

Sampling From Masked Diffusion Models

September 2025

February 2025

Large Language Diffusion Models

Shen Nie^{1 *†} Fengqi Zhu^{1 *†} Zebin You^{1 †} Xiaolu Zhang^{2 ‡} Jingyang Ou¹ Jun Hu^{2 ‡} Jun Zhou²
Yankai Lin^{1 ‡} Ji-Rong Wen¹ Chongxuan Li^{1 ‡¶}

1. Introduction

What is now proved was once only imagined.

—William Blake

5. Conclusion and Discussion

In the middle of difficulty lies opportunity.

—Albert Einstein

February 2025

Large Language Diffusion Models

Shen Nie^{1*†} Fengqi Zhu^{1*†} Zebin You^{1†} Xiaolu Zhang^{2‡} Jingyang Ou¹ Jun Hu^{2‡} Jun Zhou²
Yankai Lin^{1‡} Ji-Rong Wen¹ Chongxuan Li^{1‡¶}

1. Introduction

What is now proved was once only imagined.

—William Blake

5. Conclusion and Discussion

In the middle of difficulty lies opportunity.

—Albert Einstein



February 2025

Large Language Diffusion Models

Shen Nie^{1*†} Fengqi Zhu^{1*†} Zebin You^{1†} Xiaolu Zhang^{2‡} Jingyang Ou¹ Jun Hu^{2‡} Jun Zhou²
Yankai Lin^{1‡} Ji-Rong Wen¹ Chongxuan Li^{1‡¶}

1. Introduction

What is now proved was once only imagined.
—William Blake

5. Conclusion and Discussion

In the middle of difficulty lies opportunity.
—Albert Einstein



- “A nice direction to go in, we’ll see if it becomes useful (I’m not convinced)”

February 2025

Large Language Diffusion Models

Shen Nie^{1*†} Fengqi Zhu^{1*†} Zebin You^{1†} Xiaolu Zhang^{2‡} Jingyang Ou¹ Jun Hu^{2‡} Jun Zhou²
Yankai Lin^{1‡} Ji-Rong Wen¹ Chongxuan Li^{1‡¶}

1. Introduction

What is now proved was once only imagined.

—William Blake

5. Conclusion and Discussion

In the middle of difficulty lies opportunity.

—Albert Einstein



- “A nice direction to go in, we’ll see if it becomes useful (I’m not convinced)”
- “Unlikely that many people other than these authors (who clearly have tons of compute) will be willing to keep scaling these up”

Now

LLaDA 1.5: Variance-Reduced Preference Optimization for Large Language Diffusion Models

Fengqi Zhu^{1,*},[§] Rongzhen Wang^{1,*}, Shen Nie¹, Xiaolu Zhang³, Chunwei Wu³, Jun Hu³, Jun Zhou³,

Jianfei Chen², Yankai Lin^{1,*†}, Ji-Rong Wen¹, Chongxuan Li^{1,*‡}

¹Renmin University of China, ²Tsinghua University, ³Ant Group

* Equal contribution, [§] Work done during an internship at Ant Group, [†] Project leader,

[‡] Corresponding author

Paper Code Model

Try Our API



Products Resources Company

FEB 26, 2025 | COMPANY

Introducing Mercury, the World's First Commercial-Scale Diffusion Large Language Model.

Stefano Ermon
CEO

Dream 7B: Diffusion Large Language Models

Jiacheng Ye^{1*} Zhihui Xie^{1*} Lin Zheng^{1*} Jiahui Gao^{2*} Zirui Wu
Xin Jiang² Zhenguo Li² Lingpeng Kong¹

¹The University of Hong Kong ²Huawei Noah's Ark Lab

DreamLM/Dream Dream-7B

April 2, 2025



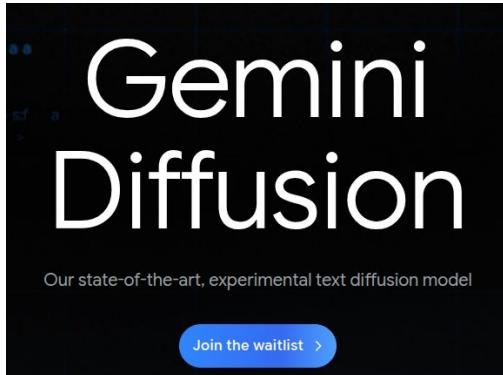
LLM generates the ENTIRE output at once (world's first diffusion LLM)

195K views • 6 months ago

Edit Flows: Flow Matching with Edit Operations

Marton Havasi¹, Brian Karrer¹, Itai Gat¹, Ricky T.Q. Chen¹

¹FAIR at Meta



Recap

Autoregressive models (ARMs) are the current standard for LLMS

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

Recap

Autoregressive models (ARMs) are the current standard for LLMS

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


Autoregressive formulation

But other paradigms, e.g. diffusion, should (theoretically) be able to work just as well at generative modelling:

$$\max_{\theta} \mathbb{E}_{p_{\text{data}}(x)} \log p_{\theta}(x) \Leftrightarrow \min_{\theta} \text{KL}(p_{\text{data}}(x) || p_{\theta}(x))$$


Generative modeling principles

It's kind of weird...

Table 4. Visualization of the Sampling Process and a Generated Multi-round Dialogue. In the response of LLaDA, darker colors indicate tokens predicted in the later stages of sampling, while lighter colors correspond to earlier predictions.

Sampling Process

User

Lily can run 12 kilometers per hour for 4 hours. After that, she runs 6 kilometers per hour.
How many kilometers can she run in 8 hours?

LLaDA

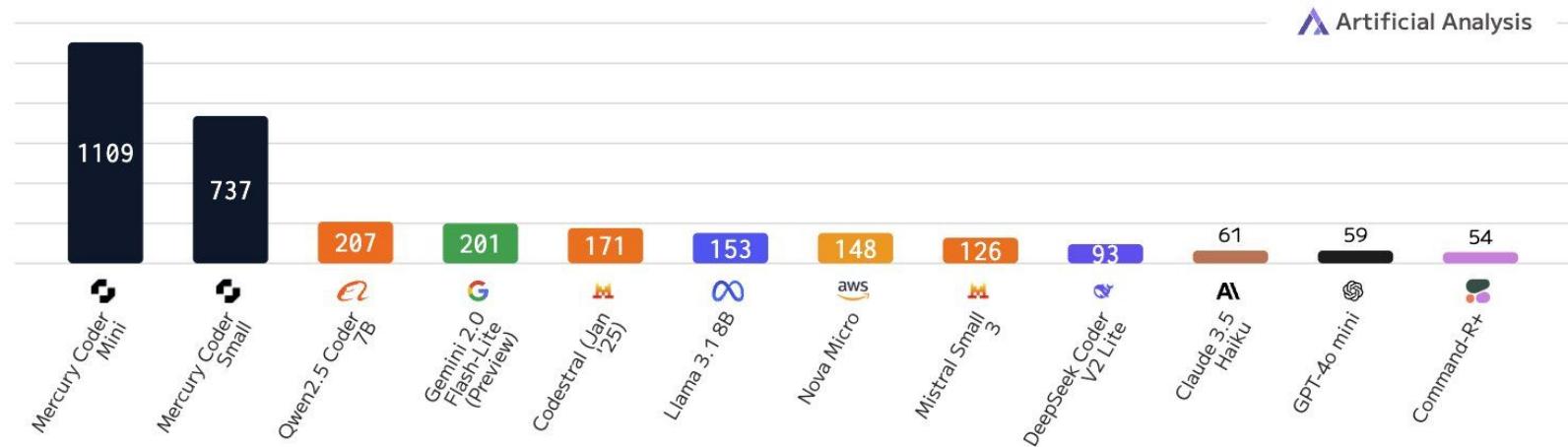
The image shows a grid of tokens representing the sampling process for a math problem. The tokens are arranged in three rows. Row 1: 'In', '4', 'hours', ',', 'Lily', 'runs', '12', '*', '4', '=', '48', 'kilometers', '.', 'After', '4'. Row 2: 'hours', 'she', 'runs', '6', '*', '4', '=', '24', 'kilometers', '.', 'In', 'total', 'she', 'runs', '48'. Row 3: '+', '24', '=', '72', 'kilometers', 'in', '8', 'hours', '.', 'Result', '72'. Darker shades of blue represent later sampled tokens.

But very fast!

(You can sample multiple tokens in one denoising step)

Output Speed: Smaller models

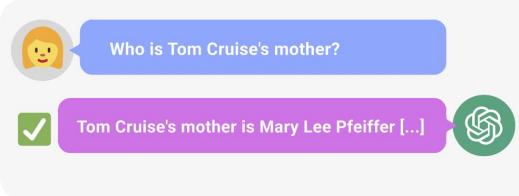
Output Tokens per Second; Higher is better; 1,000 Input Tokens; Coding focused workload



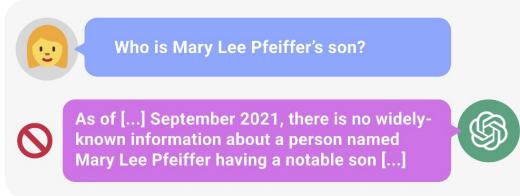
Autoregressive strategies aren't always ideal

- Reversal curse

A → B

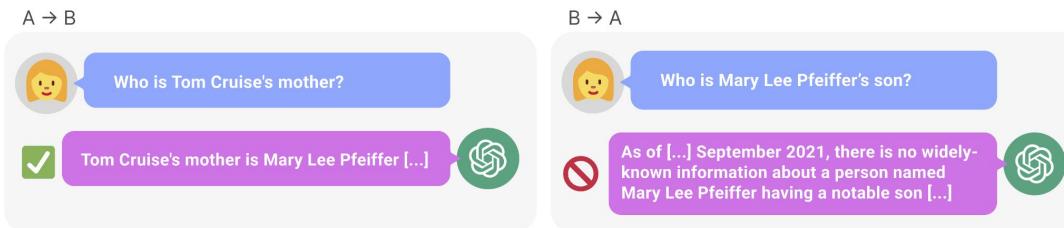


B → A



Autoregressive strategies aren't always ideal

- Reversal curse

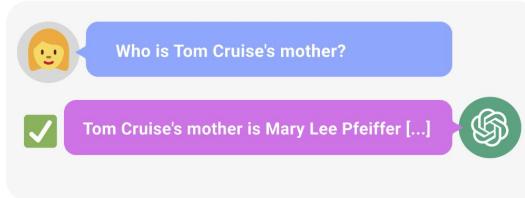


- Global-context in-filling tasks e.g. Sudoku

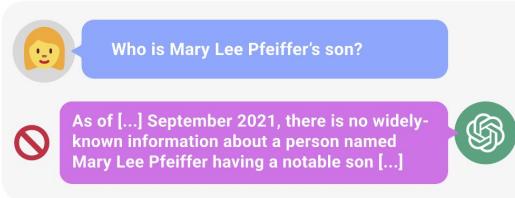
Autoregressive strategies aren't always ideal

- Reversal curse

A → B



B → A



ICLR 2025

Diffusion On Syntax Trees For Program Synthesis

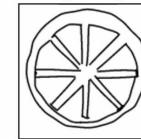
Shreyas Kapur, Erik Jenner, Stuart Russell

University of California, Berkeley

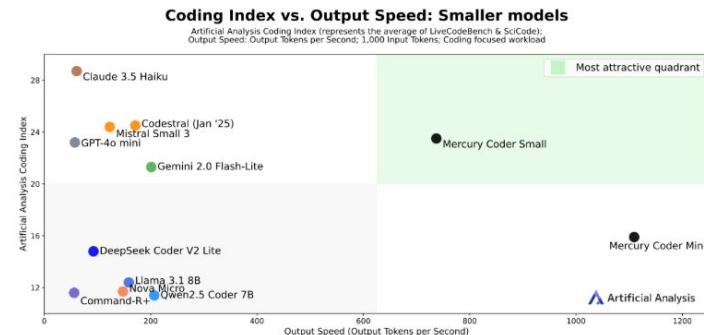
Program Output

```
(Subtract  
  (Subtract  
    Circle(x=2, y=20, r=2)  
    Circle(x=20, y=28, r=6)  
  )  
  Quad(x=26, y=14, w=30, h=24, angle=180°)  
)
```

Target Image



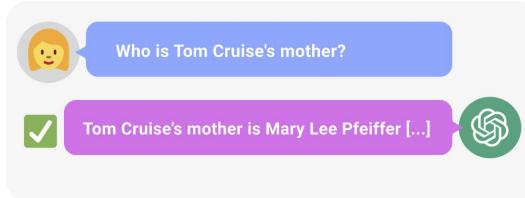
- Global-context in-filling tasks e.g. Sudoku
- Coding...?



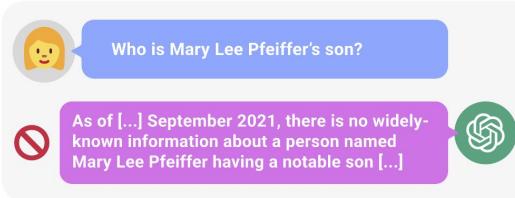
Autoregressive strategies aren't always ideal

- Reversal curse

A → B



B → A



ICLR 2025

Diffusion On Syntax Trees For Program Synthesis

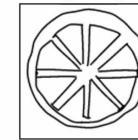
Shreyas Kapur, Erik Jenner, Stuart Russell

University of California, Berkeley

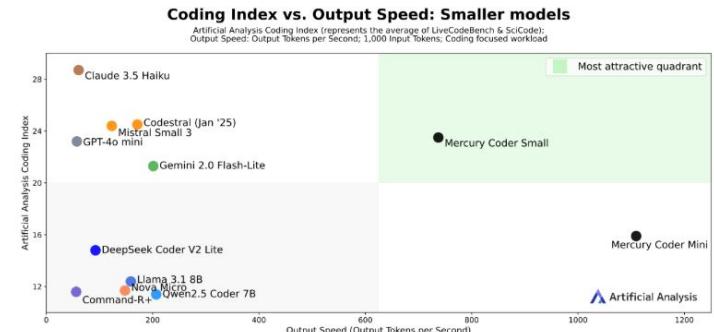
Program Output

```
(Subtract  
  (Subtract  
    Circle(x=2, y=20, r=2)  
    Circle(x=20, y=28, r=6)  
  )  
  Quad(x=26, y=14, w=30, h=24, angle=180°)  
)
```

Target Image



- Global-context in-filling tasks e.g. Sudoku
- Coding...?



