扩散大语言模型方法及应用

上海交通大学 邓志杰

CNCC 2026

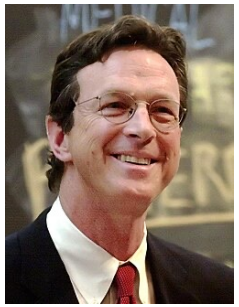


上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

高质量文本如何产生?

Books aren't written - they're rewritten.

— Michael Crichton



美国作家、编剧

《侏罗纪公园》作者



钱慧安作品

从整体构思，到局部推敲

“僧推月下门”，还是“僧敲月下门”？

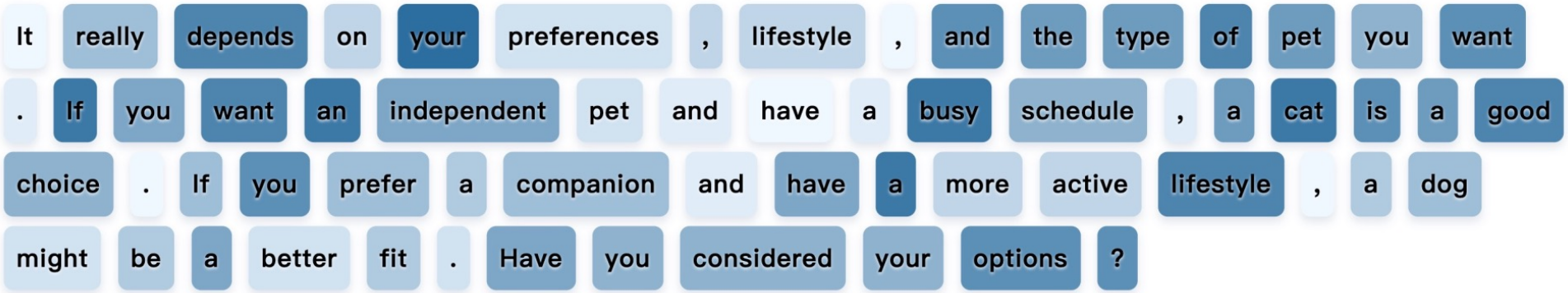
结合上下文，推敲一字之得失

框架明确后，不同部分也可以分别展开

语言生成，是否也能在多个位置同时展开？

扩散语言模型：在多个位置展开生成

同时预测多个缺失位置，逐步补全文本

User	I want to have a pet, but I don't know whether to get a cat or a dog. Do you have any suggestions?
LLaDA	

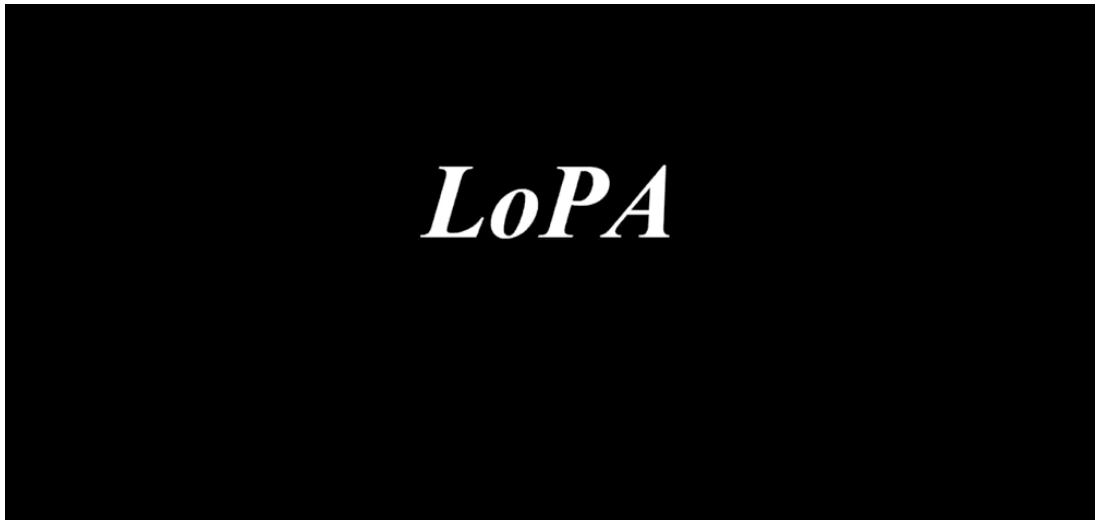
浅色：早期预测；深色：后期预测

生成不必严格从左到右，文本可以在多处逐步展开

arXiv:2502.09992

能否在保证生成质量的同时，
让扩散大语言模型生成得更快？





D2F：超越同尺寸自回归模型速度

LoPA：7B模型推理速度超1k tokes/s

报告提纲

- 基础原理与模型进展
- 模型训练与生成质量
- 推理算法与系统优化
- 应用方向与未来展望

报告提纲

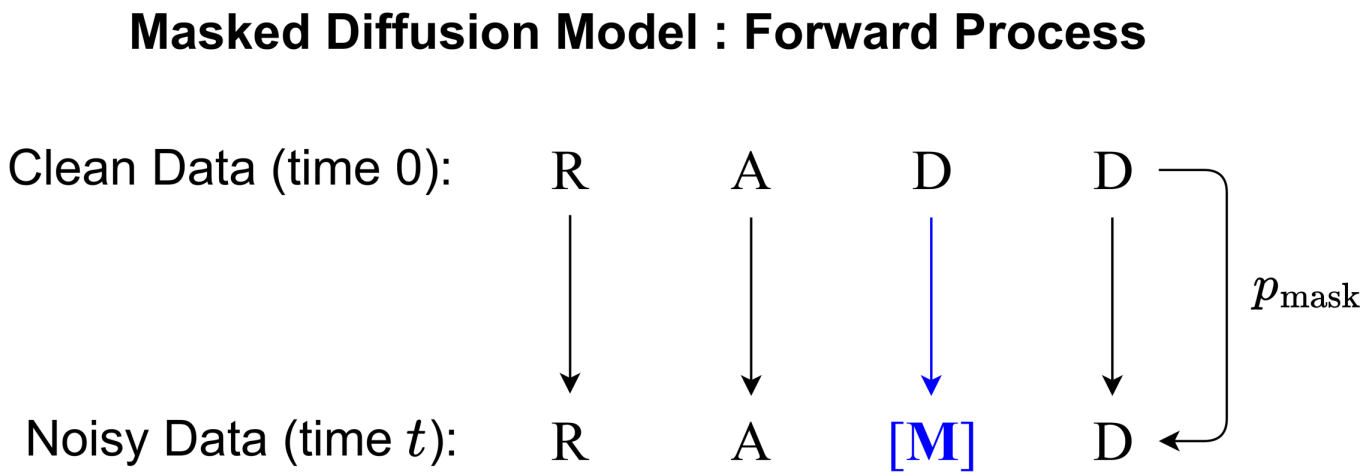
- 基础原理与模型进展
- 模型训练与生成质量
- 推理算法与系统优化
- 应用方向与未来展望

自回归生成：下一步依赖已有前缀

$$p_\theta(x) = p_\theta(x^1) \prod_{i=2}^{L} p_\theta(x^i \mid x^1, \ldots, x^{i-1})$$

标准解码按顺序推进，后续 **token** 需要等待。

掩码扩散：前向过程



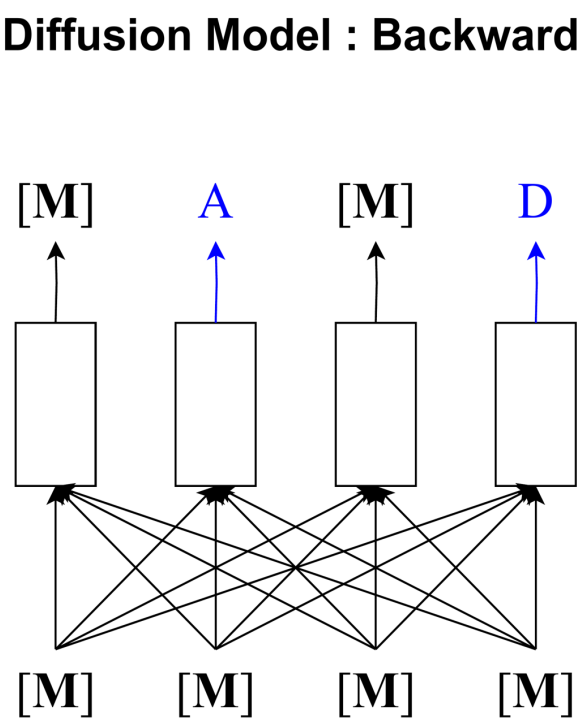
$t \in [0,1]$，各维独立地以概率 t 被掩码，否则不变

arXiv:2502.09992 | arXiv:2406.03736

掩码扩散：反向过程

从全掩码到无掩码的反向扩散过程

以时间 \boldsymbol{t} 为条件的共享网络建模转移核



arXiv:2502.09992 | arXiv:2406.03736

训练目标：随机比例掩码 + 交叉熵

- 对输入序列进行随机比例的掩码，预测掩码位置，并计算标准交叉熵损失

$$\mathcal{L}(\theta) \triangleq -\mathbb{E}_{t,x_0,x_t}\left[\frac{1}{t}\sum_{i=1}^{L}\mathbf{1}[x_t^i = \mathrm{M}]\log p_\theta(x_0^i|x_t)\right]$$

最大似然估计定理：最小化随机比例掩码交叉熵损失等价于最大化对数似然的变分下界，即：

$$-\mathbb{E}_{p_{\mathbf{data}}(\mathbf{x_0})}[\mathbf{log}\mathrm{p}_\theta(\mathrm{x}_\mathbf{0})] \leq \boldsymbol{\mathcal{L}}(\theta).$$

掩码扩散模型是一种处理离散化数据的原则性（最大似然意义下）生成框架

arXiv:2502.09992 | arXiv:2406.03736

解码：并行预测，不等于全部提交

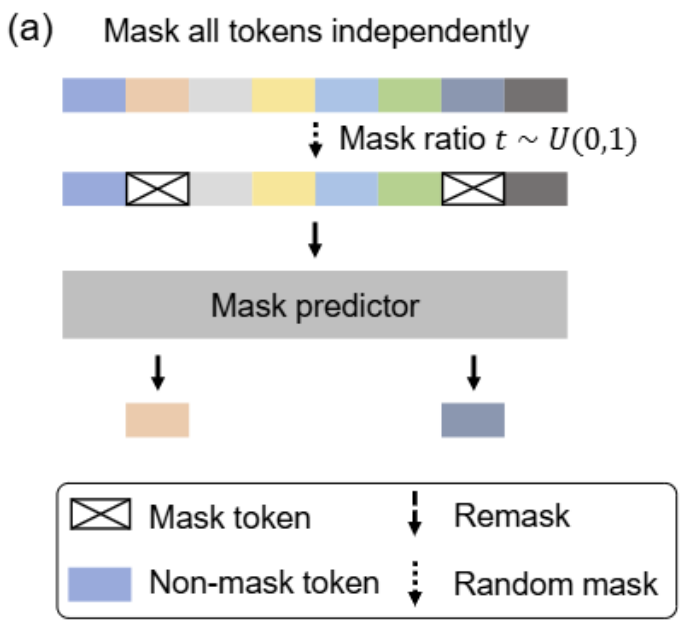
每轮预测全部 **mask**，只提交其中一部分（示意）

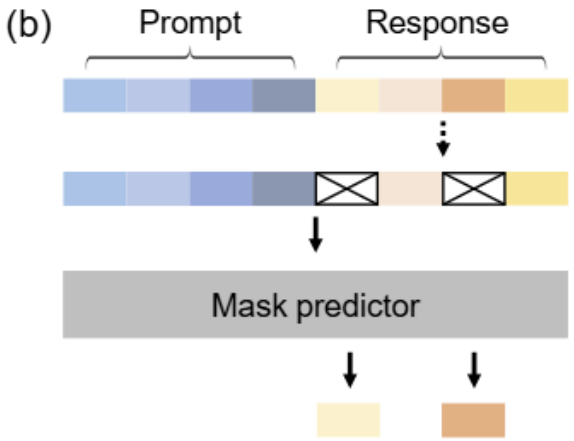
第 1 轮	**We**	[M]	[M]	[M]	**in**	[M]
第 2 轮	We	**can**	[M]	**tokens**	in	[M]
第 3 轮	We	can	**generate**	tokens	in	**parallel**

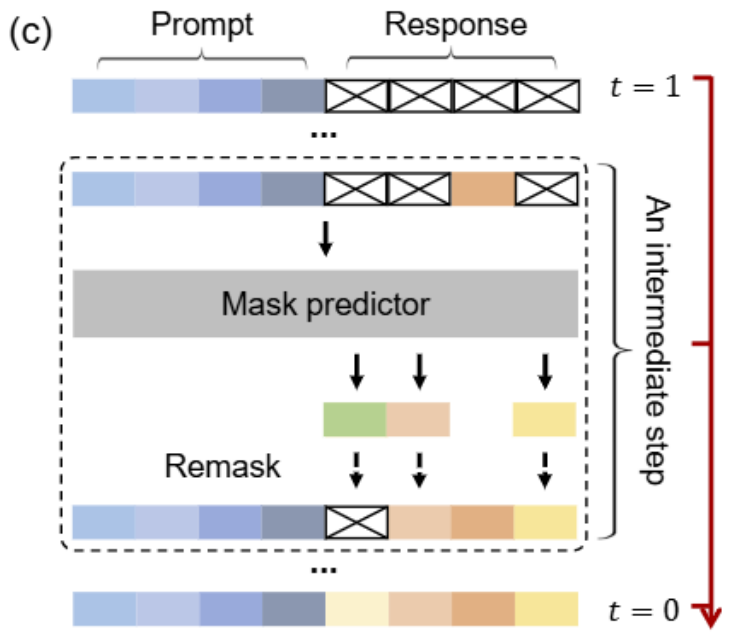
红色：本轮新提交；灰色：继续等待后续预测。

arXiv:2502.09992 | arXiv:2505.22618

LLaDA：预训练、微调与生成的完整路线







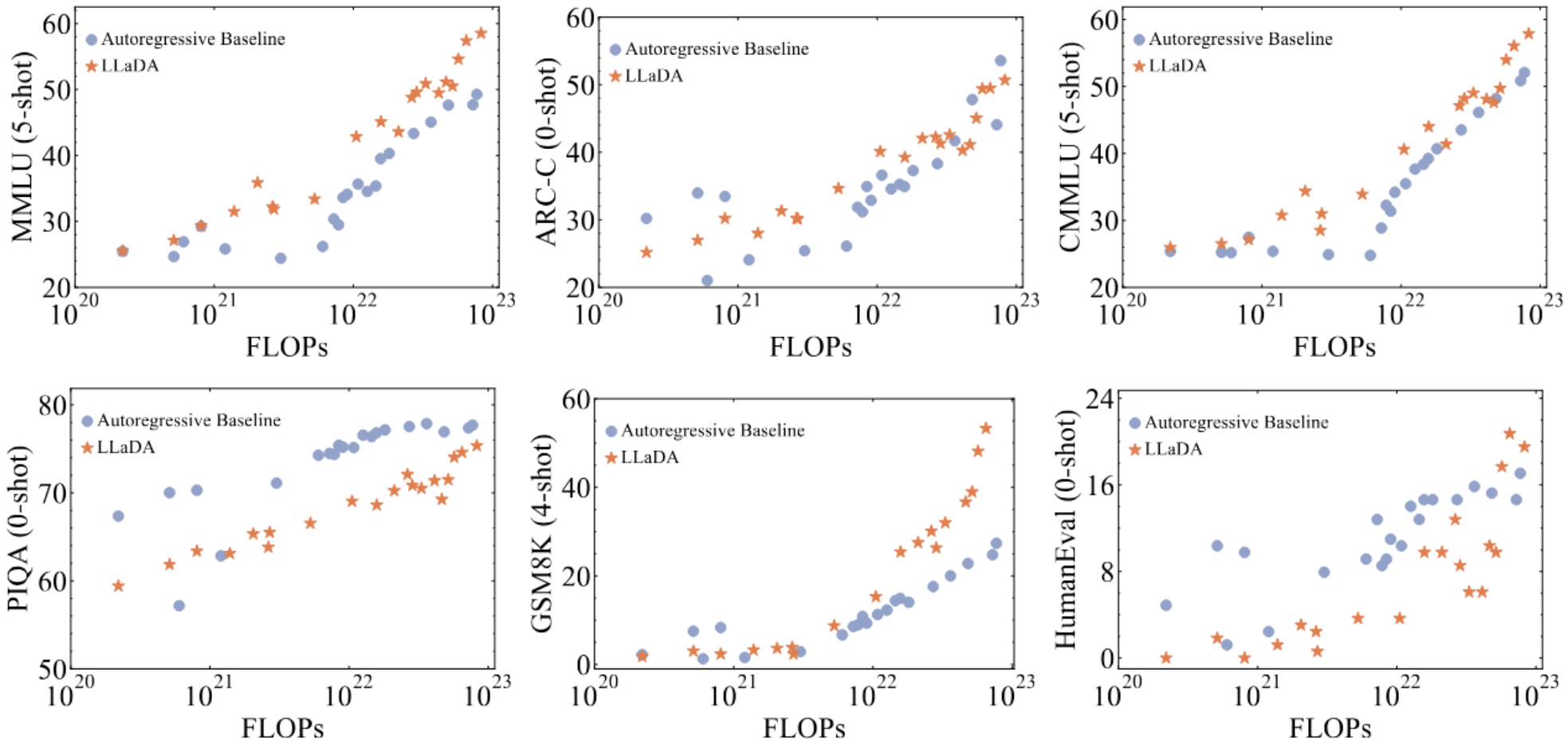
预训练：随机掩码

SFT：Response 掩码

生成：预测与保留
顺序由解码策略决定

arXiv:2502.09992

LLaDA：生成能力随训练算力扩展



arXiv:2502.09992

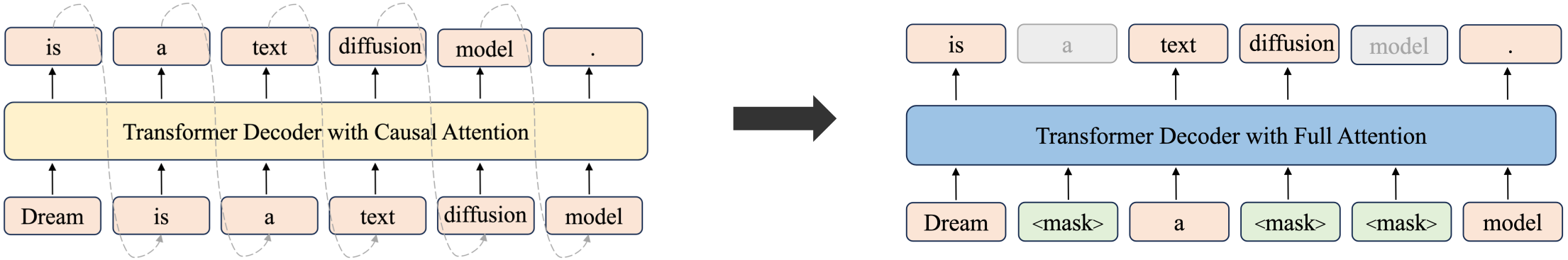
Dream：从已有 AR 模型出发

已有 **AR** 权重 → 扩散继续训练 → 掩码扩散模型

继承语言能力

适应随机掩码与双向上下文

多轮并行生成



两条路径：从零预训练，以及从 **AR** 模型转换。

arXiv:2508.15487

模型进展：从可行性到规模与应用

2025 上半年	**LLaDA • Dream • LLaDA 1.5** 可行性、AR 转换、后训练
2025 下半年	**LLaDA-MoE • LLaDA 2.0** 稀疏扩展与百亿/千亿规模
2026	**LLaDA 2.1 / 2.2** 填空、编辑与工具交互
商业路线	**Mercury 系列** 面向交互应用的产品化

