ADAPTIVE ORDER POLICIES FOR MASKED DIFFUSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Masked diffusion models have seen great success in capturing data distributions over discrete sequences in domains such as text and proteins. These models generate data by iteratively unmasking tokens starting from a fully masked sequence, with the unmasking order typically chosen at random or using a heuristic based on denoiser probabilities. In this work, we propose a scheme for learning the unmasking order using an additional lightweight policy network on top of a diffusion model. Our proposed loss reweights terms in the masked diffusion loss according to policy probabilities, and results in a policy that prefers positions where the denoiser is more likely to be correct. We study this loss in two settings: (i) training solely the policy while using a frozen pre-trained denoiser, and (ii) training the policy and denoiser jointly with the weighted loss to allow for mutual adaptation. We demonstrate that our approach outperforms common heuristics on problems that are sensitive to token ordering, such as combinatorial problems, proteins as well as various coding and language tasks.

1 INTRODUCTION

Diffusion models have established themselves as a powerful paradigm for generative modeling, achieving remarkable success in continuous domains such as images (Ho et al., 2020; Saharia et al., 2022; Rombach et al., 2022) and molecular structures (Watson et al., 2023; Abramson et al., 2024). More recently, *discrete* diffusion models – which operate directly on token sequences by iteratively masking and unmasking – have shown strong results in language modeling (Sahoo et al., 2024; Nie et al., 2025; Shi et al., 2024), protein design (Alamdari et al., 2023; Wang et al., 2024b), and drug discovery (Lee et al., 2025). (Huang et al., 2022)(Chen & Lipman, 2024)(Austin et al., 2021a) (Gat et al., 2024)

A key design choice in masked diffusion models (MDM) is the *order* in which tokens are unmasked during generation. The standard approach selects positions uniformly at random. However, practitioners have found that heuristic ordering strategies – such as unmasking the most confident position first (Nie et al., 2025) or the position with the largest probability margin (Kim et al., 2025) – can dramatically improve sample quality on downstream tasks. This effect is particularly pronounced on constraint satisfaction problems such as Sudoku and Boolean satisfiability (3-SAT), where the unmasking order directly impacts whether the model can propagate constraints correctly.

Despite the empirical success of heuristic orderings, these remain hand-designed and may be suboptimal for a given model and dataset. A natural question arises: *can we learn the unmasking order?* That is, rather than relying on fixed heuristics, can we train a lightweight auxiliary network to predict which positions to unmask, conditioned on the current partially masked sequence?

In this work, we propose a simple approach for learning adaptive unmasking orderings in MDMs. Our approach introduces a policy network $q^{\phi}(i \mid x_t)$ and a modified cross-entropy objective that can be used either to train a lightweight policy layer on top of a pretrained masked diffusion model or to jointly train the policy and denoiser. The objective weights policy probabilities by the cross entropy of the denoiser at each token position, encouraging the policy to select positions that are most informative for generation. Under joint training, the objective also optimizes the denoiser to more correctly predict positions which the policy is likely to unmask. Across structured tasks, such as Sudoku, 3-SAT, and DPLM protein generation, learned ordering improves over heuristic baselines in both policy-only and joint-training settings. Policy-only adaptation also improves performance in larger-scale settings, such as with Open-dLLM-0.5B and LLaDA-8B across code, language, and

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

reasoning tasks. The policy is lightweight, adding less than 1% to the denoiser's parameter count and requiring far fewer training updates than the base model. In addition to performance over baselines, we observe a number of benefits of the learned policy over heuristics, including: i) robustness to stochastic decoding, ii) improved efficiency at inference - retaining higher performance at fewer diffusion steps, iii) improved heuristic decoding when the denoiser is trained with policy-aware weighting.

2 METHOD

2.1 MASKED DIFFUSION MODELS

Throughout this work, we denote a sequence of length L as $x = (x^1, \dots, x^L) \in \mathcal{V}^L$, with tokens taking values in some vocabulary set $x^i \in \mathcal{V}$. We consider the case of masked diffusion, where a special masking token m is included in the vocabulary set. Other notation includes: the Kronecker symbol $\delta(i, j)$ (equal to 1 for $i = j$ and 0 otherwise), $\mathrm{Cat}(x; p)$ to denote the categorical distribution with probabilities p, and Δ^k to denote the probability simplex over k dimensions.

Masked diffusion models (MDMs) use a noising process to map the data distribution $p_{\mathrm{data}}(x)$ at time 0 to the delta distribution at the fully masked state $M = (m, \dots, m)$, $p_1(x) = \delta(x, M)$ at time 1. A typical noising process consists of converting a data token x_0^i into the masked token with some probability $1 - \alpha_t$, independently over dimensions (Sahoo et al., 2024): $p(x_t \mid x_0) = \prod_{i=1}^{L} \alpha_t \delta(x_t^i, x_0^i) + (1 - \alpha_t)\delta(x_t^i, m)$. The parameter α_t denotes a decreasing noise schedule, with $\alpha_0 = 1$ and $\alpha_1 = 0$. A typical choice is the linear schedule $\alpha_t = 1 - t$.

For reversing this process, a neural network parameterizes a distribution over clean data x_0 conditioned on the partially masked sequence x_t. In particular, the network outputs an independent distribution over each token position i, as $\mu^\theta(x_t)[i, \cdot] \in \Delta^{|\mathcal{V}|}$, which satisfies $\mu^\theta(x_t)[i, m] = 0$ (the clean data cannot contain masks) and $\mu^\theta(x_t)[i, x_t^i] = 1$ if $x_t^i \neq m$ (the clean data approximation retains unmasked positions in x_t). The function μ^θ is referred to as the denoiser.

