

CS336 Lecture 4 中文讲 义：注意力替代方案与 混合专家（MoE）

CS336 2026

Lecture 04 - Attention alternatives and mixture of experts

2026 年 10 月 17 日

目录

1	学习目标	1
2	课程主线	1
3	一、注意力机制的替代方案	2
3.1	1.1 背景：长上下文的计算瓶颈	2
3.2	1.2 线性注意力的核心思想：乘法结合律	2
3.3	1.3 Mamba-2：门控线性注意力	3
3.4	1.4 Gated Delta Net：双门控与投影更新	3
3.5	1.5 混合注意力架构的受控研究	4
3.6	1.6 DeepSeek Sparse Attention (DSA)	4
4	二、混合专家 (Mixture of Experts)	5
4.1	2.1 MoE 的基本动机	5
4.2	2.2 路由机制设计空间	6
4.3	2.3 共享专家与细粒度专家	6
4.4	2.4 训练 MoE 的挑战	7
4.5	2.5 微调 MoE 的稳定性问题	8
4.6	2.6 MoE 的系统层面考量	8
4.7	2.7 DeepSeek 系列演进	9
5	三、课程作业关联	9
6	常见误区	9
7	复习清单	10

1 学习目标

完成本讲学习后，你应该能够：

1. 理解线性时间注意力的核心思想——利用矩阵乘法的结合律改变对序列长度的复杂度依赖。
2. 掌握从线性注意力到 Mamba-2、Gated Delta Net 的演进脉络，理解门控机制的作用。
3. 了解 DeepSeek Sparse Attention (DSA) 的 Top-K 索引器方法及其与线性注意力在思路上的根本差异。
4. 理解 MoE (Mixture of Experts) 的基本动机：在不增加 FLOPs 的前提下扩大参数规模。
5. 掌握 MoE 路由的常见设计 (Top-K、共享专家、细粒度专家) 与训练中的负载均衡问题。
6. 了解 MoE 的并行化维度、通信成本、稳定性问题及 upcycling 思想。

2 课程主线

本讲分为两个大主题：

- **第一部分：注意力机制的替代方案**——从标准 softmax attention 的二次方复杂度问题出发，介绍线性时间注意力 (linear attention) 及其后续发展，并讨论稀疏注意力 (DSA) 作为另一条降低成本的路径。
- **第二部分：混合专家 (MoE)**——将 MoE 视为更高效的 MLP 替代品，讨论其路由机制、训练难点、系统层面的并行化，以及 DeepSeek 系列模型中的具体设计。

两条线共同的底层思想是：**稀疏性 + 廉价选择机制**。线性注意力用固定大小的状态压缩历史，DSA 用轻量索引器选出少量 token 做全注意力，MoE 用轻量路由器为每个 token 选择少数专家。这些方法都试图在保持模型表达能力的同时，打破对序列长度或模型规模的直接缩放成本。

3 一、注意力机制的替代方案

3.1 1.1 背景：长上下文的计算瓶颈

大语言模型的上下文窗口正快速增加，但标准注意力（full softmax attention）的计算量随序列长度 **二次方增长**。前馈网络（FFN）的计算量是线性的，因此在短序列下 FFN 占主导，但一旦序列变长，注意力很快成为主要成本。典型的控制手段包括：

- **局部注意力 + 稀疏全局注意力**：只在少数层（如每 8 层一次）做全局注意力，其余层用局部窗口。
- **系统工程优化**：Flash Attention 通过重新安排注意力计算的内存访问模式，实现了约 2 倍的实际速度提升（常数因子改进），并避免了显式物化完整的注意力矩阵。

课堂提示

课程原意：常数因子非常重要。Flash Attention 没有改变二次方复杂度，但它在实际中带来的增益极为显著，说明不应只关注大 O 复杂度。

3.2 1.2 线性注意力的核心思想：乘法结合律

标准注意力可以写成：

$$\text{Attention}(Q, K, V) = \text{softmax}(Q K^T) V$$

忽略 softmax 后（设恒等映射），利用**乘法的结合律**重排计算顺序：

$$(Q K^T) V = Q (K^T V)$$

左侧的复杂度为 $O(n^2 \cdot d)$ ，右侧为 $O(n \cdot d^2)$ 。由于序列长度 n 远大于隐藏维度 d （ n 是百万级， d 是千到万级），右侧的复杂度依赖更有利。

