DeepSeek-V3.2: Pushing the Frontier of Open Large Language Models

三个突破：

1、DeepSeek Sparse Attention(DSA)

2、scalable reinforcement learning framework

3、large-scale agentic task synthesis pipeline

该文认为现有开源大模型存在三大缺陷：

1、标准注意力会限制长序列的处理效率

2、开源模型在后训练阶段投入不足

3、开源模型在实际部署中，他的泛化能力和指令遵循能力有明显的滞后

DSA

两大关键组件：

1、**Lightning Indexer（LI）：**

$$I_{t,s} = \sum_{j=1}^{H^I} w_{t,j}^I \cdot \mathrm{ReLU}\left(\mathbf{q}_{t,j}^I \cdot \mathbf{k}_s^I\right), \tag{1}$$

LI计算query token（h_t）与前序令牌（h_s）的索引分数，来确定query token要选择哪些令牌

考虑吞吐量 选择ReLU ， FP8

2、**fine-grained token selection mechanism（FGTS）**

https://zhuanlan.zhihu.com/p/16730036197

FGTS 根据索引分数选择topk 个token，attention计算就是query token 和选择的token

$$\mathbf{u}_t = \mathrm{Attn}\left(\mathbf{h}_t, \left\{\mathbf{c}_s \,\middle|\, I_{t,s} \in \text{Top-k}(I_{t,:})\right\}\right). \tag{2}$$

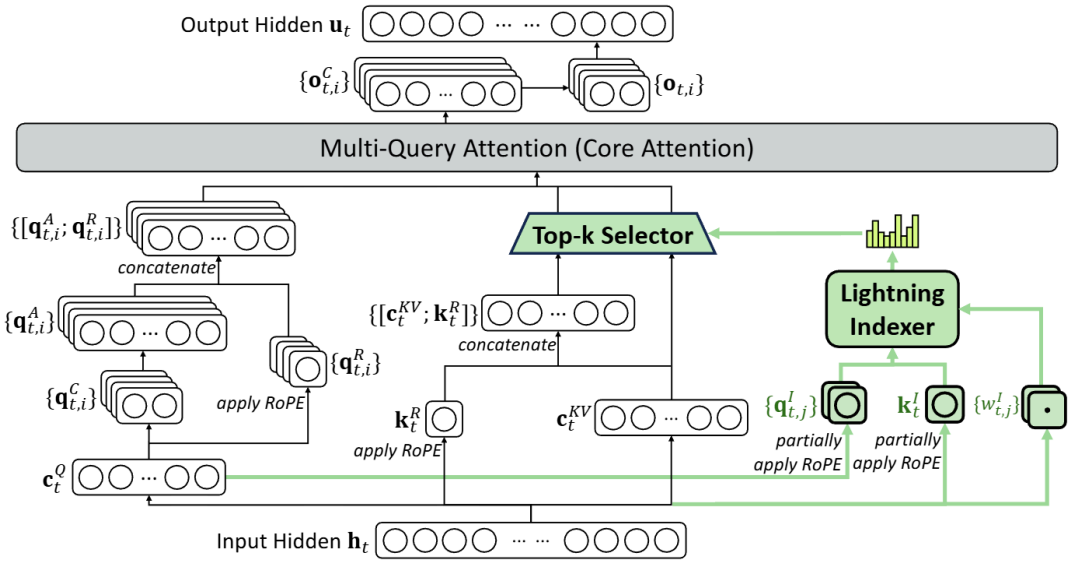


Figure 2 | Attention architecture of DeepSeek-V3.2, where DSA is instantiated under MLA. The green part illustrates how DSA selects the top-k key-value entries according to the indexer.

