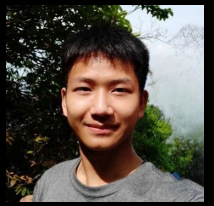
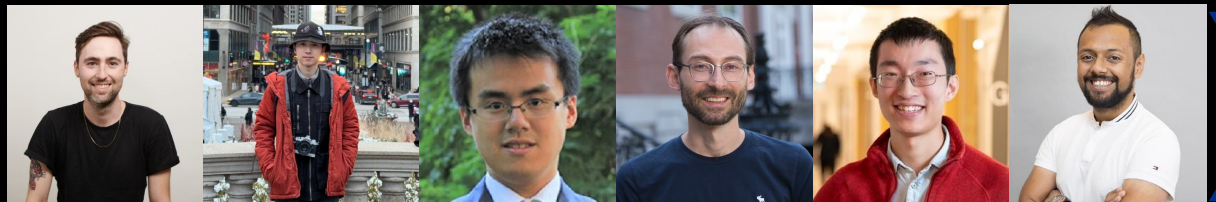
Planner Aware Path Learning in Diffusion Language Models Training



Zachary Bezemek, Fred Zhangzhi Peng,
Jarrid Rector-Brooks, Shuibai Zhang, Anru R. Zhang, Michael Bronstein,
Alexander Tong, Avishek Joey Bose

zwb@duke.edu
arXiv:2509.23405
github.com/pengzhangzhi/PAPL



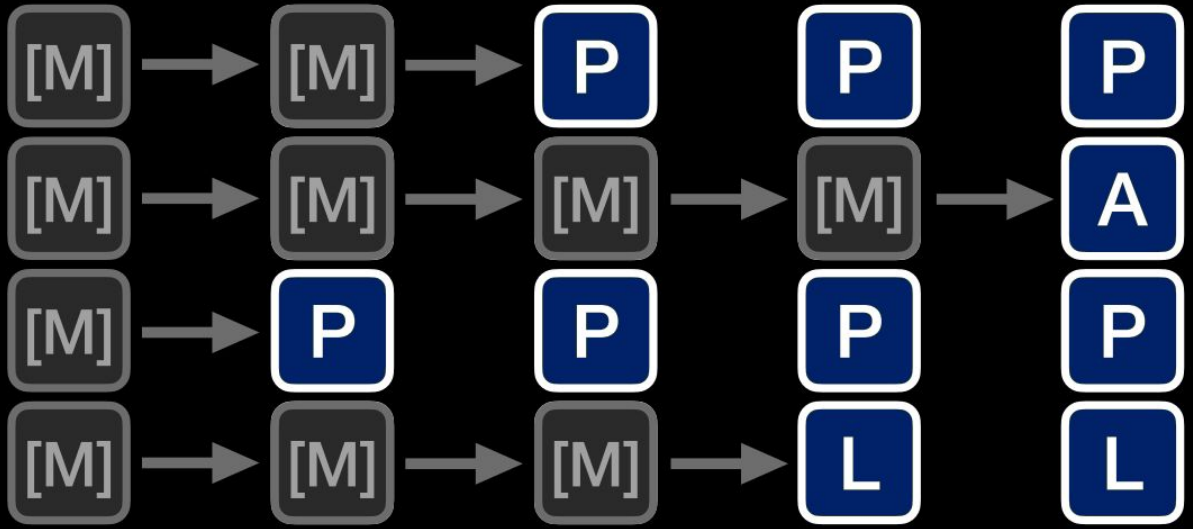
Overview

- MDLMs (4 slides)
  - Motivation (1 slide)
  - Equivalence with discrete-time Markov chain/**discrete time generator matching** (2 slides)
  - Necessitation of planning (1 slide)
- PAPL (10 slides)
  - **Planning definition** (1 slide)
  - **Objective alignment** (3 slides)
  - Form of the loss (1 slide)
  - Greedy planner specialization and ablations (1 slide)
  - Code (1 slide)
  - Performance (1 slide)
  - Takeaways (2 slides)

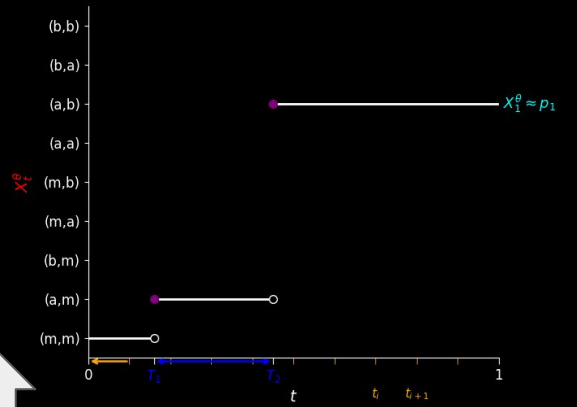
The result: A **two-line** modification to MDLM training loss which **boosts performance across all tested tasks** (protein, code).

Why discrete diffusion?

- SOTA in discrete generation tasks for domains that lack a natural causal ordering, such as biological sequence and code generation.
- Allows for tasks requiring bidirectional context vs autoregressive modeling.



Vanilla MDMs = AOARMs

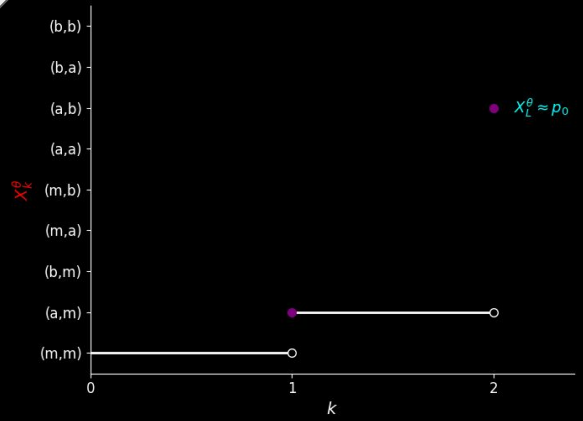


= (jump times don't affect jump location)

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{x_0 \sim p_0} \left[\int_0^1 \frac{\alpha_t'}{1-\alpha_t} \underbrace{\mathop{\mathbb{E}}_{y \sim p_t(\cdot|x_0)}} \left[\sum_{i=1, y^i = M}^{L} \log D_\theta^i(y)[x_0^i] \right] dt \right]$$

mask each coord of x_0 indep w/ prob $1-\alpha_t$

= (integrate)



$$\mathcal{L}(\theta) = - \mathop{\mathbb{E}}_{x_0 \sim p_0} \left[\sum_{k=0}^{L-1} \underbrace{\mathop{\mathbb{E}}_{y \sim \mathcal{L}(Y_k^{x_0})}} \left[\frac{1}{L-k} \sum_{i: y^i = M} \log \left(D_\theta^i(y)[x_0^i] \right) \right] \right]$$

mask L-k positions of x_0 unif at random

Vanilla MDMs as Discrete-Time Generator Matching

- Given: Samples $x_0 \in \mathcal{V}^L$ from p_0, $\mathcal{V}$ a finite vocabulary.
- Want: X^θ a **discrete time Markov chain** with $X_0^\theta = (M, \dots, M), X_L^\theta \approx p_0$, $M \notin \mathcal{V}$.

