STOP TRAINING FOR THE WORST: PROGRESSIVE UNMASKING ACCELERATES MASKED DIFFUSION TRAINING

Jaeyeon Kim[*] Harvard University
Jonathan Geuter[*] Harvard University, Kempner Institute
David Alvarez-Melis Harvard University, Kempner Institute
Sham Kakade Harvard University, Kempner Institute
Sitan Chen Harvard University

ABSTRACT

Masked Diffusion Models (MDMs) have emerged as a promising approach for generative modeling in discrete spaces. By generating sequences in any order and allowing for parallel decoding, they enable fast inference and strong performance on non-causal tasks. However, this flexibility comes with a *training complexity* trade-off: MDMs train on an exponentially large set of masking patterns, which is not only computationally expensive, but also creates a train–test mismatch between the random masks used in training and the highly structured masks induced by inference-time unmasking. In this work, we propose Progressive UnMAsking (PUMA), a simple modification of the forward masking process that aligns training-time and inference-time masking patterns, thereby focusing optimization on *inference-aligned masks* and speeding up training. Empirically, PUMA speeds up pretraining at the 125M scale by $\approx 2.5\times$ and offers complementary advantages on top of common recipes like autoregressive initialization.

1 INTRODUCTION

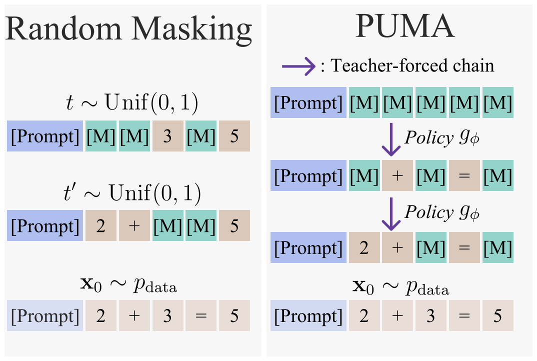
In recent years, Masked Diffusion Models (MDMs) Lou et al. (2023); Shi et al. (2024); Sahoo et al. (2024) have emerged as a compelling alternative to autoregressive models for discrete generative modeling, driven by rapid scaling efforts from 7B Ye et al. (2025); Nie et al. (2025); Gong et al. (2025); Song et al. (2025) to 100B parameters Bie et al. (2025), as well as industry-scale deployments Labs et al. (2025); DeepMind (2025). Much of their appeal comes from inference-time advantages: parallel decoding speeds up generation, while any-order inference better accommodates tasks that require non-autoregressive generation.

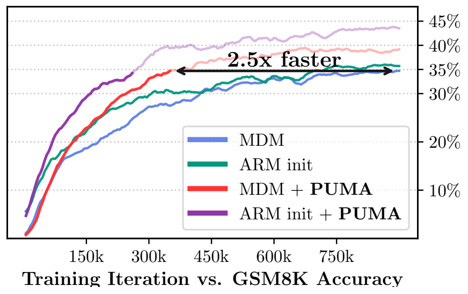
At the same time, MDMs come with a well-known trade-off Kim et al. (2025b), **Train for the Worst**: to have any-order generation ability, they must train on unmasking tasks with an exponential number of masking patterns, substantially increasing the complexity of training. Under this paradigm, compute is spread across all masking patterns, while at inference, the masking patterns occupy a negligible fraction of the ones seen during training. This is because adaptive inference prioritizes positions for which the model's predictions are most certain, causing unmasking decisions to concentrate on a small subset of masking patterns.

Yet, this mismatch can be turned into an opportunity: training complexity can be reduced by focusing on inference-aligned masking patterns. However, despite training efficiency being essential to scaling MDMs, recent work has largely emphasized inference, leaving **MDM training comparatively underexplored**.

In this work, we propose **Progressive UnMAsking (PUMA)**. With a *modified forward process*, PUMA generates training examples whose masking patterns ***provably match*** those observed at inference, thereby resolving the train-test mismatch. To our knowledge, PUMA is the first approach to **accelerate discrete diffusion training** by redesigning the forward masking process itself, rather than architectural Arriola et al. (2025b); Ni & team (2025) or recipe-level remedies, such as autoregressive initialization Bie et al. (2025); Ye et al. (2025); Gong et al. (2025); DeepMind (2025) or

[*]Equal contribution. Correspondence to `jaeyeon_kim@g.harvard.edu`.





(a) Training examples are generated via a teacher-forced chain conditioned on a clean sequence $\mathbf{x}_0$ using the current unmasking policy g_ϕ. In contrast, standard MDM training samples independently masked contexts.

(b) PUMA (blue) accelerates MDM training (red) on TinyGSM Liu et al. (2023). PUMA composes seamlessly with autoregressive initialization (purple over green).

Figure 1: Left: Illustration of PUMA trajectories compared to standard MDM masking. **Right:** Training acceleration achieved by PUMA.

block-size curriculum Bie et al. (2025); Fan et al. (2026), or loss reweighting Shi & Titsias (2025); Peng et al. (2026).

Empirically, PUMA accelerates MDM pretraining by up to 2.5$\times$ in iteration–accuracy comparisons at the 125M scale, while introducing only *one* additional hyperparameter and incurring no forward-pass overhead. We further demonstrate that PUMA is fully compatible with widely adopted approaches that instead modify the architecture or training recipe, such as autoregressive initialization and block-size curricula, offering additional speedups on top of these.

2 Related Work

Given the any-order nature of the prediction task, MDM training is inherently more challenging and less compute-efficient than autoregressive training Nie et al. (2024); Kim et al. (2025b), where the model only needs to learn causal next-token prediction. Consequently, prior work has relied on architectural or training-recipe-level strategies applied on top of the core MDM procedure. At scale, a common practice is to initialize MDMs from pretrained autoregressive models Ye et al. (2025); Song et al. (2025); Bie et al. (2025); Fan et al. (2026), along with mixture-of-experts variants Ni & team (2025). Block diffusion Arriola et al. (2025a;b) combines block-wise autoregressive generation with MDM-style inference within each block, yielding speedups but essentially *trading off* full any-order flexibility: at each step, the set of unmasked positions must lie within a block. **Notably, these approaches typically leave the underlying masking process unchanged.** In contrast, PUMA operates **at the level of the forward masking process**; it is thus largely orthogonal to these designs and can be incorporated alongside them, potentially compounding speedups.

The recent work of (Peng et al., 2026) also aims to resolve the mismatch between the training and inference distributions of MDMs. By incorporating a *planner* into the inference strategy – which is the same as our *unmasking policy* g_ϕ, see Section 3 – the authors derive a planner-aware ELBO for MDMs, in which losses are reweighted on a per-token level according to the planner. However, this ELBO depends on trajectories drawn from the unmasking policy, and in practice, Peng et al. (2026) train on the commonly used random masking scheme for MDMs, which does not fully resolve the misalignment between training and inference distributions. We will see that PUMA takes this idea a step further and aims at directly *adapting* the training distribution to the one encountered at inference. We note that in our experiments, reweighting the loss did not yield any advantage, hence we train PUMA with an unweighted loss.

3 Preliminaries

We begin by reviewing the training and inference of Masked Diffusion Models (MDMs).

Notation. Assume we aim to learn the distribution p_{data} over sequences of length L with finite vocabulary $\mathcal{V}$. Denote by $\mathbf{x}^i$ the i-th element of a given sequence $\mathbf{x} = (\mathbf{x}^1, \ldots, \mathbf{x}^L)$, and let $\Delta(\mathcal{V})$ be the simplex of probability distributions over $\mathcal{V}$.

MDM training. MDMs Lou et al. (2023); Shi et al. (2024); Sahoo et al. (2024) introduce an auxiliary mask token $\mathbf{m}$. At a high level, they learn the posterior marginal distributions over $\mathcal{V}$ at each mask token, conditioned on a partially masked sequence. Concretely, an MDM defines a *forward masking process*: for a given $\mathbf{x}_0 \sim p_{\text{data}}$, draw a time step $t \sim \text{Unif}[0,1]$, then independently replace each token $\mathbf{x}_0^i$ by $\mathbf{m}$ with probability t.[1] This results in a partially masked sequence $\mathbf{x}_t \in (\mathcal{V} \cup \{\mathbf{m}\})^L$.

This resulting joint distribution over $(\mathbf{x}_0, t, \mathbf{x}_t)$ induces the coordinate-wise posterior $p(\mathbf{x}_0^i = \cdot \,|\, \mathbf{x}_t, t)$, which we refer to as *unmasking posterior*: the distribution over clean token at masked position i given $(\mathbf{x}_t, t)$. The masking process above has an intriguing property–the unmasking posterior is *time-agnostic* Ou et al. (2024); Zheng et al. (2024); it depends only on the observed (partially masked) sequence $\mathbf{x}_t$, not the time step t; therefore, we drop the time step in the notation and write $p(\mathbf{x}_0^i = \cdot \,|\, \mathbf{x}_t) = p(\mathbf{x}_0^i = \cdot \,|\, \mathbf{z})$, adopting the notation $\mathbf{z} = \mathbf{x}_t$.

