Large Language Diffusion Models

Shen Nie[1*†] Fengqi Zhu[1*†] Zebin You[1†] Xiaolu Zhang[2‡] Jingyang Ou[1] Jun Hu[2‡] Jun Zhou[2]
Yankai Lin[1‡] Ji-Rong Wen[1] 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

TL;DR

Autoregressive models (ARMs) are the current standard for LLMS

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

TL;DR

Autoregressive models (ARMs) are the current standard for LLMS

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

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

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

TL;DR

Autoregressive models (ARMs) are the current standard for LLMS

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

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

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

The authors create a Llama8B-like diffusion model (“LLaDa 8B”) and achieve comparable (or slightly better?) performance, with seemingly decent scaling

Why hasn’t this been done before?

Diffusion has been used in NLP before, but isn’t usually great or scalable

Why hasn’t this been done before?

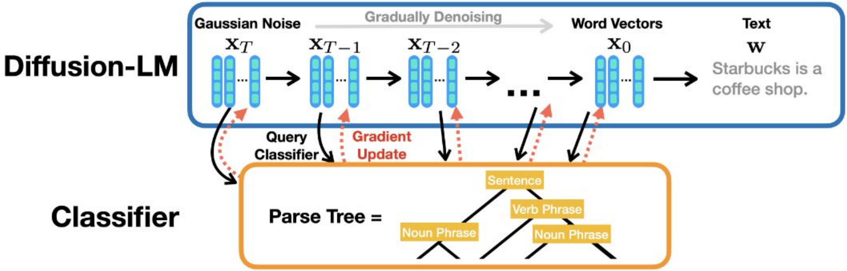
Diffusion has been used in NLP before, but isn’t usually great or scalable

1. Have to convert from continuous latent variables to discrete tokens

Why hasn’t this been done before?

Diffusion has been used in NLP before, but isn’t usually great or scalable

1. Have to convert from continuous latent variables to discrete tokens
    - Leads to weird things like “Diffusion-LM” (Li et al., 2022) where you’re generating conditioned on specific grammatical structure

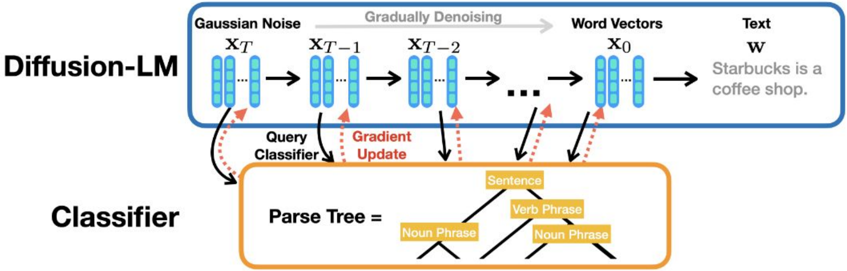


“Diffusion-LM Improves Controllable Text Generation”, Li et al. (2023)

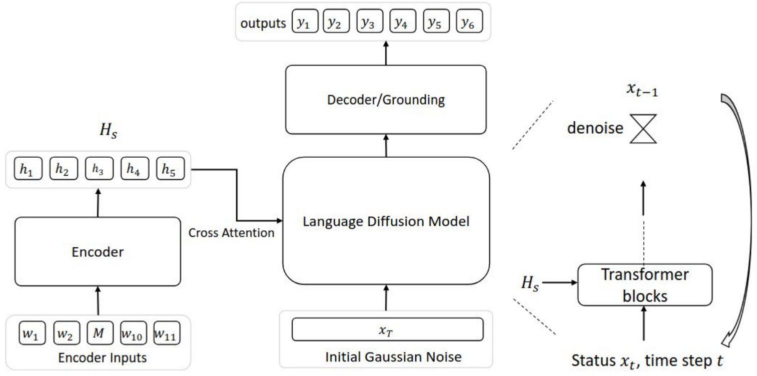
Why hasn't this been done before?

Diffusion has been used in NLP before, but isn't usually great or scalable

1. Have to convert from continuous latent variables to discrete tokens
   - ○ Leads to weird things like "Diffusion-LM" (Li et al., 2022) where you're generating conditioned on specific grammatical structure
   - ○ Or to very complicated diffusion models that work on a continuous version of discrete space, e.g. "DINOISER", (Ye et al., 2023), or "GENIE" (Lin et al., 2023).



"Diffusion-LM Improves Controllable Text Generation", Li et al. (2023)



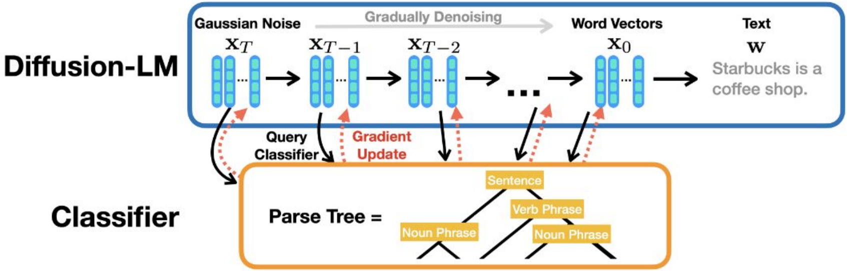
GENIE: "Text Generation with Diffusion Language Models: A Pre-training Approach with Continuous Paragraph Denoise", Lin et al. (2023)

Why hasn't this been done before?

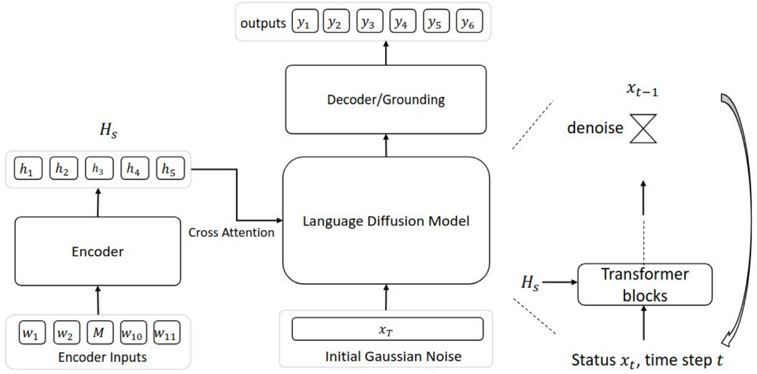
Diffusion has been used in NLP before, but isn't usually great or scalable

1. Have to convert from continuous latent variables to discrete tokens
    - Leads to weird things like "Diffusion-LM" (Li et al., 2022) where you're generating conditioned on specific grammatical structure
    - Or to very complicated diffusion models that work on a continuous version of discrete space, e.g. "DINOISER", (Ye et al., 2023), or "GENIE" (Lin et al., 2023).

2. Hard to scale this up



"Diffusion-LM Improves Controllable Text Generation", Li et al. (2023)

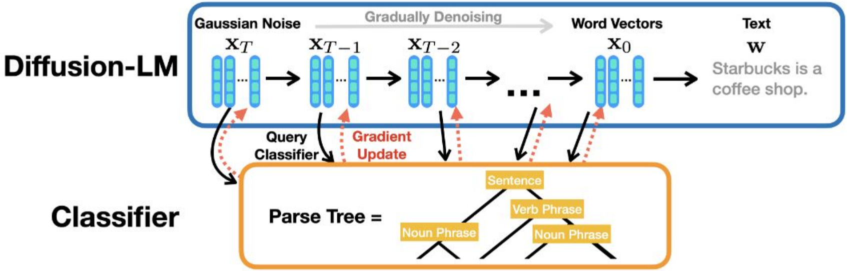


GENIE: "Text Generation with Diffusion Language Models: A Pre-training Approach with Continuous Paragraph Denoise", Lin et al. (2023)

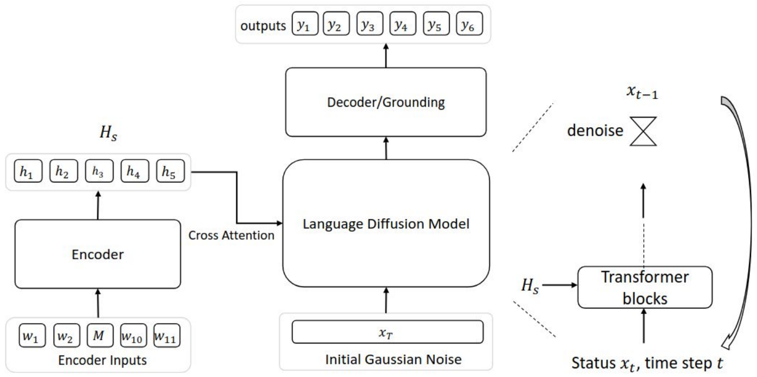
Why hasn’t this been done before?

Diffusion has been used in NLP before, but isn’t usually great or scalable

1. Have to convert from continuous latent variables to discrete tokens
    - Leads to weird things like “Diffusion-LM” (Li et al., 2022) where you’re generating conditioned on specific grammatical structure
    - Or to very complicated diffusion models that work on a continuous version of discrete space, e.g. “DINOISER”, (Ye et al., 2023), or “GENIE” (Lin et al., 2023).
2. Hard to scale this up
    - “Language Modeling is Compression” (Deletang et al., 2024), shows a 1B model like this required 64x the compute as a 1B ARM



“Diffusion-LM Improves Controllable Text Generation”, Li et al. (2023)



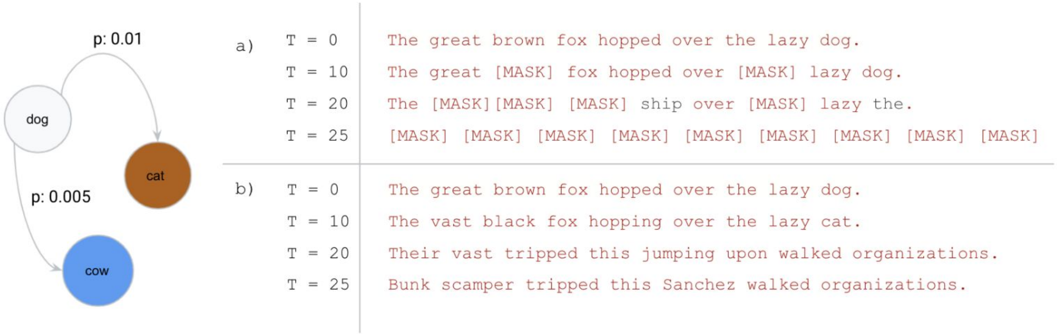
GENIE: “Text Generation with Diffusion Language Models: A Pre-training Approach with Continuous Paragraph Denoise”, Lin et al. (2023)

Discrete Diffusion

- You could do *discrete diffusion*

Discrete Diffusion

- You could do *discrete diffusion*
    - "Structured Denoising Diffusion Models in Discrete State-Spaces" (Austin et al., 2023)



a)
T = 0 The great brown fox hopped over the lazy dog.
T = 10 The great [MASK] fox hopped over [MASK] lazy dog.
T = 20 The [MASK][MASK] [MASK] ship over [MASK] lazy the.
T = 25 [MASK] [MASK] [MASK] [MASK] [MASK] [MASK] [MASK] [MASK] [MASK]

b)
T = 0 The great brown fox hopped over the lazy dog.
T = 10 The vast black fox hopping over the lazy cat.
T = 20 Their vast tripped this jumping upon walked organizations.
T = 25 Bunk scamper tripped this Sanchez walked organizations.

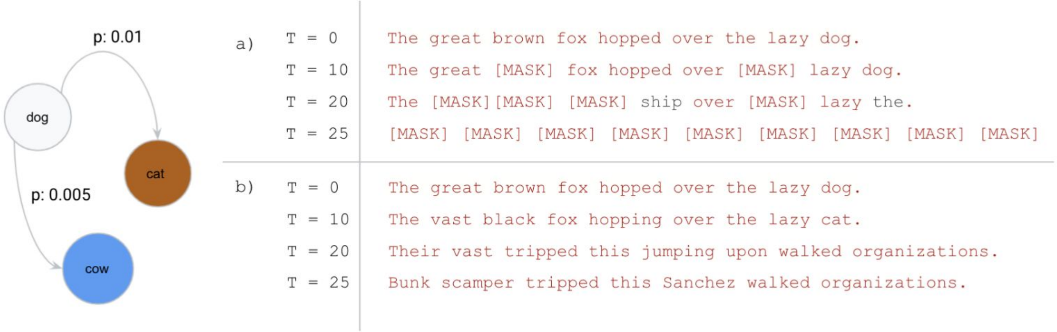
Figure 4: Two examples of noise schedules transforming text data. The top is a BERT-like absorbing + uniform diffusion which replaces tokens with [MASK] tokens (and occasionally with any other token, in black). The bottom is nearest-neighbor diffusion in embedding space. At left represents a possible column in the transition matrix.

(Austin et al., 2023)

Discrete Diffusion

- You could do *discrete diffusion*
  - “Structured Denoising Diffusion Models in Discrete State-Spaces” (Austin et al., 2023)
- In particular, *Masked Diffusion Models*
  - “Discrete Diffusion Modeling by Estimating the Ratios of the Data Distribution” (Lou et al., 2023)

(Complicated implementation using *denoising score entropy)*



a) T = 0 The great brown fox hopped over the lazy dog.
T = 10 The great [MASK] fox hopped over [MASK] lazy dog.
T = 20 The [MASK][MASK] [MASK] ship over [MASK] lazy the.
T = 25 [MASK] [MASK] [MASK] [MASK] [MASK] [MASK] [MASK] [MASK] [MASK]

b) T = 0 The great brown fox hopped over the lazy dog.
T = 10 The vast black fox hopping over the lazy cat.
T = 20 Their vast tripped this jumping upon walked organizations.
T = 25 Bunk scamper tripped this Sanchez walked organizations.

Figure 4: Two examples of noise schedules transforming text data. The top is a BERT-like absorbing + uniform diffusion which replaces tokens with [MASK] tokens (and occasionally with any other token, in black). The bottom is nearest-neighbor diffusion in embedding space. At left represents a possible column in the transition matrix.

