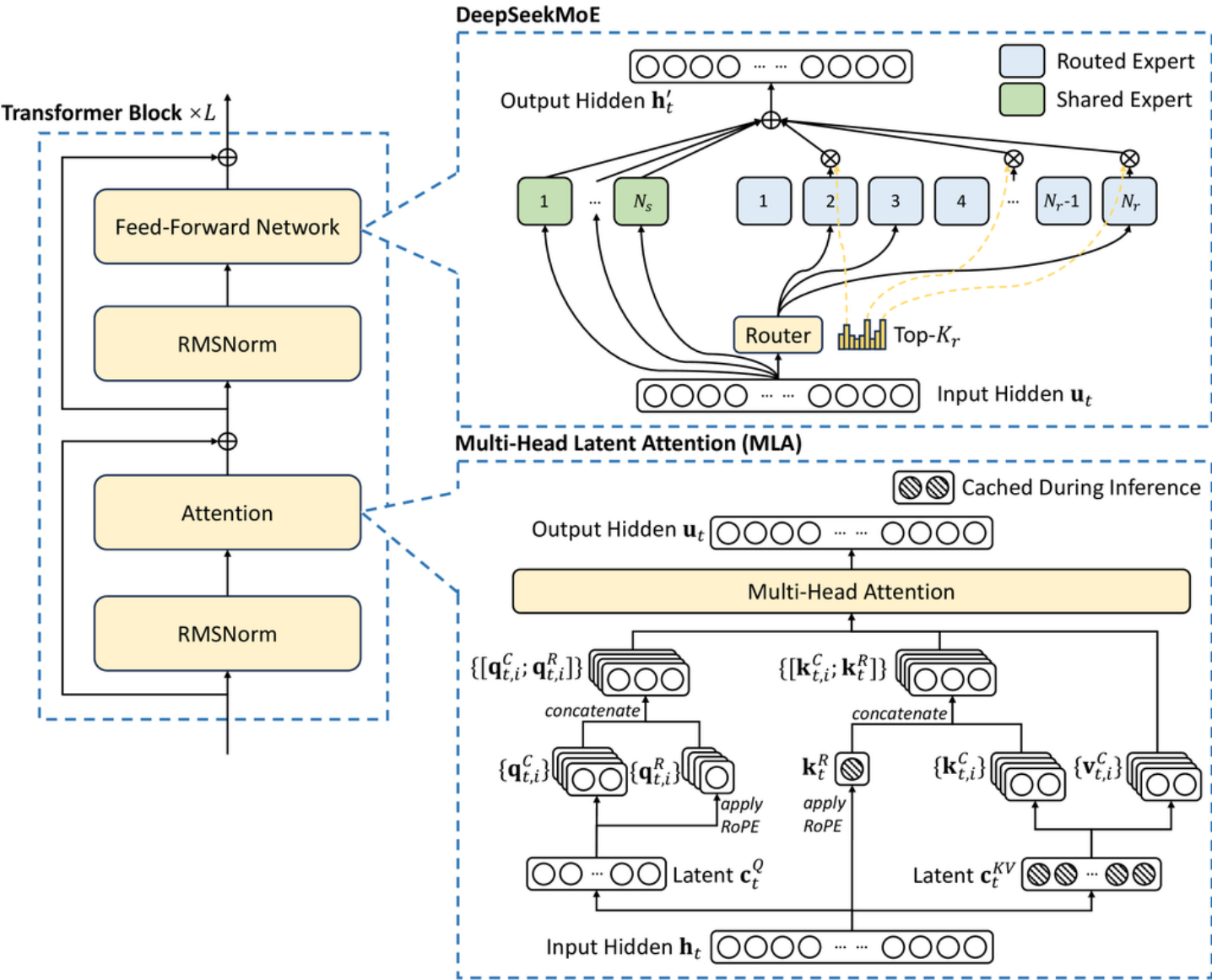


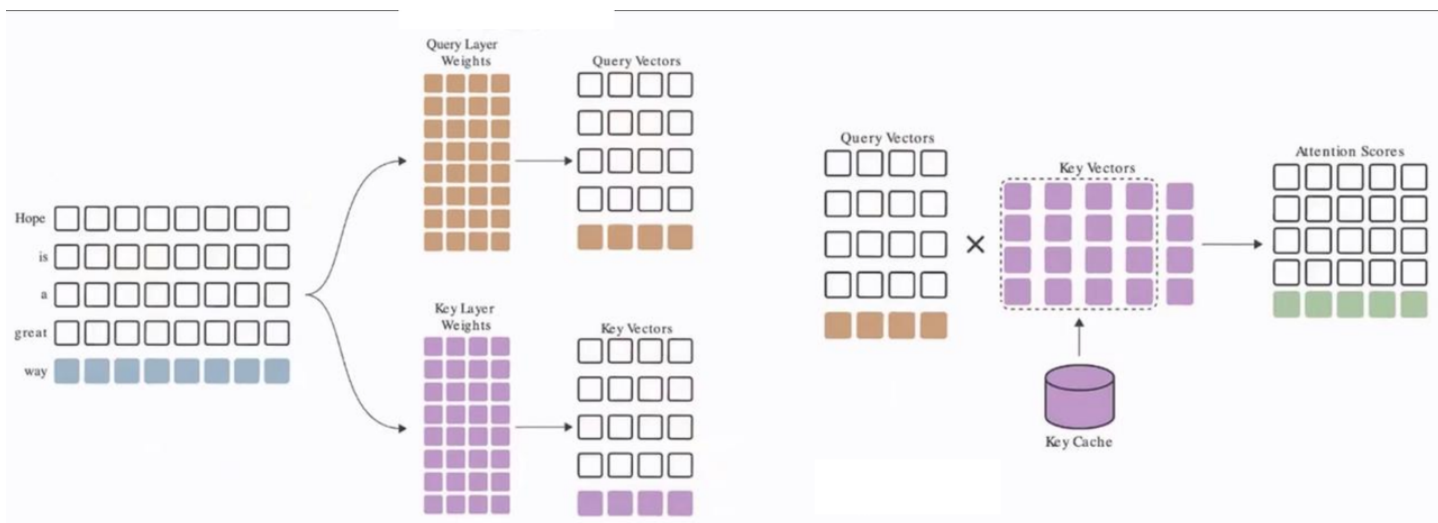
DeepSeek v3技术报告精读

一句话概括：DeepSeek V3模型有671B参数，其中37B激活参数，采用MoE架构，为了加速推理，采用多头潜在注意力MLA；此外，还采用无辅助损失的负载均衡策略，并设定了一个多Token预测训练目标以获得更强的性能。



多头潜在注意力（MLA）

先简要介绍一下KV cache，这是一种在推理阶段最常用的缓存机制，用空间换时间。在进行推理时，新生成的token会加入sequence，一起作为下一次解码的输入。新加入的token不会影响到之前的计算，因此将计算好的每层的K值和V值缓存（k cache和v cache），避免重复计算，加速模型推理速度。如下图所示，之前生成token的k和v矩阵都被存储在kv cache中，预测下一个token时可以直接使用。由于生成任务采用decoder模型，这种模型的单向性被mask为下三角矩阵后，Attention矩阵只有最后一行是有用的。



MLA的核心思想是对键和价值都进行低秩压缩，以降低推理时的KV cache显存：

