接近 2%。尽管存在这些问题，有限的累加精度仍是少数 FP8 框架中的默认选项（NVIDIA, 2024b），严重限制了训练精度。

为解决这一问题，我们采用将结果提升到 CUDA Cores 以获得更高精度的策略（Thakkar et al., 2023）。该过程如图 7 (b) 所示。具体而言，在 Tensor Cores 上执行 MMA (Matrix Multiply-Accumulate) 期间，中间结果使用有限位宽进行累加。一旦达到 N_C 间隔，这些部分结果将被复制到 CUDA Cores 上的 FP32 寄存器中，并在那里执行全精度 FP32 累加。如前所述，我们的细粒度量化沿内维度 K 应用逐组缩放因子。在反量化过程中，这些缩放因子可以在 CUDA Cores 上高效相乘，仅产生极小的额外计算成本。

值得注意的是，这一修改降低了单个 warpgroup 的 WGMMA (Warpgroup-level Matrix Multiply-Accumulate) 指令发射率。然而，在 H800 架构上，通常可以同时持有两个 WGMMA：当一个 warpgroup 执行提升操作时，另一个可以执行 MMA 操作。这种设计实现了两种操作的重叠，从而保持 Tensor Cores 的高利用率。根据我们的实验，设置 $N_C = 128$ 个元素（相当于 4 个 WGMMA）代表了能够显著提升精度且不会引入大量开销的最小累加间隔。

指数与尾数。不同于先前工作（NVIDIA, 2024b; Peng et al., 2023b; Sun et al., 2019b）采用的混合 FP8 格式，该格式在 Fprop 中使用 E4M3（4 位指数和 3 位尾数），在 Dgrad 和 Wgrad 中使用 E5M2（5 位指数和 2 位尾数）；我们在所有张量上采用 E4M3 格式，以获得更高精度。我们将这种方法的可行性归因于细粒度量化策略，即按瓦片和按块缩放。通过对更小的元素组进行操作，我们的方法有效地使这些分组元素共享指数位，从而减轻动态范围受限的影响。

在线量化。在张量级量化框架中采用延迟量化（NVIDIA, 2024b; Peng et al., 2023b），该方法维护先前迭代中的最大绝对值历史记录，以推断当前值。

跨先前迭代的值以推断当前值。为确保精确的缩放因子并简化框架，我们在线计算每个 1×128 激活瓦片或每个 128×128 权重块的最大绝对值。基于此，我们推导缩放因子，然后将激活值或权重在线量化为 FP8 格式。

3.3.3. 低精度存储与通信

结合我们的 FP8 训练框架，我们通过将缓存激活值和优化器状态压缩为更低精度的格式，进一步降低内存消耗和通信开销。

低精度优化器状态。我们在 AdamW (Loshchilov and Hutter, 2017) 优化器中采用 BF16 数据格式而非 FP32 来跟踪一阶和二阶矩，且不会导致可观察的性能下降。然而，主权重（由优化器存储）和梯度（用于批量大小累积）仍以 FP32 保留，以确保整个训练过程中的数值稳定性。

低精度激活值。如图 6 所示，Wgrad 操作以 FP8 执行。为降低内存消耗，一个自然选择是在 Linear 算子的反向传播中以 FP8 格式缓存激活值。然而，对于若干算子，需要特别考虑以实现低成本的高精度训练：

(1) **注意力算子之后的 Linear 输入。**这些激活值也用于注意力算子的反向传播，因此对精度较为敏感。我们专门为这些激活值采用定制的 E5M6 数据格式。此外，在反向传播中，这些激活值将从 1×128 量化瓦片转换为 128×1 瓦片。为避免引入额外量化误差，所有缩放因子均采用取整缩放，即 2 的整数次幂。

MoE 中 SwiGLU 算子的输入 (2)为进一步降低内存成本，我们缓存 SwiGLU 算子的输入，并在反向传播期间重新计算其输出。我们还使用细粒度量化方法将这些激活以 FP8 存储，在内存效率与计算精度之间取得平衡。

低精度通信。通信带宽是 MoE 模型训练中的关键瓶颈。为缓解这一挑战，我们将 MoE 上投影前的激活量化为 FP8，然后应用 dispatch 组件，这与 MoE 上投影中的 FP8 Fprop 兼容。与注意力算子后的 Linear 输入类似，该激活的缩放因子为 2 的整数次幂。MoE 下投影前的激活梯度也采用类似策略。对于前向和反向 combine 组件，我们将其保留为 BF16，以保持训练流水线关键部分的训练精度。

3.4. 推理与部署

我们将 DeepSeek-V3 部署在 H800 集群上，其中每个节点内的 GPU 使用 NVLink 互连，集群中的所有 GPU 均通过 IB 完全互连。为同时确保在线服务的服务级目标 (SLO) 和高吞吐量，我们采用以下部署策略，将预填充和 解码 阶段分离。

3.4.1. 预填充

预填充阶段的最小部署单元由 4 个节点和 32 个 GPU 组成。`attention` 部分采用 4 路张量并行 (TP4) 与序列并行 (SP)，并结合 8 路数据并行 (DP8)。其较小的 TP 大小 4 限制了 TP 通信开销。对于 MoE 部分，我们使用 32 路专家并行 (EP32)，确保每个专家处理足够大的批次大小，从而提高计算效率。对于 MoE all-to-all 通信，我们采用与训练相同的方法：首先通过 IB 在节点间传输词元，然后通过 NVLink 在节点内 GPU 之间转发。特别地，我们对浅层中的稠密 MLP 使用 1 路张量并行，以节省 TP 通信。

为了在 MoE 部分的不同专家之间实现负载均衡，我们需要确保每个 GPU 处理大致相同数量的 token。为此，我们引入了冗余专家的部署策略，即复制高负载专家并进行冗余部署。根据在线部署期间收集的统计数据检测高负载专家，并定期调整（例如每 10 分钟一次）。确定冗余专家集合后，我们根据观测到的负载，在节点内 GPU 之间谨慎地重新排列专家，力求在不增加跨节点 all-to-all 通信开销的情况下，尽可能平衡各 GPU 的负载。对于 DeepSeek-V3 的部署，我们在预填充阶段设置 32 个冗余专家。除原本承载的 8 个专家外，每个 GPU 还将承载一个额外的冗余专家。

此外，在预填充阶段，为提高吞吐量并隐藏 all-to-all 和 TP 通信的开销，我们同时处理两个计算负载相近的微批次，将一个微批次的 `attention` 和 MoE 与另一个微批次的 `dispatch` 和 `combine` 重叠执行。

最后，我们正在探索一种面向专家的动态冗余策略，其中每个 GPU 承载更多专家（例如 16 个专家），但每个推理步骤仅激活 9 个专家。在每层的 all-to-all 操作开始前，我们会即时计算全局最优路由方案。鉴于预填充阶段涉及大量计算，计算该路由方案的开销几乎可以忽略不计。

3.4.2. 解码

在解码过程中，我们将共享专家视为路由专家。从这一角度看，每个 token 在路由时将选择 9 个专家，其中共享专家被视为高负载专家，始终会被选中。解码阶段的最小部署单元由 40 个节点和 320 个 GPU 组成。`attention` 部分采用带 SP 的 TP4，并结合 DP80；MoE 部分使用 EP320。对于 MoE 部分，每个 GPU 仅承载一个专家，64 个 GPU 负责承载冗余专家和共享专家。`dispatch` 和 `combine` 部分的 all-to-all 通信通过 IB 上的直接点对点传输完成，以实现低延迟。此外，我们利用 IBGDA (NVIDIA, 2022) 技术进一步降低延迟并提升通信效率。

与预填充类似，我们根据在线服务中的专家负载统计信息，以一定间隔周期性确定冗余专家集合。然而，由于每个 GPU 仅承载一个专家，我们无需重新排列专家。我们还在探索用于解码的动态冗余策略。不过，这需要对计算全局最优路由方案的算法以及与 `dispatch` kernel 的融合进行更谨慎的优化，以降低开销。

此外，为提高吞吐量并隐藏 all-to-all 通信的开销，我们还在探索解码阶段同时处理两个计算负载相似的微批次。不同于预填充，`attention` 在解码阶段占用更大比例的时间。因此，我们将一个微批次的 `attention` 与另一个微批次的 `dispatch+MoE+combine` 重叠。在解码阶段，每个专家的批次大小相对较小（通常在 256 tokens 以内），瓶颈在于内存访问而非计算。由于 MoE 部分只需加载一个专家的参数，内存访问开销很小，因此使用较少的 SM 不会显著影响整体性能。因此，为避免影响 `attention` 部分的计算速度，我们可以仅分配少量 SM 给 `dispatch+MoE+combine`。

3.5. 关于硬件设计的建议

基于我们对 all-to-all 通信和 FP8 训练方案的实现，我们向 AI 硬件厂商提出以下芯片设计建议。

3.5.1. 通信硬件

在 DeepSeek-V3 中，我们实现了计算与通信的重叠，以隐藏计算期间的通信延迟。与串行执行计算和通信相比，这显著降低了对通信带宽的依赖。然而，当前的通信实现依赖昂贵的 SM（例如，在 H800 GPU 可用的 132 个 SM 中，我们为此分配了 20 个），这将限制计算吞吐量。此外，使用 SM 进行通信会导致显著低效，因为张量核心完全未得到充分利用。

目前，SM 主要为 all-to-all 通信执行以下任务：

- 在 IB (InfiniBand) 域与 NVLink 域之间转发数据，同时从单个 GPU 汇聚发往同一节点内多个 GPU 的 IB 流量。
- 在 RDMA 缓冲区（已注册的 GPU 内存区域）与输入/输出缓冲区之间传输数据。
- 执行 `reduce` 操作 以用于 `all-to-all combine`。
- 在通过 IB 和 NVLink 域向多个专家分块传输数据期间管理细粒度内存布局。

我们希望未来的厂商开发硬件，将这些通信任务从宝贵的计算单元 SM 卸载出来，充当 GPU 协处理器或网络协处理器，如 NVIDIA SHARP Graham et al. (2016)。此外，为降低应用程序编程复杂性，我们希望该硬件从计算单元的角度统一 IB（横向扩展）和 NVLink（纵向扩展）网络。借助这一统一接口，计算单元只需基于简单原语提交通信请求，即可在整个 IB-NVLink 统一域中轻松完成 `read`、`write`、`multicast` 和 `reduce` 等操作。

3.5.2. 计算硬件

Tensor Core 中更高精度的 FP8 GEMM 累加在 NVIDIA Hopper 架构当前的 Tensor Core 实现中，FP8 GEMM 的累加精度受到限制。在根据最大指数右移、对齐 32 个尾数乘积后，Tensor Core 在加法中仅使用每个尾数乘积的最高 14 位，

并截断超出该范围的位。将加法结果累加到寄存器中同样采用 14 位精度。我们的实现通过在 CUDA core 中使用 FP32 精度，将 128 次 FP8×FP8 乘法的加法结果累加到寄存器中，从而部分缓解了这一限制。尽管这有助于实现成功的 FP8 训练，但由于 Hopper 架构在 FP8 GEMM 累加精度方面存在硬件缺陷，这仅是一种折中方案。未来的芯片需要采用更高精度。

Support for Tile- and Block-Wise Quantization. 当前 GPU 仅支持逐张量量化，缺乏对我们 tile 级和 block 级细粒度量化的原生支持。在当前实现中，当达到 N_C 间隔时，部分结果将从 Tensor Cores 复制到 CUDA cores，与缩放因子相乘后加到 CUDA cores 上的 FP32 寄存器中。尽管结合精确的 FP32 累加策略后，反量化开销已显著降低，但 Tensor Cores 与 CUDA cores 之间频繁的数据移动仍限制了计算效率。因此，我们建议未来的芯片通过使用 Tensor Cores 能够接收缩放因子并执行带组缩放的 MMA，来支持细粒度量化。这样，整个部分和累加及反量化过程都可以直接在 Tensor Cores 内完成，直至生成最终结果，从而避免频繁的数据移动。

Support for Online Quantization. 尽管在线量化已在我们的研究中证明有效，当前实现仍难以对其提供有效支持。在现有流程中，我们需要从 HBM（High Bandwidth Memory）读取 128 个 BF16 激活值（前一次计算的输出）进行量化，然后将量化后的 FP8 值写回 HBM，之后又要再次读取它们以执行 MMA。为解决这一低效问题，我们建议未来芯片将 FP8 转换和 TMA（Tensor Memory Accelerator）访问集成为单一融合操作，使量化能够在激活值从全局内存传输到共享内存期间完成，避免频繁的内存读写。我们还建议支持 warp 级转换指令以提高速度，从而进一步促进层归一化与 FP8 转换的更好融合。或者，可以采用近内存计算方法，将计算逻辑置于 HBM 附近。在这种情况下，BF16 元素从 HBM 读入 GPU 时即可直接转换为 FP8，使片外内存访问量减少约 50%。

支持转置 GEMM 操作。 当前架构使矩阵转置与 GEMM 操作的融合变得繁琐。在我们的工作流程中，前向传递期间的激活被量化为 1×128 FP8 瓦片并存储。反向传递期间，需要读出矩阵、反量化、转置、重新量化为 128×1 瓦片，并存储到 HBM 中。为减少内存操作，我们建议未来的芯片在 MMA 操作之前，支持从共享内存直接转置读取矩阵，以满足训练和推理中所需的精度。结合 FP8 格式转换与 TMA 访问的融合，这一改进将显著简化量化 workflow。

4. 预训练

4.1. 数据构建

与 DeepSeek-V2 相比，我们通过提高数学和编程样本的比例来优化预训练语料库，同时将多语言覆盖范围扩展到英语和中文之外。此外，我们的数据处理流程经过改进，在保持语料库多样性的同时最大限度地减少冗余。

受 Ding et al. (2024) 启发，我们采用文档打包方法来保证数据完整性，但训练期间不加入跨样本注意力掩码。最后，DeepSeek-V3 的训练语料库在我们的分词器中由 14.8T 个高质量且多样化的词元组成。

在 DeepSeekCoder-V2 (DeepSeek-AI, 2024a) 的训练过程中，我们观察到，Fill-in-Middle (FIM) 策略在使模型能够根据上下文线索准确预测中间文本的同时，并不会损害其下一个令牌预测能力。与 DeepSeekCoder-V2 一致，我们也在 DeepSeek-V3 的预训练中采用 FIM 策略。具体而言，我们使用 Prefix-Suffix-Middle (PSM) 框架将数据组织如下：

$$\langle |f_{im\_begin}| \rangle f_{pre} \langle |f_{im\_hole}| \rangle f_{suf} \langle |f_{im\_end}| \rangle f_{middle} \langle |eos\_token| \rangle.$$

