DeepSeek-V3 Technical Report

Abstract

我们提出了 DeepSeek-V3，一个强大的混合专家（MoE）语言模型，具有 671B 个总参数，每个 token 激活 37B 参数。为了实现高效的推理和低成本的训练，DeepSeek-V3 采用了多头潜在注意力（MLA）和 DeepSeekMoE 架构，这两种架构在 DeepSeek-V2 中得到了充分的验证。此外，DeepSeek-V3 开创了负载平衡的辅助无丢失策略，并为更强的性能设置了多 token 预测训练的目标。我们在 14.8 万亿个不同的高质量的 token 上预训练 DeepSeek-V3，然后进行有监督的微调和强化学习阶段，以充分利用其能力。综合评估表明，DeepSeek-V3 优于其他开源模型，并实现了与头牌闭源模型相当的性能。但 DeepSeek-V3 的完整训练只需要 278.8M 的 H800 GPU 小时。此外，它的训练过程非常稳定。在整个训练过程中，我们没有经历任何不可恢复的损失峰值或执行任何回滚。模型检查点可在 https://github.com/deepseek-ai/DeepSeek-V3 上获得。

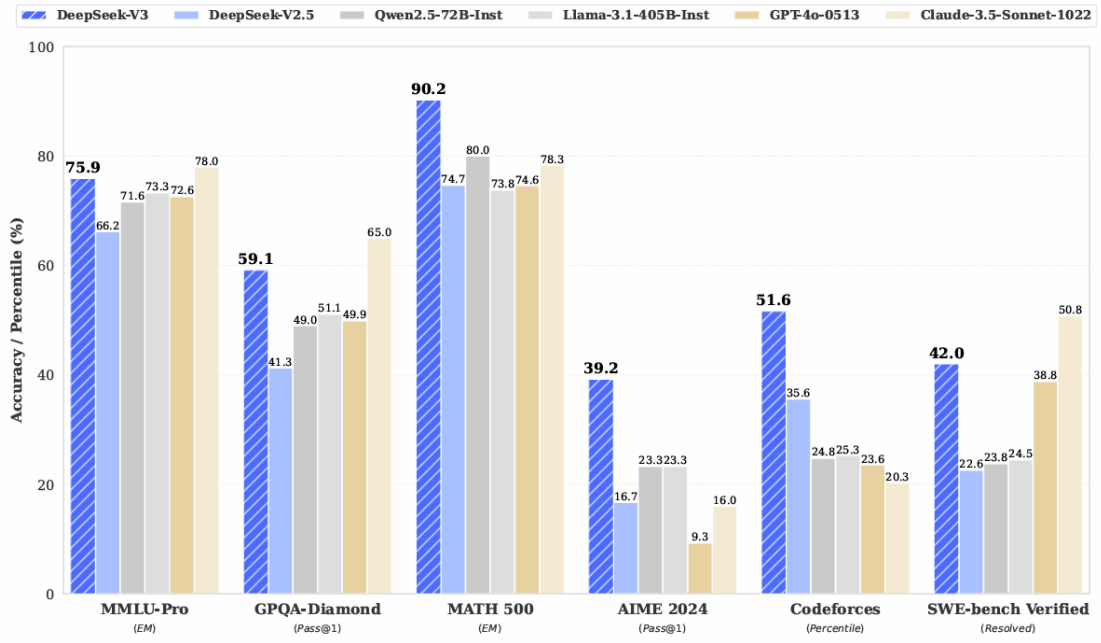

图 1 DeepSeek-V3 及其同类产品的基准性能。

1. Introduction

近年来，大型语言模型（LLM）经历了快速的迭代和发展（Anthropic,2024; Google,2024; OpenAI,2024a），逐步缩小与通用人工智能（AGI）的差距。除了闭源模型，开源模型，包括 DeepSeek 系列，LLaMA 系列，Qwen 系列，Mistral 系列，也取得了重大进展，努力缩小与闭源同行的差距。为了进一步推动开源模型能力的边界，我们扩展了模型并引入了 DeepSeek-V3，这是一个具有 671B 个参数的大型混合专家（MoE）模型，其中每个 token 能激活 37B 个参数。

以前瞻性的视角，我们始终追求强大的模型性能还有经济成本。因此，在架构上，DeepSeek-V3 仍然采用多头潜在注意力（MLA）用于高效推理和采用 DeepSeekMoE 进行低成本训练。这两种架构已经在 DeepSeek-V2 中得到验证，展示了其保持稳健模型性能的能力，能实现高效的训练和推理。在基本架构之外，我们实现了两个额外的策

略来进一步增强模型功能。首先，DeepSeek-V3 开创了一种辅助无丢失策略，用于负载平衡，目的是最大限度地减少由于鼓励负载平衡而对模型性能产生的不利影响。其次，DeepSeek-V3 采用了一个多令牌预测训练目标，我们已经观察到它在评估基准上提高了整体性能。

为了实现高效训练，我们支持 FP8 混合精确训练，并对训练框架进行全面优化。低精度训练已成为高效训练的一种有希望的解决方案(Dettmers 等人，2022；Kalamkar 等人，2019；Narang et al., 2017；Peng et al., 2023b)，其演变与硬件能力的进步密切相关(Luo et al., 2024；micicikevicius 等，2022；Rouhani et al., 2023a)。在这项工作中，我们引入了一个 FP8 混合精度训练框架，并首次在一个非常大规模的模型上验证了它的有效性。通过支持 FP8 计算和存储，我们实现了加速训练和减少 GPU 内存使用。在训练框架方面，我们设计了 DualPipe（双通道）算法来实现高效的管道并行性，该算法具有较少的管道气泡，并且通过计算-通信重叠避免了训练过程中的大部分通信。这种重叠确保了，随着模型进一步扩展，只要我们保持恒定的计算与通信比率，我们仍然可以跨节点雇用细粒度的专家，同时实现接近于零的 all-to-all 的通信开销。此外，我们还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。此外，我们精心优化了内存占用，使得在不使用代价高昂的 tensor 并行性的情况下训练 DeepSeek-V3 成为可能。结合这些努力，我们实现了高的训练效率。

在预训练期间，我们在 14.8T 高质量和多样化的 token 上训练 DeepSeek-V3。预训练过程非常稳定。在整个训练过程中，我们没有

遇到任何不可恢复的损失峰值或必须回滚。接下来，我们对 DeepSeek-V3 进行了两阶段的上下文长度扩展。在第一阶段，最大上下文长度扩展到 32K，在第二阶段，进一步扩展到 128K。在此之后，我们在 DeepSeek-V3 的基础模型上进行后期训练，包括监督微调（SFT）和强化学习（RL），以使其与人类偏好保持一致，并进一步释放其潜力。在训练后阶段，我们从 DeepSeek R1 系列模型中提取推理能力，同时小心地保持模型精度和生成长度之间的平衡。

Training Costs	Pre-Training	Context Extension	Post-Training	Total
in H800 GPU Hours	2664K	119K	5K	2788K
in USD	$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 2 美元/ GPU 小时。

我们对 DeepSeek-V3 进行了全面的基准测试。尽管训练成本较低，但综合评估显示，DeepSeek-V3-Base 已成为目前可用的最强大的开源基础模型，特别是在代码和数学方面。在一系列标准和开放式基准测试中，它的聊天版本也优于其他开源模型，并实现了与领先的闭源模型（包括 gpt-4o 和 Claude-3.5-Sonnet）相当的性能。

最后，我们再次强调 DeepSeek-V3 的经济训练成本，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，在每万亿个令牌上训练 DeepSeek-V3 只需要 180K H800 GPU 小时，即在我们拥有 2048 个 H800 GPU 的集群上需要 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，花费 2664K GPU 小时。结合上下文长度扩展的 119K GPU 小时和后训练的 5K GPU 小时，DeepSeek-V3 的完整训练仅花费 278.8 M GPU 小时。假设 H800 gpu 的租金价格为 2 美元/ GPU 小时，我们的培训总成本仅

