

2026 · PREVIEW RELEASE

迈入百万上下文 普惠时代

DeepSeek-V4 预览版 · 论文与发布深度解读

1.6 T PARAMETERS · 1 M CONTEXT · OPEN WEIGHTS · FP4 QAT

CHAPTER 00 · 一页看懂

两款模型 · 一套架构 · 四个值得记住的数字

<div>CODEFORCES RATING</div> <div>3206</div> <div>V4-Pro-Max 竞赛编程分数， 超过 GPT-5.4 与全球人类选手第 23 名</div>	<div>PUTNAM 2025</div> <div>120/120</div> <div>形式化数学满分， 超过所有已知 AI 系统</div>	<div>NATIVE CONTEXT</div> <div>1M tokens</div> <div>Pro 与 Flash 原生支持， KV cache 只剩 baseline 的 2%</div>	<div>FLASH 缓存命中价</div> <div>0.2元/M</div> <div>V4-Flash 输入价， 同性能开源中几乎最便宜</div>
---	--	---	---

模型规格	API 价格（人民币 / M Tokens）			
	V4-Pro	V4-Flash	输入（缓存命中）	输入（未命中） 输出
总参数	1.6 T	284 B	deepseek-v4-pro1 元	12 元24 元
每 token 激活	49 B	13 B	deepseek-v4-flash0.2 元	1 元2 元
Transformer 层数	61	43	官方原话：受限于高端算力，目前 Pro 的服务吞吐十分有限，预计下半年昇腾 950 超节点批量上市后，Pro 的价格会大幅下调。	
MoE 路由专家	384	256		
训练 tokens	33 T	32 T		
原生上下文	1 M	1 M		

CHAPTER 00 · 定位反题

这不是一个冲破天花板的世界最佳模型的发布，
而是让普通人用上 *1M* 超长上下文 *Agent* 模型的发布。

说人话：前沿能力的赛道上，**Claude Opus 4.7** 和 **GPT-5.5** 已经跑在前面；V4 没想追上去。它选择的是把 1M 上下文、Agent 工具链和开源权重做到人人用得
起——普惠，比单点领先更稀缺。

THE FRONTIER IN 2026



CHAPTER 00 · 贯穿整份解读的四个观点

读 V4 这篇论文，请先记住*这四件事*

THESIS 01

不是世界最佳，
是**1M Agent** 的普惠

V4 不是冲破天花板的旗舰发布，而是第一次把 **百万 token 上下文 + Agent 能力** 做成了普通人能用得起的默认配置。

展开见 · CHAPTER 09

THESIS 02

竞赛选手基因，
擅长做题解题

DeepSeek 团队多数是竞赛获奖选手，模型也继承了这个画像。对 **有明确答案、可验证** 的数学与编程任务，它是当前开源的天花板。

展开见 · CHAPTER 07

THESIS 03

品味类任务
还差一截

创意写作胜率不如 Claude Opus 4.5，更不用说 4.6、4.7。论文自己在 Limitations 里承认了这个差距。标准化题强，开放题弱。

展开见 · CHAPTER 08

THESIS 04

整体是**最诚实、本分**的模型

论文里主动承认三个「有效但不理解原理」，价格页写清「昇腾 950 上市后 Pro 降价」。不演、不藏、不画大饼。

展开见 · CHAPTER 10

一份 72 页 · 11 幕的 V4 解读地图

00

开场与四个观点

定位反题 · 核心数据 · 全篇目录

00 – 04

01

两款模型 · 同一套架构

MoE · 稀疏激活 · 32T tokens · 稳定性

05 – 10

02

架构① mHC 残差升级

双随机矩阵 · Sinkhorn-Knopp 迭代

11 – 16

03

架构② 混合注意力 CSA + HCA

粗细配合 · 1M 下 KV cache 只剩 2%

17 – 24

04

架构③ Muon 优化器

正交化更新 · 替代 AdamW

25 – 29

05

基础设施

TileLang · FP4 QAT · 磁盘 KV cache

30 – 36

06

训练范式革命

Specialist + On-Policy Distillation · DSec 沙盒

37 – 43

07

擅长什么 · 观点 ②

竞赛基因 · Codeforces 3206 · Putnam 满分

44 – 50

08

不擅长什么 · 观点 ③

创意写作差 · HLE w/tools 倒数 · 品味差

51 – 58

09

普惠的真相 · 观点 ①

1M 不是能用是不再省字 · 价格屠夫

59 – 65

10

诚实与本分 · 观点 ④

论文自认局限 · 昇腾 950 · 荀子金句

66 – 70

11

收尾

整体定位 · 鸣谢

71 – 72

CHAPTER 01 · 两款模型 · 同一套架构

一个对标闭源旗舰，一个重新定义开源性价比

旗舰 · FLAGSHIP

V4-Pro

Redefining the state-of-the-art for open models.

总参数

1.6T

TRANSFORMER 层

61

每 TOKEN 激活

49B

MOE 专家数

384

面向：性能对标 Opus 4.6、GPT-5.4；竞赛编程与形式化数学做到开源 SOTA；适合需要最强推理能力、可接受较高 API 成本的场景。

轻量 · COST-EFFICIENT

V4-Flash

Closed-model reasoning, open-model economics.

总参数

284B

TRANSFORMER 层

43

每 TOKEN 激活

13B

MOE 专家数

256

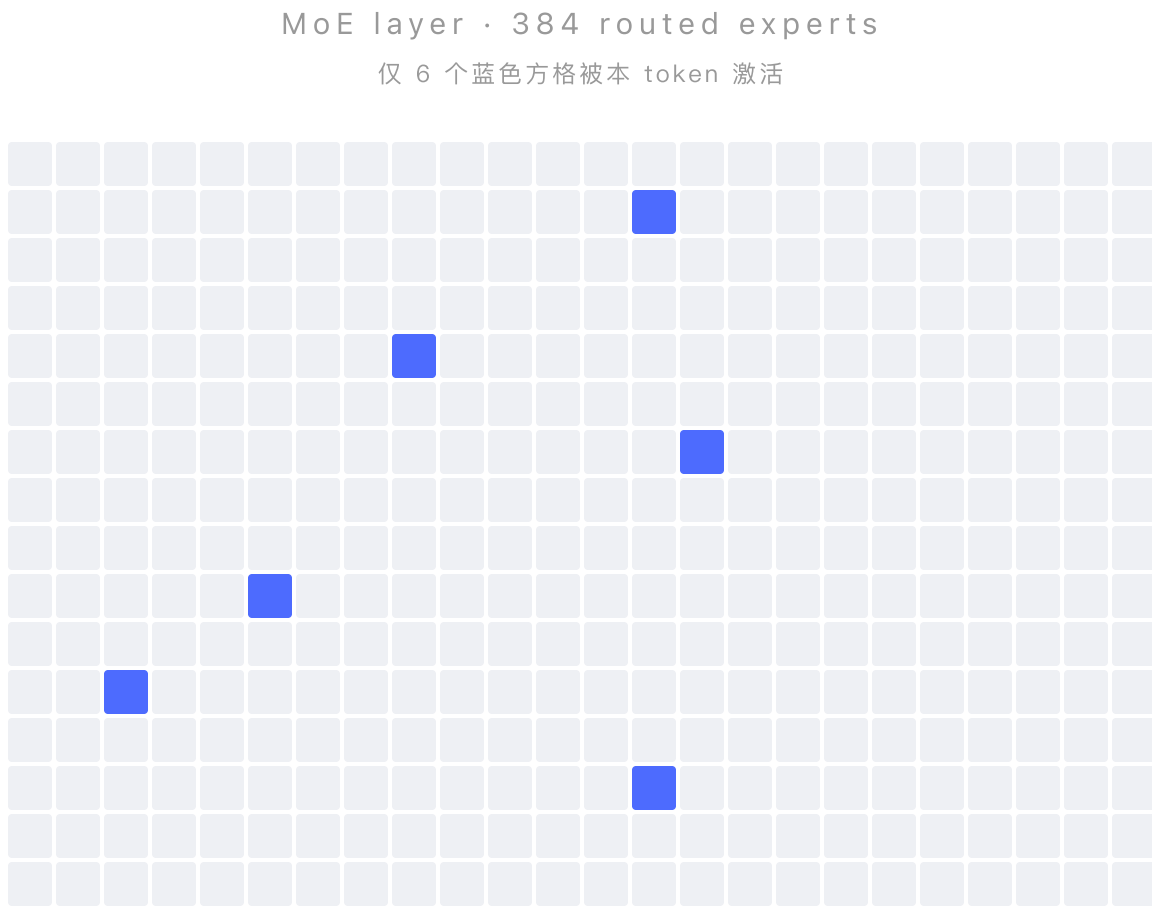
面向：用更少参数反超 V3.2-Base；输入缓存命中 0.2 元/M，真正的价格屠夫；适合大批量 agent、长文档处理、需要普惠部署的场景。

CHAPTER 01 · MOE 本质

1.6 T 参数，每次只请 6 个专家上场

MoE 的本质是专家会议，不是全员表决。
模型有 1.6 T 参数是团队规模，每 token 激活 49 B 才是真正开口说话的人。

路由机制： V4-Pro 的每一层有 384 位路由专家 + 1 位共享专家，每个 token 由门控网络挑选 6 位最相关的专家处理。前 3 层用 Hash routing 稳住早期训练；后续层用 Sigmoid → Sqrt(Softplus) 的打分函数。
代价： 部署时要把整个 1.6 T 模型加载进显存（因为任何 token 都可能命中不同专家），但每步计算只有 49 B 的量。



6 experts activated · 49 B params

OUT OF 1.6 T TOTAL

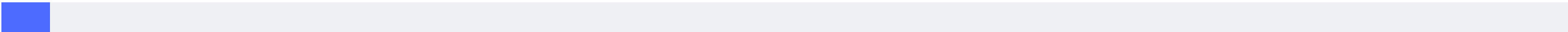
CHAPTER 01 · 稀疏激活的经济学

花*存储 1.6 T*的钱 · 享受 *49 B* 的推理速度

V4-Pro

每 token 激活 / 总参数

3.1 %

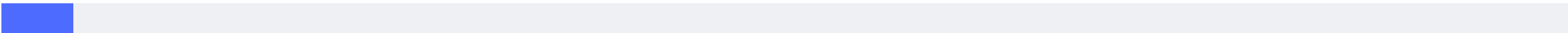


49 B ÷ 1.6 T · 只有传统 dense 模型 3% 的计算成本，走全部参数的知识密度

V4-Flash

每 token 激活 / 总参数

4.6 %

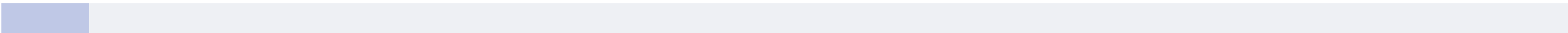


13 B ÷ 284 B · 推理几乎等于 13 B dense 模型，但知识面接近 284 B

DeepSeek-V3

前代对比

5.6 %



37 B ÷ 671 B · V4-Pro 的稀疏度比 V3 更激进

每个 token 只激活 *6 个路由专家 + 1 个共享专家*。前 3 层用 Hash routing 稳住早期训练，后续层由门控网络动态选路。

数据不是更多就好 · 怎么不坍塌才是关键

33T
V4-PRO 预训练 TOKENS / V4-FLASH 32 T

DATA 01

过滤 AI 生成内容

互联网语料里已有大量 ChatGPT/Claude 写出的文本，训练前明确识别并过滤，防止模型坍塌。

DATA 03

扩充多语言长尾

中文之外的小语种被有意识地补足，避免「高资源语种吃光，长尾跟不上」的常见失衡。

DATA 02

Mid-training 引入 Agentic 数据

工具调用轨迹、多步推理、搜索片段——把 Agent 行为在预训练阶段就灌进模型，不是靠后训练硬掰。

DATA 04

精选科学论文与技术报告

高质量技术文献被单独拎出来 up-sample，直接支撑了后续在 科学推理 上的表现。

关键技术细节：启用 **sample-level attention masking**，避免 pack 到同一序列里的样本互相「串味」——这是 32T 大规模训练的必要基础设施。

1.6 T 模型怎么**不炸**：两个「有效但不理解原理」的救命 **trick**

TRICK 01

Anticipatory Routing 预判式路由

训练中 loss 突然飙高时，路由索引被拉偏。V4 的方案是用 **前 Δt 步的历史参数 $\theta_{t-\Delta t}$** 重新估算路由，只在 spike 出现时触发。

开销：约 20% 的 routing 算力，但平摊到整体训练几乎可忽略。

TRICK 02

SwiGLU Clamping 门控限幅

SwiGLU 的 linear 分量偶尔会飞到极端值导致数值爆炸。V4 的方案很朴素：把 linear 分量 **clamp 到 $[-10, 10]$** ，gate 分量上界 10。

代价：理论上截断了梯度信号，但实测稳定性远胜代价。



These two tricks are empirically effective, but we still lack a principled understanding of why they work.