Vanilla MDMs as Discrete-Time Generator Matching

- Given: Samples $x_0 \in \mathcal{V}^L$ from p_0, $\mathcal{V}$ a finite vocabulary.
- Want: X^θ a **discrete time Markov chain** with $X_0^\theta = (M, \dots, M), X_L^\theta \approx p_0$, $M \notin \mathcal{V}$.
- Choose conditional reference DTMC: Y^{x_0} with $\mathbb{P}(Y_L^{x_0} = x_0) = 1$, **unmask 1 token at a time** according to, for $x^i = M$:

$$\mathbb{P}\left(Y_{k+1}^{x_0,i} = y^i | Y_k^{x_0} = x\right) = \mathbb{P}(\text{unmask to } y^i | \text{denoise } i\text{'th token}) \mathbb{P}(\text{denoise } i\text{'th token}) = \delta_{x_0^i}(y^i)/N_M(x)$$

$$N_M(x) = \ \# \text{ of masks in } x \text{ (uniformly random index selection)}$$

Vanilla MDMs as Discrete-Time Generator Matching

- Given: Samples $x_0 \in \mathcal{V}^L$ from p_0, $\mathcal{V}$ a finite vocabulary.
- Want: X^θ a **discrete time Markov chain** with $X_0^\theta = (M, \dots, M), X_L^\theta \approx p_0$, $M \notin \mathcal{V}$.
- Choose conditional reference DTMC: Y^{x_0} with $\mathbb{P}(Y_L^{x_0} = x_0) = 1$, **unmask 1 token at a time** according to, for $x^i = M$:

$$\mathbb{P}\left(Y_{k+1}^{x_0,i} = y^i | Y_k^{x_0} = x\right) = \mathbb{P}(\text{unmask to } y^i | \text{denoise } i\text{'th token}) \mathbb{P}(\text{denoise } i\text{'th token}) = \delta_{x_0^i}(y^i)/N_M(x)$$

$$N_M(x) = \ \# \text{ of masks in } x \text{ (uniformly random index selection)}$$

- Yields oracle DTMC: Y with $Y_L \sim p_0$

$$\mathbb{P}(Y_{k+1}^i = y^i | Y_k = x) = \mathop{\mathbb{E}}_{x_0 \sim p_0(\cdot | x_{\neq M})} \left[\mathbb{P}\left(Y_{k+1}^{x_0,i} = y^i | Y_k^{x_0} = x\right)\right] = p_0^i(y^i | x_{\neq M})/N_M(x)$$

Vanilla MDMs as Discrete-Time Generator Matching

- Given: Samples $x_0 \in \mathcal{V}^L$ from p_0, $\mathcal{V}$ a finite vocabulary.
- Want: X^θ a **discrete time Markov chain** with $X_0^\theta = (M, \dots, M), X_L^\theta \approx p_0$, $M \notin \mathcal{V}$.
- Choose conditional reference DTMC: Y^{x_0} with $\mathbb{P}(Y_L^{x_0} = x_0) = 1$, **unmask 1 token at a time** according to, for $x^i = M$:

$$\mathbb{P}\left(Y_{k+1}^{x_0,i} = y^i | Y_k^{x_0} = x\right) = \mathbb{P}(\text{unmask to } y^i | \text{denoise } i\text{'th token}) \mathbb{P}(\text{denoise } i\text{'th token}) = \delta_{x_0^i}(y^i)/N_M(x)$$

$$N_M(x) = \ \#\text{ of masks in } x \text{ (uniformly random index selection)}$$

- Yields oracle DTMC: Y with $Y_L \sim p_0$

$$\mathbb{P}(Y_{k+1}^i = y^i | Y_k = x) = \mathop{\mathbb{E}}_{x_0 \sim p_0(\cdot | x_{\neq M})} \left[\mathbb{P}\left(Y_{k+1}^{x_0,i} = y^i | Y_k^{x_0} = x\right)\right] = p_0^i(y^i | x_{\neq M})/N_M(x)$$

- Choose sampling DTMC:

$$\mathbb{P}(X_{k+1}^{\theta,i} = y^i | X_k^\theta = x) = D_\theta^i(x)[y^i]/N_M(x)$$

Vanilla MDMs as Discrete-Time Generator Matching

- Given: Samples $x_0 \in \mathcal{V}^L$ from p_0, $\mathcal{V}$ a finite vocabulary.
- Want: X^θ a **discrete time Markov chain** with $X_0^\theta = (M, \dots, M), X_L^\theta \approx p_0$, $M \notin \mathcal{V}$.
- Choose conditional reference DTMC: Y^{x_0} with $\mathbb{P}(Y_L^{x_0} = x_0) = 1$, **unmask 1 token at a time** according to, for $x^i = M$:

$$\mathbb{P}\left(Y_{k+1}^{x_0,i} = y^i | Y_k^{x_0} = x\right) = \mathbb{P}(\text{unmask to } y^i | \text{denoise } i\text{'th token}) \mathbb{P}(\text{denoise } i\text{'th token}) = \delta_{x_0^i}(y^i)/N_M(x)$$

$$N_M(x) = \ \#\text{ of masks in } x \text{ (uniformly random index selection)}$$

- Yields oracle DTMC: Y with $Y_L \sim p_0$

$$\mathbb{P}(Y_{k+1}^i = y^i | Y_k = x) = \mathop{\mathbb{E}}_{x_0 \sim p_0(\cdot | x_{\neq M})} \left[\mathbb{P}\left(Y_{k+1}^{x_0,i} = y^i | Y_k^{x_0} = x\right)\right] = p_0^i(y^i | x_{\neq M})/N_M(x)$$

