DeepSeek-V3 Technical Report

DeepSeek-AI

research@deepseek.com

Abstract

我们提出 DeepSeek-V3，一个强大的混合专家（Mixture-of-Experts, MoE）语言模型，总参数量为 671B，每个 token 激活 37B 参数。为实现高效推理与低成本训练，DeepSeek-V3 采用了多头潜在注意力（Multi-head Latent Attention, 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 token 预测训练目标以进一步提升模型性能。我们在 14.8 万亿个多样化且高质量的 token 上对 DeepSeek-V3 进行预训练，随后通过监督微调（SFT）和强化学习（RL）阶段充分释放其潜力。全面评估表明，DeepSeek-V3 优于其他开源模型，并达到了与领先闭源模型相媲美的性能。尽管性能卓越，DeepSeek-V3 的完整训练仅需 2.788M H800 GPU 小时。此外，其训练过程极其稳定。在整个训练过程中，我们未遇到任何不可恢复的损失激增，也未进行任何回滚操作。模型检查点可在 https://github.com/deepseek-ai/DeepSeek-V3 获取。

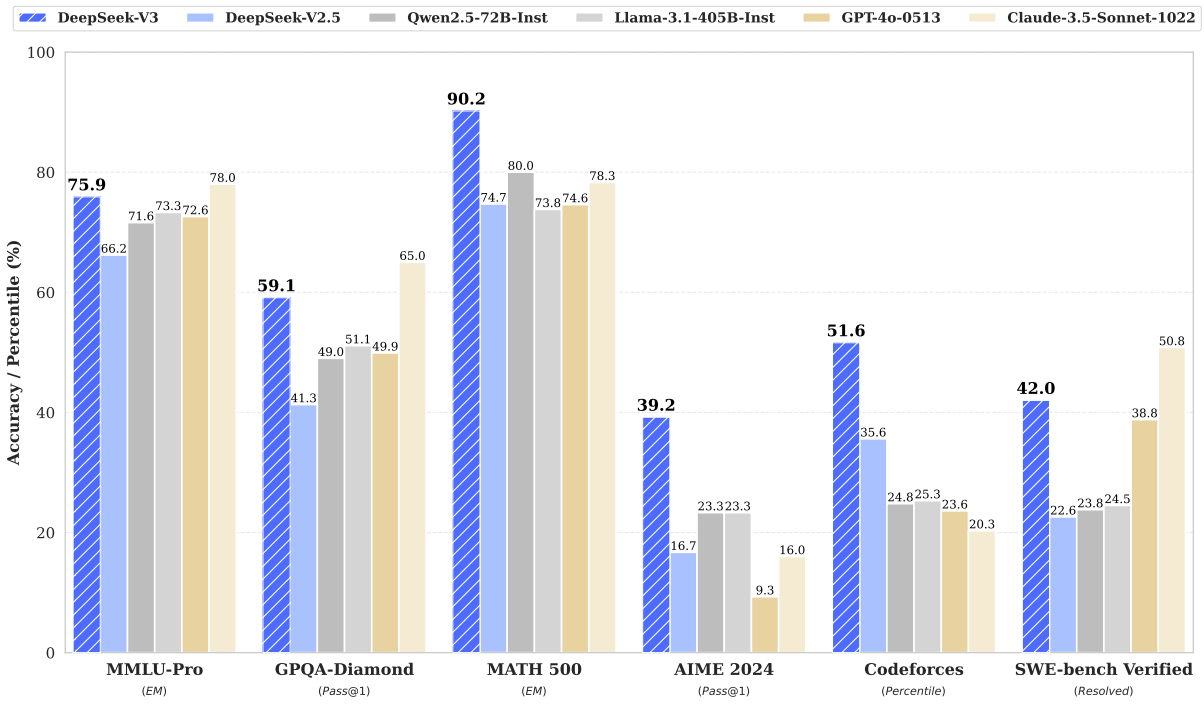


图 1 | DeepSeek-V3 及其对比模型的基准测试性能。

目录

1. 引言

近年来，大型语言模型（LLMs）正经历快速迭代与演进 (Anthropic, 2024; Google, 2024; OpenAI, 2024a)，逐步缩小与通用人工智能（AGI）之间的差距。除了闭源模型外，开源模型（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)）也取得了显著进展，致力于缩小与闭源模型的差距。为进一步突破开源模型的能力边界，我们放大了模型规模，并推出 DeepSeek-V3，这是一个拥有 671B 参数的大型混合专家（MoE）模型，其中每个 token 激活 37B 参数。

秉持前瞻性视野，我们始终致力于实现强大的模型性能与经济高效的成本。因此，在架构方面，DeepSeek-V3 依然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) 以实现低成本训练。这两种架构已在 DeepSeek-V2 (DeepSeek-AI, 2024c) 中得到验证，证明了它们在实现高效训练与推理的同时，能够保持稳健的模型性能。在基础架构之上，我们实施了另外两项策略以进一步提升模型能力。首先，DeepSeek-V3 首创了无辅助损失的负载均衡策略 (Wang et al., 2024a)，旨在最小化因鼓励负载均衡而对模型性能产生的负面影响。其次，DeepSeek-V3 采用了多 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表现。

为实现高效训练，我们支持 FP8 混合精度训练，并对训练框架进行了全面优化。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，我们引入了一种 FP8 混合精度训练框架，并首次在超大规模模型上验证了其有效性。通过对 FP8 计算与存储的支持，我们实现了训练加速与 GPU 显存占用的降低。在训练框架方面，我们设计了 DualPipe 算法以实现高效的流水线并行，该算法具有更少的流水线气泡，并通过计算与通信重叠隐藏了训练过程中的大部分通信开销。这种重叠机制确保了随着模型规模的进一步扩大，只要保持计算与通信比例恒定，我们仍能在跨节点使用细粒度专家的同时，实现接近零的全对全（all-to-all）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。此外，我们精心优化了内存占用，使得在不使用高昂的张量并行的情况下训练 DeepSeek-V3 成为可能。结合上述各项优化，我们实现了极高的训练效率。

在预训练阶段，我们在 14.8T 个高质量且多样化的 token 上训练 DeepSeek-V3。预训练过程极其稳定。在整个训练过程中，我们未遇到任何不可恢复的损失激增，也无需进行回滚。随后，我们对 DeepSeek-V3 进行了两阶段的上下文长度扩展。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。此后，我们对 DeepSeek-V3 基座模型进行了后训练，包括监督微调（SFT）和强化学习（RL），以使其对齐人类偏好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，同时仔细保持模型准确性与生成长度之间的平衡。

我们在一系列全面的基准测试上对 DeepSeek-V3 进行了评估。尽管训练成本经济高效，全

训练成本	预训练	上下文扩展	后训练	总计
H800 GPU 小时	2664K	119K	5K	2788K
美元	$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。

面评估表明 DeepSeek-V3-Base 已成为目前最强的开源基座模型，尤其在代码和数学领域表现突出。其对话版本也优于其他开源模型，并在一系列标准与开放式基准测试中达到了与 GPT-4o 和 Claude-3.5-Sonnet 等领先闭源模型相媲美的性能。

最后，我们再次强调 DeepSeek-V3 经济高效的训练成本（如表 1 所示），这是通过我们对算法、框架和硬件的优化协同设计实现的。在预训练阶段，每训练一万亿 token 仅需 180K H800 GPU 小时，即在我们拥有 2048 张 H800 GPU 的集群上运行 3.7 天。因此，我们的预训练阶段在不到两个月的时间内完成，耗时 2664K GPU 小时。加上上下文长度扩展的 119K GPU 小时和后训练的 5K GPU 小时，DeepSeek-V3 的完整训练总耗时仅为 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。需要注意的是，上述成本仅包含 DeepSeek-V3 的正式训练，不包括前期在架构、算法或数据方面的研究与消融实验所产生的成本。

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，我们首创了无辅助损失的负载均衡策略，最大限度地减少了因鼓励负载均衡而导致的性能下降。
- 我们研究了多 Token 预测（MTP）目标，并证明其对模型性能有益。该目标还可用于投机解码以实现推理加速。

预训练：迈向极致训练效率

- 我们设计了 FP8 混合精度训练框架，并首次在超大规模模型上验证了 FP8 训练的可行性与有效性。
- 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。这显著提升了我们的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 2.664M H800 GPU 小时的经济成本，我们便在 14.8T token 上完成了 DeepSeek-V3 的预训练，打造出目前最强的开源基座模型。预训练后的后续训练阶段仅需 0.1M GPU 小时。

后训练：基于 DeepSeek-R1 的知识蒸馏

- 我们提出一种创新方法，将长思维链(Chain-of-Thought, 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大语言模型（尤其是 DeepSeek-V3）中。我

们的流程优雅地将 R1 的验证与反思模式融入 DeepSeek-V3，显著提升了其推理性能。同时，我们也保持了对 DeepSeek-V3 输出风格与长度的控制。

核心评估结果总结

- **知识**：(1) 在 MMLU、MMLU-Pro 和 GPQA 等教育类基准测试中，DeepSeek-V3 超越了所有其他开源模型，在 MMLU、MMLU-Pro 和 GPQA 上分别取得了 88.5、75.9 和 59.1 的成绩。其性能与 GPT-4o 和 Claude-Sonnet-3.5 等领先的闭源模型相当，缩小了该领域开源与闭源模型之间的差距。(2) 在事实性基准测试中，DeepSeek-V3 在 SimpleQA 和 Chinese SimpleQA 上均展现出开源模型中的卓越性能。尽管在英文事实知识（SimpleQA）上略逊于 GPT-4o 和 Claude-Sonnet-3.5，但它在中文事实知识（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。
- **代码、数学与推理**：(1) 在所有非长 CoT 的开源和闭源模型中，DeepSeek-V3 在数学相关基准测试上取得了最先进的性能。值得注意的是，它甚至在 MATH-500 等特定基准测试上超越了 o1-preview，展现了其强大的数学推理能力。(2) 在代码相关任务中，DeepSeek-V3 在 LiveCodeBench 等编程竞赛基准测试中表现最佳，巩固了其在该领域的领先地位。在工程相关任务中，尽管 DeepSeek-V3 的表现略低于 Claude-Sonnet-3.5，但仍以显著优势领先于所有其他模型，展现了其在各类技术基准测试中的竞争力。

在本文的剩余部分，我们首先详细介绍 DeepSeek-V3 的模型架构（第 2 节）。随后，我们介绍我们的基础设施，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，包括训练数据的构建、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，我们探讨后训练方面的工作，包括监督微调（SFT）、强化学习（RL）、相应的评估与讨论（第 5 节）。最后，我们总结本文工作，讨论 DeepSeek-V3 的现有局限性，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 的基础架构，该架构采用多头潜在注意力（Multi-head Latent Attention, MLA）(DeepSeek-AI, 2024c) 以实现高效推理，并采用 DeepSeekMoE (Dai et al., 2024) 以实现经济高效的训练。随后，我们提出了一种多 Token 预测（MTP）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。对于其他未明确提及的细节，DeepSeek-V3 均沿用 DeepSeek-V2 (DeepSeek-AI, 2024c) 的设置。

2.1. 基础架构

DeepSeek-V3 的基础架构仍基于 Transformer (Vaswani et al., 2017) 框架。为实现高效推理与经济高效的训练，DeepSeek-V3 同样采用了经 DeepSeek-V2 充分验证的 MLA 和 DeepSeekMoE。与 DeepSeek-V2 相比，一个例外是我们为 DeepSeekMoE 额外引入了一种无辅助损失的负载均衡策略 (Wang et al., 2024a)，以缓解为确保负载均衡而带来的性能下降。图 2 展示了 DeepSeek-V3 的基础架构，本节我们将简要回顾 MLA 和 DeepSeekMoE 的细节。

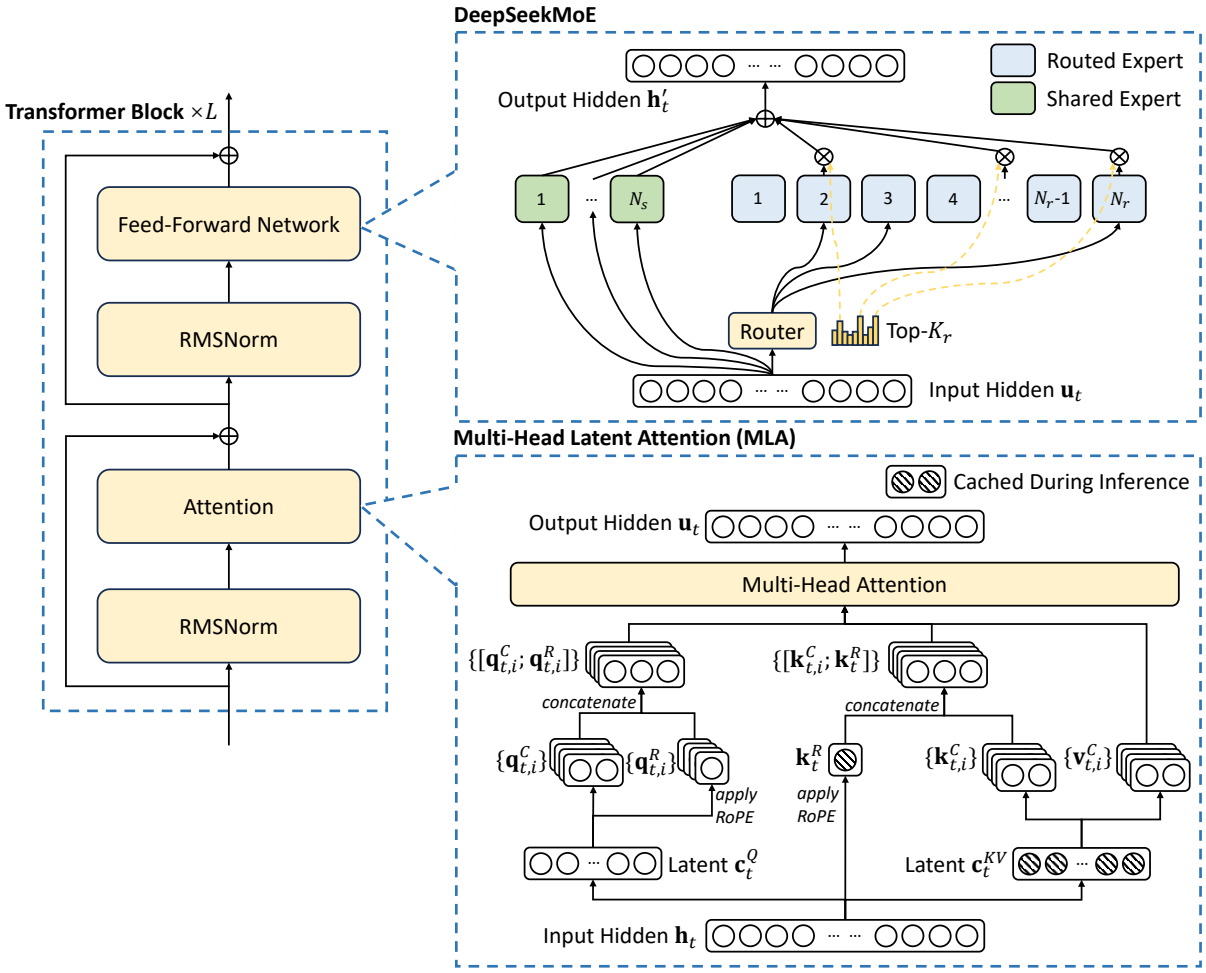


图 2 | DeepSeek-V3 基础架构示意图。沿用 DeepSeek-V2 的设计，我们采用 MLA 和 DeepSeek-MoE 以实现高效推理与经济高效的训练。

2.1.1. 多头潜在注意力

在注意力机制方面，DeepSeek-V3 采用 MLA 架构。设 d 表示嵌入维度，n_h 表示注意力头数，d_h 表示每个头的维度，$\mathbf{h}_t \in \mathbb{R}^d$ 表示给定注意力层中第 t 个 token 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，以减少推理过程中的键值（KV）缓存：

$$\boxed{\mathbf{c}^{KV}_t} = W^{DKV}\mathbf{h}_t, \tag{1}$$

$$[\mathbf{k}^C_{t,1};\mathbf{k}^C_{t,2};\ldots;\mathbf{k}^C_{t,n_h}] = \mathbf{k}^C_t = W^{UK}\mathbf{c}^{KV}_t, \tag{2}$$

$$\boxed{\mathbf{k}^R_t} = \mathrm{RoPE}(W^{KR}\mathbf{h}_t), \tag{3}$$

$$\mathbf{k}_{t,i} = [\mathbf{k}^C_{t,i};\mathbf{k}^R_t], \tag{4}$$

$$[\mathbf{v}^C_{t,1};\mathbf{v}^C_{t,2};\ldots;\mathbf{v}^C_{t,n_h}] = \mathbf{v}^C_t = W^{UV}\mathbf{c}^{KV}_t, \tag{5}$$