To learn this unmasking posterior, MDMs employ a neural network f_θ that takes a sequence $\mathbf{z}$ as an input and outputs a tensor of shape $|\mathcal{V}| \times L$. Its i-th column $f_\theta^i(\cdot \,|\, \mathbf{z}) \in \Delta(\mathcal{V})$ learns the unmasking posterior $f_\theta^i(v \,|\, \mathbf{z}) \approx p(\mathbf{x}_0^i = v \,|\, \mathbf{z})$. To train f_θ, MDMs minimize the following cross-entropy loss summed over all masked indices:

$$\mathcal{L}(\theta) = \mathbb{E}_{(\mathbf{x}_0, t, \mathbf{x}_t)} \left[\frac{1}{t} \sum_{i:\, \mathbf{x}_t^i = \mathbf{m}} - \log f_\theta^i(\mathbf{x}_0^i \,|\, \mathbf{x}_t) \right].$$

where the expectation is taken over the joint distribution of $(\mathbf{x}_0, t, \mathbf{x}_t)$ defined above. As desired, the unique minimizer $f_{\theta^\star}$ of the loss $\mathcal{L}_\theta$ satisfies $f_{\theta^\star}^i(v \,|\, \mathbf{z}) = p(\mathbf{x}_0^i = v \,|\, \mathbf{z})$. This viewpoint connects the MDM training to any-order masked language models such as BERT (Devlin et al., 2019), which also model the posterior marginals at masked positions.

MDM inference. MDM inference starts from a fully masked length-L sequence $\mathbf{x}_1 = (\mathbf{m}, \ldots, \mathbf{m})$ and proceeds over a monotonically decreasing grid of times $t_0 = 1 > \cdots > t_K = 0$. At each step t_j, given a partially masked sequence $\mathbf{x}_{t_j} \in (\mathcal{V} \cup \{\mathbf{m}\})^L$, inference proceeds by two steps to obtain $\mathbf{x}_{t_{j+1}}$:

(a) Choose a subset of masked positions $\mathcal{S}$ to unmask.

(b) $i \in \mathcal{S}$, unmask $\mathbf{x}_{t_j}^i$ with a clean token from $f_\theta^i(\cdot \,|\, \mathbf{x}_{t_j}) \in \Delta(\mathcal{V})$.

The flexibility of choosing $\mathcal{S} \subseteq \{i \,|\, \mathbf{x}_{t_j}^i = \mathbf{m}\}$ lies at the core of MDM inference; selecting which positions to unmask next can dramatically influence downstream performance both in terms of accuracy, as well as efficiency. A large body of prior work (Chang et al., 2022; Zheng et al., 2023; Kim et al., 2025b; Nie et al., 2025; Peng et al., 2025; Jazbec et al., 2025; Hong et al., 2025; Ma et al., 2025; Lee et al., 2025; Hayakawa et al., 2025; Mo et al., 2025) investigates this question.

Formally, an unmasking policy g_ϕ – either heuristic or learned – maps a partially masked sequence and time $(\mathbf{x}_t \in (\mathcal{V} \cup \{\mathbf{m}\})^L,\, t)$ to a (possibly stochastic) subset $\mathcal{S} \subseteq \{i : \mathbf{x}_t^i = \mathbf{m}\}$ of the masked positions, which is used at each inference step to select a subset to unmask. A widely used instantiation scores each masked position and then sets $\mathcal{S}$ as the top-K positions: $g_\phi(\mathbf{x}_t, t) = \text{TopK}_{i:\, \mathbf{x}_t^i = \mathbf{m}}\big[\text{score}(i)\big]$, where $\text{score}(i)$ is a confidence score for position i. Common choices for computing the score include the maximum predicted probability $\max_{v \in \mathcal{V}} f_\theta(v \,|\, \mathbf{x}_t, t)$ ("Top-K"), the margin between the top two probabilities $f_\theta(v_1 \,|\, \mathbf{x}_t, t) - f_\theta(v_2 \,|\, \mathbf{x}_t, t)$ where v_1, v_2 are the top-2 tokens with the highest assigned probability ("Top-K margin"), the negative entropy of the categorical distribution ("entropy-based"), or a position-dependent schedule (e.g., semi-autoregressive inference).

[1] We present the masking process using a linear schedule.

4 PUMA: Progressive UnMAsking

In this section, we introduce **Progressive UnMAsking (PUMA)**. In Section 4.1, we establish the theoretical foundation of PUMA and present how it resolves the MDM's train-test mismatch stated in the introduction. We then describe our empirical design choices for PUMA in Section 4.2.

4.1 Theoretical foundation of PUMA

We begin by formalizing the train–test mismatch in MDMs.

Train-test mismatch of MDM. Recall that MDM inference proceeds by following an unmasking policy g_ϕ that looks at the *currently revealed* tokens and decides which masked positions to unmask next. As a consequence, inference does not visit masked sequences uniformly (as done in the training); instead, it induces a policy-dependent distribution over intermediate masked sequences. For example, under a *semi-autoregressive* unmasking policy Nie et al. (2025), inference proceeds block-wise in an autoregressive manner, hence the maskings observed at inference are restricted to block-wise causal patterns.

Meanwhile, the training allocates the compute uniformly across all masking patterns, since each position is masked i.i.d. This suggests that training could be more compute-efficient if we *concentrate* the training distribution on masking patterns that arise at inference. This raises the question: ***Is there a masking procedure that resolves this mismatch?***

Challenge. The above question is subtle for two reasons. **(1)** At training time, we start from a clean sequence $\mathbf{x}_0$ and construct a masked example $\mathbf{x}_t$; at inference, however, intermediate states $\mathbf{x}_t$ are generated by *progressively* applying the unmasking policy to partially masked sequences, without access to any clean sequence. **(2)** We cannot arbitrarily change the forward process: modifying the joint distribution over $(\mathbf{x}_0, t, \mathbf{x}_t)$ can inadvertently change what MDM training aims to learn. Concretely, unless done carefully, changing $(\mathbf{x}_0, t, \mathbf{x}_t)$ can shift the minimizer of the loss

$$\mathcal{L}(\theta) = \mathbb{E}_{(\mathbf{x}_0,t,\mathbf{x}_t)}\left[\frac{1}{t}\sum_{i:\, \mathbf{x}_t^i=\mathbf{m}} -\log f_\theta^i(\mathbf{x}_0^i \,|\, \mathbf{x}_t)\right]. \tag{1}$$

PUMA. We now present PUMA and explain how it resolves these challenges. The *crux* of PUMA is the so-called *teacher-forced chain*, which we use to generate training-time samples. At a high level, the teacher-forced chain is a progressive unmasking process that follows the *same unmasking policy* g_ϕ used at inference time, but instead of sampling clean tokens from a posterior, it reveals the corresponding ground-truth tokens from the data sample $\mathbf{x}_0$. Surprisingly, this simple chain lets us align the distribution over masking patterns at training-time with the one at inference time.

Teacher-forced chain. Fix a clean sequence $\mathbf{x}_0$, an unmasking policy g_ϕ, and an integer $K \geq 1$. Define a time grid $t_j := 1 - j/K$ for $j = 0, 1, \ldots, K$, so that $t_0 = 1$ and $t_K = 0$. We initialize the chain at the fully masked state $\mathbf{x}_{t_0} = \mathbf{x}_1 = (\mathbf{m}, \ldots, \mathbf{m})$. To clarify, in contrast to the inference (Section 3), we do not draw a clean token from f_θ; whenever positions are selected, we reveal ground-truth tokens from $\mathbf{x}_0$. Concretely, given a partially masked state $\mathbf{x}_{t_j}$, obtain $\mathbf{x}_{t_{j+1}}$ by:

(a) Choose a subset of masked positions $\mathcal{S} = g_\phi(\mathbf{x}_{t_j}, t_j)$.

(b) For each $i \in \mathcal{S}$, unmask $\mathbf{x}_{t_j}^i = \mathbf{m}$ into clean token $\mathbf{x}_0^i$.

The chain above is defined *conditional* on a particular $\mathbf{x}_0$, i.e., it specifies a conditional distribution over trajectories $(\mathbf{x}_{t_0}, \ldots, \mathbf{x}_{t_K})$ given $\mathbf{x}_0$. At training time, we first sample $\mathbf{x}_0 \sim p_{\text{data}}$ and then run the chain (conditioned on $\mathbf{x}_0$).

Therefore, each chain run yields multiple intermediate masked sequences, each of which serves as a training example paired with the ground-truth sequence $\mathbf{x}_0$. In contrast, standard MDM training generates a single training example with random masking. We illustrate this in Figure 1a. This teacher-forced chain can be efficiently implemented in practice without any computational overhead, and we will revisit it in Section 4.2. We now informally state the **theoretical guarantee of PUMA**: ***marginal agreement*** and ***minimizer preservation***.

(PUMA in a nutshell). PUMA training, i.e., sampling $\mathbf{x}_0 \sim p_{\text{data}}$ and generating contexts $\mathbf{x}_{t_j}$ via the teacher-forced chain, **(i)** matches the marginal distributions induced by MDM inference under the same policy g_ϕ and **(ii)** preserves the unmasking-posterior minimizer of the MDM objective.

Together, these two properties resolve the challenges identified earlier: PUMA aligns training and inference distributions, while ensuring that modifying the forward process does not change what the model is trained to learn. ***As a result, PUMA enables more compute-efficient MDM training without affecting the minimizer of the underlying objective.***

Current model as an unmasking policy. An important practical consideration in operating the teacher-forced chain during pretraining is that the final unmasking policy, namely the one used at inference, is not available a priori: it is induced by the final pretrained network (see Section 3 for the policy instances). Accordingly, we construct the teacher-forced chain using the unmasking policy defined by the *current model* at that point in the training process.

Intuitively, as training proceeds and the model's prediction gets better, the induced policy g_ϕ rapidly approaches the inference-time policy of the final model. In particular, the *ranking* of which positions to unmask might tend to stabilize earlier than full posterior calibration, so the policy becomes *approximately final* relatively early in training. We indeed validate this empirically in Section 5.1, and return to implementation details in Section 4.2.