该结构在文档级别应用，作为预打包过程的一部分。FIM 策略的应用比例为 0.1，与 PSM 框架一致。

DeepSeek-V3 的分词器采用字节级 BPE (Shibata et al., 1999)，词汇表扩展至 128K 个令牌。我们对分词器的预分词器和训练数据进行了修改，以优化多语言压缩效率。此外，与 DeepSeek-V2 相比，新的预分词器引入了结合标点符号和换行符的令牌。然而，当模型处理没有末尾换行符的多行提示时，尤其是在少样本评估提示中，这一技巧可能引入令牌边界偏差 (Lundberg, 2023)。为解决这一问题，我们在训练期间随机拆分一定比例的此类组合令牌，使模型接触更广泛的特殊情况并减轻这种偏差。

4.2. 超参数

ModelHyper-Parameters. 我们将 Transformer 层数设置为 61，隐藏维度设置为 7168。所有可学习参数均以标准差 0.006 随机初始化。在 MLA 中，我们将注意力头数量 n_h 设置为 128，将每头维度 d_h 设置为 128。KV 压缩维度 d_c 设置为 512，查询压缩维度 d_{qc} 设置为 1536。对于解耦查询和键，我们将每头维度 d^R_h 设置为 64。除了最初三层外，我们将所有 FFN 替换为 MoE 层。每个 MoE 层由 1 个共享专家和 256 个路由专家组成，其中每个专家的中间隐藏维度为 2048。在路由专家中，每个 token 将激活 8 个专家，并确保每个 token 最多发送到 4 个节点。多 token 预测深度 D 设置为 1，即除了精确的下一个 token 外，每个 token 还将预测一个额外 token。与 DeepSeek-V2 一样，DeepSeek-V3 也在压缩潜向量之后采用额外的 RMSNorm 层，并在宽度瓶颈处乘以额外的缩放因子。在此配置下，DeepSeek-V3 总计包含 671B 个参数，其中每个 token 激活 37B 个参数。

训练超参数我们采用 AdamW 优化器 (Loshchilov and Hutter, 2017)，超参数设置为 $\beta_1 = 0.9$ 、 $\beta_2 = 0.95$ ，权重_衰减为 $= 0.1$ 。预训练期间我们将最大序列长度设为 4K，并使用 14.8T tokens 对 DeepSeek-V3 进行预训练。关于学习率调度，我们首先在前 2K 步内将其从 0 线性增加到 2.2×10^{-4} 。随后，在模型消耗 10T 训练 tokens 前保持学习率为 2.2×10^{-4} 。之后，按照余弦衰减曲线，在 4.3T tokens 内逐渐将学习率衰减到 2.2×10^{-5} 。在最后 500B tokens 的训练期间，前 333B tokens 保持学习率为 2.2×10^{-5} ，并切换到另一个恒定学习率

在剩余 167B tokens 中采用 7.3×10^{-6} 。梯度裁剪范数设为 1.0。我们采用批次大小调度策略：在前 469B tokens 的训练中，批次大小从 3072 逐渐增加到 15360，随后在剩余训练中保持 15360。我们利用流水线并行，将模型的不同层部署到不同 GPU 上；对于每一层，路由专家将均匀部署在属于 8 个节点的 64 个 GPU 上。对于节点受限路由，每个 token 最多发送到 4 个节点（即 $M = 4$ ）。对于无辅助损失负载均衡，前 14.3T tokens 中我们将偏置更新速度 γ 设为 0.001，剩余 500B tokens 中设为 0.0。对于均衡损失，我们将 α 设为 0.0001，仅用于避免单个序列内部出现极端不均衡。前 10T tokens 中，MTP 损失权重 λ 设为 0.3，剩余 4.8T tokens 中设为 0.1。

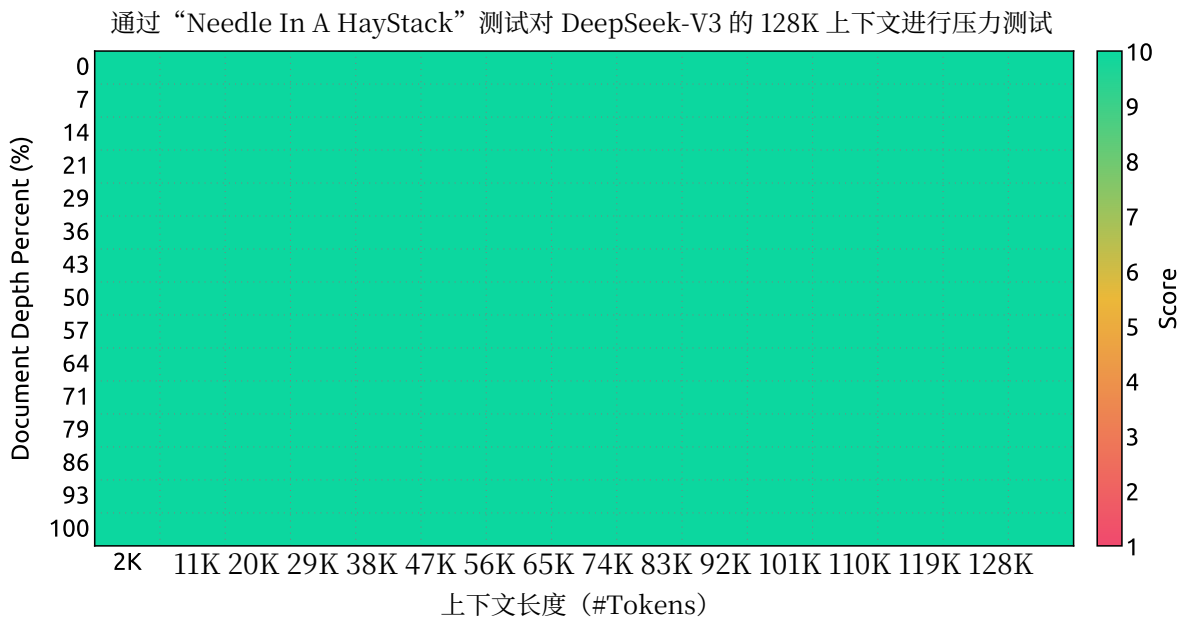


图 8 | “Needle In A Haystack” (NIAH) 测试的评估结果。DeepSeek-V3 在最长达 128K 的所有上下文窗口长度上均表现良好。

4.3. 长上下文扩展

我们采用与 DeepSeek-V2 (DeepSeek-AI, 2024c) 类似的方法，为 DeepSeek-V3 启用长上下文能力。在预训练阶段之后，我们应用 YaRN (Peng et al., 2023a) 进行上下文扩展，并执行两个额外训练阶段，每个阶段包含 1000 步，逐步将上下文窗口从 4K 扩展到 32K，再扩展到 128K。YaRN 配置与 DeepSeek-V2 中使用的配置一致，仅应用于解耦共享键 $\mathbf{k}_t^R$ 。两个阶段的超参数保持一致，缩放值为 $s = 40$ 、 $\alpha = 1$ 、 $\beta = 32$ ，缩放因子为 $\sqrt{t} = 0.1 \ln s + 1$ 。第一阶段中，序列长度设为 32K，批量大小为 1920。第二阶段中，序列长度增加到 128K，批量大小减少到 480。两个阶段的学习率均设为 7.3×10^{-6} ，与预训练阶段的最终学习率一致。

通过这两阶段的扩展训练，DeepSeek-V3 能够处理长度达 128K 的输入，同时保持强劲性能。图 8 展示了 DeepSeek-V3 在监督微调后于“干草堆中找针” (NIAH) 测试中取得的显著性能，证明其在最长达 128K 的上下文窗口长度范围内保持一致的稳健性。

4.4. 评估

4.4.1. 评估基准

DeepSeek-V3 的基础模型在以英语和中文为主的多语言语料库上进行预训练，因此我们主要在英语和中文的一系列基准以及一个多语言基准上评估其性能。我们的评估基于集成于 HAI-LLM 框架中的内部评估框架。所考虑的基准分类并列出如下，其中带下划线的基准为中文基准，带双下划线的基准为多语言基准：

多学科多项选择 数据集包括 MMLU (Hendrycks et al., 2020)、MMLU-Redux (Gema et al., 2024)、MMLU-Pro (Wang et al., 2024b)、MMMLU (OpenAI, 2024b)、C-Eval (Huang et al., 2023) 和 CMMLU (Li et al., 2023)。

语言理解与推理 数据集包括 HellaSwag (Zellers et al., 2019)、PIQA (Bisk et al., 2020)、ARC (Clark et al., 2018) 和 BigBench Hard (BBH) (Suzgun et al., 2022)。

闭卷问答 数据集包括 TriviaQA (Joshi et al., 2017) 和 NaturalQuestions (Kwiatkowski et al., 2019)。

阅读理解 数据集包括 RACE (Lai et al., 2017)、DROP (Dua et al., 2019)、C3 (Sun et al., 2019a) 和 CMRC (Cui et al., 2019)。

指代消歧 数据集包括 CLUEWSC (Xu et al., 2020) 和 WinoGrande (Sakaguchi et al., 2019)。

语言建模 数据集包括 Pile (Gao et al., 2020)。

中文理解与文化 数据集包括 CCPM (Li et al., 2021)。

数学 数据集包括 GSM8K (Cobbe et al., 2021)、MATH (Hendrycks et al., 2021)、MGSM (Shi et al., 2023) 和 CMath (Wei et al., 2023)。

代码 数据集包括 HumanEval (Chen et al., 2021)、LiveCodeBench-Base (0801-1101) (Jain et al., 2024)、MBPP (Austin et al., 2021) 和 CRUXEval (Gu et al., 2024)。

标准化考试 包括 AGIEval (Zhong et al., 2023)。请注意，AGIEval 同时包括英文和中文子集。

遵循我们之前的工作 (DeepSeek-AI, 2024b,c)，我们对 HellaSwag、PIQA、WinoGrande、RACE-Middle、RACE-High、MMLU、MMLU-Redux、MMLU-Pro、MMMLU、ARC-Easy、ARC-Challenge、C-Eval、CMMLU、C3 和 CCPM 等数据集采用基于困惑度的评估，并对 TriviaQA、NaturalQuestions、DROP、MATH、GSM8K、MGSM、HumanEval、MBPP、LiveCodeBench-Base、CRUXEval、BBH、AGIEval、CLUEWSC、CMRC 和 CMath 采用基于生成的评估。此外，我们对 Pile-test 执行基于语言建模的评估，并使用每字节比特数 (BPB) 作为指标，以确保使用不同分词器的模型之间进行公平比较。

4.4.2. 评估结果

在表3中，我们将 DeepSeek-V3 的基础模型与当前最先进的开源基础模型进行比较，包括 DeepSeek-V2-Base (DeepSeek-AI, 2024c) (我们之前发布的模型)、Qwen2.5 72B Base (Qwen, 2024b) 和 LLaMA-3.1 405B Base (AI@Meta, 2024b)。我们使用内部评估框架评估所有这些模型，并确保它们采用相同的评估设置。需要注意的是，由于过去几个月我们对评估框架进行了更改，性能

基准 (指标)	# 样本数	DeepSeek-V2 Base	Qwen2.5 LLaMA-3.1		DeepSeek-V3 Base
			72B 基础版	405B 基础模型	
架构	-	MoE	稠密	稠密	MoE
# 激活参数量	-	21B	72B	405B	37B
# 总参数量	-	236B	72B	405B	671B
英语	Pile-test (BPB)	-	0.606	0.638	0.542
	BBH (EM)	3-shot	78.8	79.8	82.9
	MMLU (EM)	5-shot	78.4	85.0	84.4
	MMLU-Redux (EM)	5-shot	75.6	83.2	81.3
	MMLU-Pro (EM)	5-shot	51.4	58.3	52.8
	DROP (F1)	3-shot	80.4	80.6	86.0
	ARC-Easy (EM)	25-shot	97.6	98.4	98.4
	ARC-Challenge (EM)	25-shot	92.2	94.5	95.3
	HellaSwag (EM)	10-shot	87.1	84.8	89.2
	PIQA (EM)	0-shot	83.9	82.6	85.9
	WinoGrande (EM)	5-shot	86.3	82.3	85.2
	RACE-中等 (EM)	5-shot	73.1	68.1	74.2
	RACE-高 (EM)	5-shot	52.6	50.3	56.8
	TriviaQA (EM)	5-shot	80.0	71.9	82.7
	自然问题 (EM)	5-shot	38.6	33.2	41.5
	AGIEval (EM)	0-shot	57.5	75.8	60.6
Code	HumanEval (Pass@1)	0-shot	43.3	53.0	54.9
	MBPP (Pass@1)	3-shot	65.0	72.6	68.4
	LiveCodeBench-Base (Pass@1)	3-shot	11.6	12.9	15.5
	CRUXEval-I (EM)	2-shot	52.5	59.1	58.5
	CRUXEval-O (EM)	2-shot	49.8	59.9	59.9
Math	GSM8K (EM)	8-shot	81.6	88.3	83.5
	MATH (EM)	4-shot	43.4	54.4	49.0
	MGSM (EM)	8-shot	63.6	76.2	69.9
	CMath (EM)	3-shot	78.7	84.5	77.3
中文	CLUEWSC (EM)	5-shot	82.0	82.5	83.0
	C-Eval (EM)	5-shot	81.4	89.2	72.5
	CMMLU (EM)	5-shot	84.0	89.5	73.7
	CMRC (EM)	1次示例	77.4	75.8	76.0
	C3 (EM)	0-shot	77.4	76.7	79.7
	CCPM (EM)	0-shot	93.0	88.5	78.6
多语言	MMMLU-非英语 (EM)	5-shot	64.0	74.8	73.8

表 3 | DeepSeek-V3-Base 与其他具有代表性的开源基础模型之间的比较。所有模型均在我们的内部框架中进行评估，并采用相同的评估设置。差距不超过 0.3 的分数被视为处于同一水平。DeepSeek-V3-Base 在大多数基准测试中取得了最佳性能，尤其是在数学和代码任务上。

DeepSeek-V2-Base 的性能与我们此前报告的结果略有不同。总体而言，DeepSeek-V3-Base 全面优于 DeepSeek-V2-Base 和 Qwen2.5 72B Base，并在大多数基准测试中超过 LLaMA-3.1 405B Base，基本上成为最强的开源模型。