(Austin et al., 2023)

$$\mathbb{E}_{x_0 \sim p_0, x \sim p(\cdot|x_0)} \left[\sum_{y \sim x} s_\theta(x)_y - \frac{p(y|x_0)}{p(x|x_0)} \log s_\theta(x)_y \right]$$

Discrete Diffusion Drawbacks

1. In a previous paper (Nie et al., 2024), the authors suggest you still need 16x the compute of an equivalent ARM to achieve the same likelihood

Discrete Diffusion Drawbacks

1. In a previous paper (Nie et al., 2024), the authors suggest you still need 16x the compute of an equivalent ARM to achieve the same likelihood
    - (The authors say not to worry: Diffusion optimises a *bound* on the likelihood, which isn’t even what you necessarily care about for downstream tasks (i.e. accuracy), so we shouldn’t write off MDMs…)

Discrete Diffusion Drawbacks

1. In a previous paper (Nie et al., 2024), the authors suggest you still need 16x the compute of an equivalent ARM to achieve the same likelihood
    - (The authors say not to worry: Diffusion optimises a *bound* on the likelihood, which isn't even what you necessarily care about for downstream tasks (i.e. accuracy), so we shouldn't write off MDMs…)

2. To get it to work there's a bunch of tricks (hacks?) that have been developed over the past couple of years

Discrete Diffusion Drawbacks

1. In a previous paper (Nie et al., 2024), the authors suggest you still need 16x the compute of an equivalent ARM to achieve the same likelihood
   - (The authors say not to worry: Diffusion optimises a *bound* on the likelihood, which isn't even what you necessarily care about for downstream tasks (i.e. accuracy), so we shouldn't write off MDMs…)

2. To get it to work there's a bunch of tricks (hacks?) that have been developed over the past couple of years

3. Also, it's just a bit weird…

It’s kind of weird…

It’s kind of weird…

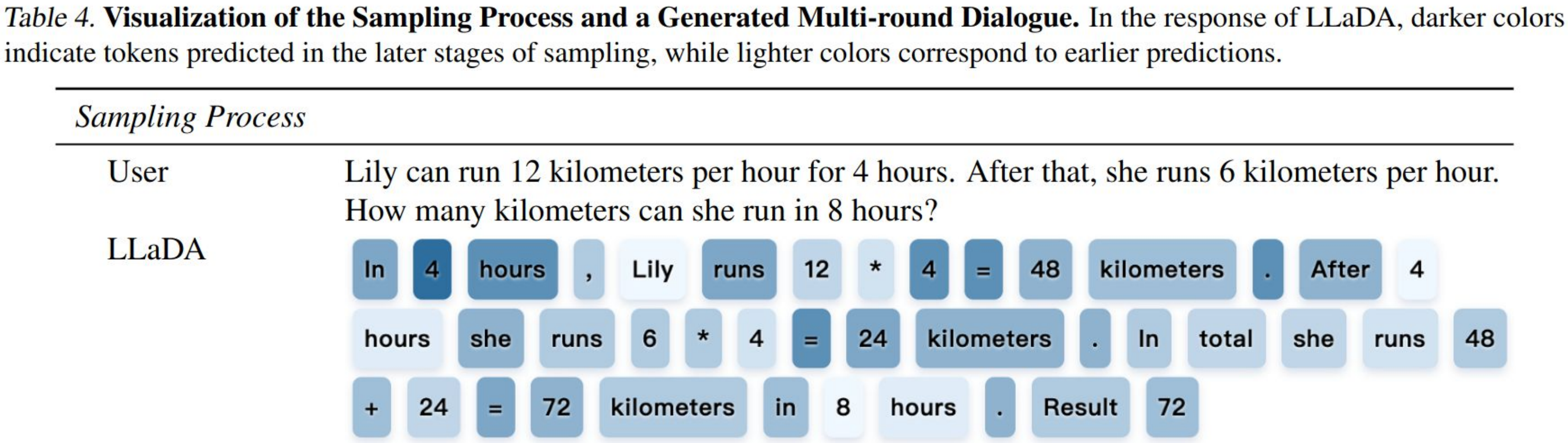


Table 4. **Visualization of the Sampling Process and a Generated Multi-round Dialogue.** In the response of LLaDA, darker colors indicate tokens predicted in the later stages of sampling, while lighter colors correspond to earlier predictions.

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	In 4 hours , Lily runs 12 * 4 = 48 kilometers . After 4 hours she runs 6 * 4 = 24 kilometers . In total she runs 48 + 24 = 72 kilometers in 8 hours . Result 72

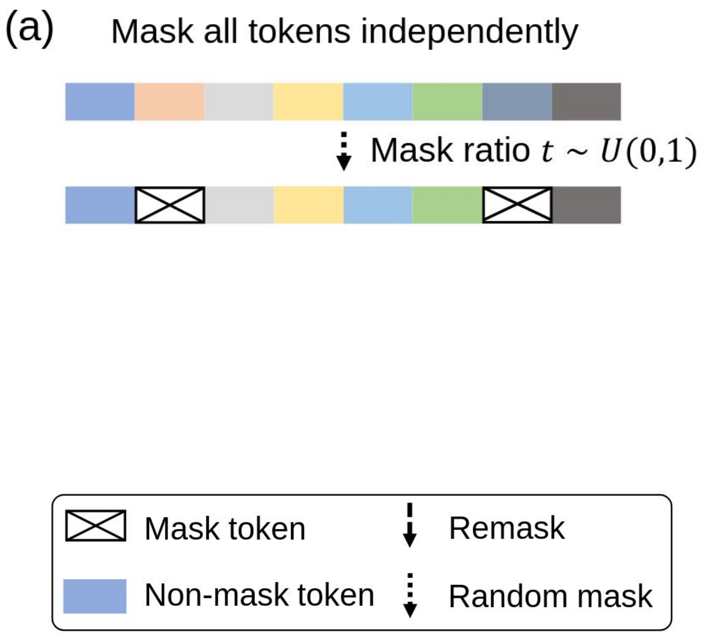
Masked Diffusion Model (MDM)

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

Masked Diffusion Model (MDM)

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

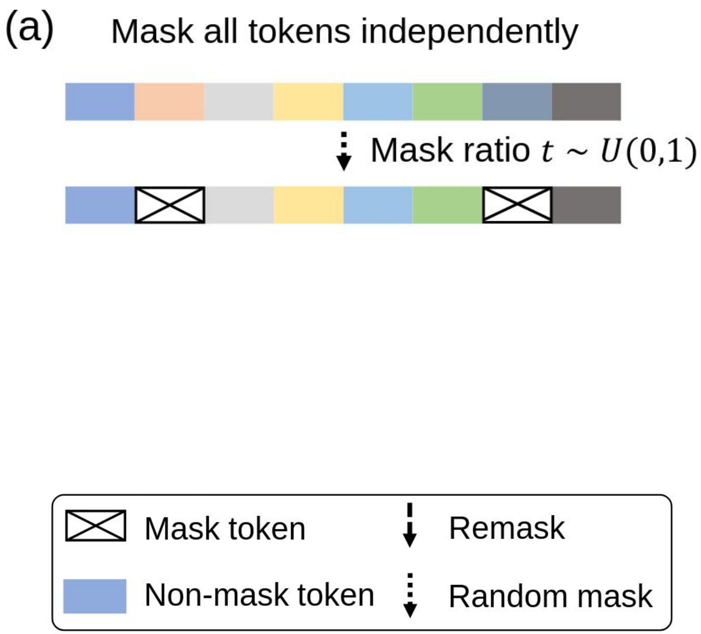
- *forward process*: gradually mask tokens (independently) until fully masked at $t = 1$:



Masked Diffusion Model (MDM)

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

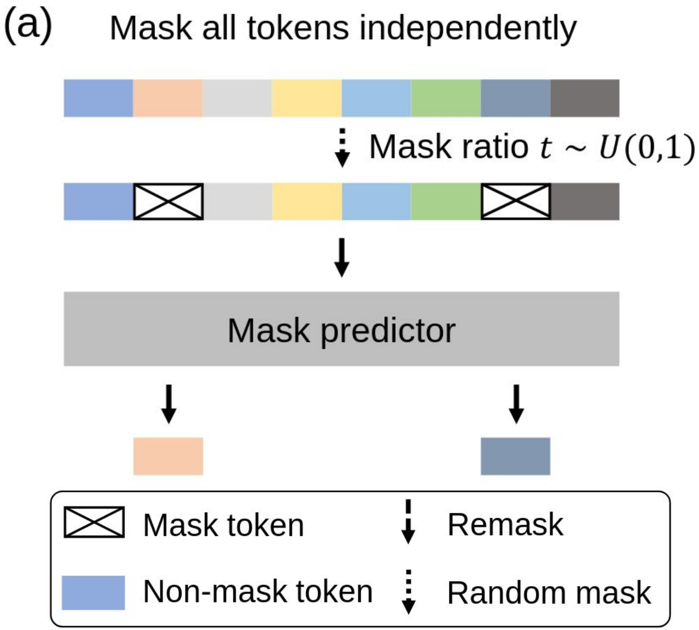
- *forward process*: gradually mask tokens (independently) until fully masked at $t = 1$:
    - At time $t \in [0, 1]$ each token is masked with prob. t
    - (Once a token is masked it stays masked for $t' > t$.)



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:



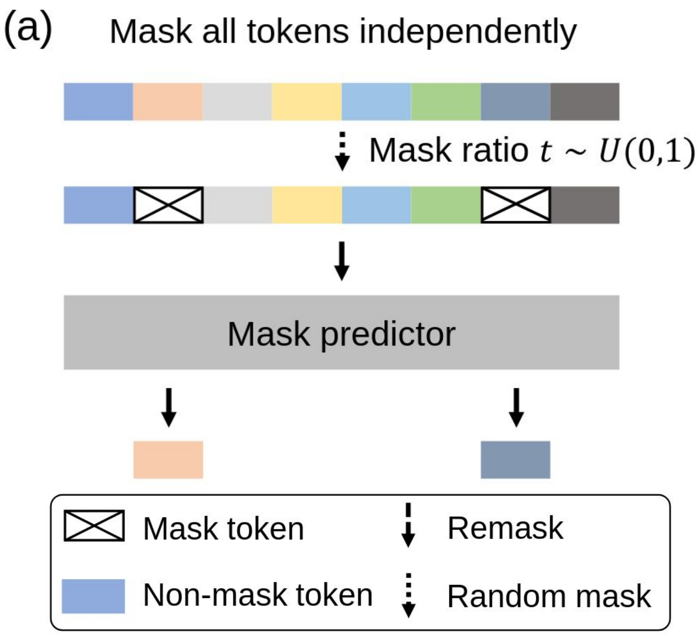
Masked Diffusion Model (MDM)

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?

Pretraining

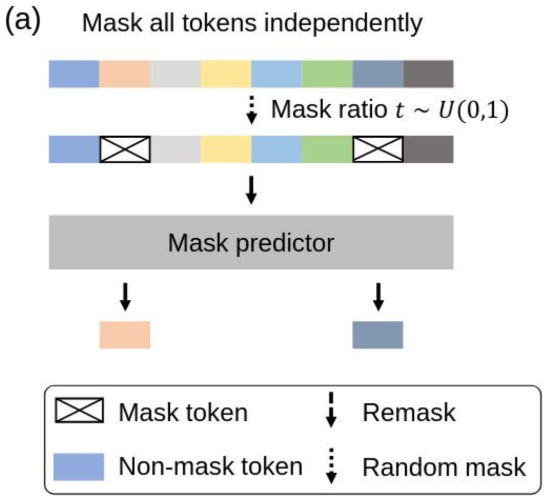


Figure 2. **A Conceptual Overview of LLaDA.** (a) Pre-training. LLaDA is trained on text with random masks applied independently to all tokens at the same ratio $t \sim U[0, 1]$. (b) SFT. Only response tokens are possibly masked. (c) Sampling. LLaDA simulates a diffusion process from $t = 1$ (fully masked)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

How do we apply the standard LLM pipeline?

Pretraining

SFT

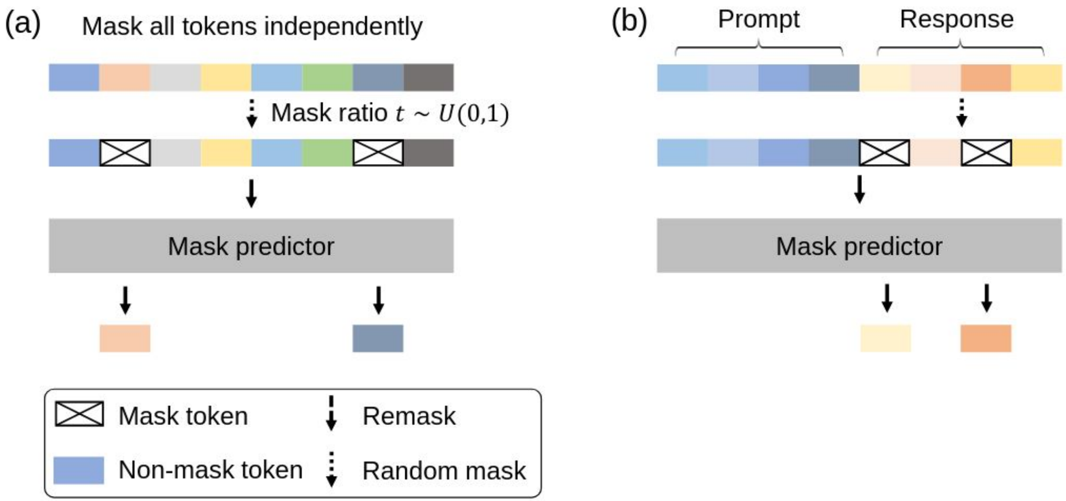


Figure 2. **A Conceptual Overview of LLaDA.** (a) Pre-training. LLaDA is trained on text with random masks applied independently to all tokens at the same ratio $t \sim U[0, 1]$. (b) SFT. Only response tokens are possibly masked. (c) Sampling. LLaDA simulates a diffusion process from $t = 1$ (fully masked)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

How do we apply the standard LLM pipeline?