$$
\begin{aligned}
\mathbf{c}_t^Q &= W^{DQ}\mathbf{h}_t, \\
[\mathbf{q}_{t,1}^C; \mathbf{q}_{t,2}^C; \ldots; \mathbf{q}_{t,n_h}^C] = \mathbf{q}_t^C &= W^{UQ}\mathbf{c}_t^Q, \\
[\mathbf{q}_{t,1}^R; \mathbf{q}_{t,2}^R; \ldots; \mathbf{q}_{t,n_h}^R] = \mathbf{q}_t^R &= \mathrm{RoPE}(W^{QR}\mathbf{c}_t^Q), \\
\mathbf{q}_{t,i} &= [\mathbf{q}_{t,i}^C; \mathbf{q}_{t,i}^R], \\
\boxed{\mathbf{c}_t^{KV}} &= W^{DKV}\mathbf{h}_t, \\
[\mathbf{k}_{t,1}^C; \mathbf{k}_{t,2}^C; \ldots; \mathbf{k}_{t,n_h}^C] = \mathbf{k}_t^C &= W^{UK}\mathbf{c}_t^{KV}, \\
\boxed{\mathbf{k}_t^R} &= \mathrm{RoPE}(W^{KR}\mathbf{h}_t), \\
\mathbf{k}_{t,i} &= [\mathbf{k}_{t,i}^C; \mathbf{k}_t^R], \\
[\mathbf{v}_{t,1}^C; \mathbf{v}_{t,2}^C; \ldots; \mathbf{v}_{t,n_h}^C] = \mathbf{v}_t^C &= W^{UV}\mathbf{c}_t^{KV}, \\
\mathbf{o}_{t,i} &= \sum_{j=1}^{t} \mathrm{Softmax}_j\left(\frac{\mathbf{q}_{t,i}^T \mathbf{k}_{j,i}}{\sqrt{d_h + d_h^R}}\right)\mathbf{v}_{j,i}^C, \\
\mathbf{u}_t &= W^O[\mathbf{o}_{t,1}; \mathbf{o}_{t,2}; \ldots; \mathbf{o}_{t,n_h}],
\end{aligned}
$$

Continued Pre-Training

以 DeepSeek-V3.1-Terminus 的基础检查点为起点，该模型的上下文长度已扩展至128K，通过持续预训练和后续训练，打造出了 DeepSeekV3.2。

Dense Warm-up Stage.

使用一个短的预热阶段来初始化 LI。保持密集注意力，并冻结除LI之外的所有模型参数。为了使索引器输出与主要注意力分布对齐，对于第t个查询令牌，我们首先通过对所有注意力头求和来聚合主要注意力分数。然后，该总和沿序列维度进行 L1 归一化，以产生目标分布p_t。使用KL散度来训练 LI

$$
\mathcal{L}^I = \sum_t \mathbb{D}_{\mathrm{KL}}\left(p_{t,:} \,\|\, \mathrm{Softmax}(I_{t,:})\right). \tag{3}
$$

Sparse Training Stage.

索引器预热之后引入了FGTS，优化所有模型参数。同时保持 LI 与主要注意力分布保持一致， 仅考虑 选择的token

$$\mathcal{L}^I = \sum_t \mathbb{D}_{\mathrm{KL}}\left(p_{t,\mathcal{S}_t} \,\middle\|\, \mathrm{Softmax}\left(I_{t,\mathcal{S}_t}\right)\right). \tag{4}$$

Inference Costs

计算复杂度 从O(L^2) → O(LK) （K<<L）

虽然LI的复杂度是O(L^2) 但是实际速度人快很多

Post-Training

沿用了与 DeepSeek-V3.2-Exp 相同的后训练流程，其中包括专家蒸馏和混合强化学习训练。

Specialist Distillation

对于每个任务，首先开发一个专门的模型，该模型专门用于特定领域，所有专家模型均从同一个Ds-v3.2 微调而来。专家模型准备就绪后，会被用于为最终检查点生成特定领域的数据。**用来合成训练数据。实验结果表明，在蒸馏后数据上训练的模型，性能仅略低于特定领域的专业模型，且通过后续的强化学习训练，性能差距可有效消除。**