...although, a properly scaled ARM can probably do most of these things correctly
– part of the pull here is doing them well with way fewer params

Masked Diffusion Model (MDM)

- Fixed length generation, N tokens

Masked Diffusion Model (MDM)

- Fixed length generation, N tokens
- ‘Noisy samples’ are just sequences of N repeated [MASK] tokens
 - Sometimes ‘noisy’ is defined with N random non-[MASK] tokens, but this seems less successful a strategy

Masked Diffusion Model (MDM)

- Fixed length generation, N tokens
- ‘Noisy samples’ are just sequences of N repeated [MASK] tokens
 - Sometimes ‘noisy’ is defined with N random non-[MASK] tokens, but this seems less successful a strategy
 - Also recent work allows us to do ‘edit’ operations to delete/insert/substitute tokens

Edit Flows: Flow Matching with Edit Operations

Marton Havasi¹, Brian Karrer¹, Itai Gat¹, Ricky T. Q. Chen¹

¹FAIR at Meta



Masked Diffusion Model (MDM)

- Fixed length generation, N tokens
- ‘Noisy samples’ are just sequences of N repeated [MASK] tokens
 - Sometimes ‘noisy’ is defined with N random non-[MASK] tokens, but this seems less successful a strategy
 - Also recent work allows us to do ‘edit’ operations to delete/insert/substitute tokens
- We want to model $p_{\text{data}}(x)$ via a mask predicting process $p_{\theta}(\cdot, x_t)$ for a partially-demasked sequence x_t at time $t \in [0, 1]$.

Edit Flows: Flow Matching with Edit Operations

Marton Havasi¹, Brian Karrer¹, Itai Gat¹, Ricky T. Q. Chen¹

¹FAIR at Meta



Masked Diffusion Model (MDM)

- Fixed length generation, N tokens
- ‘Noisy samples’ are just sequences of N repeated [MASK] tokens
 - Sometimes ‘noisy’ is defined with N random non-[MASK] tokens, but this seems less successful a strategy
 - Also recent work allows us to do ‘edit’ operations to delete/insert/substitute tokens
- We want to model $p_{\text{data}}(x)$ via a mask predicting process $p_{\theta}(\cdot, x_t)$ for a partially-demasked sequence x_t at time $t \in [0, 1]$.
- Many ways to learn/model $p_{\theta}(\cdot, x_t)$ but LLaDA still has the simplest explanation imo
 - (FYI: a good in-depth mathematical explanation is given in ‘Simplified and Generalized Masked Diffusion for Discrete Data’ by Shi et al. 2025)

Edit Flows: Flow Matching with Edit Operations

Marton Havasi¹, Brian Karrer¹, Itai Gat¹, Ricky T. Q. Chen¹

¹FAIR at Meta



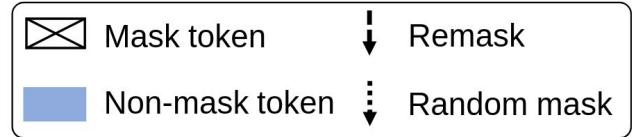
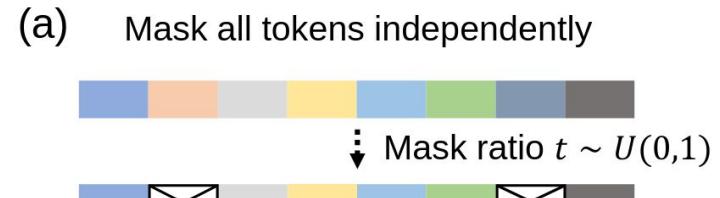
Masked Diffusion Model (MDM)

Model the data distribution $p_{\text{data}}(x)$ with a:

Masked Diffusion Model (MDM)

Model the data distribution $p_{\text{data}}(x)$ with a:

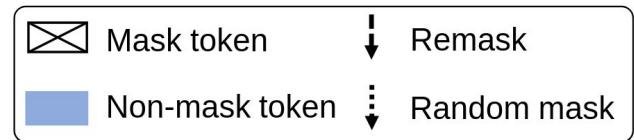
- forward process: gradually mask tokens (independently) until fully masked at $t = 1$:



Masked Diffusion Model (MDM)

Model the data distribution $p_{\text{data}}(x)$ with a:

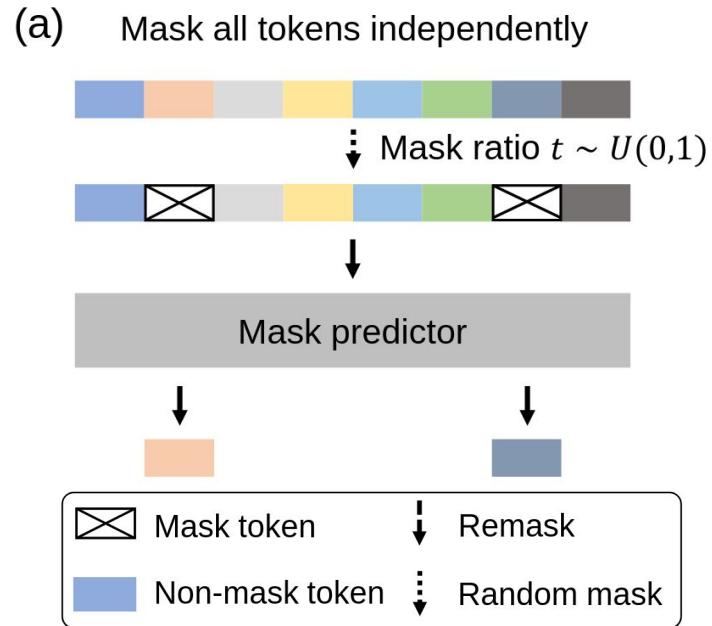
- forward process: gradually mask tokens (independently) until fully masked at $t = 1$:
 - At time $t \in [0, 1]$ each token is masked with prob. t
 - (Once a token is masked it stays masked for $t' > t$.)



Masked Diffusion Model (MDM)

Model the data distribution $p_{\text{data}}(x)$ with a:

- forward process: gradually mask tokens (independently) until fully masked at $t = 1$:
 - At time $t \in [0, 1]$ each token is masked with prob. t
 - (Once a token is masked it stays masked for $t' > t$.)
- reverse process: predict masked tokens as t moves from 0 to 1:



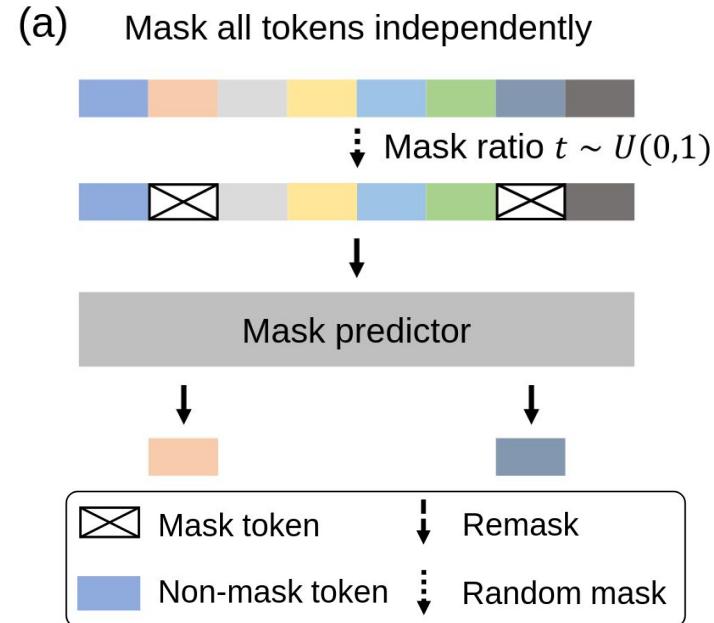
Masked Diffusion Model (MDM)

Model the data distribution $p_{\text{data}}(x)$ with a:

- forward process: gradually mask tokens (independently) until fully masked at $t = 1$:
 - At time $t \in [0, 1]$ each token is masked with prob. t
 - (Once a token is masked it stays masked for $t' > t$.)
- reverse process: predict masked tokens as t moves from 0 to 1:
 - Based on a *mask predictor* $p_\theta(\cdot, x_t)$ trained with cross-entropy loss only on the masked tokens:

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

where $x_0 \sim \mathcal{D}_{\text{train}}$, $t \sim \text{Uniform}[0, 1]$, and x_t is sampled from the forward process.



How do we apply the standard LLM pipeline?

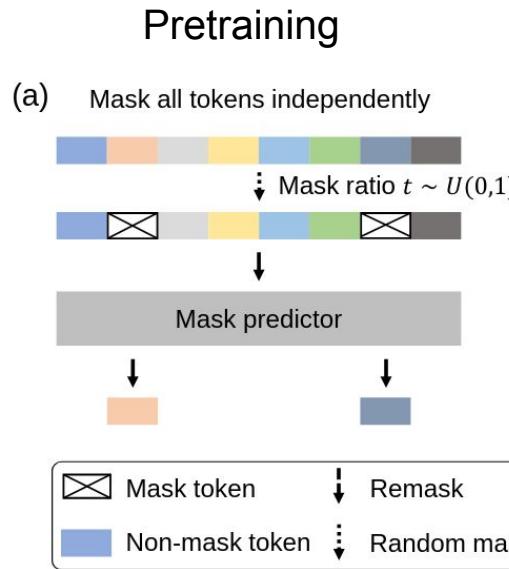


Figure 2. A Conceptual Overview of LLaDA. (a) Pre-training. LLaDA is trained on text with random masks applied independently to all tokens at the same ratio $t \sim U[0, 1]$. (b) SFT. Only response tokens are possibly masked. (c) Sampling. LLaDA simulates a diffusion process from $t = 1$ (fully masked) to $t = 0$ (unmasked), predicting all masks simultaneously at each step with flexible remask strategies.

How do we apply the standard LLM pipeline?

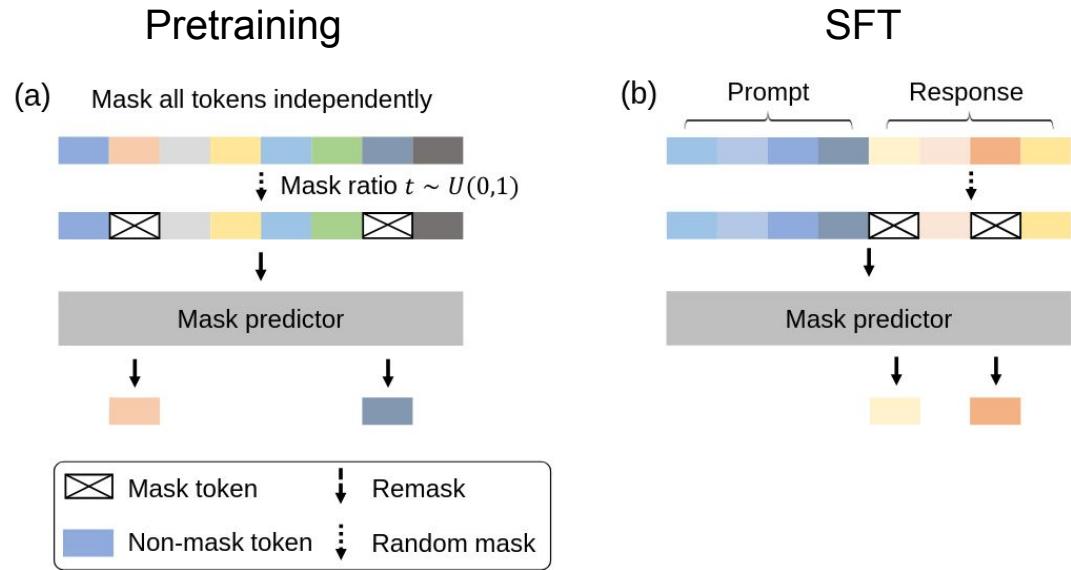


Figure 2. A Conceptual Overview of LLaDA. (a) Pre-training. LLaDA is trained on text with random masks applied independently to all tokens at the same ratio $t \sim U[0, 1]$. (b) SFT. Only response tokens are possibly masked. (c) Sampling. LLaDA simulates a diffusion process from $t = 1$ (fully masked) to $t = 0$ (unmasked), predicting all masks simultaneously at each step with flexible remask strategies.

How do we apply the standard LLM pipeline?