压缩key和value： 设输入序列的第 t 个 token 的嵌入向量为 $h_t \in R^d$ ，其中 d 是嵌入维度。设 d_h 是每个注意力头的维度， n_h 是注意力头的数量。通过一个下投影矩阵 $W^{DKV} \in R^{d_c \times d}$ ，将 h_t 压缩为一个低维的隐向量 $c_t^{KV} \in R^{d_c}$ ，其中 $d_c \ll d_h n_h$ ：

$$c_t^{KV} = W^{DKV} h_t$$

重建key和value： 通过上投影矩阵 $W^{UK} \in R^{d_h n_h \times d_c}$ 和 $W^{UV} \in R^{d_h n_h \times d_c}$ ，将压缩后的潜在向量 c_t^{KV} 重建为键和价值矩阵：

$$k_t^C = W^{UK} c_t^{KV}, v_t^C = W^{UV} c_t^{KV}$$

如果采用传统的KVcache，对于第 t 个 token 的 query，会和 t 之前所有 token 的 key, value 做注意力计算，因此就可以将之前所有 token 的 k 和 v 都存起来。对于一个 l 层的标准 MHA 的网络来说，kv cache 的大小为 $n_h d_h l$ 。采用 MLA 注意力后的 KVcache 只需要 $d_c l$ 大小。

