d3LLM: Ultra-Fast *Diffusion LLM* and Beyond

Presented by **Yu-Yang Qian**

School of Artificial Intelligence, Nanjing University, China

https://www.lamda.nju.edu.cn/qianyy/

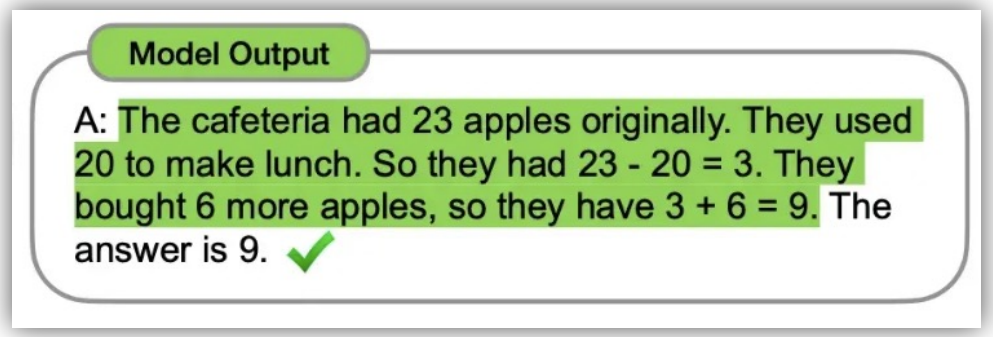
2026.07.29

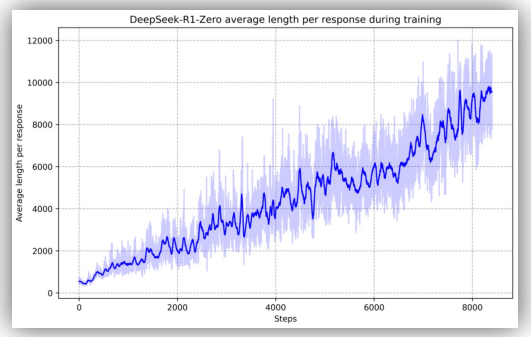
Background

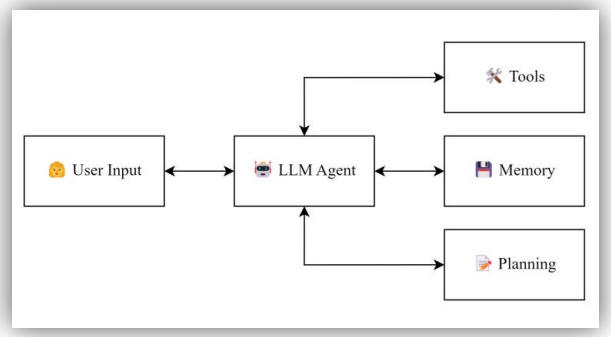
- LLMs have achieved remarkable success across diverse tasks,
  - e.g., ***chatbots***, ***vibe coding***, and long-term ***agentic tasks***.

Background

- LLMs have achieved remarkable success across diverse tasks,
  - e.g., ***chatbots***, ***vibe coding***, and long-term ***agentic tasks***.
- Not only the model size, but also the ***inference cost***, is increasing.
  - Especially with ***Chain of Thought*** (COT)
  - Recently emerging ***agentic LLM*** further increase this burden

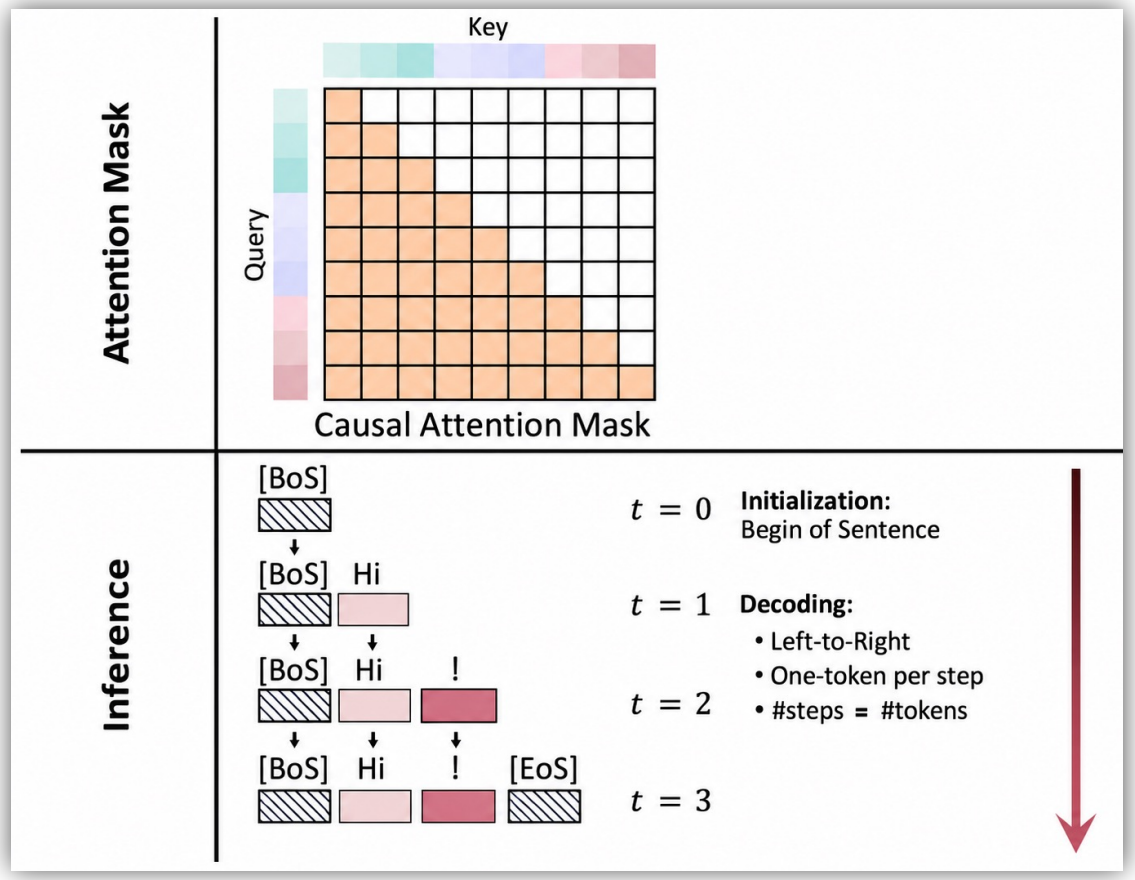






However, autoregressive (AR) paradigm incurs substantial ***inference latency***.

Background: Latency Bottleneck of AR

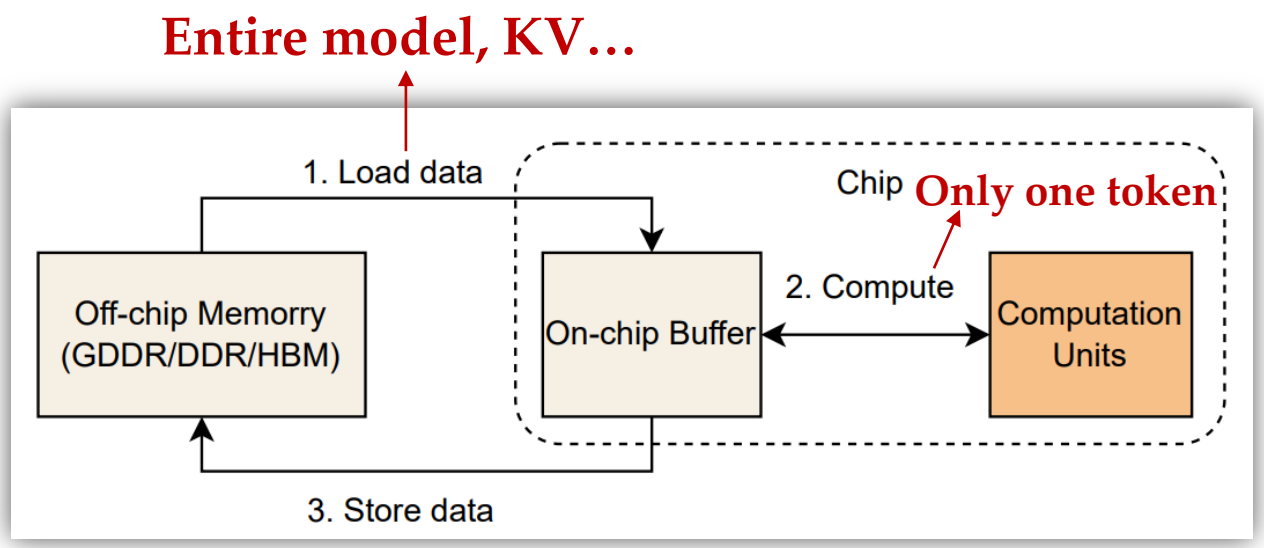


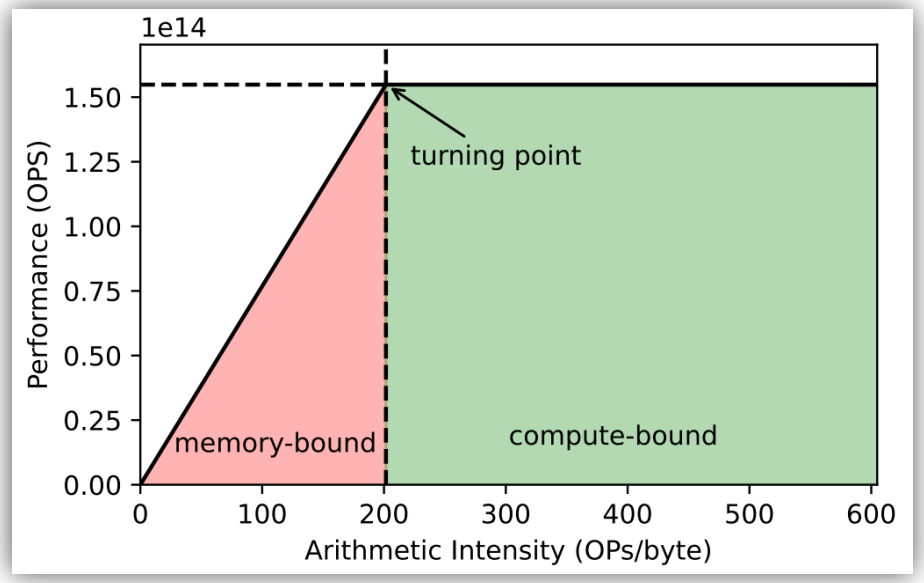
A token sequence probability
$p(x) = \prod_i p(x_i|x_{<i})$

- **Causal attention:** the model predicts the next token.
- It forms the foundation of today's mainstream LLMs such as GPT, LLaMA, and Qwen.

Background: Latency Bottleneck of AR

- Autoregressive (AR) LLMs generate one token per forward pass.
- AR's decoding is ***memory-bound***, leaving the GPU ***underutilized***.





AR's decoding is ***memory-bound***, leaving the GPU ***underutilized***.

Background: Latency Bottleneck of AR

- **Slow serial decoding:** A sequence of length L needs at least L forward passes, making it hard to emit multiple tokens in parallel.
- **Left-to-right dependency bias:** Generation is naturally skewed left→right; reversal, bidirectional infilling, and arbitrary in-place edits are unnatural.
- **No easy rollback of errors:** Early mistaken tokens are hard to fix, so later decoding often has to "continue from the mistake".

AR's decoding is ***memory-bound***, leaving the GPU ***underutilized***.

Preliminary: *Diffusion*

- ***Diffusion:*** generate data by progressively **denoising** from noise (Ho et al. 2020).

Inspired by diffusion processes

(Brownian motion)

Forward process adds noise to data

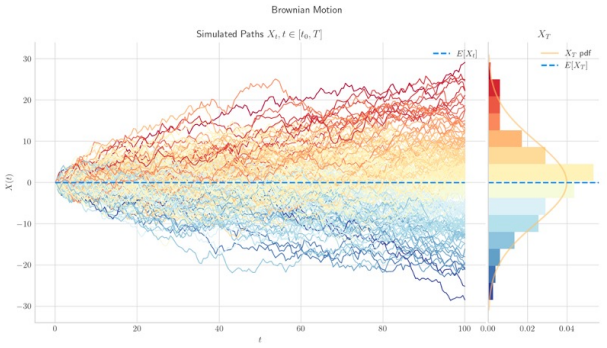
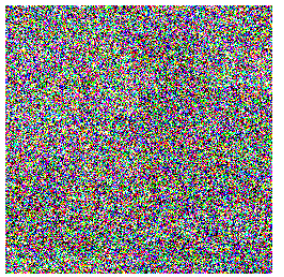
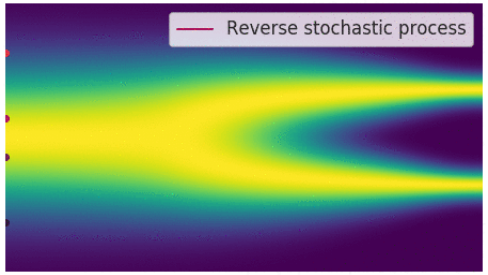


Image credit: https://quantgirluk.github.io/Understanding-Quantitative-Finance/brownian_motion.html