各项目官方发布记录，更新至 2026.10

报告提纲

- 基础原理与模型进展
- 模型训练与生成质量
- 推理算法与系统优化
- 应用方向与未来展望

评价效率：质量、并行度与实际速度

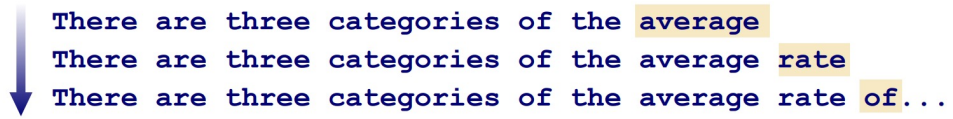
实际速度 ≈ 每轮有效提交数 / 每轮耗时

生成质量	答案正确率 / 代码通过率
算法并行度	**TPF (Token Per Forward)**：每轮实际提交的 **token** 数
执行效率	**TPS (Token Per Second)**、延迟，以及显存与硬件成本

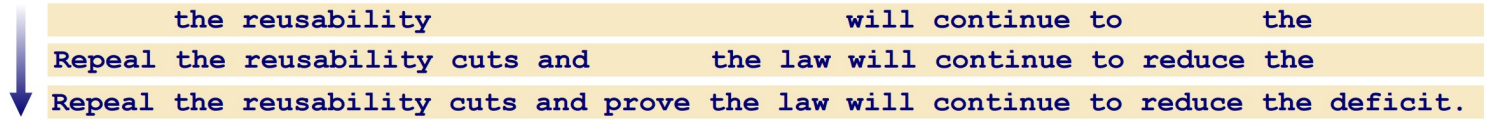
Block Diffusion：块内扩散，块间自回归

Autoregression: ✓ High quality ✓ Arbitrary-length ✓ KV caching ✗ Not Parallelizable

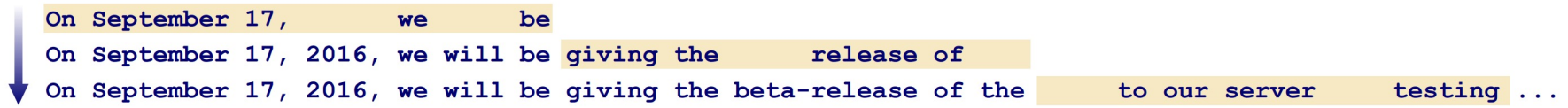
Generation steps



Diffusion: ✗ Lower quality ✗ Fixed-length ✗ No KV caching ✓ Parallelizable



Block Diffusion (Ours): ✓ High quality ✓ Arbitrary-length ✓ KV caching ✓ Parallelizable

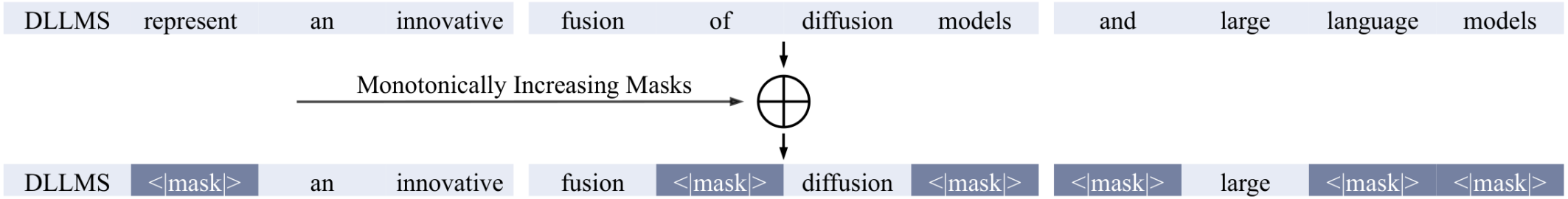


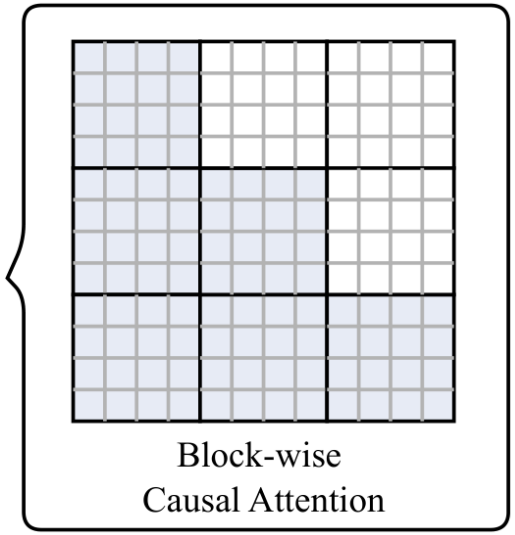
块间：完成后缓存，再启动下一块　　块内：多个位置同时预测，逐步完成当前块

后续块，能否在前序块完成之前就开始生成？

Arriola et al., Block Diffusion

Discrete diffusion forcing (D2F)：利用不完整的前序块



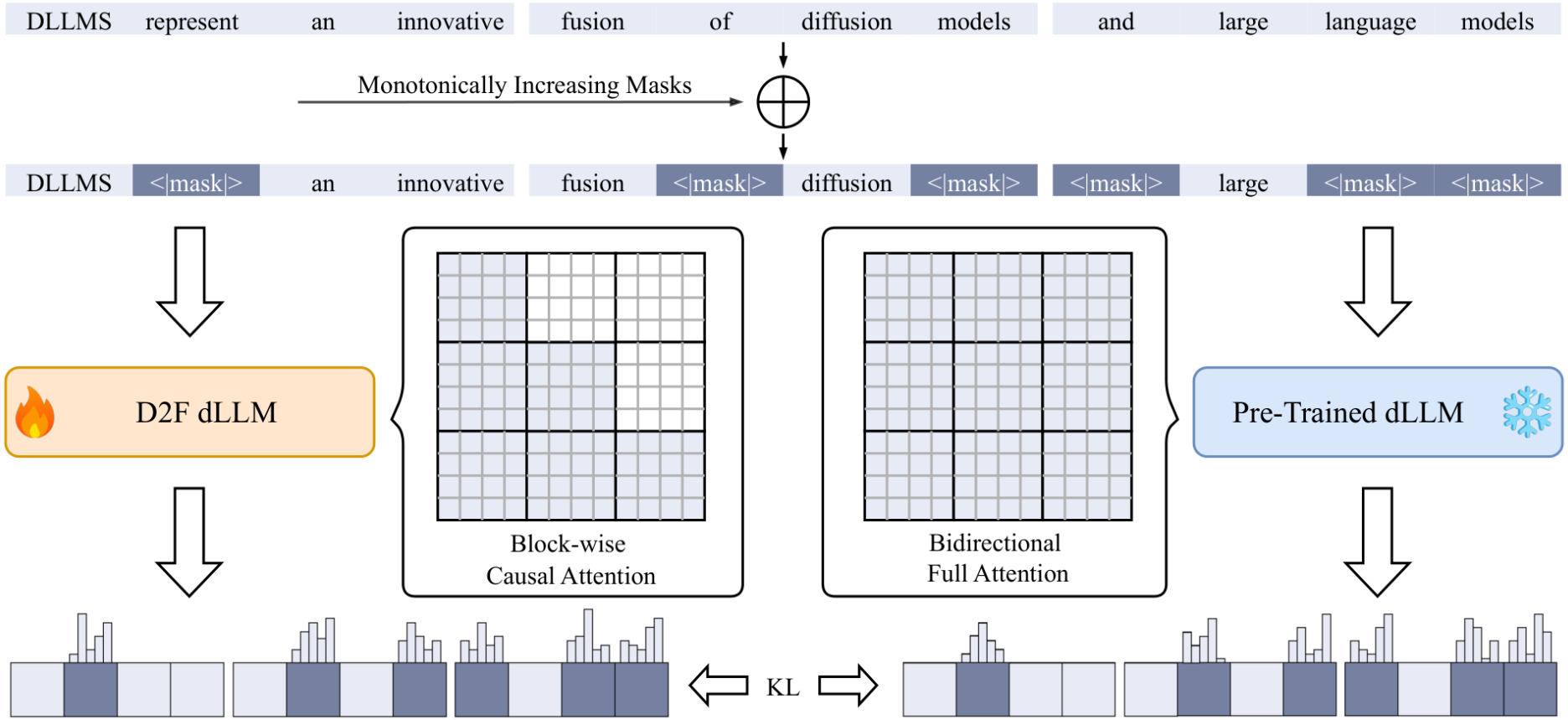


前面的块更完整，
后面的块噪声更高。

块内双向、块间因果，
后续块不必等待前序块全部完成。

arXiv:2508.09192

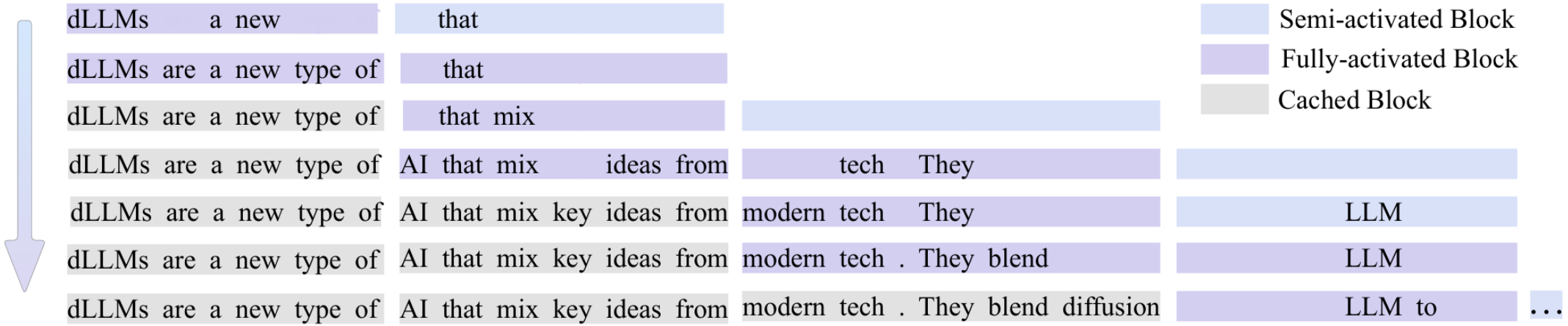
D2F：同一噪声输入，两种可见范围



冻结教师提供双向预测，学生学习在块因果视野下生成。

arXiv:2508.09192

D2F 流水线：新增、激活、缓存



新增：末块达到进度阈值，追加新块

激活：先保守提交，再允许保底推进

缓存：完成块写入 **KV cache**，后续块继续生成

arXiv:2508.09192

D2F：跨块生成带来实际加速

Test Set	Method	TPS ↑	Latency (s) ↓	Gen. Length	Score ↑
GSM8K 4-shot	LLaDA-Instruct	7.2 (1.0x)	32.3 (1.0x)	231	77.4
	dLLM-Cache	20.1 (2.8x)	11.5 (2.8x)	231	77.5
	Fast-dLLM (Prefix-Cache)	33.3 (4.6x)	7.0 (4.6x)	232	77.8
	Fast-dLLM (Dual-Cache)	35.2 (4.9x)	6.6 (4.9x)	232	**78.9**
	D2F-LLaDA	**52.5** (7.3x)	**2.8** (11.5x)	144	77.3
MBPP 3-shot	LLaDA-Instruct	0.9 (1.0x)	71.4 (1.0x)	65	**39.0**
	dLLM-Cache	2.3 (2.6x)	28.3 (2.5x)	66	37.0
	Fast-dLLM (Prefix-Cache)	13.0 (14.4x)	4.9 (14.6x)	64	37.6
	Fast-dLLM (Dual-Cache)	15.3 (17.0x)	3.8 (18.8x)	58	36.4
	D2F-LLaDA	**47.6** (52.9x)	**1.4** (51.0x)	68	38.0
HumanEval 0-shot	LLaDA-Instruct	2.8 (1.0x)	38.8 (1.0x)	107	36.0
	dLLM-Cache	4.5 (1.6x)	23.3 (1.7x)	104	39.0
	Fast-dLLM (Prefix-Cache)	13.7 (4.9x)	7.4 (5.2x)	102	38.4
	Fast-dLLM (Dual-Cache)	19.2 (6.9x)	5.2 (7.5x)	100	35.4
	D2F-LLaDA	**81.6** (29.1x)	**1.6** (24.3x)	133	**40.2**
Math 4-shot	LLaDA-Instruct	21.1 (1.0x)	11.5 (1.0x)	243	23.7
	dLLM-Cache	26.9 (1.3x)	9.1 (1.3x)	246	23.2
	Fast-dLLM (Prefix-Cache)	47.7 (2.3x)	5.2 (2.2x)	246	22.4
	Fast-dLLM (Dual-Cache)	42.5 (2.0x)	5.8 (2.0x)	246	22.4
	D2F-LLaDA	**90.2** (4.3x)	**4.3** (2.7x)	384	**29.1**

Table 1: Performance comparison of various acceleration methods on the **LLaDA-Instruct** model. Speedup ratios relative to the baseline are shown in (green). All baseline methods use the default sampling configuration from the original LLaDA implementation.