Pretraining SFT Sampling

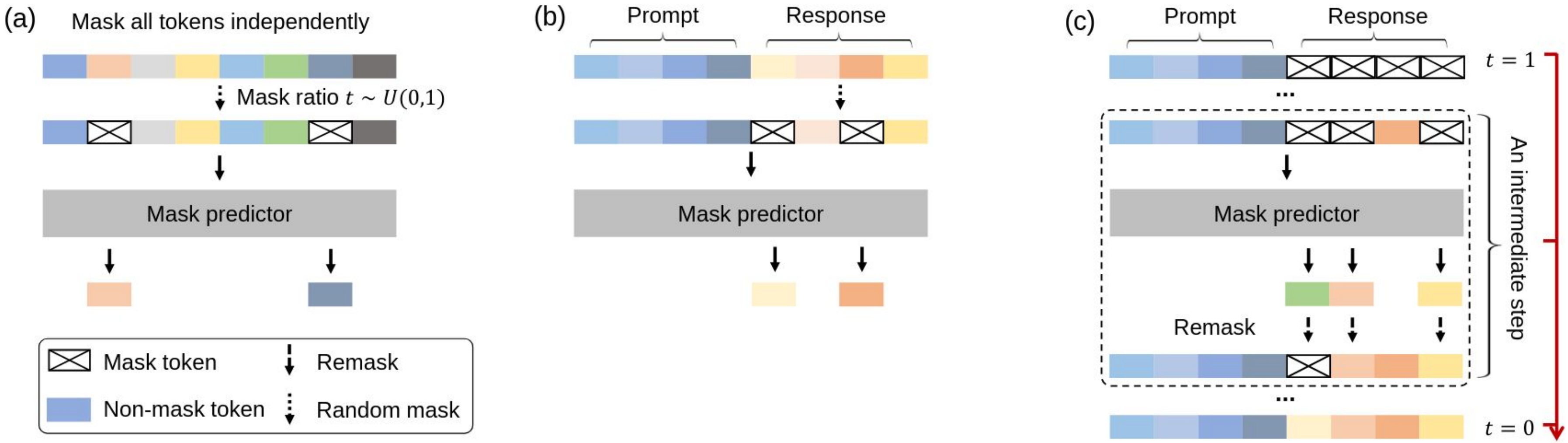
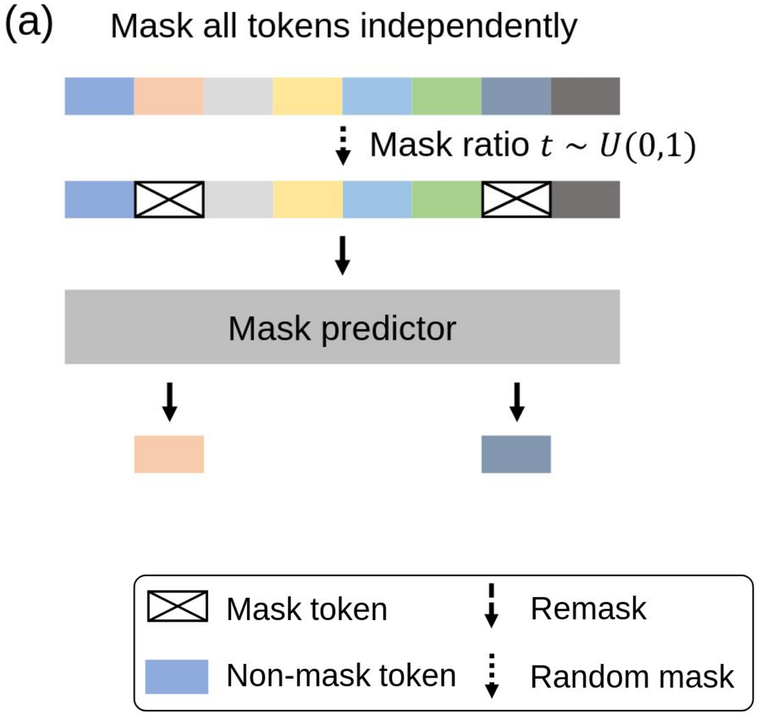


Figure 2. **A Conceptual Overview of LLaDA.** (a) Pre-training. LLaDA is trained on text with random masks applied independently to all tokens at the same ratio $t \sim U[0, 1]$. (b) SFT. Only response tokens are possibly masked. (c) Sampling. LLaDA simulates a diffusion process from $t = 1$ (fully masked)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

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

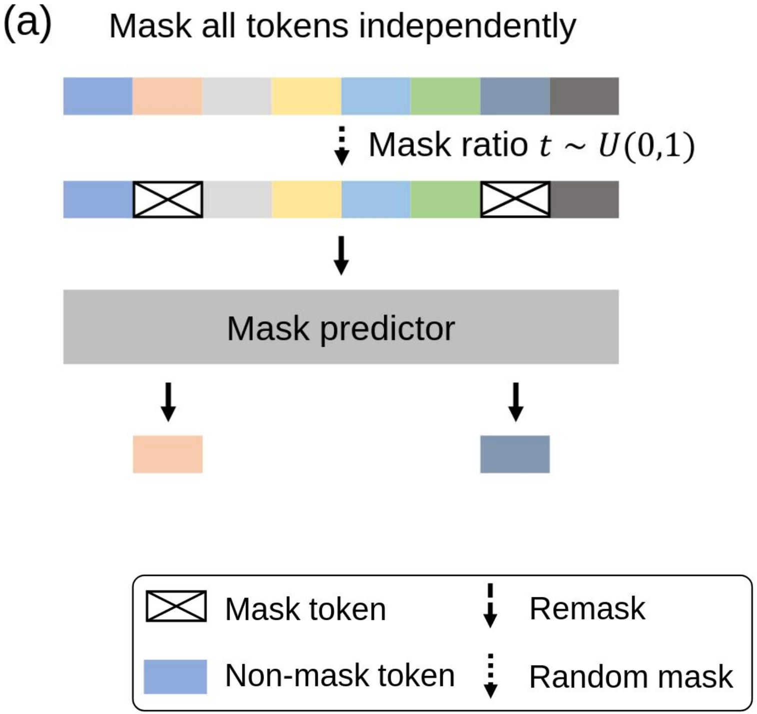


Pretraining

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

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

- Transformer-based architecture (Llama3)

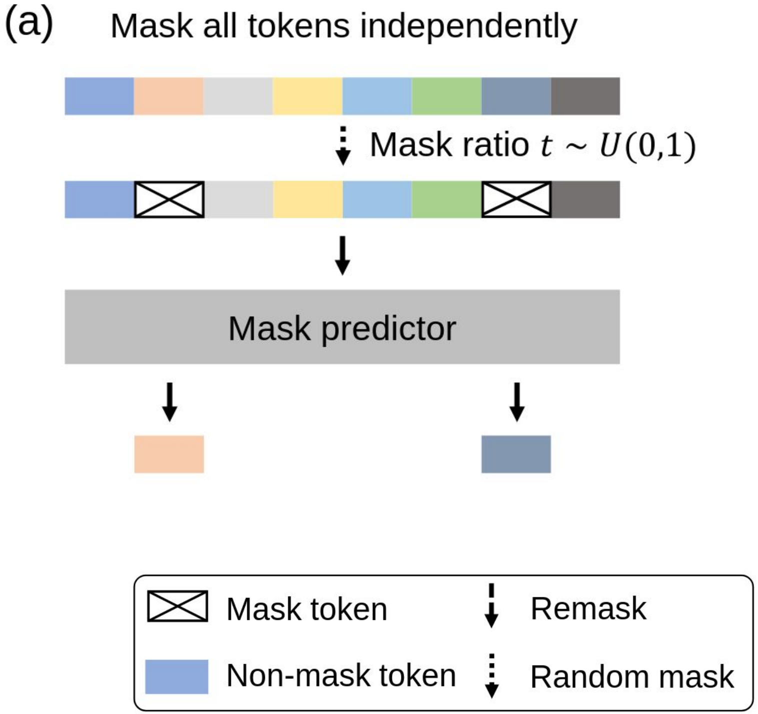


Pretraining

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

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

- Transformer-based architecture (Llama3)
- 1B & 8B models

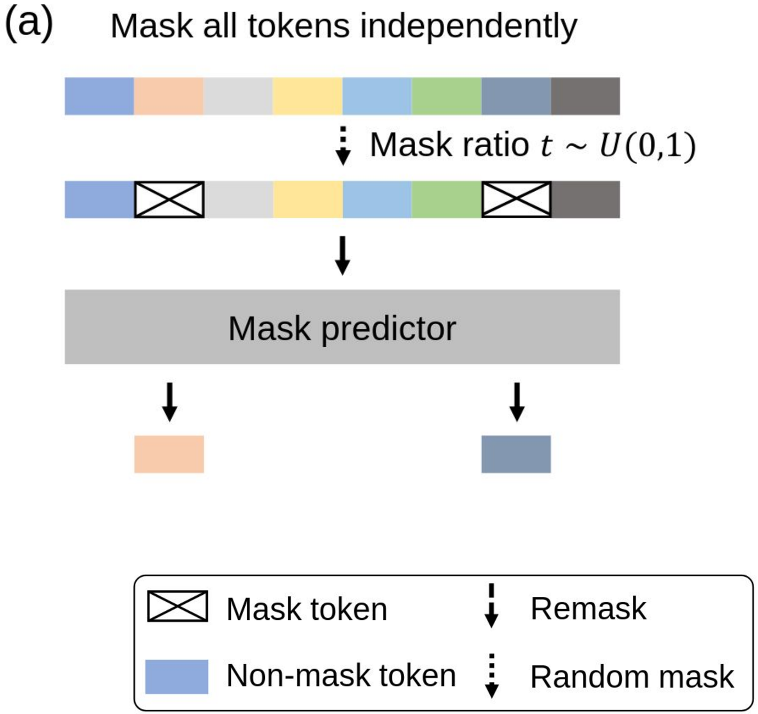


Pretraining

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

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

- Transformer-based architecture (Llama3)
- 1B & 8B models
- 2.3T tokens

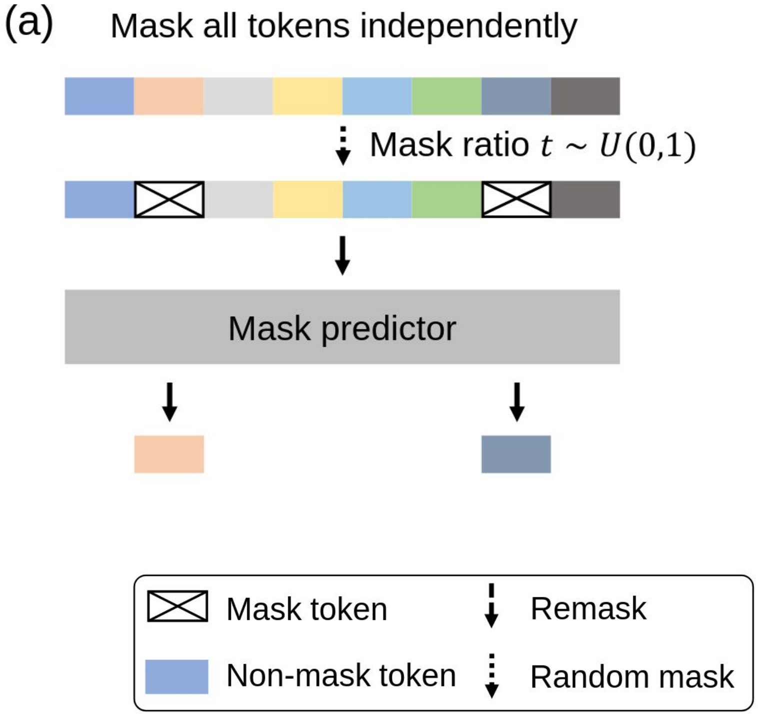


Pretraining

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

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

- Transformer-based architecture (Llama3)
- 1B & 8B models
- 2.3T tokens
- 0.13 million H800 GPU hours (8B model)



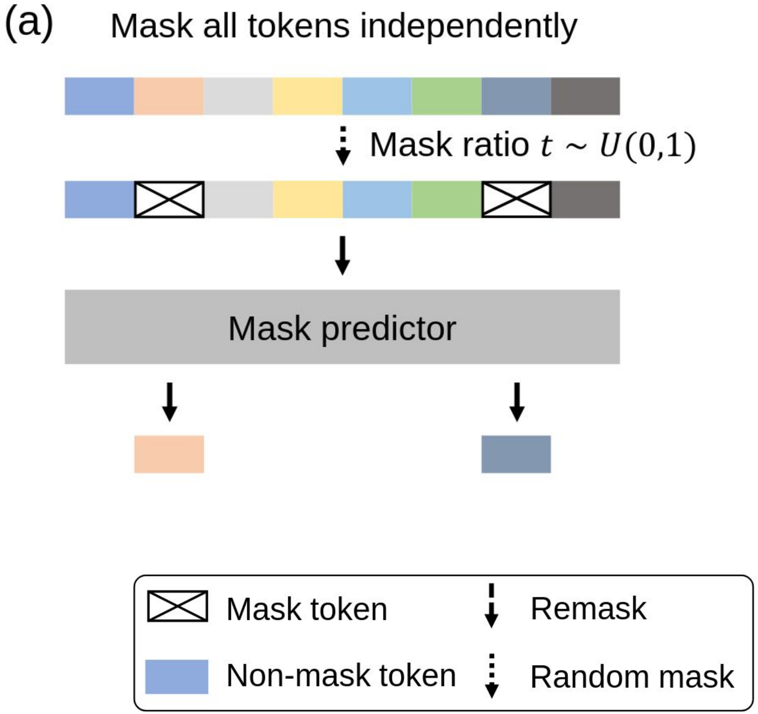
Pretraining

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

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

- Transformer-based architecture (Llama3)
- 1B & 8B models
- 2.3T tokens
- 0.13 million H800 GPU hours (8B model)

Fixed sequence length 4096:



Pretraining

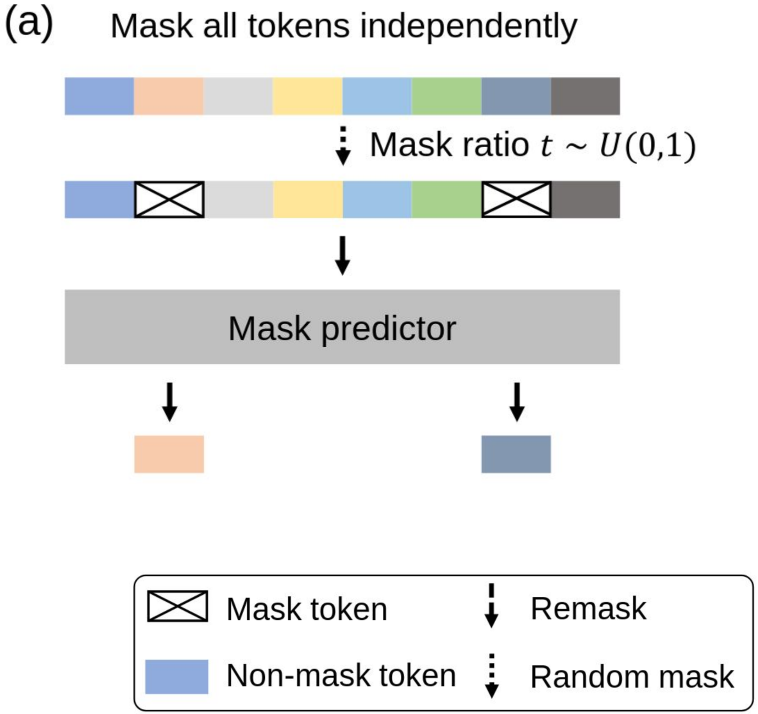
Train the mask predictor $p_\theta(\cdot, x_t)$ as we just described:

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

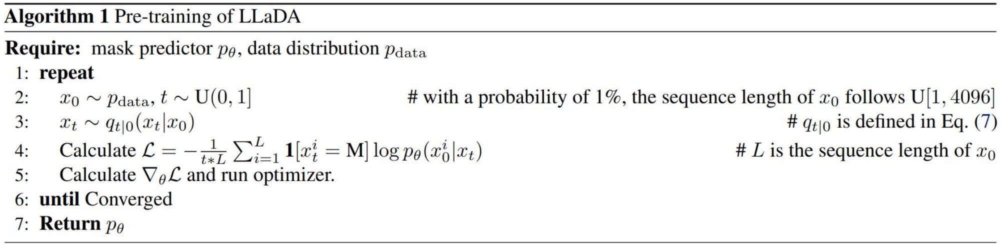
- Transformer-based architecture (Llama3)
- 1B & 8B models
- 2.3T tokens
- 0.13 million H800 GPU hours (8B model)

Fixed sequence length 4096:

- 1% of the pre-training data is set to a (uniformly) random length in [1, 4096] to help with variable-length inputs



Pretraining



Algorithm 1 Pre-training of LLaDA

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

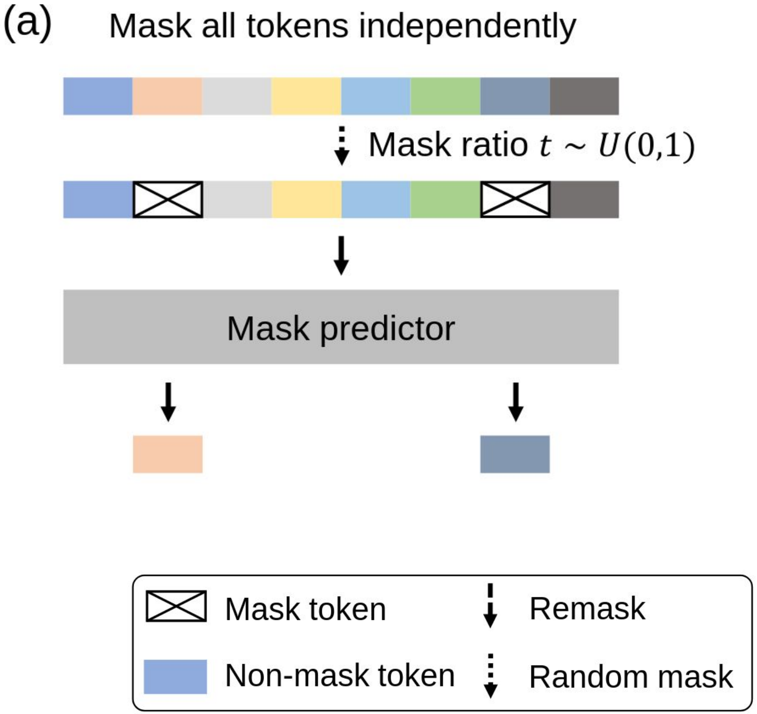
Train the mask predictor $p_\theta(\cdot, x_t)$ as we just described:

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

- Transformer-based architecture (Llama3)
- 1B & 8B models
- 2.3T tokens
- 0.13 million H800 GPU hours (8B model)

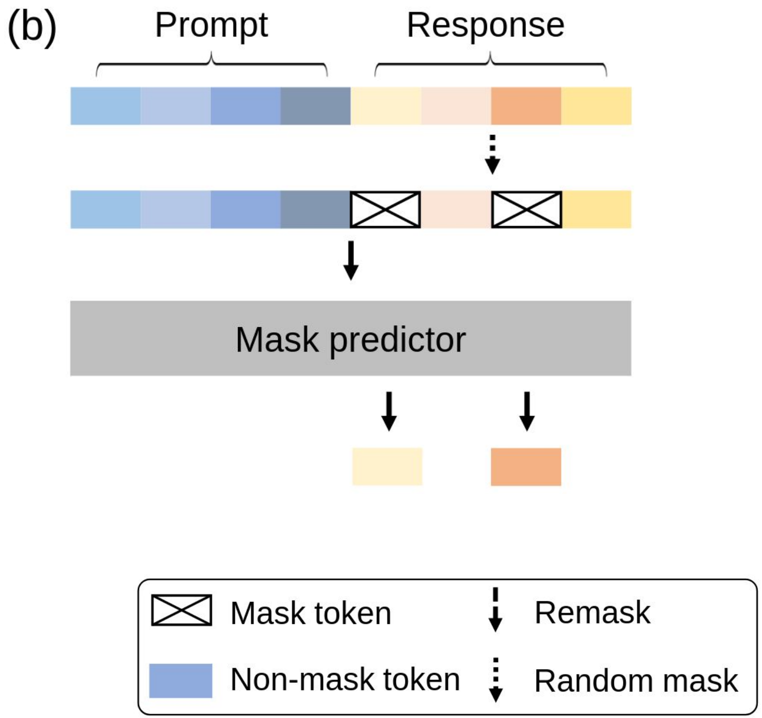
Fixed sequence length 4096:

- 1% of the pre-training data is set to a (uniformly) random length in [1, 4096] to help with variable-length inputs



SFT

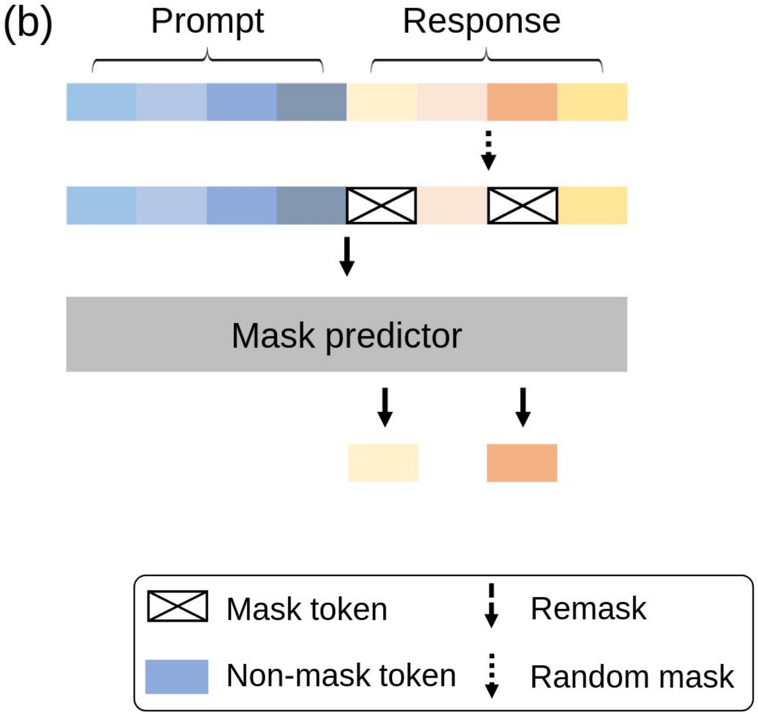
Slightly different loss (only care about response r_0 given prompt p_0 and intermediate response r_t):



SFT

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

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

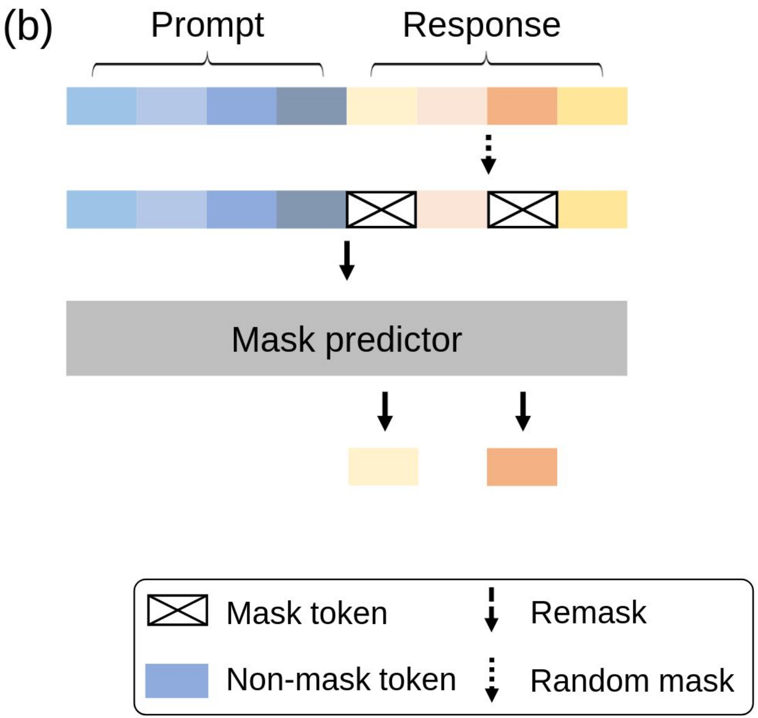


SFT

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

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

Where L' is a dynamic length:



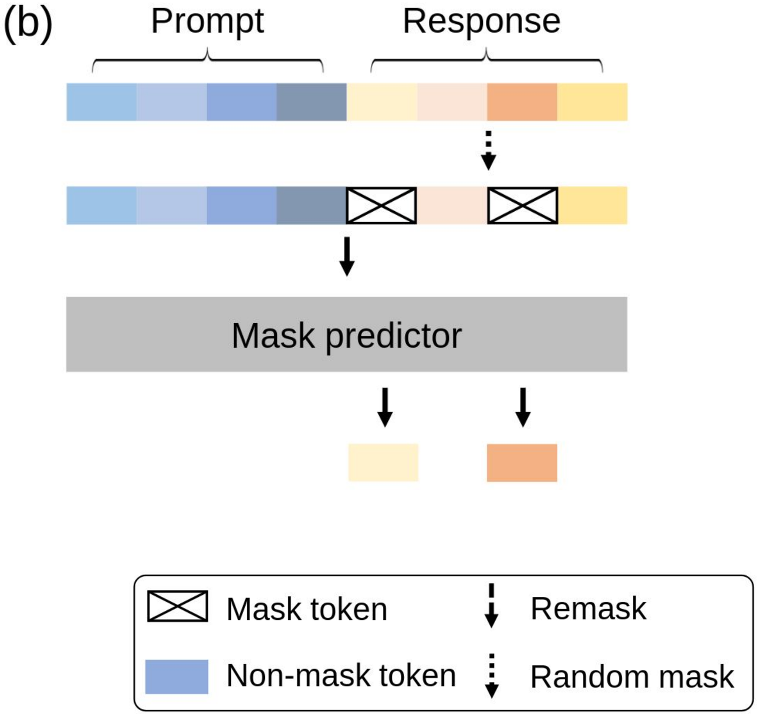
SFT

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

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

Where L' is a dynamic length:

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



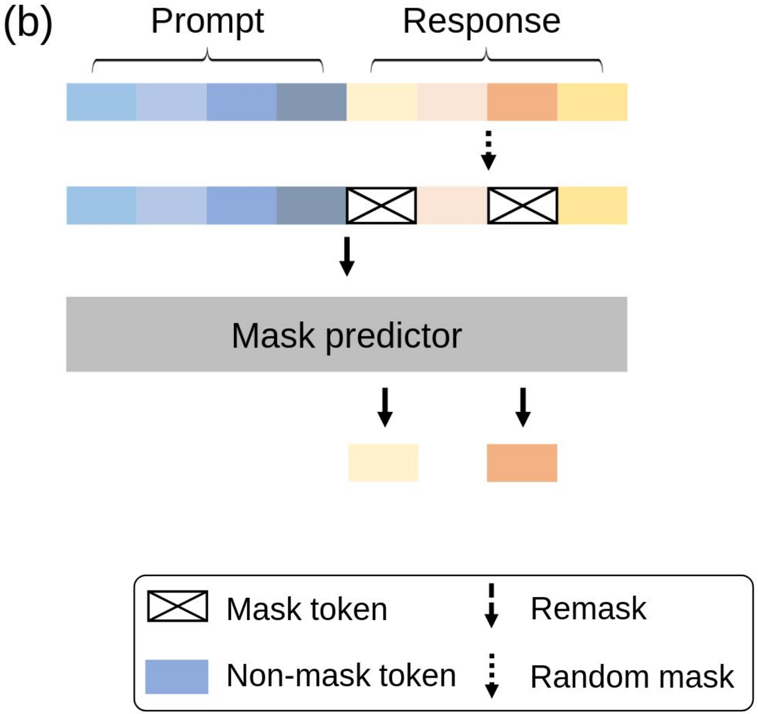
SFT

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)



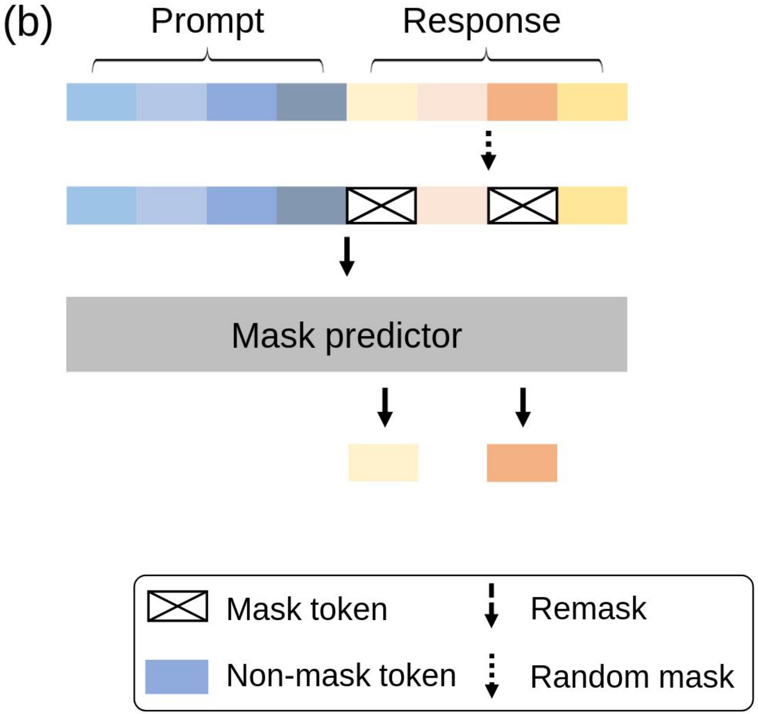
SFT

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

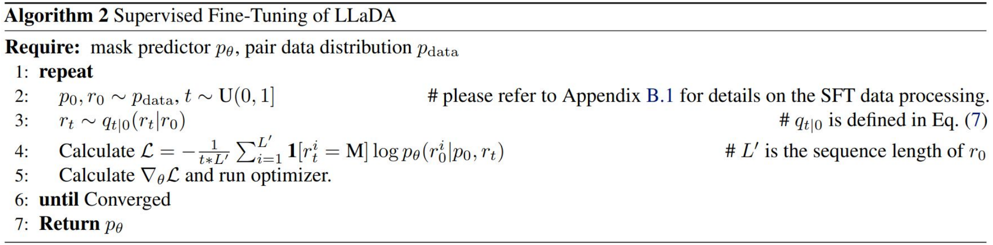
$$-\mathbb{E}_{t,p_0,r_0,r_t}\left[\frac{1}{t}\sum_{i=1}^{L'}\mathbf{1}[r_t^i = \mathbf{M}]\log p_\theta(r_0^i|p_0, r_t)\right]$$

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.