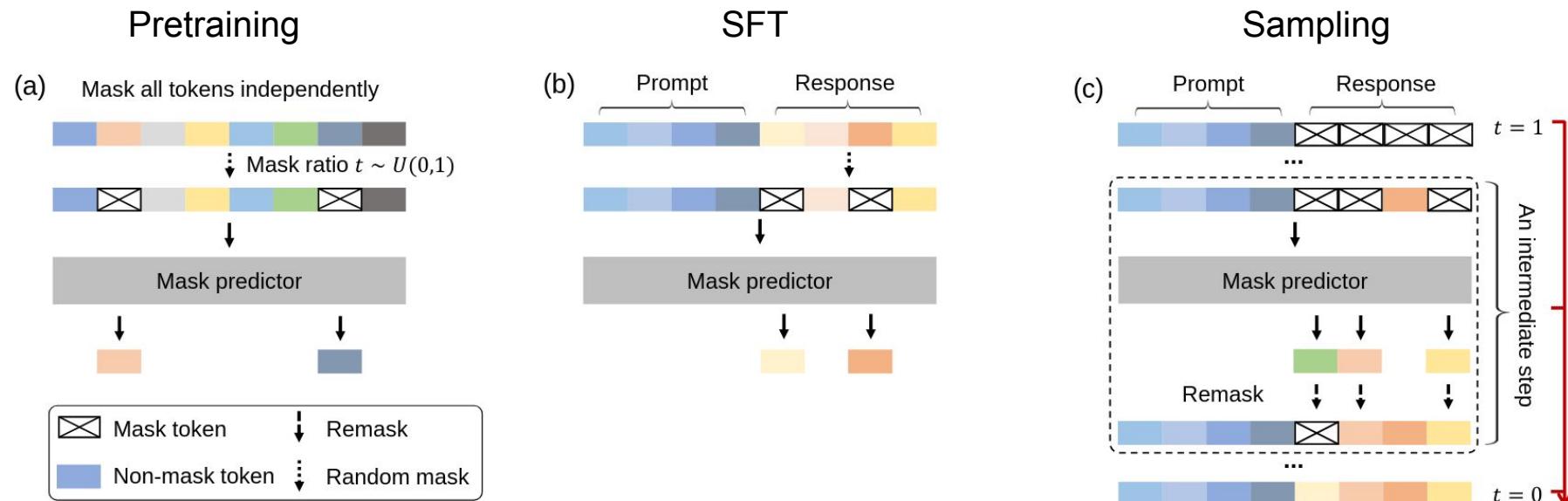


Figure 2. A Conceptual Overview of LLaDA. (a) Pre-training. LLaDA is trained on text with random masks applied independently to all tokens at the same ratio $t \sim U[0, 1]$. (b) SFT. Only response tokens are possibly masked. (c) Sampling. LLaDA simulates a diffusion process from $t = 1$ (fully masked) to $t = 0$ (unmasked), predicting all masks simultaneously at each step with flexible remask strategies.

Pretraining

Train the mask predictor $p_\theta(\cdot, x_t)$ as we just described:

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

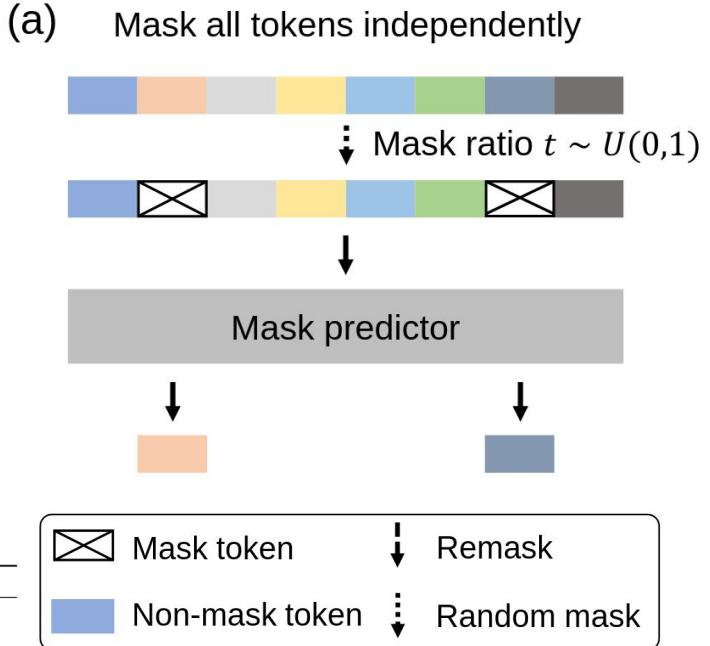
Algorithm 1 Pre-training of LLaDA

Require: mask predictor p_θ , data distribution p_{data}

- ```

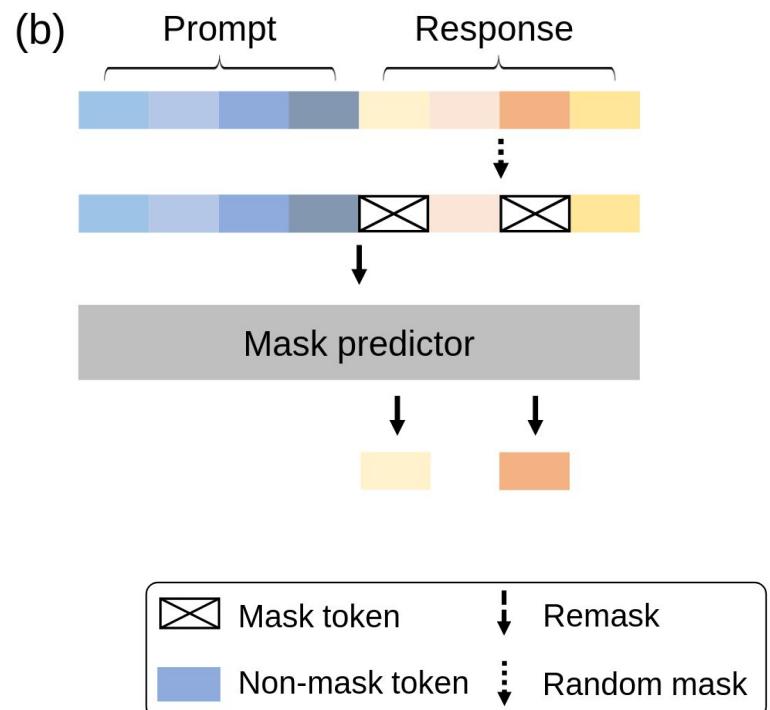
1: repeat
2: $x_0 \sim p_{\text{data}}, t \sim U(0, 1]$ # with a probability of 1%, the sequence length of x_0 follows $U[1, 4096]$
3: $x_t \sim q_{t|0}(x_t|x_0)$ # $q_{t|0}$ is defined in Eq. (7)
4: Calculate $\mathcal{L} = -\frac{1}{t^* L} \sum_{i=1}^L \mathbf{1}[x_t^i = M] \log p_\theta(x_0^i|x_t)$ # L is the sequence length of x_0
5: Calculate $\nabla_\theta \mathcal{L}$ and run optimizer.
6: until Converged
7: Return p_θ

```



# SFT

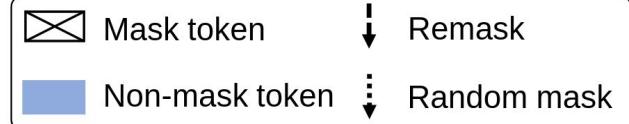
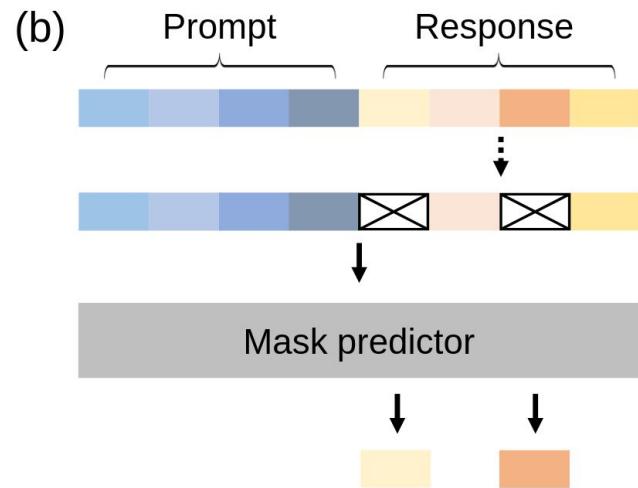
Slightly different loss (only care about response  $r_0$  given prompt  $p_0$  and intermediate response  $r_t$ ):



# SFT

Slightly different loss (only care about response  $r_0$  given prompt  $p_0$  and intermediate response  $r_t$ ):

$$-\mathbb{E}_{t,p_0,r_0,r_t} \left[ \frac{1}{t} \sum_{i=1}^{L'} \mathbf{1}[r_t^i = M] \log p_\theta(r_0^i | p_0, r_t) \right]$$

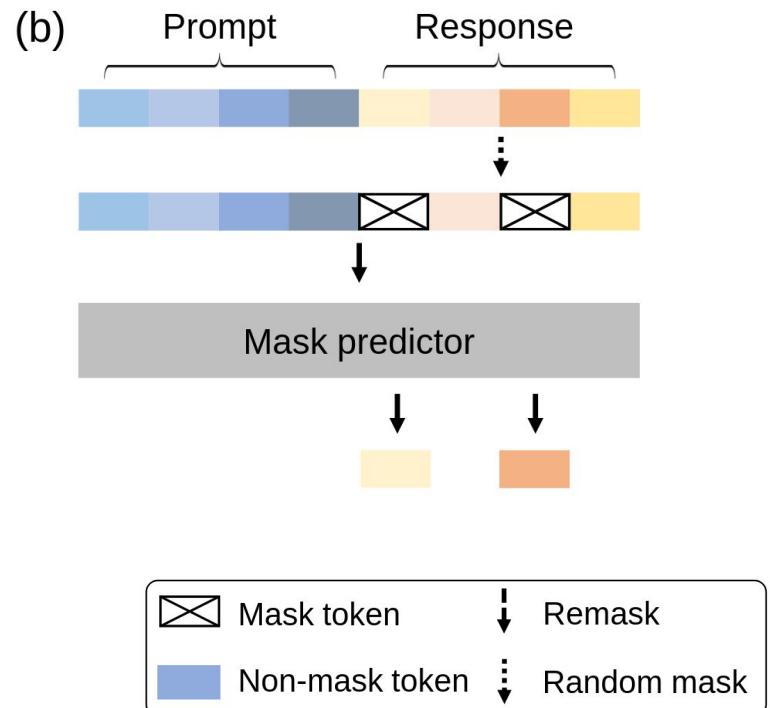


# SFT

Slightly different loss (only care about response  $r_0$  given prompt  $p_0$  and intermediate response  $r_t$ ):

$$-\mathbb{E}_{t,p_0,r_0,r_t} \left[ \frac{1}{t} \sum_{i=1}^{L'} \mathbf{1}[r_t^i = M] \log p_\theta(r_0^i | p_0, r_t) \right]$$

Where  $L'$  is a dynamic length:



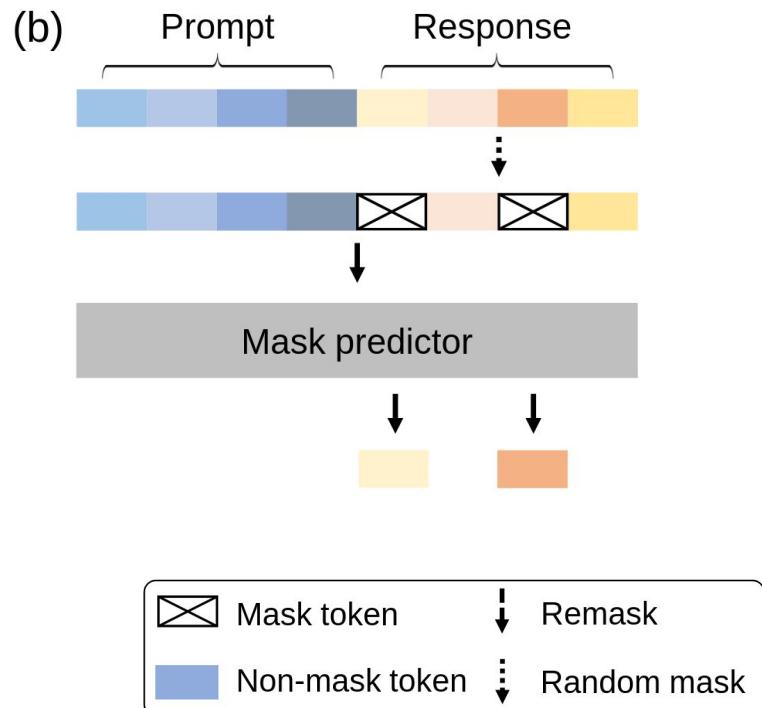
# SFT

Slightly different loss (only care about response  $r_0$  given prompt  $p_0$  and intermediate response  $r_t$ ):

$$-\mathbb{E}_{t, p_0, r_0, r_t} \left[ \frac{1}{t} \sum_{i=1}^{L'} \mathbf{1}[r_t^i = M] \log p_\theta(r_0^i | p_0, r_t) \right]$$

Where  $L'$  is a dynamic length:

- SFT training examples padded with |EOS| to get to 4096 tokens



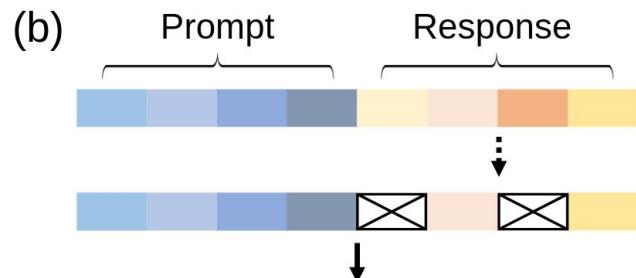
# SFT

Slightly different loss (only care about response  $r_0$  given prompt  $p_0$  and intermediate response  $r_t$ ):

$$-\mathbb{E}_{t, p_0, r_0, r_t} \left[ \frac{1}{t} \sum_{i=1}^{L'} \mathbf{1}[r_t^i = M] \log p_\theta(r_0^i | p_0, r_t) \right]$$

Where  $L'$  is a dynamic length:

- SFT training examples padded with |EOS| to get to 4096 tokens
- Model is trained to generate these |EOS| tokens (they get masked in forward process)



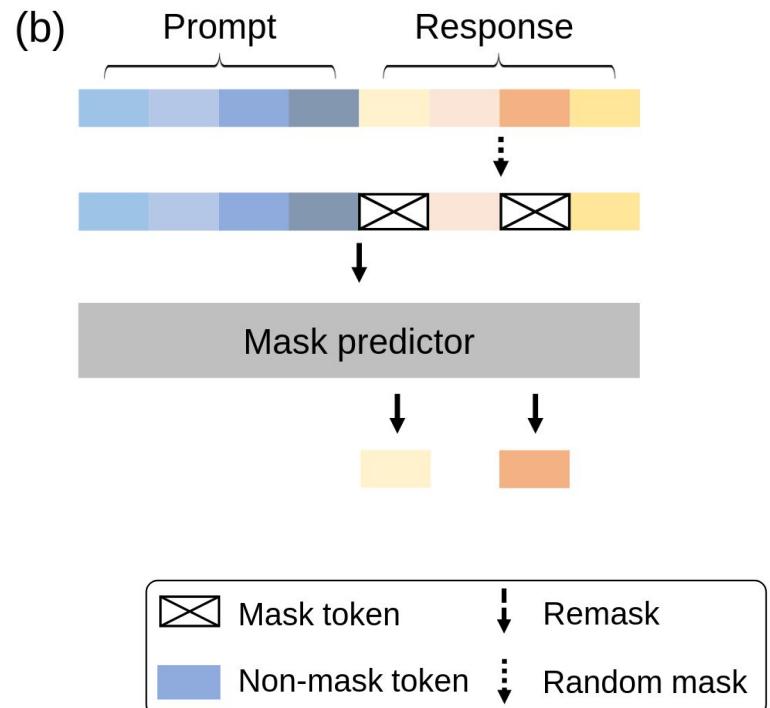
# SFT

Slightly different loss (only care about response  $r_0$  given prompt  $p_0$  and intermediate response  $r_t$ ):

$$-\mathbb{E}_{t,p_0,r_0,r_t} \left[ \frac{1}{t} \sum_{i=1}^{L'} \mathbf{1}[r_t^i = M] \log p_\theta(r_0^i | p_0, r_t) \right]$$

Where  $L'$  is a dynamic length:

- SFT training examples padded with |EOS| to get to 4096 tokens
- Model is trained to generate these |EOS| tokens (they get masked in forward process)
- When |EOS| sampled in reverse process for token  $i$ , we delete it and decrease  $L'$  by 1.