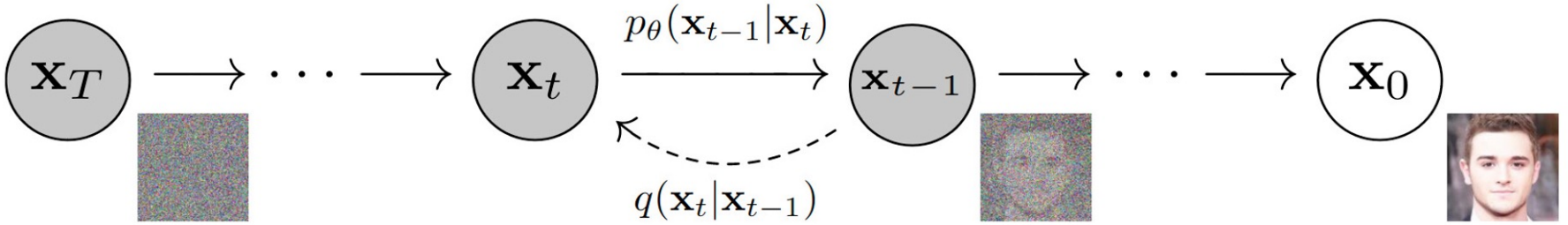
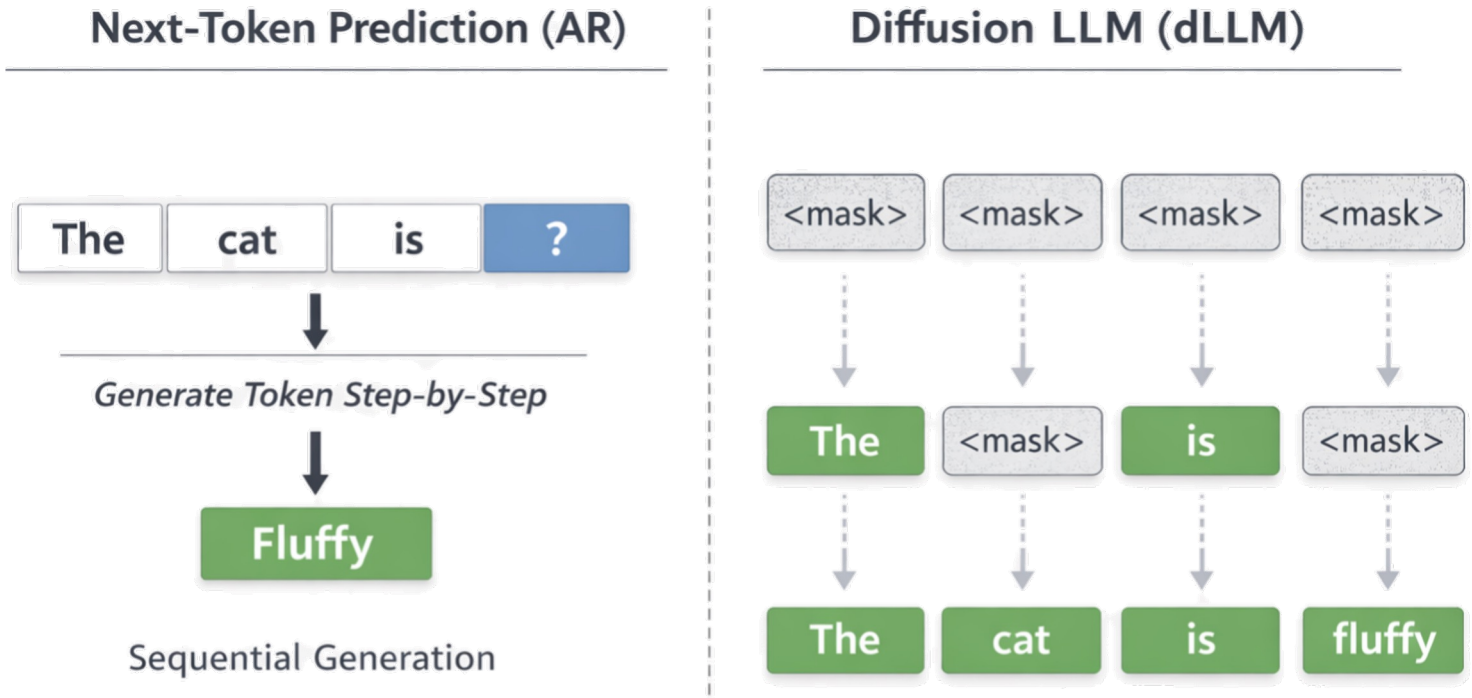


Image credit: Denoising Diffusion Probabilistic Models https://arxiv.org/pdf/2006.11239

To this end: *dLLM* is introduced

- ***dLLMs*** have shown promising capabilities, such as:
  - · Parallel generation
  - · Error correction
  - · Random-order generation



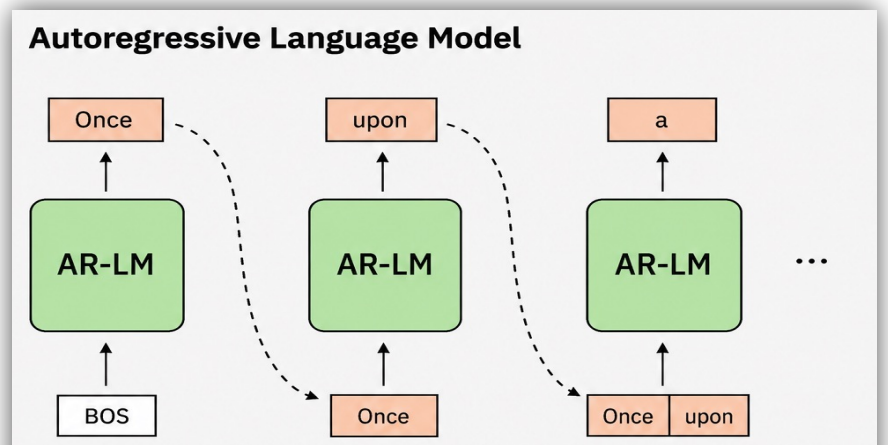
To this end: *dLLM* is introduced

- ***LLaDA*** (Nie et al. 2025): a representative open-source dLLM:

AR LLM:

$$\underbrace{p_\theta(x) = p_\theta\left(x^1\right) \prod_{i=2}^{L} p_\theta\left(x^i \mid x^1, \ldots, x^{i-1}\right)}_{\text{Autoregressive formulation}}$$

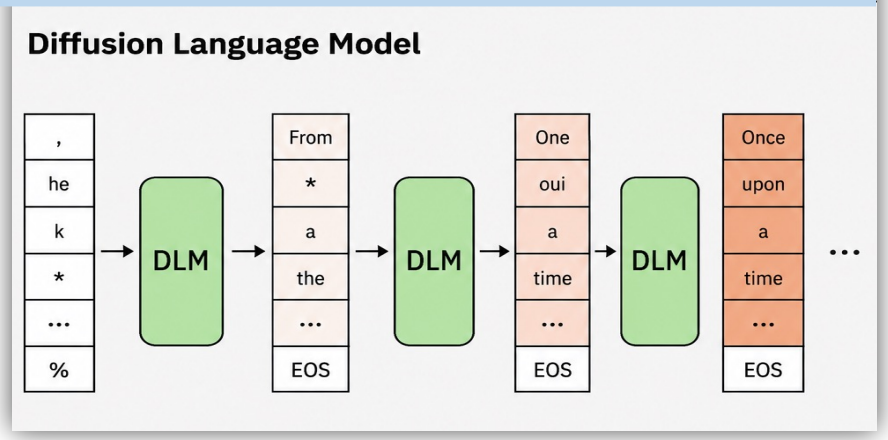
"Next token prediction"



Diffusion LLM:

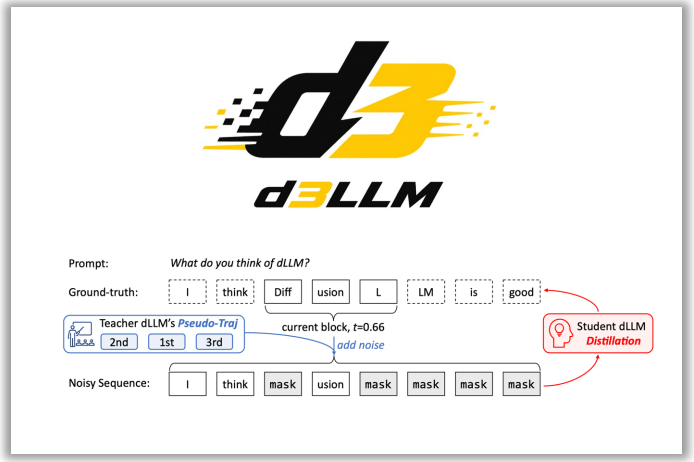
$$\underbrace{p_\theta\left(x^1, x^2, \ldots, x^L\right) = \prod_{i=1}^{L} p_\theta\left(x^i \mid x_{\text{masked}}\right)}_{\text{Diffusion formulation}}$$

"Parallel generation"

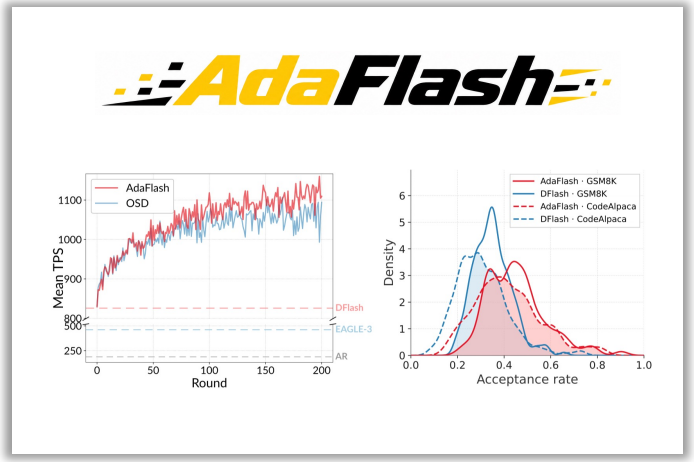


Outline

- Investigating ***diffusion LLM*** (dLLM):



d3LLM: ***accelerating*** dLLM
[ICML'26]

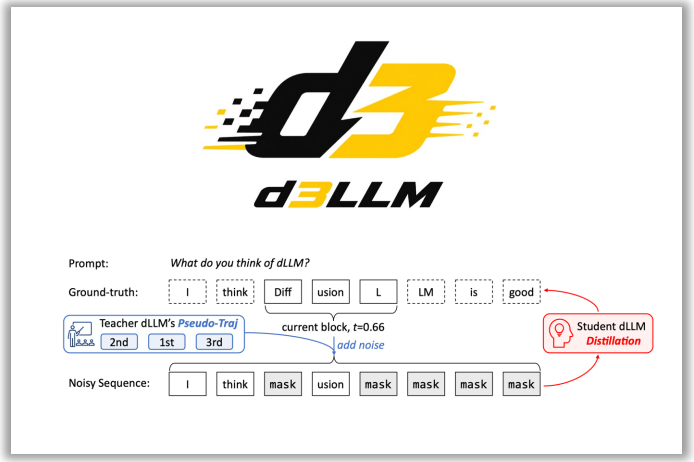


AdaFlash: dLLM in ***speculative decoding***
[arXiv'26]

Investigating ***diffusion LLMs*** toward faster inference and decoding.

Outline: Introduce *d3LLM*

- Investigating ***diffusion LLM*** (dLLM):



d3LLM: ***accelerating*** dLLM
[ICML'26]

AdaFlash: dLLM in *speculative decoding*
[arXiv'26]

Investigating ***diffusion LLMs*** toward faster inference and decoding.

d3LLM: Ultra-Fast Diffusion LLM using Pseudo-Trajectory Distillation

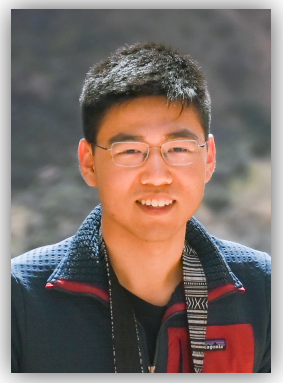
Yu-Yang Qian [1 2] **Junda Su** [1] **Lanxiang Hu** [1] **Peiyuan Zhang** [1] **Zhijie Deng** [4] **Peng Zhao**† [2 3] **Hao Zhang**† [1]

[1] University of California, San Diego [2] School of Artificial Intelligence, Nanjing University
[3] State Key Laboratory for Novel Software Technology, Nanjing University [4] Shanghai Jiao Tong University

† **Correspondence to:** Peng Zhao zhaop@lamda.nju.edu.cn, Hao Zhang haozhang@ucsd.edu

[ICML'26]

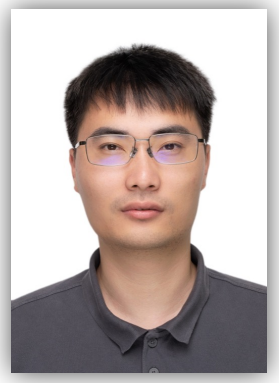

Yu-Yang Qian


Junda Su


Lanxiang Hu


Peiyuan Zhang


Zhijie Deng


Peng Zhao


Hao Zhang

Learning And Mining from DatA

ICML
International Conference On Machine Learning
Seoul, South Korea

Background: ***Diffusion*** *large language models* (**dLLMs**)

- ***dLLMs*** have shown promising capabilities, such as:
  - · Parallel generation
  - · Error correction
  - · Random-order generation

- *Closed-source* dLLMs showed effectiveness:
  - Inception - *Mercury*
  - Google - *Gemini Diffusion*
  - ByteDance - *Seed Diffusion*

Mercury©: 1109 tokens/s

- However, *open-source dLLMs* exhibited **lower speed and accuracy**:
  - · *LLaDA-8B*
  - · *Dream-7B*
  - · *SDAR*

Previous Work: *Improving dLLMs*

- ***Acceleration*** of dLLMs:
  - ***Fast-dLLM:*** efficient *training-free* decoding method;
  - ***dKV-Cache:*** KV-Cache for dLLMs;
  - ***dParallel:*** distilling dLLM to get a faster one;
  - ***D2F***: block-wise causal semi-AR-Diffusion;
  - ***dInfer***: systematic implementation of ***Efficient dLLM***;
  - ***Refusion***: adapt from AR, slot-level parallel decoding.
- Improving the ***performance*** of dLLMs:
  - ***MMaDA***: multimodal diffusion foundation models;
  - ***TraDo:*** dLLMs with *reasoning* ability;
  - ***Fast-dLLM-v2***: *directly post-training from an AR.*

However, previous works typically focus on only ***one-side of the coin***.

Previous Work: *One-side of the Coin*

- However, previous methods use ***single, isolated metrics***:
  - **Efficiency-only metrics:** tokens per second (TPS) or tokens per forward (TPF);
  - **Performance-only metrics:** accuracy (solve rate / pass@1).