- Choose sampling DTMC:

$$\mathbb{P}(X_{k+1}^{\theta,i} = y^i | X_k^\theta = x) = D_\theta^i(x)[y^i]/N_M(x)$$

- Choice Y^{x_0} determines oracle Y, with choice X^θ determines:

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{x_0 \sim p_0}\left[D_{KL}(Y^{x_0} \text{ paths} || X^\theta \text{ paths})\right] \overset{\nabla_\theta}{=} D_{KL}(Y \text{ paths} || X^\theta \text{ paths}) \geq D_{KL}(Y_L || X_L^\theta) = D_{KL}(p_0 || X_L^\theta)$$

Vanilla MDMs as Discrete-Time Generator Matching

- Given: Samples $x_0 \in \mathcal{V}^L$ from p_0, $\mathcal{V}$ a finite vocabulary.
- Want: X^θ a **discrete time Markov chain** with $X_0^\theta = (M, \dots, M), X_L^\theta \approx p_0$, $M \notin \mathcal{V}$.
- Choose conditional reference DTMC: Y^{x_0} with $\mathbb{P}(Y_L^{x_0} = x_0) = 1$, **unmask 1 token at a time** according to, for $x^i = M$:

$$\mathbb{P}\left(Y_{k+1}^{x_0,i} = y^i | Y_k^{x_0} = x\right) = \mathbb{P}(\text{unmask to } y^i | \text{denoise } i\text{'th token}) \mathbb{P}(\text{denoise } i\text{'th token}) = \delta_{x_0^i}(y^i)/N_M(x)$$

$$N_M(x) = \ \#\text{ of masks in } x \text{ (uniformly random index selection)}$$

- Yields oracle DTMC: Y with $Y_L \sim p_0$

$$\mathbb{P}(Y_{k+1}^i = y^i | Y_k = x) = \mathop{\mathbb{E}}_{x_0 \sim p_0(\cdot | x_{\neq M})} \left[\mathbb{P}\left(Y_{k+1}^{x_0,i} = y^i | Y_k^{x_0} = x\right)\right] = p_0^i(y^i | x_{\neq M})/N_M(x)$$

- Choose sampling DTMC:

$$\mathbb{P}(X_{k+1}^{\theta,i} = y^i | X_k^\theta = x) = D_\theta^i(x)[y^i]/N_M(x)$$

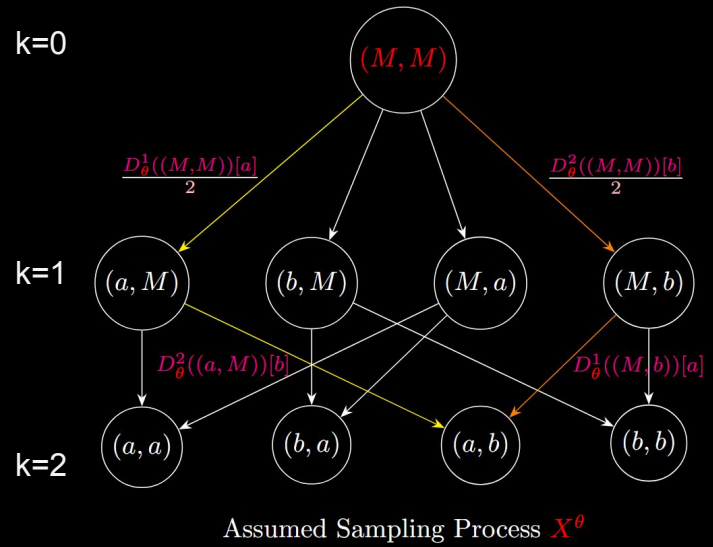
- Choice Y^{x_0} determines oracle Y, with choice X^θ determines:

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{x_0 \sim p_0}\left[D_{KL}(Y^{x_0} \text{ paths} || X^\theta \text{ paths})\right] \overset{\nabla_\theta}{=} D_{KL}(Y \text{ paths} || X^\theta \text{ paths}) \geq D_{KL}(Y_L || X_L^\theta) = D_{KL}(p_0 || X_L^\theta)$$

- For vanilla MDMs, implicit choices made are Y^{x_0} (so also Y) and X^θ select masked positions uniformly at random

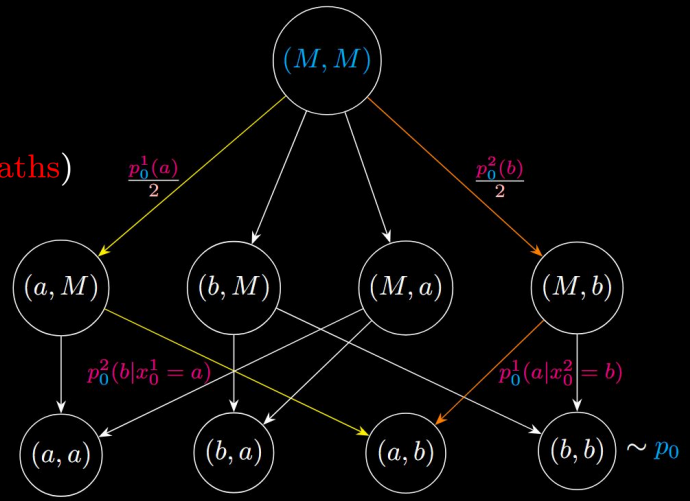
Why plan?