LLaDA-Instruct-8B
→ D2F-LLaDA

MBPP 3-shot

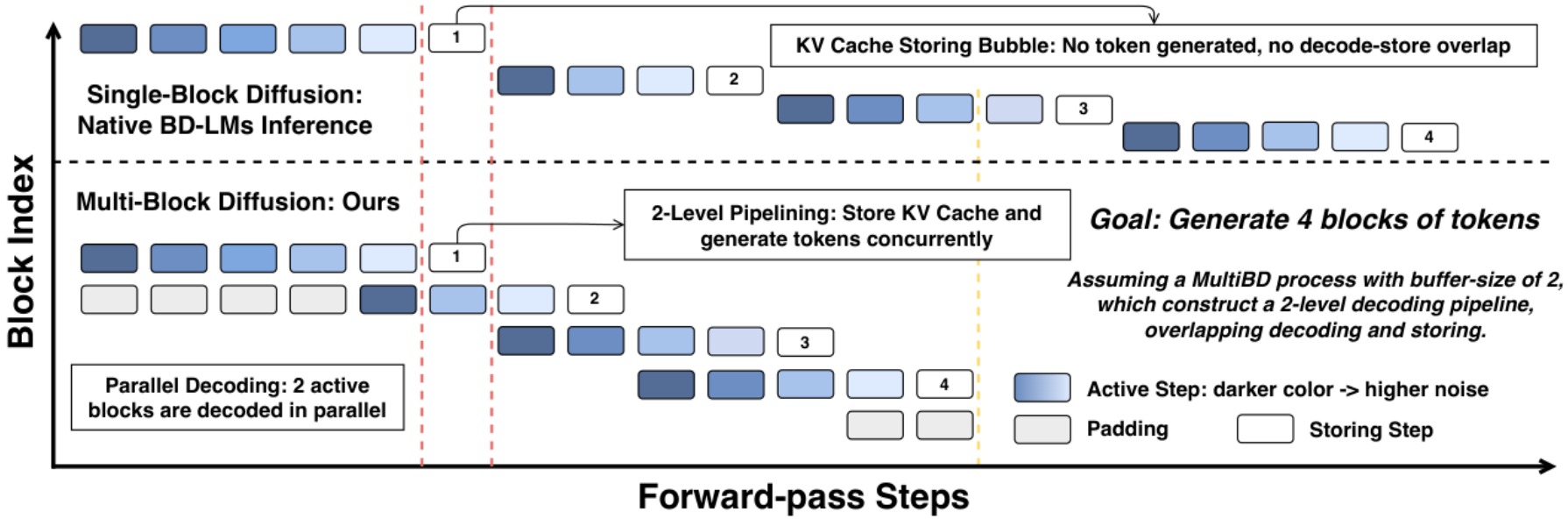
0.9 → 47.6 TPS

52.9×

任务得分
39.0 → 38.0

arXiv:2508.09192, Table 1

D2F：在有限窗口中同时推进多个块

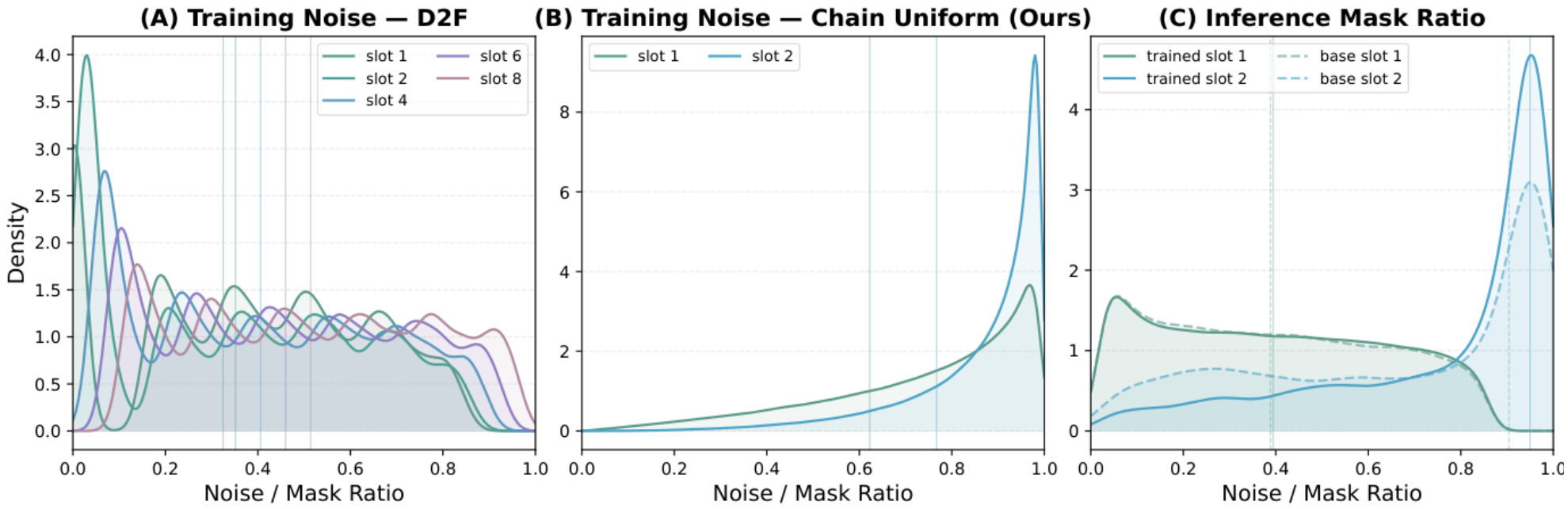


活跃块同时去噪，完成块提交缓存

新块填入空位，运行窗口不断前移

arXiv:2606.29215, Fig. 1

D2F：训练与推理的槽位噪声存在差异

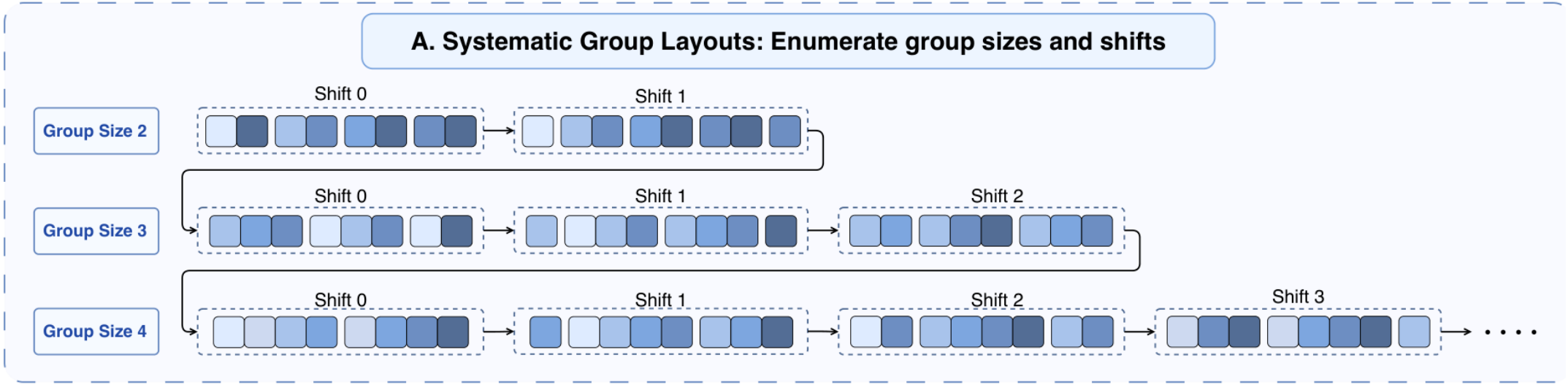


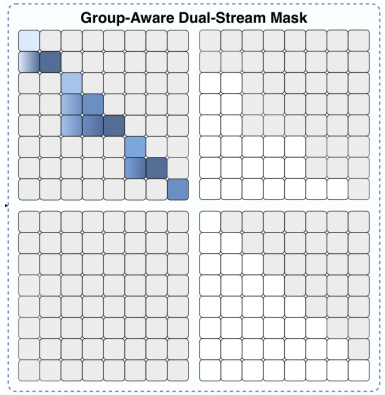
不同槽位具有不同的 **mask** 比例分布

窗口结构与噪声间隔，都需要进入训练设计

arXiv:2606.29215, Fig. 2

Multi-block diffusion (MultiBD)：覆盖组大小、位置与噪声间隔





组大小与位置：覆盖不同运行窗口

组感知注意力：限定可见的前缀与噪声块

噪声间隔：覆盖各槽位的不同生成进度

arXiv:2606.29215, Fig. 4

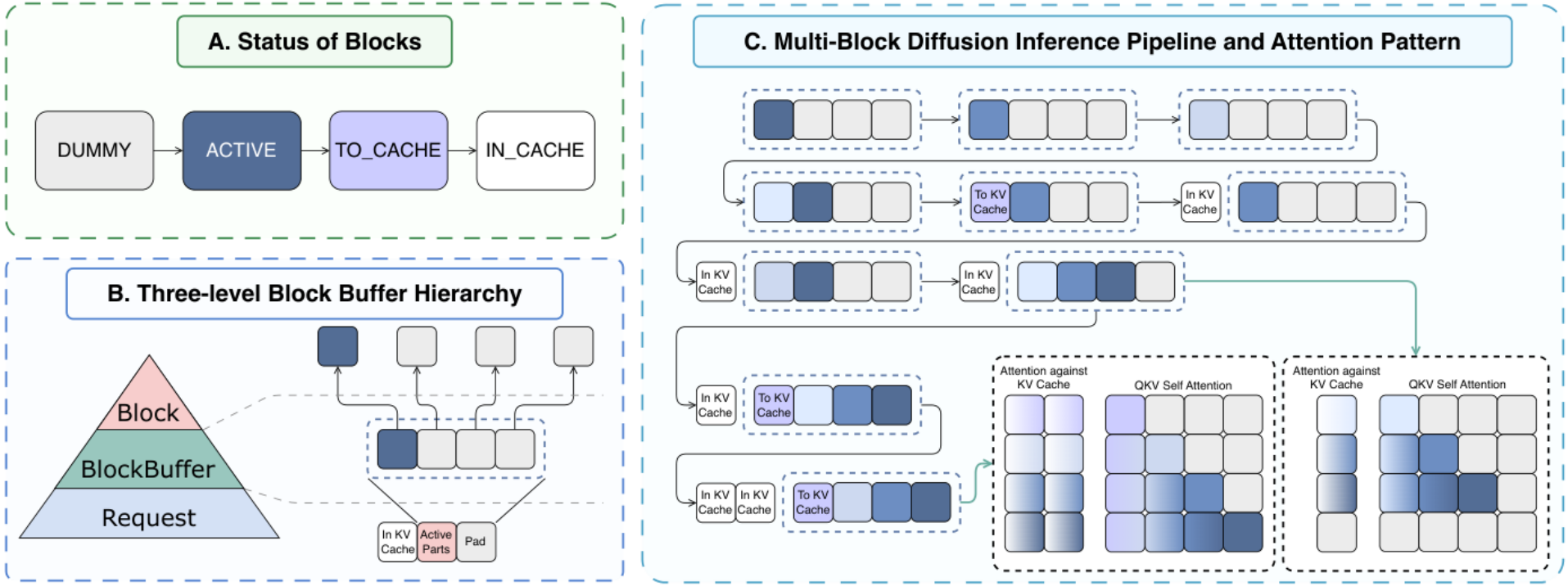
MultiBD：改善多块生成的质量与并行度

Model	GSM8K		MATH500		MBPP+		HumanEval+		Average		
	Acc ↑	TPF ↑	Acc ↑	TPF ↑	Acc ↑	TPF ↑	Acc ↑	TPF ↑	Acc ↑	TPF ↑	AUP ↑
LLaDA2-Mini-DMax *(bufsz=2, blksz=32)*											
SingleBD (Native)	**91.89**	5.70	**76.80**	6.13	**72.22**	6.14	**77.44**	7.44	**79.59**	6.35	459.54
MultiBD (training-free)	89.84	8.76	73.80	9.08	**72.22**	**8.44**	76.83	**10.96**	78.17	9.31	651.98
MBD-LLaDA2-Mini-DMax	91.74	**8.95**	75.00	**9.31**	70.11	8.34	**77.44**	10.78	78.57	**9.34**	**661.28**
LLaDA2-Mini *(bufsz=2, blksz=32)*											
SingleBD (Native)	91.89	2.27	74.20	2.83	**75.66**	3.25	78.05	5.53	79.95	3.47	247.41
MultiBD (training-free)	**92.65**	2.76	73.60	3.53	72.49	3.97	75.61	7.37	78.59	4.41	301.81
MBD-LLaDA2-Mini	91.96	**5.55**	**79.20**	**6.02**	72.49	**5.35**	**80.49**	**7.85**	**81.03**	**6.19**	**449.18**
SDAR-8B-Chat-b32 *(bufsz=4, blksz=32)*											
SingleBD (Native)	**90.07**	2.52	65.60	3.81	52.65	1.83	**67.68**	2.00	69.00	2.54	141.64
MultiBD (training-free)	89.01	2.78	60.60	5.06	52.12	1.97	65.85	2.24	66.89	3.01	156.35
MBD-SDAR-8B-Chat-b32	89.16	**3.08**	**68.00**	**5.08**	**58.99**	**4.87**	62.80	**4.82**	**69.74**	**4.46**	**210.42**
SDAR-8B-Chat-b4 *(bufsz=4, blksz=4)*											
SingleBD (Native)	91.05	1.33	**72.80**	1.46	64.80	1.13	73.70	1.07	**75.59**	1.25	85.46
MultiBD (training-free)	90.45	**2.39**	70.60	**2.68**	**65.80**	1.55	**74.39**	1.47	75.31	2.00	129.59
MBD-SDAR-8B-Chat-b4	**91.81**	2.28	72.40	2.52	64.29	**2.62**	72.56	**2.24**	75.27	**2.42**	**148.65**

LLaDA2-Mini：平均 TPF 3.47 → 6.19，得分 79.95 → 81.03

arXiv:2606.29215, Table 1

Block Buffer：动态运行，固定物理槽位



完成块进入 **KV cache**，新块复用空闲槽位

静态输入形状，减少动态多块调度的执行开销

arXiv:2606.29215, Fig. 5

Diffulex：把多块生成转化为实际速度

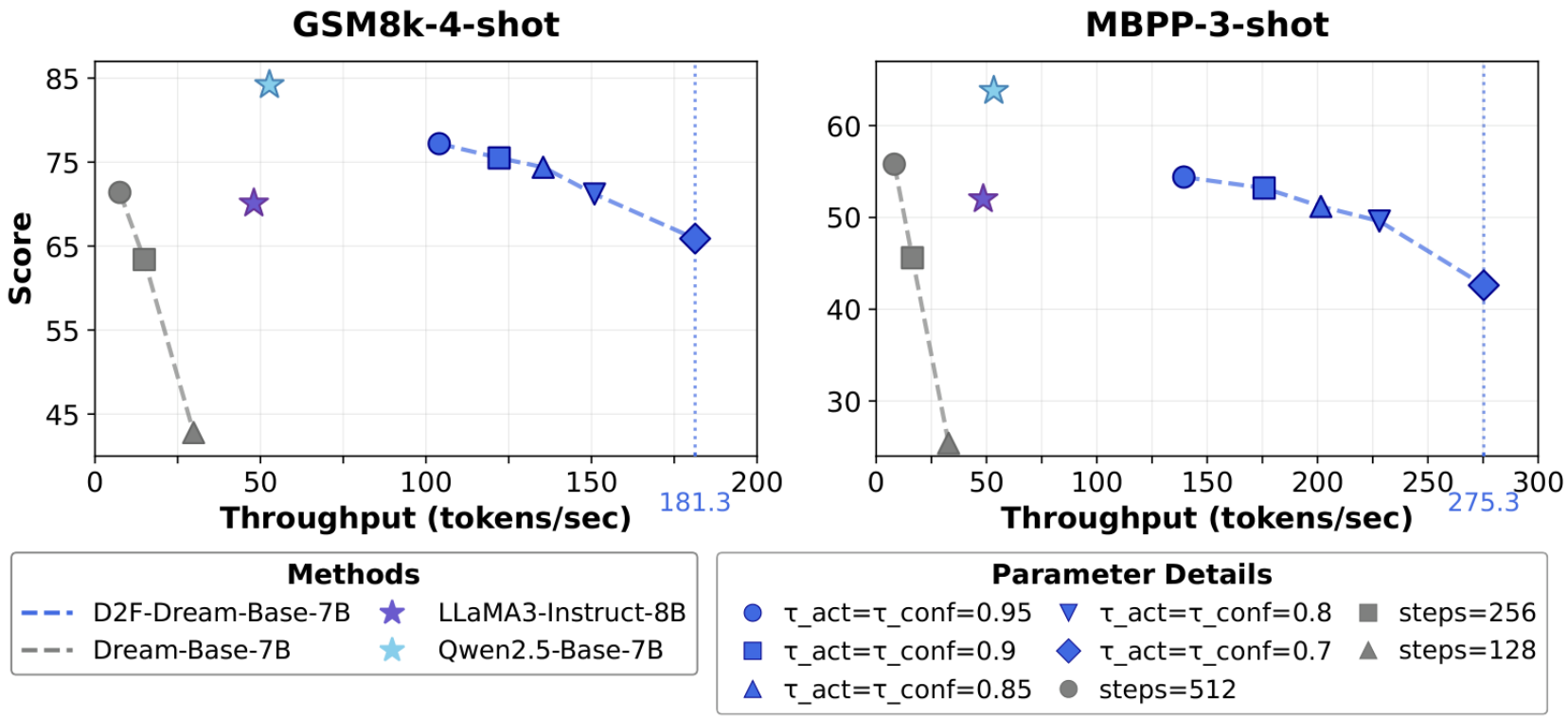
单样本，2 × H100，TP = 2

模型	平均 TPF	单步 ms	平均 TPS
LLaDA2-Mini	3.47	7.07	517.16
MBD-LLaDA2-Mini	**6.19**	**8.78**	**745.92**
LLaDA2-Mini-DMax	6.35	9.02	779.49
MBD-LLaDA2-Mini-DMax	**9.34**	**11.20**	**926.67**

每步产出提高，同时也要计入单步耗时的增加。

arXiv:2606.29215, Table 3

更快的解码，仍存在质量与速度的折衷



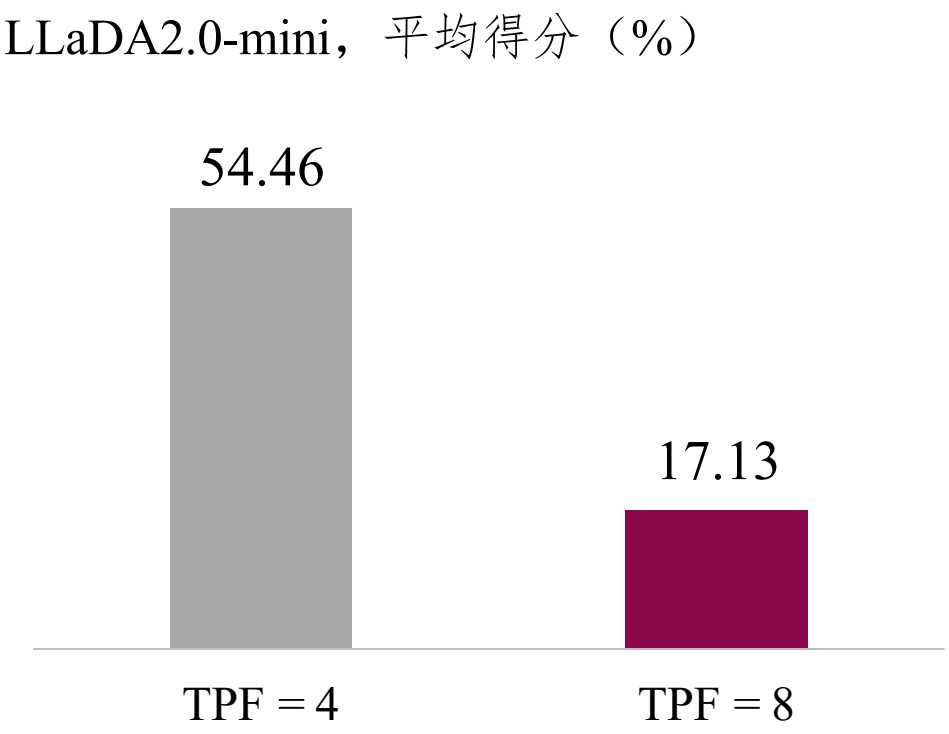
降低解码阈值可以提速，也可能降低生成质量。

arXiv:2508.09192

Consistency Force (CForce)：用后期预测改进早期生成

早期上下文不完整，
一次提交更多会放大误判

后期上下文更充分，
可以为早期预测提供监督



核心信号：让早期预测接近后期更充分的预测。

arXiv:2608.13925v1, Fig. 5

CForce：在自身解码轨迹上学习一致性

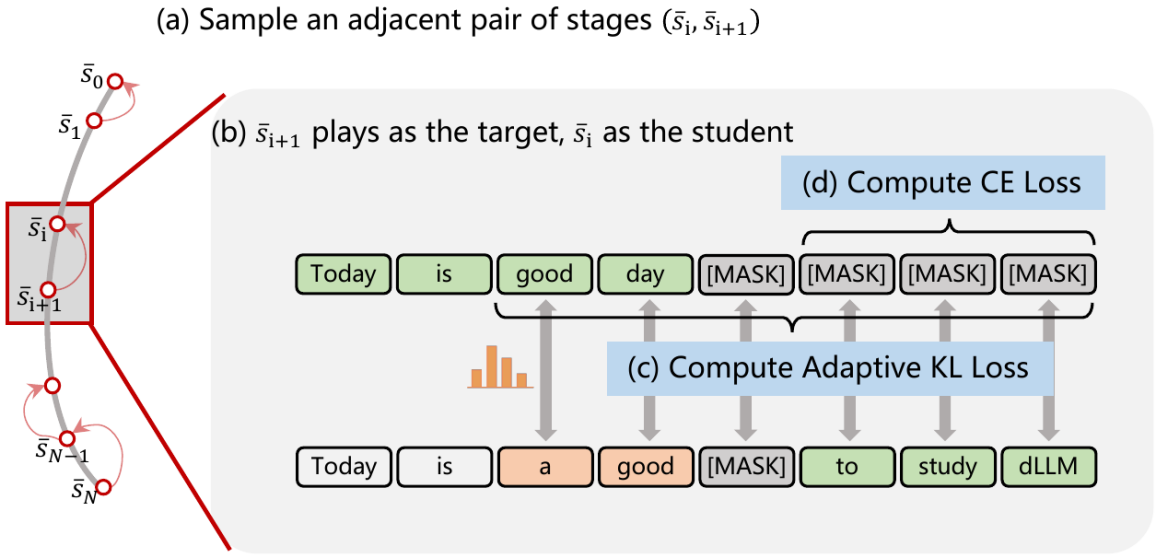
1. Collect Confidence-based Trajectory $\mathcal{T} = (s_0, s_1, \ldots, s_T)$