Given a denoiser, the reverse transition over two nearby time-steps $s < t$ is (Sahoo et al., 2024):

$$p^\theta(x_s^i \mid x_t) = \begin{cases} \mathrm{Cat}\left(x_s^i; \frac{1-\alpha_s}{1-\alpha_t}\delta(\cdot, m) + \frac{\alpha_s - \alpha_t}{1-\alpha_t}\mu^\theta(x_t)[i, \cdot]\right) & \text{if } x_t^i = m \\ \mathrm{Cat}(x_s^i; \delta(\cdot, x_t^i)) & \text{if } x_t^i \neq m \end{cases} \tag{1}$$

Let $\mathrm{CE}(i, p)$ denote the cross entropy loss with sample i and probability p: $\mathrm{CE}(i, p) = -\log p[i]$. The goal is to train $\mu^\theta(x_t)$ to match the factored posterior $\prod_{i=1}^{L} p(x_0^i \mid x_t)$. For such a denoiser, the transitions in Equation (1) correctly reverse the noising process and recover the data distribution at time 0 (Sahoo et al., 2024; Shi et al., 2024). The training objective for μ^θ is the weighted cross entropy loss:

$$\mathcal{L}_{\mathrm{MDM}}(\theta) = \mathbb{E}_{t \sim \mathcal{U}[0,1], x_t \sim p(x_t \mid x_0), x_0 \sim p_{\mathrm{data}}} \left[\frac{-\alpha_t'}{1 - \alpha_t} \sum_{i=1}^{L} \delta(x_t^i, m) \mathrm{CE}(x_0^i, \mu^\theta(x_t)[i, \cdot]) \right] \tag{2}$$

With a trained denoiser, the generation process consists of starting with a completely masked sequence $x_1 = M$ and iteratively unmasking tokens in the sequence over T steps to obtain a final sample x_0. One step of this sampling procedure is done by simulating Equation (1), which involves:

(i) Sampling an approximation of clean data from the denoiser $\hat{x}_0 \sim \mu^\theta(x_t)$

(ii) Randomly choosing which (currently masked) positions I in x_t to unmask (by replacing $x_t^i = m$ with $\hat{x}_0^i$ for all $i \in I$).

Properly simulating Equation (1) requires random selection of the set of unmasking positions I (Sahoo et al., 2024). However, a number of works have found success in selecting I through some heuristic informed by the denoiser logits $\mu^\theta(x_t)$ (Nie et al., 2025; Kim et al., 2025; Ben-Hamu et al., 2025). These involve calculating a score s_i and then prioritizing unmasking positions i with higher score (possibly with added noise).

Some options for the score calculation which have been investigated in previous work include choices such as (i) **High confidence**: The probability of the sampled clean tokens $s_i = \mu^\theta(x_t)[i, \hat{x}_0^i]$ (Nie

et al., 2025), (ii) **Margin**: The probability margin, i.e. the gap between the highest probability and the second highest probability $s_i = \mu^\theta(x_t)[i, j_1] - \mu^\theta(x_t)[i, j_2]$ where $j_1 = \arg\max_j \mu^\theta(x_t)[i, j]$ and $j_2 = \arg\max_{j \neq j_1} \mu^\theta(x_t)[i, j]$ (Kim et al., 2025), and (iii) **Entropy**: The negative entropy of the position $s_i = -H(\mu^\theta(x_t)[i, \cdot])$ (Ben-Hamu et al., 2025).

These heuristics have been shown to yield better performance on a number of downstream tasks (such as coding and math), or on tasks such as Sudoku, where generating a valid solution is highly dependent on the order of unmasking (Nie et al., 2025; Kim et al., 2025).

The problem of selecting the unmasking ordering also informs the efficiency of performing inference, since being able to unmask multiple tokens leads to fewer calls to the denoising model to generate a complete sequence. Intuitively, we expect certain token positions to be independent of others, and we would expect unmasking them in parallel to retain the same performance as unmasking one at a time.

2.2 Learnable Adaptive Order Policies

Given the importance of the unmasking ordering, we ask the following question: armed with a dataset, and a pre-trained denoiser μ^θ, can we train a lightweight *policy network* $q^\phi(i \mid x_t)$ which outputs a distribution over the next token position i to unmask for x_t? The hope is to add a small number of additional trainable parameters, and spend some additional training iterations to obtain better performance on challenging tasks.

We can note that MDMs are sensitive to the token ordering precisely due to imperfections in the denoiser model. As argued by Ben-Hamu et al. (2025) a perfect denoiser is able to sample from the target regardless of the unmasking order (since it corresponds to different factorizations of the joint distribution, according to the chain rule of probability). Therefore, a reasonable objective for the policy is one that accounts for the loss of the denoiser model. Based on this, we propose to train the policy to place higher probability on positions which obtain a smaller loss, as measured by the cross entropy. This is captured in the objective:

$$\mathcal{L}_{\text{ORDER}}(\phi) = \mathbb{E}_{t\sim\mathcal{U}[0,1], x_t\sim p(x_t|x_0), x_0\sim p_{\text{data}}} \left[\frac{-\alpha_t'}{1-\alpha_t} \sum_{i=1}^{L} q^\phi(i \mid x_t) \text{CE}(x_0^i, \mu^\theta(x_t)[i, \cdot]) \right] \quad (3)$$

Note that we assume $q(i \mid x_t) = 0$ for unmasked positions $x_t^i \neq m$. This is identical to the MDM objective Equation (2) except for the fact that the sum over masked positions is weighted by the policy q^ϕ. In particular, for a uniform policy over masked tokens, we recover, up to multiplication, the vanilla MDM loss.

We can further justify this choice of objective by relating it to an ELBO bound for the policy-guided unmasking process. Using an ELBO decomposition for such a process, derived by Peng et al. (2025a), we can show that our loss in Equation (3) results from making approximations to avoid computing intractable terms. We present the details of this derivation in Appendix C.