E.g. $\mathcal{V} = \{a, b\}, L = 2$:



Assumed Sampling Process X^θ

$$\mathcal{L}(\theta) \overset{\nabla_\theta}{=} D_{KL}(Y \text{ paths} || X^\theta \text{ paths})$$

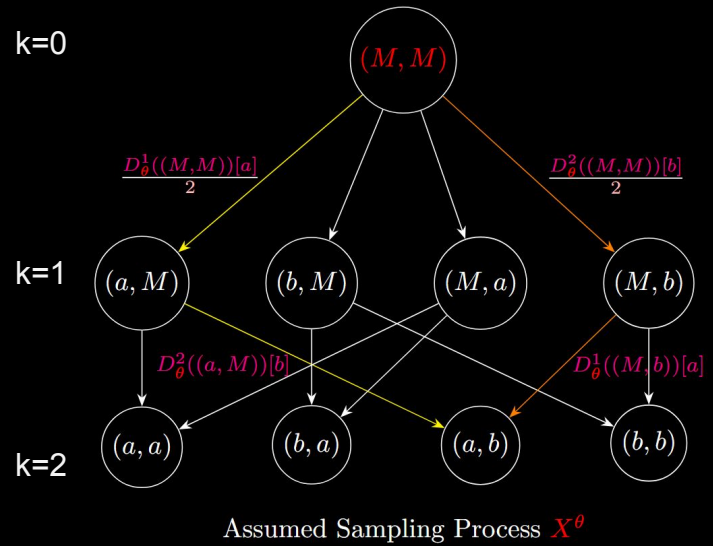


Reference Process Y

- For reference: $\mathbb{P}(\text{path1}) = \mathbb{P}(\text{path2}) = p_0((a, b))/2$, so sum $= p_0((a, b))$ ✓
- For generation, maybe: $\mathbb{P}(\text{path1}) \not\approx \mathbb{P}(\text{path2}) \approx p_0((a, b))/2$, so sum $\not\approx p_0((a, b))$ ×

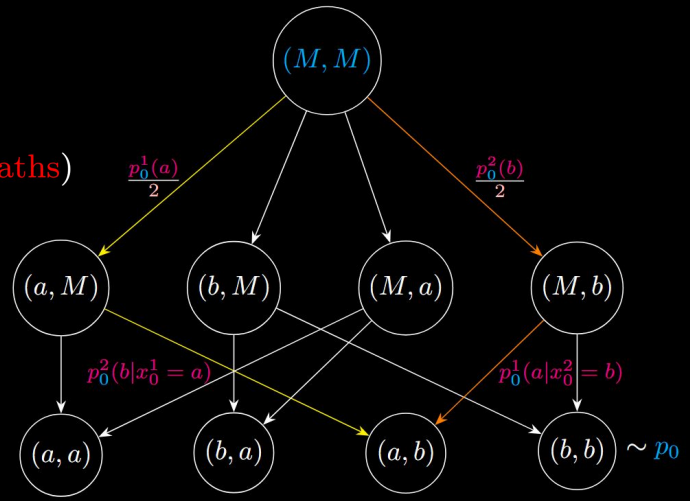
E.g. $\mathcal{V} = \{a, b\}, L = 2$:

Why plan?



Assumed Sampling Process X^θ

$$\mathcal{L}(\theta) \overset{\nabla_\theta}{=} D_{KL}(Y \text{ paths} || X^\theta \text{ paths})$$



Reference Process Y

- For reference: $\mathbb{P}(\text{path1}) = \mathbb{P}(\text{path2}) = p_0((a, b))/2$, so sum $= p_0((a, b))$ ✓
- For generation, maybe: $\mathbb{P}(\text{path1}) \not\approx \mathbb{P}(\text{path2}) \approx p_0((a, b))/2$, so sum $\not\approx p_0((a, b))$ ×

- Inference-time solution (Planning): Change sampling alg. to take path2 with high prob.
- But now **your loss has nothing to do** with the distribution of the new X^θ_L and its closeness in to p_0!

Gillespie Sampler for Masked Diffusion Models

X^θ

Initialize: $\mathbf{x}_0 \leftarrow (\mathbf{m}, \mathbf{m}, \ldots, \mathbf{m})$, denoiser D_θ
for $k = 0 : L-1$ **do**
  Choose Random Coordinate for Unmasking:
  Sample dimension $i \sim \text{Unif}\left(\mathcal{M}(\mathbf{x}_k)\right)$
  Denoise:
  Sample $z^i \sim D_\theta^i(\mathbf{x}_k)$
  $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k$
  $x_{k+1}^i \leftarrow z^i$
end for
return x_L

Plan

What is planning?

Gillespie Sampler for Masked Diffusion Models X^θ

Initialize: $\mathbf{x}_0 \leftarrow (\mathbf{m}, \mathbf{m}, \ldots, \mathbf{m})$, denoiser D_θ
for $k = 0 : L-1$ **do**
  Choose Random Coordinate for Unmasking:
  Sample dimension $i \sim \text{Unif}\left(\mathcal{M}(\mathbf{x}_k)\right)$
  Denoise:
  Sample $z^i \sim D_\theta^i(\mathbf{x}_k)$
  $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k$
  $x_{k+1}^i \leftarrow z^i$
end for
return x_L

Plan

Gillespie Sampler with a Planner X^G

Initialize: $\mathbf{x}_0 \leftarrow (m, \ldots, m)$, planner
$G : \mathcal{V}^L \times \mathcal{V}^L \to \mathcal{P}(\mathcal{M}(\mathbf{x}_k))$, denoiser D_θ
for $k = 0 : L-1$ **do**
  Plan Sample $\mathbf{z} \sim D_\theta(\mathbf{x}_k)$
  Sample dimension $i \sim G(\mathbf{z}, \mathbf{x}_k)$
  Denoise
  $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k$
  $x_{k+1}^i \leftarrow z^i$
end for
return $\mathbf{x}_L$

What is planning?

E.g. Greedy Ancestral:

$$G(\mathbf{z}, \mathbf{x}) = \delta\left(\underset{j : x_k^j = \mathbf{m}}{\text{argmax}} D_\theta^j(\mathbf{x})[z^j]\right)$$

Gillespie Sampler for Masked Diffusion Models X^θ

Initialize: $\mathbf{x}_0 \leftarrow (\mathbf{m}, \mathbf{m}, \ldots, \mathbf{m})$, denoiser D_θ
for $k = 0 : L-1$ **do**
  Choose Random Coordinate for Unmasking:
  Sample dimension $i \sim \text{Unif}\left(\mathcal{M}(\mathbf{x}_k)\right)$
  Denoise:
  Sample $z^i \sim D_\theta^i(\mathbf{x}_k)$
  $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k$
  $x_{k+1}^i \leftarrow z^i$
end for
return x_L

Plan

Gillespie Sampler with a Planner X^G

Initialize: $\mathbf{x}_0 \leftarrow (m, \ldots, m)$, planner
$G : \mathcal{V}^L \times \mathcal{V}^L \to \mathcal{P}(\mathcal{M}(\mathbf{x}_k))$, denoiser D_θ
for $k = 0 : L-1$ **do**
  Plan Sample $\mathbf{z} \sim D_\theta(\mathbf{x}_k)$
  Sample dimension $i \sim G(\mathbf{z}, \mathbf{x}_k)$
  Denoise
  $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k$
  $x_{k+1}^i \leftarrow z^i$
end for
return $\mathbf{x}_L$

What is planning?