需要注意的是，对于 key 使用 RoPE 时（**旋转位置编码不作用在 value 上**），由于 k_t^C 是压缩过的，会导致位置信息不准确，这意味着，推理时必须重新计算所有之前 tokens 的 keys，这将大大降低推理效率。因此采用了**解耦RoPE**，引入额外的计算解耦后的 key 的矩阵 $W^{KR} \in d_h^R \times d$ ， d_h^R 是解耦键的维度，占用的大小也就是 $d_h^R l$

最终的键和价值矩阵由压缩后的键和价值以及旋转位置编码后的键组合而成：

$$k_t = [k_t^C; k_t^R], v_t = v_t^C$$

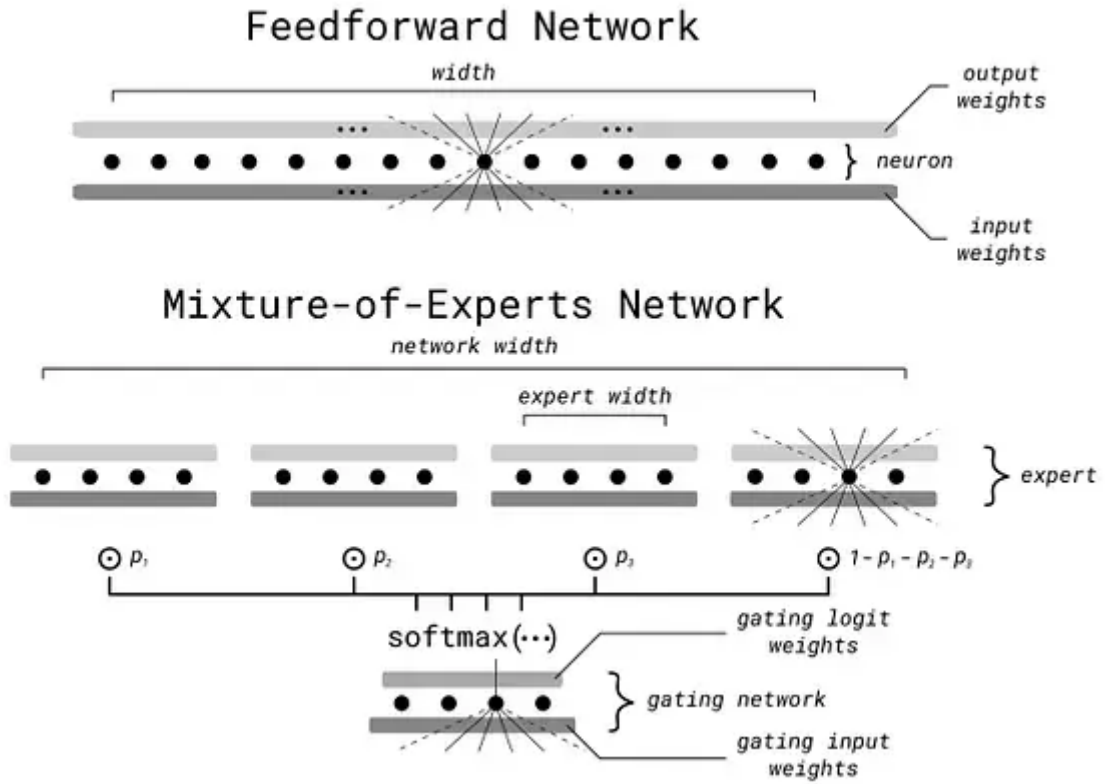
同样的，为了降低训练过程中的内存，还对**query进行低秩压缩**，原理类似，但这部分不能降低KV Cache。