4.1.1 PUMA's marginal agreement property

In this section, we formalize the marginal agreement property of PUMA. Detailed argument on the minimizer preservation can be found in Appendix A.

Intuition. The unmasking posterior $p(\mathbf{x}_0^i = \cdot \mid \mathbf{z})$ used in the MDM inference step (b) is the data conditional distribution given $\mathbf{z}$. Therefore, "revealing a token from the *true* posterior" (*idealized* MDM inference) is distributionally equivalent to "first draw $\mathbf{x}_0 \sim p_{\text{data}}$, then reveal its tokens, and later marginalize over $\mathbf{x}_0 \sim p_{\text{data}}$" (PUMA): both generate the same joint law over revealed tokens and remaining masks. Since g_ϕ only reacts to what is currently revealed (and time), both procedures induce the same distribution over intermediate masked contexts at each t.

Proposition 1 (Informal)**.** *Fix a policy g_ϕ and consider an* idealized *MDM inference procedure that, at step (b), samples clean tokens from the ground-truth unmasking posterior $p(\mathbf{x}_0^i = \cdot \mid \mathbf{z})$. Let q_t denote the distribution of $\mathbf{x}_t$ under this idealized inference. Then, for every t, the marginal distribution of $\mathbf{x}_t$ induced by the teacher-forced chain with $\mathbf{x}_0 \sim p_{\text{data}}$ coincides with q_t.*

Remark. The precise way to present Proposition 1 is under the *idealized* policy, where we only choose one position to unmask at each time step. This is because the token sampling step at inference is always done with f_θ's per-position posterior; the distributional dependency across positions is ignored. A more nuanced way to understand the teacher-forced chain is through continuous-time Markov chains Campbell et al. (2022); Gat et al. (2024); Kim et al. (2025a), under which the marginal agreement holds for all $t \in [0, 1]$. We formalize this argument in Appendix A.1.

4.1.2 PUMA's minimizer guarantee

In this section, we demonstrate that PUMA does *not* alter the minimizer of the training loss. We first investigate how the design of the forward process can influence the minimizer of the training loss. Consider the i.i.d. forward process, whose conditional distribution takes the following form:

$$\begin{aligned} &p(\mathbf{x}_t = \mathbf{z} \mid \mathbf{x}_0, t) \\ &\quad = \underline{(1-t)^{|\mathbf{um}(\mathbf{z})|} t^{L-|\mathbf{um}(\mathbf{z})|}} \cdot \mathbf{1}\{\mathbf{x}_0^{\mathbf{um}(\mathbf{z})} = \mathbf{z}^{\mathbf{um}(\mathbf{z})}\}, \end{aligned}$$

where $\mathbf{um}(\mathbf{z}) := \{i \mid \mathbf{z}^i \neq \mathbf{m}\}$ is the set of unmasked tokens $\mathbf{z}$ and $\mathbf{1}\{\mathbf{x}_0^{\mathbf{um}(\mathbf{z})} = \mathbf{z}^{\mathbf{um}(\mathbf{z})}\}$ is the indicator function that identifies whether $\mathbf{z}$ matches $\mathbf{x}_0$ on $\mathbf{um}(\mathbf{z})$. Now we observe how the first term (green factor) cancels out in the Bayes-optimal predictor, which is also the cross-entropy training loss's minimizer. By Bayes' rule,

$$\begin{aligned} \underline{p(\mathbf{x}_0 \mid \mathbf{x}_t = \mathbf{z}, t)} &\propto p(\mathbf{x}_t = \mathbf{z} \mid \mathbf{x}_0, t)\, p_{\text{data}}(\mathbf{x}_0) \\ &= \underline{(1-t)^{|\mathbf{um}(\mathbf{z})|}\, t^{L-|\mathbf{um}(\mathbf{z})|}} \cdot \mathbf{1}\{\mathbf{x}_0^{\mathbf{um}(\mathbf{z})} = \mathbf{z}^{\mathbf{um}(\mathbf{z})}\}\, p_{\text{data}}(\mathbf{x}_0)\,, \end{aligned}$$

where the coordinate-wise conditional distribution (blue term) corresponds exactly to the minimizer of the training loss. Crucially, the first term of the right-hand side (green factor) does not depend on $\mathbf{x}_0$ and therefore cancels out during normalization, leaving the posterior unchanged. The same argument applies more generally as long as the forward process follows the form

$$p(\mathbf{x}_t = \mathbf{z} \mid \mathbf{x}_0, t) \propto \underline{\alpha_t(\mathbf{z})} \cdot \mathbf{1}\{\mathbf{x}_0^{\mathbf{um}(\mathbf{z})} = \mathbf{z}^{\mathbf{um}(\mathbf{z})}\}, \tag{2}$$

for an arbitrary nonnegative function $\alpha_t(\mathbf{z})$.

Proposition 2 (Informal)**.** *Under a forward process of the form of equation* (2)*, the training loss* (1)*'s unique minimizer does not change from the ground-truth unmasking posterior.*

Importantly, the teacher-forced chain has the form of equation (2): the selection of $\mathcal{S}$ depends only on the currently observed tokens in $\mathbf{x}_{t_j}$ (and time), as the unmasking policy $g_\phi(\mathbf{x}_{t_j}, t_j)$ does so, and never on the hidden entries of $\mathbf{x}_0$. Hence, it belongs to the class of forward processes (2), so replacing the vanilla forward process with the teacher-forced chain changes only the *training distribution* while preserving the same unmasking-posterior minimizer of the MDM loss (Proposition 2).

4.2 Empirical instantiation of PUMA

In this section, we describe how we instantiate PUMA in practice.

Streaming teacher-forced chains in a minibatch. We implement PUMA as a streaming buffer of B teacher-forced chains. Concretely, the minibatch state consists of tuples $\{(\mathbf{x}_0(j),\, \mathbf{z}(j),\, n(j))\}_{j=1}^{B}$, where $\mathbf{x}_0(j)$ is the underlying clean sample, $\mathbf{z}(j)$ is the current masked context (a state on its chain), and $n(j) \in \{0, \ldots, K\}$ is the current stage. At each training step, we compute the MDM loss on the current minibatch (one gradient step). We then advance *each* sequence to the next state of its chain (by revealing additional tokens according to PUMA). When a sequence reaches the end of the chain, we refresh it by drawing a new clean example $\mathbf{x}_0$, re-initializing a new chain. Thus, every intermediate state along each chain becomes a training sample. For prompt-based tasks, we keep the prompt unmasked at all stages and apply PUMA only to the non-prompt region.

Algorithm 1 PUMA pseudocode (one training iteration)

```
1:  Require: MDM $f_\theta$, stages $K \in \mathbb{N}$, masked sequences $\{\mathbf{z}(j)\}_{j=1}^{B}$, paired clean samples
    $\{\mathbf{x}_0(j)\}_{j=1}^{B}$, and per-example stage counters $\{n(j)\}_{j=1}^{B} \in [0, K]$.
2:  Forward pass: compute logits $\{f_\theta(\cdot \mid \mathbf{z}(j))\}_{j=1}^{B}$          ▷ $B \times L \times \Delta(\mathcal{V})$
3:  Compute loss $\mathcal{L}(\theta)$ from logits and $\mathbf{x}_0(j)$; update $\theta$ with the optimizer   ▷ training update
4:  # PUMA streaming update (per example j), parallelized
5:  if $n(j) = K$
6:      Sample $\mathbf{x}_0(j) \sim p_{\text{data}}$; set $n(j) \leftarrow 0$; initialize $\mathbf{z}(j)$        ▷ refill a new sequence
7:  else
8:      Compute scores $s_i \leftarrow \max_{v\in\mathcal{V}} f_\theta^i(v \mid \mathbf{z}(j))$ for masked indices $i$     ▷ unmasking policy
9:      Select reveal set $\mathcal{S}(j)$ using $\{s_i\}$                                ▷ top-$K$ + thresholding
10:     For all $i \in \mathcal{S}(j)$, update $\mathbf{z}(j)^i \leftarrow \mathbf{x}_0(j)^i$, update $n(j)$     ▷ teacher-forced chain update
```

Policy from the current model. As mentioned in Section 4.1, we instantiate the teacher-forced chain's g_ϕ on-the-fly using the current model itself. Specifically, given logits from $f_\theta(\cdot \mid \mathbf{z}(j))$, we compute a per-position confidence score for each masked index, e.g., the maximum predicted probability, rank masked positions by confidence, and unmask roughly L/K tokens at each step. Here, we state "roughly" as we found it empirically beneficial to add a little randomness to the exact number of unmasked tokens L/K; see Appendix B for details.

No additional overhead. Importantly, PUMA does not introduce any extra compute: the logits required for the MDM loss on $\mathbf{z}(j)$ are exactly the quantities used to compute confidence scores.

4.2.1 Practical interventions

Typically, MDM inference reveals only a small number of mask tokens per step. Directly following this behavior in PUMA with a large K is sample-inefficient: a fixed $\mathbf{x}_0$ would generate a long

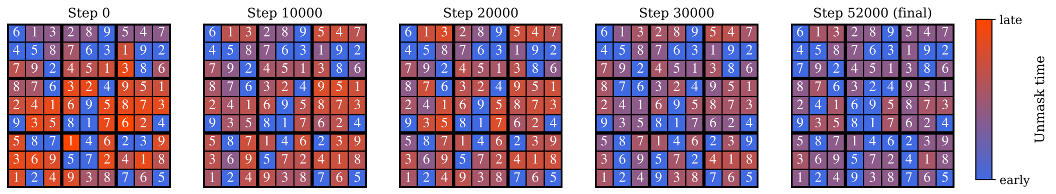


Figure 2: PUMA finds unmasking trajectories close to the final model's trajectories (right) early on in training. We show PUMA training unmasking orders at different training steps for a given Sudoku puzzle. Blue cells are unmasked early, while red cells are unmasked later.

teacher-forced chain, often providing a redundant training signal. We introduce two practical interventions that retain the guarantee of the teacher-forced chain while improving sample efficiency.