SFT

Slightly different loss (only care about response  $r_0$  given prompt  $p_0$  and intermediate response  $r_t$ ):

$$-\mathbb{E}_{t,p_0,r_0,r_t} \left[ \frac{1}{t} \sum_{i=1}^{L'} \mathbf{1}[r_t^i = \mathbf{M}] \log p_\theta(r_0^i | p_0, r_t) \right]$$

Where  $L'$  is a dynamic length:

- SFT training examples padded with |EOS| to get to 4096 tokens
  - Model is trained to generate these |EOS| tokens (they get masked in forward process)
  - When |EOS| sampled in reverse process for token  $i$ , we delete it and decrease  $L'$  by 1.

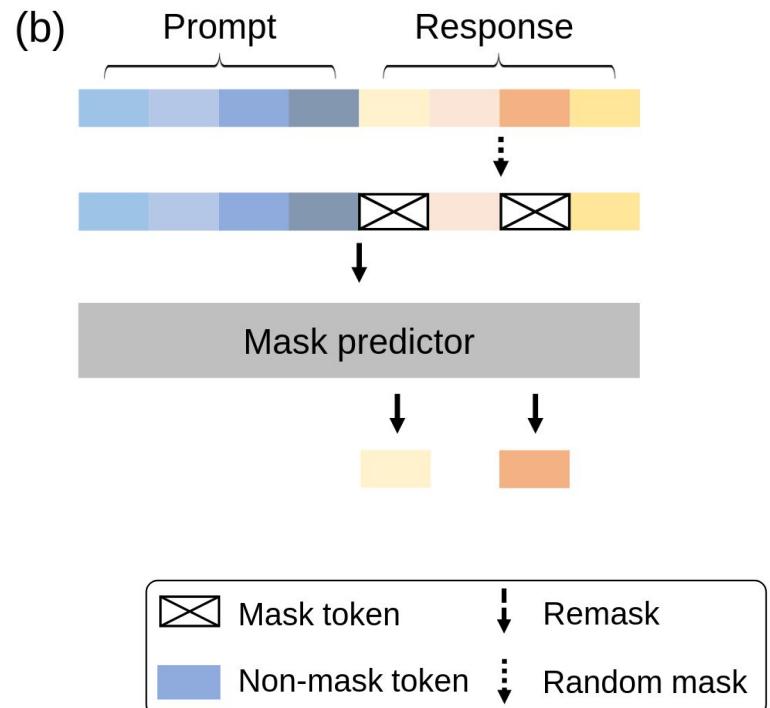
**Algorithm 2** Supervised Fine-Tuning of LLaDA

**Require:** mask predictor  $p_\theta$ , pair data distribution  $p_{\text{dat}}$

- ```

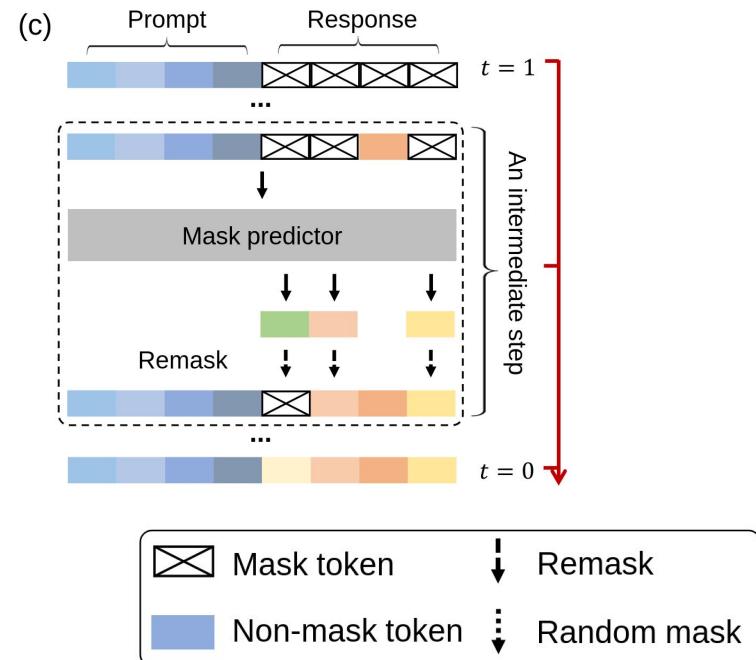
1: repeat
2:    $p_0, r_0 \sim p_{\text{data}}, t \sim U(0, 1)$  # please refer to Appendix B.1 for details on the SFT data processing.
3:    $r_t \sim q_{t|0}(r_t|r_0)$  #  $q_{t|0}$  is defined in Eq. (7)
4:   Calculate  $\mathcal{L} = -\frac{1}{t * L'} \sum_{i=1}^{L'} \mathbf{1}[r_t^i = \mathcal{M}] \log p_\theta(r_0^i | p_0, r_t)$  #  $L'$  is the sequence length of  $r_0$ 
5:   Calculate  $\nabla_\theta \mathcal{L}$  and run optimizer.
6: until Converged
7: Return  $p_\theta$ 

```



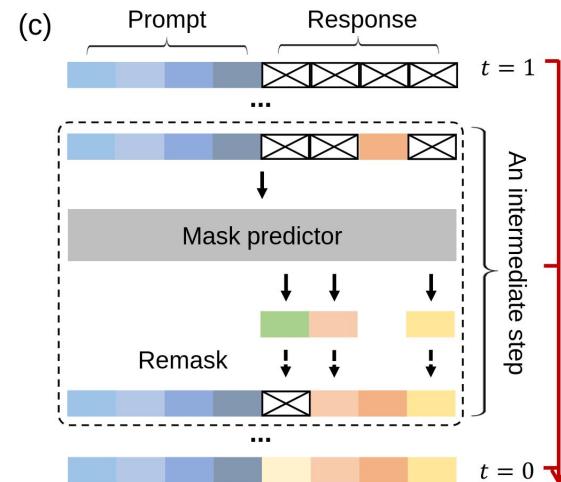
Sampling

1. Choose N sampling steps (tradeoff between efficiency and sample quality) between 1 and 0.



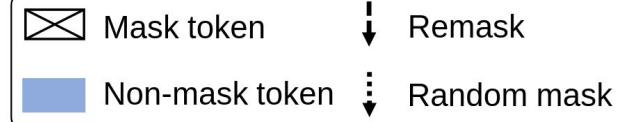
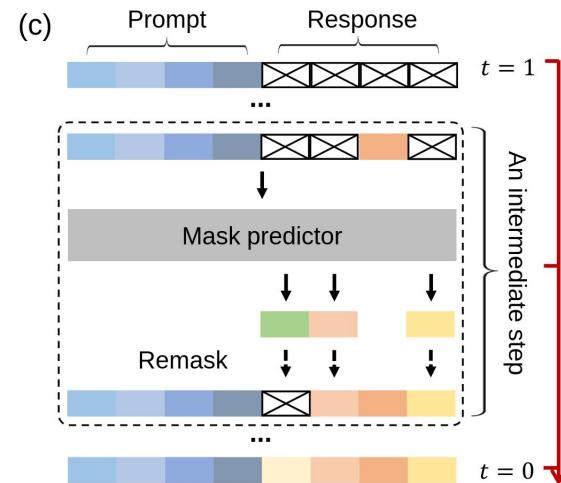
Sampling

1. Choose N sampling steps (tradeoff between efficiency and sample quality) between 1 and 0.
2. For each $t \in \{1, 1 - 1/N, \dots, 1/N\}$, predict x_t then (“at time $s = t - 1/N$ ”), remask the tokens with probability $\frac{s}{t}$.



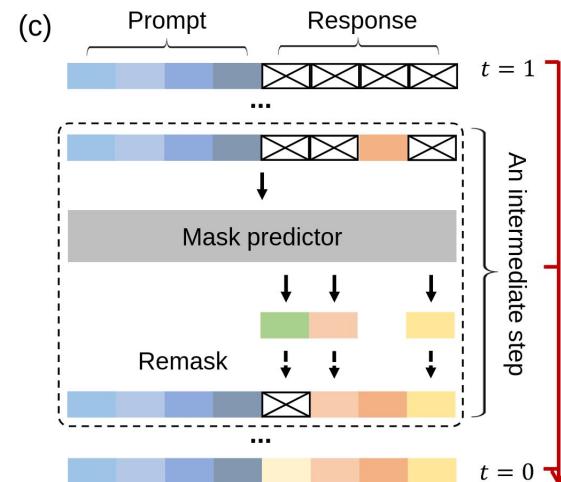
Sampling

1. Choose N sampling steps (tradeoff between efficiency and sample quality) between 1 and 0.
2. For each $t \in \{1, 1 - 1/N, \dots, 1/N\}$, predict x_t then (“at time $s = t - 1/N$ ”), remask the tokens with probability $\frac{s}{t}$.
 - a. Either remask each token independently



Sampling

1. Choose N sampling steps (tradeoff between efficiency and sample quality) between 1 and 0.
2. For each $t \in \{1, 1 - 1/N, \dots, 1/N\}$, predict x_t then (“at time $s = t - 1/N$ ”), remask the tokens with probability $\frac{s}{t}$.
 - a. Either remask each token independently
 - b. Or remask the $L \frac{s}{t}$ tokens with lowest logits.



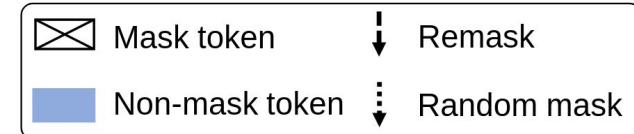
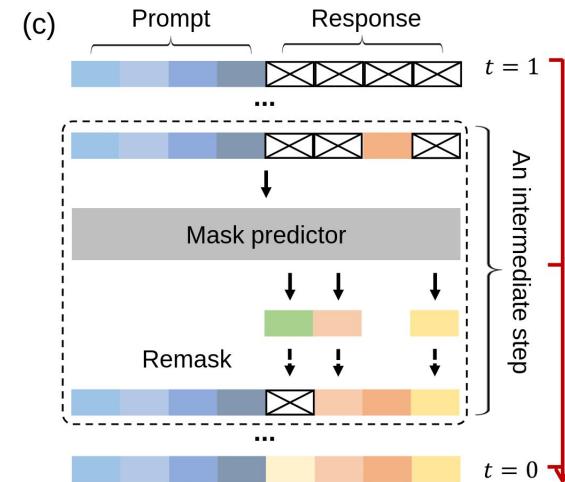
Sampling

1. Choose N sampling steps (tradeoff between efficiency and sample quality) between 1 and 0.
2. For each $t \in \{1, 1 - 1/N, \dots, 1/N\}$, predict x_t then (“at time $s = t - 1/N$ ”), remask the tokens with probability $\frac{s}{t}$.
 - a. Either remask each token independently
 - b. Or remask the $L \frac{s}{t}$ tokens with lowest logits.

	LLaDA 8B Base	LLaDA 8B Instruct
--	---------------	-------------------

Randomly remasking	52.3	72.0
Lowest confidence remasking	64.7	12.9
Lowest confidence & semi-autoregressive remasking	64.4	73.8

(on GSM8K)



Sampling

1. Choose N sampling steps (tradeoff between efficiency and sample quality) between 1 and 0.
2. For each $t \in \{1, 1 - 1/N, \dots, 1/N\}$, predict x_t then (“at time $s = t - 1/N$ ”), remask the tokens with probability $\frac{s}{t}$.
 - a. Either remask each token independently
 - b. Or remask the $L \frac{s}{t}$ tokens with lowest logits.