课堂提示

解释：这里的关键不是让二次方消失，而是把二次方从 `seq_len` 转移到 d 上。实际维度 d 远小于 n ，因此计算可行性大幅改善。

进一步地，右侧的矩阵乘法 $K^T V$ 可以写成 RNN 风格的增量更新：

```
1 S_t = S_{t-1} + k_t v_t^T
2 y_t = q_t^T S_t
```

这种形式在推理时的状态 S 大小固定，不需要缓存整个 K/V 序列，非常高效。这就是**线性注意力（linear attention）**的基本形式。它的核心优势在于训练时可用并行矩阵乘法（对偶形式），推理时可用循环形式——二者在数学上完全等价。

3.3 1.3 Mamba-2: 门控线性注意力

Mamba-2 本质上是对线性注意力的一个简单细化：

```
1 S_t = gamma_t S_{t-1} + v_t k_t^T (gamma_t 仅依赖当前输入 x_t, 不
   依赖状态)
```

- gamma_t 是一个输入相关的门控，控制多少历史信息被传向未来。
- 只要更新项仅依赖输入（无状态依赖），就保留了并行/串行的二象性。
- 关键： gamma 与状态无关，所以计算代价极低。

课堂提示

课程原意：Mamba-2 虽然最初从状态空间模型出发，但其运算机制可以看作对线性注意力加了一个简单门控。这是一种视角转换，而非严格的推导。

3.4 1.4 Gated Delta Net: 双门控与投影更新

Gated Delta Net 在 Mamba-2 上进一步加入第二个门控 beta_t ，控制当前 token 的信息是否进入状态：

```
1 S_t = (1 - beta_t k_t k_t^T) S_{t-1} + beta_t v_t k_t^T
```

- beta_t 类似 LSTM 的输入门，控制当前信息写入状态的程度。

- 投影项 ($I - \beta_t k_t k_t^T$) 的直觉: 写入新信息的同时, 擦除状态中与当前 **key** 方向相关的内容。这类似于一种最小二乘或快速权重编程的更新方式。

课堂提示

解释: 课程特别强调, 这个投影更新在多个领域中被独立重新发明过 (元学习、快速权重编程、测试时训练), 说明它是一个鲁棒的、根本性的更新规则。

实际应用: Qwen 3.5 系列采用了 3:1 的 Gated Delta Net 与全注意力的混合架构。混合架构在推理解码吞吐量上显著优于纯注意力模型, 且不损害下游任务性能。

3.5 1.5 混合注意力架构的受控研究

关于混合比例 (RNN 层与全注意力层的比例) 的影响, 目前受控研究仍然有限。已知的规律包括:

- 低混合比例 (如 25%-50% RNN 层) 下, 性能损失几乎为零。
- 超过某一点后, 长上下文检索任务 (如 single-key retrieval) 出现明显下降。
- 纯 RNN (全无注意力) 在所有评估中都有显著性能下降。
- 不同架构的下降曲线有明显差异, Gated Delta Net 在低比例时表现最好。

课堂提示

注意: 从线性注意力到循环形式是精确等价的; 性能下降的根本原因是第一步丢掉 softmax 行归一化所造成的信息损失。

3.6 1.6 DeepSeek Sparse Attention (DSA)

DSA 是一种完全不同的思路——不是改变注意力计算本身的复杂度, 而是减少需要做注意力的 **token** 数量:

1. 先用一个**轻量索引器**计算所有 token 对的 qk 内积并加 ReLU，得到每个位置的得分。
2. 用 Top-K 选出得分最高的少数位置。
3. 仅在选出的子集上执行标准全注意力。

关键特点：

- 索引器的复杂度仍是二次方（必须扫描所有 token），但可以做得非常轻量：低精度、低维度投影。
- Top-K 使得真正执行注意力计算的序列长度可控且有界。
- 训练时可以分两阶段：先正常训练标准 Transformer，然后在长上下文扩展阶段引入索引器并让模型适应它。
- DSA 已被 DeepSeek v3.2 和 GLM-5 等模型验证为有效的低成本方案。

课堂提示

课程原意： DSA 在理论上仍是二次方的，但通过让索引器非常便宜（常数因子小）并限制后续注意力的规模，实际取得了很好的效果。这再次说明常数因子的重要性。

4 二、混合专家 (Mixture of Experts)

4.1 2.1 MoE 的基本动机

MoE 本质上是一种“更高效的 MLP”：将一个大的 FFN 替换为多个小 FFN（专家），通过路由器为每个 token 只激活其中 K 个（或 K+ 共享专家）。这样：