从更细致的角度来看，我们分别将 DeepSeek-V3-Base 与其他开源基础模型进行比较。

(1) 与 DeepSeek-V2-Base 相比，由于模型架构的改进、模型规模和训练 token 数量的扩大以及数据质量的提升，DeepSeek-V3-Base 如预期般取得了显著更好的性能。

(2) 与 Qwen2.5 72B Base（当前最先进的中文开源模型）相比，在激活参数仅为其一半的情况下，DeepSeek-V3-Base 也展现出显著优势，

尤其是在英语、多语言、代码和数学基准测试上。至于中文基准测试，除 CMMLU（一项中文多学科选择题任务）外，DeepSeek-V3-Base 的表现也优于 Qwen2.5 72B。（3）与 LLaMA-3.1 405B Base（激活参数是其 11 倍的最大开源模型）相比，DeepSeek-V3-Base 在多语言、代码和数学基准测试上也表现出显著更好的性能。至于英语和中文语言基准测试，DeepSeek-V3-Base 展现出具有竞争力或更优的性能，尤其在 BBH、MMLU 系列、DROP、C-Eval、CMMLU 和 CCPM 上表现突出。

得益于高效的架构和全面的工程优化，DeepSeek-V3 实现了极高的训练效率。在我们的训练框架和基础设施下，每训练 DeepSeek-V3 一万亿个 token 仅需 180K H800 GPU 小时，这比训练 72B 或 405B 稠密模型便宜得多。

基准测试（指标）	# 样本数	小型MoE 基线	小型MoE 含 MTP	大型 MoE 基线	大型 MoE 含 MTP
# 激活参数 (Inference)	-	2.4B	2.4B	20.9B	20.9B
# 总参数 (Inference)	-	15.7B	15.7B	228.7B	228.7B
# 训练 Token 数	-	1.33T	1.33T	540B	540B
Pile-test (BPB)	-	0.729	0.729	0.658	0.657
BBH (EM)	3-shot	39.0	41.4	70.0	70.7
MMLU (EM)	5-shot	50.0	53.3	67.5	66.6
DROP (F1)	1次示例	39.2	41.3	68.5	70.6
TriviaQA (EM)	5-shot	56.9	57.7	67.0	67.3
NaturalQuestions (EM)	5-shot	22.7	22.3	27.2	28.5
HumanEval (Pass@1)	0-shot	20.7	26.8	44.5	53.7
MBPP (Pass@1)	3-shot	35.8	36.8	61.6	62.2
GSM8K (EM)	8-shot	25.4	31.4	72.3	74.0
MATH (EM)	4-shot	10.7	12.6	38.6	39.8

表 4 | MTP 策略的消融实验结果。MTP 策略在大多数评估基准上始终能够提升模型性能。

4.5. 讨论

4.5.1. 多 token 预测的消融研究

在表 4 中，我们展示了 MTP 策略的消融结果。具体而言，我们在不同规模的两个基线模型上验证了 MTP 策略。在小规模下，我们在 1.33T 个 token 上训练了一个总参数量为 15.7B 的基线 MoE 模型。在大规模下，我们在 540B 个 token 上训练了一个总参数量为 228.7B 的基线 MoE 模型。在此基础上，在保持训练数据和其他架构不变的情况下，我们分别附加一个深度为 1 的 MTP 模块，并使用 MTP 策略训练两个模型进行比较。需要注意的是，在推理期间我们会直接丢弃 MTP 模块，因此比较模型的推理成本完全相同。从表中可以观察到，MTP 策略在大多数评估基准上都能持续提升模型性能。

4.5.2. 无辅助损失均衡策略的消融研究

在表 5 中，我们展示了无辅助损失均衡策略的消融结果。我们在不同规模的两个基线模型上验证了该策略。在小规模设置下，我们训练了一个总参数量为 15.7B、使用 1.33T tokens 的基线 MoE 模型。在大规模设置下，我们训练了一个总参数量为 228.7B、使用 578B tokens 的基线 MoE 模型。

基准测试（指标）	# 样本数	小型MoE	小型MoE	大型 MoE	大型 MoE
		基于辅助损失	无辅助损失	基于辅助损失	无辅助损失
# 激活参数量	-	2.4B	2.4B	20.9B	20.9B
# 总参数量	-	15.7B	15.7B	228.7B	228.7B
# 训练 Token 数	-	1.33T	1.33T	578B	578B
Pile-test (BPB)	-	0.727	0.724	0.656	0.652
BBH (EM)	3-shot	37.3	39.3	66.7	67.9
MMLU (EM)	5-shot	51.0	51.8	68.3	67.2
DROP (F1)	1次示例	38.1	39.0	67.1	67.1
TriviaQA (EM)	5-shot	58.3	58.5	66.7	67.7
NaturalQuestions (EM)	5-shot	23.2	23.4	27.1	28.1
HumanEval (Pass@1)	0-shot	22.0	22.6	40.2	46.3
MBPP (Pass@1)	3-shot	36.6	35.8	59.2	61.2
GSM8K (EM)	8-shot	27.1	29.6	70.7	74.5
MATH (EM)	4-shot	10.9	11.1	37.2	39.6

表 5 | 无辅助损失平衡策略的消融结果。与纯粹基于辅助损失的方法相比，无辅助损失策略在大多数评估基准上始终取得更好的模型性能。

两个基线模型都完全使用辅助损失来促进负载均衡，并采用带 top-K 亲和度归一化的 sigmoid 门控函数。它们用于控制辅助损失强度的超参数分别与 DeepSeek-V2-Lite 和 DeepSeek-V2 相同。在这两个基线模型之上，保持训练数据和其他架构不变，我们移除所有辅助损失，并引入无辅助损失均衡策略进行比较。从表中可以观察到，无辅助损失策略在大多数评估基准上始终取得更好的模型性能。

4.5.3. 按批次的负载均衡 VS. 按序列的负载均衡

无辅助损失均衡与按序列辅助损失之间的关键区别在于均衡范围：按批次与按序列。与按序列辅助损失相比，按批次均衡施加了更灵活的约束，因为它不会强制每个序列在域内保持均衡。这种灵活性使专家能够更好地专门化于不同领域。为验证这一点，我们记录并分析了 Pile 测试集不同领域中一个 16B 基于辅助损失的基线模型和一个 16B 无辅助损失模型的专家负载。如图 9 所示，我们观察到，无辅助损失模型表现出更明显的专家专门化模式，符合预期。

为进一步研究这种灵活性与模型性能优势之间的相关性，我们还设计并验证了一种按批次辅助损失，鼓励在每个训练批次而非每个序列上实现负载均衡。实验结果表明，在达到相似按批次负载均衡水平时，按批次辅助损失也能取得与无辅助损失方法相近的模型性能。具体而言，在我们的 1B MoE 模型实验中，验证损失分别为：2.258（使用按序列辅助损失）、2.253（使用无辅助损失方法）和 2.253（使用按批次辅助损失）。在 3B MoE 模型上我们也观察到类似结果：使用按序列辅助损失的模型验证损失为 2.085，而使用无辅助损失方法或按批次辅助损失的模型验证损失均为 2.080。

此外，尽管按批次进行的负载均衡方法表现出持续的性能优势，但在效率方面也面临两个潜在挑战：（1）

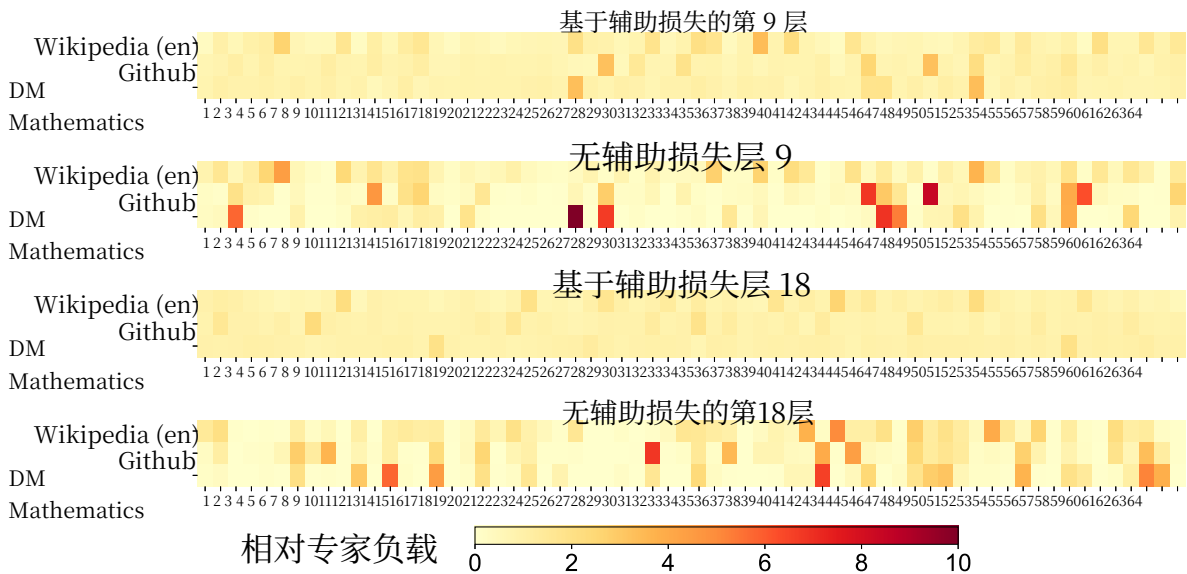


图 9 | Pile测试集三个领域中无辅助损失模型和基于辅助损失模型的专家负载。与基于辅助损失的模型相比，无辅助损失模型呈现出更明显的专家专业化模式。相对专家负载表示实际专家负载与理论平衡专家负载之间的比率。受篇幅限制，我们仅以两层的结果为例，所有层的结果见附录C。

某些序列或小批次内部的负载不均衡，以及（2）推理期间由领域偏移引起的负载不均衡。第一个挑战可由我们的训练框架自然解决，该框架采用大规模专家并行和数据并行，保证每个微批次具有较大的规模。针对第二个挑战，我们还设计并实现了一个带冗余专家部署的高效推理框架，如第 3.4 节所述，以克服这一问题。

5. 后训练

5.1. 监督微调

我们整理指令微调数据集，使其包含跨多个领域的150万条实例，每个领域采用针对其具体需求定制的不同数据创建方法。

推理数据。对于数学、代码竞赛题和逻辑谜题等推理相关数据集，我们利用内部 DeepSeek-R1模型生成数据。具体而言，尽管R1生成的数据展现出很高的准确性，但存在过度思考、格式不佳和篇幅过长等问题。我们的目标是在R1生成的推理数据的高准确性与常规格式推理数据的清晰性和简洁性之间取得平衡。

为建立我们的方法，我们首先使用结合监督微调（SFT）和强化学习（RL）的训练流水线，开发一个针对特定领域（如代码、数学或通用推理）的专家模型。该专家模型为最终模型充当数据生成器。训练过程中，对于每个实例生成两种不同类型的 SFT 样本：第一种将问题与其原始回答组合为 <问题、原始回答> 的格式，第二种则加入系统提示

并将其与问题和 R1 回答组合为 <系统提示、问题、R1 回答> 的格式。

系统提示经过精心设计，其中包含引导模型生成具有反思和验证机制的响应的指令。在 RL 阶段，即使没有显式系统提示，模型也利用高温采样生成融合 R1 生成数据与原始数据模式的响应。经过数百个 RL 步骤后，中间 RL 模型学会融入 R1 模式，从而有策略地提升整体性能。

完成 RL 训练阶段后，我们实施拒绝采样，为最终模型筛选高质量 SFT 数据，其中专家模型用作数据生成源。该方法确保最终训练数据保留 DeepSeek-R1 的优势，同时生成简洁有效的响应。

非推理数据。对于创意写作、角色扮演和简单问答等非推理数据，我们使用 DeepSeek-V2.5 生成响应，并邀请人工标注员验证数据的准确性和正确性。

SFT 设置。我们使用 SFT 数据集对 DeepSeek-V3-Base 进行两个 epoch 的微调，采用余弦衰减学习率调度，使学习率从 5×10^{-6} 开始并逐渐降低至 1×10^{-6} 。训练期间，每个单独序列由多个样本打包而成。然而，我们采用样本掩码策略，确保这些示例彼此隔离且互不可见。

5.2. 强化学习

5.2.1. 奖励模型

We 在我们的 RL 过程中采用基于规则的奖励模型（RM）和基于模型的 RM。

基于规则的 RM。对于可以使用特定规则验证的问题，我们采用基于规则的奖励系统来确定反馈。例如，某些数学问题具有确定性结果，我们要求模型以指定格式（例如，在方框中）提供最终答案，从而能够应用规则验证其正确性。同样，对于 LeetCode 问题，我们可以利用编译器根据测试用例生成反馈。通过尽可能利用基于规则的验证，我们确保更高的可靠性，因为这种方法能够抵抗操纵或利用。

基于模型的 RM对于具有自由形式标准答案的问题，我们依赖奖励模型判断回答是否匹配预期标准答案。相反，对于没有确定标准答案的问题（例如涉及创意写作的问题），奖励模型的任务是以问题及相应答案为输入提供反馈。奖励模型由 DeepSeek-V3 SFT 检查点训练得到。为提高其可靠性，我们构建了偏好数据，不仅提供最终奖励，还包括导向该奖励的思维链。这种方法有助于降低特定任务中的奖励黑客风险。

5.2.2. 组相对策略优化

与 DeepSeek-V2 (DeepSeek-AI, 2024c) 类似, 我们采用群组相对策略优化 (GRPO) (Shao 等, 2024)。该方法省去了通常与策略模型规模相同的评论家模型, 而是从群组得分估计基线。具体而言, 对于每个问题 q , GRPO 从旧策略模型 $\pi_{\theta_{old}}$ 中采样一组输出 $\{o_1, o_2, \dots, o_G\}$, 然后通过最大化以下目标来优化策略模型 π_θ :

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^G \sim \pi_{\theta_{old}}(O|q)] \frac{1}{G} \sum_{i=1}^G \left(\min \left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)} A_i, \text{clip} \left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}, 1 - \epsilon, 1 + \epsilon \right) A_i \right) - \beta \mathbb{D}_{KL}(\pi_\theta || \pi_{ref}) \right), \quad (26)$$

$$\mathbb{D}_{KL}(\pi_\theta || \pi_{ref}) = \frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - \log \frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - 1, \quad (27)$$