E.g. Greedy Ancestral:

$$G(\mathbf{z}, \mathbf{x}) = \delta\left(\underset{j:x_k^j=\mathbf{m}}{\text{argmax}} D_\theta^j(\mathbf{x})[z^j]\right)$$

Used Sampling: $\mathbb{P}(X_{k+1}^{G,i} = y^i | X_k^G = x)$

$$= \mathop{\mathbb{E}}_{z\sim D_\theta(x)} \left[\delta_{z^i}(y^i) G(z,x)[i]\right]$$

$$= D_\theta^i(x)[y^i] \overbrace{\mathop{\mathbb{E}}_{z\sim D_\theta(x)} [G(z^{-i,y^i}, x)[i]]}^{F_\theta(x,y^i,i)}$$

Gillespie Sampler for Masked Diffusion Models X^θ

Initialize: $\mathbf{x}_0 \leftarrow (\mathbf{m}, \mathbf{m}, \ldots, \mathbf{m})$, denoiser D_θ
for $k = 0 : L-1$ **do**
  Choose Random Coordinate for Unmasking:
  Sample dimension $i \sim \text{Unif}\left(\mathcal{M}(\mathbf{x}_k)\right)$
  Denoise:
  Sample $z^i \sim D_\theta^i(\mathbf{x}_k)$
  $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k$
  $x_{k+1}^i \leftarrow z^i$
end for
return x_L

Plan

Gillespie Sampler with a Planner X^G

Initialize: $\mathbf{x}_0 \leftarrow (m, \ldots, m)$, planner
$G : \mathcal{V}^L \times \mathcal{V}^L \to \mathcal{P}(\mathcal{M}(\mathbf{x}_k))$, denoiser D_θ
for $k = 0 : L-1$ **do**
  Plan Sample $\mathbf{z} \sim D_\theta(\mathbf{x}_k)$
  Sample dimension $i \sim G(\mathbf{z}, \mathbf{x}_k)$
  Denoise
  $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k$
  $x_{k+1}^i \leftarrow z^i$
end for
return $\mathbf{x}_L$

What is planning?

E.g. Greedy Ancestral:

$$G(\mathbf{z}, \mathbf{x}) = \delta\left(\underset{j: x_k^j = \mathbf{m}}{\text{argmax}} D_\theta^j(\mathbf{x})[z^j]\right)$$

Used Sampling: $\mathbb{P}(X_{k+1}^{G,i} = y^i | X_k^G = x)$

$$= \mathop{\mathbb{E}}_{z \sim D_\theta(x)} \left[\delta_{z^i}(y^i) G(z, x)[i]\right]$$

$$= D_\theta^i(x)[y^i] \overbrace{\mathop{\mathbb{E}}_{z \sim D_\theta(x)} [G(z^{-i, y^i}, x)[i]]}^{F_\theta(x, y^i, i)}$$

$$\neq \mathbb{P}(X_{k+1}^{\theta,i} = y^i | X_k^\theta = x)$$

$$= D_\theta^i(x)[y^i] / N_M(x)$$

But the loss is designed assuming the sampling strategy of X^θ !

What's the problem?

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{x_0 \sim p_0} \left[D_{KL}(Y^{x_0} \text{ paths} || X^{\theta} \text{ paths}) \right] \overset{\nabla_\theta}{=} D_{KL}(Y \text{ paths} || X^{\theta} \text{ paths}) \geq D_{KL}(p_0 || X_L^{\theta})$$

What's the problem?

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{x_0 \sim p_0} \left[D_{KL}(Y^{x_0} \text{ paths} || X^{\theta} \text{ paths}) \right] \overset{\nabla_\theta}{=} D_{KL}(Y \text{ paths} || X^{\theta} \text{ paths}) \geq D_{KL}(p_0 || X_L^{\theta})$$

No longer holds if X_L^{θ} replaced by X_L^{G} (train-inference mismatch)

What's the problem?

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{x_0 \sim p_0} \left[D_{KL}(Y^{x_0} \text{ paths} || X^{\theta} \text{ paths}) \right] \overset{\nabla_\theta}{=} D_{KL}(Y \text{ paths} || X^{\theta} \text{ paths}) \geq D_{KL}(p_0 || X_L^{\theta})$$

No longer holds if X_L^{θ} replaced by X_L^{G} (train-inference mismatch)

Could just replace X^{θ} with X^{G}, but then still training towards paths of Y (uniformly random selection)

How to fix it?

- Assumed random order sampling X^{θ}

- Chosen random order conditional reference Y^{x_0}

- Random order oracle Y

How to fix it?

- Assumed random order sampling X^{θ}

Used Planned Sampling: $\mathbb{P}(X_{k+1}^{G,i} = y^i | X_k^G = x) = D_\theta^i(x)[y^i] F_\theta(x, y^i, i)$

- Chosen random order conditional reference Y^{x_0}

- Random order oracle Y

How to fix it?

- Assumed random order sampling X^{θ}

Used Planned Sampling: $\mathbb{P}(X_{k+1}^{G,i} = y^i | X_k^G = x) = D_\theta^i(x)[y^i] F_\theta(x, y^i, i)$

- Chosen random order conditional reference Y^{x_0}

Planned conditional reference: $\mathbb{P}(Y_{k+1}^{x_0,G,i} = y^i | Y_k^{x_0,G} = x) = \delta_{x_0^i}(y^i) G(x_0, x)[i]$

- Random order oracle Y

How to fix it?