- **总参数量**增加为原来的 N 倍。
- **每次前向/反向的计算成本**只对应 K 个专家的计算量。

核心收益： 在不增加 FLOPs 的前提下获得更多参数，而经验上这通常带来更好的模型质量。

课堂提示

课程原意：MoE 不是概念上的革命，而是一个很实用的参数效率杠杆——即使同一时间只有部分参数处于活跃状态，更多参数本身通常是有益的。

4.2 2.2 路由机制设计空间

路由器 (router) 是一个极其轻量的模块，通常只是一个线性投影——计算输入与每个专家向量之间的内积，再经过 softmax 得到路由概率。以下是主要的设计选择：

路由方式	描述	使用情况
token 选择 Top-K	每个 token 挑选 K 个得分最高的专家	几乎一切现代 MoE 的标准做法
expert 选择	每个专家挑选它最偏好的 token	有少量成功案例，但不是主流
哈希路由	直接对输入进行哈希映射	论文中的基线，未在部署中使用
强化学习路由	将路由器当作策略 (policy) 学习	早期尝试，因开销大而不常用
线性分配求解	全局求解 token-专家最优分配	尚未大规模验证

Top-K 路由器具体实现：

```
1 s = softmax(W_r · x) # 每个专家的得分
2 top_k_indices = topk(s, K)
3 gating = s[top_k_indices] / sum(s[top_k_indices]) # 归一化后作为门控
4 output = sum_i gating_i · FFN_i(x)
```

4.3 2.3 共享专家与细粒度专家

DeepSeekMoE 开创了以下设计，随后成为现代 MoE 的标准：

- **细粒度专家 (fine-grained experts)**：将专家切得更小、更多，使得每个专家的功能更专注。

- **共享专家 (shared experts)**: 少量专家始终被激活, 不参与路由——它们负责处理通用的、高频的模式, 让路由专家更专精。

消融结果表明: 细粒度化总是有增益; 共享专家在某些任务上带来显著改进 (如 TriviaQA、NaturalQuestions), 但 O1MoE 的受控实验认为共享专家的帮助不显著。无论如何, 现代模型 (Qwen 3.5、GLM、DeepSeek v3) 普遍采用了这一组合设计。

4.4 2.4 训练 MoE 的挑战

训练 MoE 的主要困难源于训练期间的稀疏激活:

- 路由器不可微 (Top-K 是离散选择)。
- 无法观察到”反事实”——未被选中的专家没有梯度信号。
- 朴素训练会导致”富者愈富”: 好专家被选中更多次、获得更多梯度、变得更强, 最终导致专家坍塌 (expert collapse) 或专家饥饿 (expert starvation)。

解决方案是一组启发式方法, 核心是负载均衡损失:

Switch Transformer 的 auxiliary loss 可以写成:

```
loss_aux = N · sum_i f_i · P_i
```

- f_i : 被分派到专家 i 的 token 比例。
- P_i : 路由器分配给专家 i 的累积概率质量。

对 P_i 求导后可以看到, 这个损失本质上是在梯度空间里压低那些获得 token 过多的专家——按比例消耗其概率质量。这是对”平衡”的一种平滑近似。

DeepSeek 的扩展:

- 逐专家平衡损失 + 逐设备平衡损失 (保证多 GPU 上的负载均衡, 避免某些设备空闲)。
- DeepSeek v3 引入了在线学习的偏置项来替代部分辅助损失, 但并未完全移除辅助损失。

移除负载均衡损失的灾难性后果 (OIMoE 消融实验):

- 训练损失显著上升。
- 几乎所有 token 被路由到少数两三个专家上，其余专家的参数形同虚设。

课堂提示

课程原意: 负载均衡在早期 MoE 研究和代码中常常被忽略——这意味着某些专家会悄悄被丢弃，并可能静默返回零结果。现代框架已基本解决此类问题，但理解该机制的历史和必要性仍然重要。

4.5 2.5 微调 MoE 的稳定性问题

- MoE 的路由中引入 softmax，而 softmax 的指数运算和除法操作在数值上容易失稳（与上一讲的稳定性讨论一致）。
- 常用策略：在路由器上使用 float32 精度、添加 z-loss 来约束路由 softmax 的分布。
- MoE 微调时容易过拟合：稀疏模型在微调阶段训练-验证差距显著大于稠密模型。缓解手段包括：只微调注意力层、使用更多微调数据，或直接重用预训练数据。