其中 ϵ 和 β 为超参数; π_{ref} 为参考模型; A_i 为优势, 由奖励 $\{r_1, r_2, \dots, r_G\}$ 推导而来, 这些奖励对应于每个组内的输出:

$$A_i = \frac{r_i - \text{mean}(\{r_1, r_2, \dots, r_G\})}{\text{std}(\{r_1, r_2, \dots, r_G\})}. \quad (28)$$

在 RL 过程中, 我们融入了来自编码、数学、写作、角色扮演和问答等多种领域的提示词。这种方法不仅使模型更贴近人类偏好, 还提升了其在基准测试中的表现, 尤其是在可用 SFT 数据有限的场景下。

5.3. 评估

5.3.1. 评估设置

Evaluation Benchmarks. 除了用于基础模型测试的基准外, 我们还在 IFEval (Zhou et al., 2023)、FRAMES (Krishna et al., 2024)、LongBench v2 (Bai et al., 2024)、GPQA (Rein et al., 2023)、SimpleQA (OpenAI, 2024c)、C- SimpleQA (He et al., 2024)、SWE-Bench Verified (OpenAI, 2024d)、Aider 1、LiveCodeBench (Jain et al., 2024) (问题来自 2024 年 8 月至 2024 年 11 月)、Codeforces 2、中国全国高中数学奥林匹克竞赛 (CNMO 2024)³, 和美国数学邀请赛 2024 (AIME 2024) (MAA, 2024) 上进一步评估了指令模型。

对比基线。我们将聊天模型与多个强大的基线模型进行了全面评估, 包括 DeepSeek-V2-0506、DeepSeek-V2.5-0905、Qwen2.5 72B Instruct、LLaMA-3.1 405B Instruct、Claude-Sonnet-3.5-1022 和 GPT-4o-0513。对于 DeepSeek-V2 模型系列, 我们选择最具代表性的变体进行比较。对于闭源模型, 评估通过各自的 API 进行。

详细评估配置对于包括 MMLU、DROP、GPQA 和 SimpleQA 在内的标准基准测试, 我们采用 simple-evals 框架⁴中的评估提示。

¹<https://aider.chat>

²<https://codeforces.com>

³<https://www.cms.org.cn/Home/comp/comp/cid/12.html>

⁴<https://github.com/openai/simple-evals>

对于 MMLU-Redux 的零样本设置，我们采用 Zero-Eval 提示格式（Lin, 2024）。对于其他数据集，我们遵循其原始评估协议，使用数据集创建者提供的默认提示。对于代码和数学基准，HumanEval-Mul 数据集总共包含 8 种主流编程语言（Python、Java、Cpp、C#、JavaScript、TypeScript、PHP 和 Bash）。我们使用 CoT 和非 CoT 方法评估模型在 LiveCodeBench 上的性能，其数据收集时间为 2024 年 8 月至 2024 年 11 月。Codeforces 数据集使用参赛者百分比进行测量。SWE-Bench verified 使用 agentless 框架（Xia 等, 2024）进行评估。我们使用“diff”格式评估与 Aider 相关的基准。对于数学评估，AIME 和 CNMO 2024 使用 0.7 的温度进行评估，结果取 16 次运行的平均值，而 MATH-500 采用贪心解码。我们允许所有模型在每个基准测试中最多输出 8192 个 tokens。

基准（指标）		DeepSeek V2-0506	DeepSeek V2.5-0905	Qwen2.5 72B-Inst.	LLaMA-3.1 405B-Inst.	Claude-3.5-Sonnet-1022	GPT-4o 0513	DeepSeek V3
架构		MoE	MoE	稠密	稠密	-	-	MoE
# 激活参数量		21B	21B	72B	405B	-	-	37B
# 总参数量		236B	236B	72B	405B	-	-	671B
英语	MMLU (EM)	78.2	80.6	85.3	88.6	88.3	87.2	88.5
	MMLU-Redux (EM)	77.9	80.3	85.6	86.2	88.9	88.0	89.1
	MMLU-Pro (EM)	58.5	66.2	71.6	73.3	78.0	72.6	75.9
	DROP (3-shot F1)	83.0	87.8	76.7	88.7	88.3	83.7	91.6
	IF-Eval (Prompt Strict)	57.7	80.6	84.1	86.0	86.5	84.3	86.1
	GPQA-Diamond (Pass@1)	35.3	41.3	49.0	51.1	65.0	49.9	59.1
	SimpleQA (正确率)	9.0	10.2	9.1	17.1	28.4	38.2	24.9
	FRAMES (准确率)	66.9	65.4	69.8	70.0	72.5	80.5	73.3
	LongBench v2 (Acc.)	31.6	35.4	39.4	36.1	41.0	48.1	48.7
Code	HumanEval-Mul (Pass@1)	69.3	77.4	77.3	77.2	81.7	80.5	82.6
	LiveCodeBench (Pass@1-COT)	18.8	29.2	31.1	28.4	36.3	33.4	40.5
	LiveCodeBench (Pass@1)	20.3	28.4	28.7	30.1	32.8	34.2	37.6
	Codeforces (百分位数)	17.5	35.6	24.8	25.3	20.3	23.6	51.6
	SWEVerified (已解决)	-	22.6	23.8	24.5	50.8	38.8	42.0
	Aider-Edit (准确率)	60.3	71.6	65.4	63.9	84.2	72.9	79.7
	Aider-Polyglot (准确率)	-	18.2	7.6	5.8	45.3	16.0	49.6
Math	AIME 2024 (Pass@1)	4.6	16.7	23.3	23.3	16.0	9.3	39.2
	MATH-500 (EM)	56.3	74.7	80.0	73.8	78.3	74.6	90.2
	CNMO 2024 (Pass@1)	2.8	10.8	15.9	6.8	13.1	10.8	43.2
中文	CLUEWSC (EM)	89.9	90.4	91.4	84.7	85.4	87.9	90.9
	C-Eval (EM)	78.6	79.5	86.1	61.5	76.7	76.0	86.5
	C-SimpleQA (正确)	48.5	54.1	48.4	50.4	51.3	59.3	64.8

表 6 | DeepSeek-V3 与其他代表性聊天模型的比较。所有模型均在将输出长度限制为 8K 的配置下进行评估。对于样本数少于 1000 的基准测试，使用不同的温度设置进行多次测试，以获得稳健的最终结果。DeepSeek-V3 是表现最佳的开源模型，同时在与前沿闭源模型的比较中也展现出具有竞争力的性能。

5.3.2. 标准评估

表 6 给出了评估结果，显示 DeepSeek-V3 是性能最佳的开源模型。此外，它在与 GPT-4o 和 Claude-3.5-Sonnet 等前沿闭源模型的比较中也具有竞争力。

英语基准MMLU 是广受认可的基准，用于评估大语言模型在多样知识领域和任务上的性能。DeepSeek-V3 展现出竞争力，与 LLaMA-3.1-405B、GPT-4o 和 Claude-Sonnet 3.5 等顶级模型不相上下，同时显著优于 Qwen2.5 72B。此外，DeepSeek-V3 在更具挑战性的教育知识基准 MMLU-Pro 上表现出色，其性能仅略逊于 Claude-Sonnet 3.5。在 MMLU-Redux（修正标签后的 MMLU 改进版本）上，DeepSeek-V3 超过了同类模型。此外，在博士水平评估测试集 GPQA-Diamond 上，DeepSeek-V3 取得了显著成绩，仅次于 Claude 3.5 Sonnet，并以较大幅度超过所有其他竞争者。

在 DROP、LongBench v2 和 FRAMES 等长上下文理解基准上，DeepSeek-V3 继续展现出顶级模型的地位。在 DROP 的 3-shot 设置下，它取得了令人印象深刻的 91.6 F1 分数，超过了该类别中的所有其他模型。在 FRAMES 上，该基准要求在 100k token 上下文中进行问答，DeepSeek-V3 仅略逊于 GPT-4o，同时以显著优势超过所有其他模型。这展示了 DeepSeek-V3 处理超长上下文任务的强大能力。DeepSeek-V3 的长上下文能力还通过其在 LongBench v2 上同类最佳的表现得到进一步验证；该数据集在 DeepSeek V3 发布前仅数周才发布。在事实知识基准 SimpleQA 上，DeepSeek-V3 落后于 GPT-4o 和 Claude-Sonnet，主要原因在于其设计重点和资源分配。DeepSeek-V3 分配更多训练 token 来学习中文知识，因此在 C-SimpleQA 上表现卓越。在指令遵循基准上，DeepSeek-V3 显著超越其前代 DeepSeek-V2-series，凸显了其理解并遵守用户定义格式约束能力的提升。

代码和数学基准。 编码对于 LLM 是一项具有挑战性且实用的任务，涵盖 SWE-Bench-Verified 和 Aider 等面向工程的任务，以及 HumanEval 和 LiveCodeBench 等算法任务。在工程任务中，DeepSeek-V3 落后于 Claude-Sonnet-3.5-1022，但显著优于开源模型。开源的 DeepSeek-V3 有望推动与编码相关的工程任务发展。通过提供其强大能力，DeepSeek-V3 可以促进软件工程和算法开发等领域的创新与改进，赋能开发者和研究人员突破开源模型在编码任务中的能力边界。在算法任务中，DeepSeek-V3 展现出卓越性能，在 HumanEval-Mul 和 LiveCodeBench 等基准上超越所有基线。这一成功可归因于其先进的知识蒸馏技术，该技术有效增强了其面向算法任务的代码生成和问题解决能力。

在数学基准测试中，DeepSeek-V3 展现出卓越性能，显著超越基线，并为非 o1 类模型树立了新的 state-of-the-art。具体而言，在 AIME、MATH-500 和 CNMO 2024 上，DeepSeek-V3 在绝对得分上以约 10% 的优势超越第二名模型 Qwen2.5 72B；对于如此具有挑战性的基准测试而言，这是一个相当大的差距。这一显著能力凸显了从 DeepSeek-R1 采用蒸馏技术的有效性，事实证明该技术对非 o1 类模型极为有益。

中文基准Qwen 和 DeepSeek 是两个代表性模型系列，对中文和英文都有强大的支持。在事实性基准 Chinese SimpleQA 上，DeepSeek-V3 超过 Qwen2.5-72B 16.4 分，尽管 Qwen2.5 在包含 18T tokens 的更大语料库上训练，比 DeepSeek-V3 使用的 14.8T tokens 多 20%，而 DeepSeek-V3 是

模型	Arena-Hard	AlpacaEval 2.0
DeepSeek-V2.5-0905	76.2	50.5
Qwen2.5-72B-Instruct	81.2	49.1
LLaMA-3.1 405B	69.3	40.5
GPT-4o-0513	80.4	51.1
Claude-Sonnet-3.5-1022	85.2	52.0
DeepSeek-V3	85.5	70.0

表 7 | 英文开放式对话评估。对于AlpacaEval 2.0，我们使用长度控制胜率作为指标。

预训练于。

在 C-Eval（中文教育知识评估的代表性基准）和 CLUEWSC（中文 Winograd Schema Challenge）上，DeepSeek-V3 与 Qwen2.5-72B 展现出相近的性能水平，表明两个模型都针对具有挑战性的中文推理和教育任务进行了良好优化。

5.3.3. 开放式评估

除标准基准外，我们还使用 LLM 作为评判者，在开放式生成任务上评估模型，结果如表 7 所示。具体而言，我们遵循 AlpacaEval 2.0 (Dubois et al., 2024) 和 Arena-Hard (Li et al., 2024a) 的原始配置，使用 GPT-4-Turbo-1106 作为评判者进行成对比较。在 Arena-Hard 上，DeepSeek-V3 相对于基线 GPT-4-0314 取得了超过 86% 的惊人胜率，表现与 Claude-Sonnet-3.5-1022 等顶级模型相当。这凸显了 DeepSeek-V3 的强大能力，尤其是在处理包括编码和调试任务在内的复杂提示词方面。此外，DeepSeek-V3 还创造了一个里程碑，成为首个在 Arena-Hard 基准上超过 85% 的开源模型。这一成就显著弥合了开源模型与闭源模型之间的性能差距，为开源模型在挑战性领域能够实现的水平树立了新标准。

同样，DeepSeek-V3在AlpacaEval 2.0上展现出卓越性能，击败了闭源和开源模型。这表明它在写作任务和处理直接问答场景方面具有出色能力。值得注意的是，它以20%的显著幅度超越DeepSeek-V2.5-0905，凸显了在应对简单任务方面的重大改进，也展示了其进步的有效性。

5.3.4. 作为生成式奖励模型的DeepSeek-V3

我们将 DeepSeek-V3 的判断能力与 GPT-4o 和 Claude-3.5 等最先进模型进行比较。表8展示了这些模型在 RewardBench (Lambert 等, 2024) 上的性能。DeepSeek-V3 的性能与 GPT-4o-0806 和 Claude-3.5-Sonnet-1022 的最佳版本相当，同时超过其他版本。此外，通过投票技术还可以增强 DeepSeek-V3 的判断能力。因此，我们将 DeepSeek-V3 与投票结合，用于对开放式问题提供自我反馈，从而提升对齐过程的有效性和稳健性。

模型	聊天	聊天-困难集	安全性	推理	平均值
GPT-4o-0513	96.6	70.4	86.7	84.9	84.7
GPT-4o-0806	96.1	76.1	88.1	86.6	86.7
GPT-4o-1120	95.8	71.3	86.2	85.2	84.6
Claude-3.5-sonnet-0620	96.4	74.0	81.6	84.7	84.2
Claude-3.5-sonnet-1022	96.4	79.7	91.1	87.6	88.7
DeepSeek-V3	96.9	79.8	87.0	84.3	87.0
DeepSeek-V3 (maj@6)	96.9	82.6	89.5	89.2	89.6

Table 8 | GPT-4o、Claude-3.5-sonnet 和 DeepSeek-V3 在 RewardBenc 上的性能 h.