DeepSeekMoE

MoE

首先简要介绍 MoE（Mixture of Experts）。MoE 借鉴集成学习的思想，可以**替换transformer模型中的FFN层**，包含一个门控网络和多个路由专家网络。将 FFN 分割成多个专家，这些专家可以根据其特定的功能被设计成处理不同类型的输入或特征，构建**门控网络**决定对每个输入使用哪些专家。

这种设计的优势在于它能够显著提高模型的效率和可扩展性。由于只有相关的专家被激活，因此可以减少不必要的计算，从而加快模型的推理速度并降低运算成本。同时，这也使得模型能够更加灵活地适应不同的任务，因为不同的任务可能需要不同专家的组合来达到最优的预测效果。



DeepSeekMoE中采用了**共享专家和路由专家**两种， u_t 表示第 t 个token的输入， N_s 和 N_r 分别表示共享专家和路由专家的数量， $FFN_i^{(s)}(\cdot)$ 和 $FFN_i^{(r)}(\cdot)$ 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， $g_{i,t}$ 表示门控值， $s_{i,t}$ 表示token与专家的亲和力， e_i 是第 i 个路由专家的质心向量，FFN输出的 h'_t 为：

$$h'_t = u_t + \sum_{i=1}^{N_s} FFN_i^{(s)}(u_t) + \sum_{i=1}^{N_r} g_{i,t} FFN_i^{(r)}(u_t),$$

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

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

$$s_{i,t} = \text{Sigmoid}(u_t^T e_i),$$

负载均衡

MoE面临**负载不均衡**问题，即大量的token都交给同一个专家处理，经常被选择的专家得到更多的训练，能生成更好的回答，继而更容易被选择。常见的解决负载不均衡的方式有添加噪声、应用softmax函数等。

DeepSeekMoE采用了无辅助损失的负载均衡策略，计算TopK亲和力时，为每个专家的亲和力分数的都加入一个偏差项 b_i (这个偏置量仅仅用于topk筛选，不加入后续的权重计算)：

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

在训练过程中持续监控每个专家的负载，每个训练步骤结束的时候，如果某个专家过载，则按照某一特定比例减少其偏置量；如果某个专家负载不足，则相应的按照同一比例增加其偏置量。

此外，还采用了补充的序列级辅助损失，解决单个输入序列内部的极端负载不均衡问题。T表示输入序列的总长度， $s'_{i,t}$ 表示归一化的输入序列和各个专家的亲和力， P_i 表示第i个专家和序列内的每一个token的亲合度均值，代表了该专家和序列的整体亲合度。 f_i 表示第i个专家在该序列预测过程中的选中频率， α 是超参数。 $f_i P_i$ 代表第i个专家的负载强度，当一个专家被频繁选中时，就会增大损失 \mathcal{L}_{Bal} ，体现对负载不均衡的处罚。

$$\begin{aligned} \mathcal{L}_{Bal} &= \alpha \sum_{i=1}^{N_r} f_i P_i, \\ f_i &= \frac{N_r}{K_r T} \sum_{t=1}^T \mathbb{1} \left(s_{i,t} \in \text{Topk}(\{s_{j,t} | 1 \leq j \leq N_r\}, K_r) \right), \\ s'_{i,t} &= \frac{s_{i,t}}{\sum_{j=1}^{N_r} s_{j,t}}, \\ P_i &= \frac{1}{T} \sum_{t=1}^T s'_{i,t}, \end{aligned}$$

最后，在通信方面，使用限制路由机制来限制训练期间的通信成本，每个token最多被发送到M个算力节点，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，deepseek v3的MoE训练框架几乎可以实现完全的计算-通信重叠。DeepSeek-V3的负载均衡策略保证了在训练期间不会丢弃任何token。

多token预测

MTP (Multi-Token Prediction) 让模型在训练时一次性预测接下来的多个token。这种做法一方面提高了预测效率，另一方面也可以让模型具有更好的上下文理解能力，关注到更多的token。