Confidence-based fast-forwarding. We *fast-forward* the chain if the next unmasking decisions are trivial for the current model. Concretely, at each step, we reveal roughly L/K masked tokens selected by the policy but also reveal any masked position whose maximum predicted probability exceeds a confidence threshold, e.g., 0.9. Intuitively, highly confident positions provide little training signal if kept masked, so revealing them early lets us follow the teacher-forced trajectory without stopping at every intermediate sequence.

K-scheduling. Early in training, the model-based policy is noisy, so using a large K can waste compute on long, low-quality chains. We therefore use *K-scheduling*: start with a smaller K (yielding more frequent refreshes and greater diversity across $\mathbf{x}_0$) and gradually increase K as training progresses and the policy becomes more informative.

5 Experiments

In this section, we present experimental results for PUMA, focusing on two central claims:

- **Practicality**: PUMA speeds up MDM pretraining.
- **Compatibility**: PUMA offers complementary speedups on top of autoregressive initialization and block-size curricula.

In Section 5.1, we validate our intuition of PUMA's design on Sudoku puzzles. In Section 5.2, we demonstrate PUMA's efficiency in 125M-scale MDM pretraining. Finally, in Section 5.3, we show that these speedups persist when PUMA is combined with other recipe-level remedies.

5.1 Sudoku puzzle as a testbed

In this section, we use Sudoku puzzles as a simple testbed to probe practical viability of PUMA.

Setup. We use the Sudoku dataset from Radcliffe (2020), implemented by Shah et al. (2024), with 1.8M training and 0.1M test puzzles. We train a 6.8M bidirectional transformer with Qwen2-style attention Team et al. (2024), 8 layers, and hidden dimension 256. For PUMA, we use a fixed K-schedule with $K = 8$ and a confidence threshold of 0.9. Training is done on 2 NVIDIA H100 GPUs with a batch size of 32 per GPU, taking approximately an hour.

Policy stabilization during training. PUMA achieves 90.1% final accuracy (with the Top-2 unmasking rule and $|\mathcal{S}| = 2$), and speeds up training compared to the MDM baseline by $1.4\times$; Figure 4 the appendix. Figure 2 visualizes PUMA's teacher-forced chain for a fixed instance at different checkpoints. At the early stage of training, the induced chain is uninformative. However, after a small number of iterations the pattern of the fully trained network's unmasking policy starts to emerge. In addition, the model becomes more confident in its predictions as training proceeds, and PUMA's confidence thresholding results in shorter chains. Figure 6 shows the distance of training trajectories (formally defined in Appendix B) to the final model decrease quickly. This empirically validates that the *ranking* over positions (which drives the unmasking policy) can stabilize earlier than full posterior calibration (which drives the task accuracy), so using the current model for teacher-forced chains is a reasonable proxy already early in training.

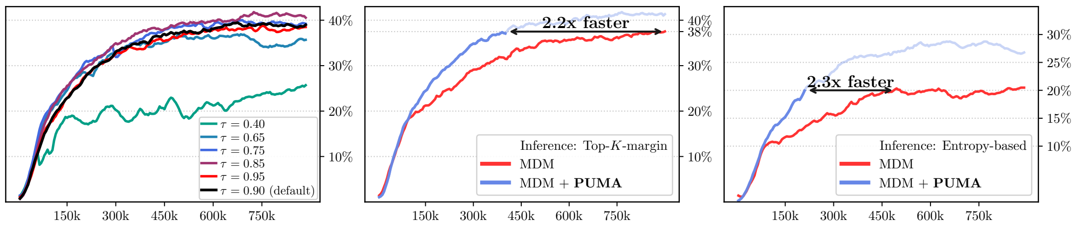


Figure 3: Left: PUMA's efficiency is largely insensitive to the confidence threshold, except when the threshold is very small. **Middle, Right**: A single PUMA-trained model (trained under one policy) remains robust to inference-time policy choices, consistently outperforming the baseline across different unmasking policies. (Top-K margin, Entropy)

5.2 PUMA ACCELERATES PRETRAINING

Moving beyond Sudoku puzzles, we test whether PUMA remains effective in a more realistic, larger-scale pretraining regime. In choosing an appropriate scale and data domain, we aimed for a setup where (1) from-scratch pretrained models can achieve reasonable performance on real downstream benchmarks, e.g., GSM8K Cobbe et al. (2021), while (2) the overall compute requirements remain feasible for running multiple controlled experiments.

Dataset. To satisfy both criteria, we adopt TinyGSM Liu et al. (2023) as a pretraining corpus with 11.8M samples. TinyGSM converts the solutions of GSM8K-style math problems written in natural language into short and structured Python programs, making the task substantially more learnable at small model scales. Indeed, Liu et al. (2023) report that even a 125M-parameter autoregressive model pretrained from scratch on TinyGSM can reach reasonable GSM8K test set performance. Motivated by this, we use TinyGSM to study PUMA pretraining under a 125M MDM, which enables multiple runs and ablations under our compute budget.

Setup. We conduct an apples-to-apples comparison by measuring the *GSM8K test accuracy as a function of gradient steps*. PUMA and the baseline differ *only* in batch control, but all other factors, including architecture, optimizer, and evaluation, are kept fixed. We use a 125M MDM with Qwen2-style attention Team et al. (2024), hidden size 512, 14 layers, 8 heads, and sequence length 512.

As stated in Section 4.2, we include K-scheduling and confidence-thresholding for PUMA. In particular, we initialize $K = 12$ and increase it by 3 every 30k iterations until $K = 42$, and set the confidence threshold as 0.9. All runs train for 900k iterations, and evaluation is done on the standard GSM8K test set every 5k steps. We train on 8 NVIDIA H100 GPUs with batch size 32 per GPU, and training takes approximately three days. At inference, we unmask with the Top-2 scoring rule, using the maximum token-wise probability as confidence as default. To improve robustness, we repeat each experiment three times with different random seeds. Furthermore, evaluation is done on EMA checkpoints as it often provides better accuracy for diffusion models (Karras et al., 2023).

Results. Figure 1b shows that PUMA accelerates (blue) MDM pretraining (red) by up to $2.5\times$ in terms of iterations-to-accuracy. Noting that PUMA does not introduce additional forward-pass computation, this indicates that PUMA is a practically efficient training strategy.

We additionally measure wall-clock throughput: *3.41 it/s* for MDM versus *3.78 it/s* for PUMA. We attribute this small throughput gap to a loss function implementation detail. MDM uses the fused PyTorch primitive `F.cross_entropy`, whereas our PUMA implementation explicitly computes token log-probabilities (needed for the teacher-forced-chain update) and then forms the cross-entropy loss, which may incur extra kernel traffic.

Robustness across inference policies. PUMA's theoretical guarantee is established under the assumption that training and inference deploy the same unmasking policy. In practice, however, models are evaluated under a variety of policies. In light of this, we evaluate each training checkpoint under multiple unmasking policies. Specifically, we consider the combinations of $|\mathcal{S}| \in \{2, 3\}$, and Top-K, Top-K margin, and entropy-based selection. As shown in Figure 3 (Middle, Right) and Figure 5 in the appendix, PUMA consistently improves upon the baseline across all evaluation choices. This indicates that the resulting speedup transfers robustly to other unmasking strategies.

5.2.1 Ablation study

We ablate two primary design choices of PUMA: K *scheduling* and *confidence thresholding*. For K scheduling, we compare our default run (with scheduling from $K = 12$ to $K = 42$) against a variant with a fixed $K = 42$ throughout training. For confidence thresholding, we sweep the threshold in $\{0.4, 0.65, 0.85, 0.9 \text{ (default)}, 0.95\}$.

Results. Figure 5 in the appendix shows that removing K scheduling reduces the speed-up, indicating that the schedule is crucial for PUMA. For confidence thresholding, Figure 3 suggests that PUMA's efficiency is broadly insensitive to the threshold, except when it is too low; 0.4 or 0.65. At such a low threshold, PUMA unmasks numerous low-confident positions, which substantially alters the teacher-forced chain dynamics, deteriorating the performance.

5.3 Combining PUMA with other remedies

In this section, we show that PUMA can be combined with common training recipes for MDMs at a scale, demonstrating that PUMA's benefit is complementary to such remedies.

Autoregressive model initialization. Initializing MDMs from pretrained autoregressive models is a well-known strategy for accelerating the training (He et al., 2022; Ye et al., 2025; Bie et al., 2025; Gong et al., 2025; DeepMind, 2025; Fan et al., 2026). To this end, we first pretrain an autoregressive model with the same transformer architecture as in Section 5.2 for 30k steps, which takes approximately 1.5 hours on 8 NVIDIA H100 GPUs. We then initialize the MDM from this checkpoint, adjust the attention into bidirectional, and train it with and without PUMA. We keep other configurations the same as in Section 5.2. Figure 1b reports the resulting learning curves (purple vs. green). While both methods benefit from ARM initialization, PUMA exhibits a substantially larger gain, achieving an additional speedup of approximately 2.3× relative to vanilla training.