— DEEPSEEK-V4 TECH REPORT, §4.2.3

CHAPTER 01 · 详细规格

两款模型 · 放在一起看

配置项	V4-Pro FLAGSHIP	V4-Flash EFFICIENT
模型规模		
总参数	1.6 T	284 B
每 token 激活参数	49 B	13 B
Transformer 层数	61	43
隐藏维度 d	7,168	4,096
MOE 配置		
路由专家数 / 层	384	256
每 token 激活专家	6 + 1 shared	6 + 1 shared
前 N 层使用 Hash Routing	3	3
训练配置		
预训练 tokens	33 T	32 T
峰值学习率	2.0×10^{-4}	2.7×10^{-4}
每步 batch size	94.4 M tokens	75.5 M tokens
优化器（主干 / 辅助）	Muon / AdamW	Muon / AdamW
上下文与精度		
原生上下文长度	1 M	1 M
Attention 热身阶段切稀疏	64 K 处	64 K 处
MoE 专家权重精度	FP4	FP4
KV Cache 精度	FP8	FP8

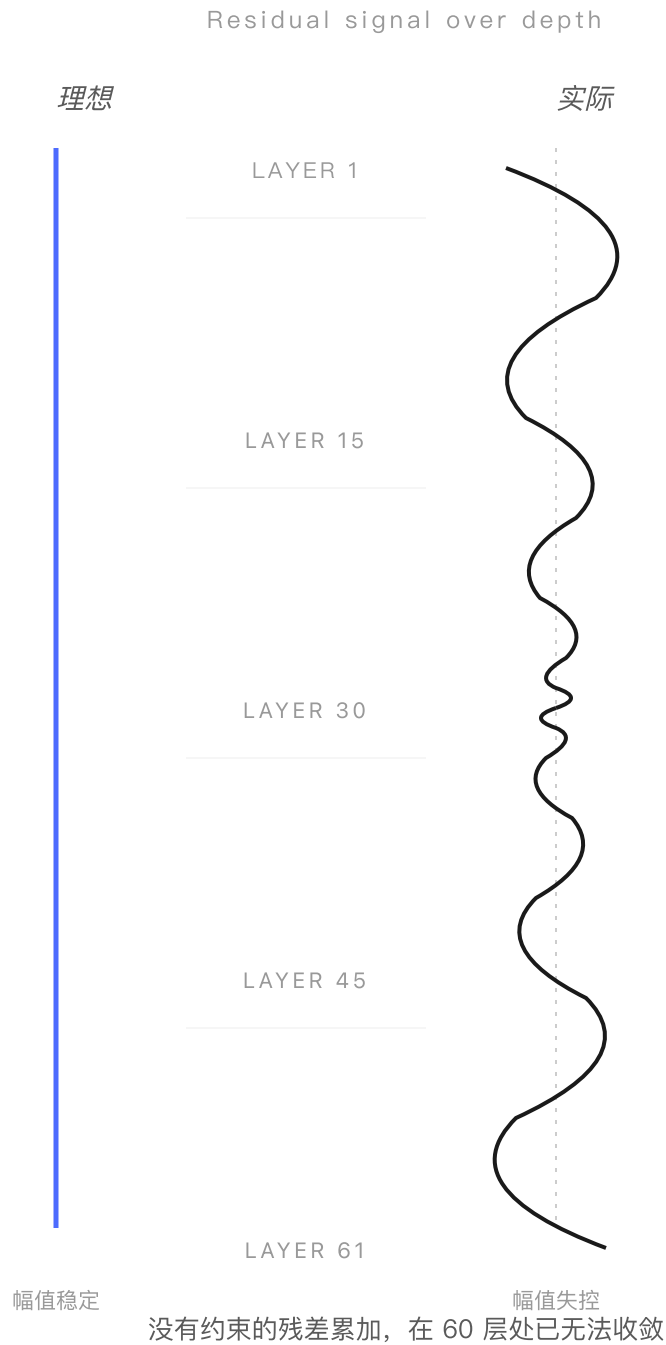
CHAPTER 02 · 残差连接的瓶颈

深层堆叠时，信号越传越失真

残差连接（ResNet）之所以有效，是因为它让信号可以「跳级」传播。但当模型深到 60 多层，跳级路径本身也开始积累噪声和放大——梯度要么消失、要么爆炸。

传统做法的局限：残差连接默认每条旁路的「权重」都是 1，等价于让所有层都同等重要。但现实中某些层就是比另一些层关键，不加约束地累加会让深层模型的训练进入危险区。

V4 的立意：不推翻残差连接，而是在它之上加一层 **数学护栏**——Manifold-Constrained Hyper-Connections（mHC）。下一页展开。



CHAPTER 02 · MHC 核心思想

给每条残差路径加一个水闸

“传统残差连接让所有层的信号自由累加。
mHC 把这些累加权重约束在一个双随机矩阵流形上——每条输入路径的贡献之和固定为 1，每条输出路径的权重之和也固定为 1。

INTUITION · 说人话

想象每一层之间有一张水网：每个入口最多放 1 单位水进去，每个出口最多接 1 单位水出来。不管网络多深，总水量永远守恒——这就是「双随机」。护栏不是软约束，是数学意义上的不可破坏。

KEY PROPERTY 01

谱范数 ≤ 1

双随机矩阵的谱范数天然 ≤ 1 ，信号幅值不会被放大——深层堆叠不再爆炸。

KEY PROPERTY 02

乘法封闭

双随机矩阵的乘积仍是双随机矩阵，层层叠加后约束自动保持——护栏不需要反复校准。

KEY PROPERTY 03

可学可微

通过 Sinkhorn-Knopp 迭代将任意矩阵投影到这个流形上，反向传播可直接优化。

双随机矩阵的两个约束

DOUBLY STOCHASTIC CONSTRAINT

$$W \in \mathbb{R}^{n \times n}, \quad W_{ij} \geq 0$$

$$\sum_j W_{ij} = 1 \quad (\text{每行和为 } 1)$$

$$\sum_i W_{ij} = 1 \quad (\text{每列和为 } 1)$$

$$\Rightarrow \|W\|_2 \leq 1 \quad \text{谱范数有界}$$

$$\Rightarrow W_1 \cdot W_2 \quad \text{仍是双随机矩阵}$$

INSIGHT 01 · 为什么要约束「行和 = 1」

入度守恒

每个下游节点收到的信号权重之和固定为 1, 所有来源贡献之和不会被无限放大。

INSIGHT 02 · 为什么要约束「列和 = 1」

出度守恒

每个上游节点流出的信号权重之和也固定为 1, 能量不会在某条路径上独占。

INSIGHT 03 · 组合起来意味着什么

深层堆叠天然稳定

每一层的残差 transform 都落在这个流形上, 61 层连乘仍然不放大、不衰减。

CHAPTER 02 · MHC 的代价

每一层要投影回流形 · 这不是免费的

SINKHORN-KNOPP 迭代次数 / 层 / 前向

20iters

把任意矩阵投影到双随机流形上的经典算法，交替归一化行和列 **20** 次，足以让误差衰减到可忽略。

工程层面的摊销

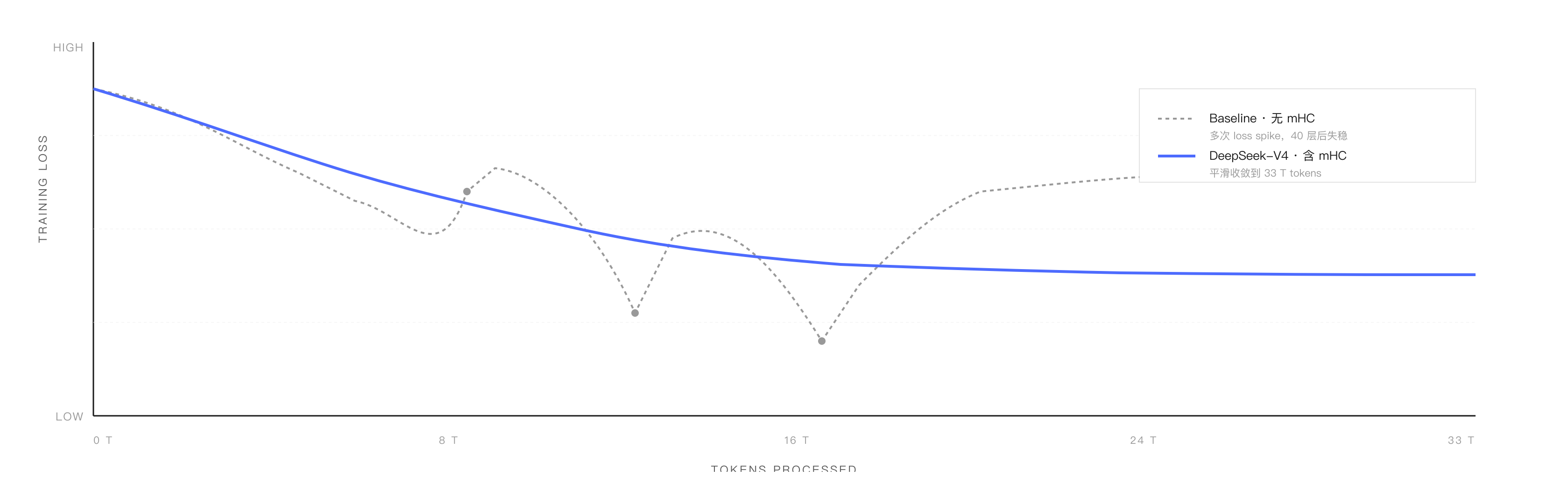
1F1B pipeline 中的时间占比	6.7%
借助 fused kernel 合并投影	✓
借助重计算省显存	✓
最终训练吞吐损失	≈ 5%

代价看起来不小，但考虑到 **深层稳定性的回报**（否则连训练都跑不完），这是极其划算的交易。

mHC 不是性能游戏，是 **可训练性** 游戏——让 1.6 T · 61 层的模型从「训不出来」变成「稳稳训下来」。

CHAPTER 02 · MHC 效果

Loss 曲线再也不炸飞



TAKEAWAY 01

baseline 在 8 T tokens 后出现 **3 次显著 loss spike**，每次都需要重启或跳批

TAKEAWAY 02

加入 mHC 后，**61 层网络全程稳定**，完整跑完 33 T 预训练 budget

TAKEAWAY 03

本页曲线为复刻论文 §2.2 Fig.3 的**示意**，具体数值请查 PDF

02 MHC · RESIDUAL BACKBONE 升级

03 HYBRID ATTENTION · CSA + HCA

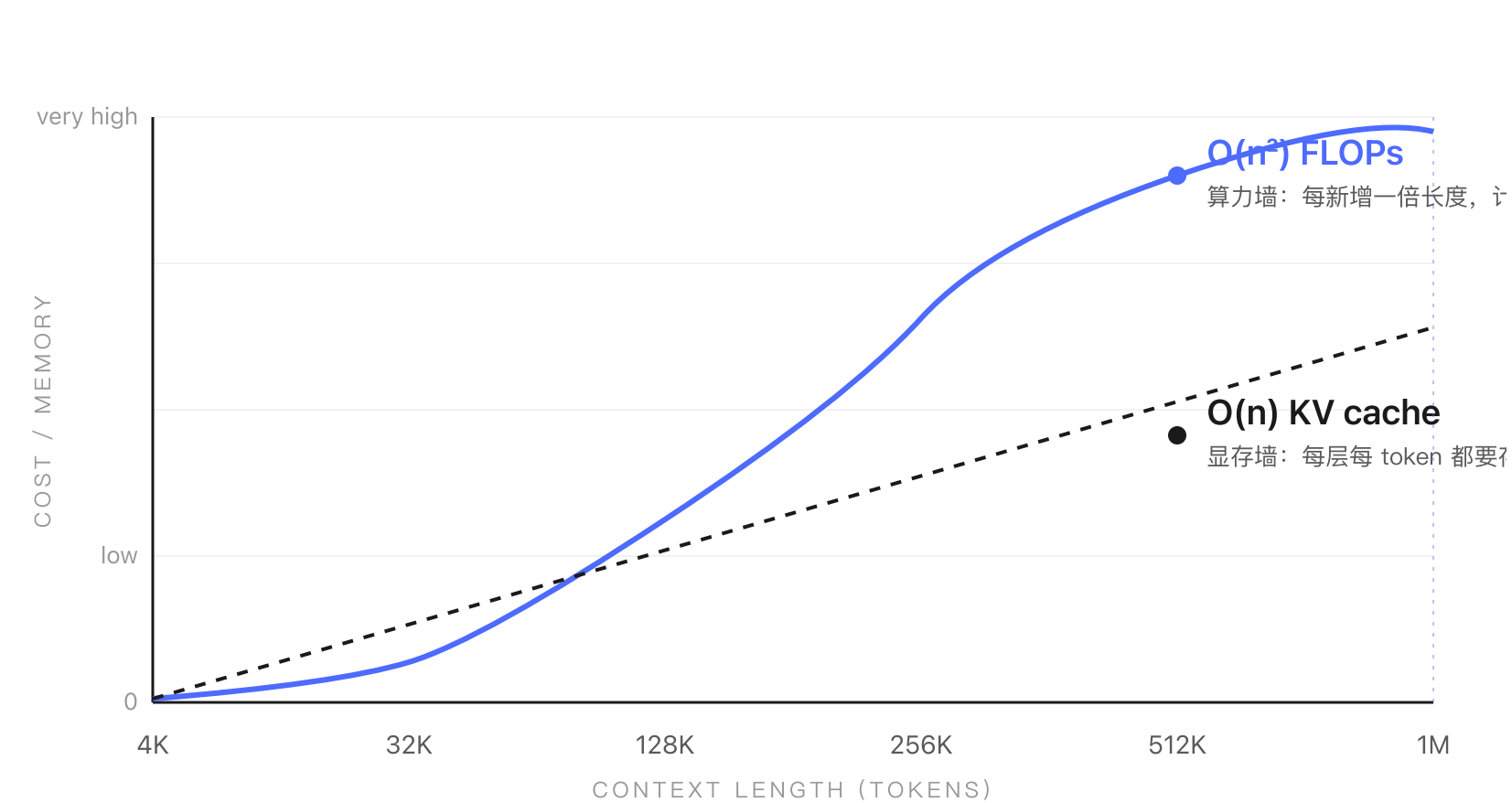
解决了深度， 接下来解决长度

mHC 让 61 层能稳稳叠起来。下一幕的问题是：当上下文从 4K 推到 1M，注意力机制怎么不被 KV cache 拖垮。

CHAPTER 03 · 混合注意力 CSA + HCA

上下文一变长，*有两堵墙会同时砸过来*

算力墙和显存墙，都在 n 这一个变量上爆炸增长。这正是百万 token 默认配置难以落地的根本原因。



示意曲线 · 原生 1M CONTEXT 下，两条曲线同时抵达天花板

BOTTLENECK · 1

$O(n^2)$ 算力

标准 self-attention 中，每个 query 要和每个 key 算一次内积。上下文翻倍，FLOPs 变四倍——把 4K 换成 1M，相当于多算了六万倍。

BOTTLENECK · 2

$O(n)$ 显存

每一层、每一个 token 的 K 和 V 都要留在显存里供后续 query 读。100 万 token 乘以几十层，KV cache 就会吃光单卡整机。

接下来三节要回答的就是一个问题：既然不能都留着，那该留什么、怎么留？V4 的答案是把注意力拆成两种，各管一边。

CHAPTER 03 · 混合注意力 CSA + HCA

两把筛子，一粗一细，一起用

查资料的人都知道：先翻目录定位到大概章节，再精读一两段细节。V4 把注意力层交替排布，正是这个动作的机器版本。

细筛 · FINE

CSA Compressed Sparse Attention

压缩 + 稀疏，定位关键 token

按 $m = 64$ 的粒度把 KV 压缩成块，再用 Lightning Indexer 打分、只挑 top-k 个最相关的块做精算。相当于翻完索引后，只盯着真正要用的那几段精读。