Deepseek V3中顺序预测附加的token，并在每个预测深度保持完整的因果链。MTP结构利用D个连续的模块预测D个token，其中第k个模块由一个embedding层Emb、一个输出层OutHead、一个transformer层TRM和一个投影矩阵 M^k 构成，其中embedding和outhead层在D个模块之间共享。对于第i个token，预测深度为k时，首先结合第i个token在第k-1深度 $h_i^{k-1} \in R^d$ 的表示和第i+k个token的 $Emb(t_{i+k}) \in R^d$ ，进行线性投影：

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

$h_i^{k'}$ 作为第k个深度的transformer块的输入，T为输入序列的长度，i:j表示切片（包含左右边界），当前深度 h_i^k 的输出为：

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

然后使用输出head层计算得到第i+k+1个token的预测概率 p_{i+k+1}^k ：

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



如何理解MTP：

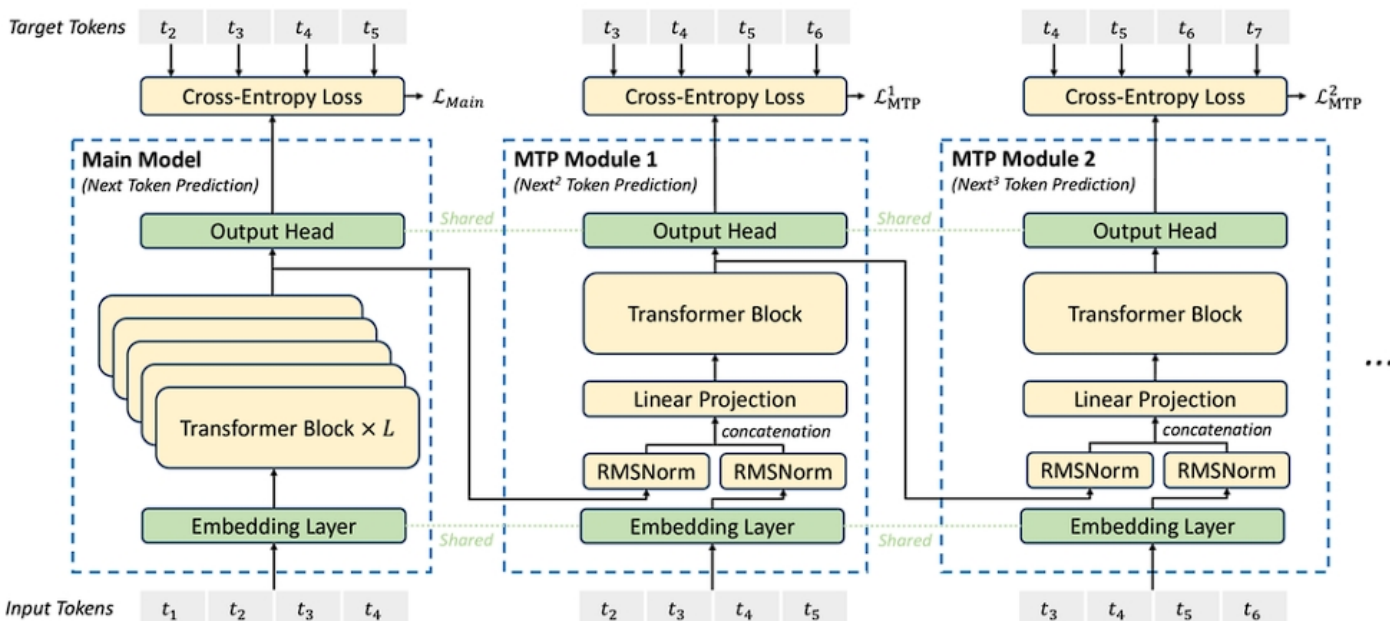
第 i+1 到第 i+k-1 个token的信息隐式存在于 h_i^{k-1} 中。第一个公式表示TRM的输入包含了当前位置和其之后k个输入token的信息。第三个公式表示第 k 个MTP层通过transformer和输出头预测了第 i+k+1 个token的取值概率分布。