Mixed RL Training

GRPO

We first review the objective of GRPO. GRPO optimizes the policy model π_θ by maximizing the following objective on a group of responses $\{o_1, \cdots, o_G\}$ sampled from the old policy π_{old} given each question q:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\text{old}}(\cdot|q)} \left[\frac{1}{G}\sum_{i=1}^{G} \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \min\left(r_{i,t}(\theta)\hat{A}_{i,t}, \text{clip}\left(r_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{i,t}\right) - \beta\mathbb{D}_{\text{KL}}\left(\pi_\theta(o_{i,t}) \,\|\, \pi_{\text{ref}}(o_{i,t})\right) \right], \quad (5)$$

where

$$r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\text{old}}(o_{i,t}|q, o_{i,<t})} \quad (6)$$

is the importance sampling ratio between the current and old policy. ε and β are hyper-parameters controlling the clipping range and KL penalty strength, respectively. $\hat{A}_{i,t}$ is the advantage of $o_{i,t}$ which is estimated by normalizing the outcome reward within each group. Specifically, a set of reward models are used to score an outcome reward R_i for each output o_i in the group, yielding G rewards $\boldsymbol{R} = \{R_1, \cdots, R_G\}$ respectively. The advantage of $o_{i,t}$ is calculated by subtracting the average reward of the group from the reward of output o_i, i.e., $\hat{A}_{i,t} = R_i - \text{mean}(\boldsymbol{R})$.

KL

$$\mathbb{D}_{\text{KL}}\left(\pi_\theta(o_{i,t}) \,\|\, \pi_{\text{ref}}(o_{i,t})\right) = \frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\text{old}}(o_{i,t}|q, o_{i,<t})} \left(\frac{\pi_{\text{ref}}(o_{i,t}|q, o_{i,<t})}{\pi_\theta(o_{i,t}|q, o_{i,<t})} - \log \frac{\pi_{\text{ref}}(o_{i,t}|q, o_{i,<t})}{\pi_\theta(o_{i,t}|q, o_{i,<t})} - 1 \right). \quad (7)$$

$$\log r = (r-1) - \frac{(r-1)^2}{2} + \frac{(r-1)^3}{3} - \frac{(r-1)^4}{4} + \cdots$$

看函数形状（非常重要）：

- $-\log r$
- $r-1$

在 $r \approx 1$ 附近：

$$-\log r \approx (r-1) - \frac{1}{2}(r-1)^2$$

也就是说：

> $-\log r$ **的一阶波动**
> **和** $r-1$ **几乎是“同一个东西”**

所以：

$$(r-1) - \log r \approx \frac{1}{2}(r-1)^2$$

结果是：

令：

$$r = 1+\epsilon, \quad |\epsilon| \ll 1$$

之前我们已经展开过：

$$\log r = \epsilon - \frac{\epsilon^2}{2} + O(\epsilon^3)$$

代入 k2：

$$\begin{aligned} k_2 &= \frac{1}{2}(\log r)^2 \\ &= \frac{1}{2}\left(\epsilon - \frac{\epsilon^2}{2} + O(\epsilon^3)\right)^2 \\ &= \frac{1}{2}\epsilon^2 + O(\epsilon^3) \end{aligned}$$

$$\boxed{k_2 \sim O(\epsilon^2)}$$

使用K3估计 (相比于KL估计，非负，无偏差，方差小)，但进行了改进，用importance ratio进行修正。

这种调整直接带来的结果是，KL 估计器的梯度变为无偏的，消除了系统性的估计误差，从而促进稳定收敛。这与原始的 K3 估计器形成鲜明对比，**尤其是在采样的 token 在当前策略下的概率远低于参考策略（即π_θ≪π_ref）时。在这种情况下，K3 估计器的梯度会给这些标记分配极大、无界的权重，以最大化它们的似然，会导致有噪声的梯度更新，这些更新不断累积，会在后续迭代中降低样本质量**，并引发不稳定的训练动态。在实践中，发现不同领域需要不同强度的 KL 正则化。**对于某些领域，比如数学领域，应用相对较弱的 KL 惩罚，甚至完全不用KL，都能带来性能提升。**