其中 $\mathbf{c}^{KV}_t \in \mathbb{R}^{d_c}$ 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；$W^{UK}, W^{UV} \in \mathbb{R}^{d_h n_h \times d_c}$ 分别为键和值的上投影矩阵；$W^{KR} \in \mathbb{R}^{d^R_h \times d}$ 是用于生成携带旋

转位置编码（RoPE）(Su et al., 2024) 的解耦键的矩阵；RoPE(·) 表示应用 RoPE 矩阵的操作；[·;·] 表示拼接。需要注意的是，对于 MLA，在生成过程中仅需缓存蓝色框内的向量（即 $\mathbf{c}_t^{KV}$ 和 $\mathbf{k}_t^R$），这显著减少了 KV 缓存，同时保持了与标准多头注意力（MHA）(Vaswani et al., 2017) 相当的性能。

对于注意力查询，我们也进行了低秩压缩，这可以减少训练过程中的激活内存：

$$\mathbf{c}_t^Q = W^{DQ}\mathbf{h}_t, \tag{6}$$

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

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

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

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；$W^{DQ} \in \mathbb{R}^{d_c' \times d}, W^{UQ} \in \mathbb{R}^{d_h n_h \times d_c'}$ 分别为查询的下投影和上投影矩阵；$W^{QR} \in \mathbb{R}^{d_h^R n_h \times d_c'}$ 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（$\mathbf{v}_{j,i}^C$）相结合，得到最终的注意力输出 $\mathbf{u}_t$：

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

$$\mathbf{u}_t = W^O[\mathbf{o}_{t,1}; \mathbf{o}_{t,2}; ...; \mathbf{o}_{t,n_h}], \tag{11}$$

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

2.1.2. 带无辅助损失负载均衡的 DeepSeekMoE

DeepSeekMoE 的基础架构。 在前馈网络（FFN）方面，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。设 $\mathbf{u}_t$ 表示第 t 个 token 的 FFN 输入，我们按如下方式计算 FFN 输出 $\mathbf{h}_t'$：

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

$$g_{i,t} = \frac{g_{i,t}'}{\sum_{j=1}^{N_r} g_{j,t}'}, \tag{13}$$

$$g_{i,t}' = \begin{cases} s_{i,t}, & s_{i,t} \in \mathrm{Topk}(\{s_{j,t} | 1 \leqslant j \leqslant N_r\}, K_r), \\ 0, & \text{otherwise}, \end{cases} \tag{14}$$

$$s_{i,t} = \mathrm{Sigmoid}\left(\mathbf{u}_t{}^T \mathbf{e}_i\right), \tag{15}$$

其中 N_s 和 N_r 分别表示共享专家和路由专家的数量；$\mathrm{FFN}_i^{(s)}(\cdot)$ 和 $\mathrm{FFN}_i^{(r)}(\cdot)$ 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 token 与专家的亲和力；$\mathbf{e}_i$ 是第 i 个路由专家的中心向量；$\mathrm{Topk}(\cdot, K)$ 表示在第 t 个 token 与所有路由专家计算出的亲和力分数中，包含最高 K 个分数的集合。与 DeepSeek-V2 略有不同，DeepSeek-V3 使用 sigmoid 函数计算亲和力分数，并对所有选中的亲和力分数进行归一化以生成门控值。

无辅助损失负载均衡。 对于 MoE 模型，专家负载不均衡会导致路由崩溃 (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取得更好的平衡，我们首创了一种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一个偏置项 b_i，并将其加到对应的亲和力分数 $s_{i,t}$ 上，以确定 Top-K 路由：

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

注意，偏置项仅用于路由。将与 FFN 输出相乘的门控值仍然由原始亲和力分数 $s_{i,t}$ 导出。在训练过程中，我们持续监控每个训练步骤中整个批次的专家负载。在每个步骤结束时，如果对应的专家负载过重，我们将偏置项减少 γ；如果负载过轻，则增加 γ，其中 γ 是一个称为偏置更新速度的超参数。通过这种动态调整，DeepSeek-V3 在训练期间保持专家负载均衡，并取得了优于仅通过纯辅助损失来促进负载均衡的模型的性能。

互补的序列级辅助损失。 尽管 DeepSeek-V3 主要依赖无辅助损失策略来实现负载均衡，但为了防止任何单个序列内部出现极端不均衡，我们还采用了一种互补的序列级平衡损失：

$$\mathcal{L}_{\mathrm{Bal}} = \alpha \sum_{i=1}^{N_r} f_i P_i, \tag{17}$$

$$f_i = \frac{N_r}{K_r T} \sum_{t=1}^{T} \mathbb{1}\left(s_{i,t} \in \mathrm{Topk}(\{s_{j,t} | 1 \leqslant j \leqslant N_r\}, K_r)\right), \tag{18}$$

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

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

其中平衡因子 α 是一个超参数，对于 DeepSeek-V3 将被赋予一个极小的值；$\mathbb{1}(\cdot)$ 表示指示函数；T 表示序列中的 token 数量。序列级平衡损失促使每个序列上的专家负载保持均衡。

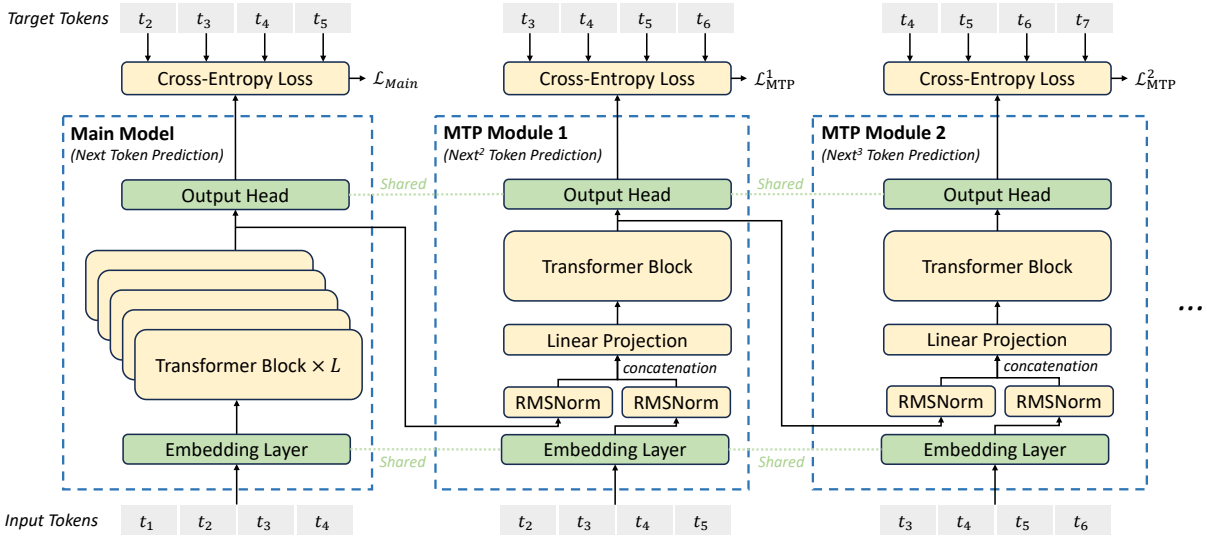


图 3 | 我们多 Token 预测（MTP）实现的示意图。我们在每个预测深度都保留了每个 token 预测的完整因果链。

节点限制路由。 与 DeepSeek-V2 使用的设备限制路由类似，DeepSeek-V3 也采用了一种受限路由机制，以限制训练期间的通信开销。简而言之，我们确保每个 token 最多被发送到 M 个节点，这些节点是根据分布在每个节点上的专家的最高 $\frac{K_r}{M}$ 个亲和力分数之和来选择的。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。

无 Token 丢弃。 得益于有效的负载均衡策略，DeepSeek-V3 在整个训练过程中保持了良好的负载平衡。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阶段的负载均衡，因此 DeepSeek-V3 在推理期间也不会丢弃 token。

2.2. 多 Token 预测

受 Gloeckle et al. (2024) 的启发，我们为 DeepSeek-V3 探索并设定了一个多 Token 预测（MTP）目标，该目标将每个位置的预测范围扩展至多个未来 token。一方面，MTP 目标使训练信号更加密集，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，以更好地预测未来 token。图 3 展示了我们的 MTP 实现。与使用独立输出头并行预测 D 个额外 token 的 Gloeckle et al. (2024) 不同，我们顺序预测额外 token，并在每个预测深度保持完整的因果链。本节将介绍我们 MTP 实现的细节。

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 token。第 k 个 MTP 模块由一个共享的嵌入层 $\mathrm{Emb}(\cdot)$、一个共享的输出头 $\mathrm{OutHead}(\cdot)$、一个 Transformer 块 $\mathrm{TRM}_k(\cdot)$ 以及一个投影矩阵 $M_k \in \mathbb{R}^{d \times 2d}$ 组成。对于第 i 个输入 token t_i，在第 k 个预测深度，我们首先通过线性投影将第 $(k-1)$ 个深度的第 i 个 token 的表示 $\mathbf{h}_i^{k-1} \in \mathbb{R}^d$ 与第 $(i+k)$ 个 token

的嵌入 $Emb(t_{i+k}) \in \mathbb{R}^d$ 进行组合：

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

其中 $[\cdot;\cdot]$ 表示拼接操作。特别地，当 $k = 1$ 时，$\mathbf{h}_i^{k-1}$ 指的是由主模型给出的表示。注意，对于每个 MTP 模块，其嵌入层与主模型共享。组合后的 $\mathbf{h}_i'^k$ 作为第 k 个深度 Transformer 块的输入，以生成当前深度的输出表示 $\mathbf{h}_i^k$：

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

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

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

输出头 $\text{OutHead}(\cdot)$ 将表示线性映射为 logits，随后应用 $\text{Softmax}(\cdot)$ 函数来计算第 k 个额外 token 的预测概率。同样，对于每个 MTP 模块，其输出头也与主模型共享。我们保持预测因果链的原则与 EAGLE (Li et al., 2024b) 类似，但其主要目标是投机解码 (Leviathan et al., 2023; Xia et al., 2023)，而我们利用 MTP 来提升训练效果。

MTP 训练目标。 对于每个预测深度，我们计算一个交叉熵损失 $\mathcal{L}_{\text{MTP}}^k$：

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

其中 T 表示输入序列长度，t_i 表示第 i 个位置的真实 token，$P_i^k[t_i]$ 表示由第 k 个 MTP 模块给出的 t_i 的对应预测概率。最后，我们计算所有深度上 MTP 损失的平均值，并将其乘以权重因子 λ，以获得整体 MTP 损失 $\mathcal{L}_{\text{MTP}}$，该损失作为 DeepSeek-V3 的额外训练目标：

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

推理中的 MTP。 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，主模型可以独立且正常运行。此外，我们还可以将这些 MTP 模块重新用于投机解码，以进一步改善生成延迟。

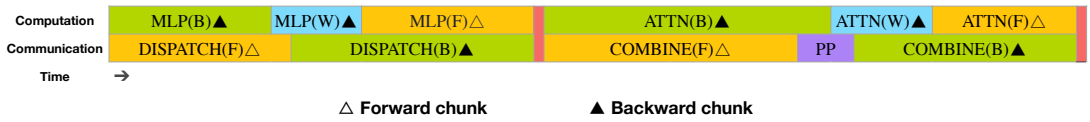


图 4 | 一对独立的前向和后向块的重叠策略（Transformer 块的边界未对齐）。橙色表示前向，绿色表示“输入反向”，蓝色表示“权重反向”，紫色表示 PP 通信，红色表示屏障。All-to-All 和 PP 通信均可完全隐藏。

3. 基础设施

3.1. 计算集群

DeepSeek-V3 在一个配备 2048 块 NVIDIA H800 GPU 的集群上进行训练。H800 集群中的每个节点包含 8 块 GPU，节点内通过 NVLink 和 NVSwitch 互联。在不同节点之间，采用 InfiniBand (IB) 互连技术以促进通信。

3.2. 训练框架

DeepSeek-V3 的训练由 HAI-LLM 框架提供支持，这是一个由我们的工程师从头构建的高效、轻量级训练框架。总体而言，DeepSeek-V3 采用了 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，我们实施了细致的工程优化。首先，我们设计了 DualPipe 算法以实现高效的流水线并行。与现有的 PP 方法相比，DualPipe 的流水线气泡更少。更重要的是，它重叠了前向和后向过程中的计算与通信阶段，从而解决了跨节点专家并行带来的沉重通信开销挑战。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多处理器（SM）。最后，我们仔细优化了训练期间的内存占用，从而使我们能够在不使用昂贵的张量并行（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比约为 1:1，效率较低。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和后向的计算-通信阶段来加速模型训练，还减少了流水线气泡。

DualPipe 的核心思想是在一对独立的前向和后向块内重叠计算与通信。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，对于后向块，`attention` 和 `MLP` 均进一步拆分为两部分：`backward for input` 和 `backward for weights`，类似于 ZeroBubble (Qi et al., 2023b) 的做法。此外，我们还有一个 `PP communication` 组件。如图 4 所示，对于一对前向和后向块，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。在此重叠策略下，我们可以确保在执行过程中 All-to-All 和 PP 通信均能被完全隐藏。基于这种高效的重叠策略，完整的 DualPipe 调度如图 5 所示。它

图 5 | 8 个 PP 秩和两个方向上 20 个微批次的 DualPipe 调度示例。反向的微批次与前向的微批次对称，因此为简化说明，我们省略了它们的批次 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×	PP
ZB1P	$(PP-1)(F+B-2W)$	1×	PP
DualPipe (Ours)	$(\frac{PP}{2}-1)(F\&B+B-3W)$	2×	$PP+1$

表 2 | 不同流水线并行方法的流水线气泡与内存使用对比。F 表示前向块的执行时间，B 表示完整后向块的执行时间，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后向块的执行时间。

采用双向流水线调度，同时从流水线的两端输入微批次，并且大部分通信可以被完全重叠。这种重叠还确保了随着模型规模的进一步扩大，只要保持计算与通信比恒定，我们仍能在跨节点使用细粒度专家的同时，实现近乎零的 All-to-All 通信开销。

此外，即使在通信负担不重的更一般场景下，DualPipe 仍表现出效率优势。在表 2 中，我们总结了不同 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，同时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副本，但由于我们在训练期间使用了较大的 EP 规模，这并不会显著增加内存消耗。与 Chimera (Li and Hoefler, 2021) 相比，DualPipe 仅要求流水线阶段数和微批次数能被 2 整除，而不要求微批次数能被流水线阶段数整除。此外，对于 DualPipe，随着微批次数量的增加，气泡和激活内存均不会增加。

3.2.2. 跨节点 *All-to-All* 通信的高效实现

为了确保 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（包括分发与合并），以节省专用于通信的 SM 数量。这些内核的实现与 MoE 门控算法以及我们集群的网络拓扑结构进行了协同设计。具体而言，在我们的集群中，跨节点 GPU 通过 IB 实现全互联，而节点内通信则通过 NVLink 处理。NVLink 提供 160 GB/s 的带宽，大约是 IB（50 GB/s）的 3.2 倍。为了有效利用 IB 和 NVLink 的不同带宽，我们限制每个 token 最多被分发到 4 个节点，从而减少 IB 流量。对于每个 token，一旦做出路由决策，它将首先通过 IB 传输到目标节点上具有相同节点内索引的 GPU。到达目标节点后，我们将尽力确保其通过 NVLink 瞬时转发到承载目标专家（expert）的特定 GPU，而不会被随后到达的 token 阻塞。通过这种方式，IB 和 NVLink 的通信完全重叠，每个 token 可以在不产生额外 NVLink 开销的情况下，高效地平均选择每个节点 3.2 个专家。这意味着，尽管 DeepSeek-V3 在实际中仅选择 8 个路由专家，但