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 对应预测概率。MTP的训练目标为：

$$L_{MTP} = \frac{\lambda}{D} \sum_{k=1}^D L_{MTP}^k$$

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

以上都是在训练时使用的MTP结构，在推理时不使用MTP。



技术点

内存优化



为了减少训练期间的内存占用，论文采用了以下技术：

RMSNorm 和 MLA 上投影的重新计算。

我们在反向传播过程中重新计算所有 RMSNorm 操作和 MLA 上投影，显著减少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估计模型性能。EMA 参数存储在 CPU 内存中，并在每个训练步骤之后异步更新。

用于多符元预测的共享嵌入和输出头。

MTP 模块和主模型之间可以物理共享共享嵌入和输出头的参数和梯度。这种物理共享机制进一步提高了我们的内存效率。

并行训练



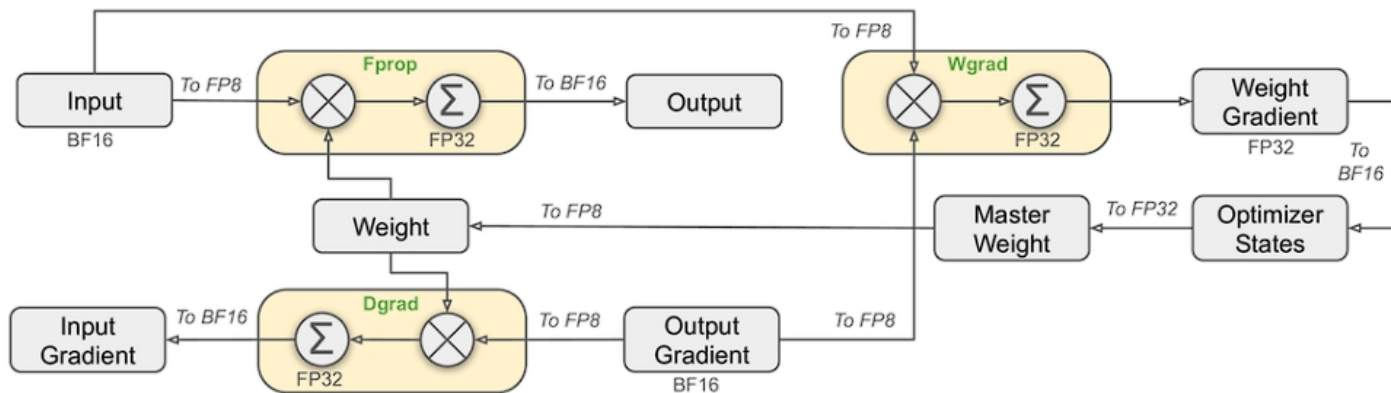
我对通信的了解不算深入，更多细节可以看原论文

- DeepSeek-V3应用了16路流水线并行（PP），跨越8个节点的64路专家并行（EP），以及ZeRO-1数据并行（DP）。
- 采用高效流水线并行的DualPipe算法以提高pipeline并行的效率，重叠了前向和后向过程中的计算和通信阶段，从而解决了跨节点专家并行引入的沉重通信开销的挑战。
- 高效的跨节点全对全通信内核，以充分利用IB和NVLink带宽，并节省专门用于通信的 Streaming Multiprocessors

FP8 混合精度训练

DeepSeek-V3采用了混合精度训练框架，核心思想是在训练过程中对不同的计算操作使用不同的精度，以平衡训练效率和数值稳定性。具体来说：

- **核心计算操作（如矩阵乘法GEMM）**：使用FP8精度（E4M3格式），以加速计算并减少内存占用。
- **关键操作（如嵌入模块、输出头、MoE门控模块、归一化操作和注意力操作）**：保持原始精度（BF16或FP32），以确保这些对精度敏感的操作能够稳定运行。
- **主权重、权重梯度和优化器状态**：存储在更高精度（FP32）中，以确保训练过程的数值稳定性。
- **激活值存储**：在反向传播中，激活值以FP8格式存储，以减少内存占用。对于一些对精度要求较高的操作（如注意力模块后的线性层输入），采用定制的E5M6格式存储激活值。