Step 0	[MASK]	[MASK]	[MASK]	[MASK]	[MASK]	[MASK]	[MASK]	[MASK]
Step 1	Today	is	[MASK]	[MASK]	[MASK]	[MASK]	[MASK]	[MASK]
Step 2	Today	is	good	day	[MASK]	[MASK]	[MASK]	[MASK]
Step 3	Today	is	a	good	[MASK]	[MASK]	[MASK]	dLLM
Step 4	Today	is	a	good	[MASK]	to	study	dLLM
Step 5	Today	is	a	good	day	to	study	dLLM

2. Construct Staged Trajectory $\overline{\mathcal{T}} = (\bar{s}_0, \bar{s}_1, \ldots, \bar{s}_N)$

Stage 0	[MASK]	[MASK]	[MASK]	[MASK]	[MASK]	[MASK]	[MASK]	[MASK]
Stage 1	Today	is	good	day	[MASK]	[MASK]	[MASK]	[MASK]
Stage 2	Today	is	a	good	[MASK]	to	study	dLLM
Stage 3	Today	is	a	good	day	to	study	dLLM

3. Train with Adjacent-stage Consistency Forcing



相邻阶段形成监督对，后期目标停止梯度。

arXiv:2608.13925v1, Fig. 4

CForce：同时约束预测变化与提交错误

$$\mathcal{L} = \mathcal{L}_{\text{CAD}} + \lambda_{\text{CE}}\mathcal{L}_{\text{CE}},$$

分布对齐：置信度自适应 KL

$$\mathcal{L}_{\text{CAD}} = \frac{1}{|\mathcal{U}_i|} \sum_{j \in \mathcal{U}_i} \left[D_{\text{KL}}(p^j_{\text{tar}} \| p^j_{\text{stu}}) + c_j D_{\text{KL}}(p^j_{\text{stu}} \| p^j_{\text{tar}}) \right],$$

学生分布：早期预测　目标分布：后期预测（停止梯度）　c_j：目标置信度

提交监督：CE Anchor

$$\mathcal{L}_{\text{CE}} = -\frac{1}{|\Delta_i|} \sum_{j \in \Delta_i} \log q_\theta(y^j \mid x, \bar{s}_i, j),$$

在新揭示位置，学习正确提交目标 token

arXiv:2608.13925v1, §4.3

CForce：同一设置下，质量与并行度共同改善

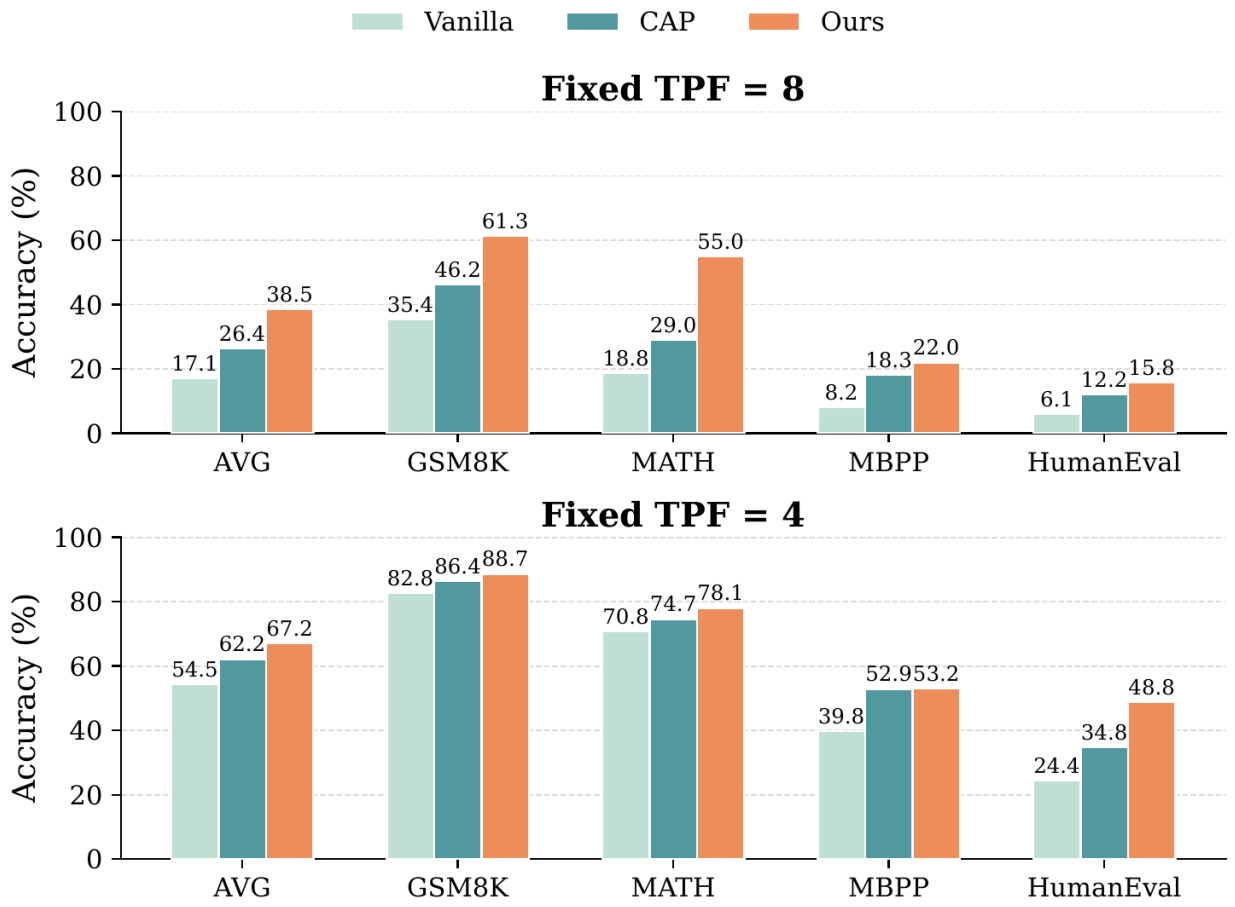
LLaDA2.1-mini；相同解码阈值与块长

任务	基座得分	CForce 得分	基座 TPF	CForce TPF
GSM8K	93.56	92.27	5.94	7.63
MATH500	85.00	84.80	7.44	10.15
MBPP	77.75	81.97	7.25	10.07
HumanEval	85.98	86.59	7.11	8.48
平均	**85.57**	**86.41**	**6.94**	**9.08**

平均 TPF +30.8%，平均得分 +0.84 个百分点

arXiv:2608.13925v1, Table 1

CForce：LLaDA2.0-mini 的固定 TPF 结果



TPF = 8

17.13 → 38.54

+21.41 个百分点

TPF = 4

54.46 → 67.19

+12.73 个百分点

arXiv:2608.13925v1, Fig. 5

CForce：并行能力转化为真实吞吐收益

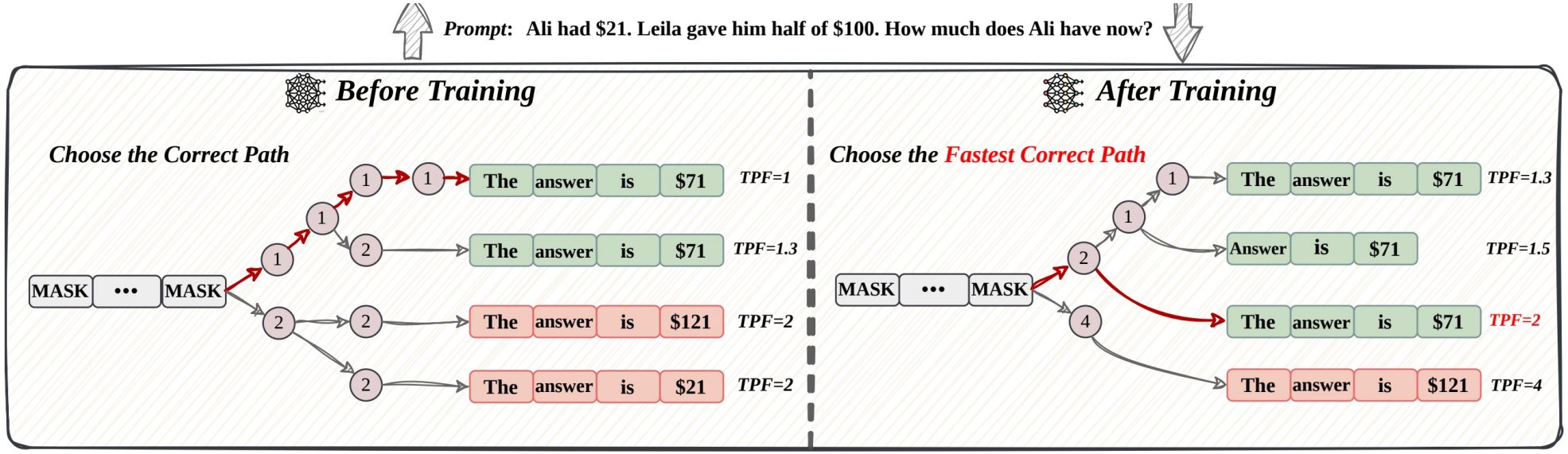
2 × NVIDIA H20，TP = 2，batch = 1

模型	GSM8K TPS	HumanEval TPS	平均 TPS
LLaDA2.0-mini	489.87	711.59	600.73
2.0 + CAP	710.88	891.94	801.41
2.0 + CForce	**783.09**	**932.82**	**857.96**
LLaDA2.1-mini	987.78	1116.10	1051.94
2.1 + CForce	**1231.52**	**1352.86**	**1292.19**