粗筛 · COARSE

HCA Heavily Compressed Attention

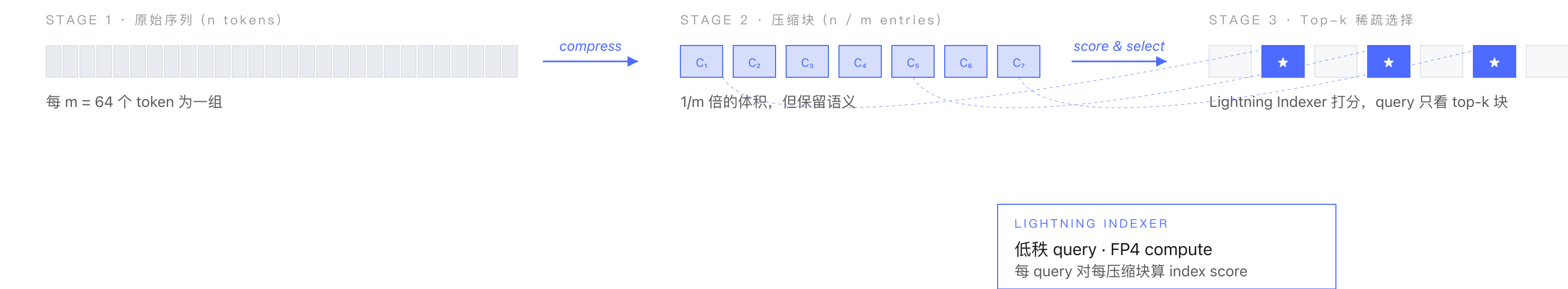
激进压缩，但扫全局

以更大的粒度 $m' \gg m$ 把序列狠狠压扁，但不做稀疏筛选，每个 query 都要 dense 地把所有压缩块扫一遍。粒度粗了，但全局视野不丢。

CHAPTER 03 · CSA · COMPRESSED SPARSE ATTENTION

把 64 个 token 揉成 1 块，再挑出最相关的几块精读

压缩 + 稀疏两次降维，既省算力又省显存。Lightning Indexer 负责打分，只让真正要紧的块走进 core attention。



STEP 01 · COMPRESS

每 64 个 token 合成 1 个 entry

用可学习的 compression weights 加权求和，序列长度降到 n / m 。两路 KV (C^a、C^b) 配合带 overlap 的 stride，保留边界信息。

STEP 02 · INDEX

Lightning Indexer 打分

low-rank 投影出 indexer query，对每个压缩块算 index score $I_{t,s}$ 。计算用 FP4，拿速度换精度里那 2% 左右的差异。

STEP 03 · SELECT

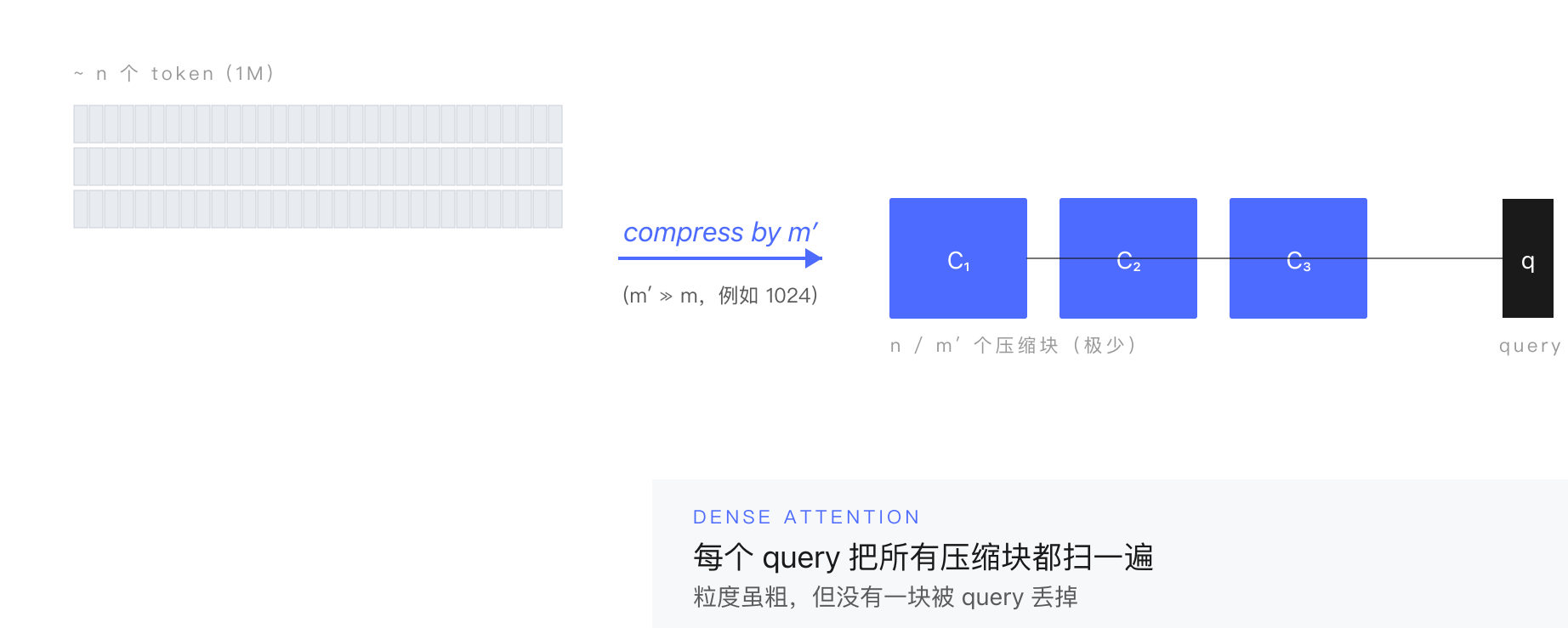
只让 top-k 走进 core attention

按 index score 选 top-k 压缩块做 DSA (DeepSeek Sparse Attention)。压缩给了 1/m 倍，稀疏再给了 $k / (n/m)$ 倍，双重节省。

CHAPTER 03 · HCA · HEAVILY COMPRESSED ATTENTION

激进压缩，但保留全局 *dense* 视野

m' 远大于 m ，序列被狠狠压扁；但不做稀疏筛选，每个 query 都要把所有压缩块扫一遍——牺牲细粒度，换全局视野。



示意 · 压缩率 $M' \gg M$ ，压缩后块数极少，QUERY 做 **FULL ATTENTION**

一个设计直觉：**粗的那一路必须 dense**——如果粗粒度再稀疏一次，全局视野就彻底断了，和 CSA 的功能会重叠。

COMPRESSION

粒度 $m' \gg m$

同样的压缩机制，但步长放大一个数量级（论文未披露具体 m' ，参考实现中 $m' = 1024$ 量级）。 n / m' 个压缩块就几十到几百个。

ATTENTION

Dense · 不走稀疏

没有 Lightning Indexer，query 对所有压缩块做完整的 MQA。块数本来就少，dense 的代价可以接受。

ROLE

粗粒度全局信息兜底

承担「整篇文档的宏观理解」——跨段、跨章节的长程依赖。和 CSA 层交替排布，形成一粗一细的双通道。

CHAPTER 03 · 那些让 CSA/HCA 真正好用的细节

主结构之外，*还有四个决定成败的小零件*

单看每一项都不起眼，但少了任何一个，百万上下文的表现都会崩一块。CSA 和 HCA 共享这套底座。

01 SHARED KV · MQA 所有 head 共用一份 KV

每个压缩 KV entry 同时充当所有 attention head 的 key 和 value。**head 数越多，省得越多**——配合压缩本身，KV cache 又小了一个量级。

03 ATTENTION SINK 给 query 一个「弃权」出口

每个 head 配一个可学习的 sink logit，加到 softmax 的分母里。让 attention 总和可以不等于 1、甚至接近 0——**query 在没话可说时就弃权**，不强行关注噪声。

02 SLIDING WINDOW 最近 n_win 个 token 不压缩

压缩会吃掉局部细节——但最近的 token 恰恰最重要。额外开一个滑动窗口分支，**不压缩地保留最近 n_win 个 token 的 KV**，和压缩块一起进 core attention。

04 PARTIAL ROPE RoPE 只作用在最后 64 维

其余维度保留无位置信息的通用表示。配合 output 上用 `position = -i` 再做一次 RoPE 的 trick，**把绝对位置偷偷换成相对位置**，百万长度下依然稳定。

CHAPTER 03 · 精度栈

省位宽，*但只在可以省的地方省*

越粗糙的任务用越少的比特，越精细的环节留高精度。结果是同一个 attention 里并存三种数值格式。

Indexer QK

只打相关性分

FP4

4 bit · 容错最高

KV cache

主体存储

FP8

8 bit · 主体存一半

RoPE 维度

位置编码敏感

BF16

16 bit · 位置保真

Core compute

主体 attention 计算

BF16

16 bit · 精算

混合精度的选择哲学

Attention 里并不是所有数值都同等重要。打分可以糙，选好了再精算——这就是 V4 把 indexer 做到 FP4、core attention 留 BF16 的原因。

RULE · 1

粗粒度判断 → 低比特

RULE · 2

存储 → 中等比特

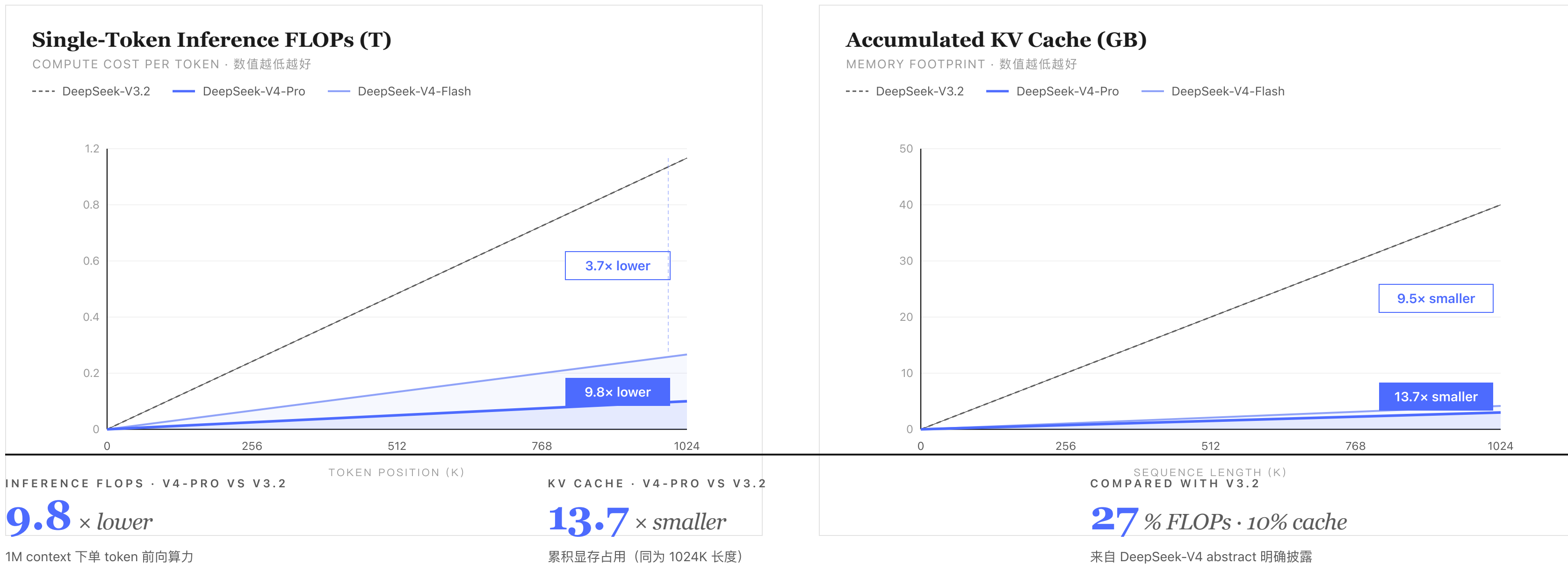
RULE · 3

位置 / 最终计算 → 高比特

CHAPTER 03 · EFFICIENCY DISCUSSION

V3.2 的曲线还在往上爬，V4 的曲线几乎是贴着地面走

同样推到 1024K token 长度，DeepSeek-V3.2 的 FLOPs 和 KV cache 线性攀升；V4-Pro 的曲线被压扁到贴近 x 轴。差距来自 CSA/HCA 双压缩 + FP4 indexer + 混合精度存储。



CHAPTER 03 · DIET SIGNAL

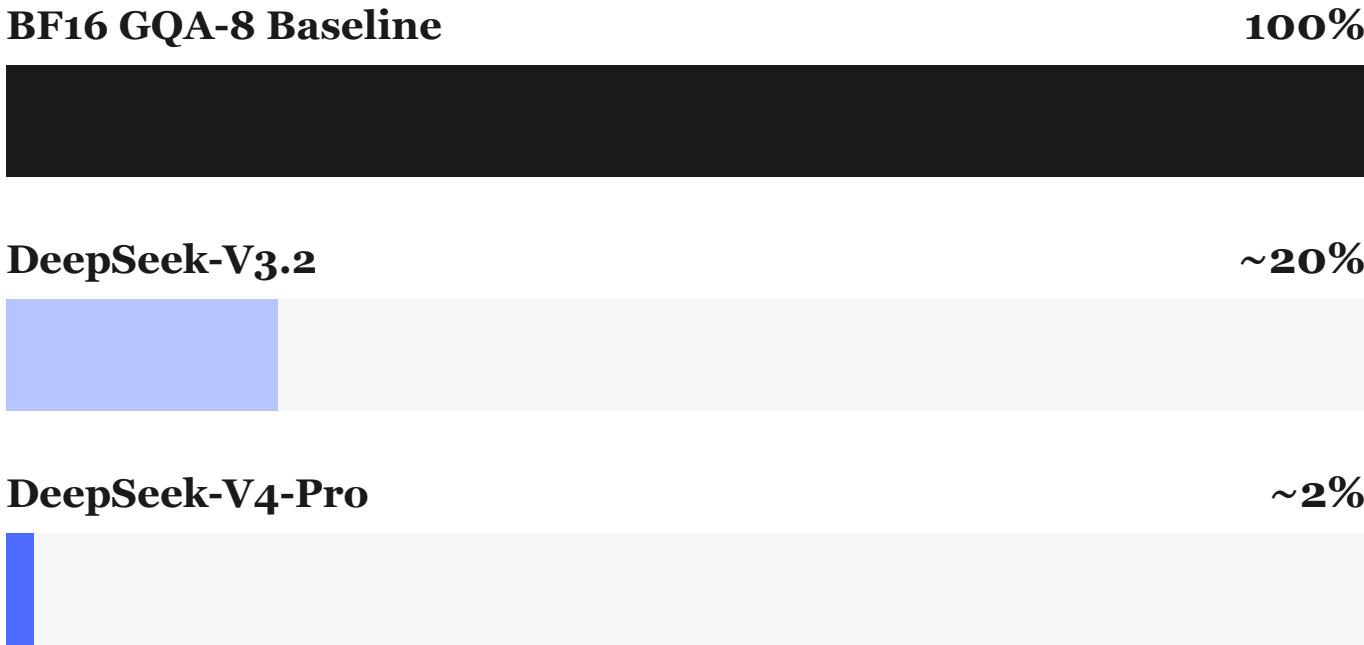
KV cache 只剩 *baseline* 的 2%

VS BF16 GQA-8 BASELINE · 1M CONTEXT

~2%

在最常见的 **BF16 GQA-8 (head dim = 128)** 对照组下，V4 系列把 1M 上下文的 KV cache 压到只剩 2% 量级。

VISUAL RATIO · 相对 BASELINE



这 2% 不是单点优化的战果，是 **CSA/HCA 双压缩 + Shared KV MQA + FP8 主存 / BF16 位置维** 四件事叠加的结果。说人话：把 50 份缩成 1 份，百万上下文就能从机房搬进一张卡。