为$ 557.6 万。请注意，上述成本仅包括 DeepSeek-V3 的官方培训，不包括与架构、算法或数据的先前研究和消融实验相关的成本。

我们的主要贡献包括:

架构：创新的负载平衡策略和训练目标

在 DeepSeek-V2 的高效架构之上，我们开创了一种辅助的无丢失负载平衡策略，最大限度地减少了由于鼓励负载平衡而引起的性能下降。

我们研究了一个多令牌预测（MTP）目标，并证明它有利于模型性能。它也可以用于推理加速的推测解码。

预训练：迈向终极训练效率

我们设计了一个 FP8 混合精度训练框架，并首次在一个极大规模模型上验证了 FP8 训练的可行性和有效性。

通过算法、框架和硬件的协同设计，克服了跨节点 MoE 训练中的通信瓶颈，实现了近乎全计算的通信重叠。这大大提高了我们的训练效率，降低了训练成本，使我们能够在没有额外开销的情况下进一步扩大模型大小。

以仅 2664 万 H800 GPU 小时的经济成本，我们在 14.8T 令牌上完成了 DeepSeek-V3 的预训练，产生了目前最强的开源基础模型。预训练后的后续训练阶段只需要 0.1M GPU 小时。

后训练：从 DeepSeek-R1 知识蒸馏

我们引入了一种创新的方法，将长思维链（CoT）模型（特别是 DeepSeek R1 系列模型之一）的推理能力提取到标准 llm 中，特别是

DeepSeek- v3。我们的管道将 R1 的验证和反射模式优雅地整合到 DeepSeek-V3 中，并显着提高了其推理性能。同时，我们还对 DeepSeek-V3 的输出样式和长度进行了控制。

核心评价结果总结

知识：(1)在 MMLU，MMLU-Pro 和 GPQA 等教育基准上，DeepSeek-V3 优于所有其他开源模型，在 MMLU 上达到 88.5 分，在 MMLU-Pro 上达到 75.9 分，在 GPQA 上达到 59.1 分。它的性能可与领先的闭源模型（如 gpt-4o 和 Claude-Sonnet-3.5）相媲美，缩小了该领域开源和闭源模型之间的差距。(2)对于事实性基准测试，DeepSeek-V3 在 SimpleQA 和中文 SimpleQA 上都表现出优于开源模型的性能。虽然它在英语事实知识(SimpleQA)上落后于 gpt-4o 和 Claude-Sonnet-3.5，但在中文事实知识（Chinese SimpleQA）上超过了这些模型，突出了它在中文事实知识方面的优势。

代码、数学和推理: (1)DeepSeek-V3 在所有 non-long-CoT 开源和闭源模型中实现了最先进的数学相关基准性能。值得注意的是，它甚至在特定的基准测试（如 MATH-500）上优于 o1-preview，这证明了它强大的数学推理能力。(2)在与编码相关的任务中，DeepSeek-V3 在 LiveCodeBench 等编码竞赛基准中表现最佳，巩固了其在该领域的领先地位。对于与工程相关的任务，虽然 DeepSeek-V3 的性能略低于 Claude-Sonnet-3.5，但它仍然远远超过所有其他模型，证明了它在各种技术基准上的竞争力。

在本文的其余部分中，我们首先详细介绍了我们的 DeepSeek-V3

模型架构（第 2 节）。随后，我们介绍了我们的基础设施，包括我们的计算集群、训练框架、对 FP8 训练的支持、推理部署策略，以及我们对未来硬件设计的建议。接下来，我们描述了我们的预训练过程，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）。之后，我们讨论了我们在后训练方面的努力，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，我们总结了这项工作，讨论了 DeepSeek-V3 的现有局限性。并提出未来研究的可能方向（第 6 部分）。

2. Architecture

我们首先介绍了 DeepSeek-V3 的基本架构，其特点是用于高效推理的 Multi-head Latent attention (MLA)（DeepSeek-AI, 2024c）和用于经济训练的 DeepSeekMoE（Dai 等人，2024）。然后，我们提出了一个多令牌预测（MTP）训练目标，我们已经观察到它可以提高评估基准的整体性能。对于其他没有明确提到的小细节，DeepSeek- v3 遵循 DeepSeek V2 的设置（DeepSeek- ai, 2024c）。

2.1. Basic Architecture

DeepSeek-V3 的基本架构仍然在 Transformer （Vaswani et al., 2017）框架内。为了高效的推理和经济的训练，DeepSeek-V3 还采用了 MLA 和 DeepSeekMoE，这两种方法都经过了 DeepSeek-V2 的彻底验证。与 DeepSeek-V2 相比，一个例外是，我们为 DeepSeekMoE 额外引入了一种辅助的无丢失负载平衡策略（Wangetal.,2024a），以减轻由于确保负载平衡而导致的性能下降。图 2 说明了 DeepSeek-V3 的

基本架构，我们将在本节中简要回顾 MLA 和 DeepSeekMoE 的细节。

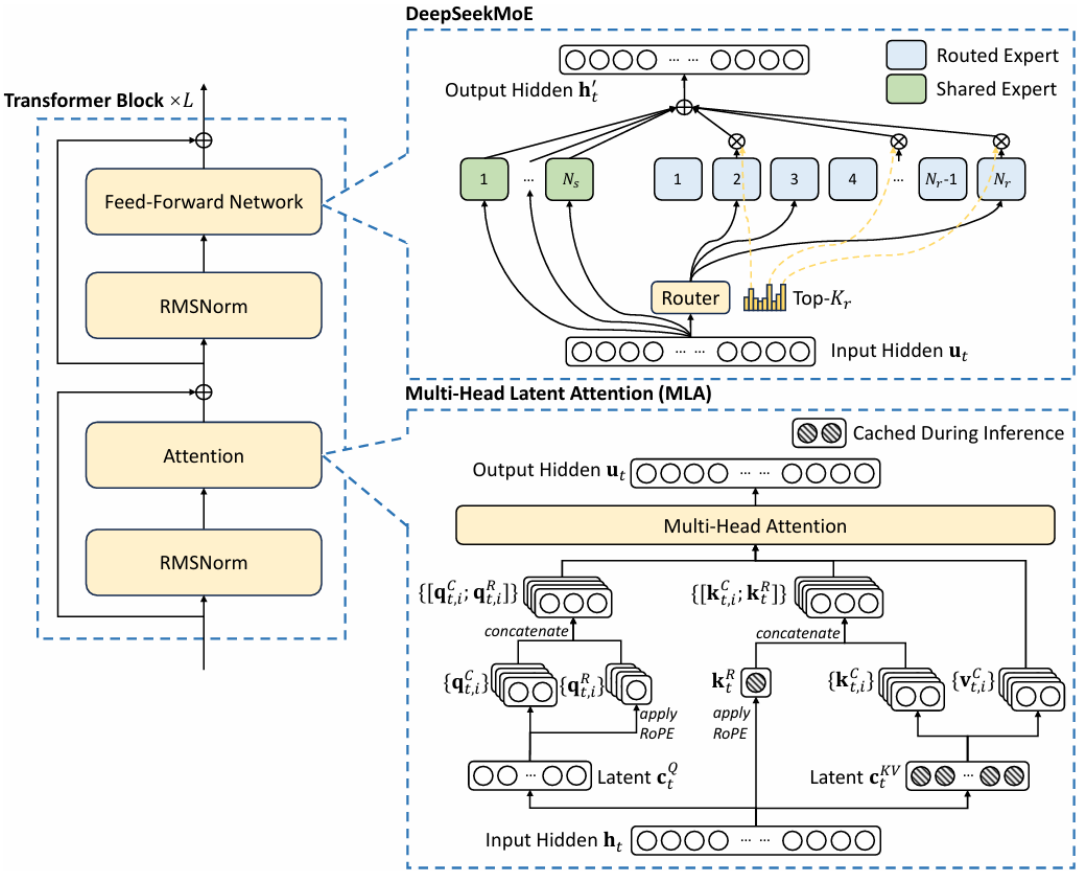

图 2 DeepSeek-V3 的基本架构示意图。继 DeepSeek-V2 之后，我们采用 MLA 和 DeepSeekMoE 进行高效推理和经济训练。

2.1.1. Multi-Head Latent Attention

值得注意的是，DeepSeek-V3 采用了 MLA 架构。设 d 表示 embedding dimension，n_h表示注意头的数量，d_h表示每个头的 dimension，$h_t \in R^d$表示在给定的注意层上第 t 个标记的 attention input。MLA 的核心是对 attention keys 和 values 进行低秩联合压缩，以减少推理过程中 Key-Value（KV）缓存。