非编辑模型 +42.82%　　可编辑模型 +22.84%

arXiv:2608.13925v1, Appendix E

LightningRL：学习正确且高并行的轨迹

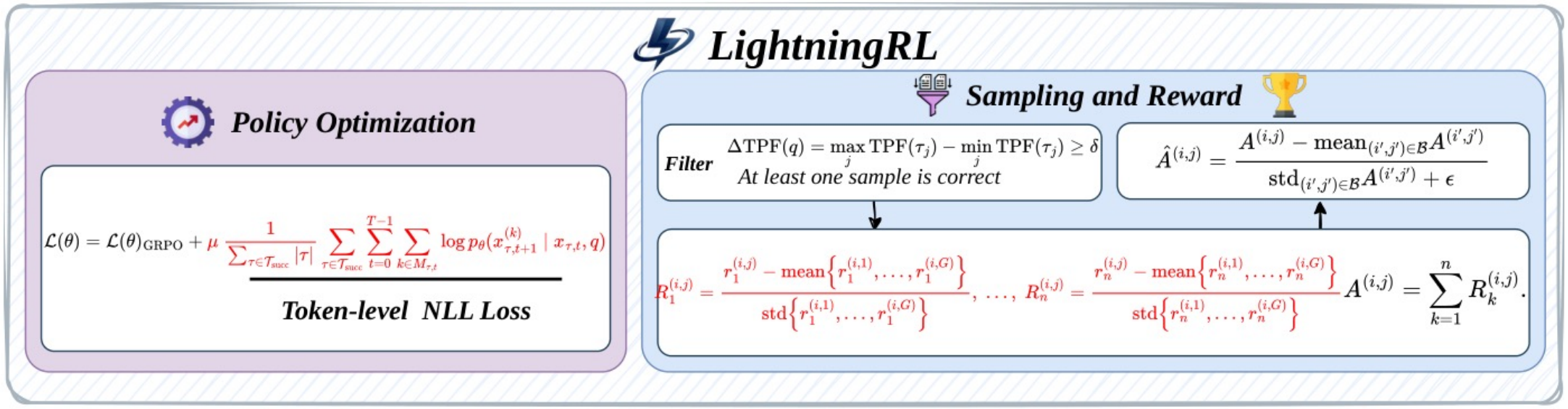


同一问题，采样多条生成轨迹

提高既正确、又能较少轮次完成的行为概率

arXiv:2603.13319v2, Fig. 2

LightningRL：质量与 TPF 构成联合训练信号

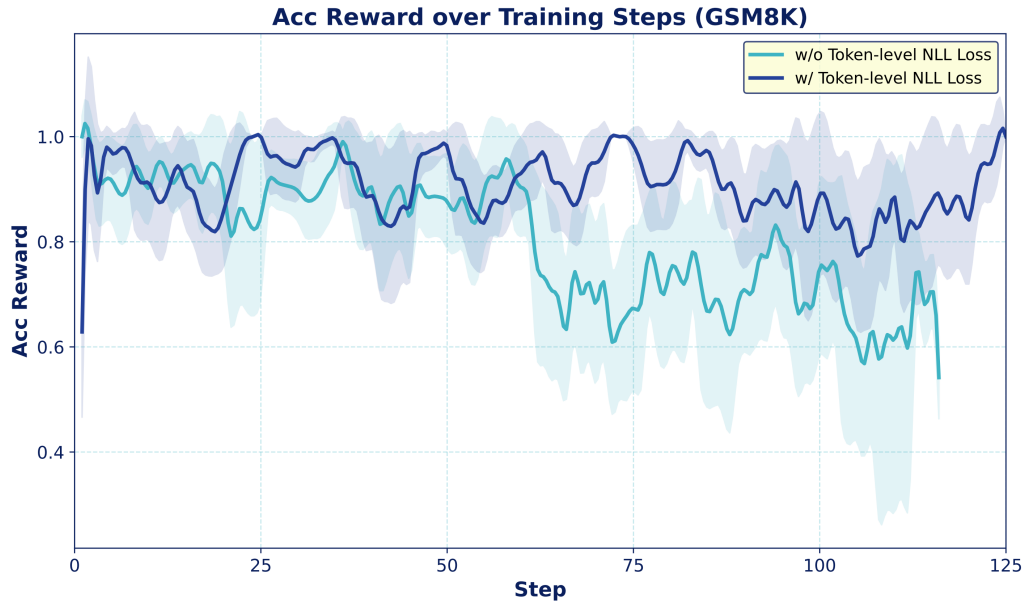


正确性奖励与 **TPF** 奖励分别标准化

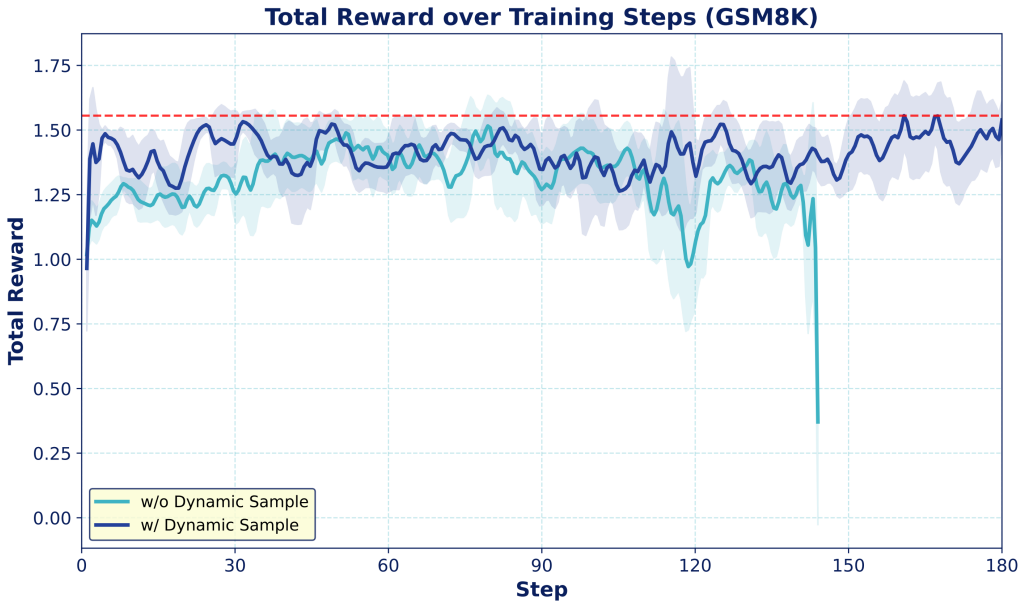
以组内比较提供信号，进行 **GRPO** 式更新

arXiv:2603.13319v2, Fig. 2

LightningRL：保留正确行为，筛选有效轨迹组



正确轨迹上的 **NLL**



具有 **TPF** 差异的有效轨迹组

arXiv:2603.13319v2, Figs. 4–5

LightningRL：同一基座上的质量与并行度

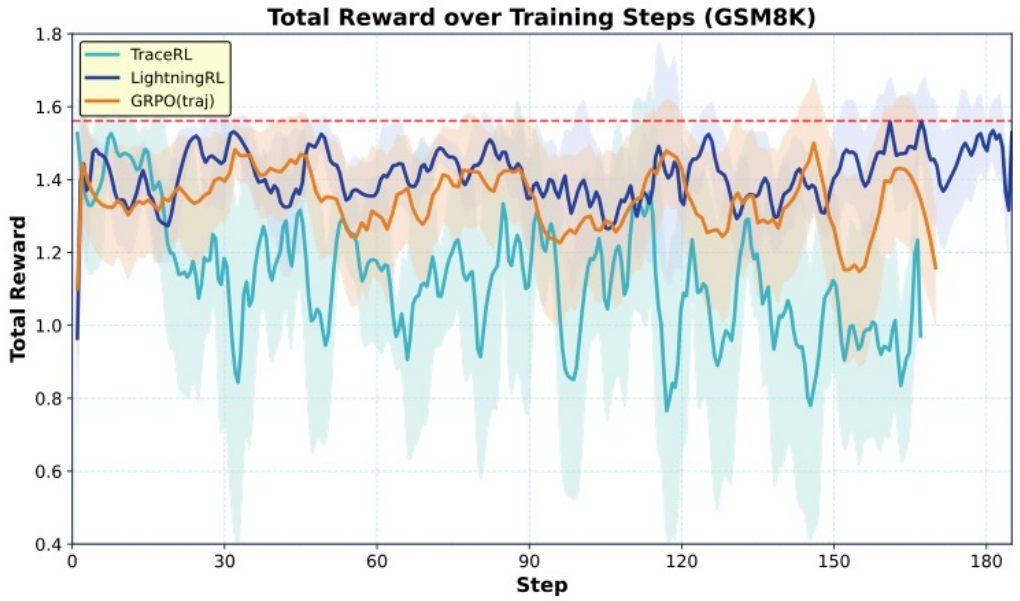
Model	GSM8K			MATH500			MBPP			HumanEval		
	Acc (%)	TPF	AUP	Acc (%)	TPF	AUP	Acc (%)	TPF	AUP	Acc (%)	TPF	AUP
SDAR (Cheng et al., 2025)	88.9	2.85	252.5	**63.6**	4.81	318.9	58.0	2.44	108.4	73.5	2.39	165.0
Coupled-GRPO (Gong et al., 2025)	75.3	4.22	254.7	59.1	4.93	283.1	54.0	2.61	139.9	68.2	2.42	164.6
TraceRL (Wang et al., 2025b)	76.9	5.04	378.6	60.6	4.82	284.6	57.8	2.50	144.2	75.0	2.29	171.6
DiRL (Zhu et al., 2026)	86.6	4.87	414.1	61.6	5.04	301.2	56.8	2.70	152.7	**76.0**	2.33	176.7
LightningRL	**90.3**	**5.58**	**507.5**	63.0	**6.28**	**409.2**	**58.3**	**11.10**	**412.9**	72.6	**6.30**	**421.5**

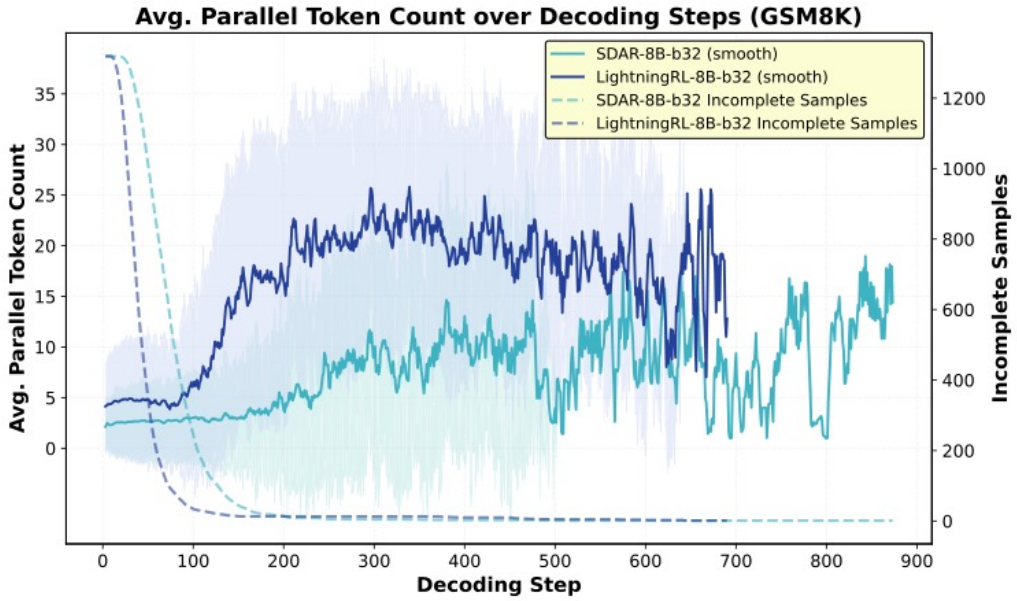
四任务平均 **TPF：7.32**

MBPP：11.10 TPF，同时保留各任务质量比较

arXiv:2603.13319v2, Table 1

LightningRL：更稳定的训练，更短的解码长尾

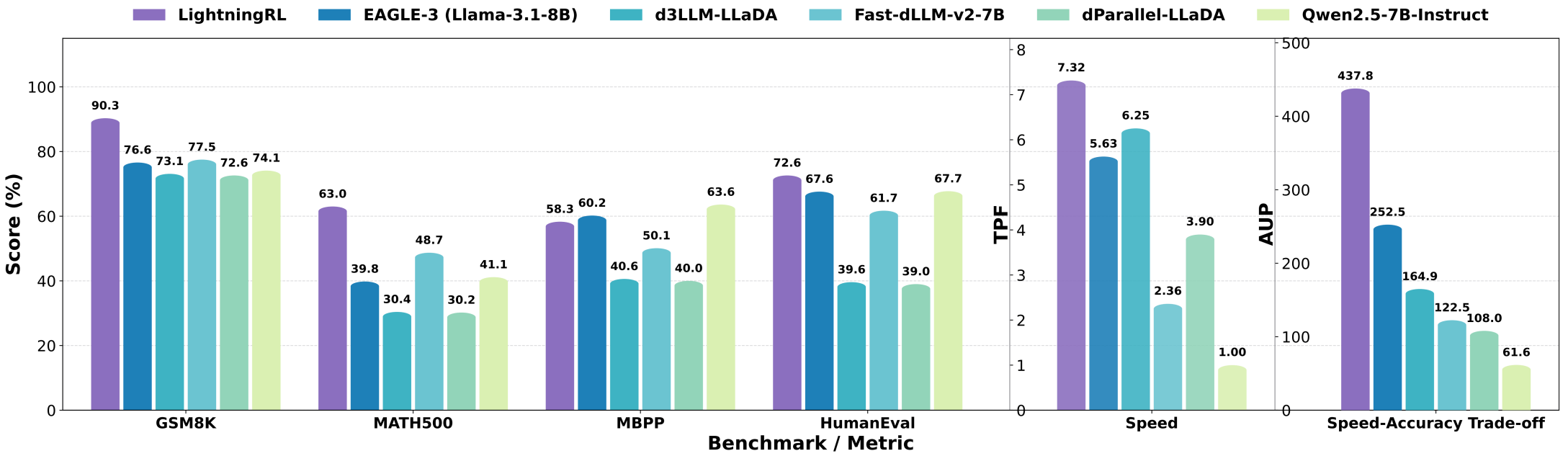




有效提交量提高，样本更早完成，减少长尾等待。

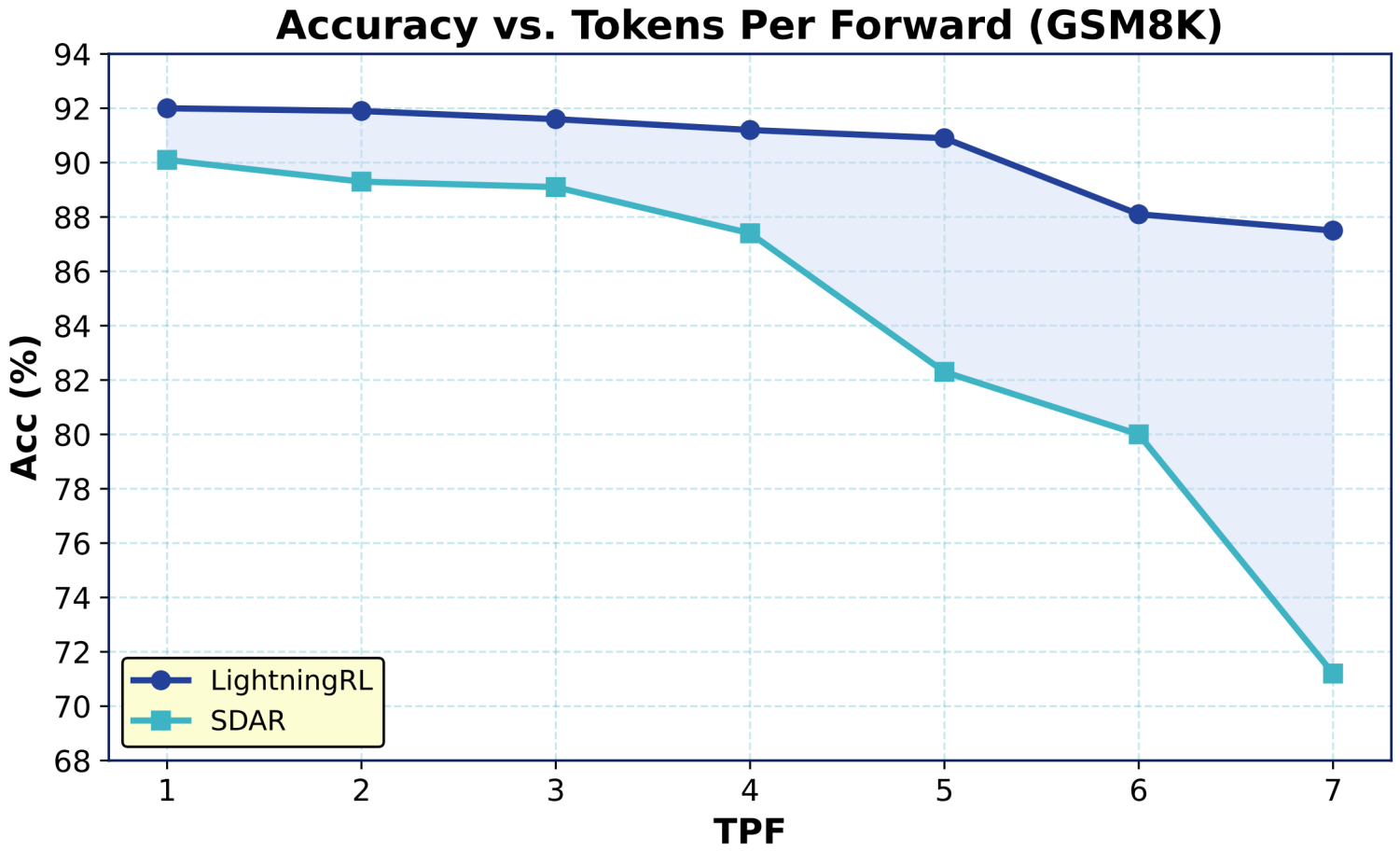
arXiv:2603.13319v2, Figs. 6–7

LightningRL：质量与并行度的综合比较



同时报告任务得分、TPF 与质量—并行度折衷。

LightningRL：相同 TPF 下，更高的质量



相同 **TPF = 7**

71.2% → 87.5%

GSM8K 准确率

SDAR-8B / block32

arXiv:2603.13319v2

报告提纲

- 基础原理与模型进展
- 模型训练与生成质量
- 推理算法与系统优化
- 应用方向与未来展望

固定模型之后，怎样进一步提高生成速度？

TPS = 每轮有效提交量 ÷ 每轮耗时

减少每轮耗时：复用计算，降低重复开销

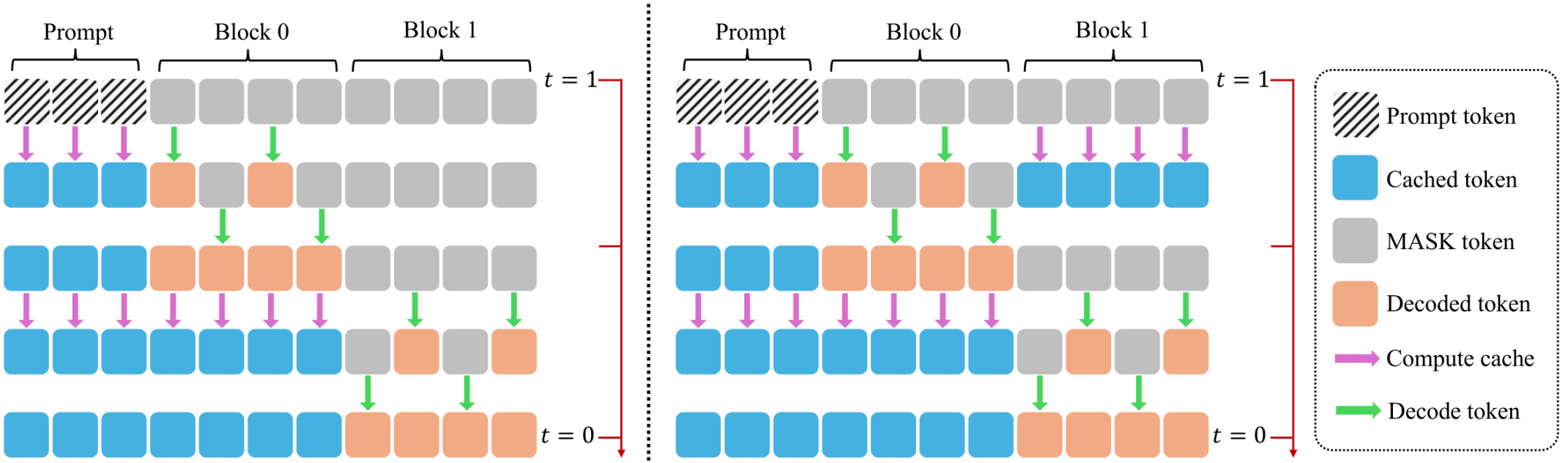
增加有效提交：选择可靠且有利的解码状态

固定模型的推理加速：Fast-dLLM 与 LoPA

Fast-dLLM 复用缓存，按当前置信度提交

LoPA 向前看一步，比较后续置信度

Fast-dLLM：用近似缓存减少重复计算



(a) Prefix KV Cache for block-wise generation.

(b) **DualCache**: Bidirectional KV cache contains prefix and suffix Cache.

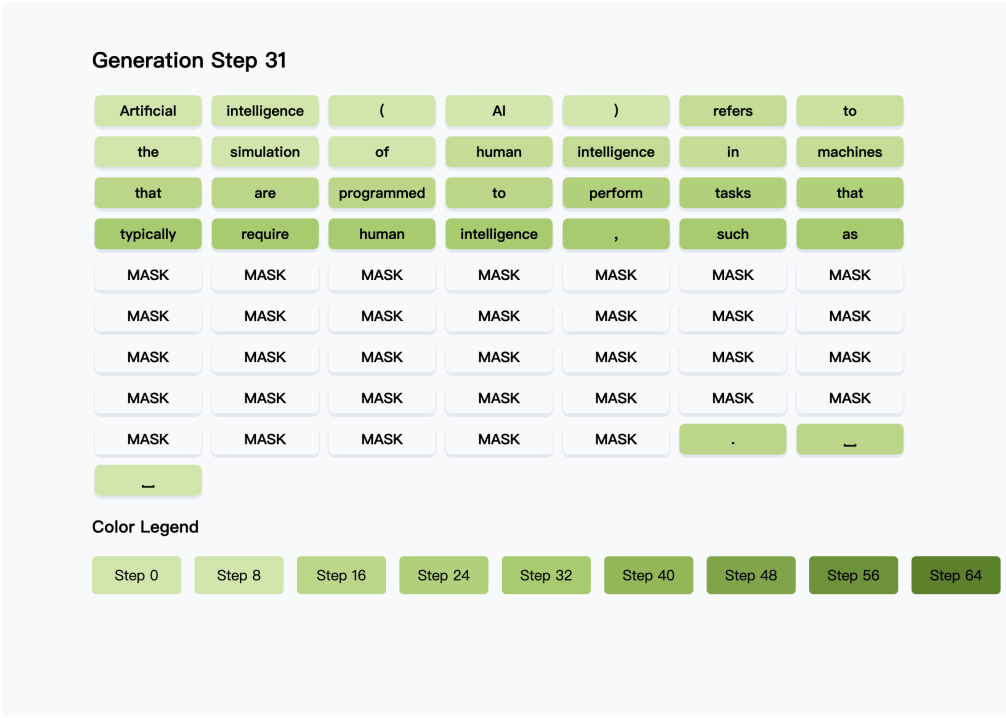
双向注意力下采用近似缓存，块内复用、块间刷新

DualCache 进一步复用前缀与掩码后缀的 KV

Fast-dLLM, arXiv:2505.22618v3, Fig. 2

Fast-dLLM：按置信度动态决定每轮提交量

标准解码（第 31 轮）



置信度解码（第 31 轮）



当前块内，超过阈值的位置一起提交，其余继续等待

没有位置达标时，仍提交当前置信度最高的位置

Fast-dLLM, arXiv:2505.22618v3; official decoding example

Fast-dLLM：缓存与置信度解码共同带来加速

LLaDA-Instruct，GSM8K 5-shot，生成上限 256
NVIDIA A100，batch = 1

方法	准确率（%）	TPF	TPS
原始 LLaDA	79.3	1.00	6.7
仅 KV Cache	79.5	1.00	21.2
仅置信度解码	79.2	3.25	16.5
Fast-dLLM（两者结合）	**78.5**	**3.01**	**54.4**