CHAPTER 04 · MUON 优化器

AdamW 的毛病不在慢，在偏科

每个参数独立调学习率，梯度差距大的方向迟早被拉开。大模型规模越大越明显。

ANALOGY

像健身时**只练一侧肌肉**：强的越练越强，弱的几乎没刺激，最后整体动作反而变形。

01 每维度独立归一化

AdamW 对每个参数除以它自己历史梯度的平方根。这让大梯度方向被压缩，小梯度方向被放大，但各维度之间的相对关系被悄悄改写。

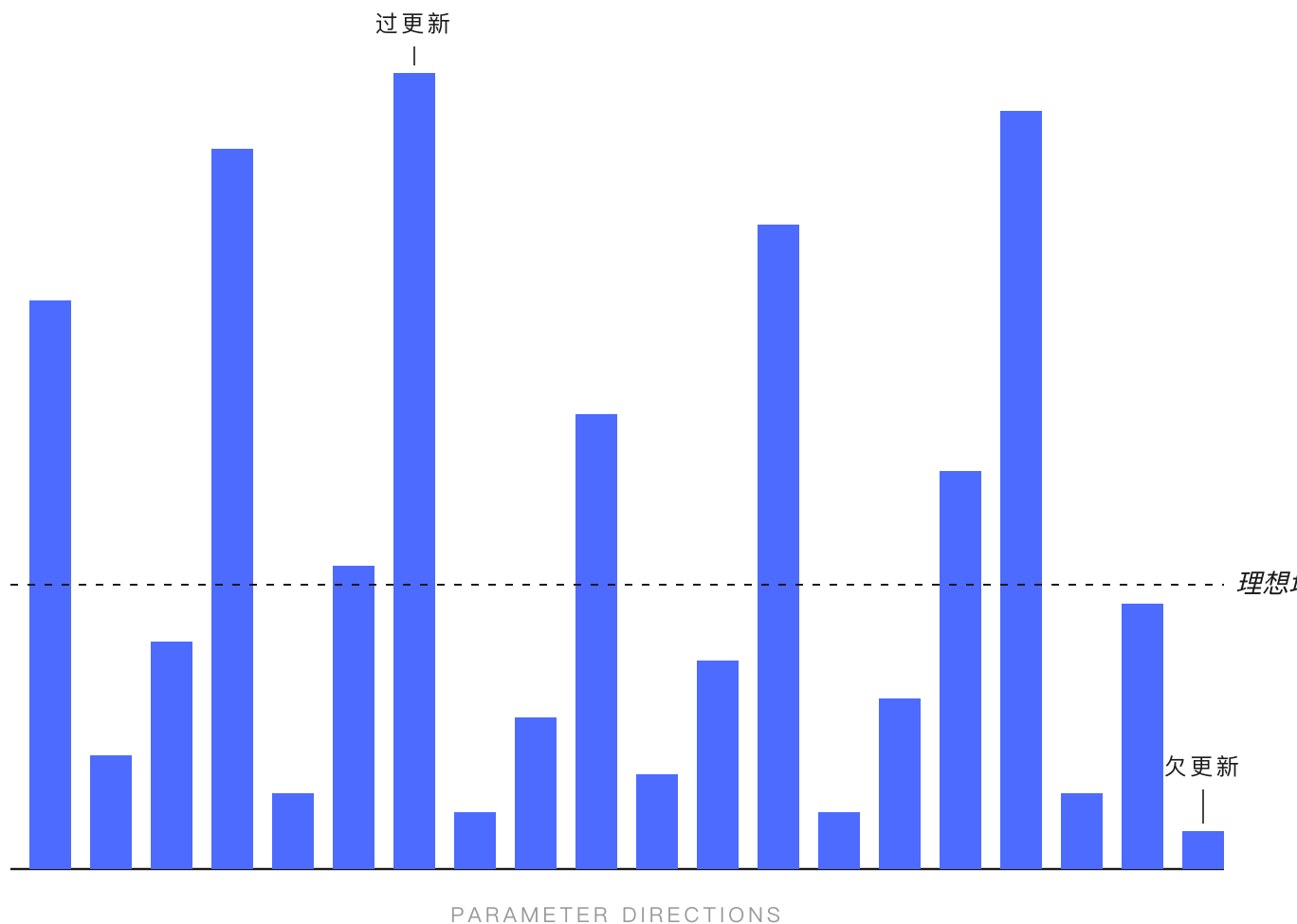
02 某些方向过更新

权重矩阵里存在「奇异方向」——梯度天然偏大的主奇异向量。AdamW 容易顺着它们反复推，把整个矩阵推到病态。

03 另一些方向几乎不动

相应地，信号弱的方向学不起来。放到 1.6T 参数的 V4-Pro 上，这种方向不均衡会直接影响收敛速度和最终 loss。

示意 · ADAMW 的更新强度分布



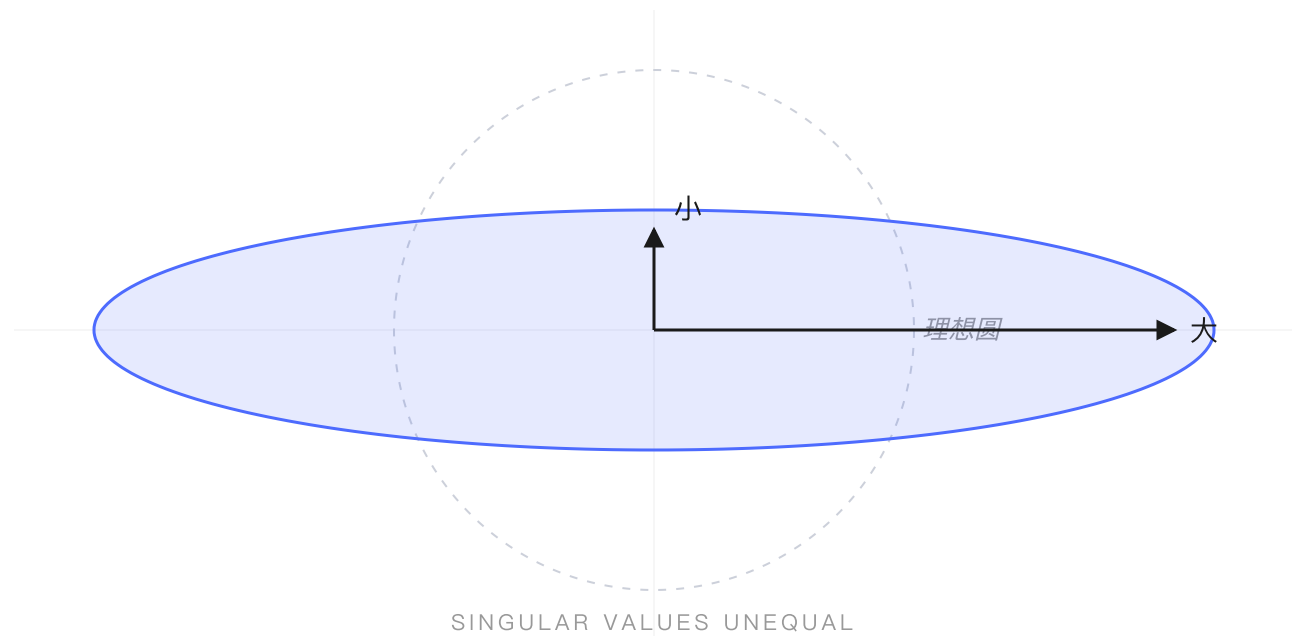
CHAPTER 04 · MUON CORE IDEA

先拿走方向间的不平衡，再去更新参数

AdamW 直接拿 momentum 更新参数；Muon 先把 momentum 矩阵投影成正交矩阵，让所有方向的步长一样大。

BEFORE · ADAMW

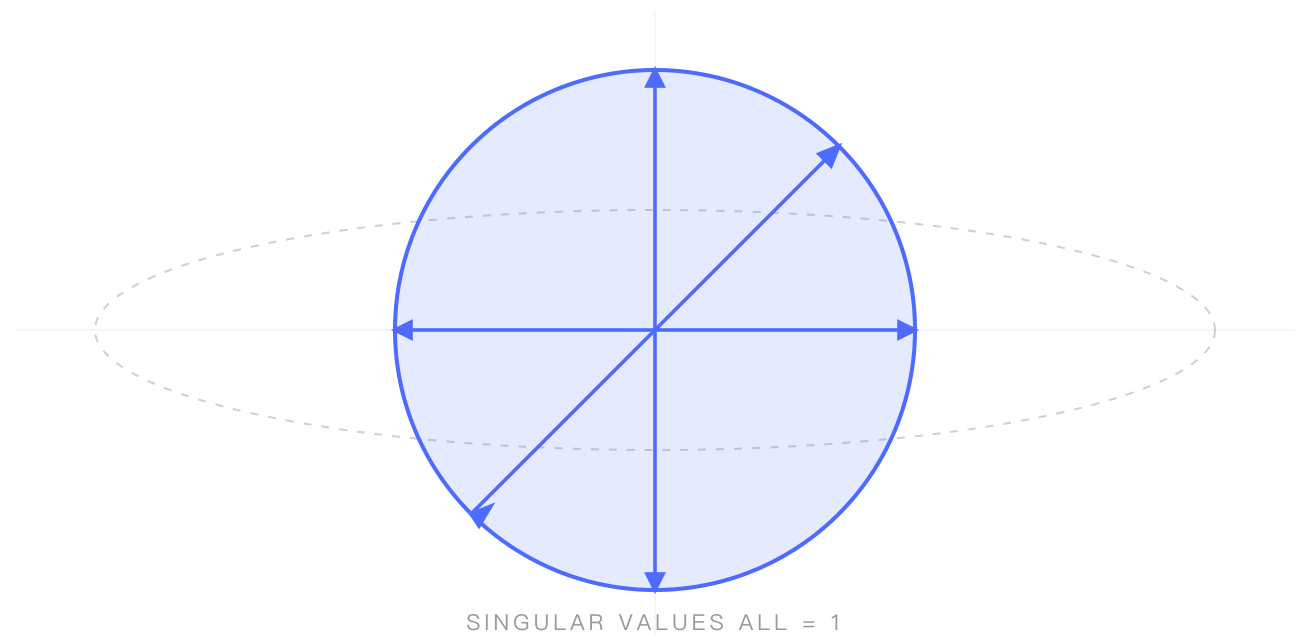
更新矩阵是个拉长的椭圆



SGD + momentum 算出的矩阵 \mathbf{M} 的奇异值参差不齐。做 SVD 会得到 $\mathbf{M} = \mathbf{U} \Sigma \mathbf{V}^T$ —— Σ 是对角阵，对角线上那些值就是每个方向的「步长」。

AFTER · MUON

正交化后变成一个圆



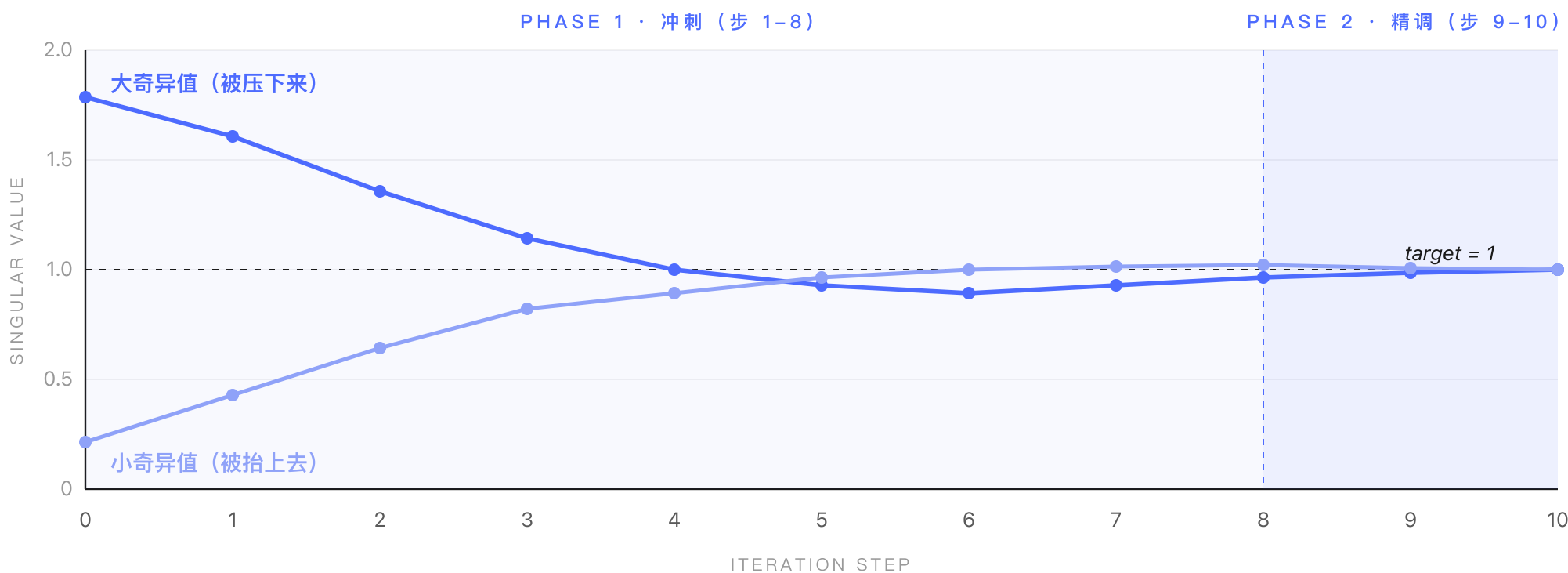
Muon 把 Σ 替换成单位阵： $\mathbf{M} \rightarrow \mathbf{U} \mathbf{V}^T$ 。所有方向同等步长，训练在参数空间里向外均匀扩张，而不是沿一条窄道滑下去。