Block-size warmup. Block diffusion (Arriola et al., 2025a;b) combines block-wise autoregressive generation with MDM-style inference within each block, with reported improved training efficiency over vanilla MDM training. Motivated by this, recent large-scale pretraining efforts (Bie et al., 2025; Fan et al., 2026) adopt a *block-size warmup*: MDM is initialized with a pretrained autoregressive model (block size equals the full sequence length) and gradually transitions toward an MDM. Empirically, this warmup improves training efficiency. We train block diffusion models using block size 256, and aggregate the training loss across blocks. For PUMA, we generate a teacher-forced chain as in Section 4.2, but restricted to each block. As shown in Figure 5, PUMA also accelerates training in this setting, which shows that its benefits are complementary to block diffusion.

6 Conclusion

Accelerating diffusion model training has been extensively studied in continuous domains. While these approaches explore different objectives and training principles, they cannot operate beyond the *standard forward-process* framework: Gaussian noise is added independently across dimensions. As a result, most existing degrees of freedom take the form of modifying a scalar noise schedule, reweighing the loss Nichol & Dhariwal (2021); Salimans & Ho (2022), or introducing an auxiliary regularizer Yu et al. (2024).

Masked diffusions, in contrast, admit a far wider design space. They admit substantial flexibility in the *decoding order–which positions are revealed when*. Prior work Zheng et al. (2024); Kim et al. (2025b) has shown that this flexibility can be exploited at inference time. This flexibility is a defining feature of masked diffusions and lies squarely outside the scope of continuous diffusion formulations.

In this work, we show that this structural freedom can also be exploited at training time. PUMA leverages the discrete nature of masked diffusion by modifying the forward masking process to align the distributions over masked sequences induced at train- and inference-time, opening the door to a new design axis for masked diffusion models.

Acknowledgments

SC was supported in part by NSF CAREER award CCF-2441635. DAM acknowledges support from the Kempner Institute, NSF award no. 2229881, and the Aramont Fellowship Fund. JG is supported by a Kempner Graduate Fellowship. SK acknowledges support from the Kempner Institute.

References

Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. *arXiv preprint arXiv:2503.09573*, 2025a.

Marianne Arriola, Yair Schiff, Hao Phung, Aaron Gokaslan, and Volodymyr Kuleshov. Encoder-decoder diffusion language models for efficient training and inference. *arXiv preprint arXiv:2510.22852*, 2025b.

Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, Chengxi Li, Chongxuan Li, Jianguo Li, Zehuan Li, Huabin Liu, Ling Liu, Guoshan Lu, Xiaocheng Lu, Yuxin Ma, Jianfeng Tan, Lanning Wei, Ji-Rong Wen, Yipeng Xing, Xiaolu Zhang, Junbo Zhao, Da Zheng, Jun Zhou, Junlin Zhou, Zhanchao Zhou, Liwang Zhu, and Yihong Zhuang. Llada2.0: Scaling up diffusion language models to 100b, 2025. URL `https://arxiv.org/abs/2512.15745`.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. *Advances in Neural Information Processing Systems*, 35:28266–28279, 2022.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11315–11325, 2022.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Google DeepMind. Gemini diffusion, 2025. URL `https://blog.google/technology/google-deepmind/gemini-diffusion/`.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Chenghao Fan, Wen Heng, Bo Li, Sichen Liu, Yuxuan Song, Jing Su, Xiaoye Qu, Kai Shen, and Wei Wei. Stable-diffcoder: Pushing the frontier of code diffusion large language model, 2026. URL `https://arxiv.org/abs/2601.15892`.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky TQ Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. *Advances in Neural Information Processing Systems*, 37: 133345–133385, 2024.

Shansan Gong, Ruixiang Zhang, Huangjie Zheng, Jiatao Gu, Navdeep Jaitly, Lingpeng Kong, and Yizhe Zhang. Diffucoder: Understanding and improving masked diffusion models for code generation. *arXiv preprint arXiv:2506.20639*, 2025.

Satoshi Hayakawa, Yuhta Takida, Masaaki Imaizumi, Hiromi Wakaki, and Yuki Mitsufuji. Demystifying maskgit sampler and beyond: Adaptive order selection in masked diffusion. *arXiv preprint arXiv:2510.04525*, 2025.

Zhengfu He, Tianxiang Sun, Kuanning Wang, Xuanjing Huang, and Xipeng Qiu. Diffusionbert: Improving generative masked language models with diffusion models, 2022. URL `https://arxiv.org/abs/2211.15029`.

Chunsan Hong, Seonho An, Min-Soo Kim, and Jong Chul Ye. Improving discrete diffusion unmasking policies beyond explicit reference policies. *arXiv preprint arXiv:2510.05725*, 2025.

Metod Jazbec, Theo X Olausson, Louis Béthune, Pierre Ablin, Michael Kirchhof, Joao Monterio, Victor Turrisi, Jason Ramapuram, and Marco Cuturi. Learning unmasking policies for diffusion language models. *arXiv preprint arXiv:2512.09106*, 2025.

Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and improving the training dynamics of diffusion models. *arXiv preprint arXiv:2312.02696*, 2023.

Jaeyeon Kim, Lee Cheuk-Kit, Carles Domingo-Enrich, Yilun Du, Sham Kakade, Timothy Ngotiaoco, Sitan Chen, and Michael Albergo. Any-order flexible length masked diffusion. *arXiv preprint arXiv:2509.01025*, 2025a.

Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham Kakade, and Sitan Chen. Train for the worst, plan for the best: Understanding token ordering in masked diffusions. *arXiv preprint arXiv:2502.06768*, 2025b.

Inception Labs, Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, et al. Mercury: Ultra-fast language models based on diffusion. *arXiv preprint arXiv:2506.17298*, 2025.

Sanghyun Lee, Seungryong Kim, Jongho Park, and Dongmin Park. Lookahead unmasking elicits accurate decoding in diffusion language models. *arXiv preprint arXiv:2511.05563*, 2025.

Bingbin Liu, Sebastien Bubeck, Ronen Eldan, Janardhan Kulkarni, Yuanzhi Li, Anh Nguyen, Rachel Ward, and Yi Zhang. Tinygsm: achieving $>$80% on gsm8k with small language models. *arXiv preprint arXiv:2312.09241*, 2023.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.16834*, 2023.

Long Ma, Fangwei Zhong, and Yizhou Wang. Reinforced context order recovery for adaptive reasoning and planning. *arXiv preprint arXiv:2508.13070*, 2025.

Yichuan Mo, Quan Chen, Mingjie Li, Zeming Wei, and Yisen Wang. Decoding large language diffusion models with foreseeing movement. *arXiv preprint arXiv:2512.04135*, 2025.

Jinjie Ni and team. Openmoe 2: Sparse diffusion language models, 2025. URL `https://github.com/JinjieNi/OpenMoE2`.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pp. 8162–8171. PMLR, 2021.

Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. Scaling up masked diffusion models on text. *arXiv preprint arXiv:2410.18514*, 2024.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *arXiv preprint arXiv:2502.09992*, 2025.

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. *arXiv preprint arXiv:2406.03736*, 2024.

Fred Zhangzhi Peng, Zachary Bezemek, Sawan Patel, Jarrid Rector-Brooks, Sherwood Yao, Avishek Joey Bose, Alexander Tong, and Pranam Chatterjee. Path planning for masked diffusion model sampling, 2025. URL `https://arxiv.org/abs/2502.03540`.

Fred Zhangzhi Peng, Zachary Bezemek, Jarrid Rector-Brooks, Shuibai Zhang, Michael M. Bronstein, Anru Zhang, Joey Bose, and Alexander Tong. Planner aware path learning in diffusion language models training. In *The Fourteenth International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=lAlI5FuIf7`.

David G. Radcliffe. 3 million sudoku puzzles with ratings, 2020. URL https://www.kaggle.com/dsv/1495975.

Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Kulin Shah, Nishanth Dikkala, Xin Wang, and Rina Panigrahy. Causal language modeling can elicit search and reasoning capabilities on logic puzzles. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=i5PoejmWoC.

Jiaxin Shi and Michalis K Titsias. Demystifying diffusion objectives: Reweighted losses are better variational bounds. *arXiv preprint arXiv:2511.19664*, 2025.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37: 103131–103167, 2024.

Yuxuan Song, Zheng Zhang, Cheng Luo, Pengyang Gao, Fan Xia, Hao Luo, Zheng Li, Yuehang Yang, Hongli Yu, Xingwei Qu, et al. Seed diffusion: A large-scale diffusion language model with high-speed inference. *arXiv preprint arXiv:2508.02193*, 2025.

Qwen Team et al. Qwen2 technical report. 2024.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b, 2025. URL https://hkunlp.github.io/blog/2025/dream.

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. *arXiv preprint arXiv:2410.06940*, 2024.

Kaiwen Zheng, Yongxin Chen, Hanzi Mao, Ming-Yu Liu, Jun Zhu, and Qinsheng Zhang. Masked diffusion models are secretly time-agnostic masked models and exploit inaccurate categorical sampling. *arXiv preprint arXiv:2409.02908*, 2024.

Lin Zheng, Jianbo Yuan, Lei Yu, and Lingpeng Kong. A reparameterized discrete diffusion model for text generation. *arXiv preprint arXiv:2302.05737*, 2023.

A Theory

A.1 Proof of Proposition 1

In this section, we provide the complete proof of Proposition 1.

Clarification of the policy notation. In the main text, we write $\mathcal{S} = g_\phi(\mathbf{z}, t)$ to denote the *output* of the unmasking policy, which in practice can be stochastic. In this appendix, we adopt the equivalent distributional view under the simplifying assumption $|\mathcal{S}| = 1$: namely, we write $g_\phi(i \mid \mathbf{z}, t)$ for the conditional probability of selecting the next index i given the current state $(\mathbf{z}, t)$. Sampling $I \sim g_\phi(\cdot \mid \mathbf{z}, t)$ and setting $\mathcal{S} = \{I\}$ recovers the main-text notation.