SFT



Algorithm 2 Supervised Fine-Tuning of LLaDA

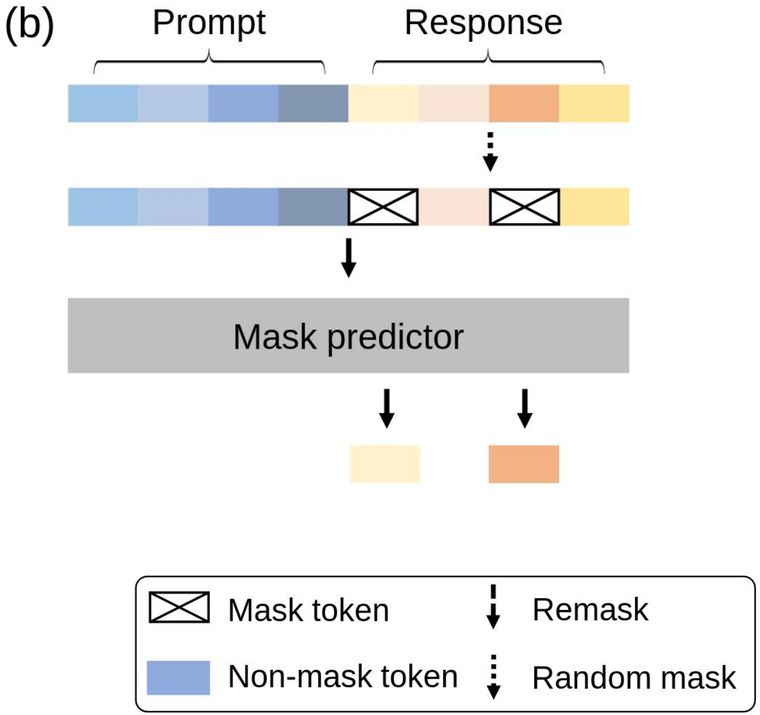
```
Require: mask predictor $p_\theta$, pair data distribution $p_{\mathrm{data}}$
repeat
    $p_0, r_0 \sim p_{\mathrm{data}}$, $t \sim \mathrm{U}(0, 1]$    # please refer to Appendix B.1 for details on the SFT data processing.
    $r_t \sim q_{t|0}(r_t|r_0)$    # $q_{t|0}$ is defined in Eq. (7)
  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

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

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

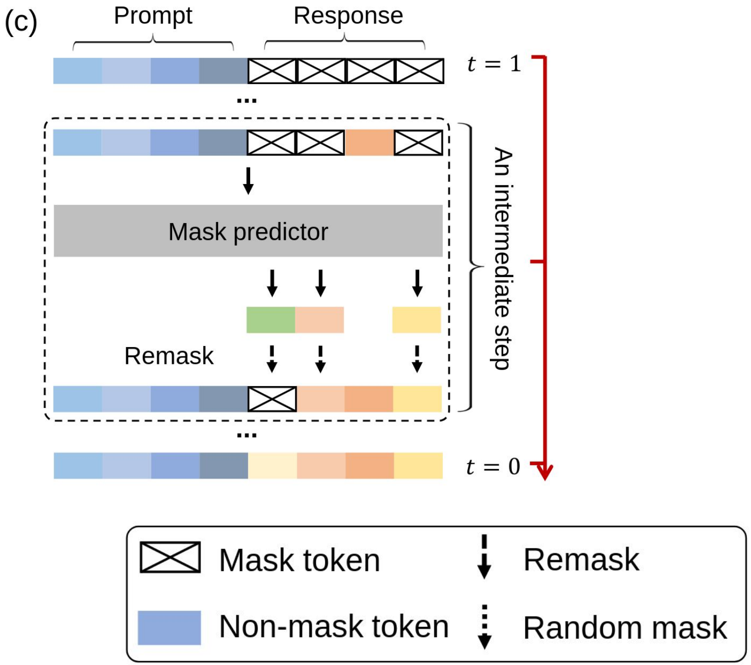
Where L' is a dynamic length:

- 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.



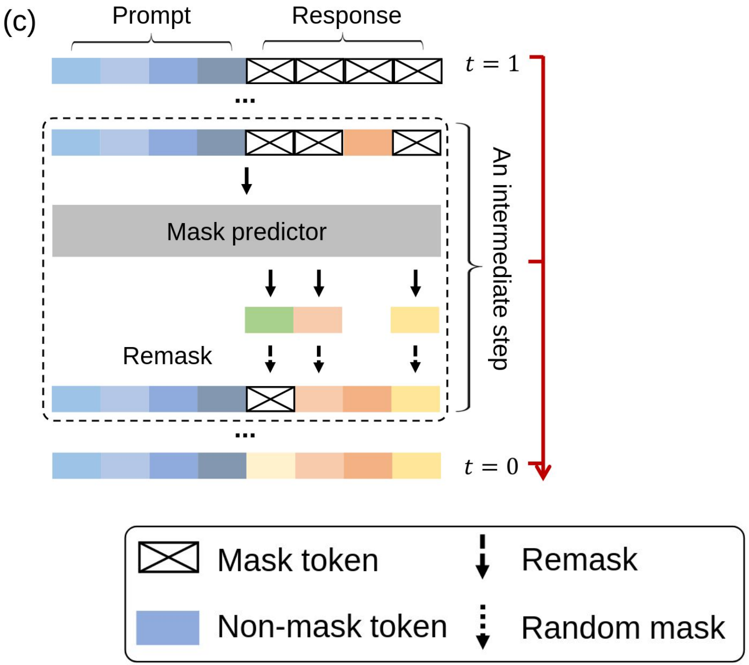
Sampling

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



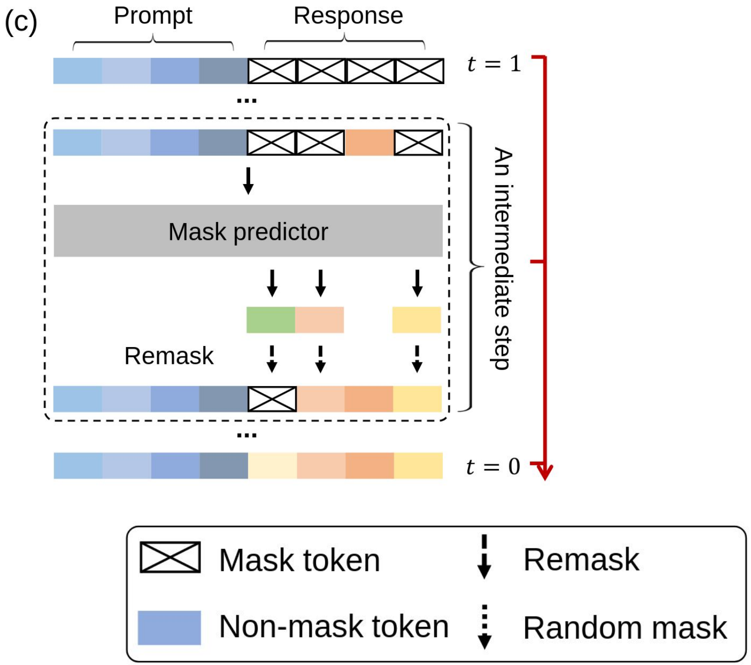
Sampling

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



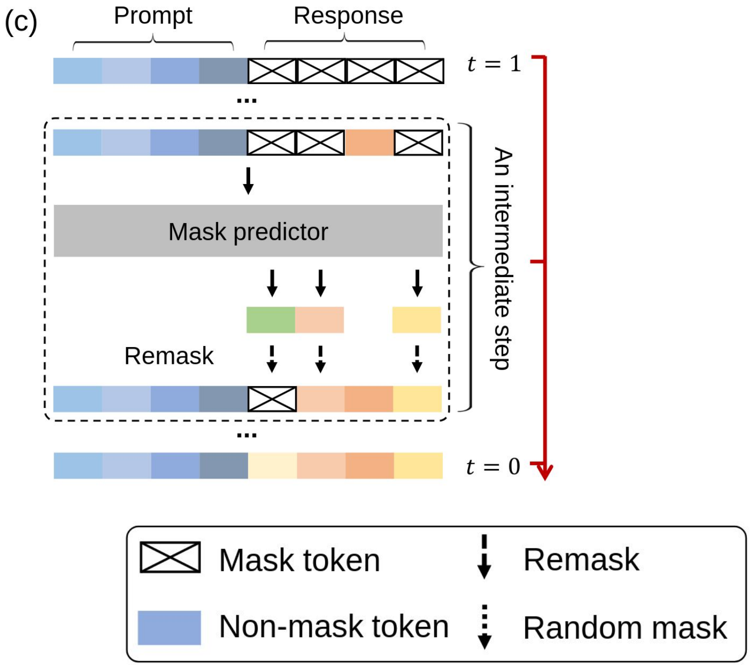
Sampling

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



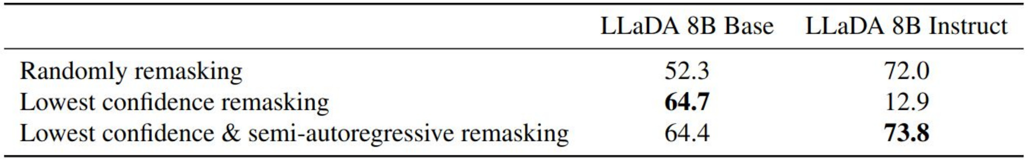
Sampling

1. Choose N sampling steps (tradeoff between efficiency and sample quality) between 1 and 0.
2. For each $t \in \{1, 1-1/N, \ldots, 1/N\}$, 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.



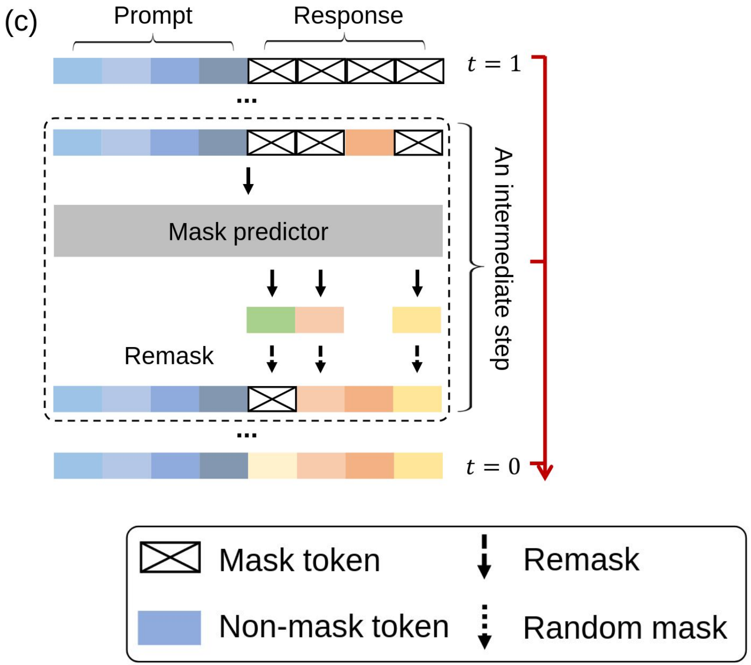
Sampling

1. Choose N sampling steps (tradeoff between efficiency and sample quality) between 1 and 0.
2. For each $t \in \{1, 1-1/N, \ldots, 1/N\}$, 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.



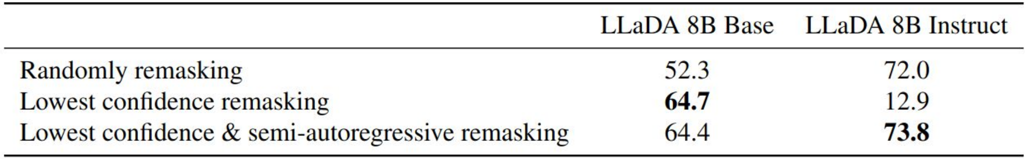
	LLaDA 8B Base	LLaDA 8B Instruct
Randomly remasking	52.3	72.0
Lowest confidence remasking	**64.7**	12.9
Lowest confidence & semi-autoregressive remasking	64.4	**73.8**

(on GSM8K)



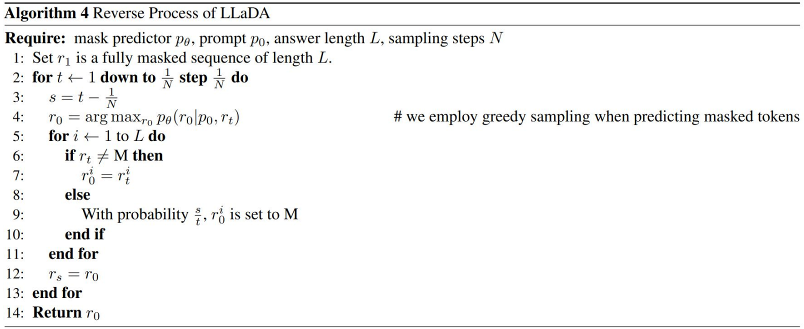
Sampling

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.



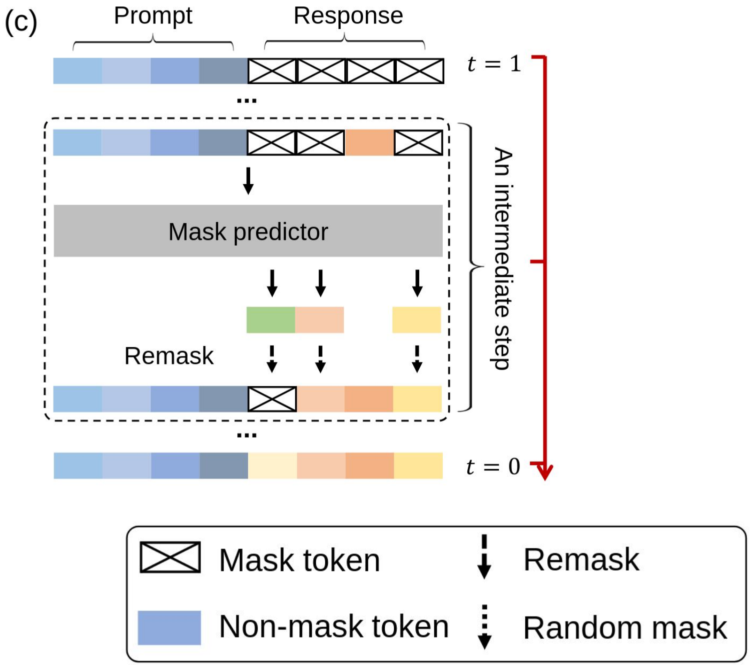
	LLaDA 8B Base	LLaDA 8B Instruct
Randomly remasking	52.3	72.0
Lowest confidence remasking	**64.7**	12.9
Lowest confidence & semi-autoregressive remasking	64.4	**73.8**

(on GSM8K)



Algorithm 4 Reverse Process of LLaDA

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



Semi-autoregressive Sampling

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

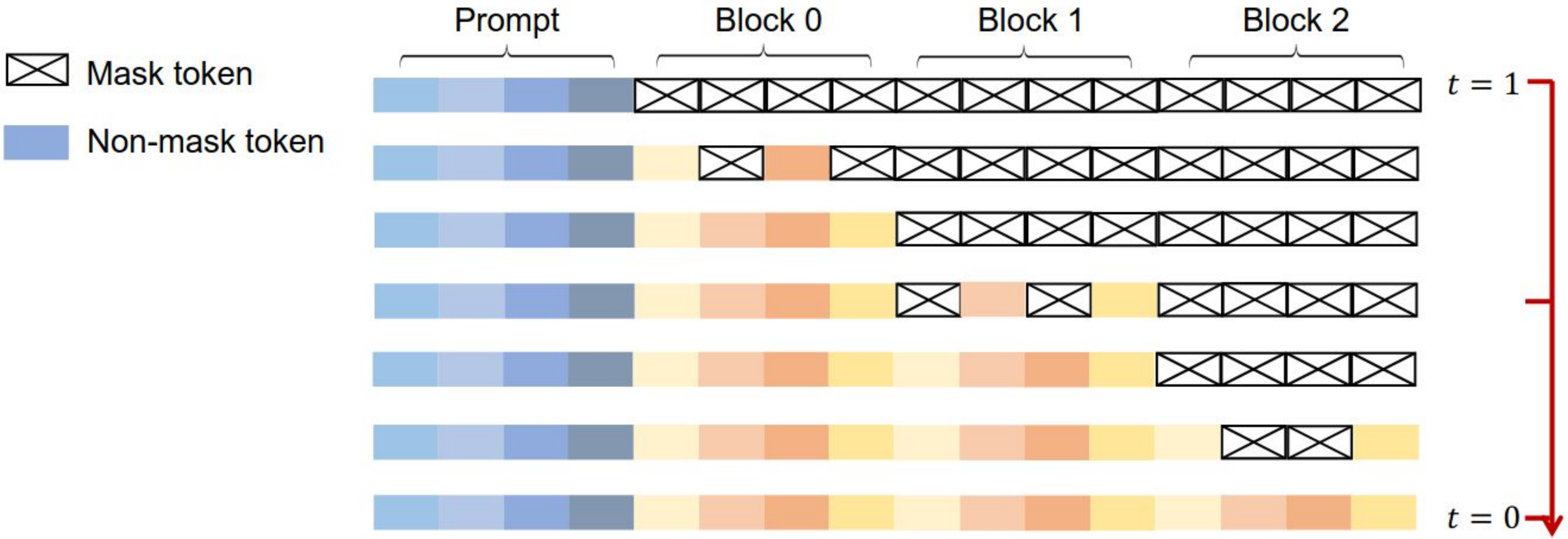
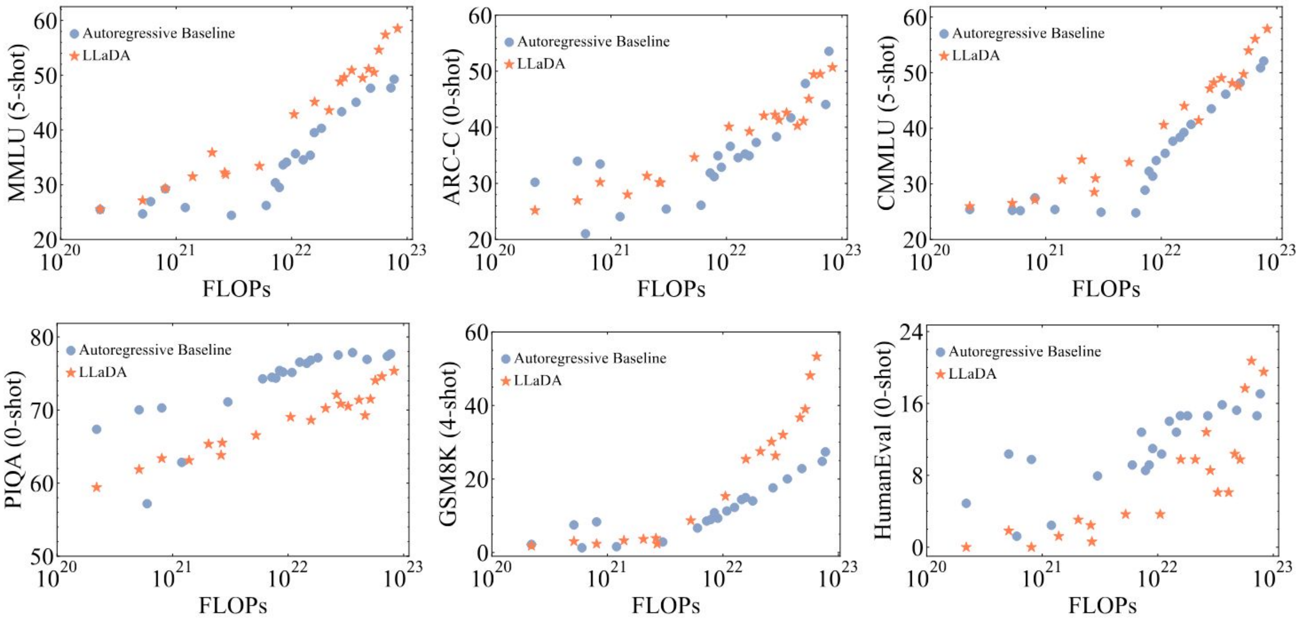


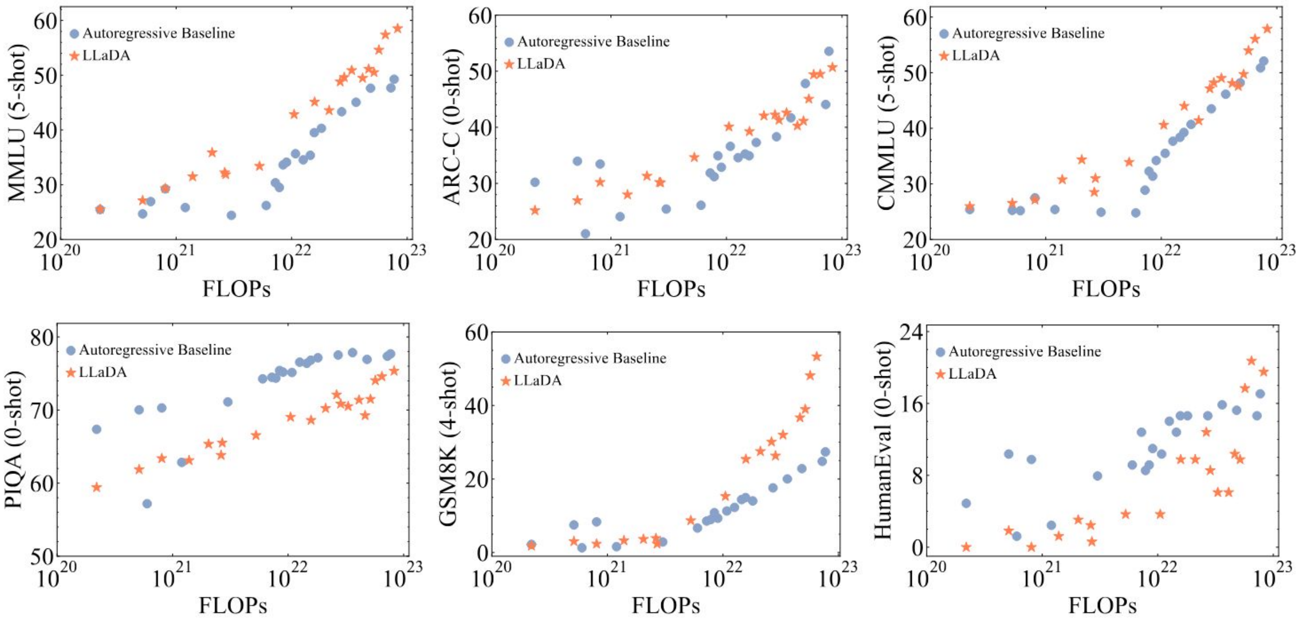
Figure 4. **A Conceptual Overview of the Semi-autoregressive Sampling.**

Experiment 1: Scalability



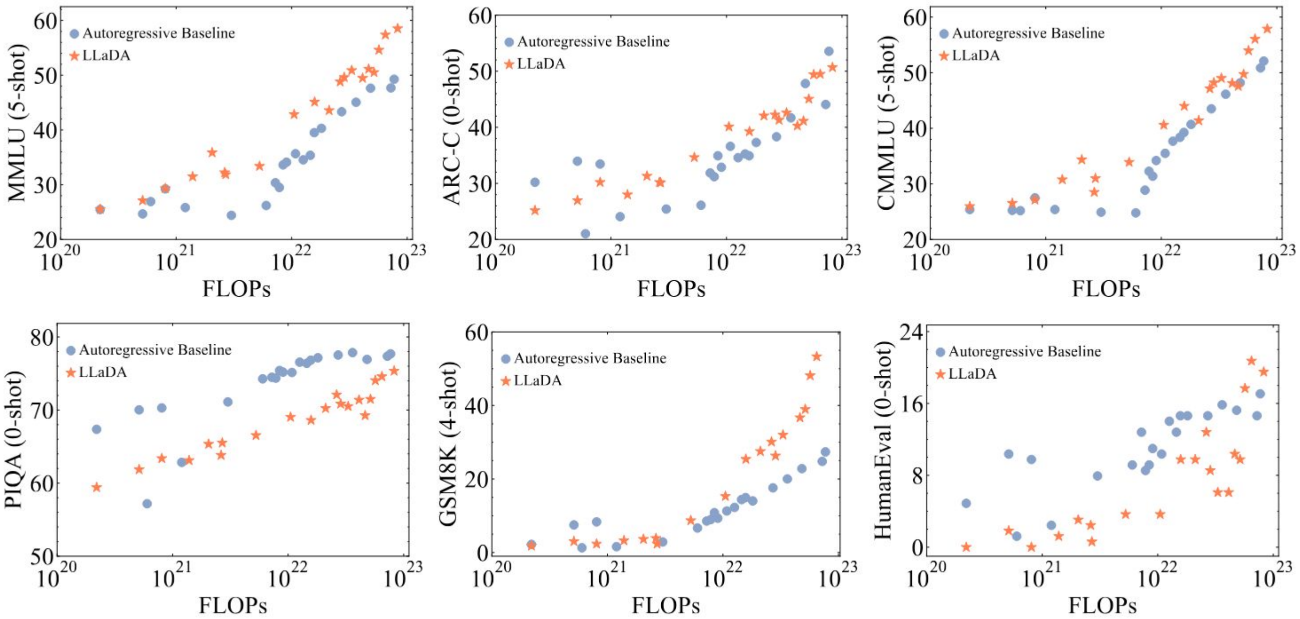
Experiment 1: Scalability

- Autoregressive Baseline = LLaDA architecture but autoregressive



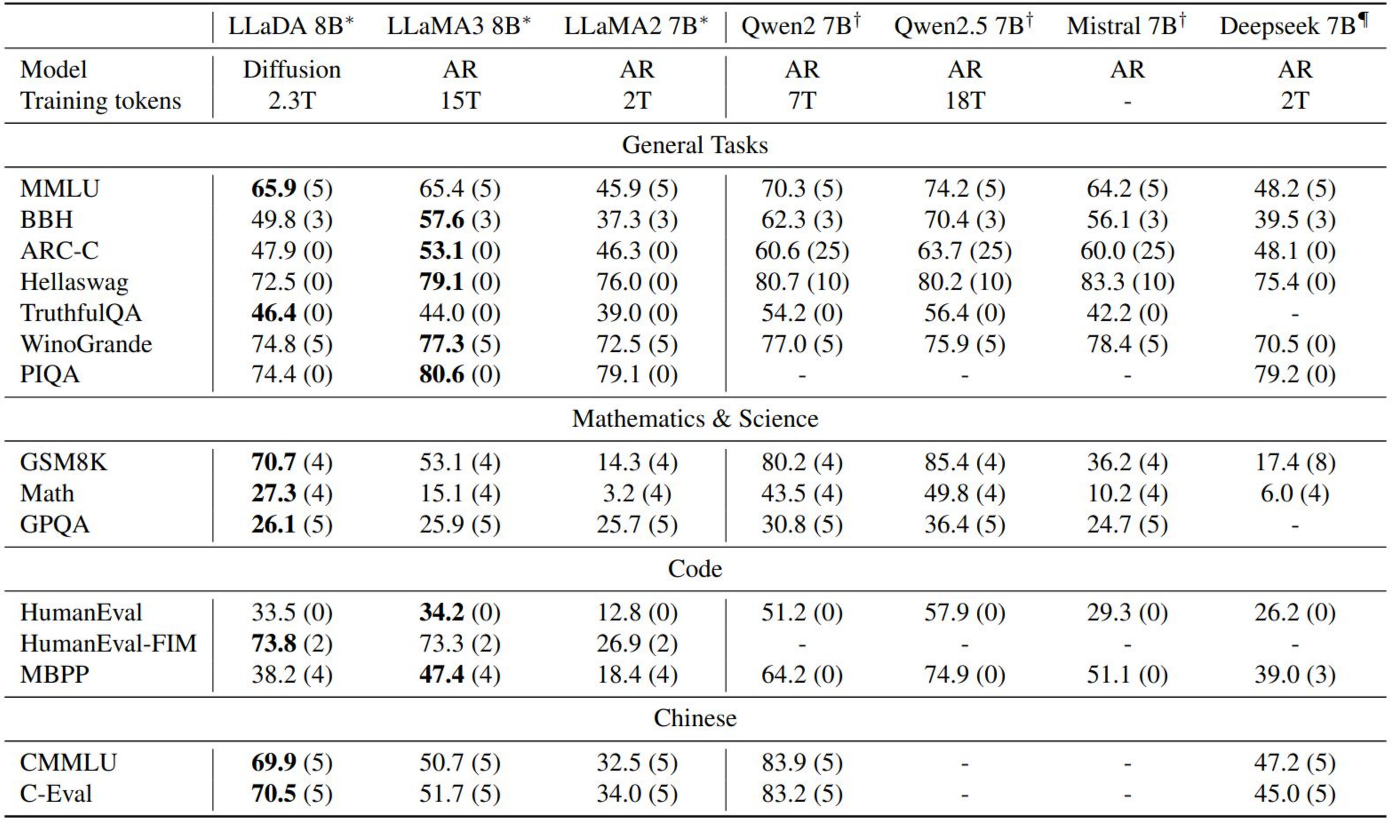
Experiment 1: Scalability

- Autoregressive Baseline = LLaDA architecture but autoregressive
- Comparable scaling across these 6 tasks



no error bars >:(

Experiment 2: benchmarks (pretrained)

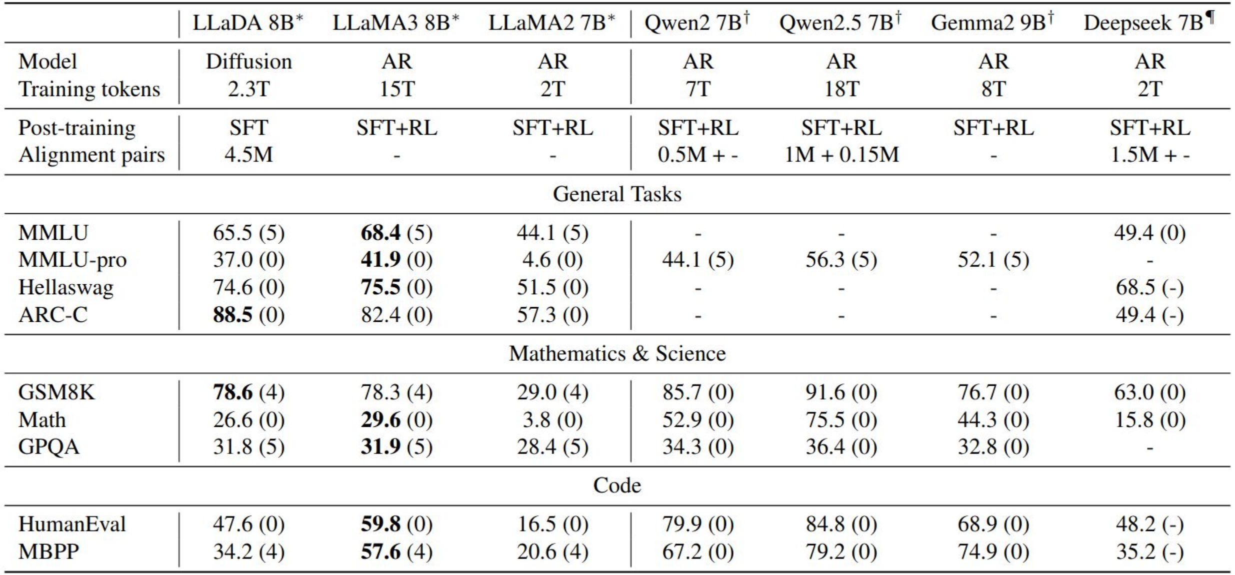


	LLaDA 8B*	LLaMA3 8B*	LLaMA2 7B*	Qwen2 7B†	Qwen2.5 7B†	Mistral 7B†	Deepseek 7B¶
Model	Diffusion	AR	AR	AR	AR	AR	AR
Training tokens	2.3T	15T	2T	7T	18T	-	2T
General Tasks							
MMLU	**65.9** (5)	65.4 (5)	45.9 (5)	70.3 (5)	74.2 (5)	64.2 (5)	48.2 (5)
BBH	49.8 (3)	**57.6** (3)	37.3 (3)	62.3 (3)	70.4 (3)	56.1 (3)	39.5 (3)
ARC-C	47.9 (0)	**53.1** (0)	46.3 (0)	60.6 (25)	63.7 (25)	60.0 (25)	48.1 (0)
Hellaswag	72.5 (0)	**79.1** (0)	76.0 (0)	80.7 (10)	80.2 (10)	83.3 (10)	75.4 (0)
TruthfulQA	**46.4** (0)	44.0 (0)	39.0 (0)	54.2 (0)	56.4 (0)	42.2 (0)	-
WinoGrande	74.8 (5)	**77.3** (5)	72.5 (5)	77.0 (5)	75.9 (5)	78.4 (5)	70.5 (0)
PIQA	74.4 (0)	**80.6** (0)	79.1 (0)	-	-	-	79.2 (0)
Mathematics & Science							