模型	LiveCodeBench-CoT		MATH-500	
	Pass@1	长度	Pass@1	长度
DeepSeek-V2.5 基线	31.1	718	74.6	769
DeepSeek-V2.5 +R1 蒸馏	37.4	783	83.2	1510

表 9 | DeepSeek-R1 蒸馏的贡献。Live-CodeBench 和 MATH-500 的评估设置与表 6 相同。

5.4. 讨论

5.4.1. 来自 *DeepSeek-R1* 的蒸馏

我们基于 DeepSeek-V2.5 消融了 DeepSeek-R1 蒸馏的贡献。基线模型使用短 CoT 数据训练，而其竞争模型使用上述专家检查点生成的数据。

表 9 展示了蒸馏数据的有效性，在 LiveCodeBench 和 MATH-500 基准上均取得显著改进。我们的实验揭示了一个有趣的权衡：蒸馏带来更好的性能，但也会大幅增加平均响应长度。为在模型准确性和计算效率之间保持平衡，我们为 DeepSeek-V3 的蒸馏仔细选择了最优设置。

我们的研究表明，从推理模型进行知识蒸馏是后训练优化的一个有前景方向。虽然当前工作聚焦于从数学和代码领域蒸馏数据，但该方法在各种任务领域具有更广泛的应用潜力。这些特定领域所展现的有效性表明，长 CoT 蒸馏可能有助于提升模型在其他需要复杂推理的认知任务中的表现。进一步探索该方法在不同领域中的应用，仍是未来研究的重要方向。

5.4.2. 自我奖励

奖励在 RL 中发挥着关键作用，引导优化过程。在可以通过外部工具轻松进行验证的领域（例如某些编码或数学场景），RL 展现出卓越的效果。然而，在更一般的场景中，通过硬编码构建反馈机制并不实际。在 DeepSeek-V3 的开发过程中，对于这些更广泛的场景，我们采用宪法式 AI 方法（Bai 等，2022），利用 DeepSeek-V3 自身的投票评估结果作为反馈来源。这种方法

产生了显著的对齐效果，大幅提升了 DeepSeek-V3 在主观评估中的性能。通过整合额外的宪法输入，DeepSeek-V3 可以朝宪法所规定的方向进行优化。我们认为，这种将补充信息与 LLM 结合、并以其作为反馈来源的范式至关重要。LLM 充当多功能处理器，能够将来自不同场景的非结构化信息转化为奖励，最终促进 LLM 的自我改进。除了自奖励之外，我们还致力于发现其他通用且可扩展的奖励方法，以持续提升模型在一般场景中的能力。

5.4.3. 多令牌预测评估

DeepSeek-V3 不仅预测下一个单独的令牌，而是通过 MTP 技术预测接下来的 2 个令牌。结合推测解码框架 (Leviathan et al., 2023; Xia et al., 2023)，它可以显著加快模型的解码速度。一个自然的问题是，额外预测令牌的接受率如何。根据我们的评估，在各种生成主题中，第二个令牌预测的接受率介于 85% 和 90% 之间，表现出一致的可靠性。这一高接受率使 DeepSeek-V3 能够显著提升解码速度，达到 1.8 倍 TPS（每秒令牌数）。

6. 结论、局限性与未来方向

本文介绍了 DeepSeek-V3，这是一种大型 MoE 语言模型，总参数量为 671B，激活参数量为 37B，在 14.8T 个令牌上训练而成。除 MLA 和 DeepSeekMoE 架构外，它还率先采用无辅助损失的负载均衡策略，并设定多令牌预测训练目标以获得更强性能。得益于 FP8 训练和精细的工程优化，DeepSeek-V3 的训练具有较高成本效益。后训练阶段还成功从 DeepSeek-R1 系列模型中蒸馏出推理能力。全面评估表明，DeepSeek-V3 已成为目前可用的最强开源模型，并在性能上可媲美 GPT-4o 和 Claude-3.5-Sonnet 等领先闭源模型。尽管性能强劲，它仍保持经济的训练成本。其完整训练（包括预训练、上下文长度扩展和后训练）仅需 2.788M 个 H800 GPU 小时。

在承认 DeepSeek-V3 具备出色性能和高性价比的同时，我们也认识到它存在一些局限，尤其是在部署方面。首先，为确保高效推理，DeepSeek-V3 推荐的部署单元相对较大，这可能给小型团队带来负担。其次，尽管我们的 DeepSeek-V3 部署策略已实现超过 DeepSeek-V2 两倍的端到端生成速度，但仍有进一步提升的空间。幸运的是，随着更先进硬件的发展，这些局限预计将自然得到解决。

DeepSeek 始终坚持具有长期主义精神的开源模型路线，旨在稳步接近 AGI（通用人工智能）的终极目标。未来，我们计划在以下方向进行战略性研究投入。

- 我们将持续研究和改进模型架构，旨在进一步提升训练和推理效率，努力实现对无限上下文长度的高效支持。此外，我们将尝试突破 Transformer 的架构限制，进一步拓展其建模能力的边界。

- 我们将持续迭代训练数据的数量和质量，并探索纳入更多训练信号来源，旨在推动数据规模化覆盖更全面的维度。
- 我们将持续探索并迭代模型的深度思考能力，旨在通过扩展推理长度和深度来增强其智能与问题解决能力。
- 我们将探索更加全面且多维度的模型评估方法，以防止研究过程中倾向于优化固定的一组基准测试；这种倾向可能会对模型能力造成误导性印象，并影响我们的基础评估。

参考文献

AI@Meta. Llama 3 模型卡, 2024a. URL https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md. AI@Meta. Llama 3.1 模型卡, 2024b. URL https://github.com/meta-llama/llama-models/blob/main/models/llama3_1/MODEL_CARD.md. Anthropic. Claude 3.5 sonnet, 2024. URL <https://www.anthropic.com/news/claude-3-5-sonnet>. J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry, Q. Le 等。《使用大语言模型进行程序合成》。arXiv 预印本 arXiv:2108.07732, 2021. Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirhoseini, C. McKinnon 等。《宪法式 AI: 来自 AI 反馈的无害性》。arXiv 预印本 arXiv:2212.08073, 2022. Y. Bai, S. Tu, J. Zhang, H. Peng, X. Wang, X. Lv, S. Cao, J. Xu, L. Hou, Y. Dong, J. Tang 和 J. Li. 《LongBench v2: 迈向对现实长上下文多任务的更深入理解与推理》。arXiv 预印本 arXiv:2412.15204, 2024. M. Bauer, S. Treichler 和 A. Aiken. 《Singe: 利用 warp 专门化实现 GPU 高性能》。载于第19届 ACM SIGPLAN 并行编程原理与实践研讨会论文集, PPOPP'14, 第119–130页, 美国纽约, 2014. Association for Computing Machinery. ISBN 9781450326568. doi: 10.1145/2555243.2555258. URL <https://doi.org/10.1145/2555243.2555258>. Y. Bisk, R. Zellers, R. L. Bras, J. Gao 和 Y. Choi. 《PIQA: 自然语言中的物理常识推理》。载于第三十四届 AAAI 人工智能会议, AAAI 2020, 第三十二届 IAAI, 及第十届 EAAI, 美国纽约, 2020年2月7–12日, 第 7432–7439页. AAAI Press, 2020. doi: 10.1609/aaai.v34i05.6239. URL <https://doi.org/10.1609/aaai.v34i05.6239>. M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse.

A. N. Carr、J. Leike、J. Achiam、V. Misra、E. Morikawa、A. Radford、M. Knight、M. Brundage、M. Murati、K. Mayer、P. Welinder、B. McGrew、D. Amodei、S. McCandlish、I. Sutskever 和 W. Zaremba。《评估在代码上训练的大语言模型》。CoRR, abs/2107.03374, 2021。URL <https://arxiv.org/abs/2107.03374>。

P. Clark、I. Cowhey、O. Etzioni、T. Khot、A. Sabharwal、C. Schoenick 和 O. Tafjord。《你认为自己解决了问答问题吗？尝试 ARC: AI2 推理挑战》。CoRR, abs/1803.05457, 2018。URL <http://arxiv.org/abs/1803.05457>。

K. Cobbe、V. Kosaraju、M. Bavarian、M. Chen、H. Jun、L. Kaiser、M. Plappert、J. Tworek、J. Hilton、R. Nakano 等。《训练验证器解决数学文字题》。arXiv 预印本 arXiv:2110.14168, 2021。

Y. Cui、T. Liu、W. Che、L. Xiao、Z. Chen、W. Ma、S. Wang 和 G. Hu。《用于中文机器阅读理解的跨度抽取数据集》。载于 K. Inui、J. Jiang、V. Ng 和 X. Wan (编), 《2019年自然语言处理经验方法会议暨第九届国际自然语言处理联合会议论文集 (EMNLP-IJCNLP)》, 第5883–5889页, 中国香港, 2019年11月。Association for Computational Linguistics。doi: 10.18653/v1/D19-1600。URL <https://aclanthology.org/D19-1600>。

D. Dai、C. Deng、C. Zhao、R. X. Xu、H. Gao、D. Chen、J. Li、W. Zeng、X. Yu、Y. Wu、Z. Xie、Y. K. Li、P. Huang、F. Luo、C. Ruan、Z. Sui 和 W. Liang。《Deepseekmoe: 迈向混合专家语言模型中的终极专家专门化》。CoRR, abs/2401.06066, 2024。URL <https://doi.org/10.48550/arXiv.2401.06066>。

DeepSeek-AI。《Deepseek-coder-v2: 打破代码智能中闭源模型的壁垒》。CoRR, abs/2406.11931, 2024a。URL <https://doi.org/10.48550/arXiv.2406.11931>。

DeepSeek-AI。《Deepseek LLM: 以长期主义扩展开源语言模型》。CoRR, abs/2401.02954, 2024b。URL <https://doi.org/10.48550/arXiv.2401.02954>。

DeepSeek-AI。《Deepseek-v2: 强大、经济且高效的混合专家语言模型》。CoRR, abs/2405.04434, 2024c。URL <https://doi.org/10.48550/arXiv.2405.04434>。

T. Dettmers、M. Lewis、Y. Belkada 和 L. Zettlemoyer。《Gpt3.int8(): 面向大规模 Transformer 的 8 位矩阵乘法》。神经信息处理系统进展, 35: 30318–30332, 2022。

H. Ding、Z. Wang、G. Paolini、V. Kumar、A. Deoras、D. Roth 和 S. Soatto。《更少截断可改善语言建模》。arXiv 预印本 arXiv:2404.10830, 2024。

D. Dua、Y. Wang、P. Dasigi、G. Stanovsky、S. Singh 和 M. Gardner。《DROP: 需要对段落进行离散推理的阅读理解基准》。载于《2019年北美计算语言学协会分会人类语言技术会议论文集》, NAACL-HLT 2019, 美国明尼阿波利斯, 2019年6月2–7日, 第1卷, 第2368–2378页。Association for Computational Linguistics, 2019。doi: 10.18653/v1/N19-1246。URL <https://doi.org/10.18653/v1/n19-1246>。

Y. Dubois、B. Galambosi、P. Liang 和 T. B. Hashimoto。《长度控制的 alpaca-eval: 一种消除自动评估器偏差的简单方法》。arXiv 预印本 arXiv:2404.04475, 2024。

W. Fedus、B. Zoph 和 N. Shazeer。《Switch Transformer: 使用简单高效的稀疏性扩展至万亿参数模型》。CoRR, abs/2101.03961, 2021。URL <https://arxiv.org/abs/2101.03961>。M. Fishman、B. Chmiel、R. Banner 和 D. Soudry。《将 FP8 训练扩展至万亿 token 的 LLM》。arXiv 预印本 arXiv:2409.12517, 2024。E. Frantar、S. Ashkboos、T. Hoefler 和 D. Alistarh。《GPTQ: 生成式预训练 Transformer 的高精度训练后量化》。arXiv 预印本 arXiv:2210.17323, 2022。L. Gao、S. Biderman、S. Black、L. Golding、T. Hoppe、C. Foster、J. Phang、H. He、A. Thite、N. Nabeshima 等。《The Pile: 用于语言建模的 800GB 多样文本数据集》。arXiv 预印本 arXiv:2101.00027, 2020。A. P. Gema、J. O. J. Leang、G. Hong、A. Devoto、A. C. M. Mancino、R. Saxena、X. He、Y. Zhao、X. Du、M. R. G. Madani、C. Barale、R. McHardy、J. Harris、J. Kaddour、E. van Krieken 和 P. Minervini。《我们已经完成 MMLU 了吗?》。CoRR, abs/2406.04127, 2024。URL <https://doi.org/10.48550/arXiv.2406.04127>。F. Gloeckle、B. Y. Idrissi、B. Rozière、D. Lopez-Paz 和 G. Synnaeve。《通过多 token 预测实现更好更快的大语言模型》。载于第四十一届国际机器学习会议, ICML 2024, 奥地利维也纳, 2024年7月21-27日。OpenReview.net, 2024。URL <https://openreview.net/forum?id=pEWAcejiU2>。Google。《我们的下一代模型: Gemini 1.5》, 2024。URL <https://blog.google/technology/ai/google-gemini-next-generation-model-february-2024>。R. L. Graham、D. Bureddy、P. Lui、H. Rosenstock、G. Shainer、G. Bloch、D. Goldenberg、M. Dubman、S. Kotchubievsky、V. Koushnir 等。《可扩展分层聚合协议 (SHArP): 用于高效数据归约的硬件架构》。载于2016年 HPC 通信优化第一届国际研讨会 (COMHPC), 第 1-10页。IEEE, 2016。A. Gu、B. Rozière、H. Leather、A. Solar-Lezama、G. Synnaeve 和 S. I. Wang。《Cruxeval: 代码推理、理解与执行基准》, 2024。D. Guo、Q. Zhu、D. Yang、Z. Xie、K. Dong、W. Zhang、G. Chen、X. Bi、Y. Wu、Y. K. Li、F. Luo、Y. Xiong 和 W. Liang。《Deepseek-coder: 当大语言模型遇上编程——代码智能的崛起》。CoRR, abs/2401.14196, 2024。URL <https://doi.org/10.48550/arXiv.2401.14196>。A. Harlap、D. Narayanan、A. Phanishayee、V. Seshadri、N. Devanur、G. Ganger 和 P. Gibbons。《Pipedream: 快速高效的流水线并行 DNN 训练》, 2018。URL <https://arxiv.org/abs/1806.03377>。B. He、L. Noci、D. Paliotta、I. Schlag 和 T. Hofmann。《理解并最小化 Transformer 训练中的离群特征》。载于第三十八届神经信息处理系统年会。Y. He、S. Li、J. Liu、Y. Tan、W. Wang、H. Huang、X. Bu、H. Guo、C. Hu、B. Zheng 等。《Chinese SimpleQA: 面向大语言模型的中文事实性评估》。arXiv 预印本 arXiv:2411.07140, 2024。D. Hendrycks、C. Burns、S. Basart、A. Zou、M. Mazeika、D. Song 和 J. Steinhardt。《衡量大规模多任务语言理解》。arXiv 预印本 arXiv:2009.03300, 2020。