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

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

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

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

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

其中$c_t^{KV} \in R^{d_c}$为键和值的压缩潜在向量；d_c（$<< d_h n_h$）表示 KV 压缩的 dimension；$W^{DKV} \in R^{d_c \times d}$为下投影矩阵；$W^{UK}, W^{UV} \in R^{d_h n_h \times d_c}$分别为键和值的上投影矩阵；$W^{KR} \in R^{d_i^R \times d}$是用于产生携带旋转位置嵌入（RoPE）的解耦键的矩阵（Su et al.,2024），RoPE（.）表示应用 RoPE 矩阵的操作；和[.;.]表示串联。注意，对于 MLA，只有蓝框向量(即，c_t^{KV}和K_t^R）需要在生成过程中进行缓存，这使得 KV 缓存显著减少，同时保持与标准的多头注意力（MHA）相当的性能（Vaswani 等人，2017）。

对于注意力查询，我们还进行了低秩压缩，可以减少训练时激活存储量：

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

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

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

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

其中$c_t^Q \in R^{d_c'}$是查询的压缩潜在向量；d_c'（$<< d_h n_h$）表示查询压缩维度；$W^{DQ} \in R^{d_c' \times d}$，$W^{UQ} \in R^{d_h n_h \times d_c'}$分别是查询的下投影和上投影的矩阵；$W^{QR} \in R^{d_i^R n_h \times d_c'}$是生成携带 RoPE 的解耦查询的矩阵。

最后，将注意力查询（$q_{t,i}$）、键（$k_{j,i}$）和值（$v_{j,i}^C$）结合起来，得

到最终的注意力输出u_t:

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

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

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

2.1.2. DeepSeekMoE with Auxiliary-Loss-Free Load Balancing

Basic Architecture of DeepSeekMoE.

未翻

Auxiliary-Loss-Free Load Balancing.

未翻

Complementary Sequence-Wise Auxiliary Loss.

未翻

Node-Limited Routing.

与 DeepSeek-V2 使用的设备受限路由一样，DeepSeek-V3 也使用受限路由机制来限制训练期间的通信成本。简而言之，我们确保每个令牌将被发送到最多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$\frac{k_r}{M}$亲和分数的总和选择的。在此约束下，我们的 MoE 训练框架几乎可以实现完全的计算-通信重叠。

NoToken-Dropping.

由于有效的负载平衡策略，DeepSeek-V3 在其完整训练过程中保持了良好的负载平衡。因此，DeepSeek-V3 在训练过程中不会丢失任何令牌。此外，我们还实现了特定的部署策略来确保推理负载平衡，

因此 DeepSeek-V3 在推理期间也不会丢失令牌。

2.2. Multi-Token Prediction

受 Gloeckle 等人（2024）的启发，我们研究并设置了 DeepSeek-V3 的多令牌预测（MTP）目标，该目标将预测范围扩展到每个位置的多个未来令牌。一方面，MTP 目标使训练信号更加密集，可以提高数据效率。另一方面，MTP 可以使模型预先计划其表示，以便更好地预测未来的令牌。图 3 说明了我们的 MTP 实现。与 Gloeckle 等人（2024）使用独立的输出头并行预测D附加令牌不同，我们顺序预测附加令牌，并在每个预测深度保持完整的因果链。在本节中，我们将介绍 MTP 实现的细节。

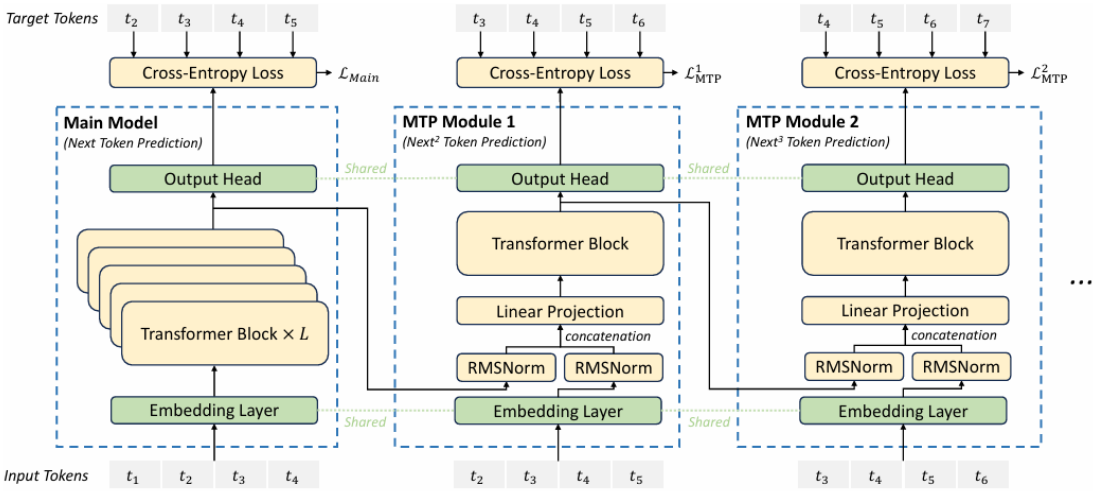

图 3 我们的多令牌预测（MTP）实现示意图。我们为每个深度的每个标记的预测保留了完整的因果链。

MTP Modules.

未翻

MTP Training Objective.

未翻

MTP in Inference. 我们的 MTP 策略主要是为了提高主模型的性能，因此在推理过程中，我们可以直接丢弃 MTP 模块，主模型可以独立正常地工作。此外，我们还可以将这些 MTP 模块重新用于推测解码，以进一步改善生成延迟。

3. Infrastructures

3.1. Compute Clusters

DeepSeek-V3 在配备 2048 个 NVIDIA H800 gpu 的集群上进行训练。H800 集群中每个节点包含 8 个 gpu，节点内通过 NVLink 和 NVSwitch 连接。在不同的节点之间，使用 IB （InfiniBand）互连来方便通信。

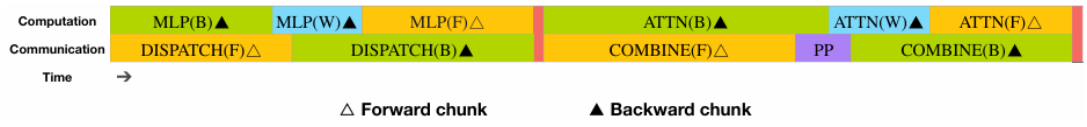

图 4 一对单独的向前和向后块的重叠策略（变压器块的边界未对齐）。橙色表示前进，绿色表示反向输入，蓝色表示反向权重，紫色表示 PP 通信，红色表示障碍。所有 all-to-all 和 PP 通信都可以完全隐藏。