它可以将此数量最多扩展至 13 个专家（4 个节点 × 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 专业化技术 (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 的转发以及 (3) NVLink 接收分别由不同的 warp 处理。分配给每个通信任务的 warp 数量会根据所有 SM 上的实际工作负载进行动态调整。同样，在合并过程中，(1) NVLink 发送、(2) NVLink 到 IB 的转发与累加，以及 (3) IB 接收与累加，也由动态调整的 warp 处理。此外，分发和合并内核均与计算流重叠，因此我们也考虑了它们对其他 SM 计算内核的影响。具体而言，我们采用了定制的 PTX（并行线程执行）指令，并自动调优通信块大小，这显著减少了对 L2 缓存的使用以及对其它 SM 的干扰。

3.2.3. 以极小开销实现极致内存节省

为了降低训练过程中的内存占用，我们采用了以下技术。

RMSNorm 与 MLA 上投影的重计算。 我们在反向传播期间重新计算所有 RMSNorm 操作和 MLA 上投影，从而消除了持久存储其输出激活值的需要。该策略仅带来微小的开销，却显著降低了存储激活值所需的内存。

CPU 中的指数移动平均。 在训练期间，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尽早评估模型性能。EMA 参数存储在 CPU 内存中，并在每个训练步骤后异步更新。该方法使我们能够在不产生额外内存或时间开销的情况下维护 EMA 参数。

多 Token 预测的共享词嵌入与输出头。 借助 DualPipe 策略，我们将模型的最浅层（包括词嵌入层）和最深层（包括输出头）部署在同一个 PP（流水线并行）rank 上。这种安排使得 MTP（多 Token 预测）模块与主模型之间能够物理共享词嵌入和输出头的参数与梯度。这种物理共享机制进一步提升了我们的内存效率。

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)，我们提出了一种利用 FP8 数据格式训练 DeepSeek-V3 的细粒度混合精度框架。尽管低精度训练前景广阔，但其常受到激活值、权重和梯度中异常值存在的限制 (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预训练的研究相对较少 (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：采用 $1\times N_c$ 元素的图块（tile）分组或 $N_c\times N_c$ 元素的块（block）分组。在我们的高精度累加过程中，相关的反量化开销得到了大幅缓解，这是实现高精度 FP8 通用矩阵乘法（GEMM）的关键环节。此外，为进一步降低 MoE 训练中的内存和通信开销，我们以 FP8 格式缓存和分发激活值，同时将低

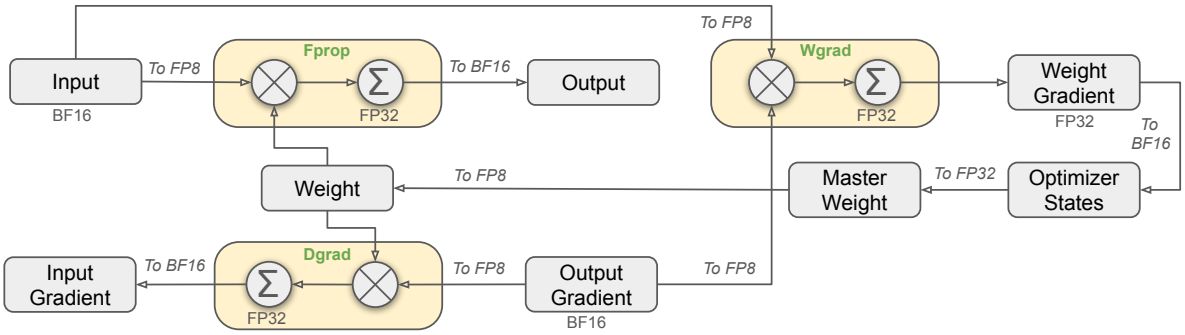


图 6 | 采用 FP8 数据格式的整体混合精度框架。为便于说明，仅展示了 `Linear` 算子。

精度优化器状态以 BF16 格式存储。我们在两个与 DeepSeek-V2-Lite 和 DeepSeek-V2 规模相近的模型上验证了所提出的 FP8 混合精度框架，训练了约 1 万亿个 token（更多细节见附录 B.1）。值得注意的是，与 BF16 基线相比，我们 FP8 训练模型的相对损失误差始终保持在 0.25% 以下，这一水平完全处于训练随机性的可接受范围内。

3.3.1. 混合精度框架

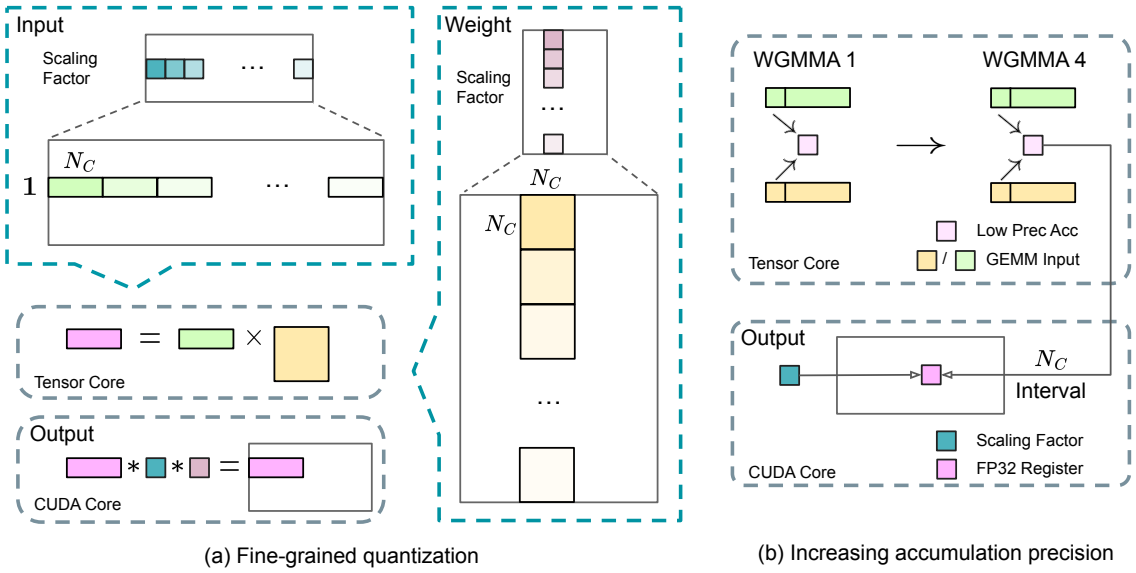
基于低精度训练中广泛采用的技术 (Kalamkar et al., 2019; Narang et al., 2017)，我们提出了一种用于 FP8 训练的混合精度框架。在该框架中，大多数计算密集型操作在 FP8 下进行，而少数关键操作则策略性地保留在其原始数据格式中，以平衡训练效率与数值稳定性。整体框架如图 6 所示。

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 操作接受 FP8 张量作为输入，并输出 BF16 或 FP32 格式的结果。如图 6 所示，与 `Linear` 算子相关的三个 GEMM 操作，即 `Fprop`（前向传播）、`Dgrad`（激活值反向传播）和 `Wgrad`（权重反向传播），均在 FP8 下执行。与原始的 BF16 方法相比，该设计在理论上可将计算速度提升一倍。此外，FP8 `Wgrad` GEMM 允许将激活值以 FP8 格式存储，以供反向传播使用。这显著降低了内存消耗。

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，仍需更高的精度。此外，一些低开销算子也可以采用更高精度，且对整体训练成本的增加微乎其微。因此，经过仔细调研，我们对以下组件保留了原始精度（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。这些针对性的高精度保留措施确保了 DeepSeek-V3 训练过程的稳定性。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，通过跨多个 DP（数据并行）进程的高效分片，其影响可降至最低。

3.3.2. 量化与乘法带来的精度提升

基于我们的混合精度 FP8 框架，我们引入了多种策略以提升低精度训练的准确性，重点关注量化方法与乘法过程。



(a) Fine-grained quantization (b) Increasing accumulation precision

图 7 | (a) 我们提出了一种细粒度量化方法以缓解特征异常值引起的量化误差；为简化说明，仅展示了 `Fprop`。(b) 结合我们的量化策略，我们通过每 $N_C = 128$ 个元素的 MMA 操作将累加过程提升至 CUDA Cores 进行高精度计算，从而提升了 FP8 GEMM 的精度。

细粒度量化。 在低精度训练框架中，由于 FP8 格式的指数位减少导致动态范围受限，溢出和下溢是常见的挑战。作为标准做法，通常通过将输入张量的最大绝对值缩放到 FP8 的最大可表示值，使输入分布对齐到 FP8 格式的可表示范围 (Narang et al., 2017)。该方法使得低精度训练对激活值异常值高度敏感，从而严重降低量化精度。为解决此问题，我们提出了一种细粒度量化方法，在更细的粒度上应用缩放。如图 7 (a) 所示，(1) 对于激活值，我们以 `1x128` 的图块（tile）为单位对元素进行分组和缩放（即每个 token 对应 128 个通道）；(2) 对于权重，我们以 `128x128` 的块（block）为单位对元素进行分组和缩放（即每 128 个输入通道对应 128 个输出通道）。该方法通过根据更小的元素组调整缩放比例，确保量化过程能更好地适应异常值。在附录 B.2 中，我们进一步讨论了当以与权重量化相同的方式按块对激活值进行分组和缩放时，训练不稳定的问题。

我们方法的一项关键改进是在 GEMM 操作的内部维度上引入了每组缩放因子。标准 FP8 GEMM 并不直接支持此功能。然而，结合我们的高精度 FP32 累加策略，它可以被高效实现。

值得注意的是，我们的细粒度量化策略与微缩放（microscaling）格式的理念高度一致，而 NVIDIA 下一代 GPU（Blackwell 系列）的 Tensor Cores 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的工作提供参考，以跟上最新 GPU 架构的发展步伐。

提升累加精度。 低精度 GEMM 操作常受下溢问题困扰，其准确性很大程度上依赖于高精度累加，这通常以 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，这显著低于 FP32 累加精度。当内部维度 `K` 较大时，该问题将变得更加突出 (Wortsman et al., 2023)，这是大规模模型训

练中增加批次大小和模型宽度的典型场景。以 K = 4096 的两个随机矩阵的 GEMM 操作为例，在我们的初步测试中，Tensor Cores 中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，有限的累加精度仍然是少数 FP8 框架中的默认选项 (NVIDIA, 2024b)，严重限制了训练精度。

为解决此问题，我们采用了提升至 CUDA Cores 进行高精度计算的策略 (Thakkar et al., 2023)。该过程如图 7 (b) 所示。具体而言，在 Tensor Cores 上执行 MMA（矩阵乘累加）操作时，中间结果使用有限的位宽进行累加。一旦达到 N_C 的间隔，这些部分结果将被复制到 CUDA Cores 上的 FP32 寄存器中，在此执行全精度 FP32 累加。如前所述，我们的细粒度量化沿内部维度 K 应用每组缩放因子。这些缩放因子可以在 CUDA Cores 上作为反量化过程高效相乘，且仅带来极小的额外计算开销。

值得注意的是，此项修改降低了单个 warpgroup 的 WGMMA（Warpgroup 级矩阵乘累加）指令发射率。然而，在 H800 架构上，通常会有两个 WGMMA 并发持续运行：当一个 warpgroup 执行提升操作时，另一个能够执行 MMA 操作。该设计实现了两种操作的重叠，保持了 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）代表了能够在不引入显著开销的情况下显著提升精度的最小累加间隔。

尾数优先于指数。 与先前工作 (NVIDIA, 2024b; Peng et al., 2023b; Sun et al., 2019b) 采用的混合 FP8 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））不同，我们在所有张量上均采用 E4M3 格式以获得更高精度。我们将该方法的可行性归因于我们的细粒度量化策略，即图块级与块级缩放。通过对更小的元素组进行操作，我们的方法在这些分组元素间有效共享指数位，从而缓解了动态范围受限的影响。

在线量化。 张量级量化框架 (NVIDIA, 2024b; Peng et al., 2023b) 采用了延迟量化，该方法维护先前迭代中最大绝对值的历史记录以推断当前值。为确保缩放比例准确并简化框架，我们在线计算每个 1x128 激活图块或 128x128 权重块的最大绝对值。基于此，我们推导出缩放因子，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，我们通过将缓存的激活值和优化器状态压缩为更低精度的格式，进一步降低了内存消耗和通信开销。

低精度优化器状态。 我们采用 BF16 数据格式而非 FP32 来跟踪 AdamW (Loshchilov and Hutter, 2017) 优化器中的一阶和二阶矩，且未引起可观察到的性能下降。然而，主权重（由优化器存储）和梯度（用于批次大小累积）仍保留为 FP32 格式，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 操作在 FP8 下执行。为降低内存消耗，将激活值以 FP8 格式缓存以供 Linear 算子的反向传播使用是一个自然的选择。然而，为了实现低成本的高精度训练，我们对几个算子进行了特殊处理：

(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了一种定制的 E5M6 数据格式。此外，在反向传播中，这些激活值将从 1x128 量化图块转换为 128x1 图块。为避免引入额外的量化误差，所有缩放因子均进行取整缩放，即 2 的整数次幂。

(2) MoE 中 SwiGLU 算子的输入。为进一步降低内存成本，我们缓存 SwiGLU 算子的输入，并在反向传播中重新计算其输出。这些激活值同样使用我们的细粒度量化方法以 FP8 格式存储，在内存效率与计算精度之间取得了平衡。

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 上投影之前的激活值量化为 FP8，然后应用 dispatch 组件，这与 MoE 上投影中的 FP8 Fprop 兼容。与注意力算子之后的 Linear 输入类似，该激活值的缩放因子也为 2 的整数次幂。类似的策略也应用于 MoE 下投影之前的激活值梯度。对于前向和反向的 combine 组件，我们均将其保留为 BF16 格式，以在训练流程的关键部分保持训练精度。

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 通过 NVLink 互连，而集群中的所有 GPU 通过 IB 实现全互联。为了同时保障在线服务的服务等级目标（SLO）与高吞吐量，我们采用了以下将 *预填充*（*prefilling*）与 *解码*（*decoding*）阶段分离的部署策略。

3.4.1. 预填充阶段

预填充阶段的最小部署单元由 4 个节点（共 32 张 GPU）组成。attention 部分采用 4 路张量并行（TP4）结合序列并行（SP），并与 8 路数据并行（DP8）相结合。其较小的 TP 规模（4）有效限制了 TP 通信的开销。对于 MoE 部分，我们采用 32 路专家并行（EP32），这确保了每个专家都能处理足够大的批次大小，从而提升计算效率。对于 MoE 的 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 token，然后通过 NVLink 在节点内 GPU 之间进行转发。特别地，为了节省 TP 通信开销，我们对浅层中的稠密 MLP 采用 1 路张量并行。

为了实现 MoE 部分不同专家之间的负载均衡，我们需要确保每张 GPU 处理的 token 数量大致相同。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。高负载专家基于在线部署期间收集的统计数据进行检测，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情况，仔细调整节点内 GPU 之间的专家分配，力求在不增加跨节点 All-to-All 通信开销的前提下，尽可能均衡各 GPU 的负载。在 DeepSeek-V3 的部署中，我们为预填充阶段设置了 32 个冗余专家。对于每张 GPU，除了原本承载的 8 个专家外，还将额外承载 1 个冗余专家。

此外，在预填充阶段，为了提高吞吐量并掩盖 All-to-All 和 TP 通信的开销，我们同时处理两个计算负载相似的微批次，将一个微批次的 attention 和 MoE 与另一个微批次的 dispatch 和 combine 进行重叠执行。

最后，我们正在探索专家的 *动态冗余* 策略，即每张 GPU 承载更多专家（例如 16 个），但在每次推理步骤中仅激活 9 个。在每一层的 All-to-All 操作开始之前，我们会动态计算全局最优的路由方案。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. 解码阶段

在解码阶段，我们将共享专家视为一个被路由的专家。从这一角度来看，每个 token 在路由过程中将选择 9 个专家，其中共享专家被视为一个高负载专家，始终会被选中。解码阶段的最小部署单元由 40 个节点（共 320 张 GPU）组成。attention 部分采用 TP4 结合 SP，并与 DP80 相结合，而 MoE 部分则使用 EP320。对于 MoE 部分，每张 GPU 仅承载一个专家，另有 64 张 GPU 专门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执行，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。