For a frozen denoiser network θ, the policy network q^ϕ is trained to predict where the current denoiser is most likely correct. The optimal policy places all probability on the position with smallest cross entropy loss. We will refer to this theoretically optimal policy as the **oracle**: $q^{\text{oracle}}(i \mid x_t) = \delta(i, \arg\min_{j, x_t^j = m} \text{CE}(x_0^j, \mu^\theta(x_t)[j, \cdot]))$. We empirically validate the choice of this loss by evaluating the oracle policy (with access to ground truth data x_0), and confirming that it improves metrics relative to other heuristic samplers in Section 3. Other approaches for training unmasking policies typically rely on more expensive gradient estimation, or RL procedures for tasks involving a reward. These are discussed in Section 4.

Finally we note that for sampling multiple positions, we can use the policy probabilities as score values $s_i = q^\phi(i \mid x_t)$ and use them in a similar way to other heuristics (for instance, unmasking the k positions with the largest policy probabilities).

2.3 Policy-Aware Denoiser Training

The policy objective in Equation (3) answers where the model should unmask next, but it also raises two related questions: can the policy improve the denoiser itself, and can the denoiser be trained while explicitly knowing that a policy will be used at sampling time? We note that the

Algorithm 1: Training with Policy-Only or Joint Updates

Input: dataset $\mathcal{D}$, denoiser μ^θ, policy q^ϕ, noise schedule α_t

while *not converged* **do**

Sample $x_0 \sim \mathcal{D}$, $t \sim \mathcal{U}[0,1]$, and $x_t \sim p(x_t \mid x_0)$

Compute denoiser logits $\mu^\theta(x_t)$ and policy probabilities $q^\phi(i \mid x_t)$ over masked positions

Compute token losses $\ell_i \leftarrow \delta(x_t^i, m)\,\mathrm{CE}(x_0^i, \mu^\theta(x_t)[i,\cdot])$ for $i = 1, \ldots, L$

Define

$$\mathcal{L}_{\mathrm{ORDER}}(\phi) = \frac{-\alpha_t'}{1-\alpha_t} \sum_{i=1}^{L} q^\phi(i \mid x_t)\ \mathrm{stopgrad}(\ell_i)$$

if *joint training is enabled* **then**

Define

$$\mathcal{L}_{\text{PA-MDM}}(\theta) = \frac{-\alpha_t'}{1-\alpha_t} \sum_{i=1}^{L} \Big(1 + \mathrm{stopgrad}(q^\phi(i \mid x_t))\Big)\, \ell_i$$

Define $\mathcal{L}_{\mathrm{total}} = \mathcal{L}_{\mathrm{ORDER}}(\phi) + \mathcal{L}_{\text{PA-MDM}}(\theta)$ `// joint training`

else

Define $\mathcal{L}_{\mathrm{total}} = \mathcal{L}_{\mathrm{ORDER}}(\phi)$ `// policy-only training`

Update trainable parameters using $\mathcal{L}_{\mathrm{total}}$

objective in Equation (3) already allows for gradients with respect to the denoiser parameters θ, and admits a sensible interpretation. Namely, viewed from the perspective of training the denoiser, the objective upweights the loss at positions proportional to the probability the policy will select them for unmasking.

A slight issue may occur with directly using Equation (3) for training the denoiser, since the policy may quickly collapse to a handful of positions and prevent denoiser training on most masked tokens. We alleviate this with a simple modification: we add the original MDM loss to the policy-weighted objective to ensure better training signal. This is a stability trick used in other works (Peng et al., 2025a).

Putting together these observations, we propose the following denoiser loss as a substitute to the vanilla MDM objective:

$$\mathcal{L}_{\text{PA-MDM}}(\theta) = \mathbb{E}\left[\frac{-\alpha_t'}{1-\alpha_t} \sum_{i=1}^{L} \delta(x_t^i, m)\,\Big(1 + q^\phi(i \mid x_t)\Big)\,\mathrm{CE}(x_0^i, \mu^\theta(x_t)[i,\cdot])\right]. \tag{4}$$

In implementation, the policy probabilities are detached inside the denoiser loss, so gradients with respect to θ do not backpropagate through q^ϕ.

The policy itself is still trained with Equation (3), with the denoiser losses detached when optimizing ϕ. Joint training therefore lets the policy identify consequential positions, while simultaneously teaching the denoiser to allocate more capacity to them. The algorithm for either policy-only training or joint training is summarized in Alg. 1.

To separate the effect of *learned* policy guidance from the effect of simply reweighting the denoiser loss, we can also compare against heuristic-aware denoiser objectives, similar to those in (Peng et al., 2025a). Concretely, we can replace $q^\phi(i \mid x_t)$ in Equation (4) by normalized weights derived from standard confidence scores computed from the denoiser logits, such as the confidence (log-probability of the sampled token) or the margin. This yields a matched control where the denoiser is still trained with emphasis on selected positions, but the emphasis is determined by a fixed heuristic rather than a learned policy.

Such a comparison isolates the main question of interest: whether informing the denoiser about the existence of a learned unmasking policy during training provides benefits beyond the gains obtainable from generic heuristic reweighting alone.

3 EXPERIMENTS

We evaluate our adaptive ordering policy approach on two constraint-satisfaction tasks known to be sensitive to token ordering, Sudoku puzzle solving and 3-SAT (Boolean satisfiability with 3 literals per clause) (Kim et al., 2025; Ye et al., 2024); protein sequence generation with DPLM (Wang et al., 2024b); code generation with Open-dLLM-0.5B (Peng et al., 2025b); and six language and reasoning benchmarks with LLaDA-8B (Nie et al., 2025).