两者结合：TPS 提升 8.1×，准确率由 79.3% 变为 78.5%

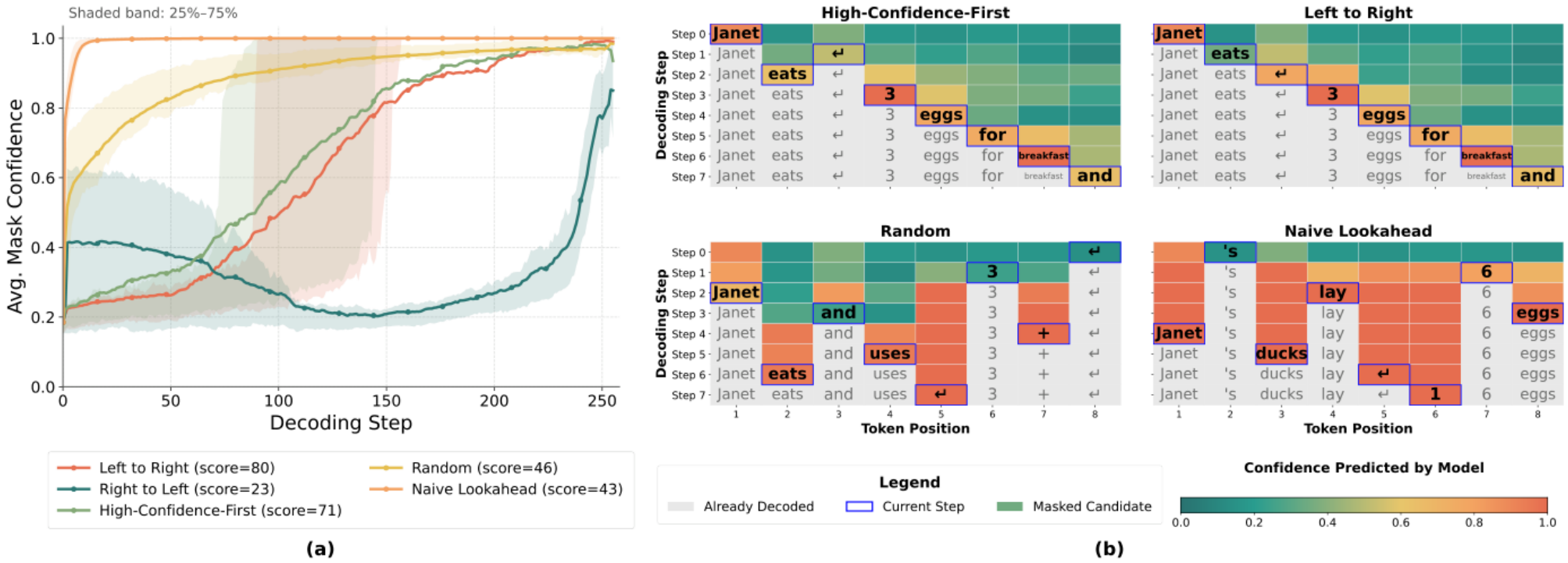
Fast-dLLM, arXiv:2505.22618v3, Table 1 & Fig. 1(b)

当前置信度之外，还能怎样选择下一步？

Fast-dLLM 复用缓存，按当前置信度提交

LoPA 向前看一步，比较后续置信度

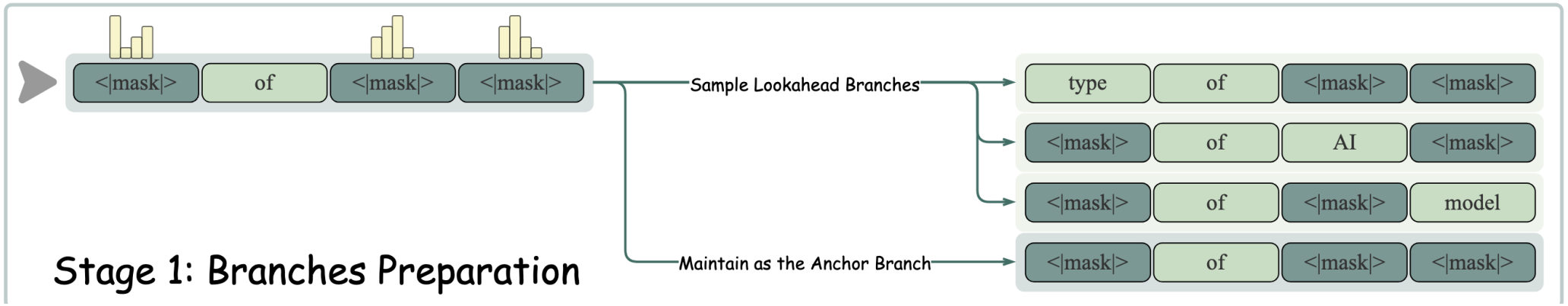
填充顺序同时影响置信度与生成质量



朴素前瞻：置信度更高，但得分仅 43（从左到右为 80）

LoPA, token filling order analysis

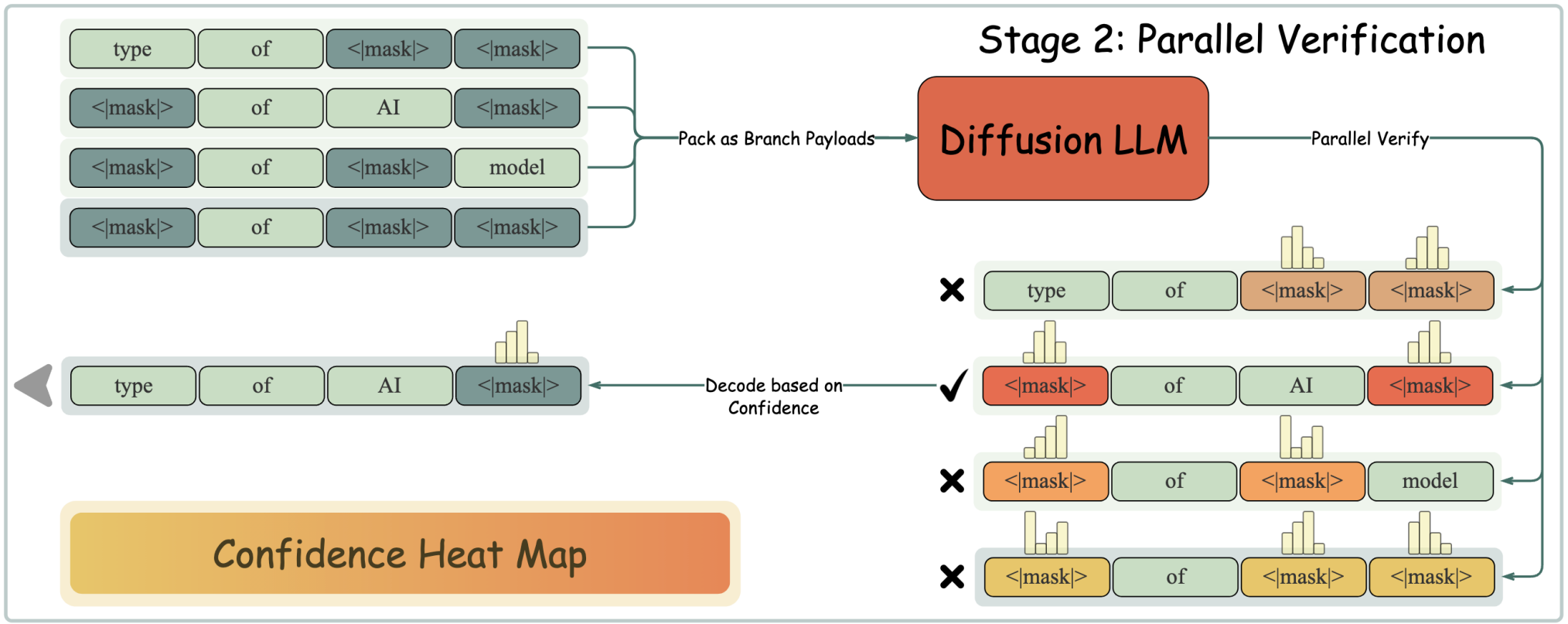
LoPA：保留锚点，构造前瞻候选



锚点：沿用置信度解码，保留基础路径

前瞻分支：每条路径额外填写一个候选位置

LoPA：比较后续置信度，复用验证结果



按剩余 **mask** 位置的平均置信度选分支，并复用 **logits**。

arXiv:2512.16229v2

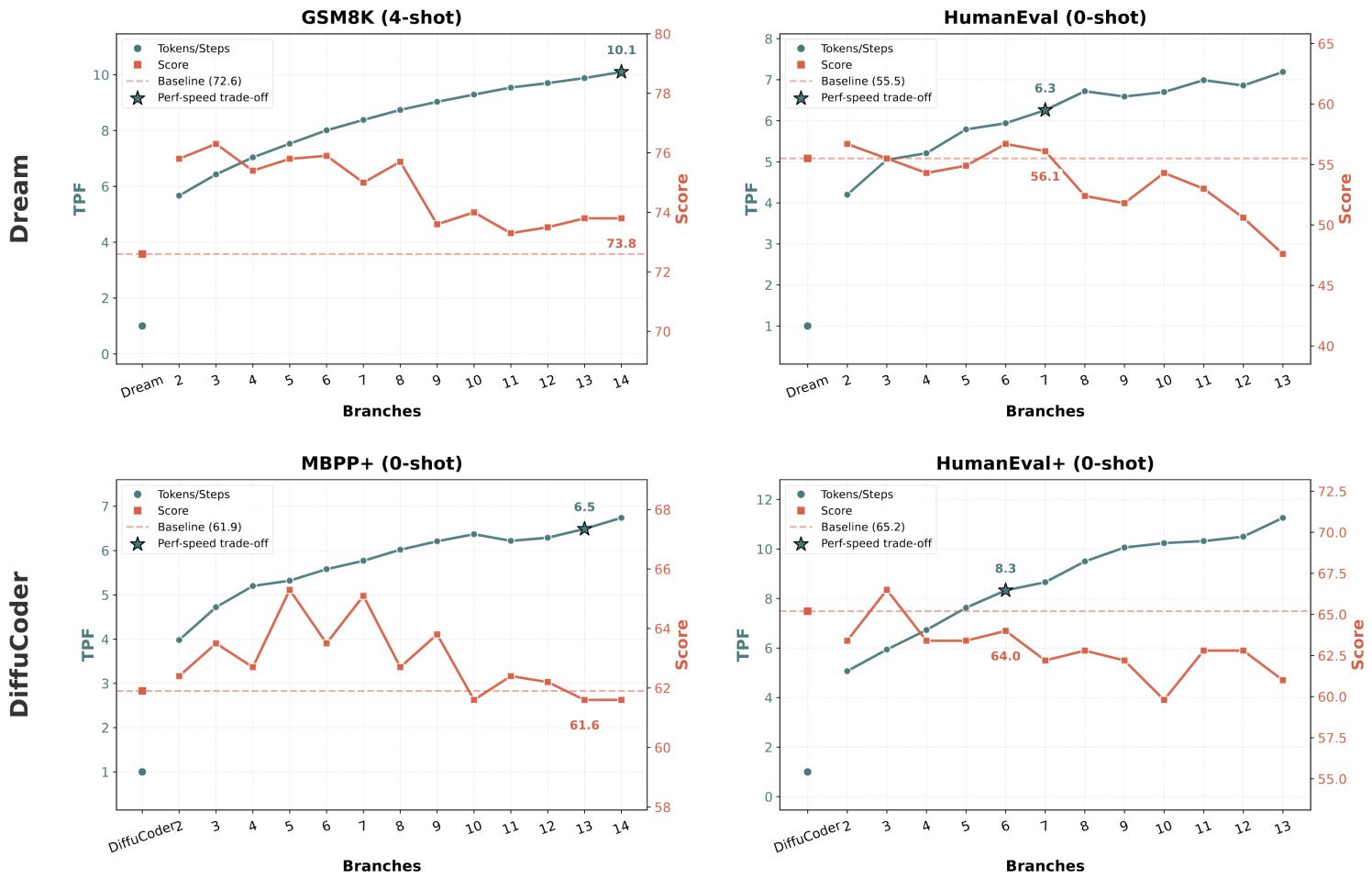
LoPA：提升不同基座上的有效提交量

Model	Decoding Algo.	GSM8K 4-shot			Math 4-shot			HumanEval 0-shot			MBPP 3-shot		
		Score	TPF	AUP	Score	TPF	AUP	Score	TPF	AUP	Score	TPF	AUP
Dream-based models													
Dream	Vanilla	72.6	1.0	72.6	33.7	1.0	33.7	**57.9**	1.0	57.9	56.2	1.0	56.2
	KLASS	74.2	2.4	157.2	36.7	2.3	71.9	56.7	2.5	141.3	**56.4**	1.7	95.4
	Few-step Decoding	65.8	2.0	121.8	25.8	2.0	51.1	30.5	2.0	90.5	42.2	2.0	93.9
	Conf-thres.	72.6	2.4	153.7	**37.6**	2.2	71.1	**57.9**	1.8	104.2	55.6	2.5	137.9
	Conf-thres. + LoPA	73.3	3.9	243.7	37.0	3.7	114.6	56.1	3.7	205.1	54.8	4.3	231.0
dParallel-Dream	dParallel	77.0	3.8	255.5	37.0	3.3	102.7	56.1	2.8	156.0	55.0	3.3	178.9
D2F-Dream	D2F	**78.5**	3.4	236.3	36.8	2.7	84.0	56.1	1.9	107.0	53.8	2.2	117.7
	D2F + LoPA	73.8	**10.7**	**652.7**	35.2	**8.3**	**229.9**	56.1	**6.1**	**335.9**	56.0	**5.5**	**304.7**
LLaDA-based models													
LLaDA	Vanilla	**77.6**	1.0	77.6	33.0	1.0	33.0	37.2	1.0	37.2	**39.8**	1.0	39.8
	KLASS	74.3	2.6	192.0	31.6	2.0	63.1	39.6	2.1	71.2	**39.8**	2.1	83.6
	Few-step Decoding	73.4	2.0	147.6	29.2	2.0	59.6	33.5	2.0	60.2	32.4	2.0	69.0
	Conf-thres.	76.4	3.0	228.1	**33.1**	2.3	75.8	**41.5**	3.0	106.0	39.5	2.7	106.5
	Conf-thres. + LoPA	75.8	5.9	440.9	31.4	4.3	131.4	39.6	4.6	148.4	38.4	5.2	195.9
dParallel-LLaDA	dParallel	76.4	7.0	529.2	31.3	4.7	142.7	39.2	5.7	179.3	39.0	4.0	154.6
D2F-LLaDA	D2F	77.5	3.1	240.1	28.3	2.7	76.4	36.6	3.1	92.8	36.6	2.8	101.5
	D2F + LoPA	76.6	**7.2**	**546.6**	29.5	**7.3**	**203.1**	40.2	**9.2**	**299.0**	38.4	**7.2**	**270.3**