这件事直接做 SVD 代价太大，Muon 用 [Newton-Schulz 迭代](#) 近似投影——下一页展开。

CHAPTER 04 · 10 步迭代的两段式

前 8 步冲刺把奇异值拉到 1 附近，后 2 步精调稳定

直接做 SVD 开销太大，Newton-Schulz 用多项式迭代近似。V4 的 hybrid 版本把 10 步分两段，换不同的系数。



示意 · 所有奇异值在 10 步内汇聚到 1，从 大 / 小 两侧逼近

注 · Newton-Schulz 是一个只用矩阵乘法的多项式迭代，每步把 M 替换成 $aM + b(MM^T)M + c(MM^T)^2M$ 。合理选择 $a/b/c$ ，迭代就能把 M 的所有奇异值逐步推向 1，从而无需真正做 SVD 就能完成正交化。

PHASE 1 · STEPS 1-8

冲刺：把奇异值快速拉到 1 附近

$(a, b, c) = (3.4445, -4.7750, 2.0315)$

系数绝对值偏大，迭代对偏离 1 的奇异值响应剧烈，代价是会 overshoot——但越界无所谓，继续迭代就好。

PHASE 2 · STEPS 9-10

精调：把值精确钉在 1

$(a, b, c) = (2, -1.5, 0.5)$

系数温和、单调，用来收尾。让所有奇异值稳定精确地停在 1，避免 Phase 1 遗留的微小抖动。

CHAPTER 04 · 不用重调超参

换优化器，*不换学习率*

Muon 的更新幅度和 AdamW 不是同一个量纲，原本要重新扫学习率。V4 加一次 RMS rescale 把它们拉回同一规模。

WITHOUT RESCALE

直接替换 = 超参全部重扫

$$\mathbf{O}_t = \text{HybridNewtonSchulz}(\mu \mathbf{M}_{t-1} + \mathbf{G}_t)$$

正交化后，更新矩阵的每个元素幅度由矩阵维度决定—— $n \times m$ 越大，RMS 越小。这意味着原来 AdamW 下调好的学习率、weight decay 到了 Muon 上全得重调。

代价：一次大规模调参实验，几万卡时起步

WITH RESCALE · V4 的做法

乘一个 $\sqrt{\max(n,m)} \cdot \gamma$

$$\mathbf{O}_t = \mathbf{O}'_t \cdot \sqrt{\max(n, m)} \cdot \gamma$$

把正交矩阵的 RMS 主动调回和 AdamW 更新相近的量级。这样 η 、 λ 这些超参 **可以直接复用 AdamW 的经验值**——不需要为 Muon 重新跑一轮调参。

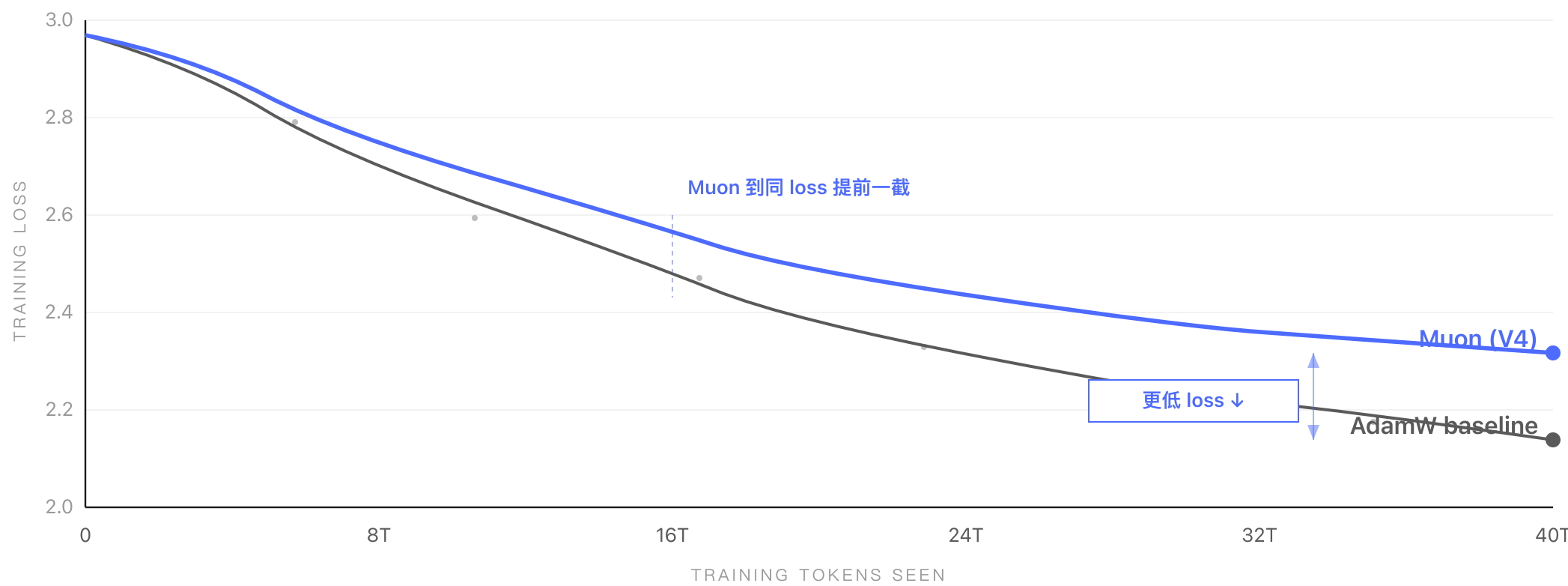
好处：复用 V3 时代的调参经验，直接起训

一个不起眼的细节，但在工程上非常关键：它让换优化器这件事，从「重新探索」变成「直接上车」。对千卡起步的训练，这省下的就是整条关键路径。

CHAPTER 04 · RESULTS

同样的数据、同样的算力，*Muon* 先一步到位

V4 论文未披露完整对照曲线，以下为根据 Jordan 2024 / Liu 2025 原始 Muon 论文和 V4 训练描述综合示意。



示意 · MUON 曲线更平滑，下降更快 · 论文未披露完整训练曲线，此图为趋势示意

V4 TEAM'S TAKEAWAY

收敛更快、训练更稳

FASTER

收敛速度

同样 loss 水平，Muon 比 AdamW 少吃一截 token。V4 论文明确列为换优化器的核心动机。

SMOOTHER

训练稳定性

正交化让梯度主方向不再垄断更新，loss 曲线抖动明显收敛，大规模 MoE 训练里尤其明显。

NO QK-CLIP

省掉配套 trick

V4 对 query/KV 都做了 RMSNorm，attention logit 天然不炸，不需要 Muon 原版论文的 QK-Clip。

范围说明：Embedding、prediction head、静态 bias、RMSNorm 这几类仍用 AdamW；其余模块全部换 Muon。这是工程折衷——小部件继续用经过千锤百炼的老家伙，主力换新。

CHAPTER 05 · INFRASTRUCTURE

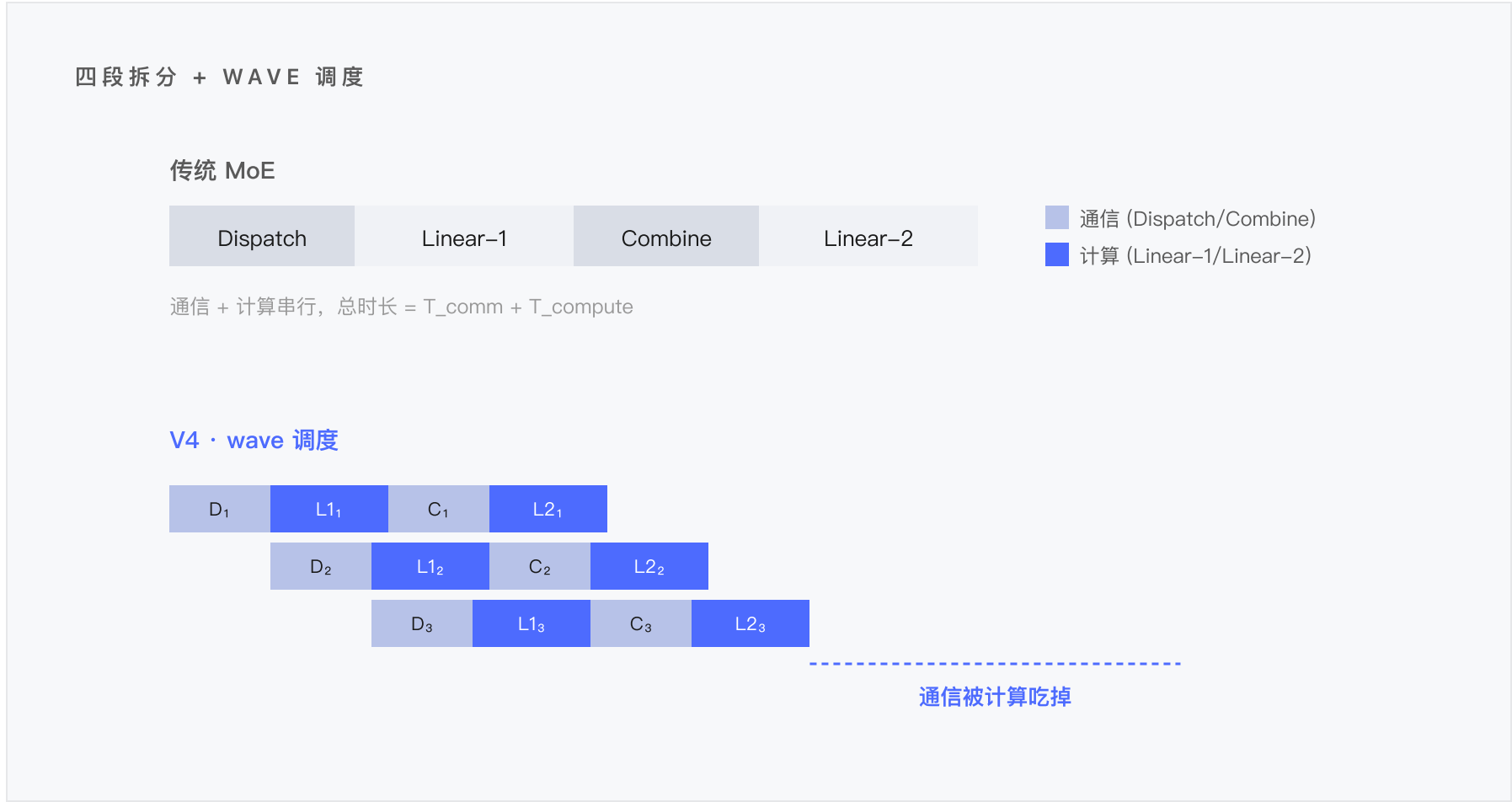
每一条缝都抠过：基建的六个模块

V4 的 Section 3 是整篇报告里 infra 味最浓的部分。DeepSeek 把训练和推理的每一个工程缝隙都抠了一遍——下面是这份「工程 checklist」的六个入口。

<div>COMMUNICATION01</div> <div>MoE 通信 计算重叠</div> <div>把 MoE 层拆成四段、用 wave 调度做细粒度流水。理论 1.92× 加速，RL rollout 实测最高 1.96×。</div>	<div>DSL02</div> <div>TileLang 领域专用语言</div> <div>用 TileLang 融合内核，Host Codegen 把 CPU 校验从几十微秒压到 1 微秒。SMT 求解器辅助整数分析。</div>	<div>DETERMINISM03</div> <div>批次无关 确定性 kernel</div> <div>训练与推理比特级一致。双 kernel 策略+独立累加 buffer，弃用 cuBLAS 换 DeepGEMM。</div>
<div>QUANTIZATION04</div> <div>FP4 量化 感知训练</div> <div>MoE 权重和 indexer QK 全上 FP4。FP4→FP8 反量化无损，top-k 提速 2×，KV recall 99.7%。</div>	<div>TRAIN FRAMEWORK05</div> <div>训练框架 三板斧</div> <div>Muon × ZeRO 用 knapsack 分桶，mHC wall-time 开销压到 6.7%，tensor 级 checkpointing 自动重计算。</div>	<div>INFERENCE FRAMEWORK06</div> <div>推理框架 KV cache 异构管理</div> <div>State Cache + KV Cache 两级池。把 shared-prefix KV 落盘，给 agent 长上下文拉低首 token 延迟。</div>

MODULE 01 · 通信计算重叠

让通信躲进计算的影子里



理论加速

1.92×

前作 Comet 只做到 1.42x，V4 把 wave 切得更细。

通用推理实测

1.50 – 1.73×

NVIDIA GPU 与华为 Ascend NPU 双平台验证。

RL ROLLOUT 峰值

1.96×

代码开源为 MegaMoE（DeepGEMM 组件）。

给硬件厂商的锚点：只要互连带宽满足 $C / B \leq 2d = 6144 \text{ FLOPs/Byte}$ ，通信就能被算力完全掩盖。超过这个阈值，继续堆带宽就是浪费硅面积——这是报告里一句写给 NVIDIA 的「需求文档」。

MODULE 02 · TILELANG DSL

开发效率 × 执行效率，它不做单选题

V4 的架构若用 Torch ATen 会产生上百个细粒度算子。TileLang 是一把折衷的手艺刀——CUDA 够底层但开发慢，Triton 够快但不够灵活，TileLang 把两头接上。

HOST CODEGEN

CPU 校验 沉到生成代码里

把 Python host 端的 shape 检查、layout 校验下沉到生成的 host 代码，基于 TVM-FFI 实现零拷贝调用。

几十 μs \rightarrow **<1 μs**

CPU 侧校验开销数量级下降

SMT SOLVER

形式化 整数分析

把 Z3 SMT solver 集成进 TileLang 代数系统，处理 layout inference、memory hazard、bound analysis 这些编译 pass。

Vectorize · Barrier · Simplify

编译时开销仅增加几秒

NUMERICAL PRECISION

数值精度 显式可控

默认关闭 fast-math。通过 T.__exp、T.__log 或 IEEE 严格模式（T.ieee_fsqrt 等）让开发者精确控制每一处数值行为。

Opt-in only

为确定性 kernel 铺路

Torch ATen 写出上百个算子，CUDA 手写工期爆炸——TileLang 在中间找到那个可落地的点。

MODULE 03 · 批次无关 & 确定性

训练与推理的比特级一致

同一个 token 无论在 batch 里的什么位置，输出比特级完全相同。Debug loss spike 可以直接对齐，线上异常能在训练环境精准复现——做后训练的团队，这是省下无数夜晚的基础能力。