- Key insight: a fundamental ***trade-off*** for dLLMs:
  - dLLM naturally lives on an ***accuracy–parallelism curve;***
  - Increasing parallelism almost always reduces accuracy, and vice versa;
  - Therefore, single metrics become misleading.

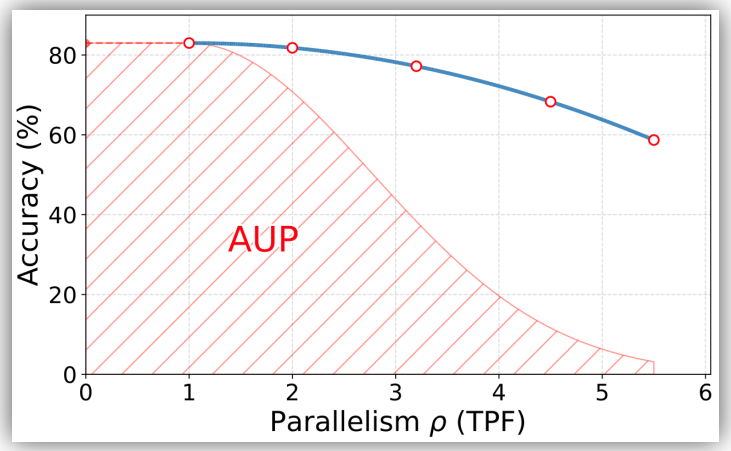
⇨ This motivates us to propose ***a better metric*** for dLLMs.

New Metric: ***AUP***

Collect many parallelism-accuracy pairs $\{(\rho_i, y_i)\}_{i=1}^{m}$:

$$\text{AUP} \triangleq \rho_1 y_1 + \sum_{i=2}^{m} (\rho_i - \rho_{i-1}) \left(\frac{y_i W(y_i) + y_{i-1} W(y_{i-1})}{2} \right)$$

where $W(y) = \min(e^{-\alpha(1-y/y_{\max})}, 1)$ is weight function.



- We introduce ***AUP*** (*Accuracy Under Parallelism*):
  - A weighted area under the accuracy–parallelism curve;
  - ***Jointly*** measures both accuracy and parallelism;
  - Exponentially ***penalizes*** accuracy drops.

Encourage dLLMs to *strike a balance* between accuracy and parallelism.

Approach: d3LLM

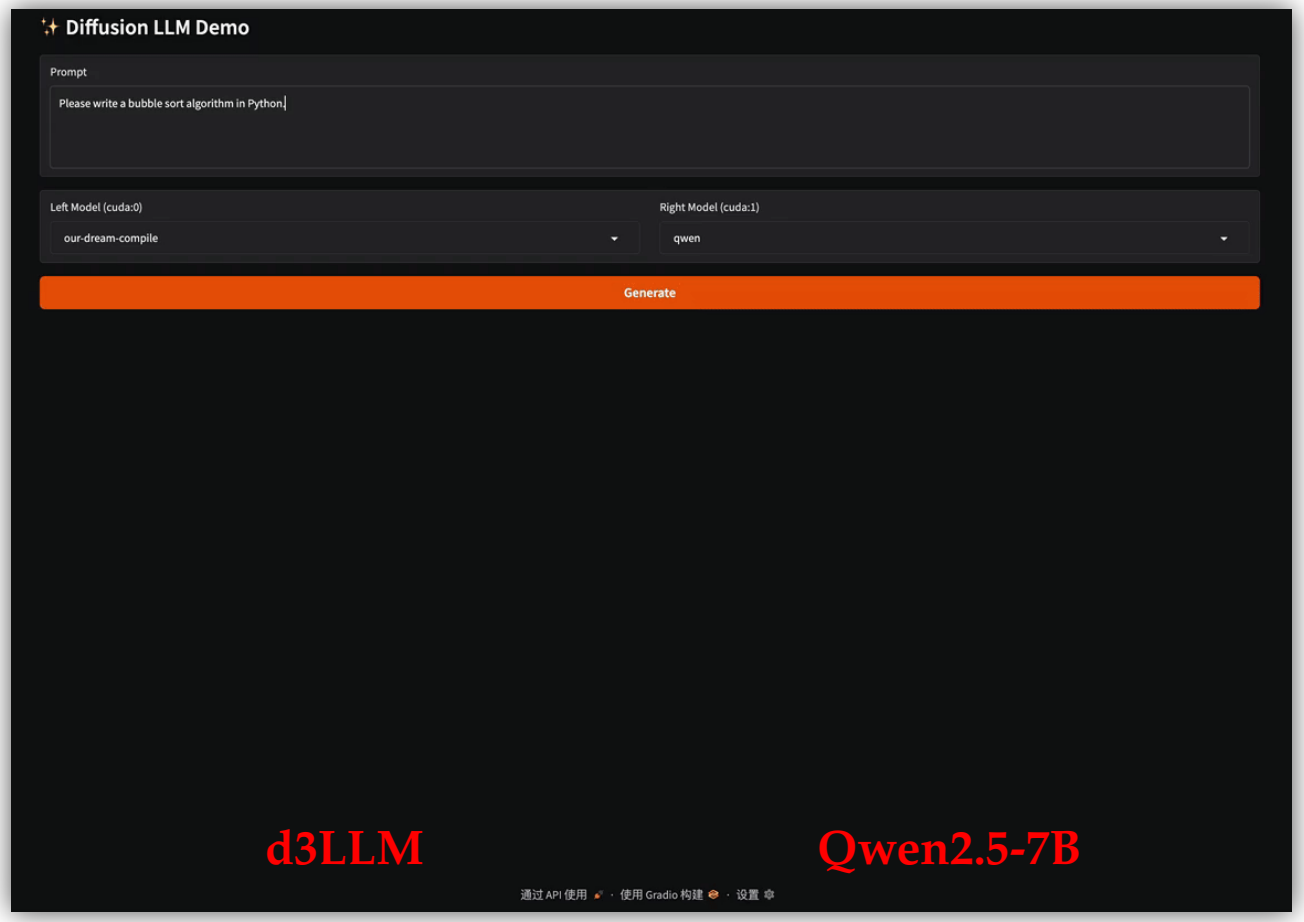
- Guided by **AUP**, we design ***d3LLM*** (*pseuDo-Distilled Diffusion LLM*):

  - A framework that improves both ***training and inference;***

    (i) Pseudo-trajectory distillation ***(training)***

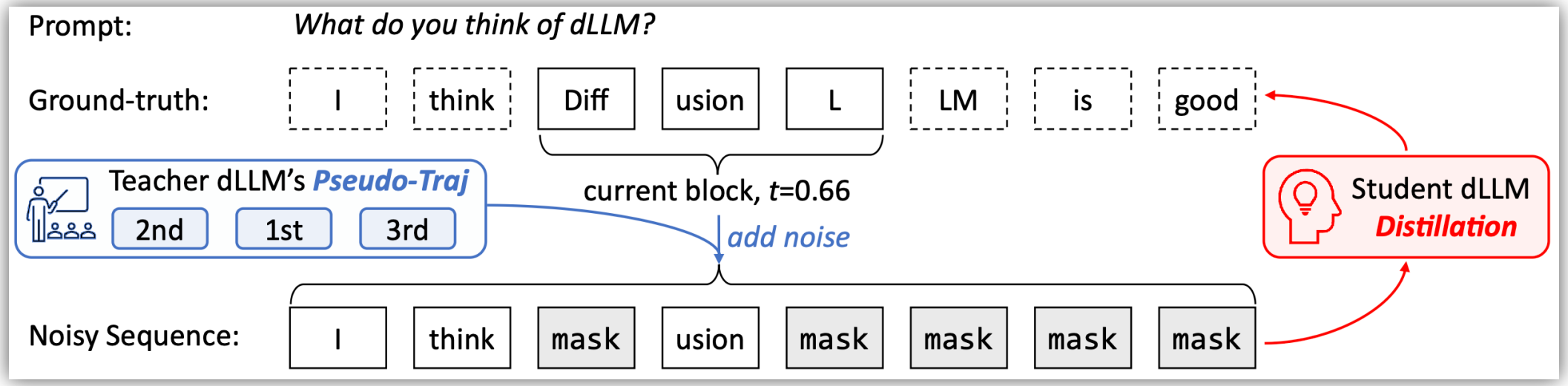
    (ii) Multi-block decoding with KV-refresh ***(inference)***

  - ***Push*** the accuracy–parallelism ***frontier*** of dLLMs.



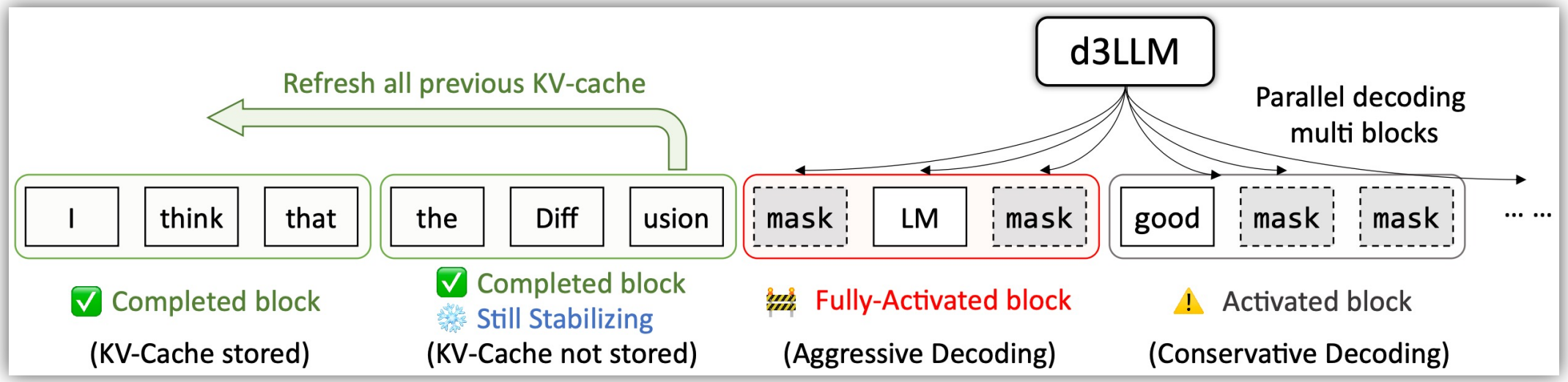
d3LLM: *(i) training recipe*

- Regarding the *(post-)**training*** 🏋️:
  - Utilizing the teacher's ***pseudo-trajectory***: which tokens can be confidently decoded earlier (+18% TPF)
  - ***Curriculum noise level***: gradually increase mask ratio (+12% TPF)
  - ***Curriculum window size***: gradually increase window size (+8% TPF)



d3LLM: *(ii) decoding strategy*

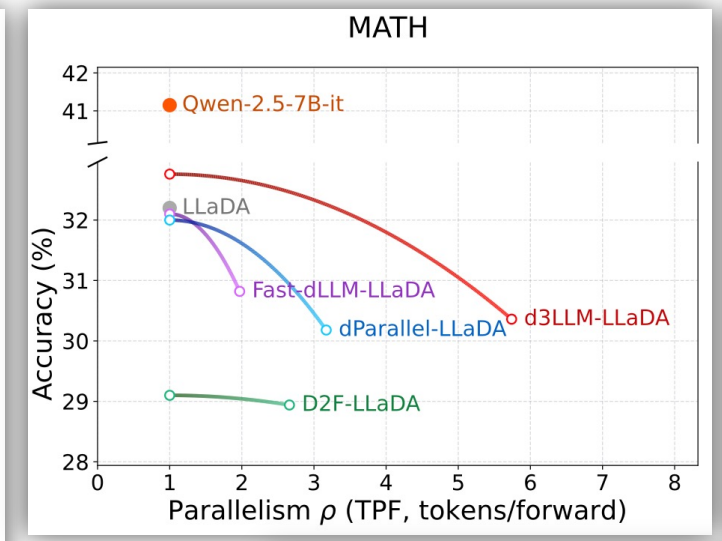
- Regarding the ***decoding*** ⚡:
  - Entropy-Based ***Multi-Block Decoding***: decoding multiple blocks (+30% TPF)
  - KV-Cache with ***Refresh***: refresh KV-caches before the stabilizing (+35% TPS)
  - ***Early Stopping*** on EOS Token: stop when EOS is generated (+5% TPF)

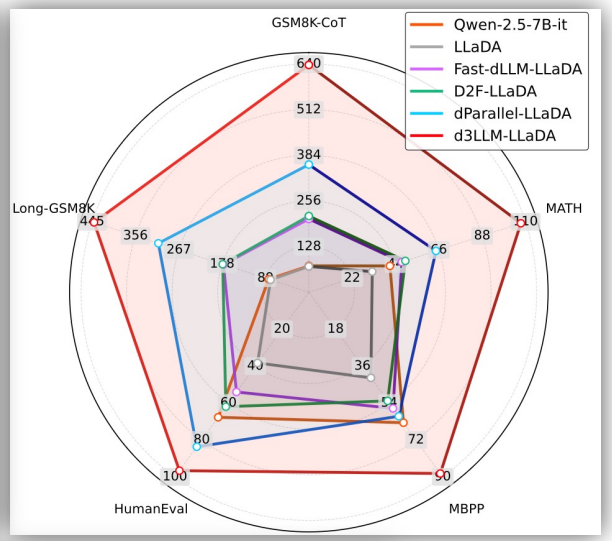


d3LLM: experimental results

- On *LLaDA*, *Dream* and *Dream-Coder*:

Benchmark	Method	TPF ↑	Acc (%) ↑	AUP Score ↑
GSM8K-CoT (0-shot)	LLaDA	1.00 ± 0.0	72.6 ± 0.2	72.6 ± 0.2
	Fast-dLLM-LLaDA	2.77 ± 0.1	74.7 ± 0.2	205.8 ± 6.4
	D2F-LLaDA	2.88 ± 0.1	73.2 ± 0.3	209.7 ± 6.1
	dParallel-LLaDA	5.14 ± 0.1	72.6 ± 0.2	358.1 ± 6.2
	d3LLM-LLaDA	**9.11** ± 0.1	73.1 ± 0.3	**637.7** ± 6.8
MATH (4-shot)	LLaDA	1.00 ± 0.0	32.2 ± 0.4	32.2 ± 0.4
	Fast-dLLM-LLaDA	1.97 ± 0.1	30.8 ± 0.3	47.2 ± 2.9
	D2F-LLaDA	2.38 ± 0.1	28.7 ± 0.2	45.5 ± 2.8
	dParallel-LLaDA	3.17 ± 0.1	30.2 ± 0.2	64.5 ± 3.1
	d3LLM-LLaDA	**5.74** ± 0.1	30.4 ± 0.3	**107.6** ± 3.2
MBPP (3-shot)	LLaDA	1.00 ± 0.0	41.7 ± 0.3	41.7 ± 0.3
	Fast-dLLM-LLaDA	2.13 ± 0.1	38.6 ± 0.3	56.6 ± 3.7
	D2F-LLaDA	1.94 ± 0.1	38.0 ± 0.2	50.0 ± 3.6
	dParallel-LLaDA	2.35 ± 0.1	40.0 ± 0.3	60.5 ± 3.9
	d3LLM-LLaDA	**4.21** ± 0.1	40.6 ± 0.2	**88.4** ± 4.0
HumanEval (0-shot)	LLaDA	1.00 ± 0.0	38.3 ± 0.5	38.3 ± 0.5
	Fast-dLLM-LLaDA	2.56 ± 0.1	37.8 ± 0.4	54.0 ± 2.9
	D2F-LLaDA	2.69 ± 0.1	36.6 ± 0.5	62.0 ± 2.7
	dParallel-LLaDA	4.93 ± 0.2	39.0 ± 0.4	83.7 ± 4.8
	d3LLM-LLaDA	**5.95** ± 0.1	39.6 ± 0.6	**96.6** ± 3.2
Long-GSM8K (5-shot)	LLaDA	1.00 ± 0.0	78.6 ± 0.2	78.6 ± 0.2
	Fast-dLLM-LLaDA	2.45 ± 0.1	78.0 ± 0.3	175.4 ± 6.4
	D2F-LLaDA	2.70 ± 0.1	73.7 ± 0.2	168.5 ± 6.0
	dParallel-LLaDA	4.49 ± 0.1	76.7 ± 0.3	309.1 ± 6.2
	d3LLM-LLaDA	**6.95** ± 0.1	74.2 ± 0.3	**441.1** ± 6.5





Method	H100 TPS ↑	A100 TPS ↑	Acc (%) ↑
Qwen-2.5-7B-it	57.3 (1.0×)	50.4 (1.0×)	74.1
LLaDA	27.9 (0.5×)	19.2 (0.4×)	72.6
Fast-dLLM-LLaDA	114.3 (2.0×)	79.1 (1.6×)	74.7
D2F-LLaDA	102.1 (1.8×)	76.2 (1.5×)	73.2
dParallel-LLaDA	172.2 (3.0×)	105.9 (2.1×)	72.6
d3LLM-LLaDA	**288.9 (5.0×)**	**183.3 (3.6×)**	73.1

🚀 **3.6×–5×** speedup over AR model (Qwen-2.5-7B-it), **10×** speedup compared to vanilla LLaDA/Dream, with *negligible acc degradation*.

Ablation study of d3LLM

Table 5. Ablation study on different distillation and decoding strategies of our method. We report the TPF, Accuracy, and AUP score of our *d3LLM-LLaDA* on GSM8K-CoT dataset (0-shot).

Distillation Recipe			Decoding Method		GSM8K-CoT (0-shot)		
Pseudo-trajectory	Curriculum Noise	Curriculum Window	Multi-block Decoding	Early Stop	TPF ↑	Acc (%) ↑	AUP Score ↑
			✓	✓	6.41 ± 0.1	72.2 ± 0.3	441.4 ± 3.2
✓			✓	✓	7.55 ± 0.1	72.1 ± 0.2	517.7 ± 3.9
✓	✓		✓	✓	8.46 ± 0.2	69.8 ± 0.4	551.3 ± 7.8
✓	✓	✓	✓	✓	**9.11 ± 0.1**	**73.1 ± 0.3**	**637.7 ± 6.8**
✓	✓	✓			7.01 ± 0.1	73.2 ± 0.1	492.9 ± 4.3
✓	✓	✓	✓		9.07 ± 0.1	73.1 ± 0.3	635.0 ± 6.7
✓	✓	✓	✓	✓	**9.11 ± 0.1**	**73.1 ± 0.3**	**637.7 ± 6.8**

Table 6. Hyperparameter analysis of *Curriculum Noise Level*. We report the TPF, Accuracy, and AUP score on GSM8K-CoT dataset of *d3LLM-LLaDA* model.

Noise	TPF ↑	Acc (%) ↑	AUP Score ↑
Fixed (t=0.5)	7.49 ± 0.1	72.8 ± 0.5	521.7 ± 5.4
Curriculum 0.2 → 0.5	8.03 ± 0.2	72.7 ± 0.5	557.7 ± 5.8
Curriculum 0.0 → 0.5	8.85 ± 0.1	72.9 ± 0.5	616.9 ± 6.5
Curriculum 0.0 → 0.8	**9.11 ± 0.1**	**73.1 ± 0.3**	**637.7 ± 6.8**

Table 7. Hyperparameter analysis of *Curriculum Window Size*. We report the TPF, Accuracy, and AUP score on GSM8K-CoT dataset of *d3LLM-LLaDA* model.

Size	TPF ↑	Acc (%) ↑	AUP Score ↑
Fixed (k=32)	8.22 ± 0.2	69.8 ± 0.5	536.0 ± 7.8
Curriculum 0 → 32	8.67 ± 0.2	72.8 ± 0.3	603.1 ± 9.1
Curriculum 16 → 32	**9.11 ± 0.1**	**73.1 ± 0.3**	**637.7 ± 6.8**
Curriculum 24 → 32	8.58 ± 0.2	71.9 ± 0.4	584.9 ± 8.9

Each component contributes. Combining all achieves the best performance.

Incorporating *SGLang* engine

- Incorporating d3LLM models into ***SGLang engine*** (PR #20615) 🏎️:

			B200	H800/H100	A800/A100	Result		
	阈值	batch size	TPS	TPS	TPS	TPF	Acc	对齐HF
Qwen2.5-7B-Instruct	/	1	274.7	108.6	96.8	1	74.1	✅
Qwen3-8B	/	1	234.2	98.3	90	1	93.63	/
d3LLM-LLaDA (8B dense)	0.5	1	**1240.99**	**545.31**	**251.61**	**9.91**	75.36	✅
	0.5	4	**1310.18**	**551.87**	249.98	**8.56**	75.12	/
d3LLM-Dream (7B dense)	0.4	1	586.77	280.48	125.57	4.89	80.89	✅
	0.4	4	676.81	281.82	127.85	4.22	80.76	/
LLaDA 2.0 mini (16B A1B)	0.95	1	401.27	241.94	179.86	2.42	92.49	✅
	0.95	4	507.22	275.25	218.57	2.25	92.95	/
LLaDA 2.1 mini (16B A1B)	0.95	1	520.81	270.24	247.66	3.04	92.65	/
	0.95	4	516.89	390.04	**310.55**	2.49	92.87	/

🚀 **3×–4.5×** speedup over AR model (Qwen) with SGLang engine.

dLLM Leaderboard

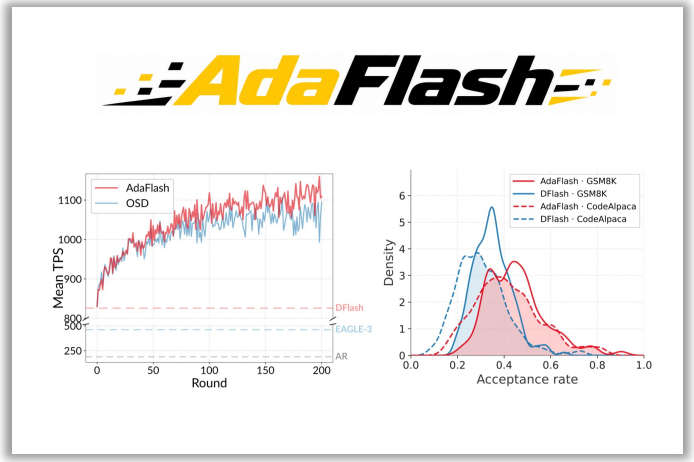
- dLLM Leaderboard - Hugging Face Space

Rank	Method	Type	Foundation Model	GSM8K-CoT	MATH	MBPP	HumanEval	Long-GSM8K	Avg AUP
1	EAGLE-3	AR	Llama-3.1-8B-it	319.0 TPF:5.12 Acc:76.6	142.1 TPF:5.72 Acc:39.8	298.6 TPF:5.69 Acc:60.2	344.8 TPF:5.98 Acc:67.6	422.2 TPF:5.57 Acc:80.5	305.3
2	d3LLM-LLaDA	dLLM	LLaDA-8B-it	637.7 TPF:9.11 Acc:73.1	107.6 TPF:5.74 Acc:30.4	88.4 TPF:4.21 Acc:40.6	96.6 TPF:5.95 Acc:39.6	441.1 TPF:6.95 Acc:74.2	274.3
3	d3LLM-Dream	dLLM	Dream-v0-it-7B	391.3 TPF:4.94 Acc:81.4	97.5 TPF:3.92 Acc:38.2	141.4 TPF:2.96 Acc:55.6	129.5 TPF:3.20 Acc:57.1	348.6 TPF:4.80 Acc:77.2	221.7
4	dParallel-LLaDA	dLLM	LLaDA-8B-it	358.1 TPF:5.14 Acc:72.6	64.5 TPF:3.17 Acc:30.2	60.5 TPF:2.35 Acc:40.0	83.7 TPF:4.93 Acc:39.0	309.1 TPF:4.49 Acc:76.7	175.2
5	dParallel-Dream	dLLM	Dream-v0-it-7B	245.7 TPF:3.02 Acc:82.1	77.9 TPF:2.94 Acc:38.7	108.0 TPF:2.24 Acc:55.4	98.8 TPF:2.57 Acc:54.3	262.4 TPF:3.49 Acc:78.6	158.6
6	Fast-dLLM-v2	dLLM	Qwen-2.5-7B-it	176.1 TPF:2.21 Acc:81.5	126.7 TPF:2.61 Acc:48.7	114.1 TPF:2.04 Acc:59.1	128.9 TPF:2.58 Acc:61.7	207.2 TPF:2.58 Acc:81.0	150.6
7	D2F-LLaDA	dLLM	LLaDA-8B-it	213.8 TPF:2.88 Acc:74.4	49.0 TPF:2.66 Acc:28.9	53.0 TPF:2.13 Acc:39.0	62.0 TPF:2.69 Acc:40.6	176.9 TPF:2.70 Acc:75.7	110.9
8	Fast-dLLM-LLaDA	dLLM	LLaDA-8B-it	205.8 TPF:2.77 Acc:74.7	47.2 TPF:1.97 Acc:30.8	56.6 TPF:2.13 Acc:38.6	54.0 TPF:2.56 Acc:37.8	175.4 TPF:2.45 Acc:78.0	107.8

Outline: Introduce *AdaFlash*

- Investigating ***diffusion LLM*** (dLLM):

d3LLM: *accelerating* dLLM
[ICML'26]



AdaFlash: dLLM in ***speculative decoding***
[arXiv'26]

Leveraging ***diffusion LLMs*** toward faster inference / decoding.

AdaFlash: Adaptive Speculative Decoding via On-Policy Distilled Diffusion Drafters

Yu-Yang Qian[1,2,*] **Hao-Cong Wu**[1,2,*] **Chen Chen**[3] **Jiacheng Sun**[3] **Zhenhua Dong**[3]
Peng Zhao[1,2,†] **Zhi-Hua Zhou**[1,2]
[1]State Key Laboratory for Novel Software Technology, Nanjing University
[2]School of Artificial Intelligence, Nanjing University [3]Huawei Foundation Model Dept