与预填充阶段类似，我们根据在线服务的专家负载统计数据，定期（按一定间隔）确定冗余专家集合。但由于每张 GPU 仅承载一个专家，因此我们无需重新调整专家分配。我们也在为解码阶段探索 *动态冗余* 策略。然而，这需要更仔细地优化计算全局最优路由方案的算法，并将其与 dispatch 内核进行融合，以降低开销。

此外，为了提升吞吐量并掩盖 All-to-All 通信的开销，我们也在探索在解码阶段同时处理两个计算负载相似的微批次。与预填充阶段不同，attention 在解码阶段占据了更大的时间比例。因此，我们将一个微批次的 attention 与另一个微批次的 dispatch+MoE+combine 进行重叠执行。在解码阶段，每个专家的批次大小相对较小（通常在 256 个 token 以内），瓶颈在于内存访问而非计算。由于 MoE 部分仅需加载一个专家的参数，内存访问开销极小，因此使用较少的 SM 不会对整体性能产生显著影响。因此，为了避免影响 attention 部分的计算速度，我们可以仅分配一小部分 SM 给 dispatch+MoE+combine。

3.5. 硬件设计建议

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，以掩盖计算过程中的通信延迟。与串行计算和通信相比，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于昂贵的 SM（流式多处理器）（例如，我们在 H800 GPU 可用的 132 个 SM 中分配了 20 个用于此目的），这将限制计算吞吐量。此外，使用 SM 进行通信会导致显著的效率低下，因为 Tensor Core（张量核

心）始终处于严重未充分利用的状态。

目前，SM 主要为 All-to-All 通信执行以下任务：

- **数据转发**：在 IB（InfiniBand）与 NVLink 域之间转发数据，同时聚合来自单个 GPU 且发往同一节点内多个 GPU 的 IB 流量。
- **数据传输**：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**：用于 `all-to-all combine` 阶段。
- **管理细粒度内存布局**：在跨 IB 和 NVLink 域将分块数据传输至多个专家的过程中管理内存布局。

我们期望未来的厂商能够开发硬件，将这些通信任务从宝贵的计算单元 SM 中卸载，使其作为 GPU 协处理器或类似 NVIDIA SHARP Graham et al. (2016) 的网络协处理器运行。此外，为降低应用程序编程复杂度，我们希望该硬件能从计算单元的视角统一 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，即可轻松在整个 IB-NVLink 统一域内完成 `read`、`write`、`multicast` 和 `reduce` 等操作。

3.5.2. 计算硬件

Tensor Core 中更高的 FP8 GEMM 累加精度。 在 NVIDIA Hopper 架构当前的 Tensor Core 实现中，FP8 GEMM 受限于较低的累加精度。在根据最大指数对 32 个尾数乘积进行右移对齐后，Tensor Core 仅使用每个尾数乘积的最高 14 位进行加法运算，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 中以 FP32 精度将 128 次 FP8×FP8 乘法的加法结果累加到寄存器中，部分缓解了该限制。尽管这有助于实现成功的 FP8 训练，但这仅仅是由于 Hopper 架构在 FP8 GEMM 累加精度方面的硬件缺陷而做出的妥协。未来的芯片需要采用更高的精度。

支持 Tile 和 Block 级量化。 当前 GPU 仅支持逐张量（per-tensor）量化，缺乏对我们这种 Tile 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 Core 复制到 CUDA Core，乘以缩放因子，并累加到 CUDA Core 上的 FP32 寄存器中。尽管结合我们精确的 FP32 累加策略显著缓解了反量化开销，但 Tensor Core 与 CUDA Core 之间频繁的数据移动仍然限制了计算效率。因此，我们建议未来的芯片通过使 Tensor Core 能够接收缩放因子并实现带组缩放的 MMA 操作，来支持细粒度量化。这样一来，整个部分和累加与反量化过程均可直接在 Tensor Core 内部完成，直至生成最终结果，从而避免频繁的数据移动。

支持在线量化。 尽管我们的研究证明了在线量化的有效性，但当前实现仍难以有效支持它。在现有流程中，我们需要从 HBM（高带宽内存）读取 128 个 BF16 激活值（前一步计算的输出）进行量化，量化后的 FP8 值随后被写回 HBM，仅为了在 MMA 操作时再次读取。为解决这一低效问题，我们建议未来的芯片将 FP8 类型转换与 TMA（张量内存加速器）访问融合为单一

操作，以便在激活值从全局内存传输到共享内存的过程中完成量化，避免频繁的内存读写。我们还建议支持 Warp 级类型转换指令以加速，这将进一步促进层归一化与 FP8 类型转换的更好融合。或者，可以采用近存计算方案，将计算逻辑放置在 HBM 附近。在这种情况下，BF16 元素在从 HBM 读入 GPU 时可直接转换为 FP8，从而将片外内存访问减少约 50%。

支持转置 GEMM 操作。 当前架构使得将矩阵转置与 GEMM 操作融合变得繁琐。在我们的工作流中，前向传播期间的激活值被量化为 `1x128` 的 FP8 Tile 并存储。在后向传播期间，该矩阵需要被读出、反量化、转置、重新量化为 `128x1` 的 Tile，并存储回 HBM。为减少内存操作，我们建议未来的芯片在 MMA 操作前，支持从共享内存直接读取矩阵的转置形式，适用于训练和推理均需要的精度。结合 FP8 格式转换与 TMA 访问的融合，这一增强将显著简化量化工作流。

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至英语和中文之外。此外，我们优化了数据处理流程，在保持语料多样性的同时最大限度地减少冗余。受 Ding et al. (2024) 的启发，我们采用了文档打包方法以保障数据完整性，但在训练过程中并未引入跨样本注意力掩码。最终，DeepSeek-V3 的训练语料库由我们分词器下的 14.8T 高质量、多样化 token 组成。

在 DeepSeekCoder-V2 (DeepSeek-AI, 2024a) 的训练过程中，我们观察到中间填充（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，并未损害其下一个 token 的预测能力。与 DeepSeekCoder-V2 保持一致，我们在 DeepSeek-V3 的预训练中也引入了 FIM 策略。具体而言，我们采用前缀-后缀-中间（Prefix-Suffix-Middle, PSM）框架对数据进行如下结构化处理：

$$\texttt{<|fim\_begin|>}f_{\text{pre}}\texttt{<|fim\_hole|>}f_{\text{suf}}\texttt{<|fim\_end|>}f_{\text{middle}}\texttt{<|eos\_token|>}.$$

该结构在文档级别应用，作为预打包流程的一部分。FIM 策略的应用比例为 0.1，与 PSM 框架保持一致。

DeepSeek-V3 的分词器采用基于字节的 BPE (Shibata et al., 1999)，词表扩展至 128K 个 token。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将标点符号与换行符结合的 token。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），这一技巧可能会引入 token 边界偏差 (Lundberg, 2023)。为解决该问题，我们在训练期间随机拆分一定比例的此类组合 token，使模型接触到更广泛的特殊情况，从而缓解这种偏差。

4.2. 超参数

模型超参数。 我们将 Transformer 层数设置为 61，隐藏层维度设置为 7168。所有可学习参数均以标准差 0.006 进行随机初始化。在 MLA 中，我们将注意力头数 n_h 设置为 128，每个头的维度 d_h 设置为 128。KV 压缩维度 d_c 设置为 512，查询压缩维度 d'_c 设置为 1536。对于解耦的查询和键，我们将每个头的维度 d^R_h 设置为 64。除前三层外，我们将所有前馈神经网络（FFN）替换为混合专家（MoE）层。每个 MoE 层包含 1 个共享专家和 256 个路由专家，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 最多被发送到 4 个节点。多 token 预测（MTP）深度 D 设置为 1，即除了精确的下一个 token 外，每个 token 还将额外预测一个 token。与 DeepSeek-V2 类似，DeepSeek-V3 也在压缩后的潜在向量后采用了额外的 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 包含 671B 总参数，其中每个 token 激活 37B 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 $\beta_1 = 0.9$，$\beta_2 = 0.95$，以及 $\text{weight\_decay} = 0.1$。预训练期间，我们将最大序列长度设置为 4K，并在 14.8T 个 token 上对 DeepSeek-V3 进行预训练。关于学习率调度，我们首先在前 2K 步中将其从 0 线性增加至 2.2×10^{-4}。随后，我们保持 2.2×10^{-4} 的恒定学习率，直到模型消耗完 10T 训练 token。接着，我们遵循余弦衰减曲线，在 4.3T 个 token 的训练过程中将学习率逐渐衰减至 2.2×10^{-5}。在最后 500B 个 token 的训练期间，我们在前 333B 个 token 中保持 2.2×10^{-5} 的恒定学习率，并在剩余的 167B 个 token 中切换至 7.3×10^{-6} 的恒定学习率。梯度裁剪范数设置为 1.0。我们采用批次大小调度策略，在前 469B 个 token 的训练中，批次大小从 3072 逐渐增加至 15360，并在后续训练中保持 15360。我们利用流水线并行将模型的不同层部署在不同的 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。对于节点限制路由，每个 token 最多被发送到 4 个节点（即 $M = 4$）。对于无辅助损失的负载均衡，我们将前 14.3T 个 token 的偏置更新速度 γ 设置为 0.001，剩余 500B 个 token 设置为 0.0。对于平衡损失，我们将 α 设置为 0.0001，仅用于避免任何单个序列内出现极端不平衡。MTP 损失权重 λ 在前 10T 个 token 中设置为 0.3，在剩余的 4.8T 个 token 中设置为 0.1。

4.3. 长上下文扩展

我们采用了与 DeepSeek-V2 (DeepSeek-AI, 2024c) 类似的方法，为 DeepSeek-V3 赋予长上下文能力。在预训练阶段之后，我们应用 YaRN (Peng et al., 2023a) 进行上下文扩展，并执行两个额外的训练阶段，每个阶段包含 1000 步，以逐步将上下文窗口从 4K 扩展至 32K，再扩展至 128K。YaRN 的配置与 DeepSeek-V2 中使用的保持一致，仅应用于解耦的共享键 $\mathbf{k}^R_t$。两个阶段的超参数保持相同，其中尺度 $s = 40$，$\alpha = 1$，$\beta = 32$，缩放因子 $\sqrt{t} = 0.1 \ln s + 1$。在第一阶段，序列长度设置为 32K，批次大小为 1920。在第二阶段，序列长度增加至 128K，批次大小降低至 480。两个阶段的学习率均设置为 7.3×10^{-6}，与预训练阶段的最终学习率保持一致。

通过这种两阶段扩展训练，DeepSeek-V3 能够处理长达 128K 的输入，同时保持强大的性

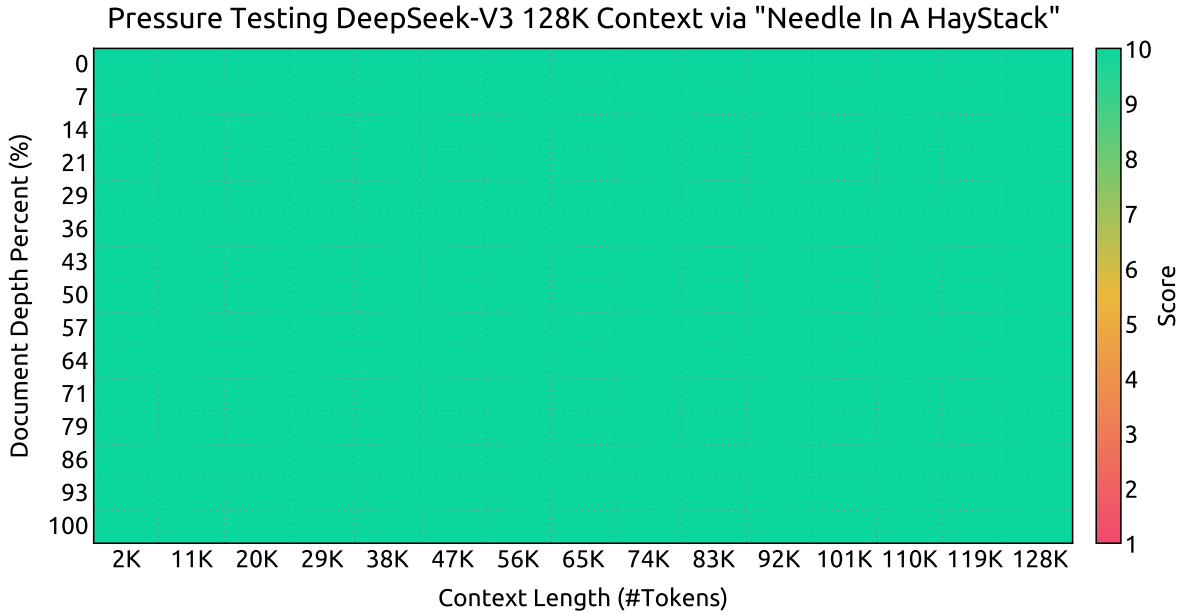


图 8 | 在 “大海捞针”（Needle In A Haystack, NIAH）测试上的评估结果。DeepSeek-V3 在所有高达 128K 的上下文窗口长度上均表现良好。

能。图 8 表明，DeepSeek-V3 在经过监督微调后，在 “大海捞针”（Needle In A Haystack, NIAH）测试中取得了显著的性能，在高达 128K 的上下文窗口长度上均表现出一致的鲁棒性。

4.4. 评估

4.4.1. 评估基准

DeepSeek-V3 的基础模型是在以英语和中文为主的多语言语料库上进行预训练的，因此我们在一系列主要以英语和中文为主的基准测试以及一个多语言基准测试上评估其性能。我们的评估基于集成在 HAI-LLM 框架中的内部评估框架。考虑的基准测试分类如下，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)。

指代消解 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)。需要注意的是，AGIEval 同时包含英语和中文子集。

遵循我们之前的工作 (DeepSeek-AI, 2024b,c)，我们对包括 HellaSwag、PIQA、WinoGrande、RACE-Middle、RACE-High、MMLU、MMLU-Redux、MMLU-Pro、MMMLU、ARC-Easy、ARC-Challenge、C-Eval、CMMLU、C3 和 CCPM 在内的数据集采用基于困惑度的评估方法，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方法。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（Bits-Per-Byte, BPB）作为指标，以保证在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 的基础模型与最先进的开源基础模型进行了对比，包括 DeepSeek-V2-Base (DeepSeek-AI, 2024c)（我们此前发布的版本）、Qwen2.5 72B Base (Qwen, 2024b) 以及 LLaMA-3.1 405B Base (AI@Meta, 2024b)。我们使用内部评估框架对所有模型进行评估，并确保它们采用相同的评估设置。需要注意的是，由于过去几个月评估框架的更新，DeepSeek-V2-Base 的性能与我们此前报告的结果存在细微差异。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越了 LLaMA-3.1 405B Base，实质上已成为最强的开源模型。

从更细致的角度来看，我们将 DeepSeek-V3-Base 与其他开源基础模型逐一进行了对比。(1) 与 DeepSeek-V2-Base 相比，得益于模型架构的改进、模型规模与训练 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优的性能。(2) 与最先进的中文开源模型 Qwen2.5 72B Base 相比，尽管激活参数仅为其一半，DeepSeek-V3-Base 仍展现出显著优势，尤其在英语、多语言、代码和数学基准测试上。在中文基准测试方面，除 CMMLU（一项中文多学科多项选择题任务）外，DeepSeek-V3-Base 的表现也优于 Qwen2.5 72B。(3) 与参数量最大的开源模型 LLaMA-3.1 405B Base（其激活参数是 DeepSeek-V3-Base 的 11 倍）相比，DeepSeek-V3-Base 在多语言、代码和数学基准测试上同样表现出更优的性能。在英语和中文语言基准测试方面，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。