GSM8K	**70.7** (4)	53.1 (4)	14.3 (4)	80.2 (4)	85.4 (4)	36.2 (4)	17.4 (8)
Math	**27.3** (4)	15.1 (4)	3.2 (4)	43.5 (4)	49.8 (4)	10.2 (4)	6.0 (4)
GPQA	**26.1** (5)	25.9 (5)	25.7 (5)	30.8 (5)	36.4 (5)	24.7 (5)	-
Code							
HumanEval	33.5 (0)	**34.2** (0)	12.8 (0)	51.2 (0)	57.9 (0)	29.3 (0)	26.2 (0)
HumanEval-FIM	**73.8** (2)	73.3 (2)	26.9 (2)	-	-	-	-
MBPP	38.2 (4)	**47.4** (4)	18.4 (4)	64.2 (0)	74.9 (0)	51.1 (0)	39.0 (3)
Chinese							
CMMLU	**69.9** (5)	50.7 (5)	32.5 (5)	83.9 (5)	-	-	47.2 (5)
C-Eval	**70.5** (5)	51.7 (5)	34.0 (5)	83.2 (5)	-	-	45.0 (5)

no error bars >:(

Experiment 3: benchmarks (with post-training)

- Less impressive?



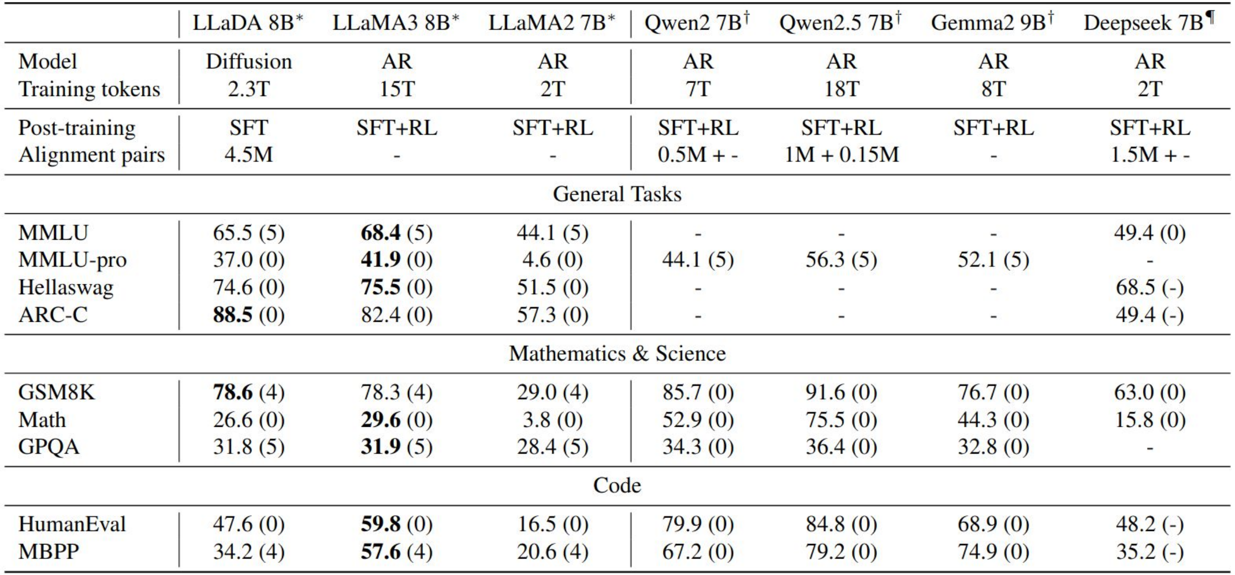
	LLaDA 8B*	LLaMA3 8B*	LLaMA2 7B*	Qwen2 7B†	Qwen2.5 7B†	Gemma2 9B†	Deepseek 7B¶
Model	Diffusion	AR	AR	AR	AR	AR	AR
Training tokens	2.3T	15T	2T	7T	18T	8T	2T
Post-training	SFT	SFT+RL	SFT+RL	SFT+RL	SFT+RL	SFT+RL	SFT+RL
Alignment pairs	4.5M	-	-	0.5M + -	1M + 0.15M	-	1.5M + -
General Tasks							
MMLU	65.5 (5)	**68.4** (5)	44.1 (5)	-	-	-	49.4 (0)
MMLU-pro	37.0 (0)	**41.9** (0)	4.6 (0)	44.1 (5)	56.3 (5)	52.1 (5)	-
Hellaswag	74.6 (0)	**75.5** (0)	51.5 (0)	-	-	-	68.5 (-)
ARC-C	**88.5** (0)	82.4 (0)	57.3 (0)	-	-	-	49.4 (-)
Mathematics & Science							
GSM8K	**78.6** (4)	78.3 (4)	29.0 (4)	85.7 (0)	91.6 (0)	76.7 (0)	63.0 (0)
Math	26.6 (0)	**29.6** (0)	3.8 (0)	52.9 (0)	75.5 (0)	44.3 (0)	15.8 (0)
GPQA	31.8 (5)	**31.9** (5)	28.4 (5)	34.3 (0)	36.4 (0)	32.8 (0)	-
Code							
HumanEval	47.6 (0)	**59.8** (0)	16.5 (0)	79.9 (0)	84.8 (0)	68.9 (0)	48.2 (-)
MBPP	34.2 (4)	**57.6** (4)	20.6 (4)	67.2 (0)	79.2 (0)	74.9 (0)	35.2 (-)

no error bars >:(

Experiment 3: benchmarks (with post-training)

- Less impressive?

- But note: LLaDA hasn’t had any RL (“*we leave RL-based alignment for future work*”)

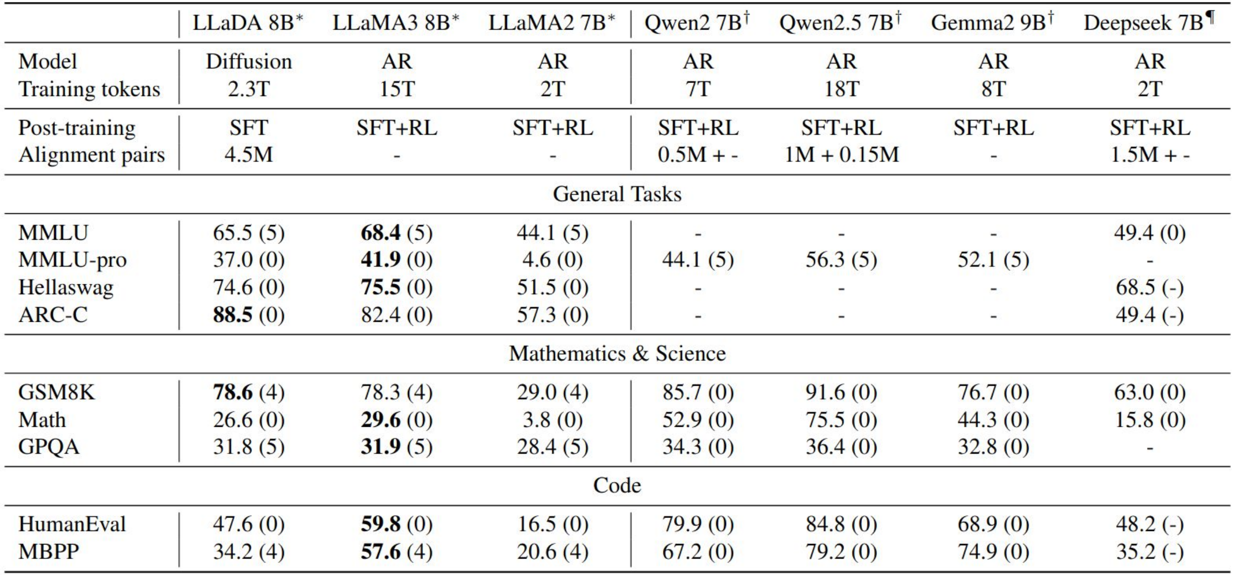


	LLaDA 8B*	LLaMA3 8B*	LLaMA2 7B*	Qwen2 7B†	Qwen2.5 7B†	Gemma2 9B†	Deepseek 7B¶
Model	Diffusion	AR	AR	AR	AR	AR	AR
Training tokens	2.3T	15T	2T	7T	18T	8T	2T
Post-training	SFT	SFT+RL	SFT+RL	SFT+RL	SFT+RL	SFT+RL	SFT+RL
Alignment pairs	4.5M	-	-	0.5M + -	1M + 0.15M	-	1.5M + -
General Tasks							
MMLU	65.5 (5)	**68.4** (5)	44.1 (5)	-	-	-	49.4 (0)
MMLU-pro	37.0 (0)	**41.9** (0)	4.6 (0)	44.1 (5)	56.3 (5)	52.1 (5)	-
Hellaswag	74.6 (0)	**75.5** (0)	51.5 (0)	-	-	-	68.5 (-)
ARC-C	**88.5** (0)	82.4 (0)	57.3 (0)	-	-	-	49.4 (-)
Mathematics & Science							
GSM8K	**78.6** (4)	78.3 (4)	29.0 (4)	85.7 (0)	91.6 (0)	76.7 (0)	63.0 (0)
Math	26.6 (0)	**29.6** (0)	3.8 (0)	52.9 (0)	75.5 (0)	44.3 (0)	15.8 (0)
GPQA	31.8 (5)	**31.9** (5)	28.4 (5)	34.3 (0)	36.4 (0)	32.8 (0)	-
Code							
HumanEval	47.6 (0)	**59.8** (0)	16.5 (0)	79.9 (0)	84.8 (0)	68.9 (0)	48.2 (-)
MBPP	34.2 (4)	**57.6** (4)	20.6 (4)	67.2 (0)	79.2 (0)	74.9 (0)	35.2 (-)

no error bars >:(

Experiment 3: benchmarks (with post-training)

- Less impressive?
- But note: LLaDA hasn't had any RL ("*we leave RL-based alignment for future work*")
- Underwhelming performance on coding tasks, which I thought might be easier for a non-ARM…?



	LLaDA 8B*	LLaMA3 8B*	LLaMA2 7B*	Qwen2 7B†	Qwen2.5 7B†	Gemma2 9B†	Deepseek 7B¶
Model	Diffusion	AR	AR	AR	AR	AR	AR
Training tokens	2.3T	15T	2T	7T	18T	8T	2T
Post-training	SFT	SFT+RL	SFT+RL	SFT+RL	SFT+RL	SFT+RL	SFT+RL
Alignment pairs	4.5M	-	-	0.5M + -	1M + 0.15M	-	1.5M + -
General Tasks							
MMLU	65.5 (5)	**68.4** (5)	44.1 (5)	-	-	-	49.4 (0)
MMLU-pro	37.0 (0)	**41.9** (0)	4.6 (0)	44.1 (5)	56.3 (5)	52.1 (5)	-
Hellaswag	74.6 (0)	**75.5** (0)	51.5 (0)	-	-	-	68.5 (-)
ARC-C	**88.5** (0)	82.4 (0)	57.3 (0)	-	-	-	49.4 (-)
Mathematics & Science							
GSM8K	**78.6** (4)	78.3 (4)	29.0 (4)	85.7 (0)	91.6 (0)	76.7 (0)	63.0 (0)