细粒度量化



为了进一步提高低精度训练的准确性，DeepSeek-V3引入了以下策略：

- 细粒度量化（Tile-wise和Block-wise量化）**：为了应对FP8格式动态范围有限的问题，DeepSeek-V3采用了细粒度量化方法。具体来说：
 - 对于激活值，按 1×128 的tile进行量化。
 - 对于权重，按 128×128 的block进行量化。
 - 在每次迭代中，DeepSeek-V3会计算每个 1×128 激活值tile或 128×128 权重block的最大绝对值，并据此推导出量化因子。
 - 通过这种方式，量化过程能够更好地适应不同元素组的特征分布，从而有效减少因特征异常值导致的量化误差。（之前的量化篇讲过，激活值的绝对值范围要远大于参数，因此要对激活值做更细粒度的量化）
- 增加累积精度**：在FP8矩阵乘法（GEMM）中，为了解决下溢问题，DeepSeek-V3通过在CUDA核心上进行高精度累积（FP32）来提高精度。具体实现方式是：
 - 在Tensor核心执行矩阵乘法时，每累积一定数量（如128个）的中间结果后，将这些结果复制到CUDA核心的FP32寄存器中进行高精度累积。
 - 细粒度量化存储每个组的量化因子，这些比例因子可以在CUDA核心上高效地进行反量化，并具有最小的额外计算成本。
 - 这种方法显著提高了FP8 GEMM的精度，同时通过合理设置累积间隔，避免了过多的计算开销。


预训练

数据构建

DeepSeek-V3的预训练数据集经过精心设计，以确保高质量和多样性。数据集主要包含以下特点：

- 多语言覆盖**：数据集不仅包含英语和中文，还扩展了其他语言的样本，以增强模型的多语言能力。

- **数学和编程样本的增强**：为了提升模型在数学和编程任务上的表现，增加了数学和编程相关样本的比例。
- **文档打包**：采用文档打包（Document Packing）来确保数据的完整性，避免跨样本的注意力掩码问题。

 DeepSeek-V3在预训练中采用了Fill-in-Middle（FIM）策略，通过在文档中随机插入空白区域，让模型学习预测这些空白区域的内容。FIM策略的具体实现如下：

- **PSM框架**：使用Prefix-Suffix-Middle（PSM）框架来结构化数据，将文档分为前缀、后缀和中间部分，中间部分被标记为需要预测的内容。
- **数据格式**：数据格式为 `<|fim_begin|> 前缀 <|fim_hole|> 后缀 <|fim_end|> 中间内容 <|eos_token|>`。
- **应用比例**：FIM策略在预训练中以0.1的概率应用，确保模型在学习预测下一个标记的同时，也能够学习预测中间文本。

DeepSeek-V3中的Tokenizer

DeepSeek-V3使用了基于字节对编码（Byte-level BPE）的Tokenizer，具有以下特点：

- **扩展词汇表**：词汇表大小为128K，以支持多语言的高效压缩。
- **特殊标记**：引入了结合标点符号和换行符的特殊标记，以提高模型对多行文本的处理能力。
- **随机分割**：为了避免模型在处理多行提示时出现边界偏差，训练过程中会随机分割一定比例的结合标记，使模型能够处理更多特殊情况。

超参数

模型架构

- **Transformer层数**：61层。
- **隐藏维度**：7168。
- **初始化标准差**：0.006。
- **多头注意力（MLA）**：
 - 注意力头数：128。
 - 每头维度：128。
 - KV压缩维度：512。
 - 查询压缩维度：1536。
 - 解耦查询和键的每头维度：64。

模型训练

- **优化器**：AdamW，参数设置为 $\beta_1=0.9$ ， $\beta_2=0.95$ ，权重衰减=0.1。
- **最大序列长度**：4K。
- **预训练token数**：14.8万亿个token。
- **学习率调度**：
 - 前2K步线性增加到 2.2×10^{-4} 。
 - 保持 2.2×10^{-4} 直到消耗10万亿个标记。
 - 在接下来的4.3万亿个标记中逐渐衰减到 2.2×10^{-5} ，遵循余弦衰减曲线。