得益于高效的架构设计与全面的工程优化，DeepSeek-V3 实现了极高的训练效率。在我们

	基准测试 (指标)	样本数	DeepSeek-V2 基座	Qwen2.5 72B 基座	LLaMA-3.1 405B 基座	DeepSeek-V3 基座
	架构	-	MoE	Dense	Dense	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**	0.548
	BBH (EM)	3-shot	78.8	79.8	82.9	**87.5**
	MMLU (EM)	5-shot	78.4	85.0	84.4	**87.1**
	MMLU-Redux (EM)	5-shot	75.6	83.2	81.3	**86.2**
	MMLU-Pro (EM)	5-shot	51.4	58.3	52.8	**64.4**
	DROP (F1)	3-shot	80.4	80.6	86.0	**89.0**
	ARC-Easy (EM)	25-shot	97.6	98.4	98.4	**98.9**
	ARC-Challenge (EM)	25-shot	92.2	94.5	**95.3**	**95.3**
	HellaSwag (EM)	10-shot	87.1	84.8	**89.2**	**88.9**
	PIQA (EM)	0-shot	83.9	82.6	**85.9**	84.7
	WinoGrande (EM)	5-shot	**86.3**	82.3	85.2	84.9
	RACE-Middle (EM)	5-shot	73.1	68.1	**74.2**	67.1
	RACE-High (EM)	5-shot	52.6	50.3	**56.8**	51.3
	TriviaQA (EM)	5-shot	80.0	71.9	**82.7**	**82.9**
	NaturalQuestions (EM)	5-shot	38.6	33.2	**41.5**	40.0
	AGIEval (EM)	0-shot	57.5	75.8	60.6	**79.6**
代码	HumanEval (Pass@1)	0-shot	43.3	53.0	54.9	**65.2**
	MBPP (Pass@1)	3-shot	65.0	72.6	68.4	**75.4**
	LiveCodeBench-Base (Pass@1)	3-shot	11.6	12.9	15.5	**19.4**
	CRUXEval-I (EM)	2-shot	52.5	59.1	58.5	**67.3**
	CRUXEval-O (EM)	2-shot	49.8	59.9	59.9	**69.8**
数学	GSM8K (EM)	8-shot	81.6	88.3	83.5	**89.3**
	MATH (EM)	4-shot	43.4	54.4	49.0	**61.6**
	MGSM (EM)	8-shot	63.6	76.2	69.9	**79.8**
	CMath (EM)	3-shot	78.7	84.5	77.3	**90.7**
中文	CLUEWSC (EM)	5-shot	82.0	82.5	**83.0**	**82.7**
	C-Eval (EM)	5-shot	81.4	89.2	72.5	**90.1**
	CMMLU (EM)	5-shot	84.0	**89.5**	73.7	88.8
	CMRC (EM)	1-shot	**77.4**	75.8	76.0	76.3
	C3 (EM)	0-shot	77.4	76.7	**79.7**	78.6
	CCPM (EM)	0-shot	**93.0**	88.5	78.6	92.0
多语言	MMMLU-non-English (EM)	5-shot	64.0	74.8	73.8	**79.4**

表 3 | DeepSeek-V3-Base 与其他代表性开源基座模型的对比。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。分差未超过 0.3 的分数视为处于同一水平。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。

的训练框架与基础设施下，训练 DeepSeek-V3 每万亿 token 仅需 18 万 H800 GPU 小时，其成本远低于训练 72B 或 405B 稠密模型。

4.5. 讨论

4.5.1. 多令牌预测的消融实验

表 4 展示了 MTP 策略的消融实验结果。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。在较小规模下，我们在 1.33T tokens 上训练了一个包含 15.7B 总参数的基线 MoE 模型。在较大规模下，我们在 540B tokens 上训练了一个包含 228.7B 总参数的基线 MoE 模型。

基准测试 (指标)	# 提示数	小型 MoE 基线	小型 MoE w/ MTP	大型 MoE 基线	大型 MoE w/ MTP
# 激活参数量 (推理)	-	2.4B	2.4B	20.9B	20.9B
# 总参数量 (推理)	-	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-shot	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均能持续提升模型性能。

在此基础上，保持训练数据和其他架构不变，我们为其附加了一个深度为 1 的 MTP 模块，并训练了两个采用 MTP 策略的模型以供对比。需要注意的是，在推理阶段，我们直接丢弃了 MTP 模块，因此对比模型的推理成本完全相同。从表中可以看出，MTP 策略在大多数评估基准上均能持续提升模型性能。

4.5.2. 无辅助损失平衡策略的消融实验

表 5 展示了无辅助损失平衡策略的消融实验结果。我们在不同规模的两个基线模型上验证了该策略。在较小规模下，我们在 1.33T tokens 上训练了一个包含 15.7B 总参数的基线 MoE 模型。在较大规模下，我们在 578B tokens 上训练了一个包含 228.7B 总参数的基线 MoE 模型。这两个基线模型均纯粹使用辅助损失来促进负载均衡，并采用带有 Top-K 亲和度归一化的 sigmoid 门控函数。它们控制辅助损失强度的超参数分别与 DeepSeek-V2-Lite 和 DeepSeek-V2 保持一致。在这两个基线模型的基础上，保持训练数据和其他架构不变，我们移除了所有辅助损失并引入了无辅助损失平衡策略以供对比。从表中可以看出，无辅助损失策略在大多数评估基准上均能取得更优的模型性能。

4.5.3. 批次级负载均衡与序列级负载均衡

无辅助损失平衡与序列级辅助损失的关键区别在于其平衡范围：批次级与序列级。与序列级辅助损失相比，批次级平衡施加了更灵活的约束，因为它不强制要求每个序列内部实现域内平衡。这种灵活性使得专家能够更好地专注于不同领域。为了验证这一点，我们记录并分析了基于辅助损失的 16B 基线模型和无辅助损失 16B 模型在 Pile 测试集不同领域上的专家负载。如图 9 所示，我们观察到无辅助损失模型正如预期那样展现出更强的专家专业化模式。

为了进一步探究这种灵活性与模型性能优势之间的关联，我们额外设计并验证了一种批次级辅助损失，该损失鼓励在每个训练批次上实现负载平衡，而不是在每个序列上。实验结果表明，在达到相似水平的批次级负载平衡时，批次级辅助损失也能取得与无辅助损失方法相近的

基准测试（指标）	提示数	小型 MoE 基于辅助损失	小型 MoE 无辅助损失	大型 MoE 基于辅助损失	大型 MoE 无辅助损失
# 激活参数	-	2.4B	2.4B	20.9B	20.9B
# 总参数	-	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-shot	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实验结果。与纯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均能持续提升模型性能。

模型性能。具体而言，在 1B MoE 模型的实验中，验证损失分别为：2.258（使用序列级辅助损失）、2.253（使用无辅助损失方法）和 2.253（使用批次级辅助损失）。我们在 3B MoE 模型上也观察到了类似的结果：使用序列级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批次级辅助损失的模型均取得了相同的验证损失 2.080。

此外，尽管批次级负载均衡方法表现出一致的性能优势，但它们也面临两个潜在的效率挑战：（1）某些序列或小批次内的负载不平衡，以及（2）推理过程中由域偏移引起的负载不平衡。第一个挑战通过我们采用大规模专家并行和数据并行的训练框架自然得到解决，该框架保证了每个微批次的大小足够大。针对第二个挑战，我们还设计并实现了一个带有冗余专家部署的高效推理框架（如第 3.4 节所述），以克服该问题。

5. 后训练

5.1. 监督微调

我们精心策划了指令微调数据集，包含 150 万条跨越多个领域的实例，每个领域均采用针对其特定需求定制的数据构建方法。

推理数据。 对于推理相关的数据集（包括专注于数学、代码竞赛问题和逻辑谜题的数据集），我们利用内部 DeepSeek-R1 模型生成数据。具体而言，尽管 R1 生成的数据表现出较高的准确性，但也存在过度思考、格式不佳和篇幅过长等问题。我们的目标是在 R1 生成推理数据的高准确性与标准格式推理数据的清晰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）的训练流程，开发针对特定领域（如代码、数学或通用推理）的专家模型。该专家模型作为最终模型的数据生成器。训练过程为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配对，格

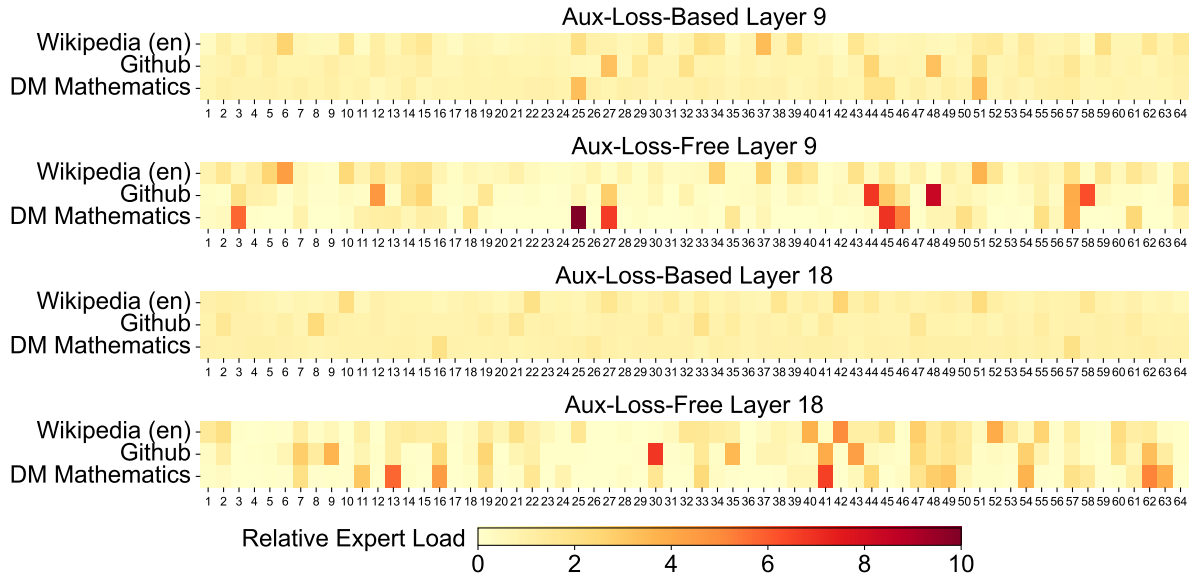


图 9 | 无辅助损失模型与基于辅助损失模型在 Pile 测试集三个领域上的专家负载。无辅助损失模型比基于辅助损失的模型展现出更强的专家专业化模式。相对专家负载表示实际专家负载与理论平衡专家负载之间的比率。受篇幅限制，我们仅以两层的结果为例进行展示，所有层的结果详见附录 C。

式为 <问题, 原始回答>；第二种则在问题和 R1 回答的基础上加入系统提示词，格式为 <系统提示词, 问题, R1 回答>。

系统提示词经过精心设计，包含引导模型生成富含反思与验证机制回答的指令。在 RL 阶段，模型利用高温采样生成融合 R1 生成数据与原始数据模式的回答，即使在没有显式系统提示词的情况下也是如此。经过数百步 RL 训练后，中间 RL 模型学会了融入 R1 模式，从而战略性地提升整体性能。

完成 RL 训练阶段后，我们采用拒绝采样来为最终模型筛选高质量的 SFT 数据，其中专家模型被用作数据生成源。该方法确保了最终训练数据在保留 DeepSeek-R1 优势的同时，能够生成简洁有效的回答。

非推理数据。 对于非推理数据（如创意写作、角色扮演和简单问答），我们利用 DeepSeek-V2.5 生成回答，并邀请人工标注员验证数据的准确性与正确性。

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策略，从 5×10^{-6} 开始逐渐降低至 1×10^{-6}。在训练过程中，每个单一序列由多个样本打包而成。然而，我们采用样本掩码策略，以确保这些示例保持隔离且互不可见。

5.2. 强化学习

5.2.1. 奖励模型

在我们的强化学习（RL）过程中，我们采用了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进行验证的问题，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，我们要求模型在指定格式（如方框内）中提供最终答案，以便我们应用规则来验证其正确性。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，我们确保了更高的可靠性，因为这种方法能够抵抗操纵或利用。

基于模型的 RM。 对于具有自由格式标准答案的问题，我们依赖奖励模型来判断回复是否与预期的标准答案匹配。相反，对于没有明确标准答案的问题（如涉及创意写作的问题），奖励模型的任务是根据输入的问题和对应答案提供反馈。该奖励模型基于 DeepSeek-V3 的 SFT 检查点训练而成。为了增强其可靠性，我们构建了偏好数据，这些数据不仅提供最终奖励，还包含推导出该奖励的思维链（chain-of-thought）。该方法有助于降低在特定任务中出现奖励操纵（reward hacking）的风险。

5.2.2. 组相对策略优化

与 DeepSeek-V2 (DeepSeek-AI, 2024c) 类似，我们采用了组相对策略优化（Group Relative Policy Optimization, GRPO）(Shao et al., 2024)。该方法摒弃了通常与策略模型规模相同的评论家（critic）模型，转而根据组得分来估计基线。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 $\{o_1, o_2, \cdots, o_G\}$，然后通过最大化以下目标函数来优化策略模型 π_θ：

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

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

其中 ε 和 β 为超参数；π_{ref} 为参考模型；A_i 为优势值（advantage），由组内各输出对应的奖励 $\{r_1, r_2, \ldots, r_G\}$ 推导得出：

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

在 RL 过程中，我们引入了来自编程、数学、写作、角色扮演和问答等多样化领域的提示词。该方法不仅使模型更紧密地契合人类偏好，还提升了在基准测试上的性能，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

评估基准。 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)（2024 年 8 月至 11 月的问题）、Codeforces[2]、全国高中数学联赛（CNMO 2024）[3] 以及美国数学邀请赛 2024（AIME 2024）(MAA, 2024) 上对指令模型进行了进一步评估。

对比基线。 我们将我们的对话模型与多个强基线模型进行了全面对比评估，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，我们选择了最具代表性的变体进行对比。对于闭源模型，评估通过其各自的 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，我们采用了 simple-evals 框架 [4] 中的评估提示词。在零样本（zero-shot）设置下，我们对 MMLU-Redux 使用了 Zero-Eval 提示词格式 (Lin, 2024)。对于其他数据集，我们遵循其原始评估协议，并使用数据集创建者提供的默认提示词。在代码和数学基准测试中，HumanEval-Mul 数据集共包含 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，该数据集收集自 2024 年 8 月至 11 月。Codeforces 数据集的评估采用击败参赛者的百分比作为衡量标准。SWE-Bench Verified 使用 agentless 框架 (Xia et al., 2024) 进行评估。我们使用 “diff” 格式评估与 Aider 相关的基准测试。在数学评估中，AIME 和 CNMO 2024 的评估温度设置为 0.7，结果取 16 次运行的平均值；而 MATH-500 采用贪婪解码。我们允许所有模型在每个基准测试中最多输出 8192 个 token。

5.3.2. 标准评估

表 6 展示了评估结果，表明 DeepSeek-V3 是表现最佳的开源模型。此外，它与 GPT-4o 和 Claude-3.5-Sonnet 等前沿闭源模型相比也具备强劲的竞争力。

英语基准测试。 MMLU 是一个广受认可的基准测试，旨在评估大语言模型在多样化知识领域和任务中的表现。DeepSeek-V3 展现出极具竞争力的性能，与 LLaMA-3.1-405B、GPT-4o 和 Claude-Sonnet 3.5 等顶级模型并驾齐驱，同时显著优于 Qwen2.5 72B。此外，在更具挑战性的教

[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

	基准测试 (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	Dense	Dense	-	-	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 (Correct)	9.0	10.2	9.1	17.1	28.4	**38.2**	24.9
	FRAMES (Acc.)	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**
代码	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 (Percentile)	17.5	35.6	24.8	25.3	20.3	23.6	**51.6**
	SWE Verified (Resolved)	-	22.6	23.8	24.5	**50.8**	38.8	42.0
	Aider-Edit (Acc.)	60.3	71.6	65.4	63.9	**84.2**	72.9	79.7
	Aider-Polyglot (Acc.)	-	18.2	7.6	5.8	45.3	16.0	**49.6**
数学	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 (Correct)	48.5	54.1	48.4	50.4	51.3	59.3	**64.8**

表 6 | DeepSeek-V3 与其他代表性对话模型的对比。所有模型均在限制输出长度为 8K 的配置下进行评估。对于样本数少于 1000 的基准测试，我们采用不同的温度设置进行多次测试以获取稳健的最终结果。DeepSeek-V3 是目前性能最佳的开源模型，同时在与前沿闭源模型的对比中也展现出极具竞争力的表现。