DIFFICULTY 01

Attention

Split-KV 会造成 wave-quantization，同一 token 在不同位置上累加顺序会变，输出漂移。

双 kernel 策略： 第一个 kernel 单 SM 处理完整序列保吞吐，第二个 kernel 多 SM 处理最后那个半满 wave 降延迟，两者严格对齐累加顺序。

DIFFICULTY 02

Matmul

cuBLAS 做不到批次不变。放弃 split-k 又会有性能损失。

换引擎： 弃用 cuBLAS，全部切到 DeepGEMM。通过一组针对性优化把放弃 split-k 的性能损失补回来。

DIFFICULTY 03

反向传播

罪魁祸首是 `atomicAdd` ——浮点加法非结合律，不同执行顺序出不同结果。

独立 buffer + 全局有序 reduce： 每个 SM 分独立累加 buffer，最后做全局确定性求和；MoE 用 token order 预处理。

这三件事合起来的含金量：**RL rollout 的每一个 token，都能和训练环境比特级对齐**。对任何做大模型后训练和 RL 的团队来说，这是降维打击级的基础能力。

bit-exact
TRAIN ↔ INFERENCE

MODULE 04 · FP4 量化感知训练

把最重的两个位置*压到 FP4*

post-training 阶段做 QAT，MoE 权重和 CSA indexer 的 Query-Key 路径全上 MXFP4。关键是 FP4→FP8 反量化无损——不用重写训练框架就能复用 FP8 pipeline。

两个压缩目标

MoE Expert Weights

GPU 显存大头。专家权重量化后显著降低 memory traffic。

CSA Indexer QK

长文本注意力打分热点。top-k selector 是推理瓶颈。

反量化链路 · 为什么无损

FP4 (E2M1)

→

FP8 (E4M3)

→

FP32 master

FP8 比 FP4 多 2 个指数位，动态范围足够大，能完整吸收 FP4 sub-block（1×32 tiles）的 scale factor。反向传播用 STE 直接把梯度传回 FP32 master weights。RL rollout 直接用真 FP4 量化权重，采样与线上部署行为完全一致。

INDEXER TOP-K SELECTOR

2×

FP32 → BF16 之后的提速。

KV ENTRIES RECALL

99.7%

几乎无损——推理质量守住。

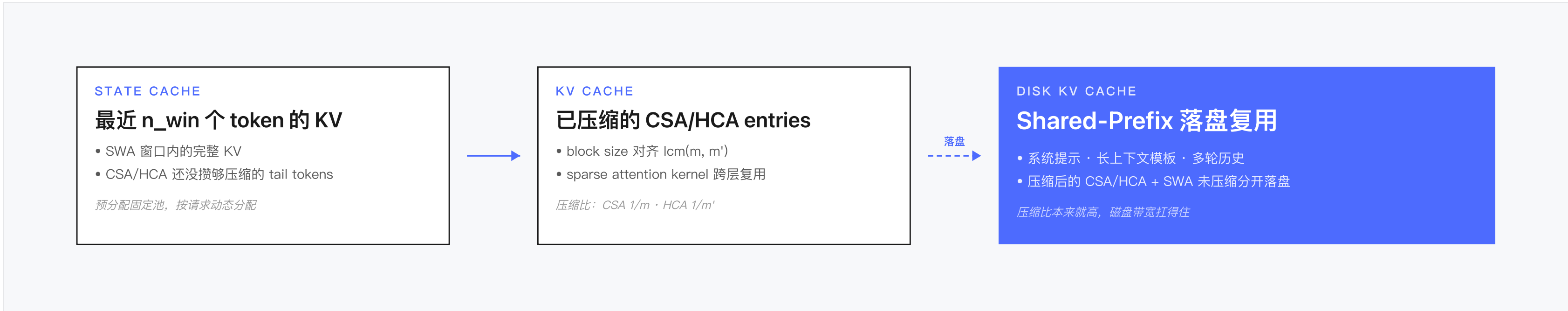
训练框架改动

复用 FP8 pipeline

不需要重写 backward。

大胆一招：把 *KV Cache* 落到磁盘

混合注意力带来 KV cache 的异构性——CSA、HCA、SWA、indexer 各有不同大小、更新规则、命中淘汰策略，PagedAttention 的假设整个失效。V4 拆成两级池，再把共享前缀落盘。



类比

像数据库给 page 做 buffer pool

热数据在内存、温数据在 SSD、冷数据在磁盘。KV cache 第一次享受到这种分层待遇。

谁受益最大

Agent 长上下文 · 多轮对话

shared-prefix 的重复度极高，省掉的 prefill 直接变首 token 延迟下降。

为什么以前没人这么做

没压缩之前磁盘扛不住

V4 的 CSA/HCA 把 KV 压到 $1/m$ ，落盘的带宽需求才降到磁盘 IO 能承受的量级。

这不是学术论文的「我们提出」，是工程师扫雷扫出来的工艺——先有压缩比，才敢谈落盘。

—— 小结 · 基建章的工程哲学

三条原则，贯穿整章

01

不吃硬件厂商的免费午餐

每一层都自己写，而且给硬件厂商反向提需求。

- MegaMoE / DeepGEMM / TileLang 三件自研
- 「6144 FLOPs/Byte」相当于写给 NVIDIA 的需求文档

02

可复现性是一等公民

训练环境与推理环境完全一致，debug 不再是玄学。

- 批次无关 + 确定性 kernel + TileLang 比特对齐
- 三件事合起来：loss spike 能精准复现

03

吝啬每一微秒、每一字节

不是论文式「我们提出 X」，是工程师每天扫雷扫出来的工艺。

- Host codegen 省几十微秒 · Muon BF16 压缩省一半通信
- 磁盘 KV 复用前缀 · mHC 只挑对的重计算

CHAPTER 06 · POST-TRAINING · 旧范式

SFT + RLHF 混炼，三个痛点

所有做过后训练的团队都踩过这些坑——它们合起来决定了为什么 V4 要把整个 RL 阶段拆掉重做。



痛点 01

RL 不稳定

学生模型直接跑 RL，训练曲线抖、崩 run 频繁。每次 debug 都得回退到上一个可用的 checkpoint，时间成本极高。

代价： 无数夜晚守着 loss 曲线。

痛点 02

多任务打架

数学、代码、agent、对话混在一个 reward 里。调高数学权重，代码掉分；加 agent 数据，对话变笨。

代价： 负迁移几乎不可避免，永远在跷跷板上走钢丝。

痛点 03

Reward Hacking

Reward model 是静态标量打分，学生很快学会「钻空子」——表面迎合 reward、实际质量下降。

代价： 需要不断对抗学生的捷径，reward model 本身要反复迭代。

V4 两阶段范式 · 总览

分治，再合并：*Specialist* → *OPD*

报告一句话点破：「mixed RL stage was entirely replaced by On-Policy Distillation」。RL 被隔离在专家阶段，学生模型只做稳定的蒸馏。



STAGE 01 · SPECIALIST TRAINING

每个领域独立训一个专家

先 SFT 打底，再 GRPO 做 RL。每个专家只管自己那块，reward signal 清晰不折中。V4 一共训了十多个专家。

单个专家的训练管线

01 SFT 打底

在该领域的高质量监督数据上做标准 fine-tuning，把基础能力带起来。

02 GRPO 做 RL

DeepSeek 自研的 Group Relative Policy Optimization。相比 PPO 不需要 value model，在一组采样里算相对优势——显存小、训练稳。V3、R1、V4 一脉相承。

03 输出：一个专家模型

整套管线重复十多次，训出数学 / 代码 / agent / 指令 / 推理等专家。每个都是独立的 checkpoint。

创新 · GENERATIVE REWARD MODEL

让 actor 自己当裁判

写作质量、helpful 程度这类「不好用规则验证」的任务，传统做法是标量打分 reward model。V4 反着来：用生成式模型当裁判，并对 GRM 本身做 RL——评估和生成联合优化，打分更鲁棒，对抗 reward hacking。

创新 · 三档推理强度

思考深度可调

Non-think 快速响应，不展开推理	Think High 默认推理强度	Think Max system prompt 强拉思考深度
--------------------------------	-----------------------------	--

STAGE 02 · ON-POLICY DISTILLATION

十多个专家当 teacher，蒸出一个学生

学生从自己采样轨迹，然后用反向 KL 对齐所有专家——多 teacher 蒸馏的新玩法。

目标函数

$$\mathcal{L}_{OPD}(\theta) = \sum_{i=1..N} w_i \cdot D_{KL}(\pi_{\theta} \parallel \pi_{E_i})$$

反向 KL

学生收敛到高概率区域

正向 KL 让学生 cover teacher 的所有模式，容易学成四不像。**反向 KL 让学生集中在 teacher 分布的高概率区域**——学的更聚焦。

ON-POLICY

学生自己采样轨迹

不是从 teacher 采样，是从 student 自己的策略采样。**学的永远是自己会遇到的分布**，不会出现「teacher 的高质量输出学生根本走不到」的错配。

FULL-VOCAB LOGIT

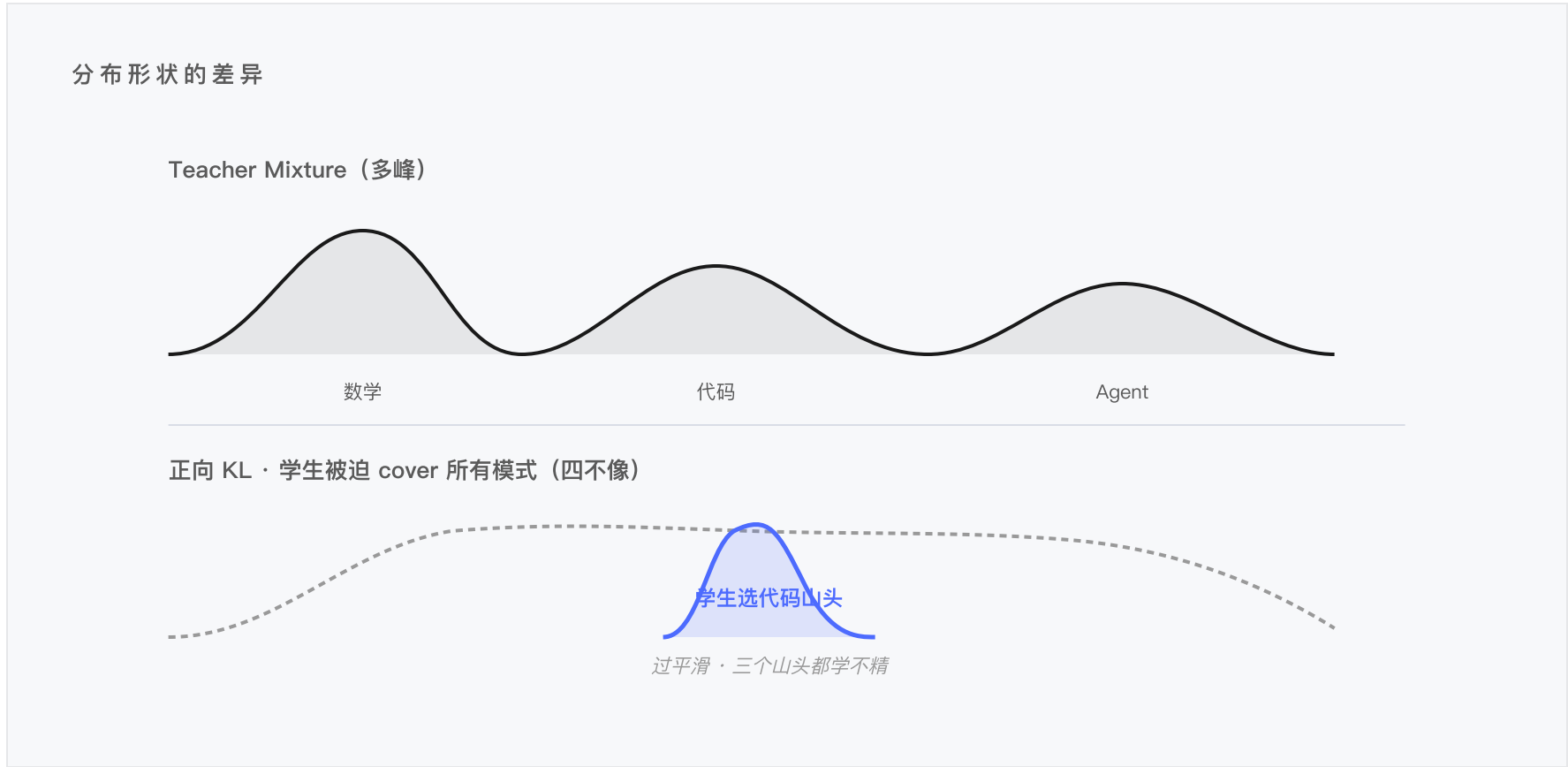
完整 logit 算精确 KL

以往工作为省算力用采样近似 KL。V4 坚持完整 logit 分布——**梯度方差更低、训练更稳**。代价是算力大幅上升，靠 infra 兜底。

—— 关键机制 · 为什么反向 KL 能 WORK

学生会自动选老师

数学任务对齐数学专家，代码任务对齐代码专家。这个「自动路由」特性，是多 teacher 蒸馏能 work 的底层秘诀。



SCENE 01

数学任务

学生的 rollout 落在「数学模式」附近，反向 KL 逼它对齐数学专家的高概率区域——自然忽略代码和 agent 的贡献。

SCENE 02

代码任务

同样的学生，遇到代码请求时 rollout 落在代码模式，自动切换对齐对象。没有人显式告诉它「这是代码」。

SCENE 03

任何新任务

加专家就能扩展能力。只要你能训出「写毛笔字专家」，加进蒸馏池——学生自然学会在相应场景下对齐它。

这是反向 KL 在多 teacher 场景里被低估的性质：**不需要一个外部 router 告诉学生「现在该听谁的」**，学生自己的 on-policy 分布就是 router。

Agent RL 的两个地基： *裁判能思考 · 执行有产房*

GENERATIVE REWARD MODEL

让 actor 自己 当裁判

对「不好用规则验证」的任务（写作质量、helpful 程度），传统做法是标量打分 reward model。V4 反着来：用生成式模型当裁判，对 GRM 本身也做 RL 优化——评估能力和生成能力联合优化。