Math	26.6 (0)	**29.6** (0)	3.8 (0)	52.9 (0)	75.5 (0)	44.3 (0)	15.8 (0)
GPQA	31.8 (5)	**31.9** (5)	28.4 (5)	34.3 (0)	36.4 (0)	32.8 (0)	-
Code							
HumanEval	47.6 (0)	**59.8** (0)	16.5 (0)	79.9 (0)	84.8 (0)	68.9 (0)	48.2 (-)
MBPP	34.2 (4)	**57.6** (4)	20.6 (4)	67.2 (0)	79.2 (0)	74.9 (0)	35.2 (-)

Experiment 4: Reversal Curse fixed

Finetune on 496 famous Chinese poems.

Experiment 4: Reversal Curse fixed

Finetune on 496 famous Chinese poems.

Given a line from a poem, generate either the next or previous line (zero-shot).

Experiment 4: Reversal Curse fixed

Finetune on 496 famous Chinese poems.

Given a line from a poem, generate either the next or previous line (zero-shot).

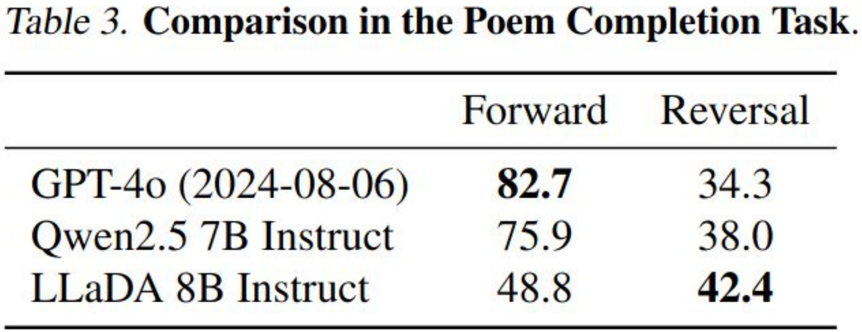


Table 3. **Comparison in the Poem Completion Task**.

	Forward	Reversal
GPT-4o (2024-08-06)	**82.7**	34.3
Qwen2.5 7B Instruct	75.9	38.0
LLaDA 8B Instruct	48.8	**42.4**

Experiment 4: Reversal Curse fixed

Finetune on 496 famous Chinese poems.

Given a line from a poem, generate either the next or previous line (zero-shot).

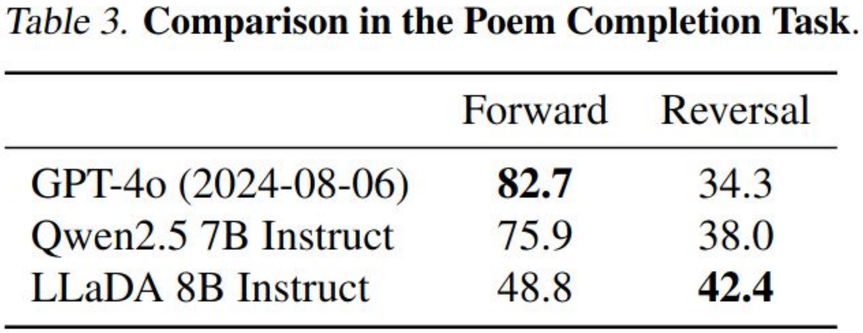


Table 3. **Comparison in the Poem Completion Task**.

	Forward	Reversal
GPT-4o (2024-08-06)	**82.7**	34.3
Qwen2.5 7B Instruct	75.9	38.0
LLaDA 8B Instruct	48.8	**42.4**

Why is it still worse at reversal?

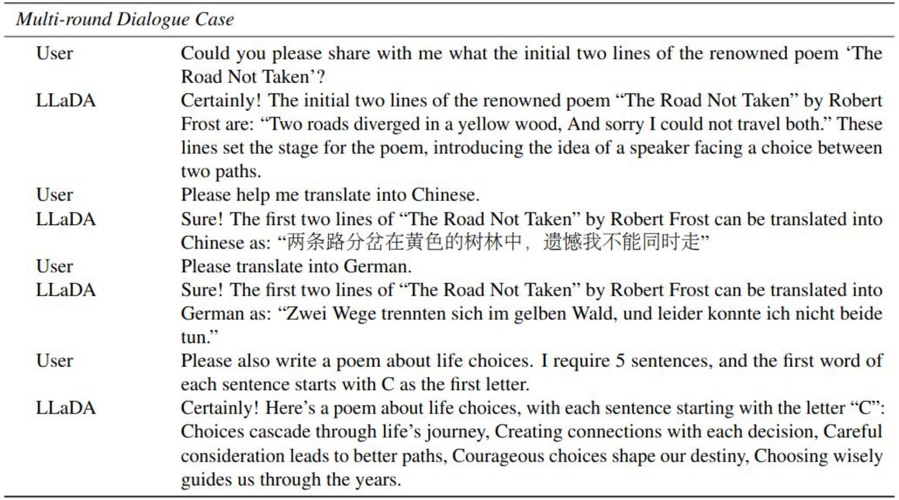
Conclusion

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

Conclusion

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

- Authors are keen to show that after finetuning it is semi-decent in conversation



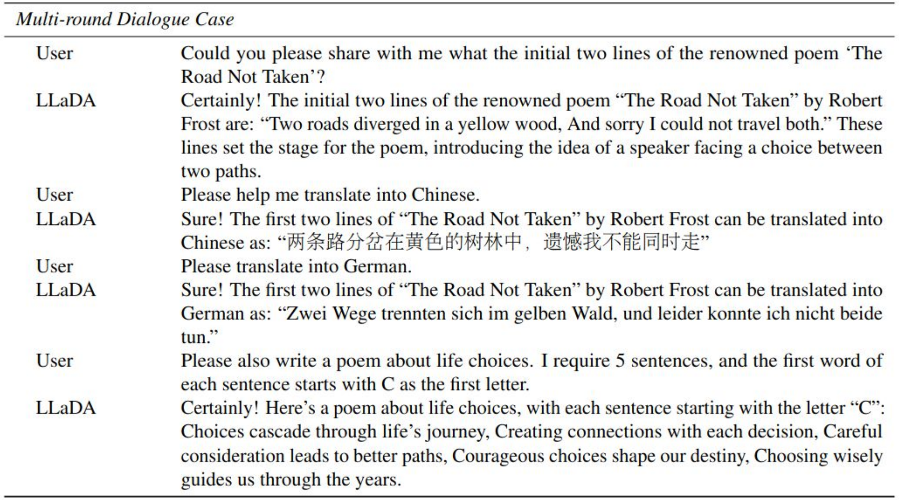
Multi-round Dialogue Case	
User	Could you please share with me what the initial two lines of the renowned poem ‘The Road Not Taken’?
LLaDA	Certainly! The initial two lines of the renowned poem “The Road Not Taken” by Robert Frost are: “Two roads diverged in a yellow wood, And sorry I could not travel both.” These lines set the stage for the poem, introducing the idea of a speaker facing a choice between two paths.
User	Please help me translate into Chinese.
LLaDA	Sure! The first two lines of “The Road Not Taken” by Robert Frost can be translated into Chinese as: “两条路分岔在黄色的树林中，遗憾我不能同时走”
User	Please translate into German.
LLaDA	Sure! The first two lines of “The Road Not Taken” by Robert Frost can be translated into German as: “Zwei Wege trennten sich im gelben Wald, und leider konnte ich nicht beide tun.”
User	Please also write a poem about life choices. I require 5 sentences, and the first word of each sentence starts with C as the first letter.
LLaDA	Certainly! Here’s a poem about life choices, with each sentence starting with the letter “C”: Choices cascade through life’s journey, Creating connections with each decision, Careful consideration leads to better paths, Courageous choices shape our destiny, Choosing wisely guides us through the years.

Conclusion

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

- Authors are keen to show that after finetuning it is semi-decent in conversation

- Unlikely that many people other than these authors (who clearly have tons of compute) will be willing to keep scaling these up



Multi-round Dialogue Case	
User	Could you please share with me what the initial two lines of the renowned poem ‘The Road Not Taken’?
LLaDA	Certainly! The initial two lines of the renowned poem “The Road Not Taken” by Robert Frost are: “Two roads diverged in a yellow wood, And sorry I could not travel both.” These lines set the stage for the poem, introducing the idea of a speaker facing a choice between two paths.