Notation. For $\mathbf{z} \in \mathcal{Z} := (\mathcal{V} \cup \{\mathbf{m}\})^L$, define the unmasked and masked index sets $\mathrm{um}(\mathbf{z}) := \{i \in [L] : \mathbf{z}_i \neq \mathbf{m}\}$ and $\mathrm{msk}(\mathbf{z}) := [L] \setminus \mathrm{um}(\mathbf{z})$. For $i \in [L]$ and $v \in \mathcal{V}$, write $z^{(i \leftarrow v)} \in \mathcal{Z}$ for the sequence obtained by replacing the i-th entry of z with v (leaving all other coordinates unchanged).

Let $\mathbf{x}_0 \sim p_{\mathrm{data}}$ be a clean sequence in $\mathcal{V}^L$. For any partially-masked $\mathbf{z} \in \mathcal{Z}$ and any $i \in \mathrm{msk}(\mathbf{z})$, the (time-agnostic) ground-truth unmasking posterior is $p(\mathbf{x}_0^i = v \mid \mathbf{z}) := \mathbb{P}(\mathbf{x}_0^i = v \mid \mathbf{x}_0^{\mathrm{um}(\mathbf{z})} = \mathbf{z}^{\mathrm{um}(\mathbf{z})})$.

We now restate the formal version of Proposition 1.

Proposition 1 (Formal)**.** *Fix an integer $K \geq 1$ and a time grid $t_j := 1 - j/K$ for $j = 0, 1, \ldots, K$. Consider the following two Markov chains on $\mathcal{Z}$, both initialized at the fully masked state $\mathbf{z}_{t_0} = (m, \ldots, m)$. Given $\mathbf{z}_{t_j}$, sample an index $I_j \sim g_\phi(\cdot \mid \mathbf{z}_{t_j}, t_j)$ (so $I_j \in \mathrm{msk}(\mathbf{z}_{t_j})$), then sample a token $V_j \sim p(\mathbf{x}_0^{I_j} = \cdot \mid \mathbf{z}_{t_j})$, and set $\mathbf{z}_{t_{j+1}} := \mathbf{z}_{t_j}^{(I_j \leftarrow V_j)}$. Let q_{t_j} denote the law of $\mathbf{z}_{t_j}$ under this procedure. First sample $\mathbf{x}_0 \sim p_{\mathrm{data}}$ and set $\tilde{\mathbf{z}}_{t_0} = (m, \ldots, m)$. Given $\tilde{\mathbf{z}}_{t_j}$, sample $I_j \sim g_\phi(\cdot \mid \tilde{\mathbf{z}}_{t_j}, t_j)$ and set $\tilde{\mathbf{z}}_{t_{j+1}} := \tilde{\mathbf{z}}_{t_j}^{(I_j \leftarrow \mathbf{x}_0^{I_j})}$. Let $\tilde{q}_{t_j}$ denote the marginal law of $\tilde{\mathbf{z}}_{t_j}$ after sampling $\mathbf{x}_0 \sim p_{\mathrm{data}}$. Then, for every $j \in \{0, 1, \ldots, K\}$, we have $q_{t_j} = \tilde{q}_{t_j}$.*

Proof. We show that the two chains have the same one-step transition kernel from every state $\mathbf{z} \in \mathcal{Z}$.

Fix a step j and a state $\mathbf{z} \in \mathcal{Z}$. Under the teacher-forced chain, the event $\{\tilde{\mathbf{z}}_{t_j} = \mathbf{z}\}$ is possible only if the sampled clean sequence $\mathbf{x}_0$ matches $\mathbf{z}$ on the revealed coordinates, i.e., $\mathbf{x}_0^{\mathrm{um}(\mathbf{z})} = \mathbf{z}^{\mathrm{um}(\mathbf{z})}$. Moreover, conditioned on $\mathbf{x}_0^{\mathrm{um}(\mathbf{z})} = \mathbf{z}^{\mathrm{um}(\mathbf{z})}$, the probability of realizing $\tilde{\mathbf{z}}_{t_j} = \mathbf{z}$ depends only on the policy randomness along the trajectory (since g_ϕ reacts only to the currently revealed tokens and t), and is therefore identical for all $\mathbf{x}_0$ that agree with $\mathbf{z}$ on $\mathrm{um}(\mathbf{z})$. Consequently, for some scalar $\alpha_j(\mathbf{z}) \geq 0$ and all $\mathbf{x} \in \mathcal{V}^L$,

$$\mathbb{P}(\tilde{\mathbf{z}}_{t_j} = \mathbf{z} \mid \mathbf{x}_0 = \mathbf{x}) = \alpha_j(\mathbf{z})\, \mathbf{1}\{\mathbf{x}^{\mathrm{um}(\mathbf{z})} = \mathbf{z}^{\mathrm{um}(\mathbf{z})}\}.$$

By Bayes' rule, this implies

$$\mathbb{P}(\mathbf{x}_0 = \mathbf{x} \mid \tilde{\mathbf{z}}_{t_j} = \mathbf{z}) \;\propto\; p_{\mathrm{data}}(\mathbf{x})\, \mathbf{1}\{\mathbf{x}^{\mathrm{um}(\mathbf{z})} = \mathbf{z}^{\mathrm{um}(\mathbf{z})}\}.$$

In particular, for any masked coordinate $i \in \mathrm{msk}(\mathbf{z})$ and any $v \in \mathcal{V}$ We show that the two chains have the same one-step transition kernel from every state $\mathbf{z} \in \mathcal{Z}$.

Fix a step j and a state $\mathbf{z} \in \mathcal{Z}$. Under the teacher-forced chain, the event $\{\tilde{\mathbf{z}}_{t_j} = \mathbf{z}\}$ is possible only if the sampled clean sequence $\mathbf{x}_0$ matches $\mathbf{z}$ on the revealed coordinates, i.e., $\mathbf{x}_0^{\mathrm{um}(\mathbf{z})} = \mathbf{z}_{\mathrm{um}(\mathbf{z})}$. Moreover, conditioned on $\mathbf{x}_0^{\mathrm{um}(\mathbf{z})} = \mathbf{z}_{\mathrm{um}(\mathbf{z})}$, the probability of realizing $\tilde{\mathbf{z}}_{t_j} = \mathbf{z}$ depends only on the policy randomness along the trajectory (since g_ϕ reacts only to the currently revealed tokens and t), and is therefore identical for all $\mathbf{x}_0$ that agree with $\mathbf{z}$ on $\mathrm{um}(\mathbf{z})$. Consequently, for some scalar $\alpha_j(\mathbf{z}) \geq 0$ and all $\mathbf{x} \in V^L$,

$$\mathbb{P}(\tilde{\mathbf{z}}_{t_j} = \mathbf{z} \mid \mathbf{x}_0 = \mathbf{x}) = \alpha_j(\mathbf{z})\, \mathbf{1}\{\mathbf{x}_{\mathrm{um}(\mathbf{z})} = \mathbf{z}_{\mathrm{um}(\mathbf{z})}\}.$$

By Bayes' rule, this implies

$$\mathbb{P}(\mathbf{x}_0 = \mathbf{x} \mid \tilde{\mathbf{z}}_{t_j} = \mathbf{z}) \;\propto\; p_{\mathrm{data}}(\mathbf{x})\, \mathbf{1}\{\mathbf{x}_{\mathrm{um}(\mathbf{z})} = \mathbf{z}_{\mathrm{um}(\mathbf{z})}\}.$$

In particular, for any masked coordinate $i \in \operatorname{msk}(\mathbf{z})$ and any $v \in V$, $\mathbb{P}(\mathbf{x}_0^i = v \mid \tilde{\mathbf{z}}_{t_j} = \mathbf{z}) = p(x_0^i = v \mid \mathbf{z})$ by the definition of the ground-truth unmasking posterior.

Now fix $i \in \operatorname{msk}(\mathbf{z})$ and $v \in V$. Using the above conditional identity and the fact that I_j is drawn from $g_\phi(\cdot \mid \mathbf{z}, t_j)$ given the current state, the teacher-forced transition probability is

$$\mathbb{P}\Big(\tilde{\mathbf{z}}_{t_{j+1}} = \mathbf{z}^{(i\leftarrow v)} \mid \tilde{\mathbf{z}}_{t_j} = \mathbf{z}\Big) = g_\phi(i \mid \mathbf{z}, t_j)\, \mathbb{P}(\mathbf{x}_0^i = v \mid \tilde{\mathbf{z}}_{t_j} = \mathbf{z}) = g_\phi(i \mid \mathbf{z}, t_j)\, p(x_0^i = v \mid \mathbf{z}).$$

On the other hand, the idealized inference chain satisfies

$$\mathbb{P}\Big(\mathbf{z}_{t_{j+1}} = \mathbf{z}^{(i\leftarrow v)} \mid \mathbf{z}_{t_j} = \mathbf{z}\Big) = g_\phi(i \mid \mathbf{z}, t_j)\, p(x_0^i = v \mid \mathbf{z}),$$

since it samples the same index distribution and then draws the revealed token from the same posterior. Therefore, the two chains have identical transition kernels and the same initial condition $\mathbf{z}_{t_0} = \tilde{\mathbf{z}}_{t_0} = (m, \ldots, m)$. It follows by induction on j that $q_{t_j} = \tilde{q}_{t_j}$ for all $j \in \{0, 1, \ldots, K\}$. $\mathbb{P}(\mathbf{x}_0^i = v \mid \tilde{\mathbf{z}}_{t_j} = \mathbf{z}) = p(\mathbf{x}_0^i = v \mid \mathbf{z})$ by the definition of the ground-truth unmasking posterior.