Off-Policy Sequence Masking

为了提高强化学习系统的效率，通常会生成大量的轨迹数据，随后将其分割成多个小批次，用于多个梯度更新步骤。由于训练 - 推理不一致性。

为了稳定训练并提高对离策略更新的容忍度，我们对那些会导致显著策略分歧的负序列进行掩码处理，这是通过数据采样策略与当前策略之间的 KL 散度来衡量的。更具体地说，我们在 GRPO 损失中引入了一个二进制掩码

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\text{old}}(\cdot|q)}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\min\left(r_{i,t}(\theta)\hat{A}_{i,t}, \text{clip}\left(r_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{i,t}\right) M_{i,t} - \beta\mathbb{D}_{\text{KL}}\left(\pi_\theta(o_{i,t})\,\|\,\pi_{\text{ref}}(o_{i,t})\right)\right], \quad (8)$$

where

$$M_{i,t} = \begin{cases} 0 & \hat{A}_{i,t} < 0, \frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\log\frac{\pi_{\text{old}}(o_{i,t}|q,o_{i,<t})}{\pi_\theta(o_{i,t}|q,o_{i,<t})} > \delta \\ 1 & \text{otherwise,} \end{cases} \quad (9)$$

Keep Routing

推理和训练框架之间的差异，再加上策略更新，可能导致即使对于相同的输入，**推理和训练过程中的专家路由也不一致**。这种不一致会导致活跃参数子空间发生突然变化，从而破坏优化的稳定性并加剧离策略问题。为了缓解这一问题，保**留了在推理框架中采样时使用的专家路由路径，并在训练过程中强制使用相同的路由路径，以确保对相同的专家参数进行优化**。研究发现，这种保持路由操作对于 MoE 模型的强化学习训练稳定性至关重要

Thinking in Tool-Use

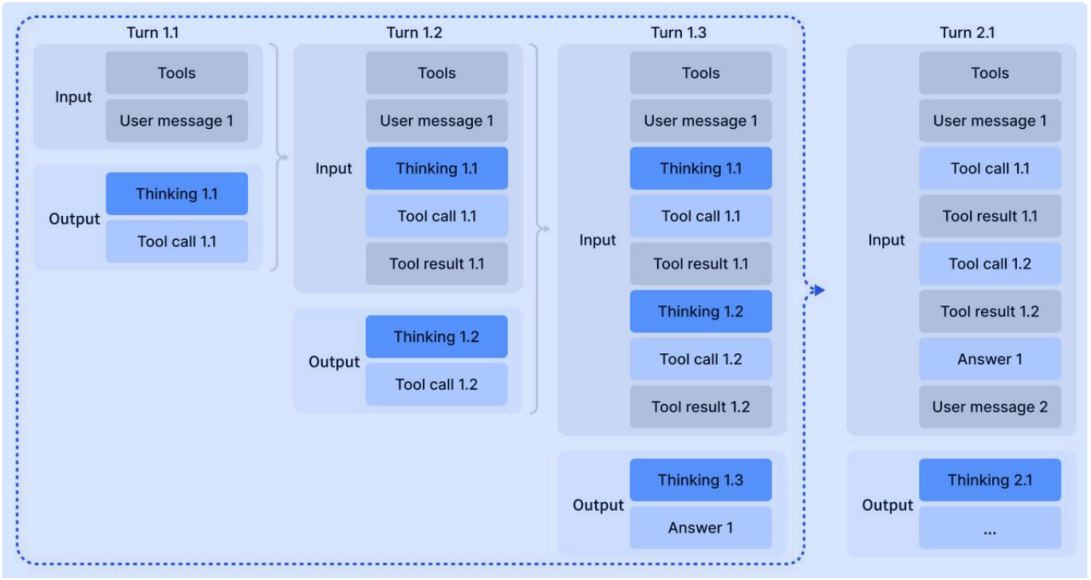
Thinking Context Management

问题：复制 DeepSeek-R1 的策略，在第二轮消息到达时丢弃推理内容，会导致严重的 token 效率低下问题。这种方法迫使模型在每次后续的工具调用中都要对整个问题进行冗余的重复推理。