	LLaDA 8B Base	LLaDA 8B Instruct
Randomly remasking	52.3	72.0
Lowest confidence remasking	64.7	12.9
Lowest confidence & semi-autoregressive remasking	64.4	73.8

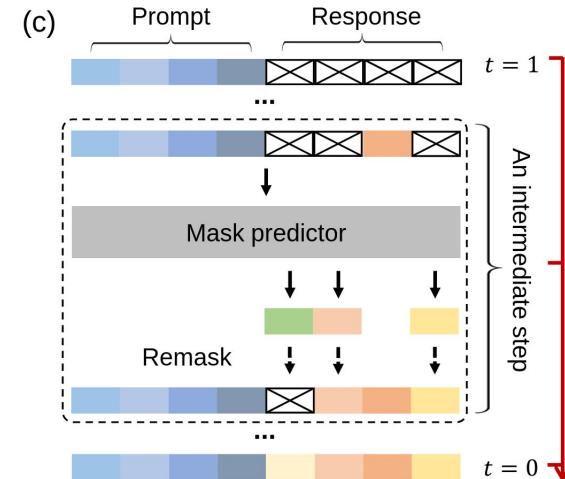
(on GSM8K)

Algorithm 4 Reverse Process of LLaDA

```

Require: mask predictor  $p_\theta$ , prompt  $p_0$ , answer length  $L$ , sampling steps  $N$ 
1: Set  $r_1$  is a fully masked sequence of length  $L$ .
2: for  $t \leftarrow 1$  down to  $\frac{1}{N}$  step  $\frac{1}{N}$  do
3:    $s = t - \frac{1}{N}$ 
4:    $r_0 = \arg \max_{r_0} p_\theta(r_0 | p_0, r_t)$                                 # we employ greedy sampling when predicting masked tokens
5:   for  $i \leftarrow 1$  to  $L$  do
6:     if  $r_t^i \neq M$  then
7:        $r_0^i = r_t^i$ 
8:     else
9:       With probability  $\frac{s}{t}$ ,  $r_0^i$  is set to M
10:    end if
11:   end for
12:    $r_s = r_0$ 
13: end for
14: Return  $r_0$ 

```



Sampling

1. Choose N sampling steps (tradeoff between efficiency and sample quality) between 1 and 0.
2. For each $t \in \{1, 1 - 1/N, \dots, 1/N\}$, predict x_t then (“at time $s = t - 1/N$ ”), remask the tokens with probability $\frac{s}{t}$.
 - a. Either remask each token independently
 - b. Or remask the $L \frac{s}{t}$ tokens with lowest logits.

LLaDA 8B Base LLaDA 8B Instruct

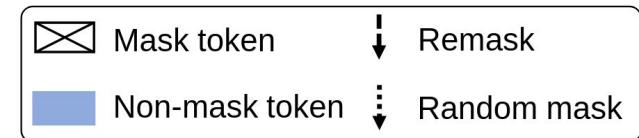
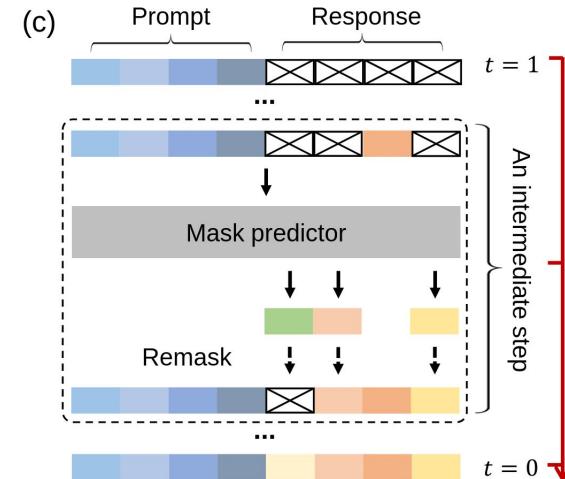
	LLaDA 8B Base	LLaDA 8B Instruct
Randomly remasking	52.3	72.0
Lowest confidence remasking	64.7	12.9
Lowest confidence & semi-autoregressive remasking	64.4	73.8
(on GSM8K)		

Algorithm 4 Reverse Process of LLaDA

```

Require: mask predictor  $p_\theta$ , prompt  $p_0$ , answer length  $L$ , sampling steps  $N$ 
1: Set  $r_1$  is a fully masked sequence of length  $L$ .
2: for  $t \leftarrow 1$  down to  $\frac{1}{N}$  step  $\frac{1}{N}$  do
3:    $s = t - \frac{1}{N}$ 
4:    $r_0 = \arg \max_{r_0} p_\theta(r_0 | p_0, r_t)$                                 # we employ greedy sampling when predicting masked tokens
5:   for  $i \leftarrow 1$  to  $L$  do
6:     if  $r_t^i \neq M$  then
7:        $r_0^i = r_t^i$ 
8:     else
9:       With probability  $\frac{s}{t}$ ,  $r_0^i$  is set to M
10:    end if
11:   end for
12:    $r_s = r_0$ 
13: end for
14: Return  $r_0$ 

```



Semi-autoregressive Sampling

Additionally, split the response up into several blocks and generate those from left to right.

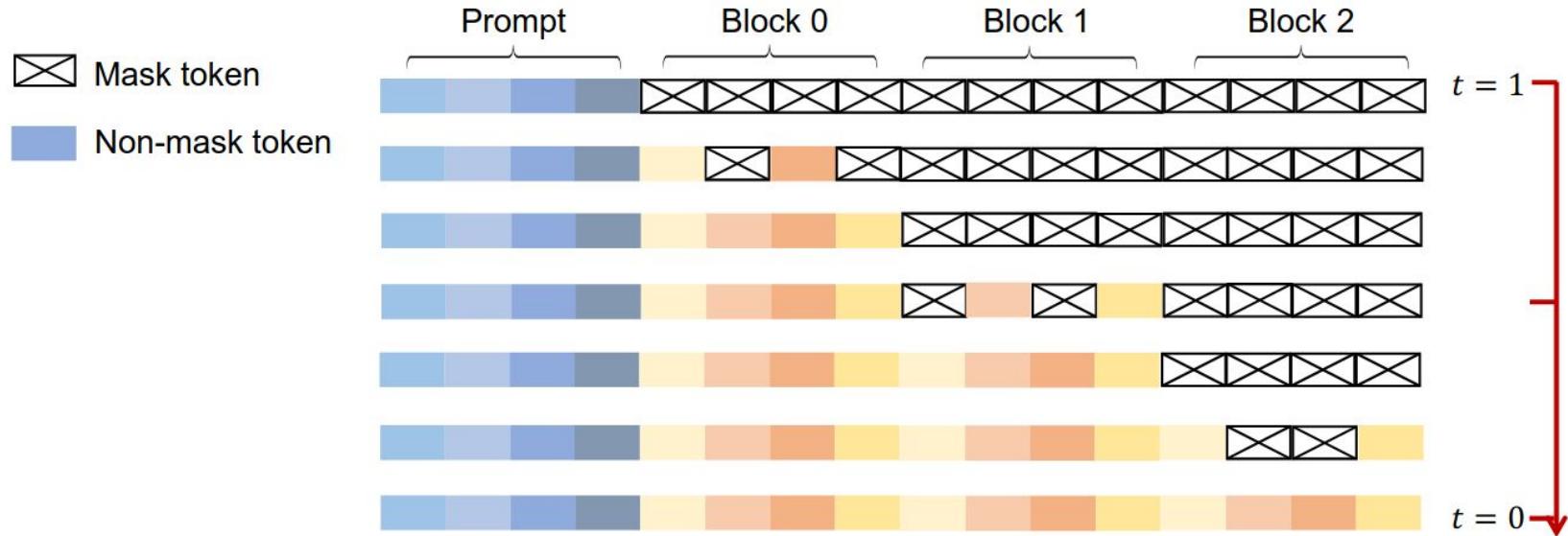


Figure 4. A Conceptual Overview of the Semi-autoregressive Sampling.

But the space of possible sampling strategies is HUGE



2025-7-4

Train for the Worst, Plan for the Best: Understanding Token Ordering in Masked Diffusions

Jaeyeon Kim ^{*}1 Kulin Shah ^{*}2 Vasilis Kontonis ² Sham Kakade ¹ Sitan Chen ¹

PC-Sampler: Position-Aware Calibration of Decoding Bias in Masked Diffusion Models

Pengcheng Huang^{1*}, Shuhao Liu^{1*}, Zhenghao Liu¹, Yukun Yan²,
Shuo Wang², Zulong Chen³, Tong Xiao¹

¹Department of Computer Science and Technology, Northeastern University, Shenyang, China

²Department of Computer Science and Technology, Institute for AI, Tsinghua University, Beijing, China

³Alibaba Group, Hangzhou, China

MASKED DIFFUSION MODELS ARE SECRETLY AGNOSTIC MASKED MODELS AND EXPLOIT INACCURATE CATEGORICAL SAMPLING

Kaiwen Zheng^{1*}, Yongxin Chen², Hanzi Mao², Ming-Yu Liu², Jun Zhu^{1†}, Qinsheng Zhang²

¹Department of Computer Science & Technology, Institute for AI, Tsinghua University

²NVIDIA

Fast-dLLM: Training-free Acceleration of Diffusion LLM by Enabling KV Cache and Parallel Decoding

Chengyue Wu^{1,2*} Hao Zhang^{2*} Shuchen Xue⁴ Zhijian Liu² Shizhe Diao² Ligeng Zhu²
Ping Luo¹ Song Han^{2,3} Enze Xie²

Accelerated Sampling from Masked Diffusion Models via Entropy Bounded Unmasking

Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas Nolte, Brian Karrer

FAIR at Meta

Unifying Autoregressive and Diffusion-Based Sequence Generation

Nima Fathi,^{*}Torsten Scholak & Pierre-André Noël
ServiceNow Research
nima.fathi@mila.quebec, {torsten.scholak,pierre-andre.noel}@servicenow.com

Simplified and Generalized Masked Diffusion for Discrete Data

Jiaxin Shi*, Kehang Han*, Zhe Wang, Arnaud Doucet, Michalis K. Titsias
Google DeepMind

Strategy 0: IID per token (e.g. MD4)

Jiaxin Shi*, Kehang Han*, Zhe Wang, Arnaud Doucet, Michalis K. Titsias
Google DeepMind

- Go from all-noise at $t = 1$ to noiseless at $t = 0$.

Strategy 0: IID per token (e.g. MD4)

Jiaxin Shi*, Kehang Han*, Zhe Wang, Arnaud Doucet, Michalis K. Titsias
Google DeepMind

- Go from all-noise at $t = 1$ to noiseless at $t = 0$.
- At time $t \in [0, 1]$ we will unmask token x_i^t at index i with probability α_t .

Strategy 0: IID per token (e.g. MD4)

Jiaxin Shi*, Kehang Han*, Zhe Wang, Arnaud Doucet, Michalis K. Titsias
Google DeepMind

- Go from all-noise at $t = 1$ to noiseless at $t = 0$.
- At time $t \in [0, 1]$ we will unmask token x_i^t at index i with probability α_t .
- For this token, we sample the new non-[MASK] token value using mask-prediction probabilities $p_\theta(x_i^t) \in [0, 1]^{|\mathcal{V}|}$.

Strategy 0: IID per token (e.g. MD4)

Jiaxin Shi*, Kehang Han*, Zhe Wang, Arnaud Doucet, Michalis K. Titsias
Google DeepMind

- Go from all-noise at $t = 1$ to noiseless at $t = 0$.
- At time $t \in [0, 1]$ we will unmask token x_i^t at index i with probability α_t .
- For this token, we sample the new non-[MASK] token value using mask-prediction probabilities $p_\theta(x_i^t) \in [0, 1]^{|\mathcal{V}|}$.
- Many options are available for the schedule α_t .

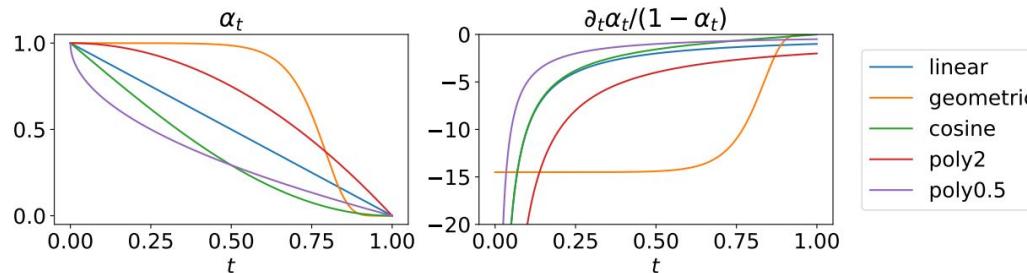


Figure 1: Masking schedules in the literature: (Left) α_t ; (Right) weight of the cross-entropy loss w.r.t. t ; Equations for these schedules are given in Tab. 4 in Appendix.

Strategy 1: Greedy/Top probability sampling

To unmask $K \in \mathbb{N}$ tokens in a step from time t to time s (often $K = \text{round}(N \frac{\alpha_t}{\alpha_s})$) :

Strategy 1: Greedy/Top probability sampling

To unmask $K \in \mathbb{N}$ tokens in a step from time t to time s (often $K = \text{round}(N \frac{\alpha_t}{\alpha_s})$) :

- Assign a confidence to each index i given by greatest token-probability:

$$c_i = \max_{v \in \mathcal{V}} p_\theta(x_i^t)^{(v)}$$

Strategy 1: Greedy/Top probability sampling

To unmask $K \in \mathbb{N}$ tokens in a step from time t to time s (often $K = \text{round}(N \frac{\alpha_t}{\alpha_s})$) :

- Assign a confidence to each index i given by greatest token-probability:

$$c_i = \max_{v \in \mathcal{V}} p_\theta(x_i^t)^{(v)}$$

- Order by confidence: $c_{(i)} = i$ th greatest confidence

Strategy 1: Greedy/Top probability sampling

To unmask $K \in \mathbb{N}$ tokens in a step from time t to time s (often $K = \text{round}(N \frac{\alpha_t}{\alpha_s})$) :

- Assign a confidence to each index i given by greatest token-probability:

$$c_i = \max_{v \in \mathcal{V}} p_\theta(x_i^t)^{(v)}$$

- Order by confidence: $c_{(i)} = i$ th greatest confidence
- Select the top K indices to decode/sample in decreasing order of $c_{(i)}$.

Strategy 2: Top Probability Margin

Similar to Top Probability (Strategy 1), except use probability margin for confidence score:

$$c_{(i)} = p_\theta(x_i^t)^{(v_1)} - p_\theta(x_i^t)^{(v_2)}$$

Where

$$v_1 = \arg \max_{v \in \mathcal{V}} p_\theta(x_i^t)^{(v)} \quad v_2 = \arg \max_{v \in \mathcal{V} \setminus \{v_1\}} p_\theta(x_i^t)^{(v)}$$

I.e. the difference between the two highest probabilities for a given token index.