D2F-Dream / GSM8K：10.7 TPF；D2F-LLaDA / HumanEval：9.2 TPF

LoPA, supplementary experiments

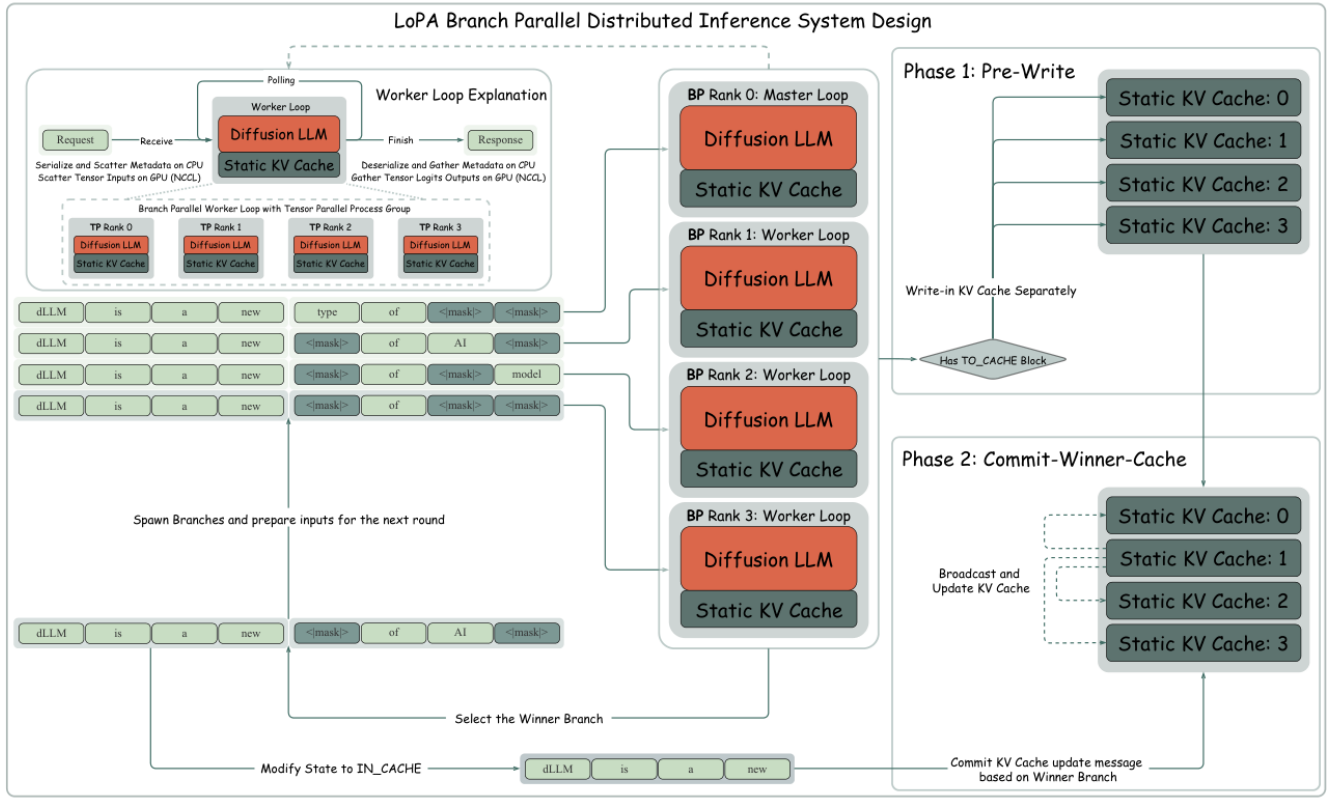
LoPA：增加分支数，扩展可选择的运行点



更多分支提高 **TPF**，也改变质量与计算成本的折衷。

LoPA, branch-budget scaling

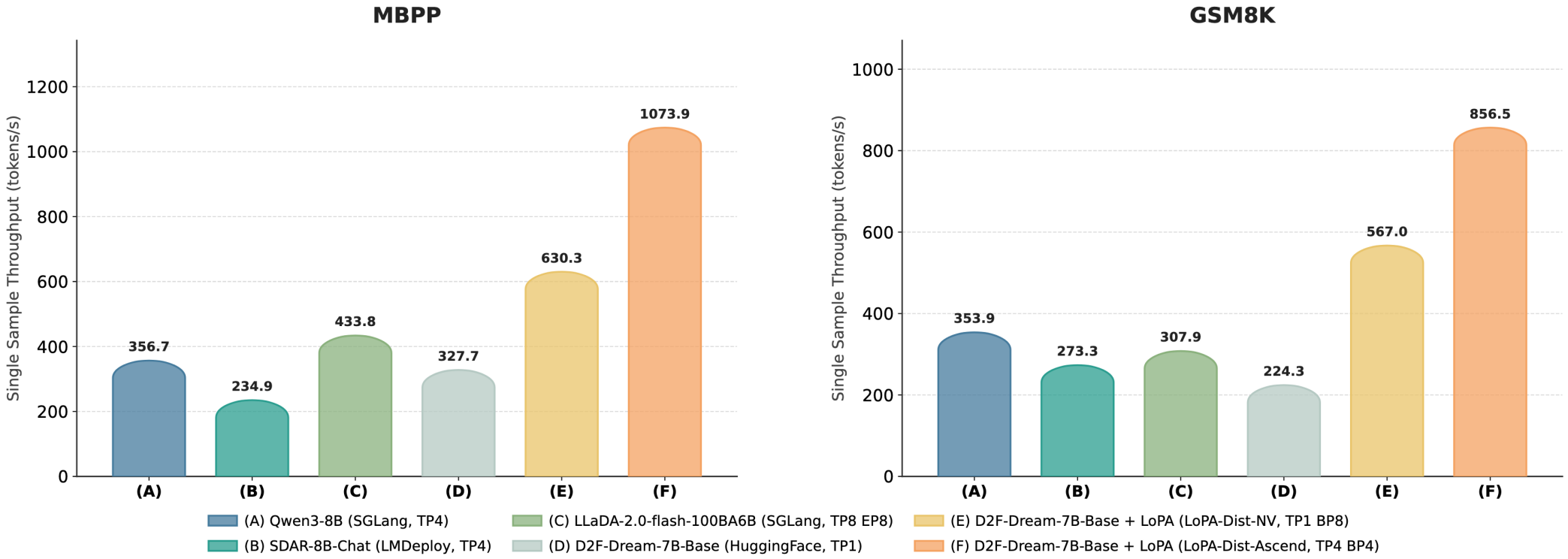
LoPA-Dist：并行执行候选分支



分发候选、并行验证，再同步下一轮状态与缓存。

arXiv:2512.16229v2

LoPA-Dist：单请求生成速度超过 1k tokens/s



开源系统实测：MBPP 达到 1073.9 tokens/s。

LoPA-Dist, system measurements

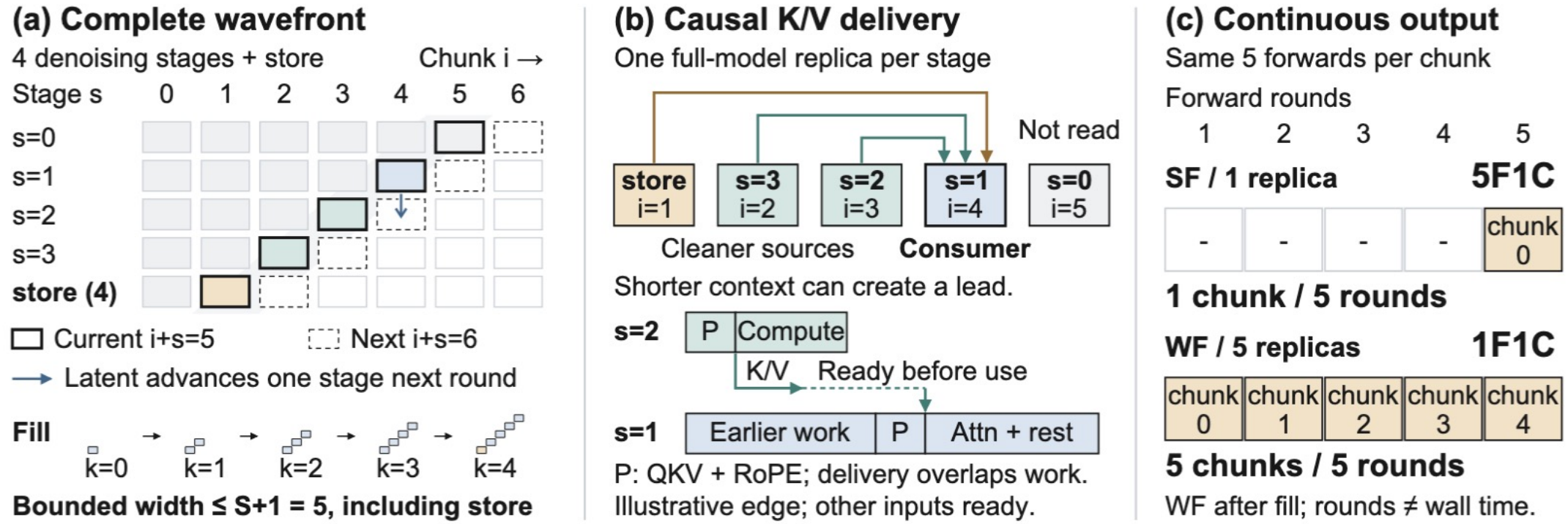
报告提纲

- 基础原理与模型进展
- 模型训练与生成质量
- 推理算法与系统优化
- 应用方向与未来展望

从文本生成，走向多模态与交互应用

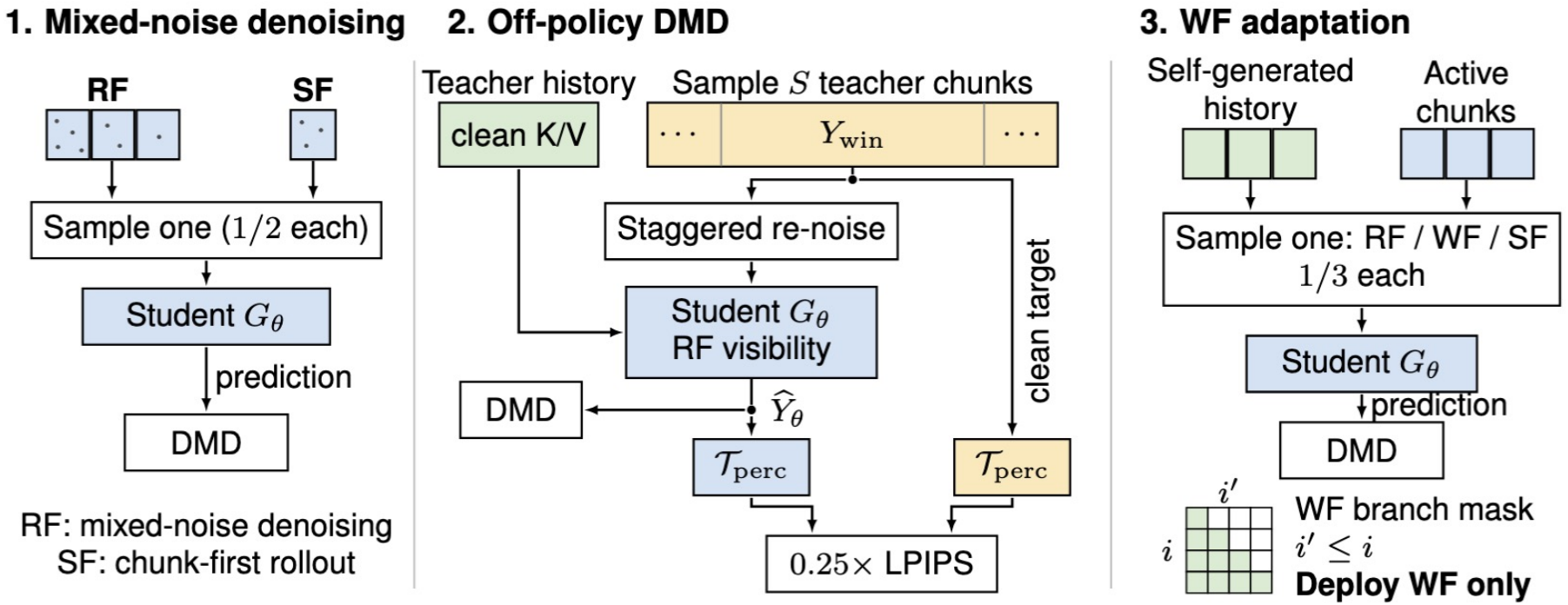
流式视频	多块流水线，让输出持续推进 WaveForcing
语音修复	结合双向上下文与文本语言先验 TIMUSE
多模态与 **Agent**	在新模态和外部反馈中迭代生成 LLaDA-o · LLaDA 2.2