- Assumed random order sampling X^{θ}

Used Planned Sampling: $\mathbb{P}(X_{k+1}^{G,i} = y^i | X_k^G = x) = D_\theta^i(x)[y^i] F_\theta(x, y^i, i)$

- Chosen random order conditional reference Y^{x_0}

Planned conditional reference: $\mathbb{P}(Y_{k+1}^{x_0,G,i} = y^i | Y_k^{x_0,G} = x) = \delta_{x_0^i}(y^i) G(x_0, x)[i]$

- Random order oracle Y

Planned oracle: $\mathbb{P}(Y_{k+1}^{G,i} = y^i | Y_k^G = x) = p_0^i(y^i | Y_k^{\cdot,G} = x) F_0(x, y^i, i), \quad Y_L^G \sim p_0$

A principled, planner-informed loss

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{x_0 \sim p_0} \left[D_{KL}(Y^{x_0} \text{ paths} || X^{\theta} \text{ paths}) \right] \overset{\nabla_\theta}{=} D_{KL}(Y \text{ paths} || X^{\theta} \text{ paths}) \geq D_{KL}(p_0 || X_L^{\theta})$$

No longer holds if X_L^{θ} replaced by X_L^{G} (train-inference mismatch)

A principled, planner-informed loss

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{x_0 \sim p_0} \left[D_{KL}(Y^{x_0} \text{ paths} || X^\theta \text{ paths}) \right] \overset{\nabla_\theta}{=} D_{KL}(Y \text{ paths} || X^\theta \text{ paths}) \geq D_{KL}(p_0 || X_L^\theta)$$

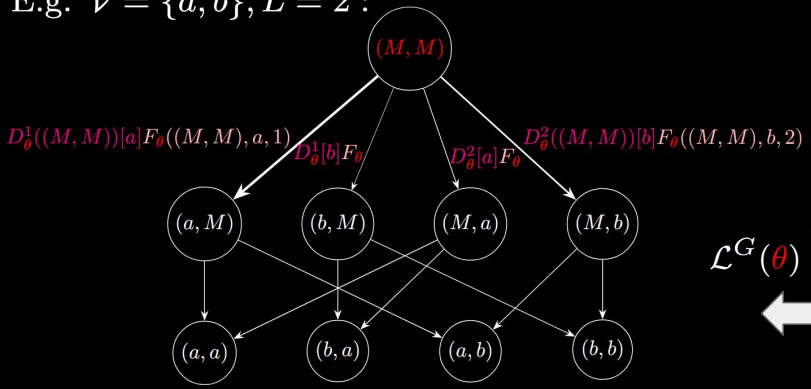
×

No longer holds if X_L^θ replaced by X_L^G (train-inference mismatch)

$$\mathcal{L}^G(\theta) = \mathop{\mathbb{E}}_{x_0 \sim p_0} \left[D_{KL}(Y^{x_0, G} \text{ paths} || X^G \text{ paths}) \right] \overset{\nabla_\theta}{=} D_{KL}(Y^G \text{ paths} || X^G \text{ paths}) \geq D_{KL}(p_0 || X_L^G)$$

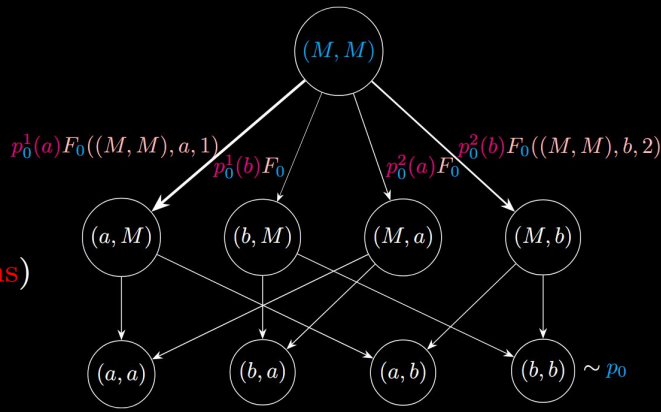
✓ ✓

E.g. $\mathcal{V} = \{a, b\}, L = 2$:



Used sampling process X^G

$\mathcal{L}^G(\theta) \overset{\nabla_\theta}{=} D_{KL}(Y^G \text{ paths} || X^G \text{ paths})$



Planned reference process Y^G

Vanilla (uniform random) MDM Loss

$$\mathcal{L}(\theta) \overset{\nabla_\theta}{=} D_{KL}(Y \text{ paths} || X^\theta \text{ paths})$$

Weight masked positions of partially masked data by 1/(# masks) = 1/(L-k)

$$\mathcal{L}(\theta) = - \mathop{\mathbb{E}}_{x_0 \sim p_0} \left[\sum_{k=0}^{L-1} \underbrace{\mathop{\mathbb{E}}_{y \sim \mathcal{L}(Y_k^{x_0})}} \left[\overbrace{\frac{1}{N_M(y)}} \sum_{i : y^i = M} \log \left(D_\theta^i(y)[x_0^i] \right) \right] \right]$$

Mask L-k positions of x_0 unif at random

Vanilla (uniform random) MDM Loss

$$\mathcal{L}(\theta) \overset{\nabla_\theta}{=} D_{KL}(Y \text{ paths} || X^\theta \text{ paths})$$

Weight masked positions of partially masked data by 1/(# masks) = 1/(L-k)

$$\mathcal{L}(\theta) = - \mathop{\mathbb{E}}_{x_0 \sim p_0} \left[\sum_{k=0}^{L-1} \mathop{\mathbb{E}}_{y \sim \mathcal{L}(Y_k^{x_0})} \left[\overbrace{\frac{1}{N_M(y)}} \sum_{i : y^i = M} \log \left(D_\theta^i(y)[x_0^i] \right) \right] \right]$$

Mask L-k positions of x_0 unif at random

Planner-Based PAPL Loss

$$\mathcal{L}^G(\theta) \overset{\nabla_\theta}{=} D_{KL}(Y^G \text{ paths} || X^G \text{ paths})$$

Match sampling planner to data planner

$$\mathcal{L}^G(\theta) = - \mathop{\mathbb{E}}_{x_0 \sim p_0} \left[\sum_{k=0}^{L-1} \mathop{\mathbb{E}}_{y \sim \underbrace{\mathcal{L}(Y_k^{x_0, G})}} \left[\sum_{i : y^i = M} \underbrace{G(x_0, y)[i]} \left(\log \left(D_\theta^i(y)[x_0^i] \right) + \log \left(\overbrace{\frac{F_\theta(y, x_0^i, i)}{G(x_0, y)[i]}} \right) \right) \right] \right]$$

Take planned path to time k

Reweight masked positions of partially masked data via G (diff. for each i!)