Strategy 2: Top Probability Margin

Similar to Top Probability (Strategy 1), except use probability margin for confidence score:

$$c_{(i)} = p_\theta(x_i^t)^{(v_1)} - p_\theta(x_i^t)^{(v_2)}$$

Where

$$v_1 = \arg \max_{v \in \mathcal{V}} p_\theta(x_i^t)^{(v)} \quad v_2 = \arg \max_{v \in \mathcal{V} \setminus \{v_1\}} p_\theta(x_i^t)^{(v)}$$

I.e. the difference between the two highest probabilities for a given token index.

Select top K of indices according to $c_{(i)}$

Strategy 2: Top Probability Margin

Similar to Top Probability (Strategy 1), except use probability margin for confidence score:

$$c_{(i)} = p_\theta(x_i^t)^{(v_1)} - p_\theta(x_i^t)^{(v_2)}$$

Where

$$v_1 = \arg \max_{v \in \mathcal{V}} p_\theta(x_i^t)^{(v)} \quad v_2 = \arg \max_{v \in \mathcal{V} \setminus \{v_1\}} p_\theta(x_i^t)^{(v)}$$

I.e. the difference between the two highest probabilities for a given token index.

Select top K of indices according to $c_{(i)}$

Helps to avoid prematurely unmasking at an index where two token values both have high probability.

Strategy 2: Top Probability Margin

Similar to Top Probability (Strategy 1), except use probability margin for confidence score:

$$c_{(i)} = p_\theta(x_i^t)^{(v_1)} - p_\theta(x_i^t)^{(v_2)}$$

Where

$$v_1 = \arg \max_{v \in \mathcal{V}} p_\theta(x_i^t)^{(v)} \quad v_2 = \arg \max_{v \in \mathcal{V} \setminus \{v_1\}} p_\theta(x_i^t)^{(v)}$$

I.e. the difference between the two highest probabilities for a given token index.

Select top K of indices according to $c_{(i)}$

Helps to avoid prematurely unmasking at an index where two token values both have high probability.

Jaeyeon Kim ^{*1} Kulin Shah ^{*2} Vasilis Kontonis ² Sham Kakade ¹ Sitan Chen ¹

Table 2. Comparison of accuracy for solving the Sudoku puzzle.

Method	# Param	Accuracy
ARM (w/o ordering)		9.73%
ARM (with ordering)	42M	87.18%
MDM (vanilla)		6.88%
MDM (Top probability)	6M	18.51%
MDM (Top prob. margin)		89.49%

Table 3. Comparison of accuracy for solving the Zebra puzzle.

Method	# Param	Accuracy
ARM (w/o ordering)		80.31 %
ARM (with ordering)	42M	91.17 %
MDM (vanilla)		76.9 %
MDM (Top probability)	19M	98.5 %
MDM (Top prob. margin)		98.3 %

Table 4. Performance of different inference strategies for LLaDa 8B model on coding and math tasks.

Method	HumanEval-Single	HumanEval-Multi	HumanEval-Split	Math	MMLU	ROCStories
Vanilla	31.8%	16.5%	14.2%	28.5%	33.2%	21.23%
Top probability	32.9%	20.8%	18.4%	31.3%	36.5%	21.10%
Top prob. margin	33.5%	25.4%	22.3%	34.3%	35.4%	21.41%

Strategy 3: Confidence-Aware Parallel Decoding

Unmask tokens that have a highest-token probability above a global threshold.

Fast-dLLM: Training-free Acceleration of Diffusion LLM by Enabling KV Cache and Parallel Decoding

Chengyue Wu^{1,2*} Hao Zhang^{2*} Shuchen Xue⁴ Zhijian Liu² Shizhe Diao² Ligeng Zhu²
Ping Luo¹ Song Han^{2,3} Enze Xie²

Strategy 3: Confidence-Aware Parallel Decoding

Unmask tokens that have a highest-token probability above a global threshold.

Like top probability, let $c_i = \max_{v \in \mathcal{V}} p_\theta(x_i^t)^{(v)}$ and
let $c_{(i)}$ be the i -th largest value of c_i .

Fast-dLLM: Training-free Acceleration of Diffusion LLM by Enabling KV Cache and Parallel Decoding

Chengyue Wu^{1,2*} Hao Zhang^{2*} Shuchen Xue⁴ Zhijian Liu² Shizhe Diao² Ligeng Zhu²
Ping Luo¹ Song Han^{2,3} Enze Xie²

Strategy 3: Confidence-Aware Parallel Decoding

Unmask tokens that have a highest-token probability above a global threshold.

Like top probability, let $c_i = \max_{v \in \mathcal{V}} p_\theta(x_i^t)^{(v)}$ and

let $c_{(i)}$ be the i -th largest value of c_i .

Unmask all tokens $1, \dots, n$ where

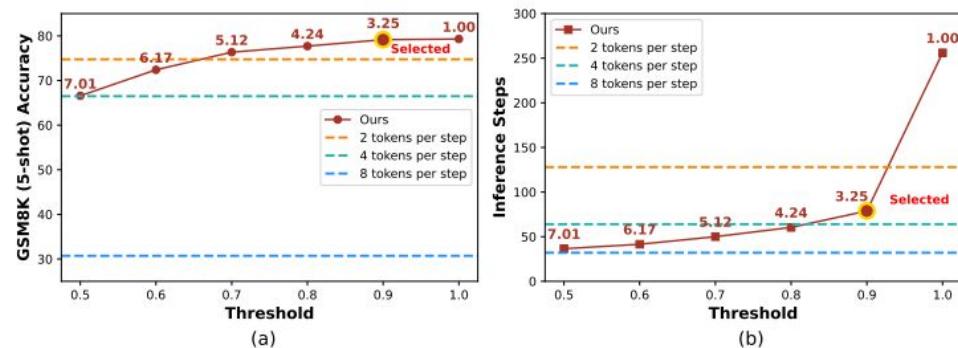
$$(n + 1)(1 - c_{(n)}) < f$$

for some global threshold f

(they test $f \in [0.5, 1]$).

Fast-dLLM: Training-free Acceleration of Diffusion LLM by Enabling KV Cache and Parallel Decoding

Chengyue Wu^{1,2*} Hao Zhang^{2*} Shuchen Xue⁴ Zhijian Liu² Shizhe Diao² Ligeng Zhu²
Ping Luo¹ Song Han^{2,3} Enze Xie²



Strategy 4: PC Samplers (Position-Aware Calibration)

Weight the confidence score as $s_t^i = w^i c_t^i$, for timestep t and
(currently masked) index i .

**Pengcheng Huang^{1*}, Shuhao Liu^{1*}, Zhenghao Liu¹, Yukun Yan²,
Shuo Wang², Zulong Chen³, Tong Xiao¹**

¹Department of Computer Science and Technology, Northeastern University, Shenyang, China

²Department of Computer Science and Technology, Institute for AI, Tsinghua University, Beijing, China

³Alibaba Group, Hangzhou, China

Strategy 4: PC Samplers (Position-Aware Calibration)

Weight the confidence score as $s_t^i = w^i c_t^i$, for timestep t and (currently masked) index i .

Write $w^i = e^{-\lambda i}$ and $c_i^t = \min(-\max_{v \in \mathcal{V}} p_\theta(x_i^t)^{(v)} \log p_{\mathcal{D}'}(v), c_{\text{MAX}})$ for $\lambda \geq 0$ and where $p_{\mathcal{D}'}(v)$ is the prob. of seeing token v in some large corpus \mathcal{D}'

**Pengcheng Huang^{1*}, Shuhao Liu^{1*}, Zhenghao Liu¹, Yukun Yan²,
Shuo Wang², Zulong Chen³, Tong Xiao¹**

¹Department of Computer Science and Technology, Northeastern University, Shenyang, China

²Department of Computer Science and Technology, Institute for AI, Tsinghua University, Beijing, China

³Alibaba Group, Hangzhou, China

Strategy 4: PC Samplers (Position-Aware Calibration)

Weight the confidence score as $s_t^i = w^i c_t^i$, for timestep t and (currently masked) index i .

Write $w^i = e^{-\lambda i}$ and $c_i^t = \min(-\max_{v \in \mathcal{V}} p_\theta(x_i^t)^{(v)} \log p_{\mathcal{D}'}(v), c_{\text{MAX}})$ for $\lambda \geq 0$ and where $p_{\mathcal{D}'}(v)$ is the prob. of seeing token v in some large corpus \mathcal{D}'

- Note that large λ encourages autoregressive-like decoding
- Small λ encourages more flexible orderings for decoding

Pengcheng Huang^{1*}, Shuhao Liu^{1*}, Zhenghao Liu¹, Yukun Yan²,
Shuo Wang², Zulong Chen³, Tong Xiao¹

¹Department of Computer Science and Technology, Northeastern University, Shenyang, China

²Department of Computer Science and Technology, Institute for AI, Tsinghua University, Beijing, China

³Alibaba Group, Hangzhou, China

Strategy 4: PC Samplers (Position-Aware Calibration)

Weight the confidence score as $s_t^i = w^i c_t^i$, for timestep t and (currently masked) index i .

Write $w^i = e^{-\lambda i}$ and $c_i^t = \min(-\max_{v \in \mathcal{V}} p_\theta(x_i^t)^{(v)} \log p_{\mathcal{D}'}(v), c_{\text{MAX}})$ for $\lambda \geq 0$ and where $p_{\mathcal{D}'}(v)$ is the prob. of seeing token v in some large corpus \mathcal{D}'

- Note that large λ encourages autoregressive-like decoding
- Small λ encourages more flexible orderings for decoding
- They hand-select some $\lambda \in [0, 1]$ depending on the task

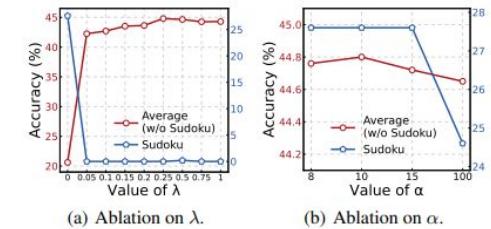


Figure 4: Ablation studies on the two key hyperparameters in our approach: (a) the positional decay coefficient λ and (b) the clipping threshold α . To analyze task-dependent effects, we split the evaluation into two groups: the blue line shows the average performance across all datasets except Sudoku, while the orange line shows the performance specifically on Sudoku.

$$(\alpha := c_{\text{MAX}})$$

Strategy 4: PC Samplers (Position-Aware Calibration)

Weight the confidence score as $s_t^i = w^i c_t^i$, for timestep t and (currently masked) index i .

Write $w^i = e^{-\lambda i}$ and $c_i^t = \min(-\max_{v \in \mathcal{V}} p_\theta(x_i^t)^{(v)} \log p_{\mathcal{D}'}(v), c_{\text{MAX}})$ for $\lambda \geq 0$ and where $p_{\mathcal{D}'}(v)$ is the prob. of seeing token v in some large corpus \mathcal{D}'

- Note that large λ encourages autoregressive-like decoding
- Small λ encourages more flexible orderings for decoding
- They hand-select some $\lambda \in [0, 1]$ depending on the task
- imo this is a wasted opportunity to learn the λ values in a more interesting way...

Pengcheng Huang^{1*}, Shuhao Liu^{1*}, Zhenghao Liu¹, Yukun Yan²,
Shuo Wang², Zulong Chen³, Tong Xiao¹

¹Department of Computer Science and Technology, Northeastern University, Shenyang, China

²Department of Computer Science and Technology, Institute for A.I., Tsinghua University, Beijing, China

³Alibaba Group, Hangzhou, China

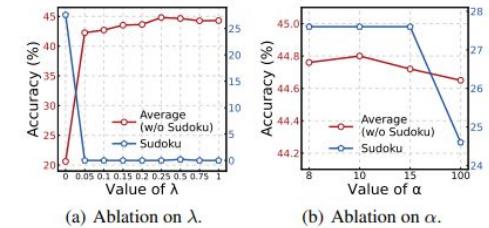
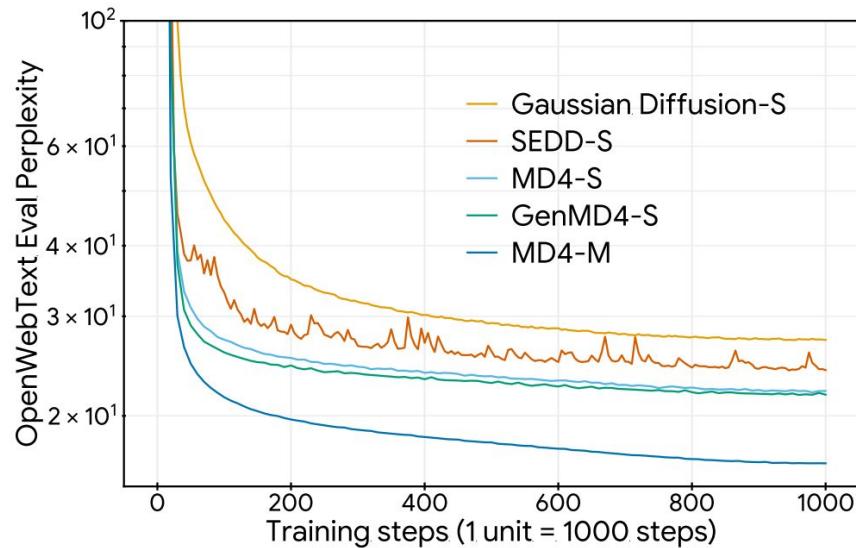


Figure 4: Ablation studies on the two key hyperparameters in our approach: (a) the positional decay coefficient λ and (b) the clipping threshold α . To analyze task-dependent effects, we split the evaluation into two groups: the blue line shows the average performance across all datasets except Sudoku, while the orange line shows the performance specifically on Sudoku.

$$(\alpha := c_{\text{MAX}})$$

Strategy 4.1: GenMD4

For each token position i , use a denoising schedule/probability $\alpha_{t,i} = 1 - t^{w_i}$ where we learn w_i per index via gradient descent.