D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song 和 J. Steinhardt. 《使用 math 数据集衡量数学问题求解》。arXiv 预印本 arXiv:2103.03874, 2021. Y. Huang, Y. Bai, Z. Zhu, J. Zhang, J. Zhang, T. Su, J. Liu, C. Lv, Y. Zhang, J. Lei 等。《C-Eval: 面向基础模型的多层次多学科中文评估套件》。arXiv 预印本 arXiv:2305.08322, 2023. N. Jain, K. Han, A. Gu, W. Li, F. Yan, T. Zhang, S. Wang, A. Solar-Lezama, K. Sen 和 I. Stoica. 《LiveCodeBench: 面向代码的大语言模型整体且无污染评估》。CoRR, abs/2403.07974, 2024. URL <https://doi.org/10.48550/arXiv.2403.07974>. A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier 等。《Mistral 7b》。arXiv 预印本 arXiv:2310.06825, 2023. M. Joshi, E. Choi, D. Weld 和 L. Zettlemoyer. 《TriviaQA: 用于阅读理解的大规模远程监督挑战数据集》。载于《第55届计算语言学协会年会论文集》(第1卷: 长篇论文), 第1601–1611页, 加拿大温哥华, 2017年7月。Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL <https://aclanthology.org/P17-1147>. D. Kalamkar, D. Mudigere, N. Mellempudi, D. Das, K. Banerjee, S. Avancha, D. T. Vooturi, N. Jammalamadaka, J. Huang, H. Yuen 等。《bfloat16 在深度学习训练中的研究》。arXiv 预印本 arXiv:1905.12322, 2019. S. Krishna, K. Krishna, A. Mohananey, S. Schwarcz, A. Stambler, S. Upadhyay 和 M. Faruqui. 《事实、检索与推理: 检索增强生成的统一评估》。CoRR, abs/2409.12941, 2024. doi: 10.48550/ARXIV.2409.12941. URL <https://doi.org/10.48550/arXiv.2409.12941>. T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. P. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, K. Toutanova, L. Jones, M. Kelcey, M. Chang, A. M. Dai, J. Uszkoreit, Q. Le 和 S. Petrov. 《自然问题: 问答研究基准》。Trans. Assoc. Comput. Linguistics, 7: 452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://doi.org/10.1162/tacl_a_00276. G. Lai, Q. Xie, H. Liu, Y. Yang 和 E. H. Hovy. 《RACE: 来自考试的大规模阅读理解数据集》。载于《2017年自然语言处理经验方法会议论文集》, EMNLP 2017, 丹麦哥本哈根, 2017年9月9–11日, 第785–794页。Association for Computational Linguistics, 2017. doi: 10.18653/V1/D17-1082. URL <https://doi.org/10.18653/v1/d17-1082>. N. Lambert, V. Pyatkin, J. Morrison, L. Miranda, B. Y. Lin, K. Chandu, N. Dziri, S. Kumar, T. Zick, Y. Choi 等。《RewardBench: 评估语言建模的奖励模型》。arXiv 预印本 arXiv:2403.13787, 2024. D. Lepikhin, H. Lee, Y. Xu, D. Chen, O. Firat, Y. Huang, M. Krikun, N. Shazeer 和 Z. Chen. 《GShard: 通过条件计算和自动分片扩展巨型模型》。载于第九届学习表征国际会议, ICLR 2021. OpenReview.net, 2021. URL <https://openreview.net/forum?id=qrwe7XHTmYb>.

Y. Leviathan, M. Kalman 和 Y. Matias。《通过推测解码实现 Transformer 的快速推理》。载于国际机器学习会议, ICML 2023, 2023年7月23-29日, 美国夏威夷檀香山, 第202卷《机器学习研究论文集》, 第19274-19286页。PMLR, 2023。URL <https://proceedings.mlr.press/v202/leviathan23a.html>。H. Li, Y. Zhang, F. Koto, Y. Yang, H. Zhao, Y. Gong, N. Duan 和 T. Baldwin。《CMMLU: 衡量中文大规模多任务语言理解》。arXiv 预印本 arXiv:2306.09212, 2023。S. Li 和 T. Hoefler。《Chimera: 利用双向流水线高效训练大规模神经网络》。载于国际高性能计算、网络、存储与分析会议论文集, SC '21, 第1-14页, 2021年11月。doi: 10.1145/3458817.3476145。URL <http://dx.doi.org/10.1145/3458817.3476145>。T. Li, W.-L. Chiang, E. Frick, L. Dunlap, T. Wu, B. Zhu, J. E. Gonzalez 和 I. Stoica。《从众包数据到高质量基准: Arena-hard 和 benchbuilder 流水线》。arXiv 预印本 arXiv:2406.11939, 2024a。W. Li, F. Qi, M. Sun, X. Yi 和 J. Zhang。《CCPM: 中文古典诗词匹配数据集》, 2021。Y. Li, F. Wei, C. Zhang 和 H. Zhang。《EAGLE: 推测采样需要重新思考特征不确定性》。载于第四十一届国际机器学习会议, ICML 2024, 奥地利维也纳, 2024年7月21-27日。——OpenReview.net, 2024b。URL <https://openreview.net/forum?id=1NdN7eXyb4>。B. Y. Lin。《ZeroEval: 语言模型评估统一框架》, 2024年7月。URL <https://github.com/WildEval/ZeroEval>。I. Loshchilov 和 F. Hutter。《解耦权重衰减正则化》。arXiv 预印本 arXiv:1711.05101, 2017。S. Lundberg。《提示设计的艺术: 提示边界与 token 修复》, 2023。URL <https://towardsdatascience.com/the-art-of-prompt-design-prompt-boundaries-and-token-healing-3b2448b0be38>。Y. Luo, Z. Zhang, R. Wu, H. Liu, Y. Jin, K. Zheng, M. Wang, Z. He, G. Hu, L. Chen 等。《用于深度学习的 Ascend HiFloat8 格式》。arXiv 预印本 arXiv:2409.16626, 2024。MAA。《美国邀请数学考试 (AIME) 》。载于《美国邀请数学考试——AIME 2024》, 2024年2月。URL <https://maa.org/math-competitions/american-invitational-mathematics-examination-aime>。P. Micikevicius, D. Stosic, N. Burgess, M. Cornea, P. Dubey, R. Grisenthwaite, S. Ha, A. Heinecke, P. Judd, J. Kamalu 等。《用于深度学习的 FP8 格式》。arXiv 预印本 arXiv:2209.05433, 2022。Mistral。《更便宜、更好、更快、更强: 继续推动 AI 前沿并让所有人都能使用》, 2024。URL <https://mistral.ai/news/mixtral-8x22b>。S. Narang, G. Diamos, E. Elsen, P. Micikevicius, J. Alben, D. Garcia, B. Ginsburg, M. Hous-ton, O. Kuchaiev, G. Venkatesh 等。《混合精度训练》。载于学习表征国际会议, 2017。

B. Noune、P. Jones、D. Justus、D. Masters 和 C. Lusch. 《深度神经网络的 8 位数值格式》。arXiv 预印本 [arXiv:2206.02915](https://arxiv.org/abs/2206.02915), 2022。NVIDIA. 《使用 NVIDIA Magnum IO NVSH-MEM 和 GPUDirect Async 提升 HPC 系统网络性能》。
<https://developer.nvidia.com/blog/improving-net-work-performance-of-hpc-systems-using-nvidia-magnum-io-nvshmem-and-gpudirect-async>, 2022。NVIDIA. 《Blackwell 架构》。
<https://www.nvidia.com/en-us/data-center/technologies/blackwell-architecture/>, 2024a。NVIDIA. 《TransformerEngine》, 2024b。URL <https://github.com/NVIDIA/TransformerEngine>。访问日期: 2024-11-19。OpenAI. 《Hello GPT-4o》, 2024a。URL <https://openai.com/index/hello-gpt-4o/>。OpenAI. 《多语言大规模多任务语言理解 (mmmlu) 》, 2024b。URL <https://huggingface.co/datasets/openai/MMMLU>。OpenAI. 《介绍 SimpleQA》, 2024c。URL <https://openai.com/index/introducing-simpleqa/>。OpenAI. 《介绍 SWE-bench verified: 我们正在发布 SWE-bench 的人工验证子集, 该子集更……》, 2024d。URL <https://openai.com/index/introducing-swe-bench-verified/>。

B. Peng、J. Quesnelle、H. Fan 和 E. Shippole. 《Yarn: 高效扩展大语言模型上下文窗口》。arXiv 预印本 [arXiv:2309.00071](https://arxiv.org/abs/2309.00071), 2023a。H. Peng、K. Wu、Y. Wei、G. Zhao、Y. Yang、Z. Liu、Y. Xiong、Z. Yang、B. Ni、J. Hu 等。《FP8-LM: 训练 FP8 大语言模型》。arXiv 预印本 [arXiv:2310.18313](https://arxiv.org/abs/2310.18313), 2023b。P. Qi、X. Wan、G. Huang 和 M. Lin. 《零气泡流水线并行》。arXiv 预印本 [arXiv:2401.10241](https://arxiv.org/abs/2401.10241), 2023a。P. Qi、X. Wan、G. Huang 和 M. Lin. 《零气泡流水线并行》, 2023b。URL <https://arxiv.org/abs/2401.10241>。

Qwen. 《Qwen 技术报告》。arXiv 预印本 [arXiv:2309.16609](https://arxiv.org/abs/2309.16609), 2023。Qwen. 《介绍 Qwen1.5》, 2024a。URL <https://qwenlm.github.io/blog/qwen1.5>。Qwen. 《Qwen2.5: 基础模型家族》, 2024b。URL <https://qwenlm.github.io/blog/qwen2.5>。

S. Rajbhandari、J. Rasley、O. Ruwase 和 Y. He. 《ZeRO: 面向万亿参数模型训练的内存优化》。载于 SC20: 国际高性能计算、网络、存储与分析会议, 第1-16页。IEEE, 2020。

D. Rein、B. L. Hou、A. C. Stickland、J. Petty、R. Y. Pang、J. Dirani、J. Michael 和 S. R. Bowman. 《GPQA: 研究生水平的 Google-proof 问答基准》。arXiv 预印本 [arXiv:2311.12022](https://arxiv.org/abs/2311.12022), 2023。B. D. Rouhani、R. Zhao、A. More、M. Hall、A. Khodamoradi、S. Deng、D. Choudhary、M. Cornea、E. Dellinger、K. Denolf 等。《用于深度学习的微缩放数据格式》。arXiv 预印本 [arXiv:2310.10537](https://arxiv.org/abs/2310.10537), 2023a。

B. D. Rouhani、R. Zhao、A. More、M. Hall、A. Khodamoradi、S. Deng、D. Choudhary、M. Cornea、E. Dellinger、K. Denolf 等。《用于深度学习的微缩放数据格式》。arXiv 预印本 arXiv:2310.10537, 2023b. K. Sakaguchi、R. L. Bras、C. Bhagavatula 和 Y. Choi。《Winogrande: 大规模对抗性 Winograd Schema 挑战》, 2019. Z. Shao、P. Wang、Q. Zhu、R. Xu、J. Song、M. Zhang、Y. Li、Y. Wu 和 D. Guo。《Deepseekmath: 推动开放语言模型数学推理的极限》。arXiv 预印本 arXiv:2402.03300, 2024. N. Shazeer、A. Mirhoseini、K. Maziarz、A. Davis、Q. V. Le、G. E. Hinton 和 J. Dean。《极其庞大的神经网络: 稀疏门控混合专家层》。载于第五届学习表征国际会议, ICLR 2017. OpenReview.net, 2017. URL <https://openreview.net/forum?id=B1ckMDqlg>. F. Shi、M. Suzgun、M. Freitag、X. Wang、S. Srivats、S. Vosoughi、H. W. Chung、Y. Tay、S. Ruder、D. Zhou、D. Das 和 J. Wei。《语言模型是多语言思维链推理器》。载于第十一届学习表征国际会议, ICLR 2023, 卢旺达基加利, 2023年5月1-5日。OpenReview.net, 2023. URL <https://openreview.net/forum?id=fR3wGCk-IXp>. Y. Shibata、T. Kida、S. Fukamachi、M. Takeda、A. Shinohara、T. Shinohara 和 S. Arikawa。《字节对编码: 加速模式匹配的文本压缩方案》, 1999. J. Su、M. Ahmed、Y. Lu、S. Pan、W. Bo 和 Y. Liu。《Roformer: 带旋转位置嵌入的增强型 Transformer》。Neurocomputing, 568: 127063, 2024. K. Sun、D. Yu、D. Yu 和 C. Cardie。《研究具有挑战性的中文机器阅读理解中的先验知识》, 2019a. M. Sun、X. Chen、J. Z. Kolter 和 Z. Liu。《大语言模型中的大规模激活》。arXiv 预印本 arXiv:2402.17762, 2024. X. Sun、J. Choi、C.-Y. Chen、N. Wang、S. Venkataramani、V. V. Srinivasan、X. Cui、W. Zhang 和 K. Gopalakrishnan。《用于深度神经网络的混合 8 位浮点 (HFP8) 训练与推理》。神经信息处理系统进展, 32, 2019b. M. Suzgun、N. Scales、N. Schärli、S. Gehrmann、Y. Tay、H. W. Chung、A. Chowdhery、Q. V. Le、E. H. Chi、D. Zhou 等。《具有挑战性的 BIG-bench 任务以及思维链能否解决它们》。arXiv 预印本 arXiv:2210.09261, 2022. V. Thakkar、P. Ramani、C. Cecka、A. Shivam、H. Lu、E. Yan、J. Kosaian、M. Hoemmen、H. Wu、A. Kerr、M. Nicely、D. Merrill、D. Blasig、F. Qiao、P. Majcher、P. Springer、M. Hohnerbach、J. Wang 和 M. Gupta。《CUTLASS》, 2023 年1月。URL <https://github.com/NVIDIA/cutlass>. H. Touvron、T. Lavril、G. Izacard、X. Martinet、M.-A. Lachaux、T. Lacroix、B. Rozière、N. Goyal、E. Hambro、F. Azhar 等。《LLaMA: 开放且高效的基础语言模型》。arXiv 预印本 arXiv:2302.13971, 2023a. H. Touvron、L. Martin、K. Stone、P. Albert、A. Almahairi、Y. Babaei、N. Bashlykov、S. Batra、P. Bhargava、S. Bhosale、D. Bikel、L. Blecher、C. Canton-Ferrer、M. Chen、G. Cucurull、D. Esiobu、J. Fernandes、J. Fu、W. Fu、B. Fuller、C. Gao、V. Goswami、N. Goyal、A. Hartshorn、S. Hosseini。