User	Please help me translate into Chinese.
LLaDA	Sure! The first two lines of “The Road Not Taken” by Robert Frost can be translated into Chinese as: “两条路分岔在黄色的树林中，遗憾我不能同时走”
User	Please translate into German.
LLaDA	Sure! The first two lines of “The Road Not Taken” by Robert Frost can be translated into German as: “Zwei Wege trennten sich im gelben Wald, und leider konnte ich nicht beide tun.”
User	Please also write a poem about life choices. I require 5 sentences, and the first word of each sentence starts with C as the first letter.
LLaDA	Certainly! Here’s a poem about life choices, with each sentence starting with the letter “C”: Choices cascade through life’s journey, Creating connections with each decision, Careful consideration leads to better paths, Courageous choices shape our destiny, Choosing wisely guides us through the years.

References

Austin, Jacob, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. 2023. ‘Structured Denoising Diffusion Models in Discrete State-Spaces’. arXiv. https://doi.org/10.48550/arXiv.2107.03006.

Delétang, Grégoire, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, et al. 2024. ‘Language Modeling Is Compression’. arXiv. https://doi.org/10.48550/arXiv.2309.10668.

Li, Xiang Lisa, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. 2022. ‘Diffusion-LM Improves Controllable Text Generation’. arXiv. https://doi.org/10.48550/arXiv.2205.14217.

Lin, Zhenghao, Yeyun Gong, Yelong Shen, Tong Wu, Zhihao Fan, Chen Lin, Nan Duan, and Weizhu Chen. 2023. ‘Text Generation with Diffusion Language Models: A Pre-Training Approach with Continuous Paragraph Denoise’. arXiv. https://doi.org/10.48550/arXiv.2212.11685.

Lou, Aaron, Chenlin Meng, and Stefano Ermon. 2024. ‘Discrete Diffusion Modeling by Estimating the Ratios of the Data Distribution’. arXiv. https://doi.org/10.48550/arXiv.2310.16834.

Nie, Shen, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. 2024. ‘Scaling up Masked Diffusion Models on Text’. arXiv. https://doi.org/10.48550/arXiv.2410.18514.

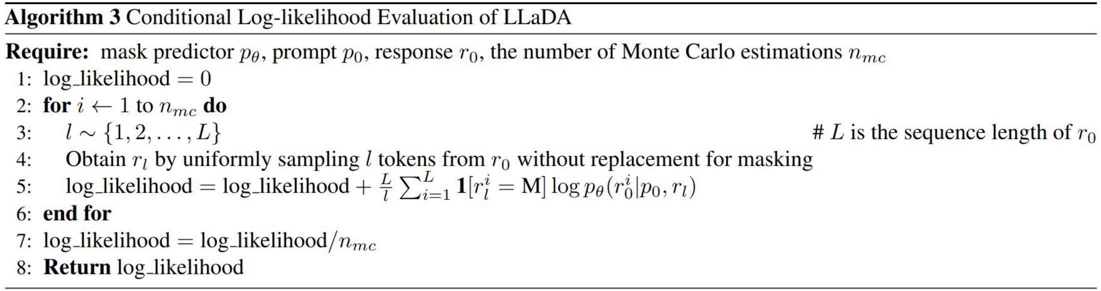
Nie, Shen, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2025. ‘Large Language Diffusion Models’. arXiv. https://doi.org/10.48550/arXiv.2502.09992.

Ye, Jiasheng, Zaixiang Zheng, Yu Bao, Lihua Qian, and Mingxuan Wang. 2024. ‘DINOISER: Diffused Conditional Sequence Learning by Manipulating Noises’. arXiv. https://doi.org/10.48550/arXiv.2302.10025.

Conditional Likelihood Evaluation

Also actually use this more stable loss function when doing conditional likelihood evaluation (rather than text generation), where $l \sim \mathrm{Uniform}\{1, 2, \ldots, L\}$

$$-\mathbb{E}_{l, r_0, r_l}\left[\frac{L}{l}\sum_{i=1}^{L}\mathbf{1}[r_l^i = \mathbf{M}]\log p_\theta(r_0^i | p_0, r_l)\right]$$



Algorithm 3 Conditional Log-likelihood Evaluation of LLaDA

```
Require: mask predictor $p_\theta$, prompt $p_0$, response $r_0$, the number of Monte Carlo estimations $n_{mc}$
log_likelihood $= 0$
for $i \leftarrow 1$ to $n_{mc}$ do
    $l \sim \{1, 2, \ldots, L\}$                                  # $L$ is the sequence length of $r_0$
    Obtain $r_l$ by uniformly sampling $l$ tokens from $r_0$ without replacement for masking
    log_likelihood $=$ log_likelihood $+ \frac{L}{l}\sum_{i=1}^{L}\mathbf{1}[r_l^i = \mathrm{M}]\log p_\theta(r_0^i | p_0, r_l)$
end for
log_likelihood $=$ log_likelihood$/n_{mc}$
Return log_likelihood
```

Comparison Against DeepSeekV3 Training

- DeepSeek V3 – 617B MoE (37B per token)
    - 15T tokens,
    - 2.8 million H800 hours
    - “~$5.6M”
- LLaDA 8B
    - 2.3T tokens
    - 0.13 million H800 GPU hours
    - ~$0.26M?