3.2. Training Framework

DeepSeek-V3 的训练由 HAI-LLM 框架支持，HAI-LLM 框架是由我们的工程师从头开始制作的高效轻量级培训框架。总体而言，DeepSeek-V3 采用了 16 路管道并行（PP）（Qi 等人，2023a），64 路专家并行（EP）（Lepikhin 等人，2021），跨越 8 个节点，以及 ZeRO-1 数据并行（DP）（Rajb handari 等人，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 算法来实现高效的流水线并行。与现

有的 PP 方法相比，DualPipe 具有更少的管道气泡。更重要的是，它将向前和向后进程的计算和通信阶段重叠，从而解决了跨节点专家并行带来的沉重通信开销的挑战。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（SMs）。最后，我们在训练过程中精心优化内存占用，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 and Computation-Communication Overlap

对于 DeepSeek-V3，跨节点专家并行带来的通信开销导致计算通信比约为 1:1 的低效率。为了解决这一挑战，我们设计了一种创新的管道并行算法，称为 DualPipe，它不仅通过有效地重叠前向和后向计算通信阶段来加速模型训练，而且还减少了管道气泡。

DualPipe 的关键思想是在一对独立的向前和向后块中重叠计算和通信。具体来说，我们将每个块划分为四个组件：attention, all-to-all dispatch, MLP, and all-to-all combine。特别地，对于后向块，attention 和 MLP 都进一步分成两部分，后向输入和后向权重，如 ZeroBubble（Qi et al., 2023b）。此外，我们有一个 PP 通信组件。如图 4 所示，对于一对向前和向后块，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 SMs 的比例。在这种重叠策略中，我们可以确保在执行过程中完全隐藏 all-to-all 和 PP 通信。考虑到有效的重叠策略，完整的 DualPipe 调度如图 5 所示。它采用双向管道调度，从管道的两端同时馈送微批，很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，只要我们保持恒定的计算与通信比率，我们仍然可以跨节点雇用细粒度的专家，同时实现接近于零的全对全通信开销。

图 5 示例 DualPipe 在两个方向上调度 8 个 PP 队列和 20 个微批。反向微批与正向微批是对称的，为了说明简单，省略了它们的批号。由共享的黑色边界包围的两个单元相互重叠计算和通信。

Method	Bubble	Parameter	Activation
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 不同管道并行方法的管道气泡和内存使用比较。F 表示向前块的执行时间，B 表示完全向后块的执行时间，w 表示“为了权重而向后”块的执行时间，f&b 表示两个相互重叠的向前和向后块的执行时间。

此外，即使在没有沉重通信负担的更一般的场景中，DualPipe 仍然显示出效率优势。在表中，我们总结了不同 PP 方法的管道气泡和内存使用情况。如表所示，与 ZB1P（Qiet al,2023b）和 1F1B（Harlap et al,2018）相比，DualPipe 显著减少了管道气泡，而峰值激活内存仅增加了$\frac{1}{PP}$倍。虽然 DualPipe 需要保留模型参数的两个副本，但这并没有显著增加内存消耗，因为我们在训练期间使用了较大的 EP 大小。与 Chimera（Li and hoefler,2021）相比，DualPipe 只要求管道级和微批可以被 2 整除，而不要求微批可以被管道级整除。此外，对于 DualPipe，气泡和激活内存都不会随着微批数量的增加而增加。

3.2.2. Efficient Implementation of Cross-NodeAll-to-All Communication

为了保证 DualPipe 具有足够的计算性能，我们定制了高效的跨节点 all-to-all 通信内核(包括调度和组合)，以保留专用于通信的 SMs 数量。核的实现是与 MoE 门控算法和我们的集群网络拓扑共同设计的。具体来说，在我们的集群中，跨节点 gpu 与 IB 完全互连，节点内通信通过 NVLink 处理。NVLink 提供 160GB/s 的带宽，大约是 IB（50GB/s）的 3.2 倍。为了有效地利用 lB 和 NVLink 的不同带宽，我们将每个令牌限制为最多 4 个节点，从而减少 IB 流量。对于每个令牌，当它的路由决定被做出时，它将首先通过 IB 传输到目标节点上具有相同 in-nodeindex 的 gpu。一旦它到达目标节点，我们将努力确保它通过 NVLink 立即转发到托管其目标专家的特定 gpu，而不会被随后到达的令牌阻止。通过这种方式，通过 IB 和 NVLink 的通信完全重叠，每个令牌可以有效地在每个节点平均选择 3.2 个专家，而不会产生 NVLink 的额外开销。这意味着，虽然 DeepSeek-V3 在实践中只选择 8 个路由专家，但它可以在保持相同通信成本的情况下将这个数字扩展到最多 13 个专家（4 个节点 × 3.2 个专家/节点）。总的来说，在这种通信策略下，仅 20 个 SMs 就足以充分利用 IB 和 NVLink 的带宽。

详细地说，我们采用了翘曲专门化技术（Bauer 等人，2014），并将 20 个 SMs 划分为 10 个通信通道。在发送过程中，(1)IB 发送，(2)IB-to-NVLink 转发，(3)NVLink 接收由各自的 warp 处理。分配给每个通信任务的转数根据所有 SMs 的实际工作负载动态调整。同样，在合并过程中，(1)NVLink 的发送，(2)NVLink 到 IB 的转发和积累，

(3)IB 的接收和积累也是通过动态调整的 warp 来处理的。此外，调度核和组合核都与计算流重叠，因此我们还考虑了它们对其他 SM 计算核的影响。具体来说，我们采用定制的 PTX（并行线程执行）指令并自动调整通信块大小，这大大减少了 L2 缓存的使用和对其他 SMs 的干扰。

3.2.3. Extremely Memory Saving with Minimal Overhead

为了在训练期间减少内存占用，我们采用了以下技术。

RMSNorm 和 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 向上投影，从而消除了持久存储其输出激活的需要。这种策略的开销很小，但显著降低了存储激活的内存需求。

指数增长的移动平均在 CPU。在训练过程中，我们保留了模型参数的指数增长的移动平均（EMA），以便在学习率衰减后对模型性能进行早期估计。EMA 参数存储在 CPU 内存中，并在每个训练步骤后异步更新。这种方法允许我们维护 EMA 参数，而不会产生额外的内存或时间开销。

Shared Embedding and Output Head for Multi-Token Prediction。使用 DualPipe 策略，我们将模型的最浅层（包括嵌入层）和最深层（包括输出头）部署在相同的 PP 等级上。这种安排使得 MTP 模块和主模型之间能够物理共享 shared embedding 和 output head 的参数和梯度。这种物理共享机制进一步提高了我们的内存效率。

3.3. FP8 Training

受到低精度训练最新进展的启发(Dettmers 等人，2022; Noune 等，2022； Peng et al., 2023b)，我们提出了一个细粒度混合精度框架，利用 FP8 数据格式来训练 DeepSeek-V3。虽然低精度训练具有很大的前景，但它通常受到激活、权重和梯度中存在异常值的限制(Fishman et al., 2024； 他等人。Sun et al., 2024)。尽管在推理量化方面取得了重大进展(Frantar et al., 2022； Xiao et al., 2023)，在大规模语言模型预训练中成功应用低精度技术的研究相对较少（Fishman et al., 2024）。为了解决这一挑战并有效地扩展 FP8 格式的动态范围，我们引入了一种细粒度的量化策略：tile-wise 的 $1 \times N_c$元素分组或 block-wise 的$N_c \times N_c$元素分组。在我们提高精度的累积过程中，相关的去量化开销在很大程度上得到了缓解，这是实现精确的 FP8 通用矩阵乘法（GEMM）的关键方面。此外，为了进一步减少 MoE 训练中的内存和通信开销，我们在 FP8 中缓存和调度活动，而在 BF16 中存储低精度优化器状态。我们在两个类似于 DeepSeek-V2-lite 和 DeepSeek-V2 的模型尺度上验证了提出的 FP8 混合精度框架，训练了大约 1 万亿个令牌（参见附录 B.1 中的更多细节）。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保持在 0.25%以下，这一水平完全在训练随机性的可接受范围内。