Specialize to greedy ancestral: $G(\mathbf{z}, \mathbf{x}) = \delta\left(\underset{j:x_k^j=\mathbf{m}}{\operatorname{argmax}} D_\theta^j(\mathbf{x})[z^j]\right)$

$$\mathcal{L}^G(\theta) = -\mathop{\mathbb{E}}_{x_0\sim p_0}\left[\sum_{k=0}^{L-1}\mathop{\mathbb{E}}_{y\sim\mathcal{L}(Y_k^{x_0},G)}\left[\sum_{i:y^i=M} G(x_0,y)[i]\left(\log\left(D_\theta^i(y)[x_0^i]\right)+\log\left(\frac{F_\theta(y,x_0^i,i)}{G(x_0,y)[i]}\right)\right)\right]\right]$$

Take planned path to time k

Reweight position via G

Match sampling planner to data planner

Deterministic given x_0 and requires k evals of D_θ

Only trains on 1 pos per y

Unstable early, small grads for pretrained D_θ

Specialize to greedy ancestral: $G(\mathbf{z}, \mathbf{x}) = \delta\left(\underset{j:x_k^j=\mathbf{m}}{\operatorname{argmax}} D_\theta^j(\mathbf{x})[z^j]\right)$

$$\mathcal{L}^G(\theta) = -\mathop{\mathbb{E}}_{x_0\sim p_0}\left[\sum_{k=0}^{L-1}\mathop{\mathbb{E}}_{y\sim\mathcal{L}(Y_k^{x_0},G)}\left[\sum_{i:y^i=M} G(x_0,y)[i]\left(\log\left(D_\theta^i(y)[x_0^i]\right)+\log\left(\frac{F_\theta(y,x_0^i,i)}{G(x_0,y)[i]}\right)\right)\right]\right]$$

Take planned path to time k

Reweight position via G

Match sampling planner to data planner

Deterministic given x_0 and requires k evals of D_θ

Only trains on 1 pos per y

Unstable early, small grads for pretrained D_θ

1. Replace G with $G^\tau(z,x)[i] \propto \exp\left(\log\left(D_\theta^i(x)[z^i]\right)/\tau\right), \quad$ (converges to greedy as $\tau\downarrow 0$ and unif. as $\tau\uparrow\infty$)
2. Replace $\mathcal{L}(Y_k^{x_0,G})$ with $\mathcal{L}(Y_k^{x_0})$ (mask L-k pos of x_0 unif at random)

Specialize to greedy ancestral: $G(\mathbf{z}, \mathbf{x}) = \delta\left(\underset{j:x_k^j=\mathbf{m}}{\operatorname{argmax}} D_\theta^j(\mathbf{x})[z^j]\right)$

$$\mathcal{L}^G(\theta) = -\mathop{\mathbb{E}}_{x_0\sim p_0}\left[\sum_{k=0}^{L-1}\mathop{\mathbb{E}}_{y\sim\mathcal{L}(Y_k^{x_0,G})}\left[\sum_{i:y^i=M} G(x_0,y)[i]\left(\log\left(D_\theta^i(y)[x_0^i]\right) + \log\left(\frac{F_\theta(y,x_0^i,i)}{G(x_0,y)[i]}\right)\right)\right]\right]$$

Take planned path to time k

Reweight position via G

Match sampling planner to data planner

Deterministic given x_0 and requires k evals of D_θ

Only trains on 1 pos per y

Unstable early, small grads for pretrained D_θ

1. Replace G with $G^\tau(z,x)[i] \propto \exp\left(\log\left(D_\theta^i(x)[z^i]\right)/\tau\right), \quad$ (converges to greedy as $\tau\downarrow 0$ and unif. as $\tau\uparrow\infty$)
2. Replace $\mathcal{L}(Y_k^{x_0,G})$ with $\mathcal{L}(Y_k^{x_0})$ (mask L-k pos of x_0 unif at random)
3. Detach gradient from planner (remove last term)
4. Discourage mode collapse via interpolation with vanilla loss, weight α

PAPL Training

while not converged **do**

$\mathbf{x}_0 \sim p_0$

$k \sim \text{Unif}([0 : L-1])$

$\mathbf{y} \sim \text{Unif}(\mathcal{X}_{L-k}(\mathbf{x}_0))$

$w^i \overset{sg}{\leftarrow} \text{softmax}\left(\log\left(D_\theta(y)[\mathbf{x}_0]\right)/\tau\right)$

// Setting $\alpha = 0$ recovers standard DLM training

$\mathcal{L}_{\text{PAPL}} \leftarrow -\sum_{i:y^i=M} \frac{1}{L-k}(1+\alpha w^i)\log D^i_\theta(\mathbf{y})[x^i_0]$

Update parameters θ using $\nabla_\theta \mathcal{L}_{\text{PAPL}}$

end while

return Trained denoiser D_θ

2 line code modification!

$$\mathcal{L}^{\text{PAPL}}(\theta) = -\mathop{\mathbb{E}}_{x_0\sim p_0}\left[\sum_{k=0}^{L-1}\mathop{\mathbb{E}}_{y\sim\mathcal{L}(Y_k^{x_0})}\left[\sum_{i:y^i=M}\frac{1+\alpha w^i}{L-k}\log\left(D^i_\theta(y)[x^i_0]\right)\right]\right], \qquad w^i \propto \exp\left(\log\left(D^i_\theta(y)[x^i_0]\right)/\tau\right)$$

Significant gains across tasks

Code generation performance.
Results marked with † are adopted from prior work.

Model	HumanEval Pass@10	HumanEval+ Pass@10	MBPP+ Pass@10
Reference Models (≥7B)			
LLaDA (8B)	50.0	43.3	69.1
Dream (7B)	59.2	53.7	72.5
Compact Models (≤1B)			
Autoregressive† (1.3B)	34.7	28.6	–
DLM (0.5B)	31.1	28.0	32.6
DLM (0.5B) + PAPL (**Ours**)	**38.4**	**35.2**	**53.6**

Protein sequence generation benchmark. Foldability is the percentage of sequences satisfying pLDDT > 80, pTM > 0.7, and pAE < 10.