育知识基准测试 MMLU-Pro 中，DeepSeek-V3 表现优异，成绩紧随 Claude-Sonnet 3.5 之后。在修正了标签的 MMLU 优化版本 MMLU-Redux 上，DeepSeek-V3 超越了其同类模型。此外，在博士级评估测试平台 GPQA-Diamond 上，DeepSeek-V3 取得了显著成果，排名仅次于 Claude 3.5 Sonnet，并以较大优势领先于所有其他竞争模型。

在 DROP、LongBench v2 和 FRAMES 等长上下文理解基准测试中，DeepSeek-V3 继续展现出其顶级模型的地位。在 DROP 的 3-shot 设置下，它取得了令人印象深刻的 91.6 F1 分数，优于该类别中的所有其他模型。在 FRAMES（一个要求在 10 万 token 上下文中进行问答的基准测试）上，DeepSeek-V3 紧随 GPT-4o 之后，同时以显著优势领先于所有其他模型。这证明了 DeepSeek-V3 在处理极长上下文任务方面的强大能力。DeepSeek-V3 在 LongBench v2 上取得的同类最佳表现进一步验证了其长上下文能力，该数据集发布于 DeepSeek V3 发布前仅几周。在事实知识基准测试 SimpleQA 上，DeepSeek-V3 落后于 GPT-4o 和 Claude-Sonnet，这主要归因于其设计侧重点和资源分配策略。DeepSeek-V3 分配了更多的训练 token 用于学习中文知识，从而在 C-SimpleQA 上取得了卓越的性能。在指令遵循基准测试中，DeepSeek-V3 显著优于其前身 DeepSeek-V2 系列，凸显了其在理解和遵循用户自定义格式约束方面能力的提升。

Model	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 | 英语开放式对话评估。对于 AlpacaEval 2.0，我们采用长度控制胜率作为评估指标。

代码与数学基准测试。 编程是 LLM 面临的一项具有挑战性且实用的任务，涵盖了以工程为导向的任务（如 SWE-Bench-Verified 和 Aider），以及算法类任务（如 HumanEval 和 LiveCodeBench）。在工程类任务中，DeepSeek-V3 落后于 Claude-Sonnet-3.5-1022，但显著优于开源模型。开源的 DeepSeek-V3 有望推动编程相关工程任务的进步。通过开放其强大的能力，DeepSeek-V3 能够推动软件工程、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程任务中的能力边界。在算法类任务中，DeepSeek-V3 展现出卓越的性能，在 HumanEval-Mul 和 LiveCodeBench 等基准测试中超越了所有基线模型。这一成功可归因于其先进的知识蒸馏技术，该技术有效提升了其在算法导向任务中的代码生成与问题解决能力。

在数学基准测试中，DeepSeek-V3 表现出卓越的性能，显著超越基线模型，并为非 o1 类模型树立了新的最先进水平（SOTA）。具体而言，在 AIME、MATH-500 和 CNMO 2024 上，DeepSeek-V3 的绝对得分比排名第二的模型 Qwen2.5 72B 高出约 10%，对于此类极具挑战性的基准测试而言，这是一个相当大的优势。这一卓越能力凸显了源自 DeepSeek-R1 的蒸馏技术的有效性，该技术已被证明对非 o1 类模型极具益处。

中文基准测试。 Qwen 和 DeepSeek 是两个在中文和英文方面均具备强大支持的代表性模型系列。在事实基准测试中文 SimpleQA 上，DeepSeek-V3 以 16.4 分的优势超越 Qwen2.5-72B，尽管 Qwen2.5 是在包含 18T token 的更大语料库上训练的，该规模比 DeepSeek-V3 预训练使用的 14.8T token 多出 20%。

在代表中文教育知识评估的基准测试 C-Eval 以及 CLUEWSC（中文 Winograd 模式挑战）上，DeepSeek-V3 与 Qwen2.5-72B 表现出相近的性能水平，表明两款模型均针对具有挑战性的中文推理和教育任务进行了充分优化。

5.3.3. 开放式评估

除了标准基准测试外，我们还使用大语言模型作为裁判，在开放式生成任务上评估了我们的模型，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，利用 GPT-4-Turbo-1106 作为裁判进行成对比较。在 Arena-Hard 基准上，DeepSeek-V3 相对于基线模型 GPT-4-0314 取得了超过 86% 的惊人胜率，其表现与 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

模型	对话	困难对话	安全性	推理	平均
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

表 8 | GPT-4o、Claude-3.5-sonnet 和 DeepSeek-V3 在 RewardBench 上的性能表现。

复杂提示（包括代码编写和调试任务）方面。此外，DeepSeek-V3 取得了突破性进展，成为首个在 Arena-Hard 基准上得分超过 85% 的开源模型。这一成就显著缩小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所能达到的水平树立了新标准。

同样地，DeepSeek-V3 在 AlpacaEval 2.0 上展现出卓越的性能，超越了闭源和开源模型。这证明了其在写作任务和处理简单问答场景方面的出色能力。值得注意的是，它以 20% 的显著优势超越了 DeepSeek-V2.5-0905，凸显了其在处理简单任务方面的实质性改进，并展示了其技术升级的有效性。

5.3.4. DeepSeek-V3 作为生成式奖励模型

我们将 DeepSeek-V3 的判断能力与最先进的模型（即 GPT-4o 和 Claude-3.5）进行了比较。表 8 展示了这些模型在 RewardBench (Lambert et al., 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，同时超越了其他版本。此外，DeepSeek-V3 的判断能力也可以通过投票技术得到进一步提升。因此，我们结合投票技术使用 DeepSeek-V3 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鲁棒性。

5.4. 讨论

5.4.1. 基于 *DeepSeek-R1* 的蒸馏

我们基于 DeepSeek-V2.5 对来自 DeepSeek-R1 的蒸馏贡献进行了消融实验。基线模型在短思维链（CoT）数据上进行训练，而对比模型则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 基准测试上均取得了显著提升。我们的实验揭示了一个有趣的权衡：蒸馏虽然带来了更好的性能，但也大幅增加了平均回复长度。为了在模型准确性和计算效率之间保持平衡，我们为 DeepSeek-V3 的蒸馏过程仔细选择了最优设置。

我们的研究表明，从推理模型中进行知识蒸馏为后训练优化提供了一个有前景的方向。尽管我们目前的工作主要集中在数学和代码领域的蒸馏数据上，但该方法在更广泛的任务领域中展现出应用潜力。在这些特定领域展现出的有效性表明，长思维链（CoT）蒸馏对于提升其他需

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

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

要复杂推理的认知任务中的模型性能可能具有重要价值。在不同领域进一步探索该方法仍是未来研究的重要方向。

5.4.2. 自我奖励

奖励在强化学习（RL）中起着至关重要的作用，引导着优化过程。在通过外部工具进行验证较为直接的领域（如某些编程或数学场景）中，强化学习展现出了卓越的效能。然而，在更广泛的场景中，通过硬编码构建反馈机制是不切实际的。在 DeepSeek-V3 的开发过程中，针对这些更广泛的上下文，我们采用了宪法 AI 方法 (Bai et al., 2022)，利用 DeepSeek-V3 自身的投票评估结果作为反馈来源。该方法产生了显著的对齐效果，大幅提升了 DeepSeek-V3 在主观评估中的表现。通过整合额外的宪法输入，DeepSeek-V3 能够朝着宪法指引的方向进行优化。我们认为，这种将补充信息与大型语言模型（LLM）结合作为反馈来源的范式至关重要。大型语言模型充当了多功能处理器，能够将来自不同场景的非结构化信息转化为奖励，最终促进大型语言模型的自我改进。除了自我奖励机制外，我们还致力于探索其他通用且可扩展的奖励方法，以持续提升模型在通用场景中的能力。

5.4.3. 多 *Token* 预测评估

与仅预测下一个单一 token 不同，DeepSeek-V3 通过 MTP 技术预测接下来的 2 个 token。结合投机解码框架 (Leviathan et al., 2023; Xia et al., 2023)，它能够显著加速模型的解码速度。一个自然的问题是，额外预测的 token 的接受率如何。根据我们的评估，在各种生成主题中，第二个 token 预测的接受率在 85% 到 90% 之间，表现出一致的可靠性。这一高接受率使 DeepSeek-V3 实现了显著提升的解码速度，达到了 1.8 倍的 TPS（每秒 Token 数）。

6. 结论、局限性与未来方向

在本文中，我们介绍了 DeepSeek-V3，这是一个大型混合专家（MoE）语言模型，总参数量为 671B，激活参数量为 37B，在 14.8T tokens 上进行训练。除了 MLA 和 DeepSeekMoE 架构外，它还首创了一种无辅助损失的负载均衡策略，并设定了多 token 预测训练目标以实现更强的性能。得益于 FP8 训练的支持和细致的工程优化，DeepSeek-V3 的训练具有极高的成本效益。后训练阶段也成功从 DeepSeek-R1 系列模型中蒸馏了推理能力。综合评估表明，DeepSeek-V3 已成为目前最强的开源模型，并达到了与 GPT-4o 和 Claude-3.5-Sonnet 等领先闭源模型相媲美的性能。尽管性能强劲，它仍保持了经济实惠的训练成本。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 H800 GPU 小时。

在肯定其强劲性能和成本效益的同时，我们也认识到 DeepSeek-V3 存在一些局限性，尤其是在部署方面。首先，为了确保高效的推理，DeepSeek-V3 的推荐部署单元相对较大，这可能会给小型团队带来负担。其次，尽管我们为 DeepSeek-V3 制定的部署策略已实现超过 DeepSeek-V2 两倍的端到端生成速度，但仍存在进一步提升的潜力。幸运的是，随着更先进硬件的发展，这些局限性有望得到自然解决。

DeepSeek 始终秉持长期主义，坚持开源模型的发展路线，旨在稳步迈向通用人工智能（AGI）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。

- 我们将持续研究并优化模型架构，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 的架构局限，从而拓展其建模能力的边界。
- 我们将持续迭代训练数据的数量与质量，并探索引入额外的训练信号源，旨在推动数据在更广泛维度上的扩展。
- 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提升其智能水平与问题解决能力。
- 我们将探索更全面、多维度的模型评估方法，以避免在研究过程中出现针对固定基准进行优化的倾向，从而防止对模型能力产生误导性认知并影响我们的基础评估。

参考文献

AI@Meta. Llama 3 model card, 2024a. URL https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

AI@Meta. Llama 3.1 model card, 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, et al. Program synthesis with large language models. arXiv preprint arXiv:2108.07732, 2021.

Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirhoseini, C. McKinnon, et al. Constitutional AI: Harmlessness from AI feedback. arXiv preprint 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, and J. Li. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. arXiv preprint arXiv:2412.15204, 2024.

M. Bauer, S. Treichler, and A. Aiken. Singe: leveraging warp specialization for high performance on GPUs. In Proceedings of the 19th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPoPP '14, page 119 – 130, New York, NY, USA, 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, and Y. Choi. PIQA: reasoning about physical commonsense in natural language. In The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, pages 7432–7439. 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, and W. Zaremba. Evaluating large language models trained on code. CoRR, abs/2107.03374, 2021. URL https://arxiv.org/abs/2107.03374.

P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. Think you have solved question answering? try arc, the AI2 reasoning challenge. 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, et al. Training verifiers to solve math word problems. arXiv preprint arXiv:2110.14168, 2021.

Y. Cui, T. Liu, W. Che, L. Xiao, Z. Chen, W. Ma, S. Wang, and G. Hu. A span-extraction dataset for Chinese machine reading comprehension. In K. Inui, J. Jiang, V. Ng, and X. Wan, editors, Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 5883–5889, Hong Kong, China, Nov. 2019. 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, and W. Liang. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. CoRR, abs/2401.06066, 2024. URL https://doi.org/10.48550/arXiv.2401.06066.

DeepSeek-AI. Deepseek-coder-v2: Breaking the barrier of closed-source models in code intelligence. CoRR, abs/2406.11931, 2024a. URL https://doi.org/10.48550/arXiv.2406.11931.

DeepSeek-AI. Deepseek LLM: scaling open-source language models with longtermism. CoRR, abs/2401.02954, 2024b. URL https://doi.org/10.48550/arXiv.2401.02954.

DeepSeek-AI. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. CoRR, abs/2405.04434, 2024c. URL https://doi.org/10.48550/arXiv.2405.04434.

T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. Advances in Neural Information Processing Systems, 35: 30318–30332, 2022.

H. Ding, Z. Wang, G. Paolini, V. Kumar, A. Deoras, D. Roth, and S. Soatto. Fewer truncations improve language modeling. arXiv preprint arXiv:2404.10830, 2024.

D. Dua, Y. Wang, P. Dasigi, G. Stanovsky, S. Singh, and M. Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In J. Burstein, C. Doran, and T. Solorio, editors, Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), pages 2368–2378. 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, and T. B. Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators. arXiv preprint arXiv:2404.04475, 2024.

W. Fedus, B. Zoph, and N. Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. CoRR, abs/2101.03961, 2021. URL https://arxiv.org/abs/2101.03961.

M. Fishman, B. Chmiel, R. Banner, and D. Soudry. Scaling FP8 training to trillion-token llms. arXiv preprint arXiv:2409.12517, 2024.

E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. arXiv preprint arXiv:2210.17323, 2022.

L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima, et al. The Pile: An 800GB dataset of diverse text for language modeling. arXiv preprint 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, and P. Minervini. Are we done with mmlu? CoRR, abs/2406.04127, 2024. URL https://doi.org/10.48550/arXiv.2406.04127.

F. Gloeckle, B. Y. Idrissi, B. Rozière, D. Lopez-Paz, and G. Synnaeve. Better & faster large language models via multi-token prediction. In Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024. OpenReview.net, 2024. URL https://openreview.net/forum?id=pEWAcejiU2.

Google. Our next-generation model: 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. Goldenerg, M. Dubman, S. Kotchubievsky, V. Koushnir, et al. Scalable hierarchical aggregation protocol (SHArP): A hardware architecture for efficient data reduction. In 2016 First International Workshop on Communication Optimizations in HPC (COMHPC), pages 1–10. IEEE, 2016.

A. Gu, B. Rozière, H. Leather, A. Solar-Lezama, G. Synnaeve, and S. I. Wang. Cruxeval: A benchmark for code reasoning, understanding and execution, 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, and W. Liang. Deepseek-coder: When the large language model meets programming - the rise of code intelligence. 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, and P. Gibbons. Pipedream: Fast and efficient pipeline parallel dnn training, 2018. URL https://arxiv.org/abs/1806.03377.

B. He, L. Noci, D. Paliotta, I. Schlag, and T. Hofmann. Understanding and minimising outlier features in transformer training. In The Thirty-eighth Annual Conference on Neural Information Processing Systems.

Y. He, S. Li, J. Liu, Y. Tan, W. Wang, H. Huang, X. Bu, H. Guo, C. Hu, B. Zheng, et al. Chinese simpleqa: A chinese factuality evaluation for large language models. arXiv preprint arXiv:2411.07140, 2024.

D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding. arXiv preprint arXiv:2009.03300, 2020.

D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt. Measuring mathematical problem solving with the math dataset. arXiv preprint arXiv:2103.03874, 2021.

Y. Huang, Y. Bai, Z. Zhu, J. Zhang, J. Zhang, T. Su, J. Liu, C. Lv, Y. Zhang, J. Lei, et al. C-Eval: A multi-level multi-discipline chinese evaluation suite for foundation models. arXiv preprint arXiv:2305.08322, 2023.

N. Jain, K. Han, A. Gu, W. Li, F. Yan, T. Zhang, S. Wang, A. Solar-Lezama, K. Sen, and I. Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. 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, et al. Mistral 7b. arXiv preprint arXiv:2310.06825, 2023.

M. Joshi, E. Choi, D. Weld, and L. Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In R. Barzilay and M.-Y. Kan, editors, Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1601–1611, Vancouver, Canada, July 2017. 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, et al. A study of bfloat16 for deep learning training. arXiv preprint arXiv:1905.12322, 2019.

S. Krishna, K. Krishna, A. Mohananey, S. Schwarcz, A. Stambler, S. Upadhyay, and M. Faruqui. Fact, fetch, and reason: A unified evaluation of retrieval-augmented generation. 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, and S. Petrov. Natural questions: a benchmark for question answering research. 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, and E. H. Hovy. RACE: large-scale reading comprehension dataset from examinations. In M. Palmer, R. Hwa, and S. Riedel, editors, Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017, pages 785–794. 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, et al. Rewardbench: Evaluating reward models for language modeling. arXiv preprint arXiv:2403.13787, 2024.