WaveForcing：把多块流水线迁移到视频生成



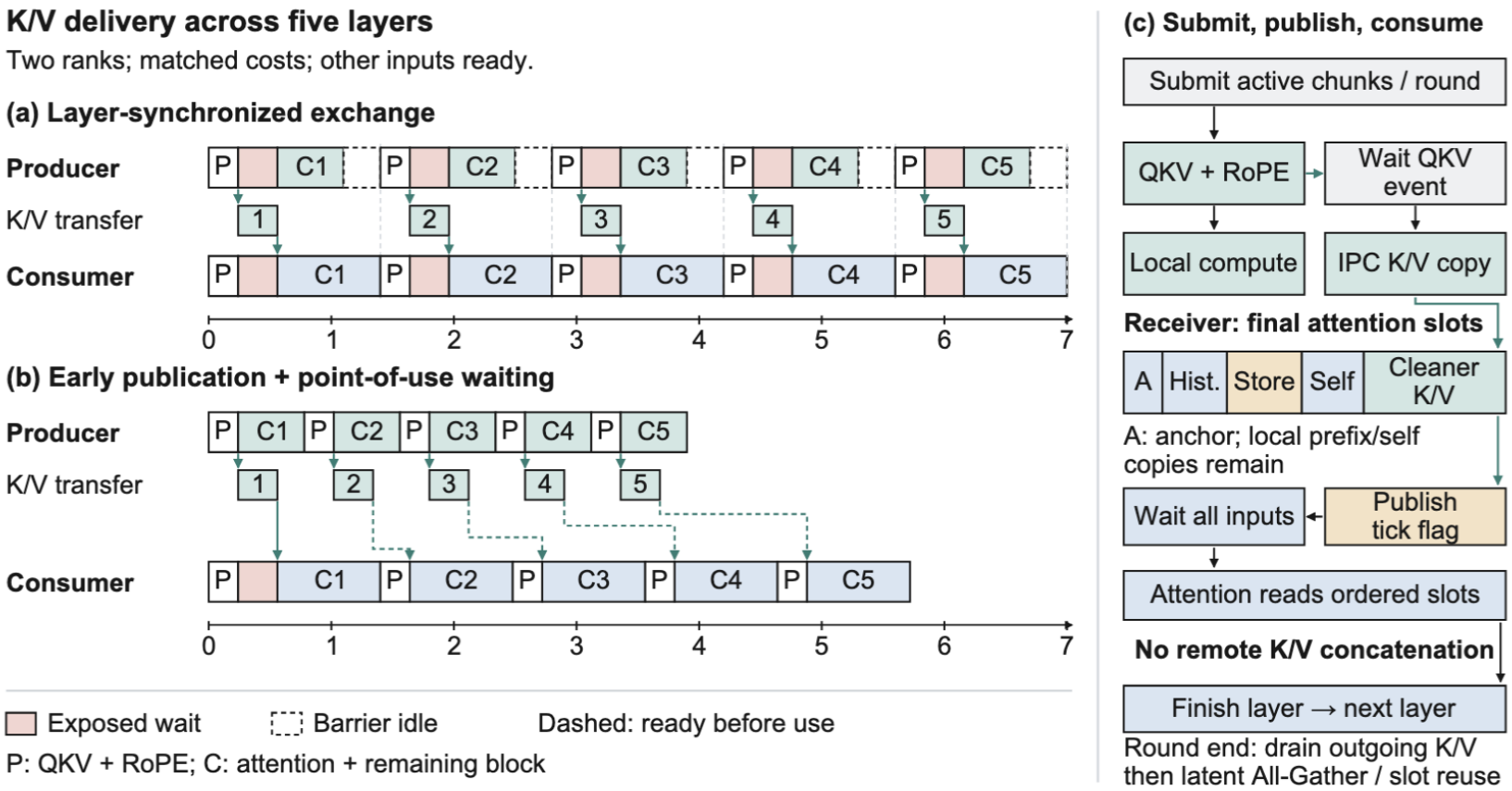
每块仍多步去噪；流水线填满后，每轮持续输出一块。

WaveForcing：三阶段训练适配流水线生成



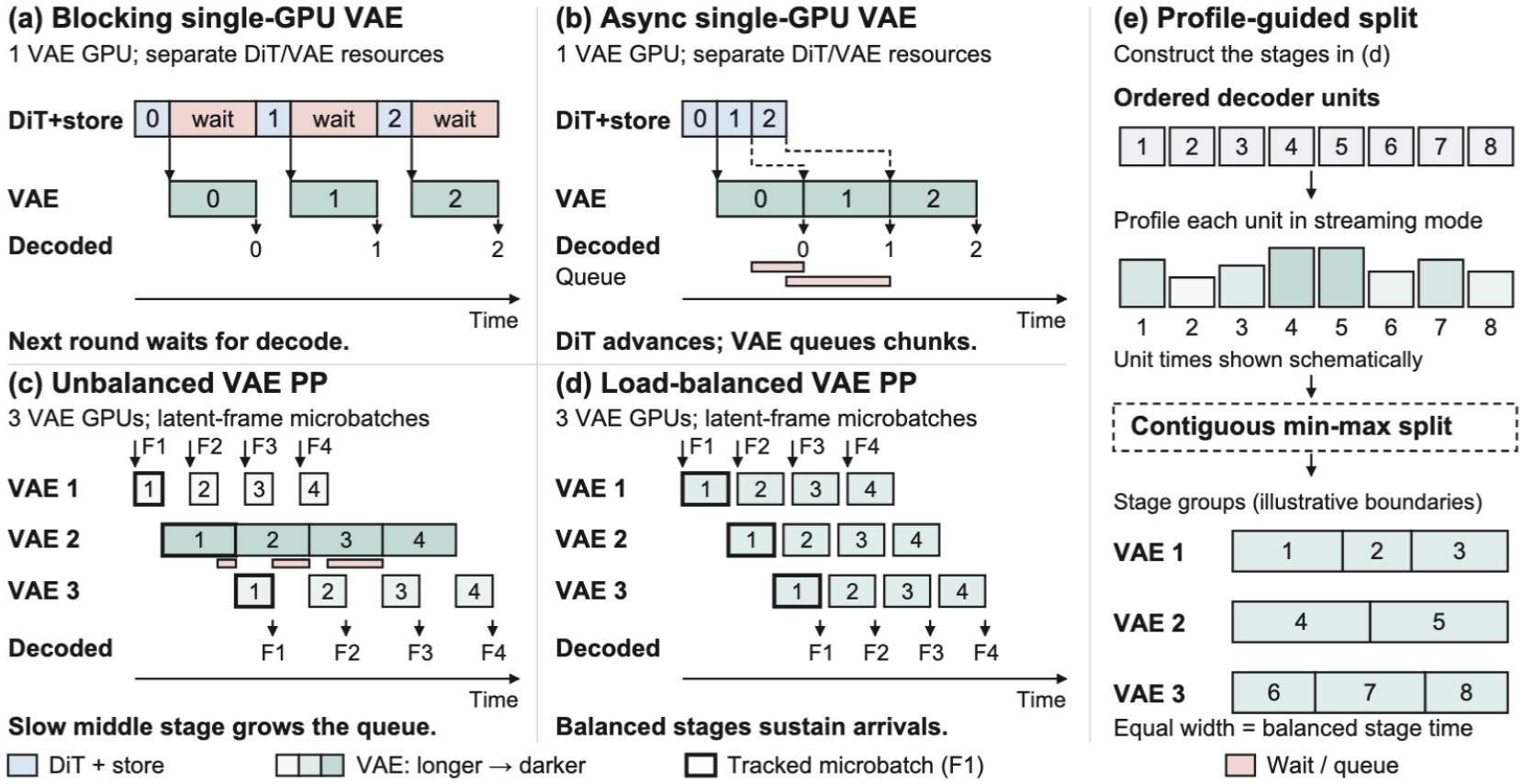
混合噪声去噪　　强化感知监督　　适配 WF 计算图

WaveForcing：利用计算间隙，重叠 KV 通信



较短上下文带来计算间隙，提前传递 **KV** 以隐藏通信。

WaveForcing：让 VAE 解码跟上 DiT 的输出



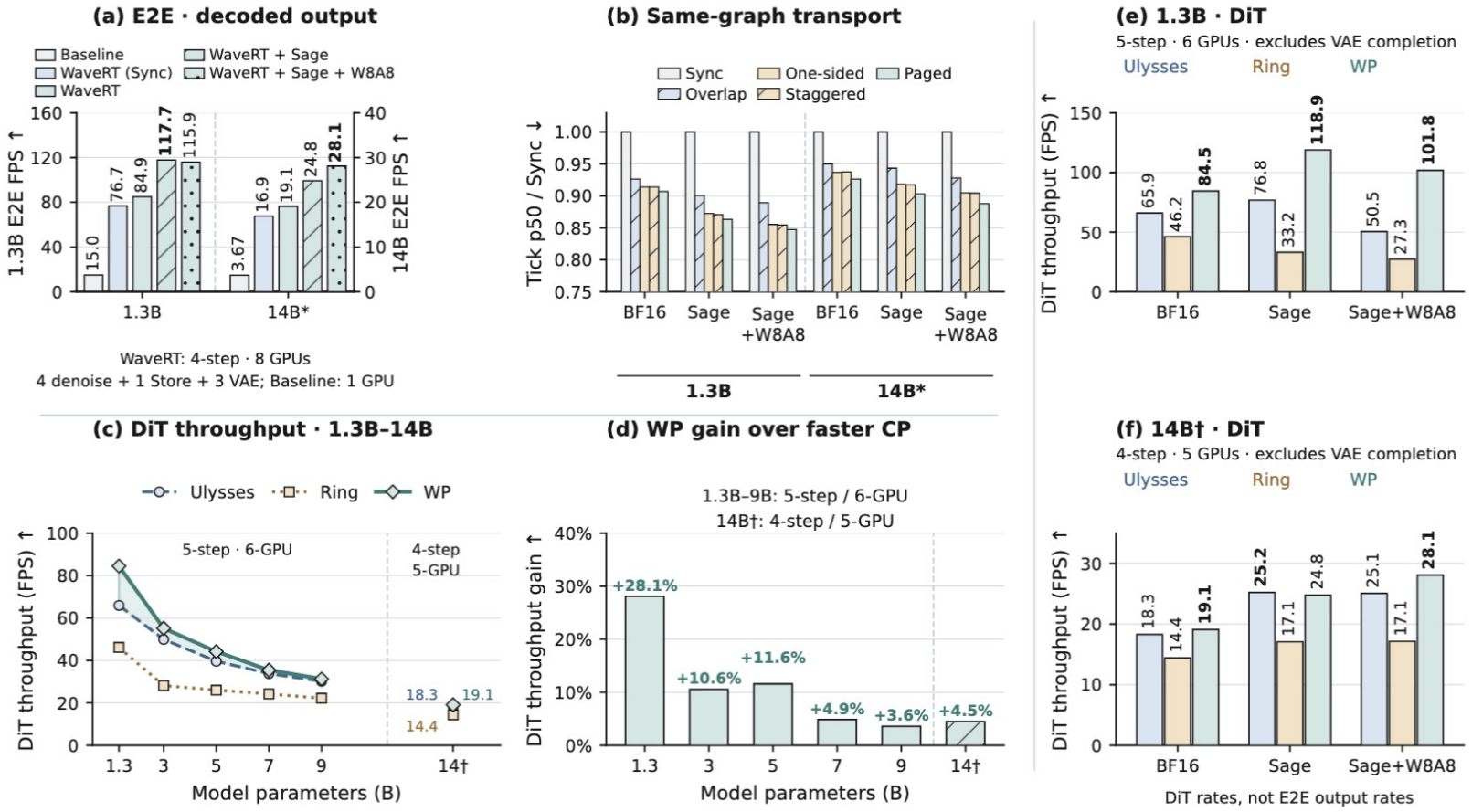
异步解码之后，按层负载均衡，降低最慢阶段的限制。

WaveForcing：兼顾视频质量与持续输出

Method	Steps	GPUs	Standard VBench ↑				VBench-Long ↑				
			Total	Quality	Semantic	Dynamic	Quality	Dynamic	Subj. ↑	Drift ↓	E2E FPS ↑
Rolling Forcing	5	1	81.15	83.77	70.66	51.39	81.18	36.63	92.75	**0.17**	11.90
+ Causal Attention	5	8	80.89	83.35	**71.05**	37.50	80.50	46.60	91.26	0.40	82.71
Causal Forcing (Full Attn.)	4	1	79.61	82.83	66.72	**100.00**	**82.15**	**96.61**	71.11	11.14	15.00
Wave Forcing (Ours)	4	8	80.74	83.93	67.95	84.72	80.89	74.97	91.88	0.36	84.90
+ SageAttention		8	81.03	84.02	69.07	65.28	81.16	77.24	90.79	0.36	**117.70**
Wave Forcing (Ours)	5	8	**81.64**	**84.40**	70.59	94.44	81.26	91.45	**97.10**	0.37	83.70
+ SageAttention		8	81.02	83.92	69.41	80.56	81.14	91.14	94.43	0.37	84.98

兼顾生成质量、动态度与长视频稳定性。

WaveForcing：系统优化进一步提高吞吐



最新优化

150 FPS

WaveForcing：连续生成中的运动与稳定性



观察连续帧：运动是否自然，主体与镜头是否稳定。

WaveForcing, qualitative comparison

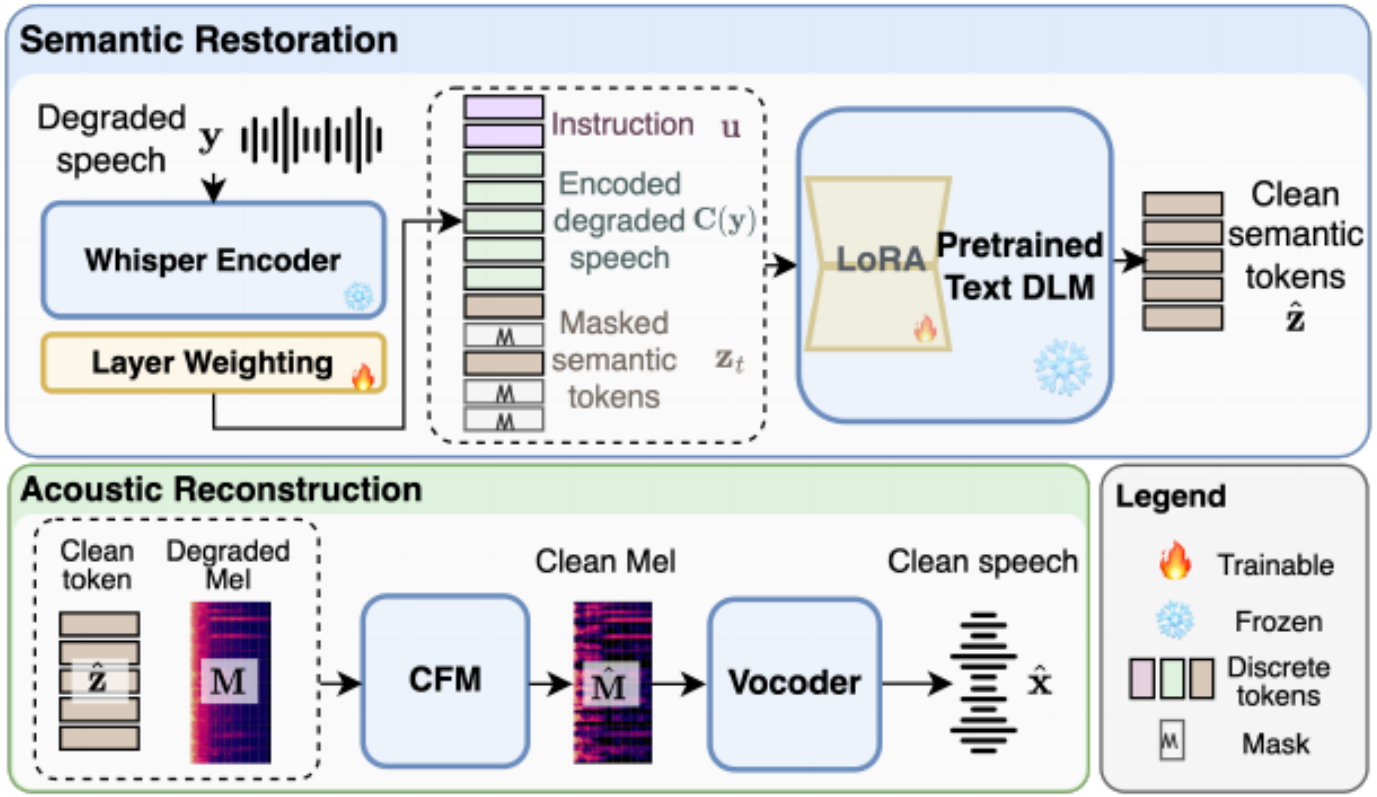
语音修复：声音自然，还要内容准确

噪声、带宽限制与丢包，会同时破坏声音和语言证据。

双向上下文　　结合整句前后文，恢复受损位置的语义

文本语言先验　　声学证据不足时，利用文本预训练知识

TIMUSE：先恢复语义，再重建声学细节



语义恢复

Whisper 编码
文本 dLLM + LoRA
生成干净语义 token

声学重建

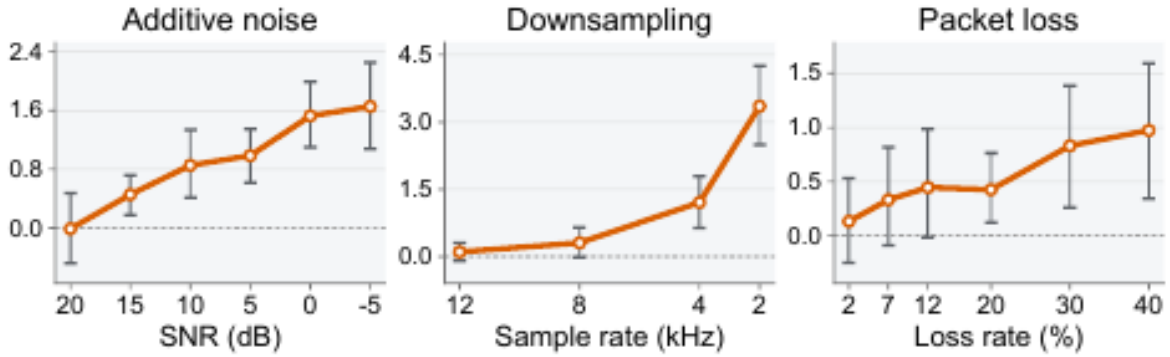
语义 token + 退化 Mel
CFM + Vocoder

Dream-7B；语义生成 2 steps；CFM 8 steps；输出 24 kHz

TIMUSE：掩码生成与文本先验分别带来收益

语义模型	CAcc ↑	DNSOVRL ↑	SpkSim ↑
AR，无文本预训练	72.84%	3.194	0.611
MGM，无文本预训练	78.83%	3.228	0.642
MGM，文本初始化	**80.61%**	**3.225**	**0.640**

文本初始化的 CER 降幅（pp，95% CI）



MGM 相对 AR

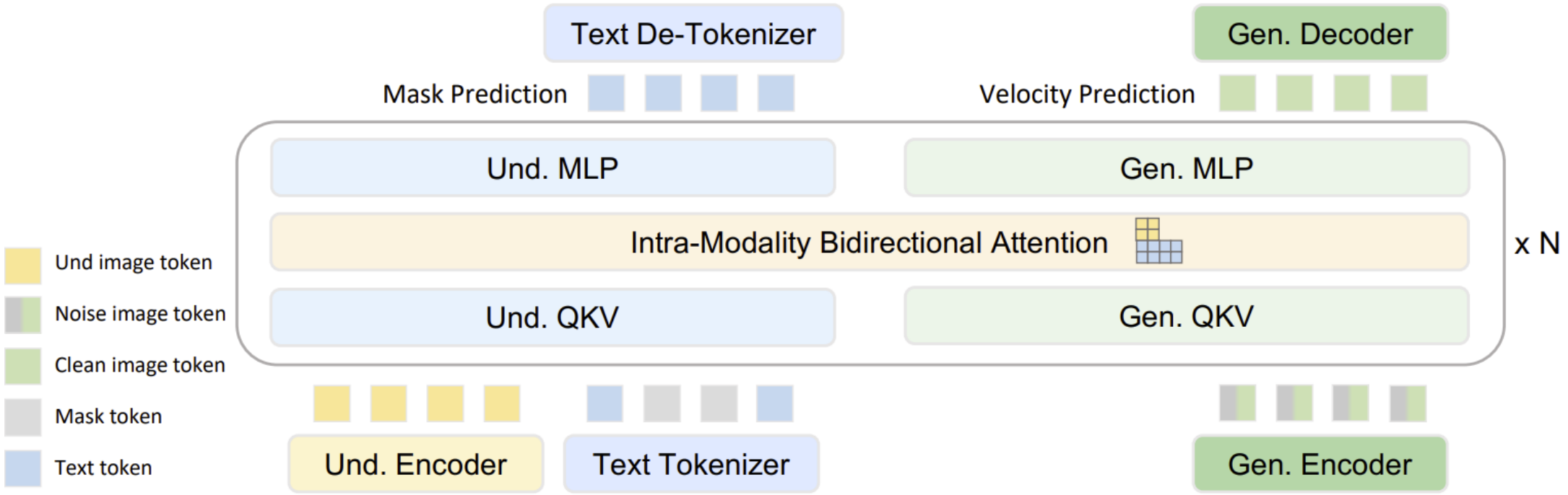
+5.99 pp

文本初始化

+1.78 pp

内容更准确，
音质与音色基本保持

多模态LLaDA-o：从文本扩散走向统一生成



共享主干，协调不同模态的表示与生成过程。

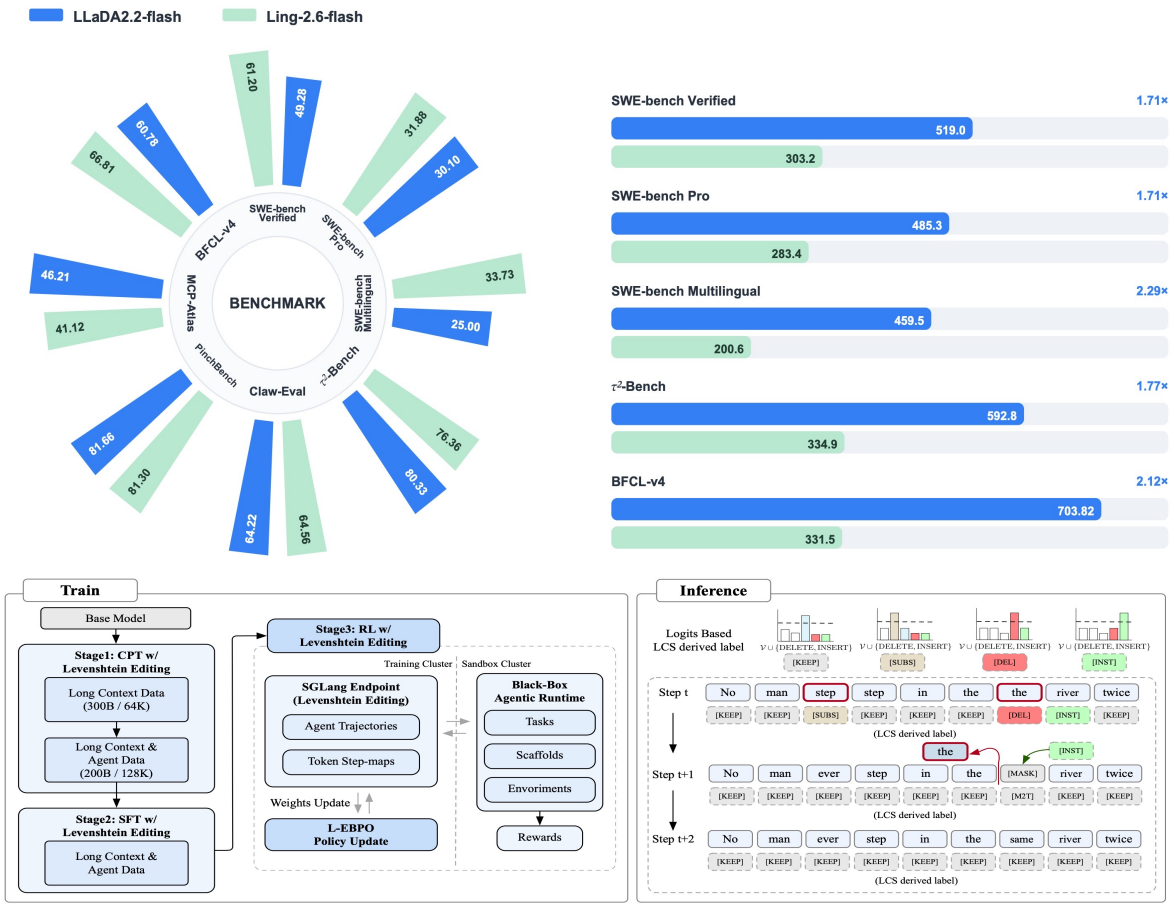
arXiv:2603.01068

Agent：生成、工具调用与迭代编辑

生成初稿　形成计划或工具调用

接收反馈　工具结果补充新的上下文

局部编辑　删除、插入与重新生成



代表方向：**LLaDA 2.2 的 DELETE / INSERT 编辑。**

开源社区

D2F

Diffulex

github.com/SJTU-DENG-Lab

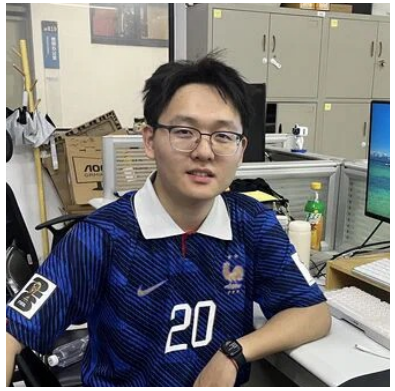
欢迎交流，也欢迎加入我们

合作者



徐晨开

硕士生



金义杰

博士生



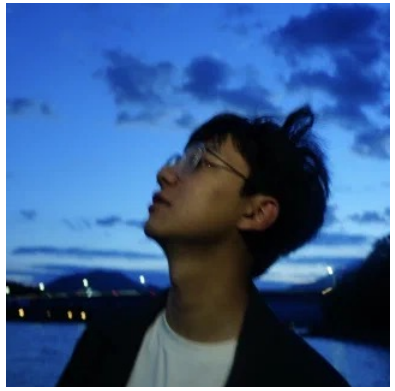
任玉矶

硕士生



胡彦哲

本科生



刘宇轩

本科生

谢谢！欢迎交流