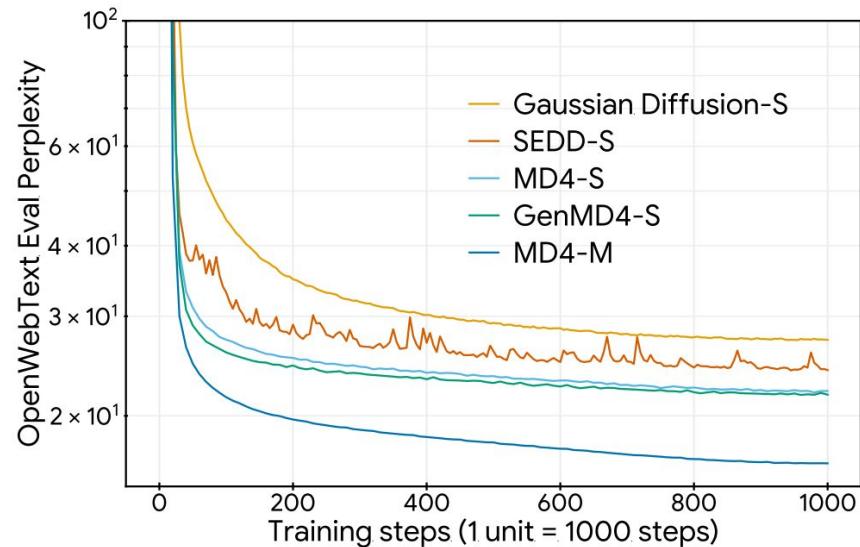


Jiaxin Shi*, Kehang Han*, Zhe Wang, Arnaud Doucet, Michalis K. Titsias
Google DeepMind

Strategy 4.1: GenMD4

For each token position i , use a denoising schedule/probability $\alpha_{t,i} = 1 - t^{w_i}$ where we learn w_i per index via gradient descent.

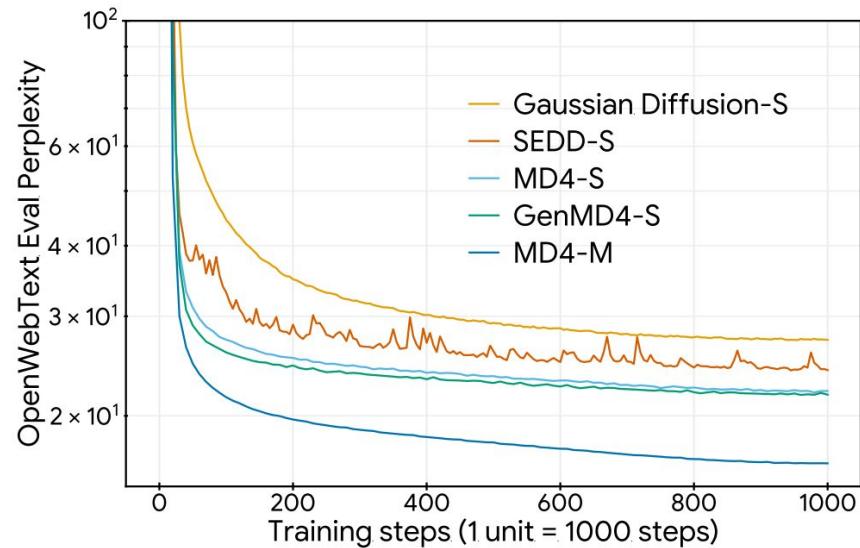
- Difficult to optimise
 - Use reinforce leave-one-out



Strategy 4.1: GenMD4

For each token position i , use a denoising schedule/probability $\alpha_{t,i} = 1 - t^{w_i}$ where we learn w_i per index via gradient descent.

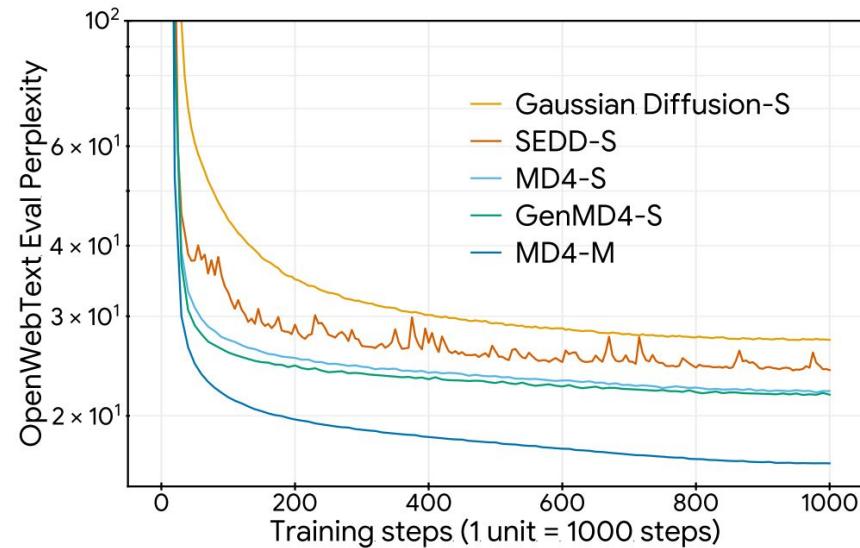
- Difficult to optimise
 - Use reinforce leave-one-out
- “Easily overfits”



Strategy 4.1: GenMD4

For each token position i , use a denoising schedule/probability $\alpha_{t,i} = 1 - t^{w_i}$ where we learn w_i per index via gradient descent.

- Difficult to optimise
 - Use reinforce leave-one-out
- “Easily overfits”
- Doesn’t seem like a feasible solution long-term



Strategy 4.2: Hyperschedules

Nima Fathi, Torsten Scholak & Pierre-André Noël

ServiceNow Research

nima.fathi@mila.quebec, {torsten.scholak,pierre-andre.noel}@servicenow.com

Unifying autoregression and diffusion: basically just set (by hand) some patterns for $\alpha_t =: \tau_t$ depending on the token index to encourage decoding to behave more or less autoregressively (depending on the task).

Strategy 4.2: Hyperschedules

Nima Fathi, Torsten Scholak & Pierre-André Noël

ServiceNow Research

nima.fathi@mila.quebec, {torsten.scholak,pierre-andre.noel}@servicenow.com

Unifying autoregression and diffusion: basically just set (by hand) some patterns for $\alpha_t =: \tau_t$ depending on the token index to encourage decoding to behave more or less autoregressively (depending on the task).

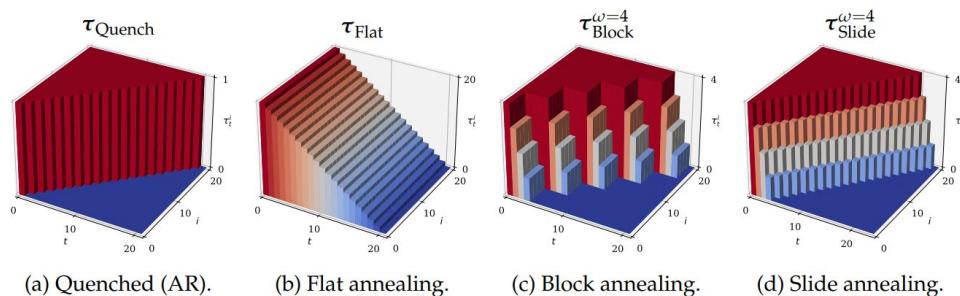


Figure 2: We introduce τ -hyperschedules, subjecting different token positions i with different noise levels (red high; blue low) at different generation step t . (a) Standard AR models (e.g., GPT) determine tokens one by one, “quenching” each of them to full determination in a single step. They may thus be construed as an extreme case of a diffusion model. (b) Standard diffusion models (e.g., SEDD) gradually anneal all tokens independently of their position. (c) Block-wise application of flat annealing, here for blocks of width $\omega = 4$. (d) Annealing with a sliding window (“smoothed” AR), here using window width $\omega = 4$. These last two examples share important features of both AR and diffusion models.

Strategy 4.2: Hyperschedules

Nima Fathi, Torsten Scholak & Pierre-André Noël

ServiceNow Research

nima.fathi@mila.quebec, {torsten.scholak,pierre-andre.noel}@servicenow.com

Unifying autoregression and diffusion: basically just set (by hand) some patterns for $\alpha_t =: \tau_t$ depending on the token index to encourage decoding to behave more or less autoregressively (depending on the task).

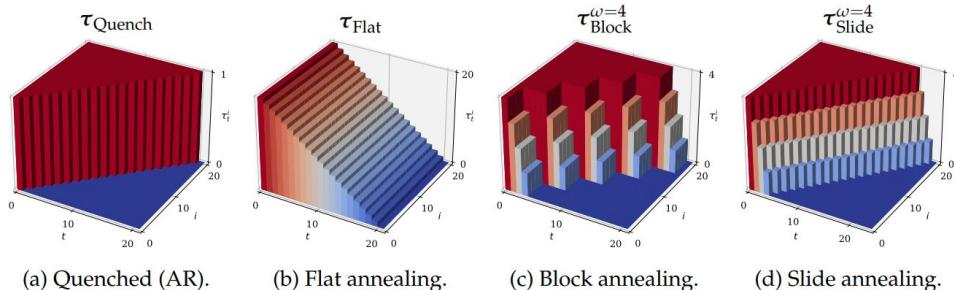


Figure 2: We introduce τ -hyperschedules, subjecting different token positions i with different noise levels (red high; blue low) at different generation step t . (a) Standard AR models (e.g., GPT) determine tokens one by one, “quenching” each of them to full determination in a single step. They may thus be construed as an extreme case of a diffusion model. (b) Standard diffusion models (e.g., SEDD) gradually anneal all tokens independently of their position. (c) Block-wise application of flat annealing, here for blocks of width $\omega = 4$. (d) Annealing with a sliding window (“smoothed” AR), here using window width $\omega = 4$. These last two examples share important features of both AR and diffusion models.

		Parameters	PPL (\downarrow)
Autoregressive	OmniNet _T (Tay et al., 2021)	100M	21.5
	Transformer (65B tokens) (Sahoo et al., 2024) [†]	110M	22.3
Diffusion	SEDD (65B tokens) (Lou et al., 2024)	110M	32.8
	MDLM (65B tokens) (Sahoo et al., 2024)	110M	31.8
	BD3-LMs $L' = 4$ (65B tokens) (Arriola et al., 2025)	110M	28.2
γ -Hybrid (Ours)	$[\gamma = 0.02, \tau_{\text{Flat}, \text{ALIGNED}}^{t=0 \dots t=64}]$ (56B)	110M	27.8
	$[\gamma = 0.02, \tau_{\text{Flat}, \text{SHIFTED}}^{t=0 \dots t=64}]$ (56B)	110M	28.3
γ -Hybrid (Ours)	$[\gamma = 0.02, \tau_{\text{Block}, \text{ALIGNED}}^{t=0 \dots t=64}]$ (65B)	110M	27.1
	$[\gamma = 0.02, \tau_{\text{Block}, \text{SHIFTED}}^{t=0 \dots t=64}]$ (65B)	110M	27.0
	$[\gamma = 0.02, \tau_{\text{Block}, \text{ALIGNED}}^{t=0 \dots t=4}]$ (65B)	110M	27.5
	$[\gamma = 0.02, \tau_{\text{Block}, \text{SHIFTED}}^{t=0 \dots t=4}]$ (65B)	110M	26.6

Table 2: Test perplexities (PPL; \downarrow) on LM1B. Perplexity values for diffusion models are upper-bound estimations. [†]Reported in He et al. (2022). [‡]Reported in Sahoo et al. (2024). Best diffusion value is bolded.

Strategy 5: Entropy Bound Samplers

Unmask the largest subset of token indices $r \subseteq [N]$ such that

$$\sum_{i \in r} H(p_\theta(x_i^t)) - \max_{i \in [N]} H(p_\theta(x_i^t)) \leq \gamma$$

for some $\gamma \geq 0$.

Strategy 5: Entropy Bound Samplers

Unmask the largest subset of token indices $r \subseteq [N]$ such that

$$\sum_{i \in r} H(p_\theta(x_i^t)) - \max_{i \in [N]} H(p_\theta(x_i^t)) \leq \gamma$$

for some $\gamma \geq 0$.

- $\gamma = 0$ will unmask one token at a time (with the lowest entropy prediction distribution)
- $\gamma = \infty$ will unmask all tokens in one go

Strategy 5: Entropy Bound Samplers

Unmask the largest subset of token indices $r \subseteq [N]$ such that

$$\sum_{i \in r} H(p_\theta(x_i^t)) - \max_{i \in [N]} H(p_\theta(x_i^t)) \leq \gamma$$

for some $\gamma \geq 0$.

- $\gamma = 0$ will unmask one token at a time (with the lowest entropy prediction distribution)
- $\gamma = \infty$ will unmask all tokens in one go
- They experiment with $\gamma \in \{0.1, 0.01, 0.001\}$.

Strategy 5: Entropy Bound Samplers

Unmask the largest subset of token indices $r \subseteq [N]$ such that

$$\sum_{i \in r} H(p_\theta(x_i^t)) - \max_{i \in [N]} H(p_\theta(x_i^t)) \leq \gamma$$

for some $\gamma \geq 0$.

- $\gamma = 0$ will unmask one token at a time (with the lowest entropy prediction distribution)
- $\gamma = \infty$ will unmask all tokens in one go
- They experiment with $\gamma \in \{0.1, 0.01, 0.001\}$.