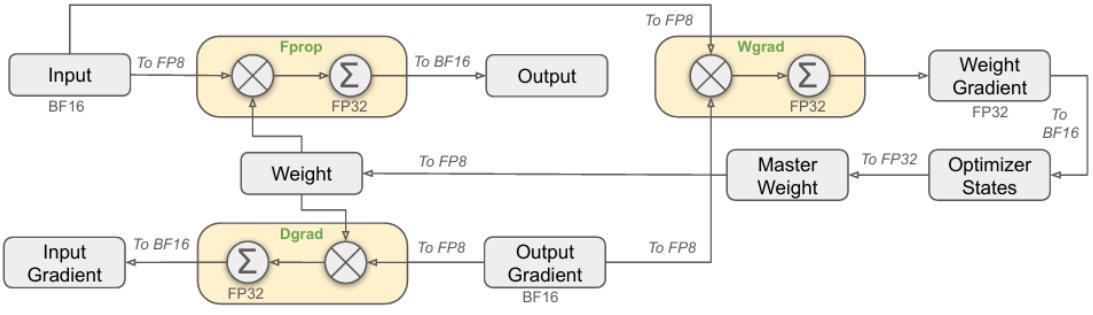

图 6 采用 FP8 数据格式的整体混合精度框架。为了说明，这里只说明了线性算子。

3.3.1. Mixed Precision Framework

基于低精度训练中广泛采用的技术(Kalamkar 等人，2019; Narang 等人，2017)，我们提出了 FP8 训练的混合精度框架。在该框架中，大多数计算密度操作在 FP8 中进行，而一些关键操作策略性地保持其原始数据格式，以平衡训练效率和数值稳定性。整个框架如图 6 所示。

首先，为了加速模型训练，大多数核心计算内核，即 GEMM operations，都是在 FP8 精度下实现的。这些 GEMM operations 接受 FP8 张量作为输入，并在 BF16 或 FP32 中产生输出。如图 6 所示，与线性运算符相关的所有三个 GEMMs，即 Fprop（正向传递）、Dgrad（激活向后传递）和 Wgrad（权重向后传递），都在 FP8 中执行。该设计在理论上使计算速度比原来的 BF16 方法提高了一倍。此外，FP8 Wgrad GEMM 允许将激活信息存储在 FP8 中，以便在反向传递中使用。这大大减少了内存消耗。

尽管 FP8 格式具有效率优势，但由于对低精度计算的敏感性，某些运算符仍然需要更高的精度。此外，一些低成本的操作员也可以利用更高的精度，而总体培训成本的开销可以忽略不计。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，归一化运算符和注意运算符。这些有针对性的高精度保留确保了 DeepSeek-V3 的稳定训练动态。为了进一步保证数值稳定性，我们以更高的精度存储主权重、权重梯度和优化器状态。虽然这些高精度组件会产生一些内存开销，但通过在分布式

训练系统中跨多个 DP 排名进行有效的分片，可以将其影响降到最低。

3.3.2. Improved Precision from Quantization and Multiplication

基于我们的混合精度 FP8 框架，我们介绍了几种提高低精度训练精度的策略，重点关注量化方法和乘法过程。

细粒度的量化。在低精度的训练框架中，由于 FP8 格式的动态范围有限，溢出和下溢是常见的挑战，这受到其减少的指数位的限制。作为标准做法，通过将输入张量的最大绝对值缩放到 FP8 的最大可表示值，将输入分布对齐到 FP8 格式的可表示范围（Narang 等人，2017）。这种方法使得低精度训练对激活异常值高度敏感，严重降低量化精度。为了解决这个问题，我们提出了一种细粒度量化方法，在更细粒度的级别上应用缩放。如图 7 (a)、(1)所示，对于激活，我们以 1x128 块为基础对元素进行分组和缩放（即，每个令牌每 128 个通道）；(2)对于权重，我们以 128x128 块为基础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）。这种方法通过根据较小的元素组调整尺度，确保量化过程可以更好地适应异常值。在附录 B.2 中，我们进一步讨论了当我们以与权重量化相同的方式在块基础上分组和缩放激活时的训练不稳定性。

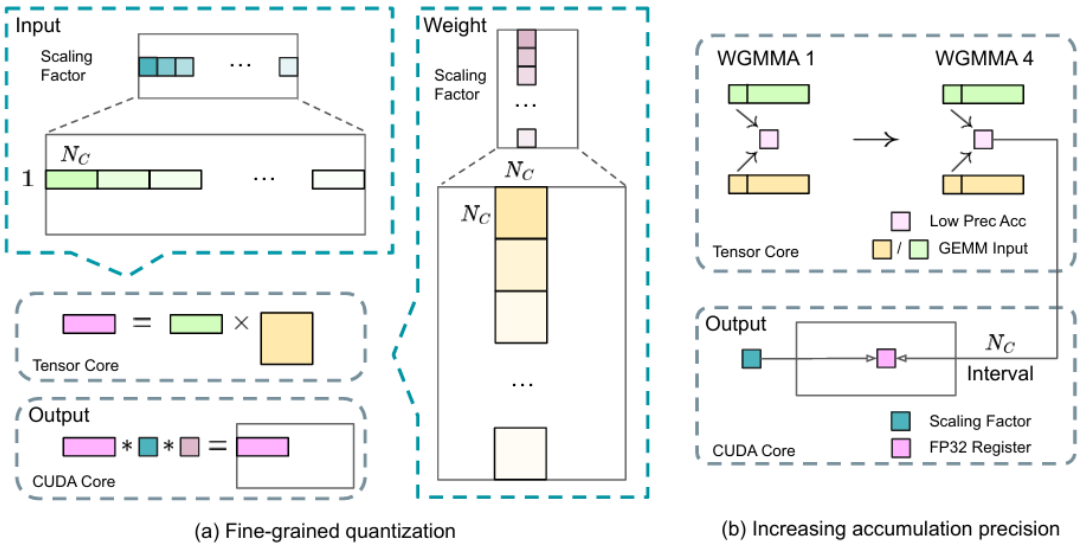

图 7 (a)我们提出了一种细粒度量化方法，以减轻特征异常值引起的量化误差；为了说明简单，只说明了 Fprop。(b)结合我们的量化策略，我们提高了 FP8 的 GEMM 精度，以N_c=128 元素 MMA 的间隔提升到 CUDA 内核，以实现高精度的积累。

我们的方法中的一个关键修改是在 GEMM 操作的内部维度上引入了每组缩放因子。标准 FP8 GEMM 不直接支持此功能。然而，结合我们精确的 FP32 积累策略，它可以有效地实现。

值得注意的是，我们的细粒度量化策略与微尺度格式的思想高度一致（Rouhani 等人，2023b），而 NVIDIA 下一代 gpu （Blackwell 系列）的 Tensor Cores 已经宣布支持具有更小量化粒度的微尺度格式（NVIDIA, 2024a）。我们希望我们的设计可以作为未来工作的参考，以跟上最新的 GPU 架构。

增加积累精度。低精度的 GEMM 操作通常会遇到回流问题，其精度在很大程度上取决于高精度积累，这通常在 FP32 精度中进行 (Kalamkar 等人，2019；Narang 等人，2017)。然而，我们观察到 FP8 GEMM 在 NVIDIA H800 gpu 上的积累精度被限制在 14 位左右，明显低于 FP32 的积累精度。当内部维度 K 很大时，这个问题会变得更加