D. Lepikhin, H. Lee, Y. Xu, D. Chen, O. Firat, Y. Huang, M. Krikun, N. Shazeer, and Z. Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. In 9th International Conference on Learning Representations, ICLR 2021. OpenReview.net, 2021. URL https://openreview.net/forum?id=qrwe7XHTmYb.

Y. Leviathan, M. Kalman, and Y. Matias. Fast inference from transformers via speculative decoding. In International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA, volume 202 of Proceedings of Machine Learning Research, pages 19274–19286. PMLR, 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

H. Li, Y. Zhang, F. Koto, Y. Yang, H. Zhao, Y. Gong, N. Duan, and T. Baldwin. CMMLU: Measuring massive multitask language understanding in Chinese. arXiv preprint arXiv:2306.09212, 2023.

S. Li and T. Hoefler. Chimera: efficiently training large-scale neural networks with bidirectional pipelines. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC ' 21, page 1 – 14. ACM, Nov. 2021. 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, and I. Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. arXiv preprint arXiv:2406.11939, 2024a.

W. Li, F. Qi, M. Sun, X. Yi, and J. Zhang. Ccpm: A chinese classical poetry matching dataset, 2021.

Y. Li, F. Wei, C. Zhang, and H. Zhang. EAGLE: speculative sampling requires rethinking feature uncertainty. In Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024. OpenReview.net, 2024b. URL https://openreview.net/forum?id=1NdN7eXyb4.

B. Y. Lin. ZeroEval: A Unified Framework for Evaluating Language Models, July 2024. URL https://github.com/WildEval/ZeroEval.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101, 2017.

S. Lundberg. The art of prompt design: Prompt boundaries and token healing, 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, et al. Ascend HiFloat8 format for deep learning. arXiv preprint arXiv:2409.16626, 2024.

MAA. American invitational mathematics examination - aime. In American Invitational Mathematics Examination - AIME 2024, February 2024. 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, et al. FP8 formats for deep learning. arXiv preprint arXiv:2209.05433, 2022.

Mistral. Cheaper, better, faster, stronger: Continuing to push the frontier of ai and making it accessible to all, 2024. URL https://mistral.ai/news/mixtral-8x22b.

S. Narang, G. Diamos, E. Elsen, P. Micikevicius, J. Alben, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, et al. Mixed precision training. In Int. Conf. on Learning Representation, 2017.

B. Noune, P. Jones, D. Justus, D. Masters, and C. Luschi. 8-bit numerical formats for deep neural networks. arXiv preprint arXiv:2206.02915, 2022.

NVIDIA. Improving network performance of HPC systems using NVIDIA Magnum IO NVSHMEM and GPUDirect Async. https://developer.nvidia.com/blog/improving-network-performance-of-hpc-systems-using-nvidia-magnum-io-nvshmem-and-gpudirect-async, 2022.

NVIDIA. Blackwell architecture. https://www.nvidia.com/en-us/data-center/technologies/blackwell-architecture/, 2024a.

NVIDIA. TransformerEngine, 2024b. URL https://github.com/NVIDIA/TransformerEngine. Accessed: 2024-11-19.

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

OpenAI. Multilingual massive multitask language understanding (mmmlu), 2024b. URL https://huggingface.co/datasets/openai/MMMLU.

OpenAI. Introducing SimpleQA, 2024c. URL https://openai.com/index/introducing-simpleqa/.

OpenAI. Introducing SWE-bench verified we're releasing a human-validated subset of swe-bench that more, 2024d. URL https://openai.com/index/introducing-swe-bench-verified/.

B. Peng, J. Quesnelle, H. Fan, and E. Shippole. Yarn: Efficient context window extension of large language models. arXiv preprint arXiv:2309.00071, 2023a.

H. Peng, K. Wu, Y. Wei, G. Zhao, Y. Yang, Z. Liu, Y. Xiong, Z. Yang, B. Ni, J. Hu, et al. FP8-LM: Training FP8 large language models. arXiv preprint arXiv:2310.18313, 2023b.

P. Qi, X. Wan, G. Huang, and M. Lin. Zero bubble pipeline parallelism. arXiv preprint arXiv:2401.10241, 2023a.

P. Qi, X. Wan, G. Huang, and M. Lin. Zero bubble pipeline parallelism, 2023b. URL https://arxiv.org/abs/2401.10241.

Qwen. Qwen technical report. arXiv preprint arXiv:2309.16609, 2023.

Qwen. Introducing Qwen1.5, 2024a. URL https://qwenlm.github.io/blog/qwen1.5.

Qwen. Qwen2.5: A party of foundation models, 2024b. URL https://qwenlm.github.io/blog/qwen2.5.

S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He. Zero: Memory optimizations toward training trillion parameter models. In SC20: International Conference for High Performance Computing, Networking, Storage and Analysis, pages 1–16. IEEE, 2020.

D. Rein, B. L. Hou, A. C. Stickland, J. Petty, R. Y. Pang, J. Dirani, J. Michael, and S. R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. arXiv preprint arXiv:2311.12022, 2023.

B. D. Rouhani, R. Zhao, A. More, M. Hall, A. Khodamoradi, S. Deng, D. Choudhary, M. Cornea, E. Dellinger, K. Denolf, et al. Microscaling data formats for deep learning. arXiv preprint arXiv:2310.10537, 2023a.

B. D. Rouhani, R. Zhao, A. More, M. Hall, A. Khodamoradi, S. Deng, D. Choudhary, M. Cornea, E. Dellinger, K. Denolf, et al. Microscaling data formats for deep learning. arXiv preprint arXiv:2310.10537, 2023b.

K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019.

Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, M. Zhang, Y. Li, Y. Wu, and D. Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. arXiv preprint arXiv:2402.03300, 2024.

N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. V. Le, G. E. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In 5th International Conference on Learning Representations, 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, and J. Wei. Language models are multilingual chain-of-thought reasoners. In The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net, 2023. URL https://openreview.net/forum?id=fR3wGCk-IXp.

Y. Shibata, T. Kida, S. Fukamachi, M. Takeda, A. Shinohara, T. Shinohara, and S. Arikawa. Byte pair encoding: A text compression scheme that accelerates pattern matching. 1999.

J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu. Roformer: Enhanced transformer with rotary position embedding. Neurocomputing, 568:127063, 2024.

K. Sun, D. Yu, D. Yu, and C. Cardie. Investigating prior knowledge for challenging chinese machine reading comprehension, 2019a.

M. Sun, X. Chen, J. Z. Kolter, and Z. Liu. Massive activations in large language models. arXiv preprint arXiv:2402.17762, 2024.

X. Sun, J. Choi, C.-Y. Chen, N. Wang, S. Venkataramani, V. V. Srinivasan, X. Cui, W. Zhang, and K. Gopalakrishnan. Hybrid 8-bit floating point (HFP8) training and inference for deep neural networks. Advances in neural information processing systems, 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, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. arXiv preprint 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, and M. Gupta. CUTLASS, Jan. 2023. 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, et al. LLaMA: Open and efficient foundation language models. arXiv preprint 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, and T. Scialom. Llama 2: Open foundation and fine-tuned chat models. 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, and I. Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.

L. Wang, H. Gao, C. Zhao, X. Sun, and D. Dai. Auxiliary-loss-free load balancing strategy for mixture-of-experts. 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, and W. Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. CoRR, abs/2406.01574, 2024b. URL https://doi.org/10.48550/arXiv.2406.01574.

T. Wei, J. Luan, W. Liu, S. Dong, and B. Wang. Cmath: Can your language model pass chinese elementary school math test?, 2023.

M. Wortsman, T. Dettmers, L. Zettlemoyer, A. Morcos, A. Farhadi, and L. Schmidt. Stable and low-precision training for large-scale vision-language models. Advances in Neural Information Processing Systems, 36:10271–10298, 2023.

H. Xi, C. Li, J. Chen, and J. Zhu. Training transformers with 4-bit integers. Advances in Neural Information Processing Systems, 36:49146–49168, 2023.

C. S. Xia, Y. Deng, S. Dunn, and L. Zhang. Agentless: Demystifying llm-based software engineering agents. arXiv preprint, 2024.

H. Xia, T. Ge, P. Wang, S. Chen, F. Wei, and Z. Sui. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation. In Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023, pages 3909–3925. 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, and S. Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In International Conference on Machine Learning, pages 38087–38099. 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, and Z. Lan. CLUE: A chinese language understanding evaluation benchmark. In D. Scott, N. Bel, and C. Zong, editors, Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020, pages 4762–4772. 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.

R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi. HellaSwag: Can a machine really finish your sentence? In A. Korhonen, D. R. Traum, and L. Màrquez, editors, Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers, pages 4791–4800. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1472. URL https://doi.org/10.18653/v1/p19-1472.

W. Zhong, R. Cui, Y. Guo, Y. Liang, S. Lu, Y. Wang, A. Saied, W. Chen, and N. Duan. AGIEval: A human-centric benchmark for evaluating foundation models. CoRR, abs/2304.06364, 2023. doi: 10.48550/arXiv.2304.06364. URL https://doi.org/10.48550/arXiv.2304.06364.

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

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
Shanghao Lu
Shangyan Zhou
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
Yuduan 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 Zhang
Zhewen Hao
Zhibin Gou
Zhicheng Ma
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

商务与合规

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

在每个组别中，作者按名字字母顺序排列。带有 * 标记的名字表示已离开我们团队的成员。

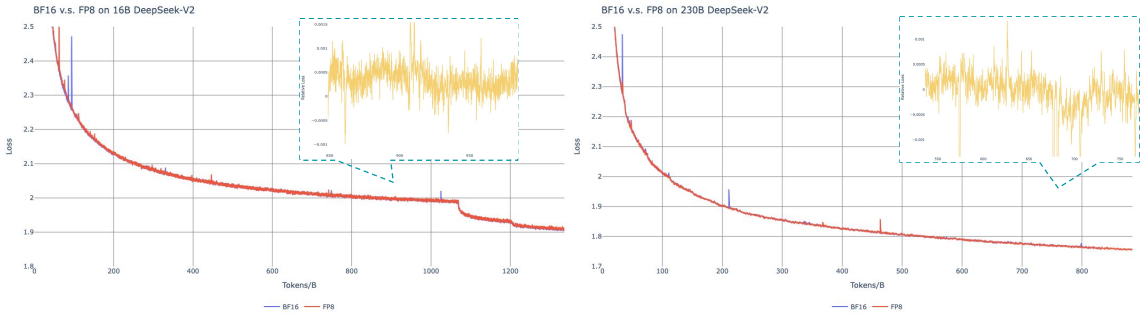


图 10 | BF16 与 FP8 训练的 Loss 曲线对比。结果已通过系数为 0.9 的指数移动平均（EMA）进行平滑处理。

B. 低精度训练的消融实验

B.1. FP8 与 BF16 训练对比

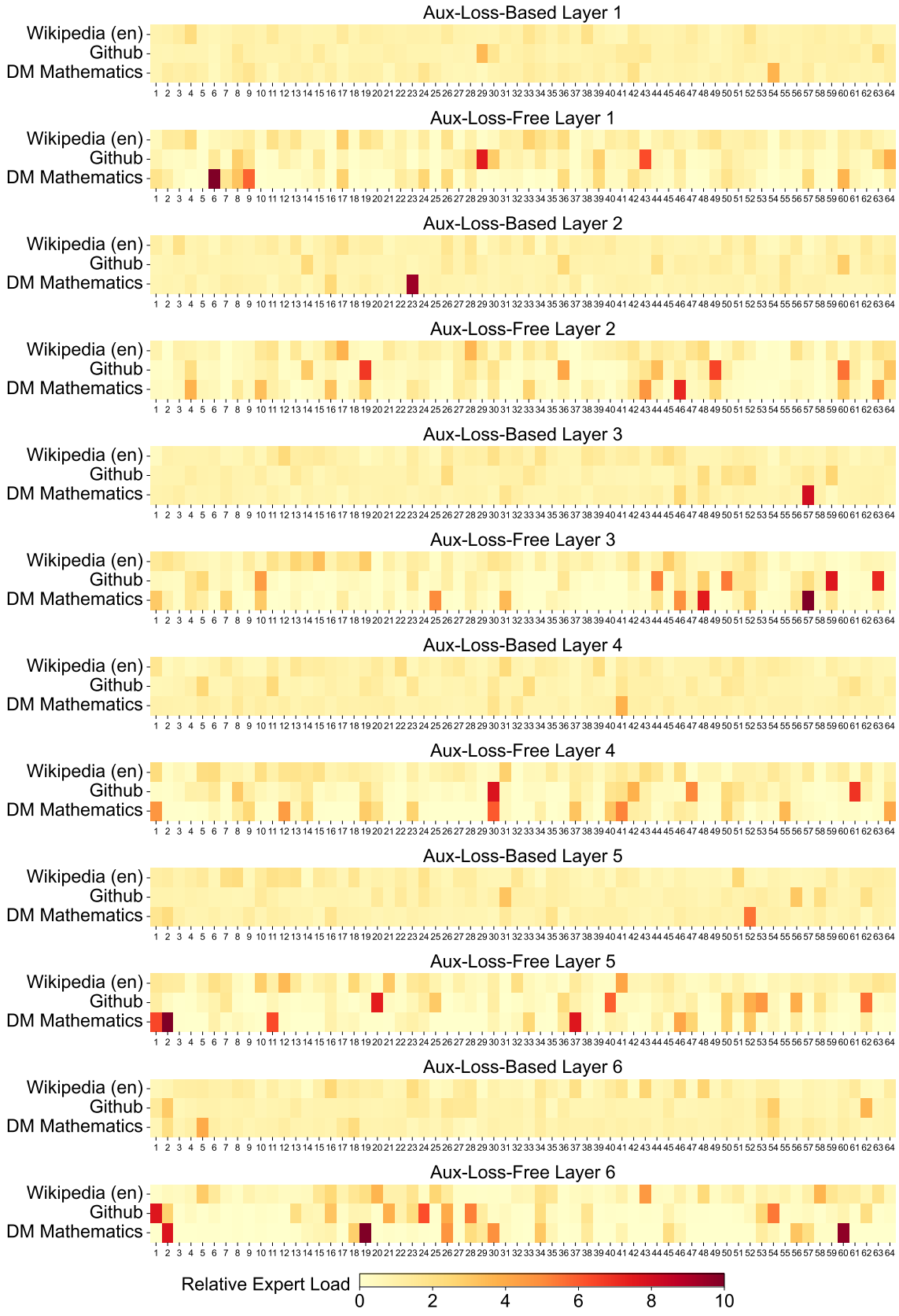
我们在不同规模的两个基线模型上，通过将 FP8 混合精度框架与 BF16 训练进行对比，验证了其有效性。在小规模实验中，我们在 1.33T tokens 上训练了一个包含约 16B 总参数的基线 MoE 模型。在大规模实验中，我们在约 0.9T tokens 上训练了一个包含约 230B 总参数的基线 MoE 模型。我们在图 10 中展示了训练曲线，并证明借助我们的高精度累加和细粒度量化策略，相对误差始终保持在 0.25% 以下。