Policy architecture. We parameterize q^ϕ as a lightweight per-token MLP that conditions on both the denoiser's confidence scores (max log-probability) and the hidden states from the last layer of the base model. Specifically, each scalar confidence score is projected to the hidden dimension (d=384) via a linear layer, summed with the corresponding hidden-state vector, and passed through a two-layer MLP (hidden dimension 128, ReLU activation) that outputs a per-position routing logit. The policy adds $\sim$50K parameters ($<$1% of the base model). In the policy-only setting, it is trained on top of a frozen denoiser using Equation (3), while in the joint-training setting it is optimized together with the denoiser under the objectives described in Section 2.3. We also experimented with a transformer-based policy variant (Jazbec et al., 2025), which did not yield further improvement (see Appendix B).

Baselines. At inference, we compare against baselines such as **High conf.**, **Margin**, **Entropy**, **Random**, and **Oracle** ordering strategies described in Section 2, based on applicability and relevance. For our joint training experiments, we train using the modified denoiser objective from Equation (4), with baselines being trained using heuristic weightings, as described in Section 2.3.

Task Details. For Sudoku and 3-SAT we use a 6M-parameter GPT-2 denoiser trained as an MDM with 20 and 10 denoising steps respectively. For protein evaluation we adapt the DPLM-150M model (Wang et al., 2024a) and evaluate it for both policy-only and joint policy-denoiser training. For large-scale settings, we do policy-only training with Open-dLLM-0.5B for code generation, and LLaDA-8B across code, reasoning, question answering, translation, and summarization tasks. The Open-dLLM policy is trained on FineCode, while the LLaDA policy uses a mixture of general web, mathematical, and code data. Full training and evaluation details are provided in Appendix A.

3.1 RESULTS ON STRUCTURED DATA

Policy-only training. The upper block of Table 1 summarizes policy-only adaptation with a frozen denoiser using Equation (3). Across Sudoku, 3-SAT, and protein generation with DPLM, the learned policy outperforms the heuristic baselines. On DPLM-150M, the policy improves predicted foldability while maintaining diversity close to the confidence and margin orderings, as also shown in the top row of Figure 1. This establishes that even without modifying the base denoiser, learning the unmasking order alone yields consistent gains across domains.

Table 1: Main results for policy-only and joint training. We report mean accuracy for Sudoku and 3-SAT, and mean pLDDT on sequences of length $\{100, 200, 300, 400, 500\}$ for DPLM-150M. All results are averaged over 3 seeds. Higher is better; best within each block in **bold**.

	Sudoku	**3-SAT**	**DPLM-150M**
Policy-only training			
Random	07.33 $\pm$ 0.07	87.23 $\pm$ 0.74	40.04 $\pm$ 0.78
Entropy	25.48 $\pm$ 0.15	87.63 $\pm$ 0.25	55.88 $\pm$ 0.93
High conf.	18.71 $\pm$ 0.16	86.87 $\pm$ 0.76	82.20 $\pm$ 0.76
Margin	91.53 $\pm$ 0.04	88.33 $\pm$ 0.15	82.02 $\pm$ 0.31
Policy	**93.75 $\pm$ 0.20**	**89.00 $\pm$ 0.00**	**83.85 $\pm$ 0.80**
Oracle policy	100.00 $\pm$ 0.00	93.90 $\pm$ 0.00	N/A
Joint training			
Base-model	18.71 $\pm$ 0.16	86.87 $\pm$ 0.76	82.20 $\pm$ 0.76
High conf.	19.44 $\pm$ 0.03	88.80 $\pm$ 0.26	82.85 $\pm$ 0.85
Margin	90.23 $\pm$ 0.03	88.67 $\pm$ 0.15	82.28 $\pm$ 0.53
Policy	**93.52 $\pm$ 0.11**	**90.57 $\pm$ 0.06**	**84.94 $\pm$ 1.00**

The policy adds $<$50K parameters ($<$1% of the base model).

Joint training. The lower block of Table 1 evaluates the policy-aware denoiser objective from Equation (4). Here, *High conf.* and *Margin* replace the learned policy weights with normalized weights derived from the confidence and margin heuristics respectively. The main pattern is that learned policy reweighting improves the denoiser more reliably than matched heuristic reweighting. On Sudoku and 3-SAT, policy-aware scaling gives the strongest stochastic results among the compared training objectives. The same comparison also extends to DPLM, where

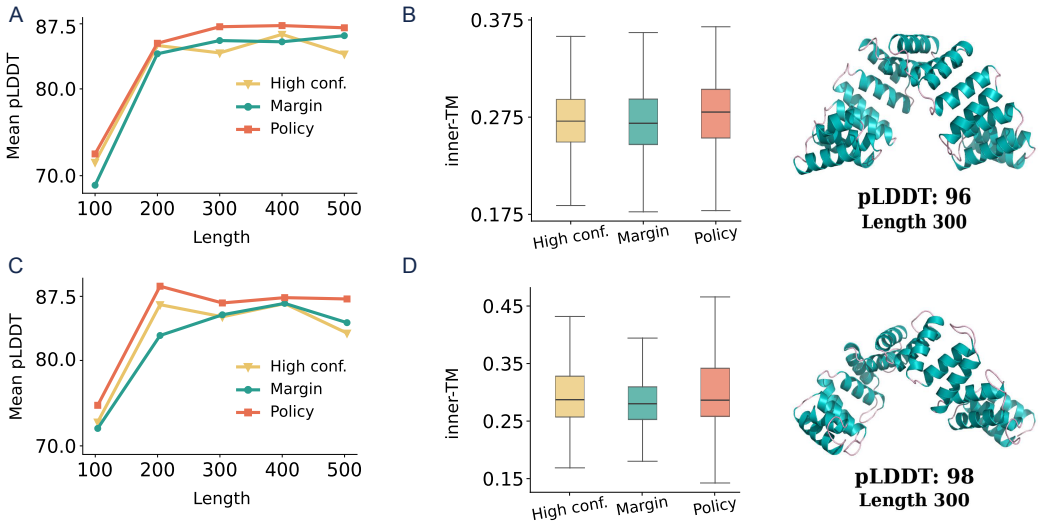

Figure 1: Adaptive ordering on DPLM-150M. Top (A, B): policy-only adaptation. Bottom (C, D): joint training. Left: mean pLDDT by target sequence length. Right: inner-TM diversity, where lower is better. Representative predicted 3D folds illustrate jointly trained samples across different sequence lengths. The learned policy improves foldability while maintaining diversity close to the heuristic baselines.

the learned policy-weighted objective improves protein generation relative to the heuristic-weighted controls, as also illustrated in the bottom row of Figure 1. Taken together, these results show that the policy is useful not only as an inference-time ordering policy but also as a training signal for the denoiser.