R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov 和 T. Scialom。《Llama 2: 开放基础模型和微调聊天模型》。CoRR, abs/2307.09288, 2023b。doi: 10.48550/arXiv.2307.09288。URL <https://doi.org/10.48550/arXiv.2307.09288>。A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser 和 I. Polosukhin。《Attention is all you need》。神经信息处理系统进展, 30, 2017。L. Wang, H. Gao, C. Zhao, X. Sun 和 D. Dai。《混合专家的无辅助损失负载均衡策略》。CoRR, abs/2408.15664, 2024a。URL <https://doi.org/10.48550/arXiv.2408.15664>。Y. Wang, X. Ma, G. Zhang, Y. Ni, A. Chandra, S. Guo, W. Ren, A. Arulraj, X. He, Z. Jiang, T. Li, M. Ku, K. Wang, A. Zhuang, R. Fan, X. Yue 和 W. Chen。《MMLU-Pro: 更稳健、更具挑战性的多任务语言理解基准》。CoRR, abs/2406.01574, 2024b。URL <https://doi.org/10.48550/arXiv.2406.01574>。T. Wei, J. Luan, W. Liu, S. Dong 和 B. Wang。《CMath: 你的语言模型能通过中国小学数学测试吗?》, 2023。M. Wortsman, T. Dettmers, L. Zettlemoyer, A. Morcos, A. Farhadi 和 L. Schmidt。《大规模视觉语言模型的稳定低精度训练》。神经信息处理系统进展, 36: 10271–10298, 2023。H. Xi, C. Li, J. Chen 和 J. Zhu。《使用 4 位整数训练 Transformer》。神经信息处理系统进展, 36: 49146–49168, 2023。C. S. Xia, Y. Deng, S. Dunn 和 L. Zhang。《Agentless: 揭示基于 LLM 的软件工程智能体》。arXiv 预印本, 2024。H. Xia, T. Ge, P. Wang, S. Chen, F. Wei 和 Z. Sui。《推测解码: 利用推测执行加速 seq2seq 生成》。载于计算语言学协会 EMNLP 2023 论文集, 2023 年 12 月 6–10 日, 新加坡, 第 3909–3925 页。Association for Computational Linguistics, 2023。URL <https://doi.org/10.18653/v1/2023.findings-emnlp.257>。G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth 和 S. Han。《SmoothQuant: 大语言模型准确高效的训练后量化》。载于国际机器学习会议, 第 38087–38099 页。PMLR, 2023。L. Xu, H. Hu, X. Zhang, L. Li, C. Cao, Y. Li, Y. Xu, K. Sun, D. Yu, C. Yu, Y. Tian, Q. Dong, W. Liu, B. Shi, Y. Cui, J. Li, J. Zeng, R. Wang, W. Xie, Y. Li, Y. Patterson, Z. Tian, Y. Zhang, H. Zhou, S. Liu, Z. Zhao, Q. Zhao, C. Yue, X. Zhang, Z. Yang, K. Richardson 和 Z. Lan。《CLUE: 中文语言理解评估基准》。载于 D. Scott, N. Bel 和 C. Zong (编), 《第 28 届国际计算语言学会议论文集》, COLING 2020, 西班牙巴塞罗那 (线上), 2020 年 12 月 8–13 日, 第 4762–4772 页。International Committee on Computational Linguistics, 2020。doi: 10.18653/V1/2020.COLING-MAIN.419。URL <https://doi.org/10.18653/v1/2020.coling-main.419>。

附录

A. 贡献与致谢

研究与工程

Aixin Liu Bing
Xue Bingxuan
Wang Bochao
Wu Chengda Lu
Chenggang
Zhao Chengqi
Deng Chenyu
Zhang* Chong
Ruan Damai
Dai Daya Guo
Dejian Yang
Deli Chen
Erhang Li
Fangyun Lin
Fucong Dai Fuli
Luo* Guangbo
Hao Guanting
Chen Guowei Li
H. Zhang Han
Bao* Hanwei
Xu Haocheng
Wang* Haowei
Zhang Honghui
Ding Huajian
Xin* Huazuo
Gao Hui Qu
Jianzhong Guo
Jiashi Li Jiawei
Wang*
Jingchang Chen
Jingyang Yuan
Junjie Qiu
Junlong Li
Junxiao Song
Kai Dong Kai
Hu* Kaige Gao
Kang Guan
Kexin Huang
Kuai Yu Lean
Wang

Lecong Zhang
Liang Zhao Litong
Wang Liyue
Zhang
Mingchuan Zhang
Minghua Zhang
Minghui Tang
Panpan Huang
Peiyi Wang
Qiancheng Wang
Qihao Zhu Qinyu
Chen Qiushi Du
Ruiqi Ge Ruisong
Zhang Ruizhe Pan
Runji Wang
Runxin Xu Ruoyu
Zhang Shanghai
Lu Shangyan Z
hou Shanhuang
Chen Shengfeng
Ye Shirong Ma
Shiyu Wang
Shuiping Yu
Shunfeng Zhou
Shuting Pan Tao
Yun Tian Pei
Wangding Zeng
Wanjia Zhao*
Wen Liu
Wenfeng Liang
Wenjun Gao
Wenqin Yu
Wentao Zhang
Xiao Bi Xiaodong
Liu Xiaohan
Wang Xiaokang
Chen Xiaokang
Zhang Xiaotao
Nie Xin Cheng Xin
Liu

Xin Xie
Xingchao Liu
Xingkai Yu
Xinyu Yang
Xinyuan Li
Xuecheng Su
Xuheng Lin
Y.K. Li Y.Q.
Wang Y.X.
Wei Yang
Zhang
Yanhong Xu
Yao Li Yao
Zhao Yaofeng
Sun Yaohui
Wang Yi Yu
Yichao Zhang
Yifan Shi
Yiliang Xiong
Ying He Yishi
Piao Yisong
Wang Yixuan
Tan Yiyang
Ma* Yiyuan Liu
Yongqiang Guo
Yu Wu Yuan
Ou Yudian
Wang Yue
Gong Yuheng
Zou Yujia He
Yunfan Xiong
Yuxiang Luo
Yuxiang You
Yuxuan Liu
Yuyang Zhou
Z.F. Wu Z.Z.
Ren Zehui Ren
Zhangli Sha
Zhe Fu Zhean
Xu Zhenda Xie
Zhengyan Zhan

Zhigang Yan
Zhihong
Shao Zhiyu
Wu Zhuoshu
Li Zihui Gu
Zijia Zhu
Zijun Liu*
Zilin Li Ziwei
Xie Ziyang
Song Ziyi
Gao Zizheng
Pan

数据标注Bei
Feng Hui Li J.L.
Cai Jiaqi Ni Lei
Xu Meng Li
Ning Tian R.J.
Chen R.L. Jin
Ruyi Chen S.S.
Li Shuang Zhou
Tianyu Sun
X.Q. Li
Xiangyue Jin
Xiaojin Shen
Xiaosha Chen
Xiaowen Sun
Xiaoxiang
Wang Xinnan
Song Xinyi
Zhou Y.X. Zhu
Yanhong Xu
Yanping
Huang Yaohui
Li Yi Zheng
Yuchen Zhu
Yunxian Ma
Zhen Huang
Zhipeng Xu
Zhongyu Zhang

Zhewen Hao
Zhibin Gou
Zhicheng Ma

8

商业与合规Dongjie Ji

Jian Liang Jin
Chen Leyi Xia
Miaojun
Wang
Mingming Li
Peng Zhang
Shaoqing Wu
Shengfeng Ye
T. Wang

W.L. Xiao
Wei An
Xianzu Wang
Xinxia Shan
Ying Tang
Yukun Zha
Yuting Yan
Zhen Zhang

在每个角色内，作者按名字的首字母排序。名字带 * 的人员表示已离开我们团队。

B. 低精度训练的消融研究

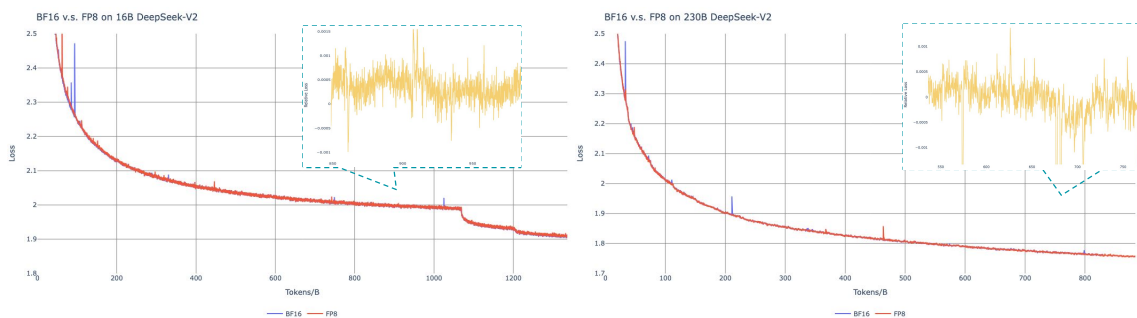


图 10 | BF16 与 FP8 训练的损失曲线比较。结果通过系数为 0.9 的指数移动平均（EMA）进行平滑。

B.1. FP8 与 BF16 训练

我们通过在两个不同规模的基线模型上将 FP8 混合精度框架与 BF16 训练进行比较，验证了该框架。在小规模下，我们在 1.33T 个 token 上训练了一个总参数量约为 16B 的基线 MoE 模型。在大规模下，我们在约 0.9T 个 token 上训练了一个总参数量约为 230B 的基线 MoE 模型。我们在图 10 中展示了训练曲线，并证明采用高精度累加和细粒度量化策略后，相对误差保持在 0.25% 以下。

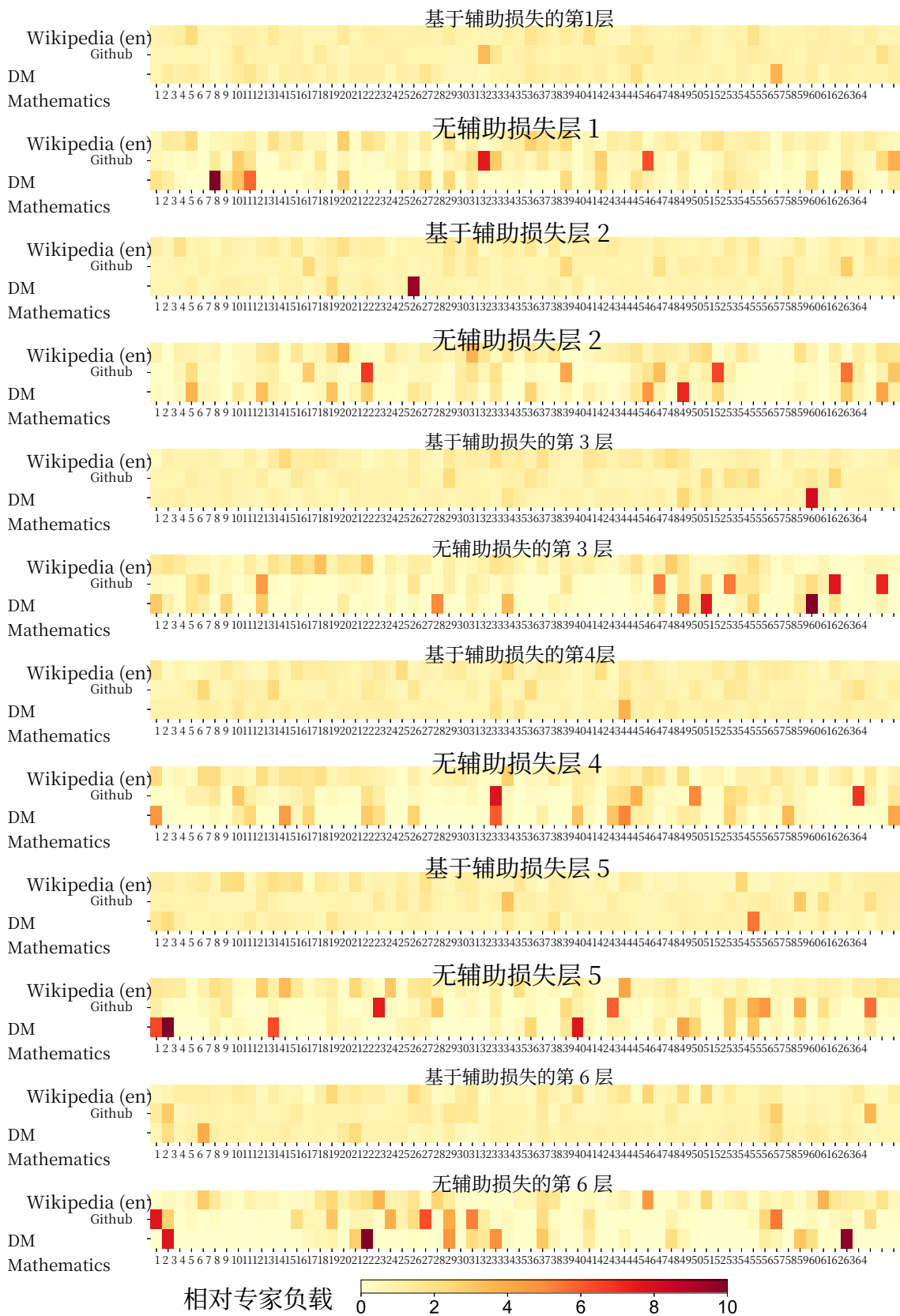
B.2. 关于分块量化的讨论

尽管我们的逐 tile 细粒度量化有效缓解了特征离群值引入的误差，但它要求对激活量化采用不同的分组方式，即前向传播使用 1×128 ，反向传播使用 128×1 。激活梯度也需要类似的处理。一种直接策略是像量化模型权重那样，按每 128×128 个元素进行块级量化。这样，反向传播只需进行转置。因此，我们开展了一项实验，将与 Dgrad 相关的所有张量都按块级方式量化。结果表明，Dgrad 操作（该操作计算激活梯度，并以链式方式反向传播至浅层）对精度高度敏感。具体而言，对激活梯度进行块级量化会导致