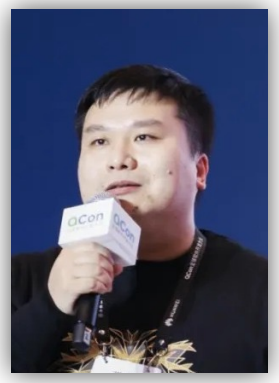
arXiv:2607.19223 **2026, Jul 21**

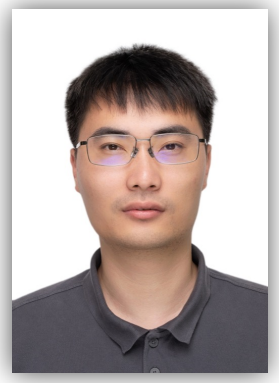

Yu-Yang Qian*


Hao-Cong Wu*


Chen Chen


Jiacheng Sun


Zhenhua Dong

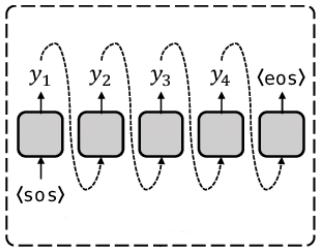

Peng Zhao

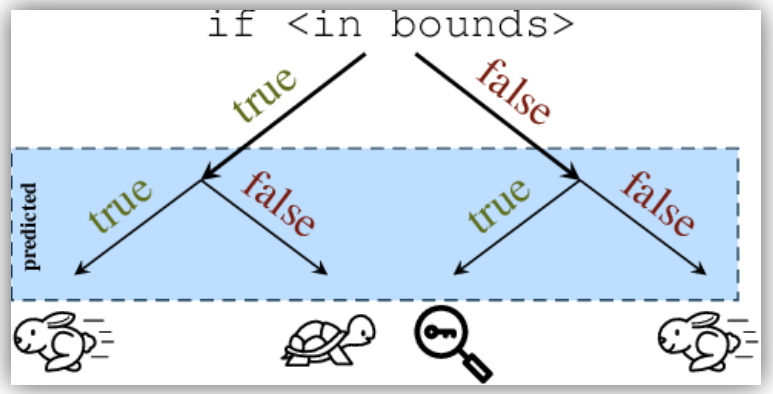

Zhi-Hua Zhou

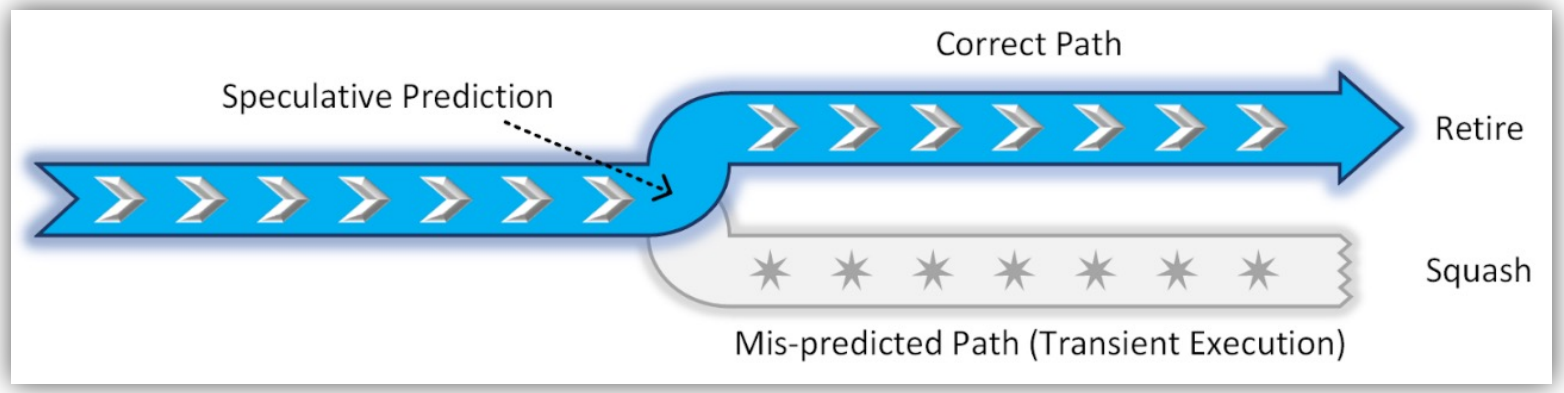
Preliminary: Speculative Execution

AdaFlash

- Autoregressive (AR) models speedup bottleneck
  - ***Sequential*** dependency
- Insight: commonly used in modern ***CPU Processors:***
  - Speculative (投机) execution



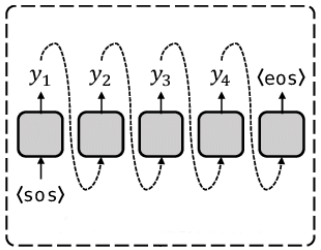




Speculative execution: 提前 **"猜测"** 执行后续指令：若猜对 则可以加速，猜错 则回滚状态。

Preliminary: Speculative Decoding

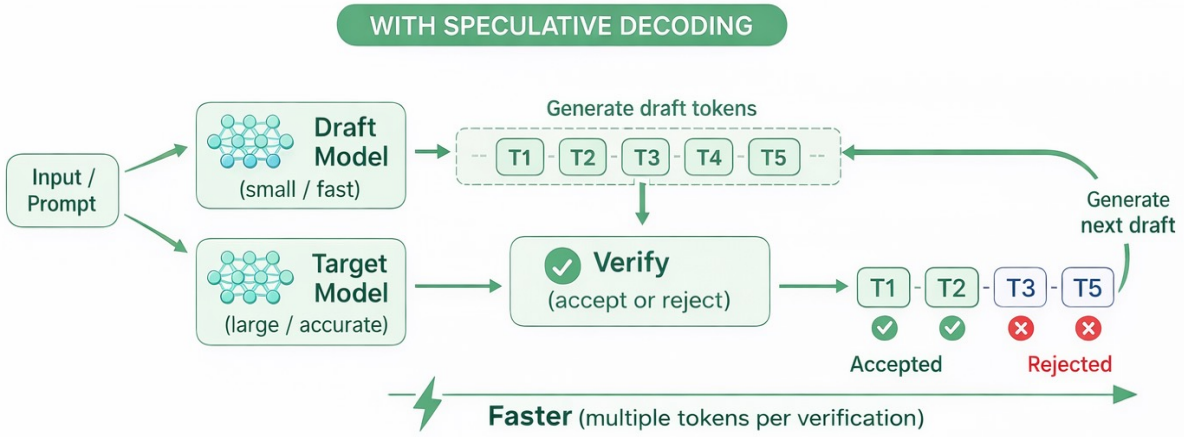
- Autoregressive (AR) models speedup bottleneck
  - *Sequential* dependency

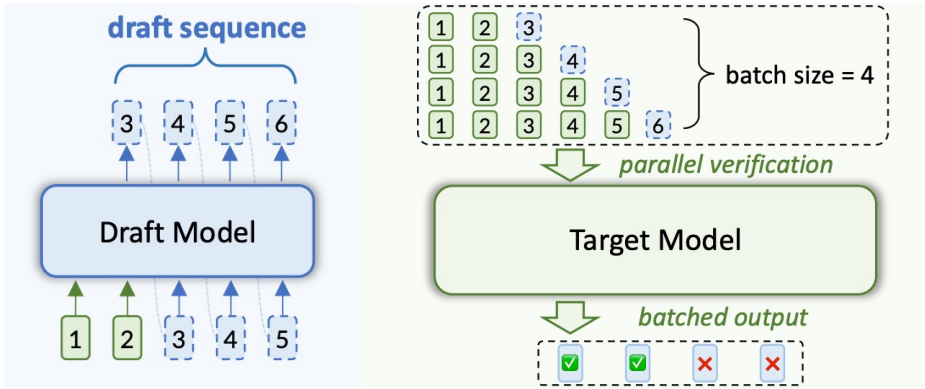


- **Speculative Decoding:** An LLM inference acceleration method.

  Use a small *"draft model"* to first quickly generate tokens as drafts;

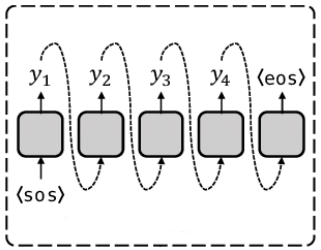
  Use a big *"target model"* to ***parallelly*** verify the drafts.





Preliminary: Speculative Decoding

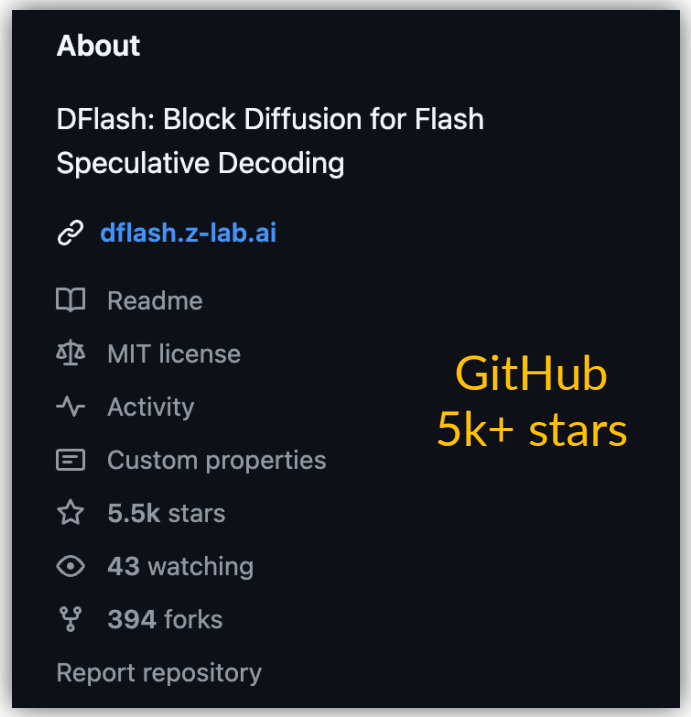
- Autoregressive (AR) models speedup bottleneck
  - *Sequential* dependency



- **Speculative Decoding:** An LLM inference acceleration method.

Current SOTA Method: DFlash

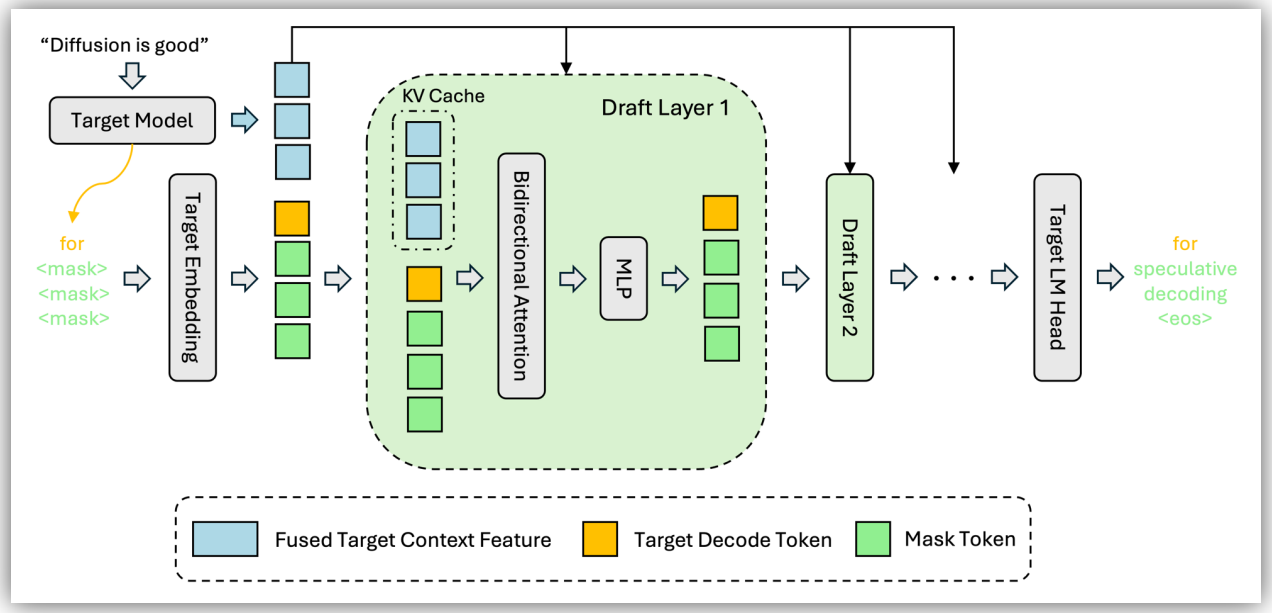
- SOTA of Speculative decoding: **5-6x** speedup compared to AR:



- DFlash is adopted by Xiaomi Mimo, DeepSeek, etc.

Current SOTA Method: DFlash

- DFlash's Key idea: employ diffusion LLM as drafter
  - **Parallel drafting:** generate a block of tokens in one forward pass;
  - **Feature fusion:** hidden states are compressed into one ***fused context feature;***
  - ***KV injection:*** injecting fused context feature as diffusion model's each layer's input.



Model architecture of DFlash

Drawbacks of Diffusion Drafters

AdaFlash

- We uncover a ***central pitfall*** of diffusion drafters:

Bidirectional attention in diffusion drafters is a ***double-edged sword.***

dLLM's bi-directional attn introduces ***high variance*** in draft sequences.