判官和 policy 统一——和 Anthropic Constitutional AI 的方向一致，避免 reward hacking。

DSec · DeepSeek Elastic Compute

LIBDSEC · PYTHON SDK

Function Call

预热容器池

无状态调用，消除冷启动。最快一档。

Container

Docker + EROFS

标准任务，镜像按需加载。

microVM

Firecracker

高密度安全隔离，overlaybd 快照。

fullVM

QEMU

任意操作系统，给最复杂的 agent。

几十万

单集群并发 sandbox 实例

毫秒级

overlaybd 快照恢复

开源世界第一个工业级 **agent infra**。没有生产级 sandbox，agent RL 基本没法 scale——OpenAI 的 codex、Anthropic 的 Claude Code 都建在类似基础设施上。DeepSeek 把这套东西写进技术报告，给复现留了条路。

—— 小结 · 为什么说这是范式革命

RL 的角色被重新定位了

以前 RL 是「让最终模型对齐人类偏好」的最后一公里，现在 RL 是「训练专家」的中间步骤。

维度	传统 SFT + RLHF	DEEPSEEK V4 两阶段
多任务冲突	联合优化易打架，reward 权重无止境拉锯	物理隔离，专家之间互不干扰
RL 稳定性	学生直接 RL，曲线抖、崩 run 频繁	RL 限定在专家阶段，学生只做蒸馏
扩展新能力	要重调 reward balance，可能影响已有能力	加一个专家就行，路径清晰
基础设施门槛	一般	极高：full-vocab 蒸馏 + 多 teacher 调度

作者观点

把 RL 的不稳定性用工程手段绕过：与其让万能模型在复杂 reward landscape 上挣扎，不如让若干专精模型各自在简单 landscape 上走稳。这比任何架构创新都更深刻。

CHAPTER 07 · 团队基因

DeepSeek 招的是竞赛获奖选手——所以模型也像竞赛选手

招谁像谁。一家公司的模型，长得像它的员工。

CASE 01

罗福莉

MoE 核心研究员

北京大学 计算机硕士

本科 北京师范大学

ACL 发表 8 篇一作

「95 后 AI 天才少女」

CASE 02

高华佐

MLA 核心作者

物理奥赛 金牌

北京大学 物理系

DeepSeek-V2 MLA 提出者

Mesh-TensorFlow 贡献者

CASE 03

邵智宏

GRPO 算法提出者

清华大学 交互式信息 PhD

DeepSeekMath 一作

GRPO 方法被 OpenAI 借用

清华奖学金获得者

公开可查的 DeepSeek 研究员里，竞赛金牌、清北、中科院出身是默认配置。招这样的人，模型就会在「标准化、有明确答案」的任务上表现最好——就像理科状元，命题作文答得漂亮。

Codeforces Rating 3206——开源首次反超闭源

3206

CODEFORCES RATING

相当于全球真人选手
第23名

SAME BENCHMARK · OTHER MODELS

DeepSeek-V4-Pro-Max	3206
GPT-5.4-xHigh	3168
Gemini 3.1 Pro	3052
Claude Opus 4.6	未公开

论文原文点名：Pro-Max 在 Codeforces 上排所有参赛真人选手第 23 名。这是第一次开源模型在单一竞赛项目上真正反超闭源旗舰——GPT-5.4-xHigh 被甩开 38 分。

CHAPTER 07 · 形式化数学

Putnam-2025 形式化证明 满分 *120 / 120*

120 / 120

HYBRID FORMAL-INFORMAL

第一个在 *Putnam*
拿到满分的 AI 系统

超过的已知系统

Axiom · 被超越

Seed-Prover · 被超越

Lean v4.28.0-rc1 工具链

最多 500 次工具调用 · 满分达成

Putnam 是北美大学生数学竞赛，以题目刁钻著称。hybrid formal-informal 设置要求模型先用自然语言推理、再把证明翻译成 Lean 形式化代码——DeepSeek-V4-Pro-Max 拿下 120 的满分，全场唯一。

CHAPTER 07 · 数学全景

数学赛道：Apex Shortlist 全场第一

BENCHMARK	OPUS 4.6	GPT-5.4	GEMINI 3.1	K2.6	GLM-5.1	DS-V4-PRO
HMMT 2026 Feb 高中数学竞赛	96.2 ★	97.7	94.7	92.7	89.4	95.2
IMOAnswerBench 奥数答案	75.3 ★	91.4	81.0	86.0	83.8	89.8
Apex 研究级数学	34.5	54.1 ★	60.9	24.0	11.5	38.3
Apex Shortlist 简答版	85.9	78.1	89.1	75.5	72.4	★ 90.2
Putnam-2025 形式化证明	—	—	—	—	—	★ 120/120

DeepSeek-V4 在**五项数学评测里拿下两个第一**（Apex Shortlist 90.2、Putnam 满分），剩下三项全部追到 Top-2。**IMOAnswerBench 比 Opus 4.6 高 14.5 分，接近 GPT-5.4 的巅峰。**

CHAPTER 07 · 编程全景

编程赛道： 竞赛编程封神，工程代码一线

BENCHMARK		OPUS 4.6	GPT-5.4	GEMINI 3.1	K2.6	GLM-5.1	DS-V4-PRO
LiveCodeBench 实时竞赛编程		88.8	—	91.7	89.6	—	★ 93.5
Codeforces Rating 算法竞赛		—	3168	3052	—	—	★ 3206
SWE-Bench Verified 工程修 bug	★	80.8	—	80.6	80.2	—	80.6
SWE-Bench Pro 工程 · 更难		57.3	57.7	54.2 ★	58.6	58.4	55.4
SWE-Bench Multilingual 多语言工程	★	77.5	—	—	76.7	73.3	76.2

竞赛编程维度全场第一（LiveCodeBench + Codeforces 两冠），但 SWE 系列工程代码只是接近而非超越——跟 Opus 4.6 打成 80.6 vs 80.8 的贴身差距。论文原话：「第一次开源模型在竞赛编程上追平闭源」。

CHAPTER 07 · 最新对比

标准化任务 vs 2026-04 最新闭源旗舰

论文对标止步于 Opus 4.6 / GPT-5.4。但这十天里，Opus 4.7（04-16）和 GPT-5.5（04-23）都发布了。

BENCHMARK	OPUS 4.7	GPT-5.5	GEMINI 3.1	K2.6	GLM-5	DS-V4-PRO
Codeforces Rating算法竞赛	—	—	3052	—	—	3206
LiveCodeBench实时竞赛编程	—	—	91.7	89.6	—	93.5
Putnam-2025 形式化数学证明 · 满分 120	—	—	—	—	—	120/120
Apex Shortlist研究级数学	—	—	89.1	75.5	72.4	90.2
SWE-Bench Verified工程修 bug	87.6	—	80.6	80.2	—	80.6
SWE-Bench Pro工程 · 更难	64.3	58.6	54.2	58.6	58.4	55.4
FrontierMath T1-3前沿数学	43.8	51.7	36.9	—	—	—

竞赛编程和形式化数学上 DeepSeek-V4 是全场第一，开源首次断档领先。但是换到工程代码 SWE-Bench Verified，Opus 4.7 甩开 7 个百分点——这是「标准化 vs 工程复杂度」的分水岭。

CHAPTER 07 · 小结

THE VERDICT

标准化、有明确答案的题，
DeepSeek 是开源天花板。

CODERFORCES

3206

全场第一
超过 GPT-5.4

LIVECODEBENCH

93.5

全场第一
超过 Gemini 3.1

APEX SHORTLIST

90.2

全场第一
研究级数学

PUTNAM-2025

120/120

首个满分系统
形式化证明

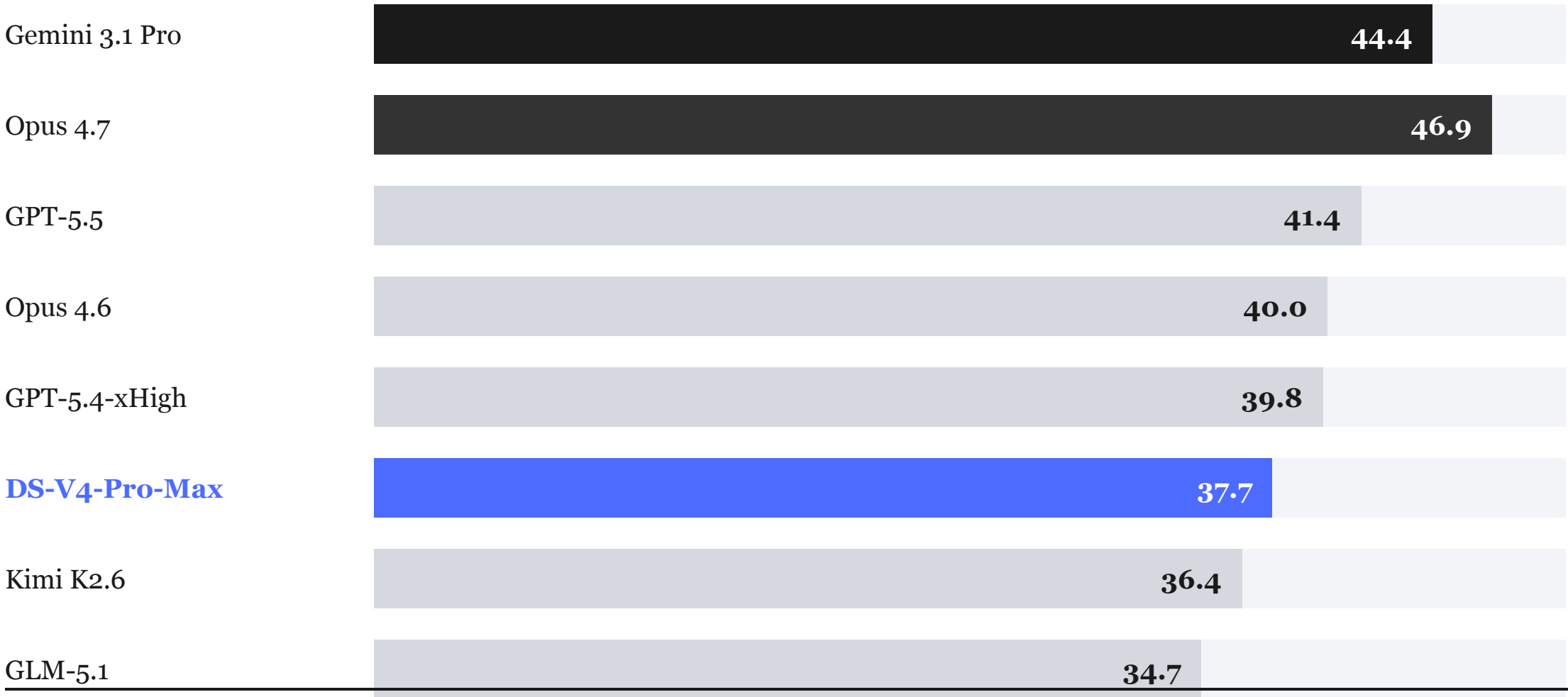
竞赛选手基因里的习惯——给一道有标准答案的题，用尽全部推理能力把它做对——在 DeepSeek-V4 身上成了护城河。

CHAPTER 08 · 不擅长什么

但是。

一旦离开「标准答案」这个安全区，模型的性格就会露出来。

HLE——人类最终考试，差距第一次露出来



DS-V4 VS GEMINI

—6.7pp

落后 Gemini 3.1 Pro
6.7 个百分点

落后 Opus 4.7
9.2 个百分点

HLE（Humanity's Last Exam）是跨学科综合硬推理题。不是做题，是「你有多通」。DeepSeek-V4 的 37.7 分在开源阵营仍是第一，但在闭源对比下开始吃瘪——这是「擅长什么」和「不擅长什么」的分水岭。

给 HLE 配上工具——*DeepSeek* 反而倒数第一

HLE · NO TOOLS	
只靠内置推理	
Gemini 3.1 Pro	44.4
Opus 4.6	40.0
GPT-5.4	39.8
DS-V4-Pro-Max	37.7
Kimi K2.6	36.4

HLE · WITH TOOLS	
能调搜索、代码、计算器	
Kimi K2.6	54.0
Opus 4.6	53.1
GPT-5.4	52.0
Gemini 3.1 Pro	51.6
GLM-5.1	50.4
DS-V4-Pro-Max	48.2

倒 1

从原本倒 4
变成全场倒 1

所有其他模型加上工具都会涨分 11-18 个百分点。DeepSeek-V4 只涨了 10.5 分，名次却从倒 4 掉到倒 1——连 Kimi K2.6 都超不过。这是 Agent 能力的结构性短板：工具使用的协调不行。

CHAPTER 08 · TERMINAL BENCH 2.0

真实终端任务：差 GPT-5.5 约 15 个百分点



-14.8_{pp}

论文只对标到 GPT-5.4（差 7pp）。但 GPT-5.5 于 2026-04-23 发布，Terminal-Bench 2.0 直接做到 82.7 分——DeepSeek-V4 与新 SOTA 差距实际上扩大到 14.8 个百分点。

CHAPTER 08 · 创意写作

打 Gemini 能赢，打 Opus 4.5 就输了

中文写作胜率 (DeepSeek-V4-Pro vs 基线模型) —— 论文 Table 18 原数据

功能性写作 · vs Gemini 3.1 Pro

日常白领任务：邮件、报告、总结



创意写作 · 写作质量 · vs Gemini 3.1 Pro

Gemini 「经常让风格偏好压过用户明确需求」



高难度约束 / 多轮场景 · vs Claude Opus 4.5

复杂指令跟随、多约束创作



「对于复杂约束/多轮场景下的创意写作任务，DeepSeek-V4 相对 Claude Opus 4.5 的胜率为 45.9%，略低于 52.0% 的基线。」
— DEEPSEEK-V4 TECHNICAL REPORT · TABLE 18

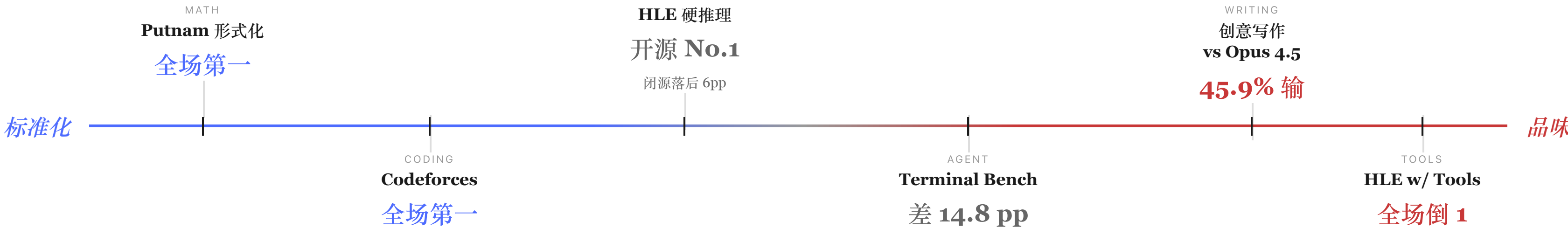
CHAPTER 08 · 分水岭

差距的秘密：有没有 *Ground Truth*

任务光谱——从「有标准答案」滑向「需要品味」，DeepSeek-V4 的排名也随之滑落。

有 GROUND TRUTH · 擅长

需要品味 · 吃瘪



左端：每道题有一个唯一正确答案，评分函数明确。强化学习能稳定拉动能力上升，DeepSeek 是开源天花板。**右端：**「好」与「不好」是连续的、主观的，没有可优化的 reward 信号。参赛选手基因不一定是优势——甚至可能是劣势。