4.6 2.6 MoE 的系统层面考量

并行化: MoE 提供了额外的并行维度——专家并行 (expert parallelism)，将不同专家放在不同设备上并路由 token 至对应设备。但代价是通信开销。近期的做法（如 Nemotron-3）是先对残差流降维再进行通信，以减少传输量。

计算模式: 多个专家 + Top-K 激活，可以通过结构化的稀疏矩阵乘法（如块对角或更通用的结构化稀疏）在 GPU 上高效实现。这属于硬件-架构协同设计，也是 MoE 能快速落地的重要原因。

Upcycling (向上循环): 将一个训练好的稠密模型复制其 MLP 多次，随机初始化路由器，然后继续训练得到 MoE。早期实验表明这比继续训练稠密模型效果更好。但如今已被“直接训练 MoE”取代。

4.7 2.7 DeepSeek 系列演进

- **DeepSeekMoE v1**: 提出共享专家 + 细粒度专家, 使用标准 Top-K 路由和辅助损失平衡。
- **DeepSeekMoE v2**: 扩大规模, 细化到 2 个共享专家 + 更多细粒度专家, 增加逐设备与通信层面的平衡损失, 强调系统级的约束。
- **DeepSeekMoE v3**: 保留共享与细粒度设计, 引入无辅助损失的平衡策略, 并将专家加权方式改为 sigmoid + softmax。此外还引入:
- **MLA (Multi-head Latent Attention)**: 将 Q/K/V 压缩为低维潜在变量 c , 显著减少 KV cache 开销。相对复杂之处在于与 RoPE 的兼容性: 位置编码需要施加在非潜在维度上。
- **MTP (Multi-Token Prediction)**: 同时预测多个未来 token (本质上是把推测解码器训练进权重)。

5 三、课程作业关联

- 本讲内容与课程的注意力实现作业 (如 Flash Attention) 无直接依赖, 但相关思想 (常数因子的重要性、内存带宽优化) 会在系统讲座和作业中展开。
- 作业可能要求你实现 MoE 路由和负载均衡损失, 并体会”去掉负载均衡后训练崩溃”的现象。
- 如果你需要处理长上下文的序列建模任务, 线性注意力的对偶形式 (并行训练 + 串行推理) 可能比标准注意力 + Flash Attention 更省显存。

6 常见误区

1. 把”线性注意力”理解为”完全丢掉 softmax 就行”: 实际上, 去掉 softmax 的归一化会造成信息丢失, 这是线性注意力性能瓶颈的根源, 同时也是后续所有改进 (门控、投影等) 的出发点。

2. **认为 MoE 只是”多个 FFN 拼在一起”**：路由机制、负载均衡、通信调度才是真正决定 MoE 能否工作的关键。
3. **误以为”专家”具有语义知识**（如”金融专家”、”日语专家”）：路由器只是基于内积的轻量打分，没有显式的领域知识。某些 token 类型（如标点、特殊字符）可能会集中到某几个专家，但这不是语义分工。
4. **忽略负载均衡损失的必要性**：只关注主损失而完全不使用任何均衡机制会导致训练迅速退化。
5. **认为 DSA 是线性时间注意力**：DSA 的索引器仍是二次方的，只是常数因子极小且后续注意力规模有界。复杂度上它不属于线性时间方法。

7 复习清单

能否从矩阵乘法的结合律推导出线性注意力的对偶形式（稠密 vs RNN）？

能否解释为什么线性注意力在推理时高效、训练时可并行？

Mamba-2 与 Gated Delta Net 分别在线性注意力上加了什么改动？各自的作用是什么？

为什么混合注意力（部分 RNN 层 + 部分全注意力层）在低比例时性能几乎无损？

DSA 的工作原理是什么？它的索引器为什么可能比全注意力便宜？

MoE 的核心收益是什么？”更多参数但更少 FLOPs”为什么有用？

请写出 Top-K 路由的概率计算与门控融合步骤。

什么是共享专家？细粒度专家？它们各自解决什么问题？

为什么负载均衡损失是有效的？写出 f_i 与 P_i 分别代表什么。

为什么 MoE 在微调时容易过拟合？有哪些公开的缓解手段？

upcycling 与直接训练 MoE 的差异与优劣。

DeepSeek v1 $\rightarrow$ v2 $\rightarrow$ v3 各自的架构变化是什么？

MLA 如何减少 KV cache？为什么需要特殊处理 RoPE？

课堂提示

课堂原话引用（选择性保留）：“注意力中的全对全连接极其强大，但如果你的状态大小远小于上下文长度，压缩必然会有信息损失——这是没有免费午餐的。”