Protein sequence generation. For protein generation, we conduct a more comprehensive evaluation of our joint policy-denoiser model by generating 100 sequences at lengths 200, 300, . . . , 800, which are then folded into 3D structures using ESMFold (Lin et al., 2022). Results are shown in Table 2, which reports structure quality, foldability, and diversity metrics (the precise metric definitions collected in Appendix A.2.2). We note that both our results and those of Peng et al. (2025a) reported in the table use the same number of parameters (150M).

Table 2: Protein sequence generation results including our joint-policy, with baselines results taken from Peng et al. (2025a). Each model generates 100 sequences at lengths 200, 300, . . . , 800, which are folded into 3D structures using ESMFold. Structure quality is measured by pLDDT, pTM, and pAE, while diversity is measured by token entropy and sequence uniqueness. Foldability is the percent of sequences with pLDDT $>$ 80, pTM $>$ 0.7, and pAE $<$ 10. Best values in each column are shown in **bold**.

Model	pLDDT ↑	pTM ↑	pAE ↓	Foldability (%) ↑	Entropy ↑	Diversity (%) ↑
Large						
ESM3	34.13	0.23	24.65	1.50	3.99	**93.44**
ProGen2-medium	57.94	0.38	20.81	12.75	2.91	91.45
ProGen2-large	55.07	0.35	22.00	11.87	2.73	91.48
DPLM-650M	79.53	0.66	11.85	49.14	3.18	92.22
150M-scale						
EvoDiff	31.84	0.21	24.76	0.43	4.05	93.19
ProGen2-small	49.38	0.28	23.38	4.48	2.55	89.31
DPLM-150M	80.23	0.65	12.07	48.14	3.14	92.80
DLM-150M	81.32	0.65	12.00	42.43	3.21	92.45
DLM-150M + PAPL	81.48	0.72	**8.97**	**59.40**	3.12	91.73
Joint policy (ours)	**86.43**	**0.76**	9.68	54.14	**4.12**	93.06

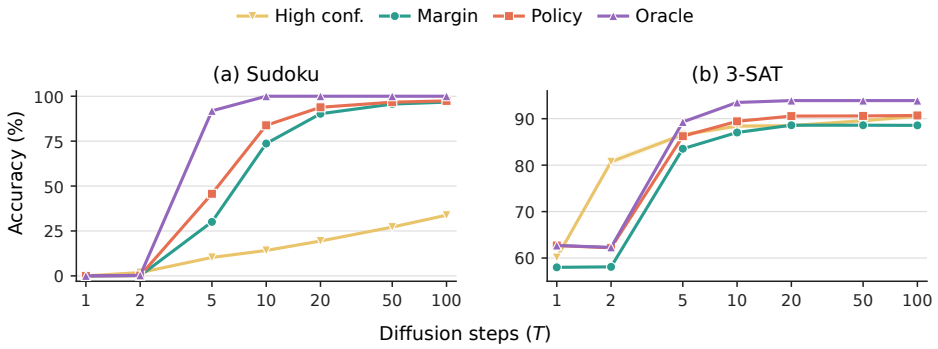

Figure 4: Accuracy as a function of the number of reverse diffusion steps T on Sudoku (left) and 3-SAT (right). Better ordering is especially valuable at small step budgets, where improved unmasking policies can recover substantially more accuracy for the same inference cost.

Transfer to heuristic decoding. An important property of policy-aware training is that its benefits are not limited to decoding with the learned policy itself. As shown in Figure 2, the denoiser trained with policy-aware scaling also improves heuristic decoding on both tasks. This suggests that the policy-weighted objective does not merely tailor the denoiser to one decoding rule, but improves the underlying denoiser in a way that transfers across different unmasking heuristics.

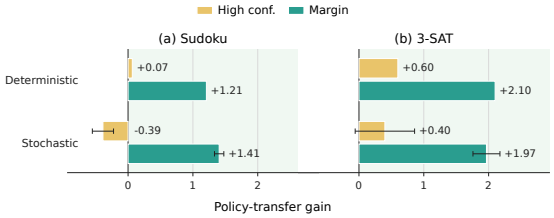

Figure 2: Transfer from policy-weighted training to heuristic decoding. We plot the accuracy difference between the denoiser trained with policy-weighted and heuristic weighted joint objectives, and when decoding with the matched heuristic. Policy-weighted training improves both heuristics.

3.2 Ablations

Deterministic versus stochastic decoding. We study the effect of decoding noise in Figure 3. We evaluate the methods on different *decoding strategies*: namely deterministic decoding which unmasks the highest scoring position each step, and stochastic, which adds Gumbel noise to this. One notable observation is the discrepancy between high-confidence Sudoku accuracy under deterministic and stochastic decoding. The latter is reported by Kim et al. (2025), who propose the margin heuristic as an improvement. We find that this gap is primarily due to the sensitivity of the high-confidence heuristic to the decoding strategy. In fact, with deterministic decoding, it outperforms the margin heuristic. The margin heuristic is less sensitive to decoding. Our learned policy is similarly robust across decoding regimes. The lower block shows the same pattern in joint training. Therefore, the policy-weighted objective avoids pathological changes in performance due to decoding parameters, while outperforming the alternative methods.