Now fix $i \in \operatorname{msk}(\mathbf{z})$ and $v \in \mathcal{V}$. Using the above conditional identity and the fact that I_j is drawn from $g_\phi(\cdot \mid \mathbf{z}, t_j)$ given the current state, the teacher-forced transition probability is

$$\mathbb{P}\Big(\tilde{\mathbf{z}}_{t_{j+1}} = \mathbf{z}^{(i\leftarrow v)} \mid \tilde{\mathbf{z}}_{t_j} = \mathbf{z}\Big) = g_\phi(i \mid \mathbf{z}, t_j)\, \mathbb{P}(\mathbf{x}_0^i = v \mid \tilde{\mathbf{z}}_{t_j} = \mathbf{z}) = g_\phi(i \mid \mathbf{z}, t_j)\, p(\mathbf{x}_0^i = v \mid \mathbf{z}).$$

On the other hand, the idealized inference chain satisfies

$$\mathbb{P}\Big(\mathbf{z}_{t_{j+1}} = \mathbf{z}^{(i\leftarrow v)} \mid \mathbf{z}_{t_j} = \mathbf{z}\Big) = g_\phi(i \mid \mathbf{z}, t_j)\, p(\mathbf{x}_0^i = v \mid \mathbf{z}),$$

since it samples the same index distribution and then draws the revealed token from the same posterior. Therefore, the two chains have identical transition kernels and the same initial condition $\mathbf{z}_{t_0} = \tilde{\mathbf{z}}_{t_0} = (m, \ldots, m)$. It follows by induction on j that $q_{t_j} = \tilde{q}_{t_j}$ for all $j \in \{0, 1, \ldots, K\}$. $\square$

Remark (continuous-time limit). The discrete-time proof above is stated on a grid and under the one-index-per-step idealization. In the continuous-time limit $K \to \infty$, the corresponding dynamics converge to a continuous-time Markov chain on $\mathcal{Z}$. The same argument applies at the level of rate matrices, which can be understood as the *infinitesimal transition kernel*: conditioned on the current partially revealed state, the next revealed token in the teacher-forced construction has the same conditional distribution as sampling from the ground-truth posterior, hence the two generators coincide. Therefore, the marginal agreement extends to all $t \in [0, 1]$ in the continuous-time formulation.

A.2 Proof of Proposition 2

This section contains the formal version of Proposition 2 and its proof.

We start with a general lemma about the minimizers of weighted cross-entropy loss functions.

Lemma 1 (Weighted cross-entropy is minimized by the true conditional)**.** *Let (X, Y) be any jointly distributed random variables where Y takes values in a finite set $\mathcal{V}$. Let $w(X) \geq 0$ be any measurable function.*

Consider the functional

$$\mathcal{L}(q) := \mathbb{E}\big[w(X)\,(-\log q(Y \mid X))\big],$$

where $q(\cdot \mid x)$ ranges over all conditional distributions on $\mathcal{V}$.

Then any minimizer $q^\star$ satisfies

$$q^\star(\cdot \mid x) = \mathbb{P}(Y = \cdot \mid X = x)$$

for $\mathbb{P}$-almost every x such that $w(x) > 0$. If $w(x) > 0$ holds $\mathbb{P}$-almost surely, the minimizer is unique up to $\mathbb{P}$-a.s. equality.

Proof. Conditioning on $X = x$,

$$\mathbb{E}[-\log q(Y \mid X) \mid X = x] = H\big(\mathbb{P}(Y = \cdot \mid X = x)\big) + \mathrm{KL}\big(\mathbb{P}(Y = \cdot \mid X = x) \,\|\, q(\cdot \mid x)\big).$$

Multiply by $w(x) \geq 0$ and take expectations over X. The entropy term is independent of q, while the KL term is nonnegative and equals 0 iff $q(\cdot \mid x) = \mathbb{P}(Y = \cdot \mid X = x)$. This yields the claim and uniqueness on $\{w > 0\}$. $\square$

Proposition 2 (Formal)**.** *Let the masking process take the form*

$$p(\mathbf{x}_t = \mathbf{z} \mid \mathbf{x}_0, t) \propto \alpha_t(\mathbf{z}) \cdot \mathbf{1}\{\mathbf{x}_0^{\mathbf{um}(\mathbf{z})} = \mathbf{z}^{\mathbf{um}(\mathbf{z})}\},$$

where $\alpha_t : (\mathcal{V} \cup \{\mathbf{m}\})^L \to [0,1]$ is a weighing function. Denote the density over (x_0, t, x_t) induced by this forward process by p_f, i.e.

$$p_f(x_0, t, x_t) = p(x_t|x_0, t)p_{\mathrm{data}}(x_0)\mathbf{1}_{[0,1]}(t),$$

where $\mathbf{1}_{[0,1]}$ is the density of the uniform distribution over $[0,1]$. Then any minimizer f_θ^* of the training loss

$$\mathcal{L}(\theta) = \mathbb{E}_{(\mathbf{x}_0,t,\mathbf{x}_t)\sim p_f}\left[\frac{1}{t}\sum_{i:\, \mathbf{x}_t^i=\mathbf{m}} -\log f_\theta^i(\mathbf{x}_0^i \,|\, \mathbf{x}_t)\right]$$

is the ground-truth posterior of p_{data}, and this posterior is equal to the distribution conditioned on the unmasked indices of x_0 and x_t coinciding, i.e.

$$f_\theta^*(x_0 \mid x_t) = \mathbb{P}(x_0 \mid x_t) \propto \mathbf{1}\{x_0^{\mathbf{um}(x_t)} = x_t^{\mathbf{um}(x_t)}\}\, p_{\mathrm{data}}(x_0).$$

Proof. We start by showing the equality

$$\mathbb{P}(x_0 \mid x_t) \propto \mathbf{1}\{x_0^{\mathbf{um}(x_t)} = x_t^{\mathbf{um}(x_t)}\}\, p_{\mathrm{data}}(x_0).$$

Fix $t \in (0,1)$ and a masked context $x_t \in (\mathcal{V} \cup \{\mathbf{m}\})^L$. Write $\mathbf{um}(x_t) \subseteq [L]$ for the unmasked coordinates and $\mathbf{msk}(x_t) := \{i : x_t^i = \mathbf{m}\}$. By assumption,

$$p(x_t \mid x_0, t) \;\propto\; \alpha_t(x_t)\, \mathbf{1}\{x_0^{\mathbf{um}(x_t)} = x_t^{\mathbf{um}(x_t)}\}.$$

Then Bayes' rule gives, for any x_0,

$$\mathbb{P}(x_0 \mid x_t, t) \;\propto\; p(x_t \mid x_0, t)\, p_{\mathrm{data}}(x_0) \;\propto\; \alpha_t(x_t)\, \mathbf{1}\{x_0^{\mathbf{um}(x_t)} = x_t^{\mathbf{um}(x_t)}\}\, p_{\mathrm{data}}(x_0),$$

The factor $\alpha_t(x_t)$ does not depend on x_0 and cancels after normalization. Furthermore, the dependency on t also disappears, yielding the *data conditional* form

$$\mathbb{P}(x_0 \mid x_t) = \mathbb{P}(x_0 \mid x_t, t) \propto \mathbf{1}\{x_0^{\mathbf{um}(x_t)} = x_t^{\mathbf{um}(x_t)}\}\, p_{\mathrm{data}}(x_0). \tag{3}$$

In particular, the right-hand side depends on x_t only through its revealed coordinates, and thus does not depend on α_t.

Now let $(\mathbf{x}_0, t, \mathbf{x}_t) \sim p_f$. Fix a coordinate $i \in [L]$ and define the random variables

$$X := \mathbf{x}_t, \qquad Y := \mathbf{x}_0^i \in \mathcal{V}.$$

The contribution of coordinate i to the loss is

$$\mathcal{L}_i(\theta) := \mathbb{E}\Big[\frac{1}{t}\, \mathbf{1}\{X^i = \mathbf{m}\}\, \big(-\log f_\theta^i(Y \mid X)\big)\Big].$$

Conditioning on (X, t) and using (3) (which implies $\mathbb{P}(Y = \cdot \mid X, t) = \mathbb{P}(Y = \cdot \mid X)$), we can write

$$\begin{aligned}\mathcal{L}_i(\theta) &= \mathbb{E}\Big[\frac{1}{t}\, \mathbf{1}\{X^i = \mathbf{m}\}\, \mathbb{E}\big[-\log f_\theta^i(Y \mid X) \mid X, t\big]\Big] \\ &= \mathbb{E}\Big[w_i(X)\, \mathbb{E}\big[-\log f_\theta^i(Y \mid X) \mid X\big]\Big],\end{aligned}$$

where $w_i(X) := \frac{1}{t}\, \mathbf{1}\{X^i = \mathbf{m}\} \geq 0$ (note that we are assuming $t > 0$). Thus, by Lemma 1, any minimizer f_θ^* equals the true posterior $\mathbb{P}(x_0 \mid x_t)$, which finishes the proof. □

B Training Details and Experiments

B.1 Omitted details of PUMA.

The overall PUMA pipeline follows Algorithm 1, with one detail on how the number of newly unmasked tokens $|\mathcal{S}_j|$ is determined at each step.

Summarization. The basic idea behind the design is twofold: First, we aim to unmask a roughly uniform fraction of tokens (L/K) at each stage of the diffusion process, while introducing controlled stochasticity in the exact number of tokens unmasked. This stochasticity is important in practice: deterministically unmasking the same number of tokens at every stage can lead to an imbalance in the distribution of masking patterns encountered during training.