✅ Pros of ***diffusion drafter:***

- Global contextual modeling
- One-pass parallel generation

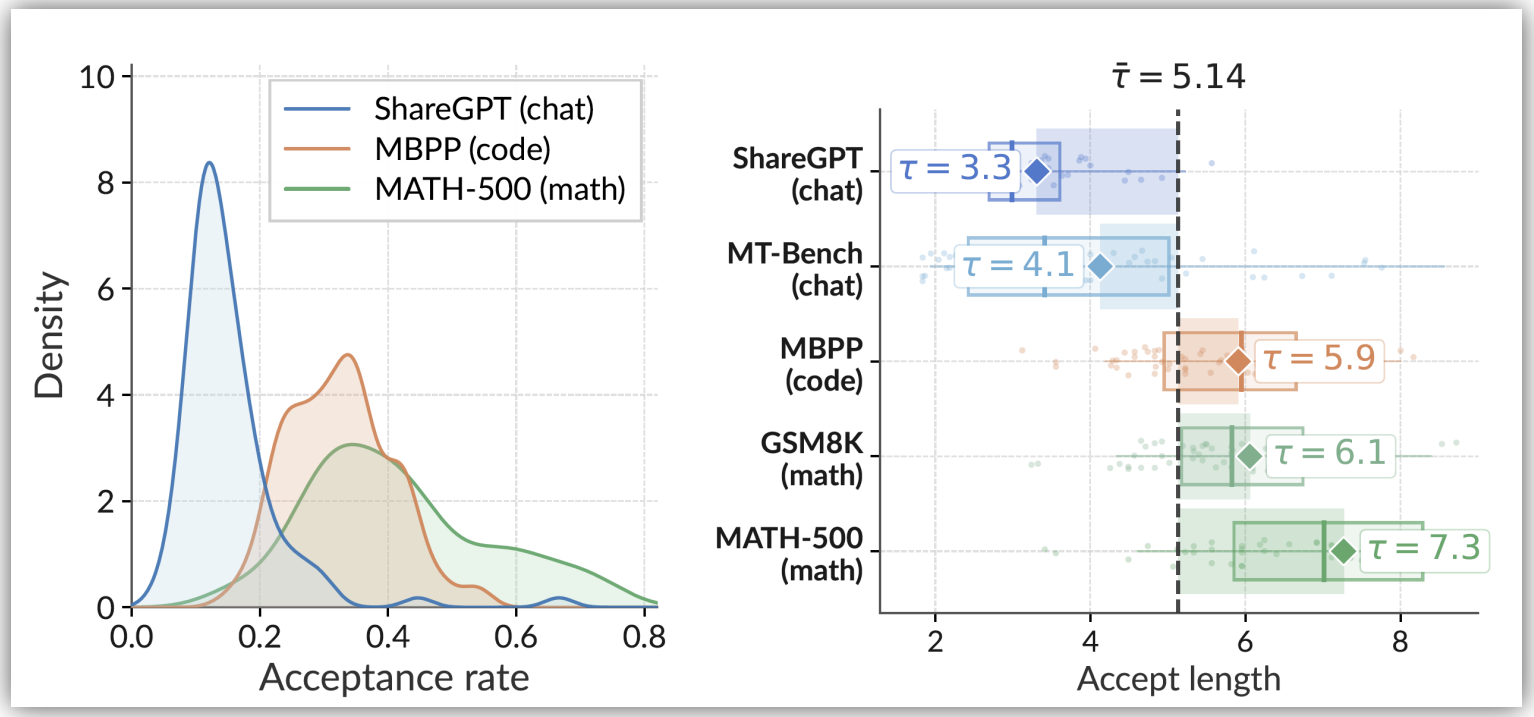
❌ Cons: high variance

- Domain-level variance
- Token-level variance

Observation 1: Domain-Level Variance

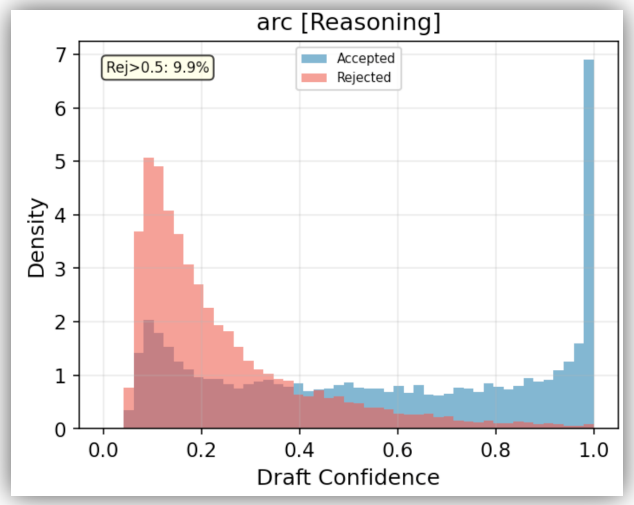
AdaFlash

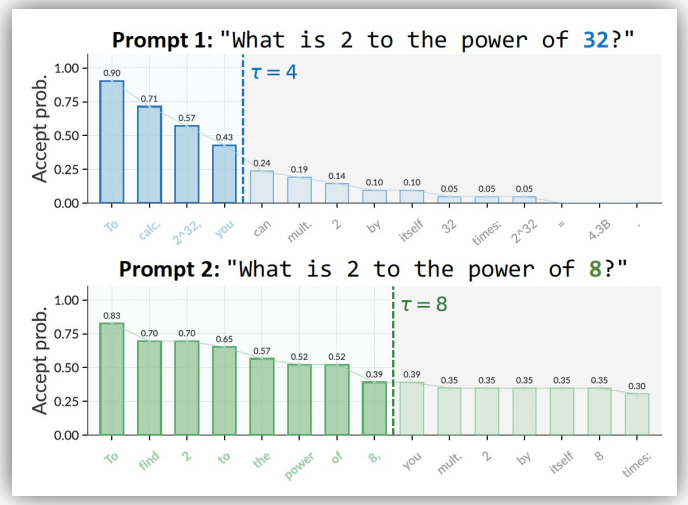
- Drafter's acceptance rate varies on different task domains:



Motivation: ***online update*** the draft model during deployment.

Observation 2: Token-Level Variance





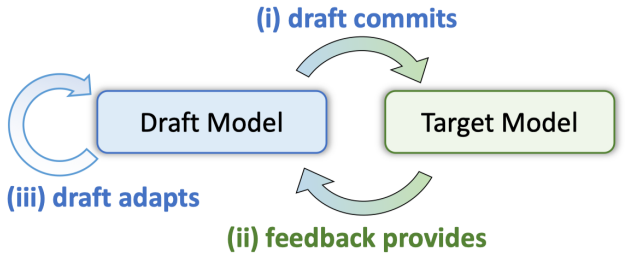
- Verify all k tokens is consuming, especially under ***high concurrency.***
- Draft quality varies along positions, do not need to verify all *k* tokens

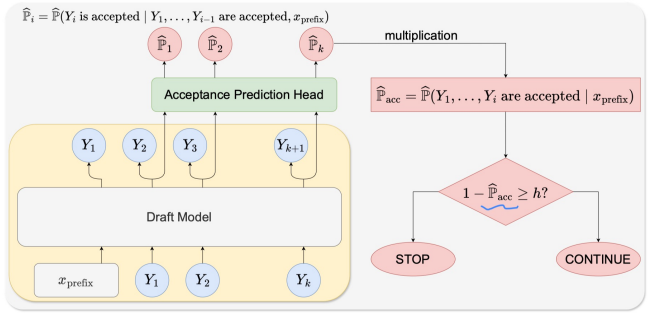
The verify length should be adjusted according to ***task / prompt.***

Motivation: target model's verify length should be ***adaptive.***

AdaFLash: Overview

- Diffusion Drafter: bi-directional attention
  - Pros: parallel drafting;
  - **Cons:** ***high variance, domain/task specific.***
- Real-world Deployment: domain inconsistency, high concurrency
  1. For the domain-level variance:
     - Utilize ***interactive feedback***
     - On-policy distillation to adapt to new domain
  2. For the token-level variance:
     - ***Adaptive length head*** for dynamic verify length
     - Reducing unnecessary verification

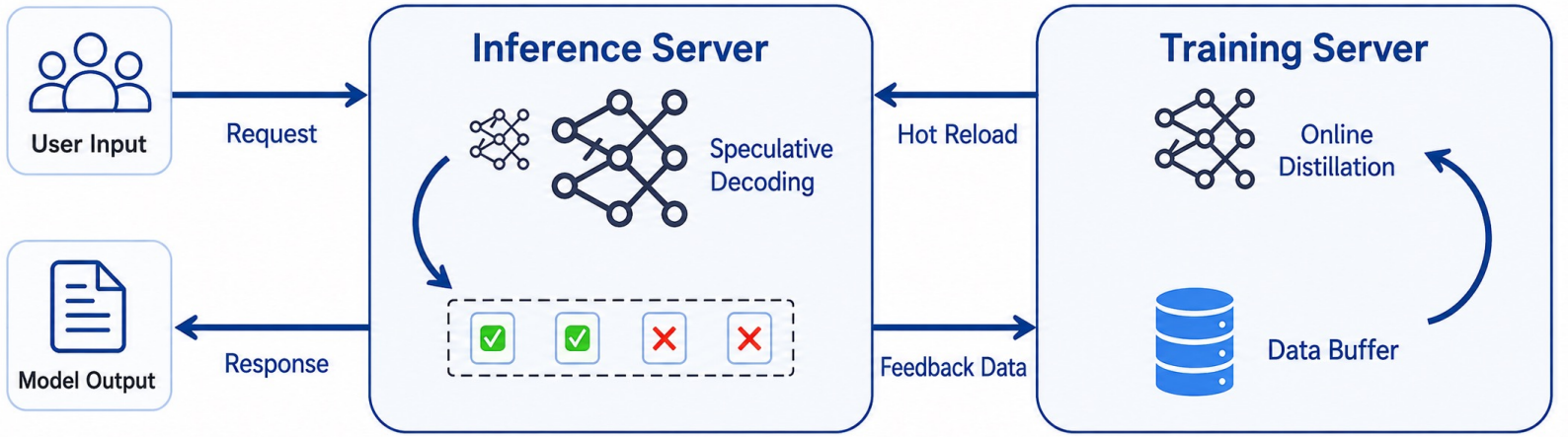




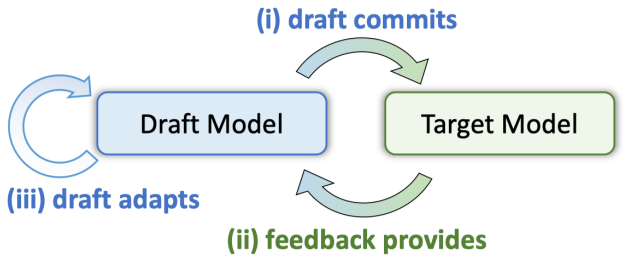
Part I: *OPD for Diffusion Drafters*

AdaFlash

- Utilize the ***free feedback*** in inference to help the draft model adapt:



- Continuously improve the draft model ***on-the-fly***:
  - Acceptance length ↑
  - Throughput (Speedup) ↑



Part I: *OPD for Diffusion Drafters*

AdaFlash

- Mixture OPD loss with ***reverse-KL:*** $\ell_{\text{OPD}} = \alpha \ell_{\text{hard}} + (1-\alpha)\ell_{\text{rkl}}$

$$\ell_{\text{rkl}} = \frac{1}{k}\sum_{i=1}^{k} \text{KL}\big([q_{\mathbf{w}}(\cdot \mid \mathbf{x}_{<t}, \mathbf{z})]_i \,\|\, p_{\mathbf{v}}(\cdot \mid \mathbf{x}_{<t+i})\big)$$

Reverse KL loss: mitigate "*mode covering*"

$$\ell_{\text{hard}} = -\frac{1}{k}\sum_{i=1}^{k} \log q_{\mathbf{w}}(x_i^* \mid \mathbf{x}_{<t+i}), \quad (x_i^* = \arg\max_x p_{\mathbf{v}}(x \mid \mathbf{x}_{<t+i}))$$

Hard label loss: strengthen the top-1 token signal

- Entry-wise ***divergence clipping***:

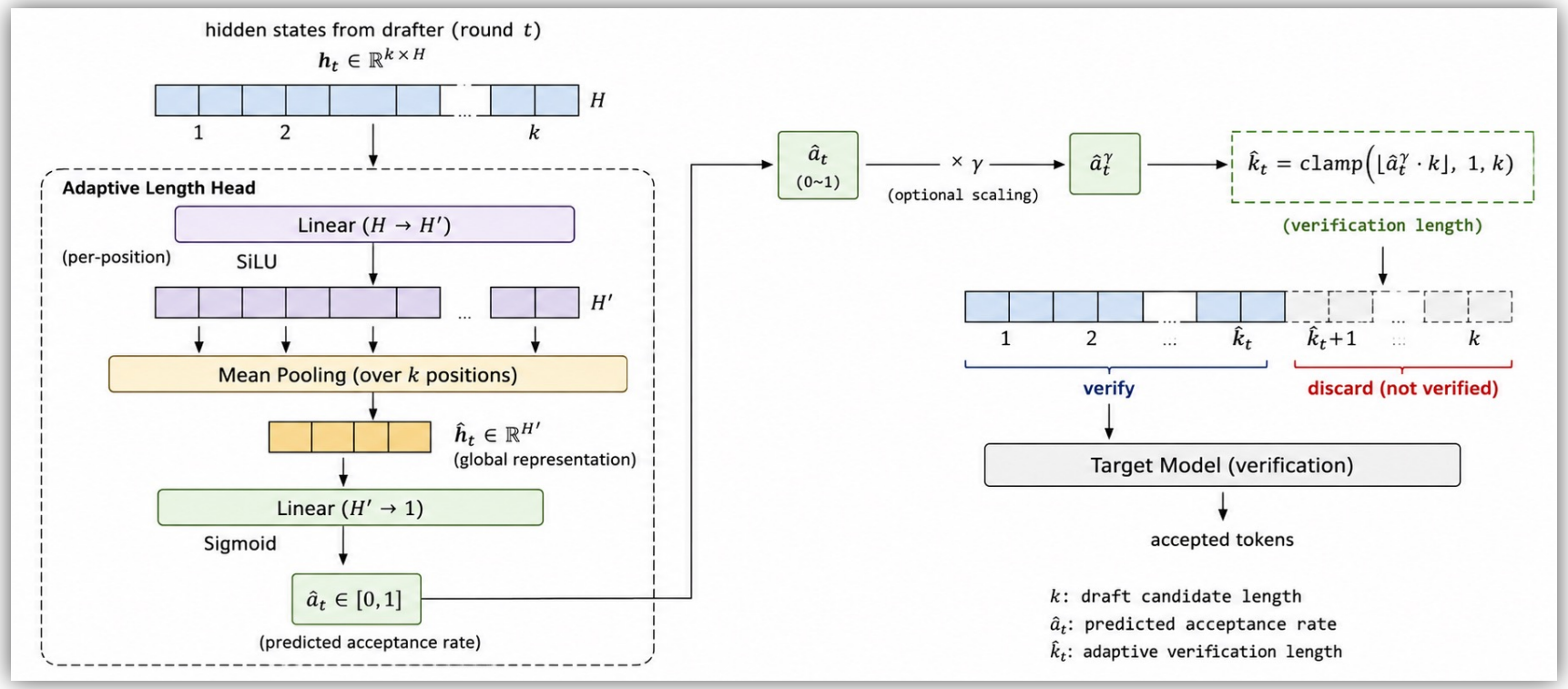
Prevent the diffusion drafter from being misled by outlier *tokens*

$$\ell_{\text{rkl}}^{\text{clip}} = \frac{1}{k}\sum_{i=1}^{k}\sum_{y\in\mathcal{V}} \min\left\{[q_{\mathbf{w}}(y \mid \mathbf{x}_{<t}, \mathbf{z})]_i \log \frac{[q_{\mathbf{w}}(y \mid \mathbf{x}_{<t}, \mathbf{z})]_i}{p_{\mathbf{v}}(y \mid \mathbf{x}_{<t+i})}, \delta\right\}$$

$$\ell_{\text{OPD}} = \alpha\, \ell_{\text{hard}} + (1-\alpha)\, \ell_{\text{rkl}}^{\text{clip}}$$

Part II: *Adaptive Length Head*

- A lightweight ***adaptive length head*** that predicts the acceptance ratio on the fly.