B.2. 关于分块量化的讨论

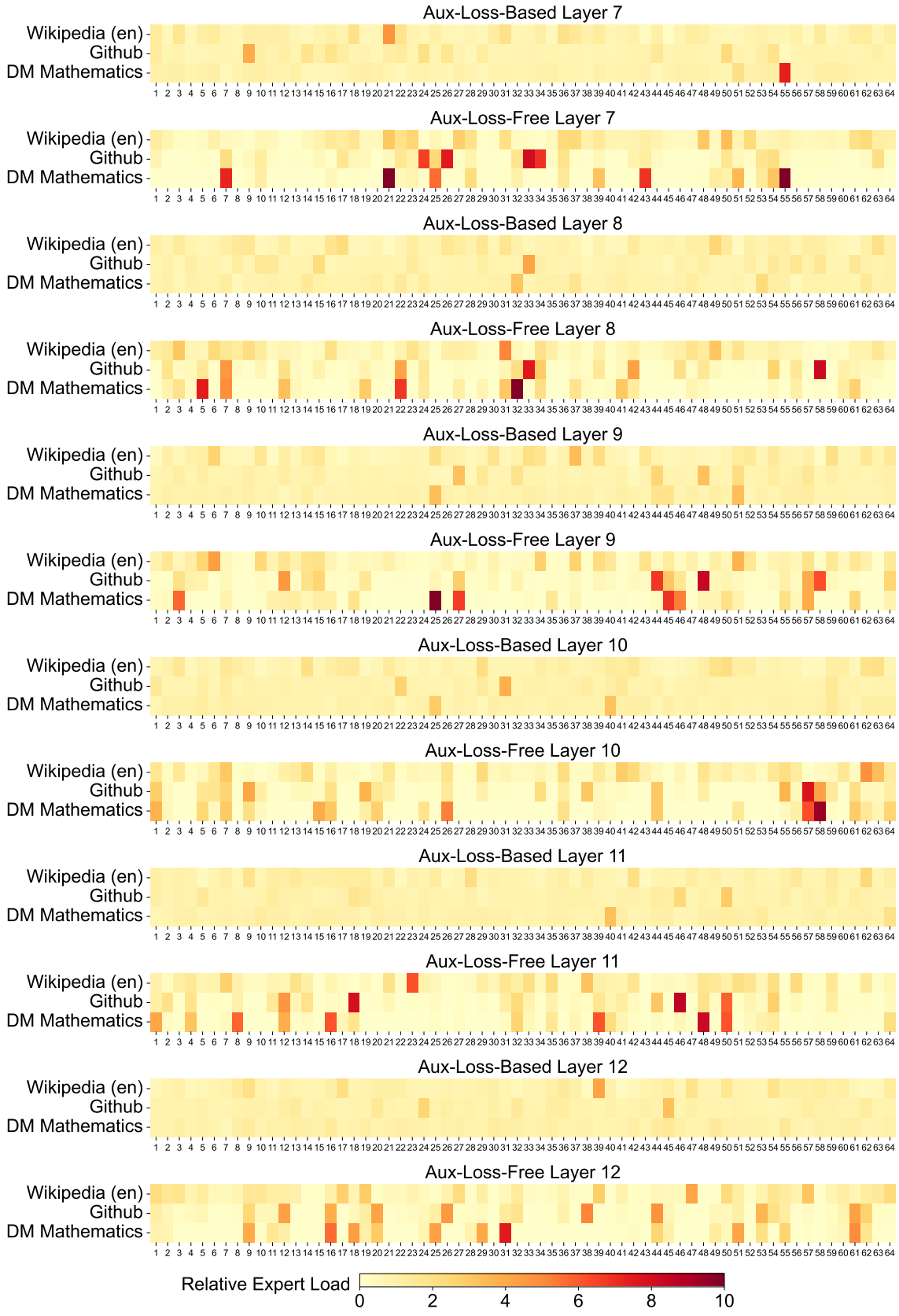
尽管我们的分块细粒度量化有效缓解了特征异常值引入的误差，但它需要为激活值量化采用不同的分组方式，即前向传播使用 `1x128`，反向传播使用 `128x1`。激活梯度也需要类似的处理过程。一种直接的策略是像量化模型权重那样，按每 `128x128` 个元素进行分块量化。这样一来，反向传播仅需进行转置操作。因此，我们进行了一项实验，将所有与 `Dgrad` 相关的张量均按分块方式进行量化。结果表明，`Dgrad` 操作（以链式方式计算激活梯度并向浅层反向传播）对精度高度敏感。具体而言，对激活梯度进行分块量化会导致一个包含约 16B 总参数、训练了约 300B tokens 的 MoE 模型发生发散。我们假设这种敏感性源于激活梯度在不同 token 之间高度不平衡，从而产生了与 token 相关的异常值 (Xi et al., 2023)。这些异常值无法通过分块量化方法得到有效控制。

C. 16B 基于辅助损失与无辅助损失模型的专家专业化模式

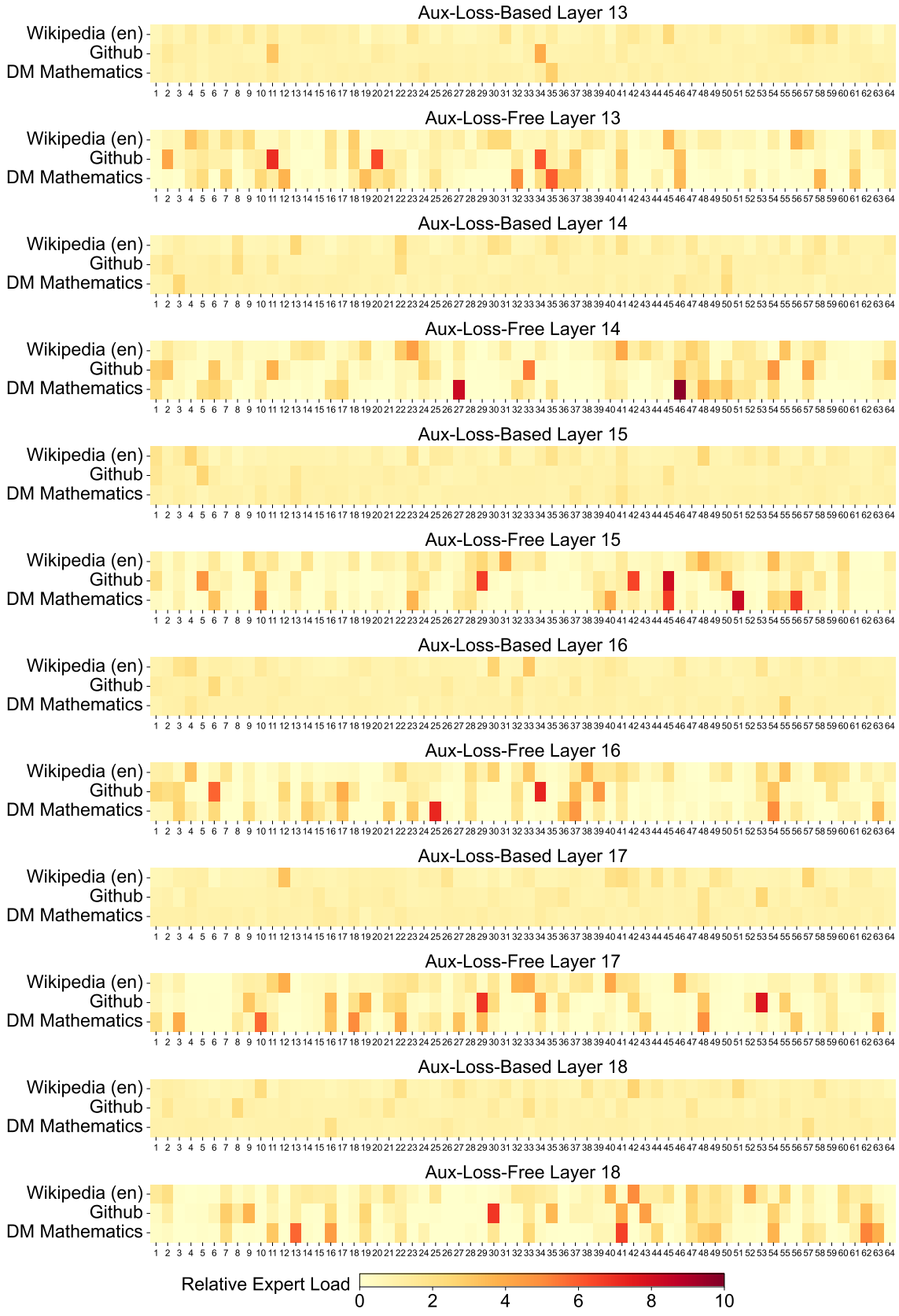
我们记录了 16B 基于辅助损失的基线模型与无辅助损失模型在 Pile 测试集上的专家负载情况。如图 10 所示，无辅助损失模型在所有层中均表现出更强的专家专业化倾向。


Aux-Loss-Based Layer 1
Aux-Loss-Free Layer 1
Aux-Loss-Based Layer 2
Aux-Loss-Free Layer 2
Aux-Loss-Based Layer 3
Aux-Loss-Free Layer 3
Aux-Loss-Based Layer 4
Aux-Loss-Free Layer 4
Aux-Loss-Based Layer 5
Aux-Loss-Free Layer 5
Aux-Loss-Based Layer 6
Aux-Loss-Free Layer 6
Wikipedia (en)
Github
DM Mathematics
Relative Expert Load
0
2
4
6
8
10

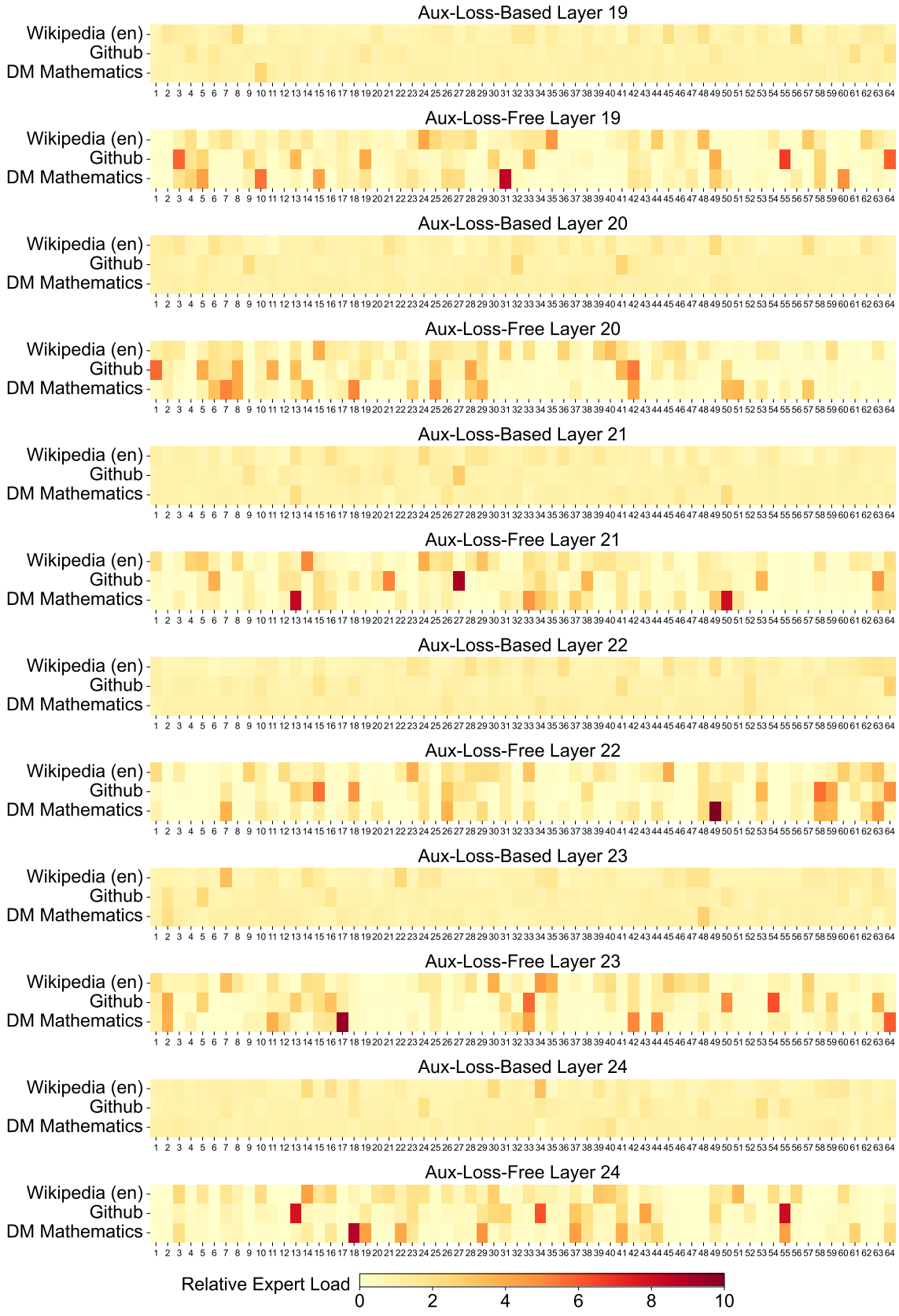
(a) 第 1-7 层


Aux-Loss-Based Layer 7
Aux-Loss-Free Layer 7
Aux-Loss-Based Layer 8
Aux-Loss-Free Layer 8
Aux-Loss-Based Layer 9
Aux-Loss-Free Layer 9
Aux-Loss-Based Layer 10
Aux-Loss-Free Layer 10
Aux-Loss-Based Layer 11
Aux-Loss-Free Layer 11
Aux-Loss-Based Layer 12
Aux-Loss-Free Layer 12
Wikipedia (en)
Github
DM Mathematics
Relative Expert Load
0
2
4
6
8
10

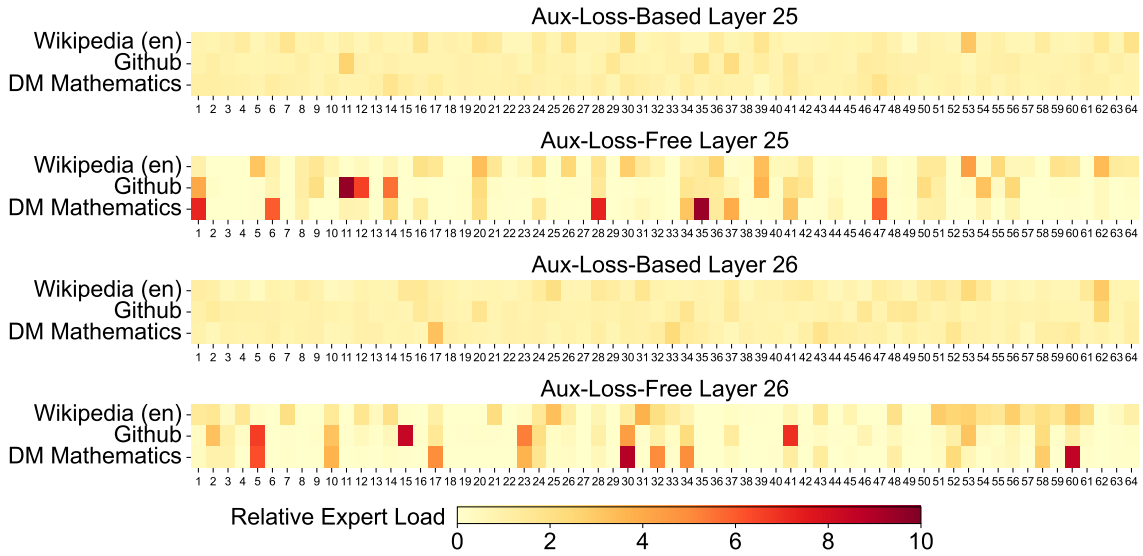
(b) 第 7-13 层


Aux-Loss-Based Layer 13
Aux-Loss-Free Layer 13
Aux-Loss-Based Layer 14
Aux-Loss-Free Layer 14
Aux-Loss-Based Layer 15
Aux-Loss-Free Layer 15
Aux-Loss-Based Layer 16
Aux-Loss-Free Layer 16
Aux-Loss-Based Layer 17
Aux-Loss-Free Layer 17
Aux-Loss-Based Layer 18
Aux-Loss-Free Layer 18
Wikipedia (en)
Github
DM Mathematics
Relative Expert Load
0
2
4
6
8
10

(c) 第 13-19 层


Aux-Loss-Based Layer 19
Aux-Loss-Free Layer 19
Aux-Loss-Based Layer 20
Aux-Loss-Free Layer 20
Aux-Loss-Based Layer 21
Aux-Loss-Free Layer 21
Aux-Loss-Based Layer 22
Aux-Loss-Free Layer 22
Aux-Loss-Based Layer 23
Aux-Loss-Free Layer 23
Aux-Loss-Based Layer 24
Aux-Loss-Free Layer 24
Wikipedia (en)
Github
DM Mathematics
Relative Expert Load
0
2
4
6
8
10

(d) 第 19-25 层



(e) 第 25-27 层

图 10 | 无辅助损失模型与基于辅助损失模型在 Pile 测试集三个领域上的专家负载情况。与基于辅助损失的模型相比，无辅助损失模型展现出更强的专家专业化模式。相对专家负载表示实际专家负载与理论上均衡的专家负载之比。