Second, in addition to stage-based unmasking, we optionally apply confidence-based thresholding, which further unmasks tokens whose confidence scores exceed a fixed threshold (see Section 4.1).

Stage partitioning. Fix an integer K. We partition the interval $[0, 1]$ into K uniform subintervals $I_\ell := \left[\frac{\ell}{K}, \frac{\ell+1}{K}\right]$ for $\ell = 0, \ldots, K-1$. For an intermediate masked sequence $\mathbf{x}_t$, we define its *stage* $n \in \{0, \ldots, K-1\}$ based on the fraction of unmasked tokens. Concretely, $\mathbf{x}_t$ is said to be at stage n if the number of unmasked (i.e., clean) tokens lies in the range $\left[L_{\text{eff}} \cdot \frac{n}{K}, L_{\text{eff}} \cdot \frac{n+1}{K}\right]$, where L_{eff} denotes the effective sequence length, i.e., the number of tokens in $\mathbf{x}_0$ excluding prompt tokens.

Teacher-forced chain movement. Assume we are given a tuple $(\mathbf{x}_t, n, \mathbf{x}_0)$ consisting of the current masked sequence, its stage index, and the corresponding clean sequence. To advance the teacher-forced chain from stage n to (approximately) stage $n+1$, we proceed as follows:

1. Sample target unmasking ratio: Sample $r \sim \text{Unif}(I_{n+1})$, which specifies the desired fraction of unmasked tokens for the next stage.
2. Determine the number of tokens to unmask: Let U_t denote the number of currently unmasked tokens in $\mathbf{x}_t$. The target number of unmasked tokens is $L_{\text{eff}} \cdot r$, and hence the number of newly unmasked tokens is $\Delta U = L_{\text{eff}} \cdot r - U_t$.
3. Generate the next state: We unmask ΔU tokens according to the PUMA unmasking policy, obtaining the next intermediate sequence, which we nominally associate with stage $n+1$.

Interaction with confidence thresholding. When confidence thresholding is enabled, we additionally unmask all tokens whose confidence scores exceed a fixed threshold τ. Since this step may unmask more tokens than prescribed by ΔU, the total number of unmasked tokens can exceed $L_{\text{eff}} \cdot r$. In this case, after all unmasking operations are completed, we recompute the stage index based on the final number of unmasked tokens, ensuring that the stage assignment remains consistent with the definition above.

Training configuration. Table 1 shows our training and evaluation hyperparameters for both our Sudoku and TinyGSM experiments. All hyperparameters are identical across PUMA and the MDM baseline (except for PUMA-specific parameters, such as the K-schedule, which do not apply to the baseline).

B.2 Additional experimental results

Discussion on the block diffusion experiment. Block diffusion additionally *predefines* the inference-time unmasking order (block-by-block). This rigidity can partially counteract PUMA's speed-up, which in part comes from aligning the training-time masking process with the inference-time unmasking policy. Consequently, while we do not sweep over block sizes, we expect PUMA's incremental gain to decrease as the block size becomes smaller (i.e., as the inference order becomes more constrained).

Importantly, our main claim (Section 5.3) is that PUMA is complementary to block-size warmup: the warmup schedule necessarily passes through a regime with relatively large blocks, and we expect this is precisely the regime where PUMA's policy-alignment effect applies most strongly.

Sudoku training speedup. Figure 4 shows the training speedup achieved by PUMA over the MDM baseline in our training run from Section 5.1.

Sudoku training trajectories. Figure 6 shows the distance between training trajectories found by PUMA over the course of training to the trajectories found by the final model. Here, the distance at training time t is defined as

$$d_{\text{abs}}(t, T) \;:=\; \mathbb{E}_{x_0}\left[\frac{1}{81}\sum_{(i,j)\in[9]\times[9]} \left|u_{ij}^t(x_0) - u_{ij}^T(x_0)\right|\right], \tag{4}$$

where T is the time at the end of training, and u_{ij}^t is the integer *step* in the trajectory at which cell (i, j) is unmasked; e.g., if a cell is unmasked at the kth step in the trajectory, $u_{ij}^t = k$. In Figure 6, we average over 100 (fixed) samples. The figure shows that the distance decreases rapidly early during training, which validates using the online trajectories from the current model for teacher-forced chains. However, interestingly, the distance saturates at around 0.5, which means a small gap in trajectory distance persists.

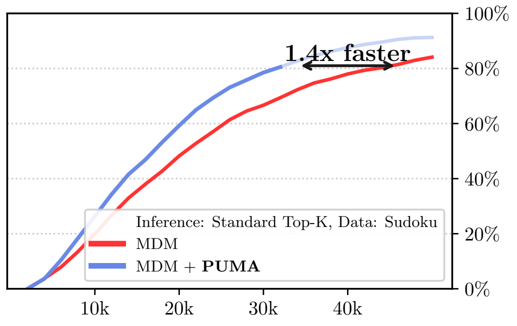


Figure 4: PUMA speeds up training on Sudoku by $1.4\times$.

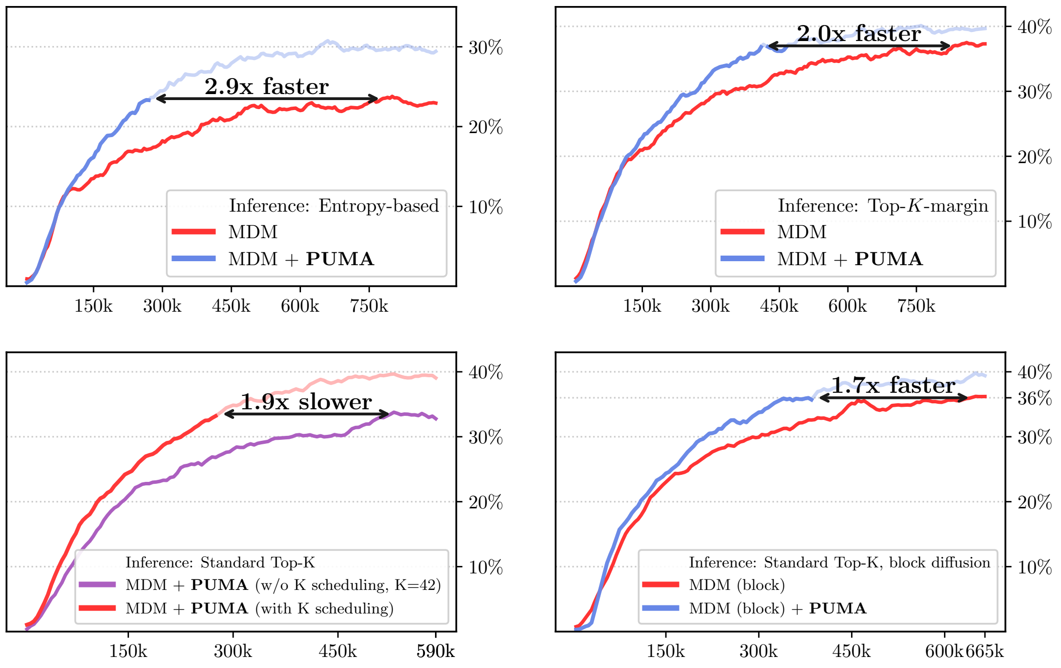


Figure 5: Lower Left: K scheduling is an important factor for PUMA's efficiency; without scheduling ($K = 42$) degrades the iteration-accuracy efficiency. (purple curve). **Upper**: A single PUMA-trained model (trained under one policy) remains robust to inference-time policy choices, consistently outperforming the baseline across different unmasking policies. Figure 3 shows the iteration-accuracy curves with $|S| = 2$. Here, we present the curves with $|S| = 3$ under the Top-K margin (Upper Right) and entropy-based (Upper Left) policy. **Lower Right**: PUMA also yields speed-up for the block diffusion training with the block size 256.

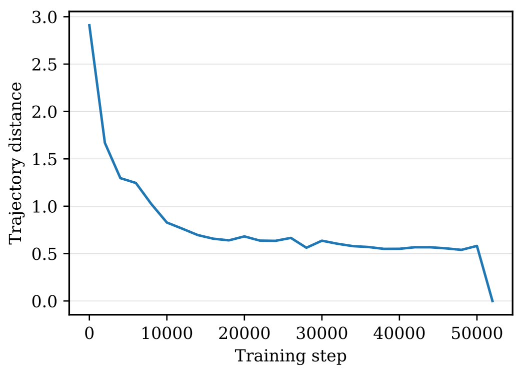


Figure 6: Training trajectory distance to the fully trained model diminishes quickly. We show the distance (4) between the training trajectory found by PUMA over the course of training, averaged over a fixed set of 100 samples on Sudoku.

Table 1: Training and evaluation hyperparameters on Sudoku and TinyGSM.

	Sudoku	**TinyGSM**
Tokenizer	Custom	Qwen/Qwen2-0.5B
Vocab size	10	151645
Model size	6.8M	125M
Hidden dim	256	512
MLP dim	768	1536
Number of layers	8	14
Number of heads	8	8
Max length	162	512
RMSNorm epsilon	$1e{-6}$	$1e{-6}$
Dropout	0	0
Confidence threshold	0.9	0.9
K schedule	8 (fixed)	12–42 (in increments of 3, with every 30k)
Learning rate	$3e{-4}$	$3e{-4}$
Warmup steps	1000	1000
EMA value	0.9999	0.9999
Weight decay	0.01	0.01
Epochs	2	20
Max grad norm	1	1
Batch size (per GPU)	32	32
Number of GPUs	2	8
Sampling temperature	0.0	0.0