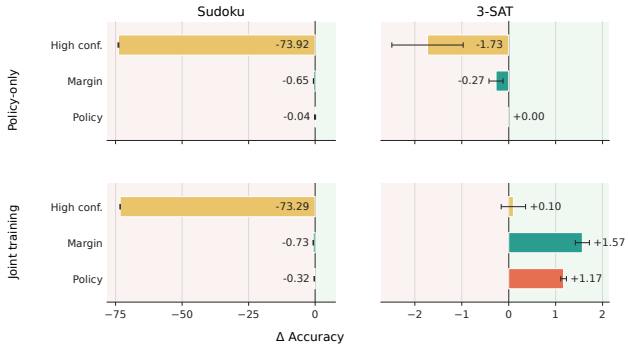

Figure 3: Deterministic and stochastic decoding on combinatorial tasks. We compute Δ as the difference between stochastic accuracy and deterministic accuracy. Negative values indicate degradation under stochastic decoding.

Efficiency as a function of diffusion steps. We also study how ordering quality interacts with the number of reverse steps T, which directly controls inference cost. As shown in Figure 4, the learned ordering improves efficiency by achieving higher accuracy for the same step budget. On Sudoku, the learned policy substantially outperforms both heuristic baselines and nearly matches the oracle at T=100. On 3-SAT, the learned policy consistently improves over the high-confidence heuristic

and is competitive with, or slightly better than, the margin heuristic across moderate and large step budgets. In both tasks, the oracle remains better than the learned policy, indicating that there is still substantial room to improve learned unmasking strategies. Overall, these results show that ordering affects both accuracy and inference efficiency.

Comparison with additional denoiser training Given a pre-trained denoiser network, one could choose to improve the model either by training the policy-network and freezing the denoiser (as in our algorithm), or simply continuing to train the denoiser (with the standard MDM loss). We compare against this alternative for the Sudoku task, with a matched training budget, in Table 3. We observe that continued denoiser training yields little improvement, with both the high-confidence and margin heuristics. In contrast, the policy-only training more substantially improves the accuracy. We also note that updating the lightweight policy is cheaper per step than full denoiser training (owing to fewer parameters).

Table 3: Comparison with continued training on the denoiser. Higher is better; best in **bold**.

Ordering	Sudoku
High Conf.	18.71 ± 0.16
Margin	91.53 ± 0.04
High Conf. (Continued)	18.73 ± 0.14
Margin (Continued)	91.52 ± 0.04
Policy	$\mathbf{93.75 \pm 0.20}$

3.3 Language and code generation

Table 4: Best pass@1 (%) on Open-dLLM-0.5B code benchmarks. Higher is better; best in **bold**.

Ordering	HumanEval	HumanEval+	MBPP	MBPP+
High conf.	17.62 ± 2.45	15.61 ± 2.34	15.58 ± 1.24	$\mathbf{23.28 \pm 1.65}$
Margin	12.56 ± 1.78	11.65 ± 1.79	8.72 ± 0.70	13.47 ± 1.00
Policy	$\mathbf{18.90 \pm 3.00}$	$\mathbf{16.46 \pm 2.90}$	$\mathbf{16.20 \pm 1.31}$	22.80 ± 1.76

Open-dLLM code generation. We train only the policy on the 0.5B-parameter Open-dLLM code model, whose Qwen2-based backbone has a vocabulary of 151,936 tokens (Peng et al., 2025b; Yang et al., 2024). Table 4 reports the best pass@1 obtained for each ordering method on each benchmark after independently sweeping its decoding hyperparameters. The policy outperforms high-confidence ordering on three of four datasets and margin ordering on all four.

LLaDA language and reasoning. We additionally train the policy on LLaDA-8B and evaluate on various coding and language tasks (with details on tasks and metrics in Section A.3). The policy outperforms high-confidence ordering on three and ties on one (Table 5). We further evaluate the policy on the four tasks where it has the smallest gains, using $1/4$th the denoising steps. The policy's relative performance improves on all four, reinforcing the earlier observation that the learned policy supports few-step inference better than competing heuristics. Importantly, our supervised objective uses a general (non-task specific) training set, and does not require constructing a task-specific reward. This renders the method applicable to code and math tasks, where verifiable rewards may be available, as well as question answering, translation, and summarization, where reward design is less direct.

Table 5: Policy-only training results on LLaDA-8B. The top table decodes with 1 token/step, while the bottom one does 4 tokens/step

Ordering	HumanEval	BBH	CoQA	WMT16	MBPP	XSum
High conf.	**34.15**	**44.82**	**80.48**	12.031	39.60	23.051
Policy	**34.15**	43.96	78.83	**12.076**	**39.80**	**23.265**

Ordering	HumanEval	BBH	CoQA	WMT16
High conf.	12.80	**41.94**	76.93	7.666
Policy	**14.02**	41.41	**77.34**	**8.020**

3.4 Discussion

Several observations emerge from these results. First, the learned policy improves over heuristic orderings on both structured tasks (constraint-satisfaction, proteins) and on broader text tasks (coding, language), using a lightweight auxiliary network with less than 1% parameter overhead. The latter

setting demosntrates the scalability of our method to large model and vocabulary sizes. Second, our experiments suggest that the learned policy is more amenable to parallel decoding, and generally outperforms other unmasking strategies at fewer diffusion steps. Third, the policy is useful both as an inference-time ordering rule and as a training signal for the denoiser (even when using other ordering strategies). Finally, the remaining gap to the oracle suggests that better policies are still achievable.

4 Related Works

Wang et al. (2025) also train a policy for the unmasking order, by treating the order as a latent variable z. They propose a variational method for optimizing it, which requires parameterizing the posterior approximation $q^{\phi}(z \mid x)$, in addition to the trainable policy over orders $p^{\theta}(z \mid x)$. The former is only used as part of the training objective for the latter, and not during inference. In addition, the optimization of the variational posterior requires gradient estimation techniques to reduce variance, and complicates the optimization loop. Our loss by contrast is much simpler.