CHAPTER 08 · 诚实的承认

论文里 *自己承认* 的三个局限

Section 6 · Limitations——DeepSeek 把硬话自己说了，这在国产大模型论文里少见。

01

ARCHITECTURAL BLOAT

架构臃肿——叠了太多组件，不知道哪个真正重要

「现有架构叠加了多项组件（mHC / HCA / CSA / DTA / Muon / LPO），我们并未系统消融各组件的独立贡献。未来需要更严格的分解实验。」

02

EMPIRICAL, NOT UNDERSTOOD

关键技术起效了，但不理解为什么

「Anticipatory Routing 和 SwiGLU Clamping 在实验中被证实有效，但我们缺乏对其底层机制的理论理解——它们为何起作用、何时失效，仍是开放问题。」

03

SPARSITY NOT EXTREME ENOUGH

Sparse 还不够极致——激活比还能往下压

「我们相信进一步稀疏化仍有空间——当前 49B/1.6T 的 3% 激活比离理论上界仍有距离。极致稀疏化会带来更严重的路由不平衡和训练不稳定，但也是开源打赢闭源的下一个关键。」

这不是模糊的谦虚——是三个具体、可验证、可被追问的技术承认。[最诚实本分的模型](#)不是口号。

CHAPTER 08 · 小结

THE VERDICT

需要品味、模糊判断、
长链工具调用的任务，
DeepSeek 的差距明显。

CREATIVE WRITING

品味类

45.9%

vs Opus 4.5 的胜率
论文自认的硬输

TERMINAL BENCH 2.0

Agent 类

-14.8pp

vs GPT-5.5
长链工具调用差

HLE WITH TOOLS

工具协调

倒数 1

连 Kimi K2.6 都不如
全场最弱项

竞赛选手擅长把题做对，不擅长决定「什么题值得做」。DeepSeek-V4 的护城河和短板，都来自同一个基因。

CHAPTER 09 · 观点一 · 普惠的真相

1M 不是「能用」，*是*不再省字

之前你是在字字计较地挑哪段塞给模型。现在你想塞多少塞多少，规划本身变成了多余动作。

BEFORE · 128K 时代

字字计较

打开一个代码仓库，先问自己「哪几个文件最关键、怎么切片、怎么做 RAG」。

你在为模型做**取舍规划**，不是在解决问题。每一次上下文都像是打车前算路费。



AFTER · 1M 默认

想塞多少塞多少

整个仓库直接丢进去，一周的对话历史不用清，30 份 PDF 同时读。

你不再**为上下文省字**。规划节流这个动作本身被消除了——就像从打车变成了包月公交。

1M context 真正的价值，不是「多了 8 倍容量」这种数字。是**让一个之前需要专家判断的前置动作，变成了可以默认跳过的步骤**。这才是「普惠」。

CHAPTER 09 · 观点一 · 普惠的真相

1M 是容量上限，不是日常用量

论文 Table 10：V4-Agentic Search 的实际 prefill 只用了 13,649 tokens。1M 的真正用途是让你「不再需要规划」。

容量上限

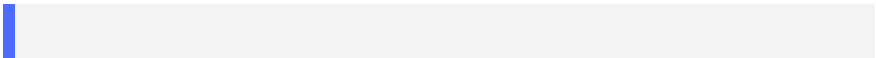
1M CONTEXT



1,000,000

官方 Agentic Search 实测

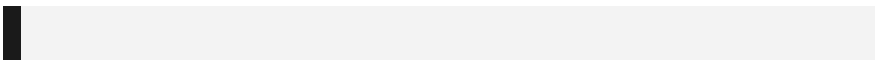
PAPER TABLE 10



13,649

日常 Coding Agent

ESTIMATED



10 – 20K

WHY IT MATTERS

容量买的不是字数，
是自由度。

你大部分时候仍然只用几万 token。但不用再提前做切片、RAG、排优先级——该塞就塞，该忘就忘。

类比：你每天只喝一杯水，但家里还是要通自来水。因为需要时不能临时想办法。

1M 的商业价值不在「装得下」，而在规划开销的边际消失。这是 agent 从 demo 走到生产的关键阈值。

CHAPTER 09 · 观点一 · 普惠的真相

官方价格 · 两款模型一张表

单位：人民币元 / 百万 tokens。两款模型均支持 1M 上下文。

模型	输入 · 缓存命中	输入 · 缓存未命中	输出	上下文
<div><div>deepseek-v4-pro</div><div>FLAGSHIP · 对标闭源顶级</div></div> <div>1元 / M</div>	12元 / M	24元 / M	1M	
<div><div>deepseek-v4-flash</div><div>COST-EFFICIENT · 价格屠夫</div></div> <div>0.2元 / M</div>	1元 / M	2元 / M	1M	

Pro 的定位是「对标顶级闭源的开源最佳」——便宜是结果，不是主打。**真正的价格屠夫是 Flash**——缓存命中输入 0.2 元，每百万 tokens 一杯蜜雪冰城都不到。

CHAPTER 09 · 观点一 · 普惠的真相

V4-Pro vs 最新闭源旗舰：输出便宜22 倍

对标 Opus 4.7、GPT-5.5、Gemini 3.1 Pro。单位：美元 / 百万 tokens（输入未缓存 · 输出）。

模型	输入 · 未缓存	输出	VS V4-PRO
<div>DeepSeek V4-Pro</div> <div>OPEN FLAGSHIP</div>	\$1.67 / M	\$3.33 / M	baseline
<div>Claude Opus 4.7</div> <div>ANTHROPIC · 2026</div>	\$15 / M	\$75 / M	9× / 22.5×
<div>GPT-5.5</div> <div>OPENAI · 2026</div>	\$5 – 10 / M	\$30 – 40 / M	3 – 6× / 9 – 12×
<div>Gemini 3.1 Pro</div> <div>GOOGLE · 2026</div>	~\$5 / M	~\$30 / M	3× / 9×

CAVEAT · 诚实说

V4-Pro 不是无脑便宜。是「对标顶级闭源的开源性价比」。
\$1.67 的输入价格在全球模型里只是中档——便宜是相对旗舰，绝对价格已经是主流水平。

CHAPTER 09 · 观点一 · 普惠的真相

V4-Flash：*同性能里最便宜*

普惠叙事的主力不是 Pro，是 Flash。它把 1M 上下文 + 接近顶级推理能力压到了全球最低价位。

V4-FLASH · 输入缓存命中

0.2

元 / 百万 tokens · 约 \$0.028 / M

缓存未命中 **1 元**，输出 **2 元**，上下文 **1M**。
一杯蜜雪冰城的钱，够跑完一本 50 万字的小说。

同档对比 · 输入 / 输出（\$ / M）	
<div>V4-Flash</div> <div>DEEPSEEK · 2026</div>	<div>\$0.14 / \$0.28</div>
<div>Haiku 4.7</div> <div>ANTHROPIC</div>	<div>~\$1 / ~\$5</div>
<div>GPT-5.5-mini</div> <div>OPENAI</div>	<div>~\$0.5 / ~\$2</div>
<div>Gemini Flash</div> <div>GOOGLE</div>	<div>~\$0.3 / ~\$2.5</div>
相比 Haiku 4.7：输入便宜 7× · 输出便宜 18×	

如果说 Pro 是「让闭源发抖」的一张牌，Flash 才是「让每个独立开发者都能塞百万上下文」的那张牌。官方的「迈入百万上下文普惠时代」，兑现它的是 Flash。

CHAPTER 09 · 观点一 · 普惠的真相

不抢终端，进开发者的 agent 工具链

官方发布文明确提到已为这四个 agent 产品做了适配和优化——DeepSeek 的战场不在 chat.xxx.com，在开发者每天打开的终端里。

ADAPTED · 01

Claude Code

Anthropic 官方 CLI。V4 完全兼容 **Anthropic API 接口**，改个 model_name 即可接入。

ADAPTED · 02

OpenClaw

花叔每天用的开源 Claude Code 替代品。V4 做了针对性调优，**可以直接当后端**。

ADAPTED · 03

OpenCode

开源 coding agent 生态。V4 进入默认 model list，**开源 + 开源**的组合。

ADAPTED · 04

CodeBuddy

国内开发者常用的 IDE 插件。V4-Flash 的价格让 **coding agent 后台任务**真正可持续跑。

STRATEGIC TAKE

闭源大厂在抢谁用它们的 chat。DeepSeek 在抢谁被开发者嵌进脚本里。
一个是 toC 入口战，一个是 toDev 基础设施战——后者更慢，但更不可替代。

01

普惠

从「字字计较」 到「包月随用」的 agent 时代。

这不是一个冲破天花板的世界最佳发布。这是一个让每个独立开发者都能默认用上 1M agent 的普惠发布——Flash 把价格砸到地板，Pro 把 agent 工具链生态吃下来。

CHAPTER 10 · 观点四 · 诚实本分

论文里，三个「我们还不理解」

Section 6 的 Limitations——DeepSeek 主动写出来的三个短板。这种坦白在 LLM 论文里不常见。

01

ARCHITECTURE · 架构臃肿

验证过的组件
堆得太多了

「We retained many preliminarily validated components and tricks, which, while effective, made the architecture relatively complex.」

为了降低训练风险，把一堆验证过的小技巧全堆上去了。下一版要做减法，蒸馏出最本质的设计。

02

STABILITY · 机制不理解

管用但
不知道为什么管用

「Anticipatory Routing and SwiGLU Clamping ... their underlying principles remain insufficiently understood.」

专家预路由和激活裁剪确实有效，但原理还没搞清楚。白纸黑字写进论文，这种诚实 LLM 圈子里少见。

03

SPARSITY · 还不够极致

稀疏性的
天花板还没碰到

「Beyond MoE and sparse attention ... explore new dimensions of sparsity.」

MoE 和 sparse attention 只是起点，要沿更多维度挖。V5 大概率会走向 scalable lookup 式记忆模块。

别人写论文是证明自己无懈可击，DeepSeek 写论文是告诉你哪里还没做好。这是工程师的语言，不是市场部的语言。

V5 路线图：直接告诉你下一步做什么

论文 Section 6 最后一段是未来五个方向。别人藏着掖着，DeepSeek 直接铺出来。

01

更极致的 sparse

不止 MoE 和 attention 稀疏，沿**新维度挖稀疏性**。

02

低延迟架构

让长上下文部署**更响应**，为 agent 交互做准备。

03

Long-horizon agent

多轮、跨小时的**长周期任务**作为下一步主攻。

04

多模态

V5 大概率是**原生多模态**，不再走 adapter 路线。

05

数据合成与 curation

持续优化数据策略，尤其是**合成数据**的质量。

V5 预告 · 最关键的一条

参考文献指向 Cheng 等 2026 年的「**Conditional Memory via Scalable Lookup**」——
V5 大概率会引入**可扩展查找式的记忆模块**，超越 attention 本身的范式。

Pro 现在贵，官方亲口承认

“

DEEPSEEK 官方发布文 · 原话

受限于高端算力，目前 Pro 的服务吞吐十分有限，
预计下半年昇腾 950 超节点批量上市后，
Pro 的价格会大幅下调。

—— 2026-04-24 官方公众号

这不是营销辞令
这是诚实

承认当前贵 · 点名国产算力 · 给出时间节点 · 不画大饼——一句话包含了四层诚实。这种姿态在 2026 年的 AI 发布会里，已经是稀缺品。

四个观点 · 回到最后一个

01

THESIS · 普惠

不是冲破天花板 是让普通人用上 1M agent

Flash 的 0.2 元缓存命中价 + Pro 进入开发者工具链生态。普惠的兑现者是 Flash，不是 Pro。

02

THESIS · 基因

竞赛选手团队的 解题型模型

DeepSeek 招的大多是竞赛获奖选手，模型也体现出明显的「做题」特点——数学、编程、有标准答案的任务都打得很好。

03

THESIS · 短板

品味类任务 还差一些

创意写作不如 Opus 4.5，更不用说 4.7。论文里 DeepSeek 自己也写到了这些困难——这是结构性差距。

04

THESIS · 姿态

最诚实、最本分的 开源模型

承认局限、给出路线图、点名算力约束、不画大饼。这才是 DeepSeek 真正的护城河——在 AI 营销噪音里，诚实本身就是稀缺品。

CHAPTER 10 · 情感收尾

不诱于誉，不恐于诽，
率道而行，端然正己。

——
出自《荀子·非十二子》

被 DeepSeek 写在 V4 官方发布文的结尾

CHAPTER 11 · 收尾 · 一句话定位

DeepSeek-V4 是 最诚实、最本分的 开源旗舰。

不是世界最佳，不必是。它做的事更重要——把百万上下文、agent 原生支持、开发者工具链生态，打包成独立开发者也能用得起的底座。地板抬高的速度，决定了 AI 应用爆发的规模。

1M · 上下文普惠 \$0.14 · Flash 输入未缓存 昇腾 950 · 下半年降价 不诱于誉 · 官方姿态

感谢你读到这里 —— 72 / 72

本次深度解读主要基于 DeepSeek-V4 论文、官方发布文与花生在开源 agent 生态里的实战经验。

一手资料 · PRIMARY

DeepSeek-V4 Technical Report

55 页论文，Section 1–6 + Appendix · 2026-04

官方发布文

DeepSeek 官方公众号 · 2026-04-24

官方价格表

含昇腾 950 降价预告原文

花生旧文 · BACKLOG

DeepSeek-V3.2 实测

混合 attention 初探

Claude Code 橙皮书

agent 工具链生态实战

OpenClaw 系列

开源 coding agent 实战笔记

工具与制作

huashu-design skill

HTML 高保真 PPT 设计 workflow

Claude Opus 4.7

内容解读与撰稿

DeepSeek V4-Pro

论文交叉验证

花叔 · AI Native Coder

公众号 · B站 · X · YouTube · 小红书
全平台搜「花叔」