明显（Wortsman et al., 2023），这是大规模模型训练中的典型场景，其中批量大小和模型宽度都会增加。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 有限的累积精度导致最大相对误差接近 2%。尽管存在这些问题，有限的积累精度仍然是少数 FP8 框架（NVIDIA, 2024b）的默认选项，严重制约了训练精度。

为了解决这个问题，我们采用提升到 CUDA 内核的策略以获得更高的精度（Thakkar et al., 2023）。该过程如图 7 (b)所示。具体而言，在张量核上执行 MMA（矩阵乘法-累积）期间，中间结果使用有限的位宽度累积。一旦达到了一个时间间隔，这些部分结果将被复制到 CUDA 内核上的 FP32 寄存器，在那里执行全精度的 FP32 累积。如前所述，我们的细粒度量化沿着内部维度 K 应用每组缩放因子。这些比例因子可以有效地在 CUDA 内核上相乘作为去量化过程，而额外的计算成本最小。

值得注意的是，这种修改降低了单个 warpgroup 的 WGMMA（Warpgroup-level Matrix Multiply-Accumulate）指令发放率。然而，在 H800 架构上，典型的情况是两个 WGMMA 同时持久化：当一个 warpgroup 执行提升操作时，另一个 warpgroup 能够执行 MMA 操作。这种设计使两个操作重叠，保持张量核的高利用率。根据我们的实验，设置N_c=128 元素，相当于 4 个 WGMMAs，它代表了最小的累积间隔，可以在不引入大量开销的情况下显著提高精度。

Mantissa over Exponents. 与先前工作采用的混合 FP8 格式

(NVIDIA, 2024b; 彭等，20023b; Sun et al., 2019b)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 格式以获得更高的精度。我们将这种方法的可行性归因于我们的细粒度量化策略，即逐块缩放。通过在较小的元素组上操作，我们的方法有效地在这些分组元素之间共享指数位，减轻了有限动态范围的影响。

在线量化。延迟量化被用于张量量化框架(NVIDIA, 2024b; Peng et al., 2023b)，它在之前的迭代中维护最大绝对值的历史，以推断当前值。为了确保准确的尺度并简化框架，我们在线计算每个 1x128 激活块或 128x128 权重块的最大绝对值。在此基础上推导出比例因子，然后将激活或权重在线量化为 FP8 格式。

3.3.3. Low-Precision Storage and Communication

结合我们的 FP8 训练框架，我们通过将缓存的激活和优化器状态压缩成较低精度的格式，进一步减少内存消耗和通信开销。

低精度优化器状态。我们采用 BF16 数据格式而不是 FP32 来跟踪 AdamW 优化器（Loshchilov and Hutter, 2017）中的第一和第二时刻，而不会产生可观察到的性能下降。然而，主权重（由优化器存储）和梯度（用于批量大小累积）仍然保留在 FP32 中，以确保整个训练过程中的数值稳定性。

低精度激活。如图 6 所示，Wgrad 操作是在 FP8 中执行的。为了减少内存消耗，自然会选择以 FP8 格式缓存线性运算符的向后传递的激活。然而，对一些操作员进行低成本高精度培训需要特别考虑：

(1)注意算子后的线性输入。这些激活也用于注意操作符的反向传递，这使得它对精度很敏感。我们专门为这些激活采用定制的 E5M6 数据格式。此外，这些激活将在反向传递中从 1x128 的量化 tile 转换为 128x1 的 tile。为了避免引入额外的量化误差，所有的比例因子都是规整的取值，即 2 的整数次幂。

(2) MoE 中 SwiGLU 算子的输入。为了进一步降低内存开销，我们缓存 SwiGLU 操作符的输入，并在反向传递中重新计算其输出。这些激活也通过我们的细粒度量化方法存储在 FP8 中，在内存效率和计算精度之间取得平衡。

低精度的通信。通信带宽是 MoE 模型训练的关键瓶颈。为了缓解这一挑战，我们量化了 MoE 向上投影到 FP8 之前的 activation，然后应用调度组件，这与 FP8 Fprop 在 MoE 上投影中兼容。与注意算子之后的线性输入一样，该激活的缩放因子是 2 的整数次幂。在 MoE 向下投射之前，对激活梯度应用了类似的策略。对于前向和后向组合组件，我们在 BF16 中保留它们，以保持训练管道关键部分的训练精度。

3.4. Inference and Deployment

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，集群内所有 gpu 通过 IB 完全互连。为了同时确保在线服务的服务水平目标（Service-Level Objective， SLO）和高吞吐量，我们采用以下部署策略，将 prefilling 和 decoding 阶段分开。

3.4.1. Prefilling

预填充阶段的最小部署单元为 4 个节点，32 个 gpu。注意力部分采用 4 路张量并行(TP4)和序列并行(SP), 结合 8 路数据并行(DP8)。它的小 TP 大小为 4，限制了 TP 通信的开销。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处理规模，从而提高计算效率。对于 MoE all-to-all 通信，我们使用与训练中相同的方法: 首先通过 IB 在节点之间传输令牌, 然后通过 NVLink 在节点内 gpu 之间转发。特别是，我们对浅层的密集 MLPs 使用单向张量并行，以节省 TP 通信。

为了在 MoE 部分实现不同专家之间的负载平衡，我们需要确保每个 GPU 处理大约相同数量的令牌。为此, 引入*冗余专家*部署策略，将高负载专家复制并冗余部署。高负载专家根据在线部署过程中收集的统计数据进行检测，并定期（如每 10 分钟）进行调整。在确定冗余专家集合后，我们根据观察到的负载，在节点内的 gpu 之间仔细地重新安排专家，力求在不增加跨节点全对全通信开销的情况下，尽可能地平衡 gpu 之间的负载。对于 DeepSeek-V3 的部署，我们为预填充阶段设置了 32 名冗余专家。对于每个 GPU，除了它所托管的原来的 8 个专家之外，它还将托管一个额外的冗余专家。

此外，在预填充阶段，为了提高吞吐量并隐藏所有对所有和 TP 通信的开销，我们同时处理两个具有相似计算工作量的微批，将一个微批的注意力和 MoE 与另一个微批的调度和组合重叠。

最后，我们正在探索专家的*动态冗余策略*，其中每个 GPU 托管更多的专家（例如，16 个专家），但在每个推理步骤中只有 9 个将被

激活。在每一层的全对全操作开始之前，我们动态地计算全局最优路由方案。考虑到预填充阶段所涉及的大量计算，计算这种路由方案的开销几乎可以忽略不计。

3.4.2. Decoding

在解码过程中，我们将共享专家视为路由专家。从这个角度来看，每个令牌将在路由过程中选择 9 个专家，其中共享的专家被视为将始终被选择的重载专家。解码阶段的最小部署单元由 40 个节点和 320 个 gpu 组成。注意部分采用 TP4 + SP，结合 DP80， MoE 部分采用 EP320。对于 MoE 部分，每个 GPU 只承载一个专家，64 个 GPU 负责托管冗余专家和共享专家。调度和组合部分的所有对所有通信通过 IB 上的直接点对点传输来执行，以实现低延迟。此外，我们还利用 IBGDA （NVIDIA, 2022）技术进一步减少延迟并提高通信效率。

与预填充类似，我们根据在线服务的统计专家负载，在一定间隔内定期确定冗余专家集。然而，我们不需要重新安排专家，因为每个 GPU 只承载一个专家。我们也在探索解码的动态冗余策略。然而，这需要对计算全局最优路由方案的算法进行更仔细的优化，并与调度内核融合以减少开销。