Another set of works assume access to verifiable reward functions rather than a dataset (Hong et al., 2025; Jazbec et al., 2025). These frame the generation process of a masked diffusion model as a Markov Decision Process and optimize the unmasking policy using RL objectives. By contrast, our supervised objective requires data but no task-specific reward construction. It therefore supports both domains where verifiable rewards could be defined, such as code and mathematical reasoning, and domains where a suitable reward is less direct such as summarization and protein generation.

Peng et al. (2025a) propose a modification of the MDM loss which accounts for using heuristics when determining the unmasking order (rather than random unmasking). The loss resembles our objective Equation (4), and we outline the connections between our objective and their ELBO bound in Appendix C. The main difference with our work is that our objective makes necessary simplifications to enable joint training of the denoiser and policy (as opposed to only training the denoiser, as done in their work).

Recent works also learn auxiliary token-level scoring modules that can be used to remask tokens during inference, to improve generation quality. Huang et al. (2025) train a confidence head for self-reflective remasking in diffusion language models: for unmasked tokens in x_t, the head predicts whether they are correct, while for masked positions it is supervised by the denoiser's probability of recovering the ground-truth token. Meshchaninov et al. (2025) similarly fine-tune a lightweight error-prediction layer on top of a pretrained model, to predict whether tokens in a denoiser-produced $\hat{x}_0$ are incorrect. These works are related to our policy-only setting in that they train an additional scoring layer on top of a pretrained denoiser. However, while their framework is concerned with using scores to remask tokens, our method more directly aims to learn an unmasking order. Our work differs in the objective and training setup: we directly optimize a policy over unmasking decisions with a cross-entropy-based objective weighted by denoiser losses, rather than training an error classifier, and we additionally study a joint training objective that couples the policy and denoiser, which neither of these approaches explore.

5 Conclusion

This work studies a method for training a policy over token orderings for masked diffusion models, by learning to sample positions with lower cross-entropy loss. We demonstrate that our approach outperforms common heuristics on logical tasks that are sensitive to the unmasking order, and that the same adaptation transfers to protein generation with DPLM as well as code, language, and reasoning benchmarks with Open-dLLM and LLaDA. In the policy-only setting, these gains require only a lightweight auxiliary network, while joint training further improves performance through a policy-aware denoising objective. Our results suggest that the unmasking order in masked diffusion models should be treated as a learnable component rather than a fixed heuristic. In addition to improved task performance, the proposed method is observed to yield benefits such as improving efficiency under limited diffusion-step budgets.

Future work could explore possible training-free, inference-time extensions (alleviating the need for a dataset) as well as alternate policy parameterizations and training schemes to close the gap between current and oracle performance.

AI USE STATEMENT

In this work, we used generative AI tools for editing the writing (eg. for polishing grammar, word-choice), as well as assistance in generating and editing code implementations and plotting scripts. The authors reviewed all AI-assisted text, inspected and ran the generated code, and checked its outputs against the intended behavior.

We have not used generative AI tools for helping develop theoretical models or conceptual frameworks, formulating or proving mathematical claims, refining hypotheses, providing feedback on research methodology or to interpret experimental results, and others research subtasks such as generating synthetic data, are not applicable to this work.

We take responsibility for the final content of this work, including all text, claims, code, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

The work proposes a method for improving masked diffusion models by learning the unmasking order. This is a general algorithm which has potential uses in biological applications (e.g. generating protein or DNA sequences) as well as potential harmful consequences such as for instance, in generating code with exploitable weaknesses. However, we discourage such applications.

REPRODUCIBILITY STATEMENT

We include experimental details in the main paper and in Appendix A (including compute resources, metrics, training schemes etc.). All data and models trained on or fine-tuned are publicly available.

REFERENCES

Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J Ballard, Joshua Bambrick, et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, 630(8016):493–500, 2024.

Sarah Alamdari, Nitya Thakkar, Rianne van den Berg, Neil Tenenholtz, Bob Strome, Alan Moses, Alex Xijie Lu, Nicolo Fusi, Ava Pardis Amini, and Kevin K Yang. Protein generation with evolutionary diffusion: sequence is all you need. *BioRxiv*, pp. 2023–09, 2023.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021a. URL https://openreview.net/forum?id=h7-XixPCAL.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021b.

Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas Nolte, and Brian Karrer. Accelerated sampling from masked diffusion models via entropy bounded unmasking. *arXiv preprint arXiv:2505.24857*, 2025. URL https://arxiv.org/abs/2505.24857.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurélie Névéol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. Findings of the 2016 conference on machine translation. In Ondřej Bojar, Christian Buck, Rajen Chatterjee, Christian Federmann, Liane Guillou, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Aurélie Névéol, Mariana Neves, Pavel Pecina, Martin Popel, Philipp Koehn, Christof Monz, Matteo Negri, Matt Post, Lucia Specia, Karin Verspoor, Jörg Tiedemann, and Marco Turchi (eds.), *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pp. 131–198, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/W16-2301. URL https://aclanthology.org/W16-2301/.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv 2107.03374*, 2021.

Ricky T. Q. Chen and Yaron Lipman. Flow matching on general geometries. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=g7ohDlTITL.

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 (NeurIPS)*, 37:133345–133385, 2024.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*, 2020.

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. URL https://arxiv.org/abs/2510.05725.

Chin-Wei Huang, Milad Aghajohari, Joey Bose, Prakash Panangaden, and Aaron Courville. Riemannian diffusion models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=ecevn9kPm4.

Zemin Huang, Yuhang Wang, Zhiyang Chen, and Guo-Jun Qi. Don't settle too early: Self-reflective remasking for diffusion language models. *arXiv preprint arXiv:2509.23653*, 2025. URL https://arxiv.org/abs/2509.23653.