Truncates the draft sequence to ***reduce unnecessary verification.***

Part III: *Infrastructure Design*

AdaFlash

- Target model verification candidate lengths: *Fixed* → ***Adaptive length***

```
if self.use_prob_head:
    # run prob head on a secondary CUDA stream, overlapping with LM head
    prob_stream = _get_prob_head_stream(self.device)
    main_stream = torch.cuda.current_stream(self.device)
    prob_stream.wait_stream(main_stream)  # ensure draft_hidden is ready

    if self.tp_rank == 0:
        with torch.cuda.stream(prob_stream):
            with torch.no_grad():
                prob_logits = self.draft_model.prob_head(draft_hidden)  # [bs, block_size,
1]
                # torch.compile fuses sigmoid + cumprod + candidate_lens
                prob_fn = _get_compiled_prob_head_fn()
                candidate_lens = prob_fn(
                    prob_logits,
                    self.prob_head_mul_threshold,
                    self.prob_head_candidate_len_min,
                    self.block_size,
                )
```

- Exponential moving average (**EMA**) dynamically estimates the number of affordable requests of the current task.
- ***Infra*** optimization: operator fusion, concurrent launch, torch compile…
- The scheduler returns requests to pending queue when exceeding the budget.

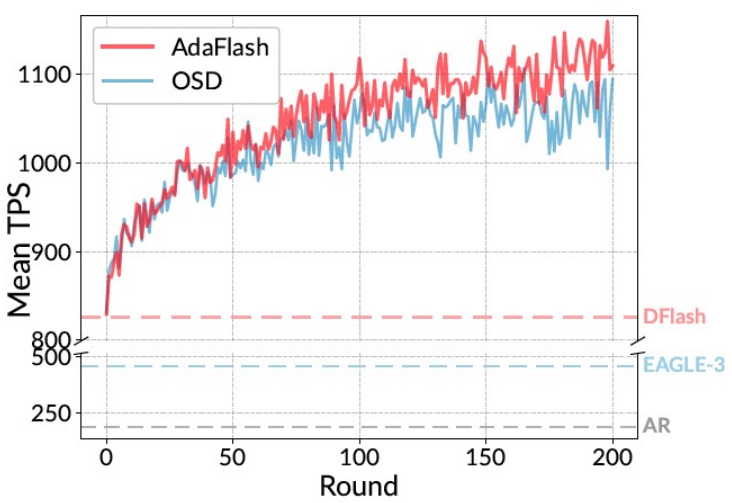
Experimental Results

Model	Method	Conc.	MATH				CODE				CHAT		HYBRID		*Avg.*	
			MathQA		GSM8K		OpenCodeInstruct		CodeAlpaca		ShareGPT		Blend			
			Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ
Q3-8B	EAGLE-3	1	2.45×	4.60	2.55×	4.75	2.20×	4.06	2.37×	4.58	2.09×	3.89	2.36×	4.54	2.34×	4.40
		32	0.72×	4.60	0.70×	4.76	0.68×	4.07	0.67×	4.58	0.61×	3.90	0.70×	4.54	0.68×	4.41
		64	0.46×	4.59	0.43×	4.75	0.44×	4.07	0.40×	4.57	0.40×	3.90	0.45×	4.54	0.43×	4.40
		128	0.34×	4.59	0.35×	4.75	0.33×	4.07	0.32×	4.58	0.30×	3.90	0.33×	4.54	0.33×	4.41
	DFlash	1	4.38×	7.09	3.84×	6.27	4.08×	6.69	3.39×	5.63	2.11×	3.40	3.39×	6.07	3.53×	5.86
		32	1.86×	7.10	1.59×	6.28	1.77×	6.68	1.46×	5.66	1.05×	3.42	1.51×	6.04	1.54×	5.86
		64	1.24×	7.11	1.03×	6.29	1.17×	6.67	0.91×	5.65	0.70×	3.41	0.98×	6.06	1.01×	5.86
		128	0.90×	7.11	0.82×	6.28	0.86×	6.67	0.74×	5.66	0.51×	3.40	0.74×	6.04	0.76×	5.86
	OSD	1	5.13×	9.44	4.33×	7.45	4.78×	8.21	3.65×	6.59	2.18×	3.78	3.60×	6.83	3.95×	7.05
		32	2.16×	9.42	1.80×	7.47	2.03×	8.20	1.55×	6.61	1.08×	3.79	1.57×	6.82	1.70×	7.05
		64	1.45×	9.44	1.16×	7.43	1.34×	8.20	0.97×	6.60	0.72×	3.80	1.07×	6.85	1.12×	7.05
		128	1.05×	9.47	0.92×	7.44	0.97×	8.20	0.78×	6.62	0.53×	3.81	0.76×	6.82	0.83×	7.06
	ADAFLASH	1	**5.32**×	9.83	**4.54**×	7.75	**4.87**×	8.40	**3.75**×	6.80	**2.24**×	3.92	**3.66**×	7.00	**4.06**×	7.28
		32	**2.27**×	9.84	**1.83**×	7.76	**2.03**×	8.39	**1.58**×	6.79	**1.10**×	3.92	**1.61**×	6.97	**1.74**×	7.28
		64	**1.65**×	9.51	**1.32**×	7.28	**1.51**×	8.02	**1.18**×	6.31	**1.04**×	3.61	**1.26**×	6.56	**1.33**×	6.88
		128	**1.34**×	9.51	**1.27**×	7.27	**1.25**×	8.05	**1.11**×	6.30	**0.87**×	3.58	**1.05**×	6.55	**1.15**×	6.88
Q3-30B	EAGLE-3	1	1.79×	4.39	1.56×	4.06	1.96×	5.15	1.93×	4.94	1.32×	3.08	1.79×	4.49	1.73×	4.35
		32	1.33×	4.40	1.15×	4.06	2.85×	5.15	1.71×	4.93	1.15×	3.08	1.49×	4.50	1.61×	4.35
		64	1.03×	4.40	0.86×	4.06	1.44×	5.15	1.33×	4.93	0.91×	3.08	1.16×	4.48	1.12×	4.35
		128	0.71×	4.41	0.63×	4.06	1.15×	5.15	1.10×	4.93	0.60×	3.08	0.77×	4.49	0.83×	4.35
	DFlash	1	2.09×	5.15	1.84×	4.85	2.56×	6.86	2.39×	6.03	1.04×	2.79	2.07×	5.43	2.00×	5.19
		32	1.61×	5.14	1.41×	4.85	3.80×	6.84	2.15×	6.01	1.14×	2.79	1.74×	5.42	1.98×	5.18
		64	1.26×	5.16	1.05×	4.84	1.95×	6.84	1.68×	6.01	0.93×	2.81	1.39×	5.42	1.38×	5.18
		128	0.87×	5.15	0.77×	4.84	1.55×	6.84	1.40×	6.00	0.60×	2.80	0.92×	5.42	1.02×	5.18
	OSD	1	2.79×	8.38	2.25×	6.44	2.96×	8.28	2.58×	7.09	1.30×	3.48	2.24×	6.38	2.35×	6.68
		32	2.04×	8.39	1.60×	6.43	4.30×	8.27	2.26×	6.94	1.23×	3.35	1.83×	6.37	2.21×	6.63
		64	1.59×	8.36	1.26×	6.43	2.21×	8.27	1.77×	6.95	0.98×	3.37	1.43×	6.35	1.54×	6.62
		128	1.11×	8.39	0.89×	6.43	1.75×	8.27	1.47×	6.95	0.65×	3.39	0.99×	6.35	1.14×	6.63
	ADAFLASH	1	**2.86**×	8.53	**2.32**×	6.69	**2.99**×	8.46	**2.65**×	7.21	**1.32**×	3.52	**2.27**×	6.56	**2.40**×	6.83
		32	**2.05**×	8.53	**1.59**×	6.69	**4.32**×	8.43	**2.33**×	7.08	**1.23**×	3.42	**1.85**×	6.53	**2.23**×	6.78
		64	**1.60**×	8.08	**1.32**×	6.42	**2.35**×	7.96	**1.98**×	6.63	**1.08**×	3.12	**1.51**×	6.16	**1.64**×	6.40
		128	**1.57**×	8.11	**1.28**×	6.43	**2.57**×	7.96	**2.32**×	6.59	**1.04**×	3.11	**1.33**×	6.14	**1.69**×	6.39

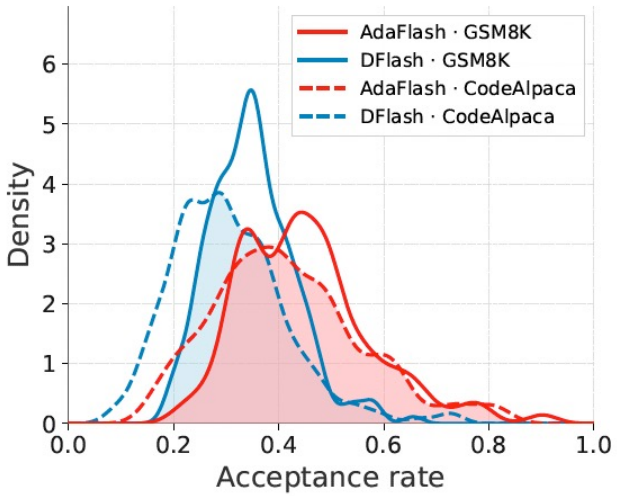
AdaFlash consistently improves speedup.

Significant gains in ***high-concurrency*** scenarios.

Experiment: Analysis



(a)



(b)

(c)

Figure 2: Analysis of ADAFLASH using Qwen3-8B as target model: (a) Speedup vs. online adaptation rounds on the MathQA dataset, showing continuous improvement over time. (b) Domain-level variance of DFlash and ADAFLASH, illustrating how on-policy distillation mitigates domain-level variance. (c) Token-level variance of DFlash and ADAFLASH on the MathQA dataset, showing acceptance probability along token position.

AdaFlash reduces domain-level and token-level variance.

Experiment: Cross-Domain

Table 5: Cross-domain generalization on Qwen3-8B. We compare offline DFlash with AdaFlash trained offline on PerfectBlend, a mixed-domain dataset distinct from each test benchmark, and evaluated with fixed weights without further online adaptation. In-domain AdaFlash results are reported in Table 1.

Method	Conc.	Math				Code				*Avg.*	
		MathQA		GSM8K		OpenCodeInstruct		CodeAlpaca			
		Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ
DFlash	1	4.38×	7.09	3.84×	6.27	4.08×	6.69	3.39×	5.63	3.92×	6.42
	32	1.86×	7.10	1.59×	6.28	1.77×	6.68	1.46×	5.66	1.67×	6.43
	64	1.24×	7.11	1.03×	6.29	1.17×	6.67	0.91×	5.65	1.09×	6.43
	128	0.90×	7.11	0.82×	6.28	0.86×	6.67	0.74×	5.66	0.83×	6.43
AdaFlash (cross-domain)	1	4.88×	8.37	4.65×	8.10	4.11×	6.98	3.43×	5.93	4.27×	7.35
	32	2.07×	8.39	1.84×	8.11	1.76×	6.96	1.48×	5.95	1.79×	7.35
	64	1.55×	7.91	1.37×	7.79	1.38×	6.65	1.13×	5.53	1.36×	6.97
	128	1.29×	7.93	1.35×	7.76	1.15×	6.64	1.07×	5.54	1.22×	6.97

AdaFlash generalizes well across different domains.

Experiment: Long-Sequence Generation

Table 6: Long-sequence generation on MATH-500 (32K context, Qwen3-8B, thinking mode enabled).

Method	Conc.	Speedup	τ	Acc. Rate
EAGLE-3	1	2.26×	4.25	0.105
	32	1.82×	4.37	0.109
DFlash	1	2.60×	3.97	0.198
	32	2.46×	4.04	0.203
OSD	1	3.51×	5.28	0.285
	32	3.40×	5.31	0.287
AdaFlash	1	**3.67×**	**5.37**	**0.291**
	32	**3.52×**	**5.14**	**0.276**

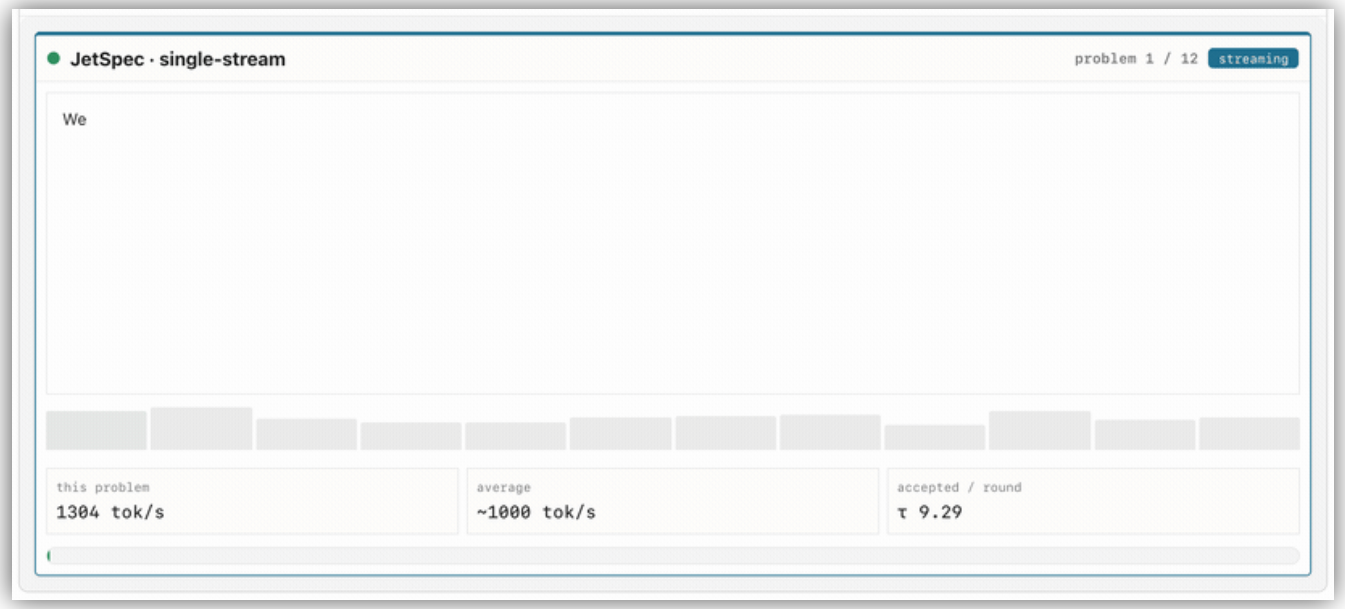
Table 7: Long-sequence generation on AIME25 (32K context, Qwen3-8B, thinking mode enabled).