此外，为了提高吞吐量并隐藏所有对所有通信的开销，我们还在探索在解码阶段同时处理具有相似计算工作负载的两个微批。与预填充不同，注意在解码阶段消耗的时间更大。因此，我们将一个微批的注意力与另一个微批的调度+MoE+组合重叠。在解码阶段，每个专家的批处理大小相对较小（通常在 256 个令牌内），瓶颈是内存访问而

不是计算。由于 MoE 部分只需要加载一个专家的参数，因此内存访问开销很小，因此使用更少的 SMs 不会显著影响整体性能。因此，为了避免影响注意力部分的计算速度，我们可以只分配一小部分 SMs 用于 dispatch+MoE+combine。

3.5. Suggestions on Hardware Design

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

未翻。

4. Pre-Training

4.1. Data Construction

与 DeepSeek-V2 相比，我们优化了预训练语料库，提高了数学和编程样本的比例，同时扩大了英语和汉语以外的多语言覆盖范围。此外，我们的数据处理管道进行了改进，以尽量减少冗余，同时保持语料库的多样性。受 Ding 等人（2024）的启发，我们实现了数据完整性的文档打包方法，但在训练过程中没有结合跨样本 attention masking。最后，DeepSeek-V3 的训练语料库由我们的标记者中的 14.8T 高质量和多样化的标记组成。

在 DeepSeekCode-V2（DeepSeek-AI, 2024a）的训练过程中，我们观察到中间填充（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，不会影响下一个令牌的预测能力。为了与 DeepSeekCoder-V2 保持一致，我们还将 FIM 策略纳入 DeepSeek-V3

的预训练中。具体来说，我们采用前缀-后缀-中间（PSM）框架对数据进行如下结构:

<|fim_begin|>f_{pre}<|fim_hole|>f_{suf}<|fim_end|>f_{middle}<|eos_token|>.

该结构作为预打包过程的一部分应用于文档级别。FIM 策略以 0.1 的速率应用，与 PSM 框架一致。

DeepSeek-V3 的标记器采用字节级 BPE (Shibataetal.,1999)，扩展了 128K 个标记的词汇表。我们的标记者的预标记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，新的预标记器引入了结合标点和换行符的标记。然而，当模型处理没有终止换行的多行提示时，这种技巧可能会引入令牌边界偏差（Lundberg, 2023），特别是对于少量的评估提示。为了解决这个问题，我们在训练期间随机分割了一定比例的这种组合令牌，这使模型暴露在更广泛的特殊情况下，并减轻了这种偏差。

4.2. Hyper-Parameters

模型的 Hyper-Parameters。我们将 Transformer 层的数量设置为 61，hidden 维度设置为 7168。所有可学习参数随机初始化，标准差为 0.006。在 MLA 中，我们将注意头n_h的数量设置为 128，每个头的维度d_h设置为 128。KV 压缩维数d_c为 512，查询压缩维数d'_c为 1536。对于解耦的查询和键，我们将每个头的维度d_i^R设置为 64。我们将除前三层外的所有 FFNs 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，每个专家的中间 hidden 维数为 2048。在路由的专家中，每个令牌激活 8 个专家，保证每个令牌最多发送到 4 个

节点。多令牌预测深度D被设置为 1，即，除了确切的下一个令牌之外，每个令牌将预测一个额外的令牌。与 DeepSeek-V2 一样，DeepSeek-V3 在压缩潜在向量后还使用了额外的 RMSNorm 层，并在宽度瓶颈处增加了额外的缩放因子。在这种配置下，DeepSeek-V3 总共包含 671B 个参数，每个 token 激活 37B 参数。

Training Hyper-Parameters. 我们使用 AdamW 优化器（Loshchilov 和 Hutter， 2017），超参数设置为β_1= 0.9，β_2 = 0.95, weight_decay = 0.1。我们在预训练期间将最大序列长度设置为 4K，并在 14.8T token 上预训练 DeepSeek-V3。对于学习率调度，我们首先在前 2k 步中从 0 线性增加到 2.2×10^{-4}。然后，我们保持恒定的学习率为 2.2×10^{-4}，直到模型消耗了 10T 训练 token。随后，我们逐渐将学习率衰减到 2.2×10^{-5}在 4.3T 令牌中，遵循余弦衰减曲线。在最后 500B 个 token 的训练过程中，我们在前 333B 个 token 中保持 2.2×10^{-5}的恒定学习率，在剩下的 167B 个 token 中切换到 7.3×10^{-6}的恒定学习率。梯度裁剪规范（gradient clipping norm）设置为 1.0。我们采用批大小调度策略，在第一个 469B 令牌的训练中，批大小从 3072 逐渐增加到 15360，然后在剩余的训练中保持 15360。我们利用流水线并行性将模型的不同层部署在不同的 gpu 上，每层路由专家将统一部署在属于 8 个节点的 64 gpu 上。对于节点限制路由，每个令牌将最多发送到 4 个节点（即 M＝4）。为了辅助无丢失负载平衡，我们将前 14.3T 令牌的偏差更新速度（bias update speed）γ 设置为 0.001，其余 500B 个令牌设置为 0.0。为了平衡损失，我们将 α 设置为 0.0001，

只是为了避免任何单个序列中的极端不平衡。在前 10T token 中，MTP 损失权重 λ 设置为 0.3，在剩下的 4.8T token 设置为 0.1。

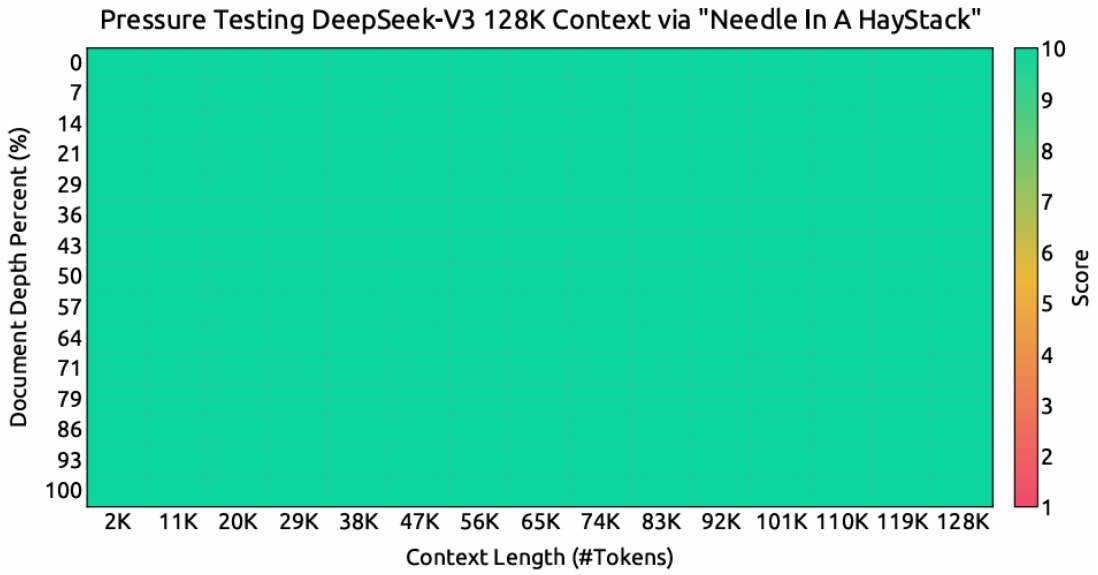

图 8 “大海捞针”（NIAH）测试的评估结果。DeepSeek-V3 在高达 128K 的所有上下文窗口长度上都表现良好。

4.3. Long Context Extension

我们采用了与 DeepSeek-V2（DeepSeek-AI,2024q）类似的方法来实现 DeepSeek-V3 的长上下文功能。在预训练阶段之后，我们应用 YaRN（Peng 等人，2023a）进行上下文扩展，并执行两个额外的训练阶段，每个阶段包括 1000 个步骤，逐步将上下文窗口从 4K 扩展到 32K，然后扩展到 128K。YaRN 配置与 DeepSeek-V2 中使用的配置一致，使用专门解耦的共享密钥k_t^R。两个阶段的超参数保持相同，$s=40$，$\alpha=1$，$\beta=32$，比例因子$\sqrt{t}=0.1\ln s+1$。在第一阶段，序列长度设置为 32K，批大小为 1920。在第二阶段，序列长度增加到 128K，批量大小减少到 480。两个阶段的学习率设置为 7.3×10^{-6}，与预训练阶段的最终学习率相匹配。