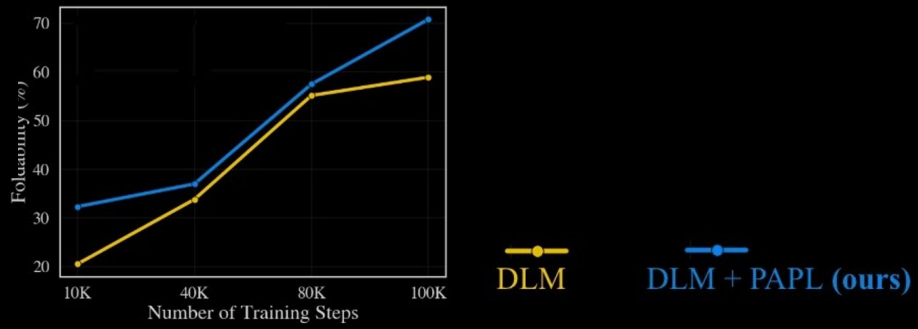
	Model	Foldability (%)↑	Entropy↑	Diversity (%)↑
Large	ESM3	1.50	3.99	**93.44**
	ProGen2-large	11.87	2.73	91.48
	DPLM-650M	49.14	3.18	92.22
≈150M	ProGen2-small	4.48	2.55	89.31
	DLM-150M	42.43	3.21	92.45
	DLM-150M + PAPL (ours)	**59.40**	3.12	91.73

Code generation:

- Up to +21% pass@10, 64% relative improvement
- 500M DLM+PAPL surpasses 1.3B autoregressive

Protein sequence generation:

- +17% foldability, 40% relative improvement
- 150M DLM+PAPL surpasses SOTA 650M DLM



Takeaways

- Continuous time formulation = discrete time formulation for MDMs (simpler).

Takeaways

- Continuous time formulation = discrete time formulation for MDMs (simpler).
- **Go beyond one token/pixel** in your mathematical formulation**.** 1 token + independence is **enforcing structure** on your reference/sampling process.

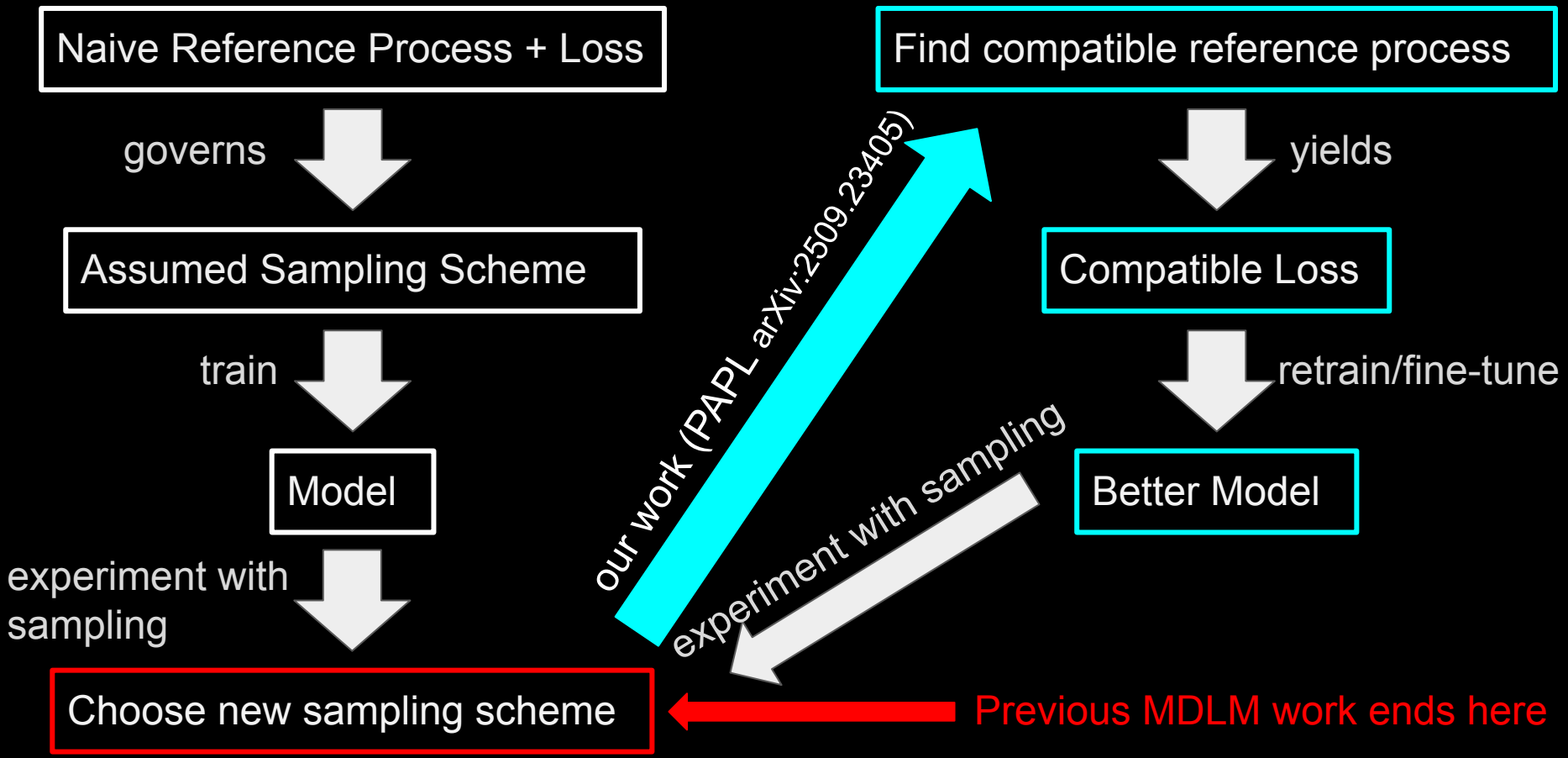
Takeaways

- Continuous time formulation = discrete time formulation for MDMs (simpler).
- **Go beyond one token/pixel** in your mathematical formulation**.** 1 token + independence is **enforcing structure** on your reference/sampling process.
- Study your loss **beyond just pointwise**. Minimizer is not enough (you wont reach optimality). Instead, **identify what gradients are doing**.

Takeaways

- Continuous time formulation = discrete time formulation for MDMs (simpler).
- **Go beyond one token/pixel** in your mathematical formulation**.** 1 token + independence is **enforcing structure** on your reference/sampling process.
- Study your loss **beyond just pointwise**. Minimizer is not enough (you wont reach optimality). Instead, **identify what gradients are doing**.
- Your **loss governs an assumed sampling scheme**. If you want to use a different sampling scheme which yields different 1D time marginals, you probably could have chosen a better loss.

Completing the cycle



Completing the cycle

Thank you! Me:

Naive Reference Process + Loss

governs

Assumed Sampling Scheme

train

Model

experiment with sampling

Choose new sampling scheme

Previous MDLM work ends here

Find compatible reference process

yields

Compatible Loss

retrain/fine-tune

Better Model

our work (PAPL arXiv:2509.23405)

experiment with sampling