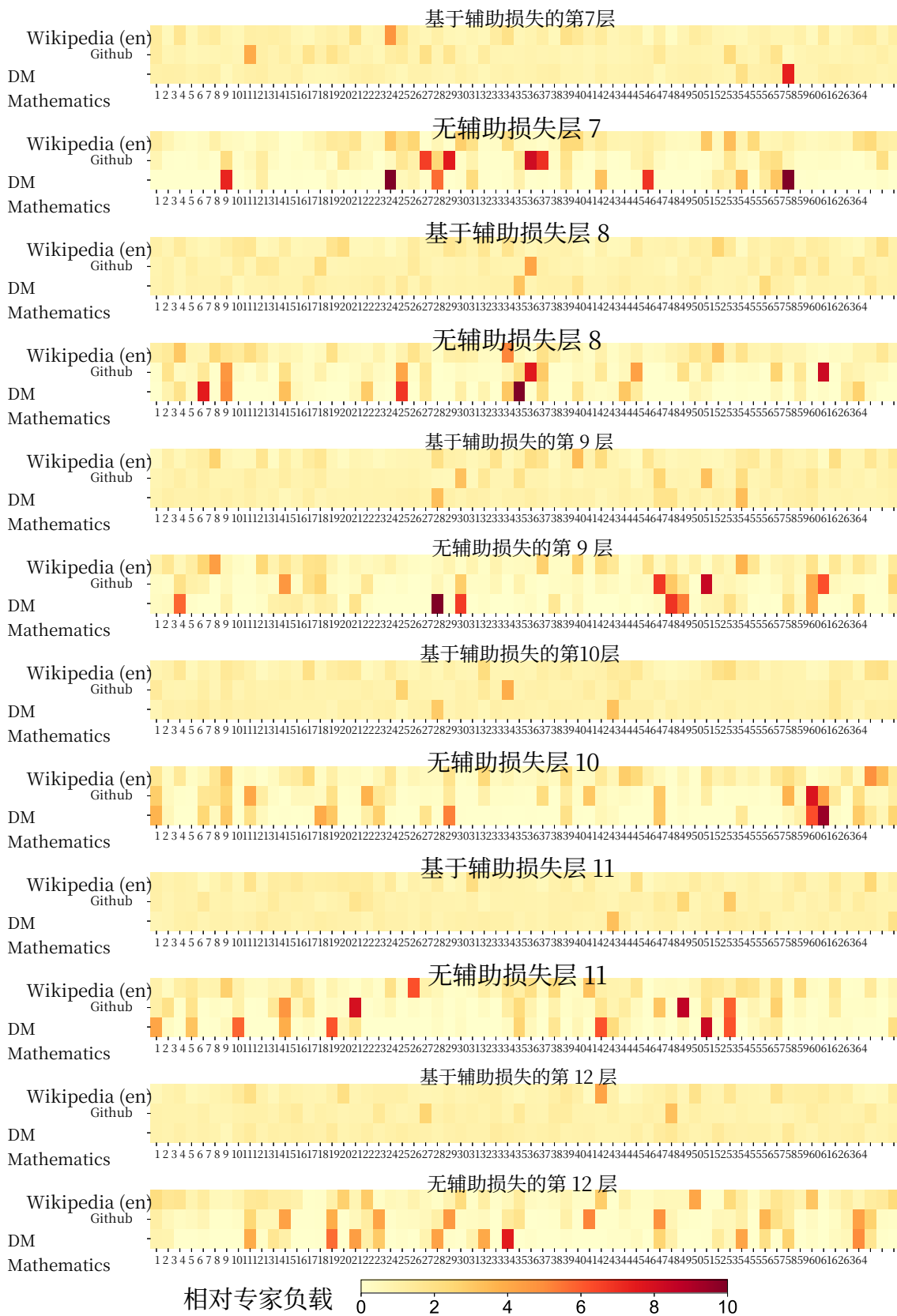
由约 16B 总参数组成、训练约 300B 个 token 的 MoE 模型发生模型发散。我们推测，这种敏感性源于不同 token 之间的激活梯度高度不均衡，从而产生与 token 相关的离群值 (Xi et al., 2023)。块级量化方法无法有效处理这些离群值。

C. 16B 基于辅助损失模型与无辅助损失模型的专家专门化模式

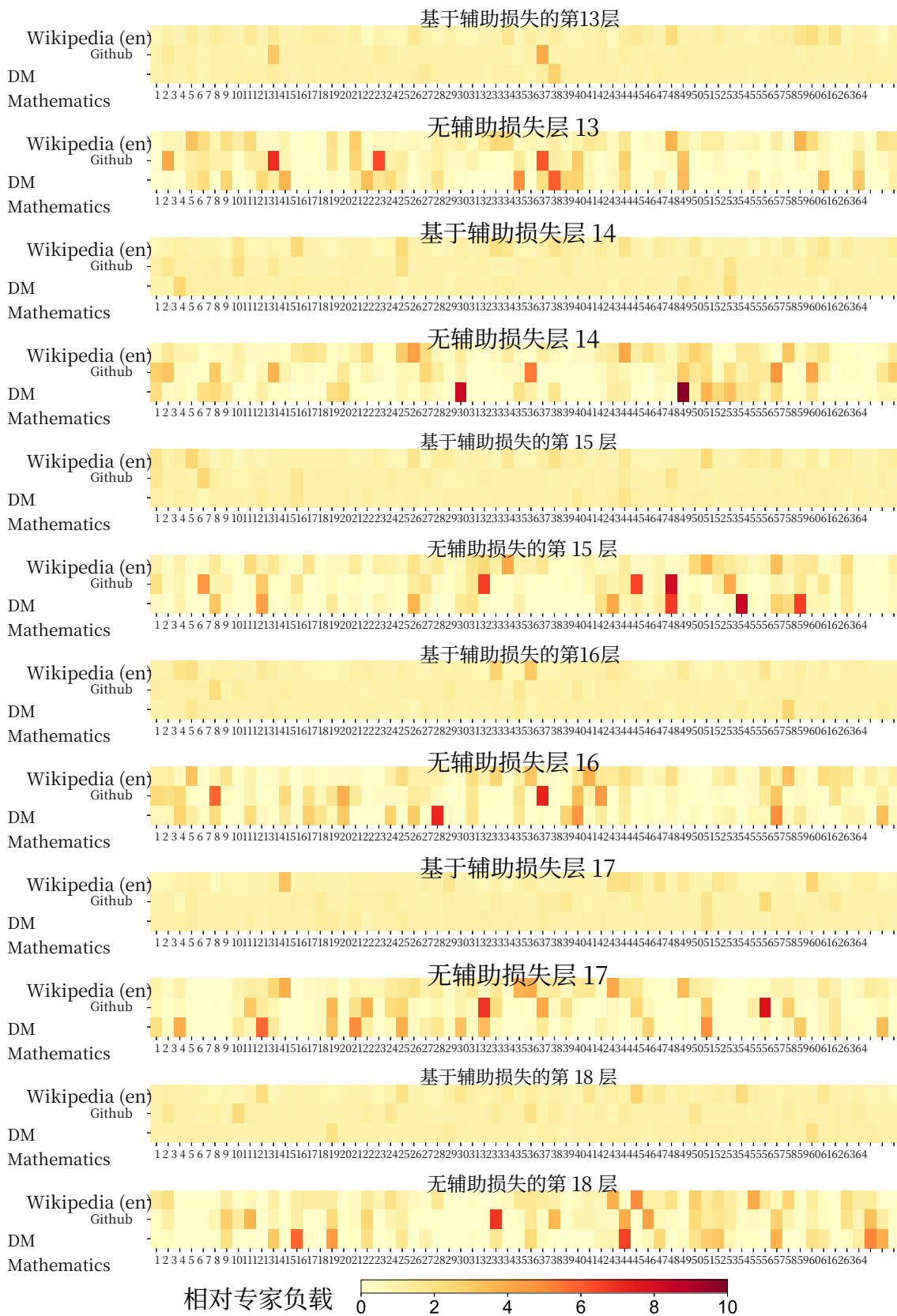
我们记录了在 Pile 测试集上、基于辅助损失的 16B 基线模型和无辅助损失模型的专家负载。正如图 10 所示，无辅助损失模型在所有层中往往具有更高的专家专门化程度。



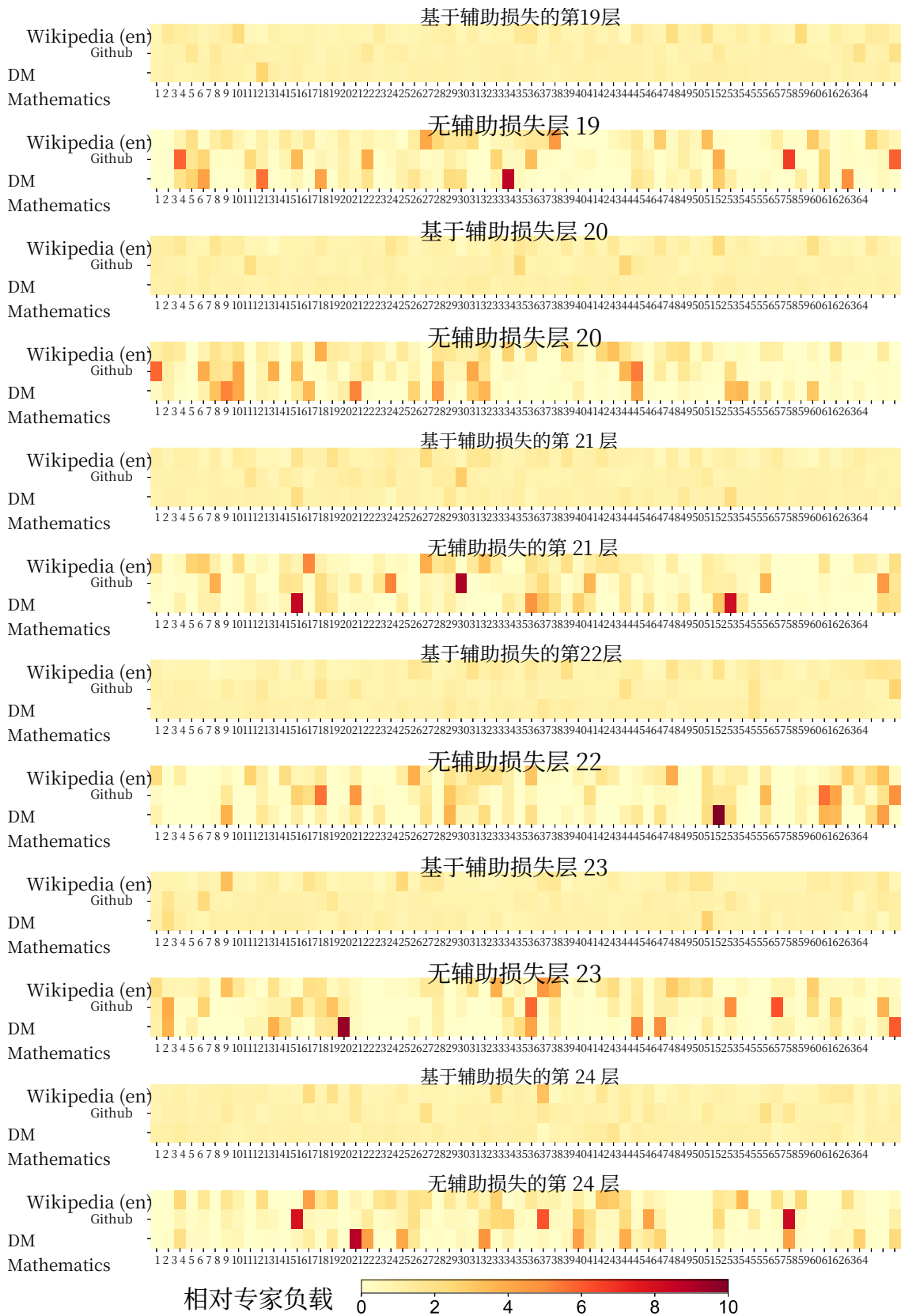
(a) 第 1-7 层



(b) 第 7-13 层



(c) 第 13-19 层



(d) 第 19-25 层

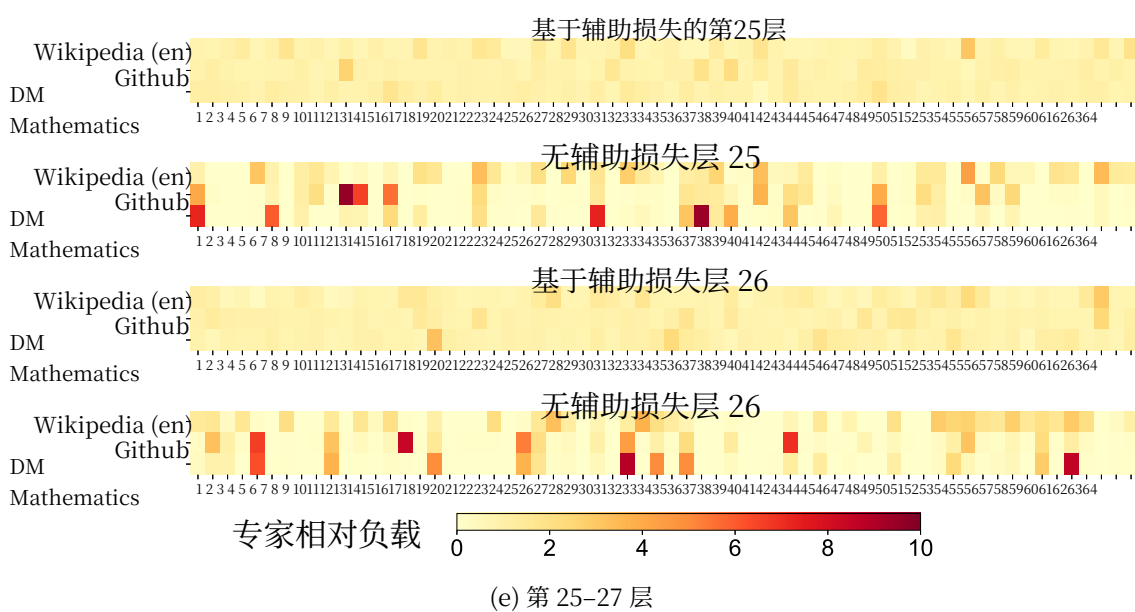


图 10 | Pile 测试集三个领域中无辅助损失模型与基于辅助损失模型的专家负载。无辅助损失模型相比基于辅助损失的模型呈现出更强的专家专门化模式。专家相对负载表示实际专家负载与理论平衡专家负载之比。