4.4. Evaluations

未翻

4.5. Discussion

4.5.1. AblationStudiesforMulti-TokenPrediction

未翻

4.5.2. Ablation Studies for the Auxiliary-Loss-Free Balancing Strategy

未翻

4.5.3. Batch-Wise Load Balance VS. Sequence-Wise Load Balance

未翻

5. Post-Training

5.1. Supervised Fine-Tuning

我们创建指令调优数据集，以包含跨越多个域的 150 万个实例，每个域使用针对其特定需求量身定制的不同数据创建方法。

Reasoning Data.

未翻

Non-Reasoning Data. 对于非推理数据，如创意写作、角色扮演和简单问答，我们使用 DeepSeek-V2.5 来生成响应，并招募人类注释

者来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 进行了两个 epoch 的微调，使用从 5×10^{-6}开始逐渐降低到 1×10^{-6}的余弦衰减学习率调度。在训练过程中，每个序列都是由多个样本打包而成的。然而，我们采用 sample masking 策略，以确保这些例子保持孤立和相互不可见。

5.2. Reinforcement Learning

5.2.1. Reward Model

我们在 RL 过程中采用基于规则的奖励模型（RM）和基于模型的奖励模型。

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的结果，我们要求模型以指定的格式（例如，在框中）提供最终答案，从而允许我们应用规则来验证正确性。类似地，对于 LeetCode 问题，我们可以利用编译器根据测试用例生成反馈。通过尽可能地利用基于规则的验证，我们可以确保更高级别的可靠性，因为这种方法可以抵抗伪造或利用。

基于模型的 RM。对于具有自由形式的基本真理答案的问题，我们依靠奖励模型来确定响应是否与预期的基本真理相匹配。相反，对于那些没有明确的基本事实的问题，比如那些涉及创造性写作的问题，奖励模型的任务是根据问题和相应的答案作为输入提供反馈。奖励模型是从 DeepSeek-V3 SFT 检查点训练的。为了提高其可靠性，我们构

建了偏好数据，该数据不仅提供最终奖励，还包括导致奖励的思维链。这种方法有助于降低在特定任务中被 reward hacking 的风险。

5.2.2. Group Relative Policy Optimization

未翻

5.3. Evaluations

未翻

5.4. Discussion

5.4.1. Distillation from DeepSeek-R1

我们在 DeepSeek-V2.5 的基础上去掉了 DeepSeek-R1 的蒸馏贡献。基线是在短 CoT 数据上训练的，而它的竞争对手使用由上述的专家检查点生成的数据。

表 9 展示了蒸馏数据的有效性，显示了 LiveCodeBench 和 MATH-500 基准测试的显著改进。我们的实验揭示了一个有趣的权衡：蒸馏带来了更好的性能，但也大大增加了平均响应长度。为了保持模型精度和计算效率之间的平衡，我们仔细选择了 DeepSeek-V3 在蒸馏中的最佳设置。

我们的研究表明，从推理模型中提取知识是训练后优化的一个有前途的方向。虽然我们目前的工作主要集中在从数学和编码领域提取数据，但这种方法显示了在各种任务领域更广泛应用的潜力。在这些特定领域证明的有效性表明，长 CoT 蒸馏对于提高模型在其他需要

复杂推理的认知任务中的性能是有价值的。在不同领域进一步探索这种方法仍然是未来研究的重要方向。

5.4.2. Self-Rewarding

奖励在强化学习中扮演着关键角色，引导着优化过程。在通过外部工具直接进行验证的领域中，例如一些编码或数学场景，RL 表现出非凡的效果。然而，在更一般的场景中，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，对于这些更广泛的背景，我们采用了原发性人工智能的方法（Bai 等人，2022），利用 DeepSeek-V3 本身的投票评估结果作为反馈源。该方法产生了显著的对齐效果，显著提高了 DeepSeek-V3 主观评价的性能。通过整合额外的原发性输入，DeepSeek-V3 可以朝着原发性方向进行优化。我们认为，这种将补充信息与 LLMs 作为反馈源相结合的模式是至关重要的。LLM 作为一个多功能处理器，能够将各种场景中的非结构化信息转化为奖励，最终促进 LLM 的自我完善。除了自我奖励之外，我们还致力于发现其他通用和可扩展的奖励方法，以在一般情况下持续推进模型功能。

5.4.3. Multi-Token Prediction Evaluation

DeepSeek-V3 不是仅仅预测下一个令牌，而是通过 MTP 技术预测下两个令牌。结合 speculative decoding 的框架(Leviathan et al., 2023; Xia et al., 2023)，可以显著加快模型的解码速度。关于额外预测的令牌的接受率，自然会出现一个问题。根据我们的评估，第二个令牌预测的接受率在各种生成主题之间的 85%到 90%之间，显示出一致的

可靠性。这种高接受率使 DeepSeek-V3 能够实现显着提高的解码速度，提供 1.8 倍的 TPS（(Tokens Per Second)。

6. Conclusion, Limitations, and Future Directions

在本文中，我们介绍了 DeepSeek-V3，这是一个大型 MoE 语言模型，具有 671B 的总参数和 37B 个激活参数，在 14.8T token 上训练。除了 MLA 和 DeepSeekMoE 架构外，它还开创了用于负载平衡的辅助无丢失策略，并为更强的性能设置了多令牌预测训练目标。由于支持 FP8 培训和细致的工程优化，DeepSeek-V3 的培训具有成本效益。后训练还成功地从 DeepSeek-R1 系列模型中提取了推理能力。综合评估表明，DeepSeek-V3 已成为目前可用的最强开源模型，其性能可与领先的闭源模型（如 gpt-4o 和 Claude-3.5-Sonnet）媲美。不仅它的性能很强，而且也保持了经济的训练成本。它的完整训练只需要 2.788M H800 GPU 小时，包括预训练、上下文长度扩展和后训练。

在承认其强大的性能和成本效益的同时，我们也认识到 DeepSeek-V3 有一些局限性，特别是在部署方面。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，尽管我们的 DeepSeek-V3 部署策略已经实现了端到端生成速度是 DeepSeek-V2 的两倍以上，但仍有进一步增强的潜力。幸运的是，随着更高级硬件的开发，这些限制有望自然得到解决。

DeepSeek 始终坚持开源模型的长远路线，旨在稳步接近 AGI（人工通用智能）的最终目标。未来，我们计划战略性地投资于以下几个方向的研究。

- 我们将不断研究和完善我们的模型架构，旨在进一步提高训练和推理效率，努力接近对无限上下文长度的有效支持。此外，我们将尝试突破 Transformer 的架构限制，从而推动其建模功能的边界。
- 我们将不断迭代我们的训练数据的数量和质量，并探索额外的训练信号源的合并，旨在推动数据在更全面的维度范围内扩展。
- 我们将不断探索和迭代我们的模型的深度思考能力，旨在通过扩展他们的推理长度和深度来提高他们的智力和解决问题的能力。
- 我们将探索更全面和多维的模型评估方法，以防止在研究过程中倾向于优化一组固定的基准，这可能会造成对模型能力的误导印象，并影响我们的基础评估。