- **MoE层：**
 - 从第4层开始替换为MoE层。
 - 每个MoE层包含1个共享专家和256个路由专家。
 - 每个专家的中间隐藏维度：2048。
 - 每个标记激活的专家数：8。
 - 每个标记最多发送到的节点数：4。
- **多标记预测（MTP）深度：1**，即每个标记预测下一个标记和一个额外的标记。
 - 在最后5000亿个标记中，前3330亿个标记保持 2.2×10^{-4} ，剩余1670亿个标记保持 7.3×10^{-4} 。
- **梯度裁剪范数：1.0。**
- **批大小调度：**
 - 在前4690亿个标记的训练中，批大小从3072逐渐增加到15360。在剩余训练中保持15360。
- **辅助损失自由负载平衡：**
 - 偏置更新速度（ β ）：前14.3万亿个标记为0.001，最后5000亿个标记为0。
- **平衡损失权重（ α ）：0.0001。**
- **MTP损失权重（ λ ）：**前10万亿个标记为0.3，剩余4.8万亿个标记为0.1。

长上下文

在与训练后，进行额外的两个阶段的预训练。

- **第一阶段：**将上下文长度从4K扩展到32K，训练1000步。
 - **序列长度：**32K。
 - **批大小：**1920。
 - **学习率：** 7.3×10^{-4} 。
- **第二阶段：**将上下文长度从32K进一步扩展到128K，再训练1000步。
 - **序列长度：**128K。
 - **批大小：**480。
 - **学习率：** 7.3×10^{-4} 。

后训练

SFT

推理数据

对于推理相关数据集（如数学、编程竞赛问题和逻辑难题），DeepSeek-V3利用内部DeepSeek-R1模型生成数据。R1生成的数据虽然准确度高，但存在过度思考、格式不佳和长度过长等问题。为了平衡R1生成数据的高准确性和清晰简洁的格式，DeepSeek-V3采用以下方法：

- **专家模型**：针对特定领域（如编程、数学或通用推理）开发专家模型，使用SFT和强化学习（RL）流水线训练进行训练。
- **数据生成**：专家模型生成两种类型的SFT样本：
 - **<问题， 原始回答>**：将问题与R1生成的原始回答配对。
 - **<系统提示， 问题， R1回答>**：将问题与R1生成的回答以及系统提示配对，系统提示用于引导模型生成具有反思和验证机制的回答。
- **数据筛选**：在RL训练阶段，模型通过高温采样生成结合R1模式和原始数据的回答。经过数百步RL训练后，使用拒绝采样从专家模型中筛选高质量的SFT数据，确保最终训练数据既保留R1的优点，又简洁有效。

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），DeepSeek-V3使用DeepSeek-V2.5生成回答，并由人工标注员验证数据的准确性和正确性。

推理设置

进行了两轮微调，使用余弦衰减学习率调度，初始学习率为 5×10^{-6} ，逐渐降低到 1×10^{-6} 。在训练过程中，每个序列由多个样本组成，但采用样本掩码策略确保这些样本相互独立且不可见。

强化学习



DeepSeek-V3在RL过程中使用了规则化奖励模型（Rule-Based RM）和基于模型的奖励模型（Model-Based RM）。

- **规则化奖励模型**：对于可以通过特定规则验证的问题，DeepSeek-V3采用规则化奖励系统。例如，某些数学问题有确定的答案，模型需要在指定格式（如方框内）提供最终答案，以便通过规则验证。类似地，对于LeetCode问题，可以使用编译器根据测试用例生成反馈。
- **基于模型的奖励模型**：对于没有确定答案的问题，DeepSeek-V3依赖于奖励模型来确定回答是否符合预期答案。对于没有明确答案的问题（如创意写作），奖励模型需要根据问题和相应的回答提供反馈。奖励模型基于DeepSeek-V3 SFT检查点进行训练。为了增强其可靠性，DeepSeek-V3构建了偏好数据，这些数据不仅提供最终奖励，还包括导致奖励的思考过程。