Method	Conc.	Speedup	τ	Acc. Rate
EAGLE-3	1	2.30×	4.16	0.102
	32	1.68×	4.26	0.105
DFlash	1	1.89×	3.32	0.155
	32	1.54×	3.36	0.157
OSD	1	3.01×	5.21	0.281
	32	2.52×	5.24	0.283
AdaFlash	1	**3.18×**	**5.32**	**0.288**
	32	**2.77×**	**4.98**	**0.265**

AdaFlash maintains its advantage in ***long-sequence generation***.

JetSpec: Breaking the Scaling Ceiling of Speculative Decoding with Parallel Tree Drafting

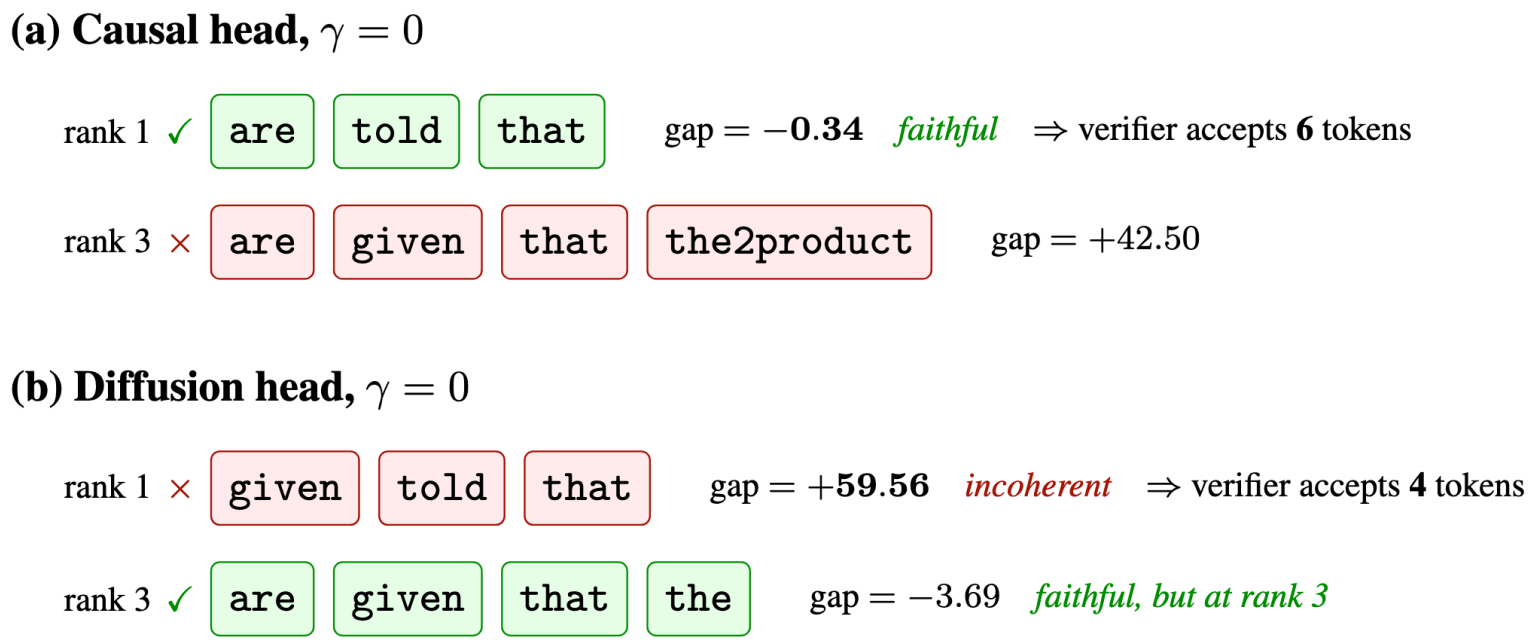
Lanxiang Hu[1] **Zhaoxiang Feng**[1] **Yulun Wu**[2] **Haoran Yuan**[3] **Yujie Zhao**[1] **Yu-Yang Qian**[4]
Bojun Wang[5] **Peng Zhao**[4] **Daxin Jiang**[5] **Yibo Zhu**[5] **Tajana Rosing**[1] **Hao Zhang**[1]
[1]UC San Diego [2] Zhejiang University [3]UIUC [4]Nanjing University [5]StepFun



arXiv:2606.18394
2026, Jun 25

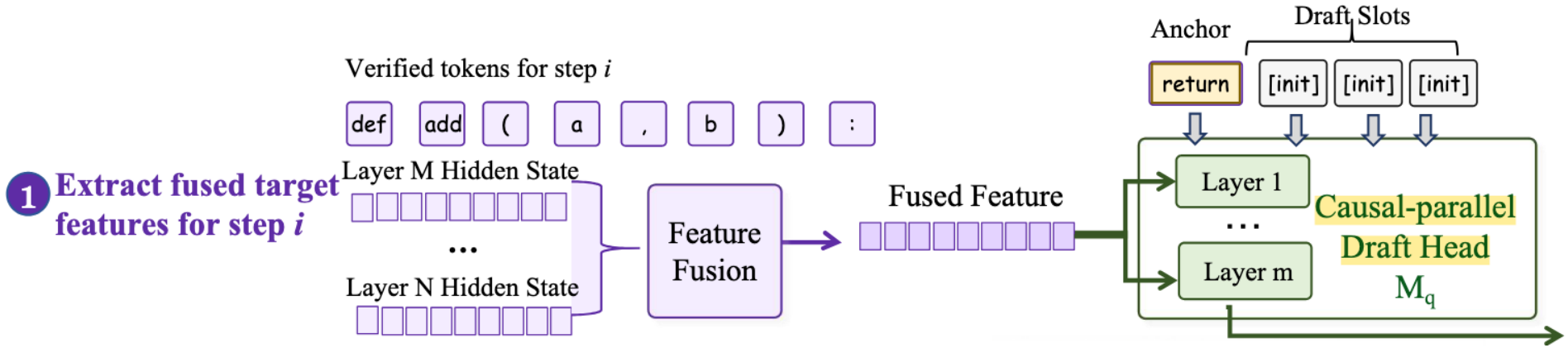
JetSpec

- 并行树形草稿生成的一个关键要求是，每个节点分布都应以其自身的分支前缀（branch prefix）基础上，计算条件概率。
- 然而，当今常用的 DFlash 方法，所有草稿 token 同时解码，没有时序性



JetSpec

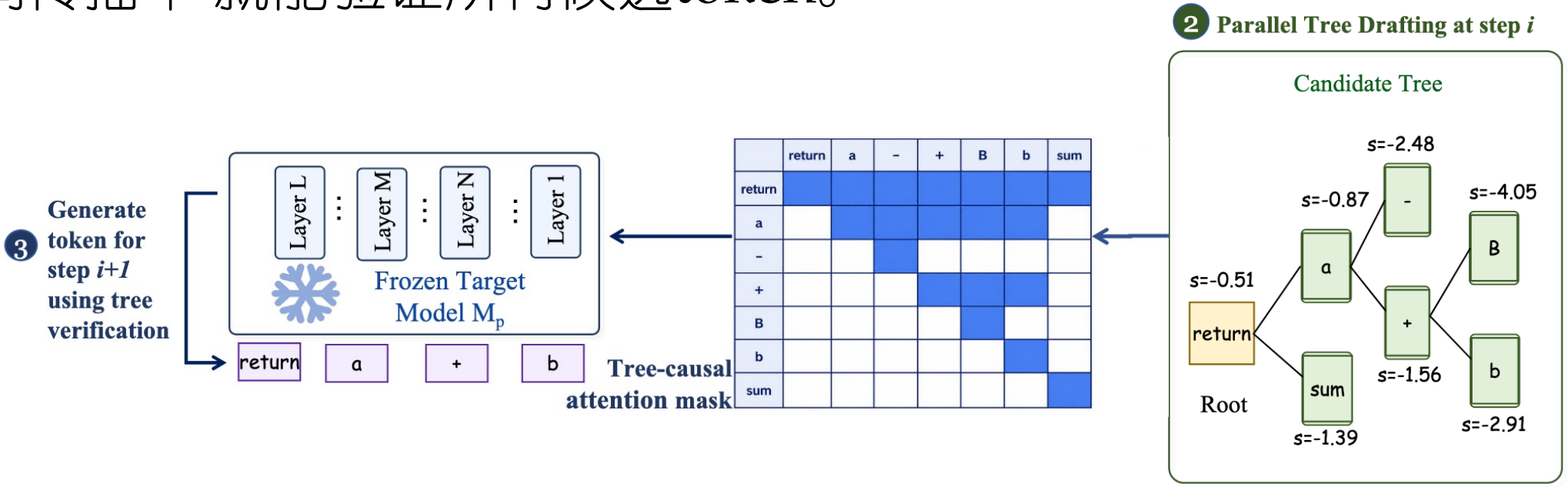
- JetSpec 采用树状草稿（tree-drafting）与树状验证，在目标模型的一次前向传播中 就能验证所有候选token。



- 对草稿头应用树状因果注意力掩码。
- 在保持并行的同时，确保每个分支遵循类似自回归的依赖结构。

JetSpec

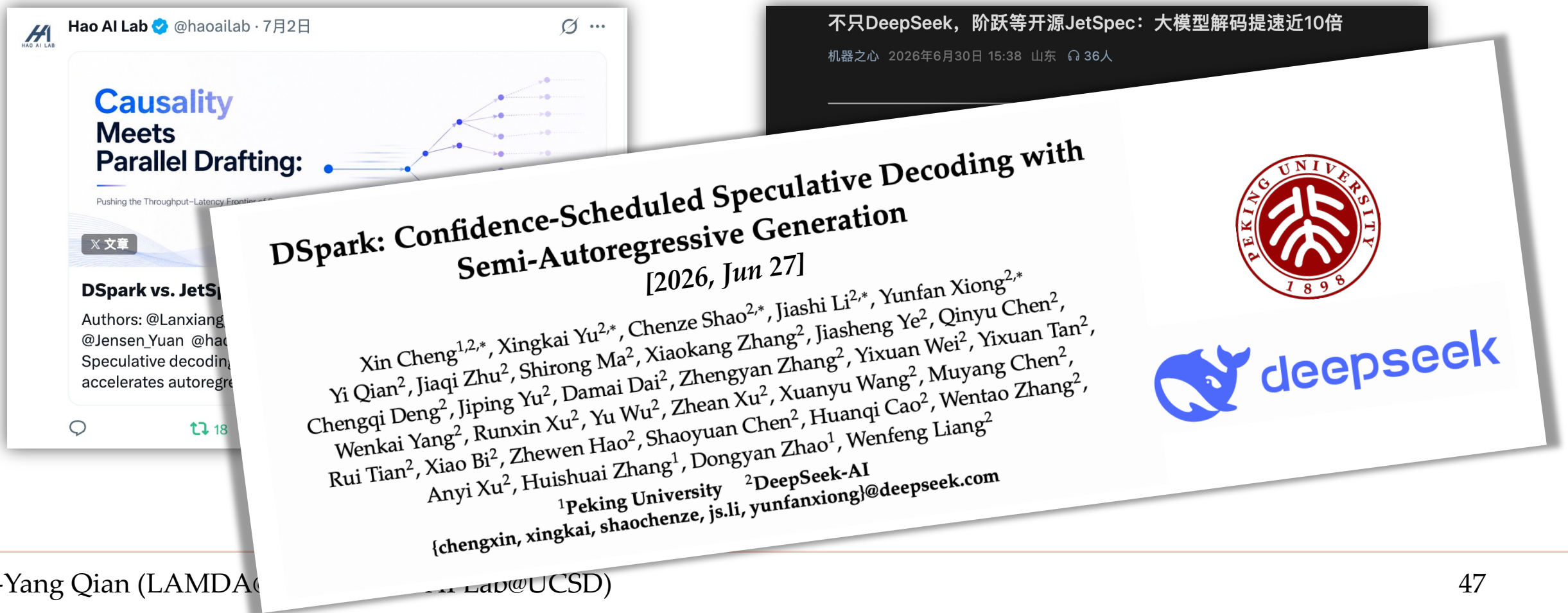
- JetSpec 采用树状草稿（tree-drafting）与树状验证，在目标模型的一次前向传播中 就能验证所有候选token。



StepFun with JetSpec

JetSpec

- 我们的 JetSpec 与 DeepSeek 的 **DSpark** 同期发布，都关注到了 diffusion drafter的“时序性”问题 。



Summary

- In this talk, we investigate a new paradigm: Diffusion LLM (***dLLM***)
  - · Parallel generation · Error correction · Random-order generation

[d3LLM]: Analyze *accuracy-parallelism trade-off*, propose ***"AUP score"***, and accelerate dLLMs via distillation and multi-block decoding.

[AdaFlash]: Leverage diffusion drafters for *online speculative decoding*, with OPD and adaptive length head for deployment-time acceleration.

[JetSpec]: Introduce *causal-parallel drafting* for tree-structured diffusion drafters, further improve the speedup of speculative decoding.

- ***dLLM*** is a promising, exciting, and rapidly evolving paradigm.

We welcome everyone to join and ***build the dLLM together!***

Summary

- In this talk, we investigate a new paradigm: Diffusion LLM (***dLLM***)
  - Our d3LLM introduce a new metric for dLLM: ***AUP* score**
- ***dLLM*** is a promising, exciting, and rapidly evolving paradigm.

We welcome everyone to join and ***build the dLLM together!***

Yu-Yang Qian
PhD @ NJU,
visiting PhD @ UCSD

d3LLM
https://arxiv.org/abs/2601.07568

AdaFlash
http://arxiv.org/abs/2607.19223

JetSpec
https://arxiv.org/abs/2606.18394

Thanks !