方法：

1、只有当新的用户消息被引入对话时，历史推理内容才会被丢弃。如果仅添加工具相关消息（例如工具输出），则推理内容会在整个交互过程中保留。

2、当推理轨迹被移除时，工具调用的历史及其结果仍会保留在上下文中。



Results

	Benchmark (Metric)	Claude-4.5-Sonnet	GPT-5 High	Gemini-3.0 Pro	Kimi-K2 Thinking	MiniMax M2	DeepSeek-V3.2 Thinking
English	MMLU-Pro (EM)	88.2	87.5	**90.1**	84.6	82.0	**85.0**
	GPQA Diamond (Pass@1)	83.4	85.7	**91.9**	**84.5**	77.7	82.4
	HLE (Pass@1)	13.7	26.3	**37.7**	23.9	12.5	**25.1**
Code	LiveCodeBench (Pass@1-COT)	64.0	84.5	**90.7**	82.6	83.0	**83.3**
	Codeforces (Rating)	1480	2537	**2708**	-	-	2386
Math	AIME 2025 (Pass@1)	87.0	94.6	**95.0**	**94.5**	78.3	93.1
	HMMT Feb 2025 (Pass@1)	79.2	88.3	**97.5**	89.4	-	**92.5**
	HMMT Nov 2025 (Pass@1)	81.7	89.2	**93.3**	89.2	-	**90.2**
	IMOAnswerBench (Pass@1)	-	76.0	**83.3**	**78.6**	-	78.3
Code Agent	Terminal Bench 2.0 (Acc)	42.8	35.2	**54.2**	35.7	30.0	**46.4**
	SWE Verified (Resolved)	**77.2**	74.9	76.2	71.3	69.4	**73.1**
	SWE Multilingual (Resolved)	**68.0**	55.3	-	61.1	56.5	**70.2**
Search Agent	BrowseComp (Pass@1)	24.1	**54.9**	-	-/60.2*	44.0	**51.4/67.6***
	BrowseCompZh (Pass@1)	42.4	63.0	-	62.3	48.5	**65.0**
	HLE (Pass@1)	32.0	35.2	**45.8**	**44.9**	31.8	40.8
ToolUse	τ^2-Bench(Pass@1)	84.7	80.2	**85.4**	74.3	76.9	**80.3**
	MCP-Universe (Success Rate)	46.5	47.9	**50.7**	35.6	29.4	**45.9**
	MCP-Mark (Pass@1)	33.3	**50.9**	43.1	20.4	24.4	**38.0**
	Tool-Decathlon (Pass@1)	**38.6**	29.0	36.4	17.6	16.0	**35.2**

令牌效率仍然是未来研究的一个关键领域

Benchmark	GPT-5 High	Gemini-3.0 Pro	Kimi-K2 Thinking	DeepSeek-V3.2 Thinking	DeepSeek-V3.2 Speciale
AIME 2025 (Pass@1)	94.6 (13k)	95.0 (15k)	94.5 (24k)	93.1 (16k)	**96.0** (23k)
HMMT Feb 2025 (Pass@1)	88.3 (16k)	97.5 (16k)	89.4 (31k)	92.5 (19k)	**99.2** (27k)
HMMT Nov 2025 (Pass@1)	89.2 (20k)	93.3 (15k)	89.2 (29k)	90.2 (18k)	**94.4** (25k)
IMOAnswerBench (Pass@1)	76.0 (31k)	83.3 (18k)	78.6 (37k)	78.3 (27k)	**84.5** (45k)
LiveCodeBench (Pass@1-COT)	84.5 (13k)	**90.7** (13k)	82.6 (29k)	83.3 (16k)	88.7 (27k)
CodeForces (Rating)	2537 (29k)	**2708** (22k)	-	2386 (42k)	2701 (77k)
GPQA Diamond (Pass@1)	85.7 (8k)	**91.9** (8k)	84.5 (12k)	82.4 (7k)	85.7 (16k)
HLE (Pass@1)	26.3 (15k)	**37.7** (15k)	23.9 (24k)	25.1 (21k)	30.6 (35k)