Accelerated Sampling from Masked Diffusion Models via Entropy Bounded Unmasking

Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas Nolte, Brian Karrer

FAIR at Meta

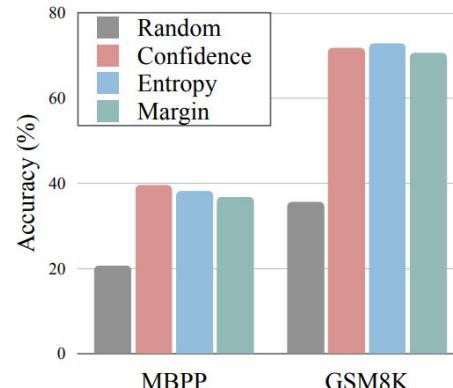
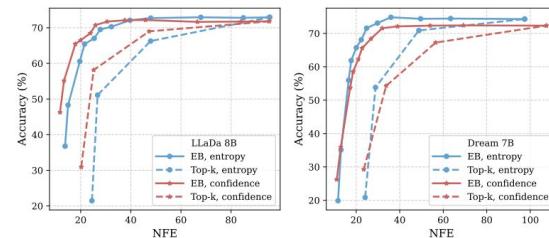


Figure 2 Performance of greedy sampling with various unmasking criteria from LLaDa 8B Base model.



Strategy 6: Learning an auxiliary index-selection model

As in the previous strategy, consider the problem as finding a subset of token indices $r \subseteq [N]$ for each denoising step (in discrete time) r_1, \dots, r_T s.t. $\cup_t r_t = [N]$ and $r_t \cap r'_{t'} = \emptyset$ for all $t \neq t'$ (i.e. r_1, \dots, r_T form a partition of $[N]$).

Strategy 6: Learning an auxiliary index-selection model

As in the previous strategy, consider the problem as finding a subset of token indices $r \subseteq [N]$ for each denoising step (in discrete time) r_1, \dots, r_T s.t. $\cup_t r_t = [N]$ and $r_t \cap r'_{t'} = \emptyset$ for all $t \neq t'$ (i.e. r_1, \dots, r_T form a partition of $[N]$).

Let's treat r_t as a random variable with approximate posterior $Q_\phi(r_{t+1}|x^t, x_{\text{data}})$ and generative distribution $P_\theta(r_{t+1}|x^t)$, then optimise with VI.

Strategy 6: Learning an auxiliary index-selection model

As in the previous strategy, consider the problem as finding a subset of token indices $r \subseteq [N]$ for each denoising step (in discrete time) r_1, \dots, r_T s.t. $\cup_t r_t = [N]$ and $r_t \cap r'_t = \emptyset$ for all $t \neq t'$ (i.e. r_1, \dots, r_T form a partition of $[N]$).

Let's treat r_t as a random variable with approximate posterior $Q_\phi(r_{t+1}|x^t, x_{\text{data}})$ and generative distribution $P_\theta(r_{t+1}|x^t)$, then optimise with VI.

Hopefully this will give us:

Strategy 6: Learning an auxiliary index-selection model

As in the previous strategy, consider the problem as finding a subset of token indices $r \subseteq [N]$ for each denoising step (in discrete time) r_1, \dots, r_T s.t. $\cup_t r_t = [N]$ and $r_t \cap r'_{t'} = \emptyset$ for all $t \neq t'$ (i.e. r_1, \dots, r_T form a partition of $[N]$).

Let's treat r_t as a random variable with approximate posterior $Q_\phi(r_{t+1}|x^t, x_{\text{data}})$ and generative distribution $P_\theta(r_{t+1}|x^t)$, then optimise with VI.

Hopefully this will give us:

- Sensible simultaneous decoding (of tokens which don't contradict each other)
 - Maybe better overall performance

Strategy 6: Learning an auxiliary index-selection model

As in the previous strategy, consider the problem as finding a subset of token indices $r \subseteq [N]$ for each denoising step (in discrete time) r_1, \dots, r_T s.t. $\cup_t r_t = [N]$ and $r_t \cap r'_{t'} = \emptyset$ for all $t \neq t'$ (i.e. r_1, \dots, r_T form a partition of $[N]$).

Let's treat r_t as a random variable with approximate posterior $Q_\phi(r_{t+1}|x^t, x_{\text{data}})$ and generative distribution $P_\theta(r_{t+1}|x^t)$, then optimise with VI.

Hopefully this will give us:

- Sensible simultaneous decoding (of tokens which don't contradict each other)
 - Maybe better overall performance
- Decoding in many fewer steps, T, than other methods
 - Hopefully a better performance-speed trade-off

Final thoughts

- I may have been too pessimistic about the future of discrete diffusion LLMs in my earlier presentation

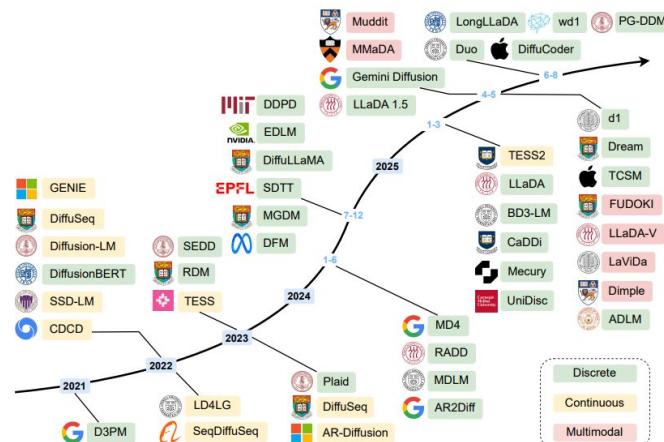


Fig. 1. Timeline of Diffusion Language Models. This figure highlights key milestones in the development of DLMs, categorized into three groups: continuous DLMs, discrete DLMs, and recent multimodal DLMs. We observe that while early research predominantly focused on continuous DLMs, discrete DLMs have gained increasing popularity in more recent years.

Final thoughts

- I may have been too pessimistic about the future of discrete diffusion LLMs in my earlier presentation
- I'm still not certain if they'll 'beat' ARMs (but it's not impossible)

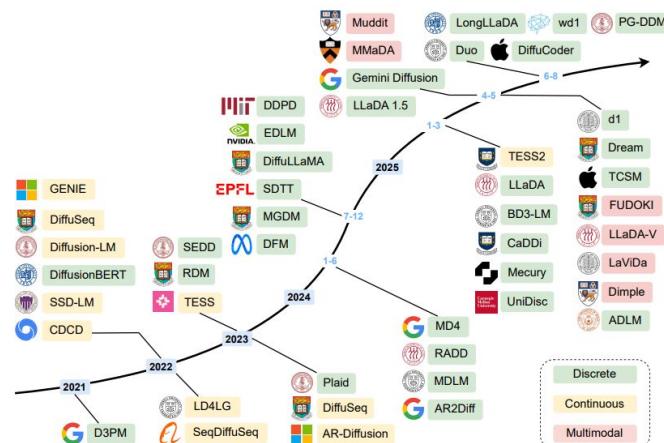


Fig. 1. Timeline of Diffusion Language Models. This figure highlights key milestones in the development of DLMs, categorized into three groups: continuous DLMs, discrete DLMs, and recent multimodal DLMs. We observe that while early research predominantly focused on continuous DLMs, discrete DLMs have gained increasing popularity in more recent years.

Final thoughts

- I may have been too pessimistic about the future of discrete diffusion LLMs in my earlier presentation
- I'm still not certain if they'll 'beat' ARMs (but it's not impossible)
- But I do still think they're fun

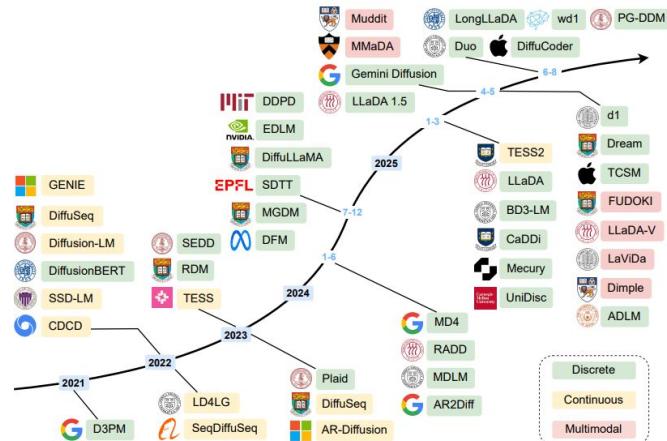
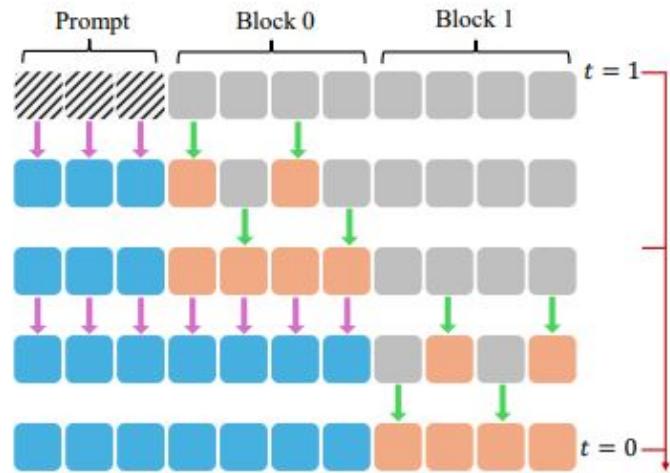


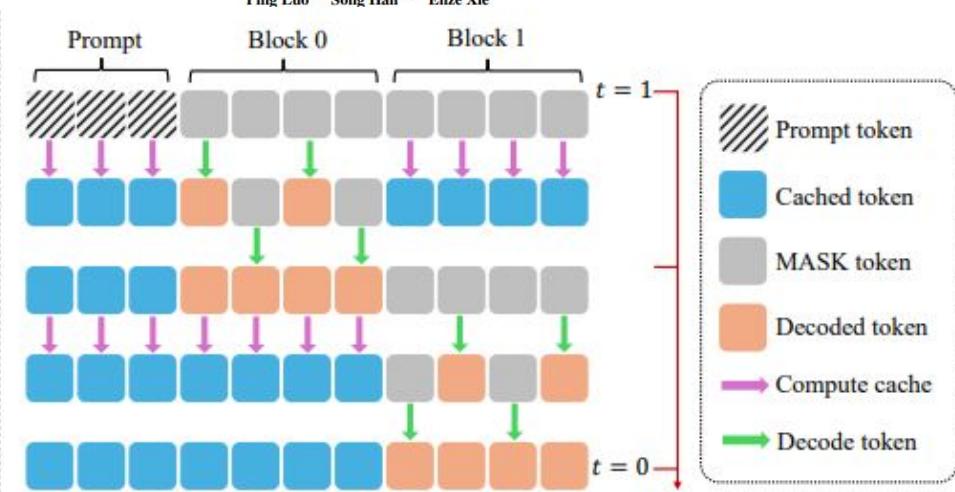
Fig. 1. Timeline of Diffusion Language Models. This figure highlights key milestones in the development of DLMs, categorized into three groups: continuous DLMs, discrete DLMs, and recent multimodal DLMs. We observe that while early research predominantly focused on continuous DLMs, discrete DLMs have gained increasing popularity in more recent years.



Bonus: Blockwise KV-Caching



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



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

Figure 2 | Illustration of our Key-Value Cache for Block-Wise Decoding. (a) During prefix-only caching, the KV cache is computed once for the prompt and reused across multiple decoding steps within each block. The cache is updated after completing a block to maintain consistency, with negligible overhead. (b) DualCache extends this approach by caching both prefix and masked suffix tokens, further accelerating decoding. The high similarity of KV activations across steps allows effective reuse with minimal approximation error.

Fast-dLLM: Training-free Acceleration of Diffusion LLM by Enabling KV Cache and Parallel Decoding

Chengyue Wu^{1,2*} Hao Zhang^{2*} Shuchen Xue⁴ Zhijian Liu² Shizhe Diao² Ligeng Zhu²
Ping Luo¹ Song Han^{2,3} Enze Xie²

Strategy 2: Greedy + Gumbel Noise

Like previous but adds noise (from a truncated Gumbel distribution) to the categorical probabilities to improve numerical stability.

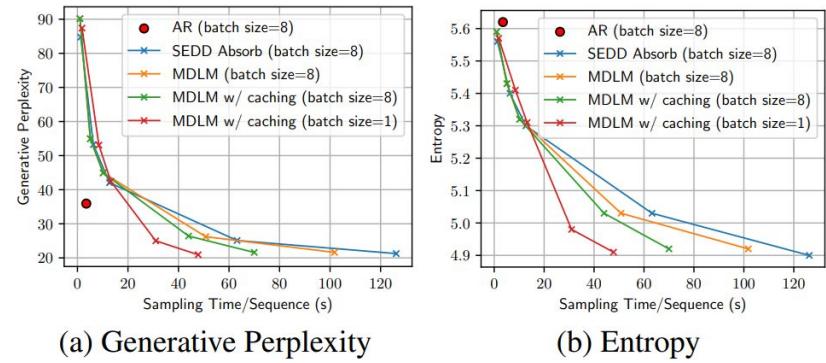
Authors noticed that MDMs get very low generative perplexity (as scored by some other LM, e.g. GPT2...), but often with crappy generations *if float precision isn't high enough*

MASKED DIFFUSION MODELS ARE SECRETLY TIME-AGNOSTIC MASKED MODELS AND EXPLOIT INACCURATE CATEGORICAL SAMPLING

Kaiwen Zheng^{1*}, Yongxin Chen², Hanzi Mao², Ming-Yu Liu², Jun Zhu^{1†}, Qinsheng Zhang²

¹Department of Computer Science & Technology, Institute for AI, Tsinghua University

²NVIDIA



There is the following definition:

The “right lane” on the lane lane lane. From the lane lane lane from lane lane to lane lane on a lane in lane lane on a front lane.

From lane lane lane the lane lane lane on the right lane from lane front lane lane to top lane.

From the right lane from a lane lane lane with the lane on the lane lane lane. The “that lane lane” on the rear lane.

Figure 6: Segment of generated text by SEDD Absorb (Lou et al., 2023) at 50k sampling steps.