Metod Jazbec, Theo X. Olausson, Louis Béthune, Pierre Ablin, Michael Kirchhof, João Monteiro, Victor Turrisi, Jason Ramapuram, and Marco Cuturi. Learning unmasking policies for diffusion language models. *arXiv preprint arXiv:2512.09106*, 2025. URL https://arxiv.org/abs/2512.09106.

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*, 2025. URL https://arxiv.org/abs/2502.06768.

Seul Lee, Karsten Kreis, Srimukh Prasad Veccham, Meng Liu, Danny Reidenbach, Yuxing Peng, Saee Paliwal, Weili Nie, and Arash Vahdat. Genmol: A drug discovery generalist with discrete diffusion. *arXiv preprint arXiv:2501.06158*, 2025.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013/.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. Evolutionary-scale prediction of atomic level protein structure with a language model. *bioRxiv*, 2022. doi: 10.1101/2022.07.20.500902. URL https://www.biorxiv.org/content/early/2022/10/31/2022.07.20.500902.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation, 2023. URL https://arxiv.org/abs/2305.01210.

Viacheslav Meshchaninov, Egor Shibaev, Artem Makoian, Ivan Klimov, Danil Sheshenya, Andrei Malinin, Nikita Balagansky, Daniil Gavrilov, Aibek Alanov, and Dmitry Vetrov. Guided star-shaped masked diffusion. *arXiv preprint arXiv:2510.08369*, 2025. URL https://arxiv.org/abs/2510.08369.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization, 2018. URL https://arxiv.org/abs/1808.08745.

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. URL https://arxiv.org/abs/2502.09992.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Pierre Isabelle, Eugene Charniak, and Dekang Lin (eds.), *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040/.

Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. Openwebmath: An open dataset of high-quality mathematical web text, 2023. URL https://arxiv.org/abs/2310.06786.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale, 2024. URL https://arxiv.org/abs/2406.17557.

Fred Zhangzhi Peng, Zachary Bezemek, Jarrid Rector-Brooks, Shuibai Zhang, Anru R. Zhang, Michael Bronstein, Avishek Joey Bose, and Alexander Tong. Planner aware path learning in diffusion language models training, 2025a. URL https://arxiv.org/abs/2509.23405.

Fred Zhangzhi Peng, Shuibai Zhang, Alex Tong, and contributors. Open-dllm: Open diffusion large language models. https://github.com/pengzhangzhi/Open-dLLM, 2025b.

Siva Reddy, Danqi Chen, and Christopher D. Manning. Coqa: A conversational question answering challenge, 2019. URL https://arxiv.org/abs/1808.07042.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:36479–36494, 2022.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *arXiv preprint arXiv:2406.07524*, 2024.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K Titsias. Simplified and generalized masked diffusion for discrete data. *arXiv preprint arXiv:2406.04329*, 2024.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them, 2022. URL https://arxiv.org/abs/2210.09261.

Gilad Turok, Chris De Sa, and Volodymyr Kuleshov. Duel: Exact likelihood for masked diffusion via deterministic unmasking, 2026. URL https://arxiv.org/abs/2603.01367.

Xinyou Wang, Zaixiang Zheng, Fei Ye, Dongyu Xue, Shujian Huang, and Quanquan Gu. Diffusion language models are versatile protein learners. *arXiv preprint arXiv:2402.18567*, 2024a.

Xinyou Wang, Zaixiang Zheng, Fei Ye, Dongyu Xue, Shujian Huang, and Quanquan Gu. Diffusion language models are versatile protein learners. *International Conference on Machine Learning (ICML)*, 2024b.

Zhe Wang, Jiaxin Shi, Nicolas Heess, Arthur Gretton, and Michalis K. Titsias. Learning-order autoregressive models with application to molecular graph generation. *arXiv preprint arXiv:2503.05979*, 2025. URL https://arxiv.org/abs/2503.05979.

Joseph L Watson, David Juergens, Nathaniel R Bennett, Brian L Trippe, Jason Yim, Helen E Eisenach, Woody Ahern, Andrew J Borst, Robert J Ragotte, Lukas F Milles, et al. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, 2023.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

Jiacheng Ye, Jiahui Gao, Shansan Gong, Lin Zheng, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Beyond autoregression: Discrete diffusion for complex reasoning and planning. *arXiv preprint arXiv:2410.14157*, 2024.

Yang Zhang and Jeffrey Skolnick. Scoring function for automated assessment of protein structure template quality. *Proteins: Structure*, 57, 2004.

A Experimental Details

A.1 Combinatorial Tasks

Tasks and datasets. We consider two tasks. **Sudoku**: 9×9 puzzles where the model fills in blank cells given a partially completed grid, represented as a flat sequence of 81 tokens (digits 1–9). **3-SAT**: Boolean satisfiability instances with 9 variables and 3 literals per clause, where the model must find a satisfying assignment given the clause structure. For Sudoku, we follow the dataset and evaluation protocol of Kim et al. (2025); for 3-SAT, we use the dataset of Ye et al. (2024). We report instance-level accuracy: an instance is correct only if the entire solution is valid.

Base model and training. We use a 6M-parameter GPT-2 architecture (3 layers, 384 hidden dimensions, 12 attention heads, vocabulary size 31) as the denoiser μ^θ, trained as a masked diffusion model (MDM) with T=20 diffusion steps. Training uses focal-loss-style token reweighting (α=0.25, γ=1) and linear time reweighting, with a learning rate of 10^{-3}, batch size of 1024, cosine learning rate schedule, and mixed-precision (fp16) on a single A100 GPU. We study two training regimes on top of this base model. In the *policy-only* regime, the denoiser is frozen and only the lightweight policy head is optimized using Equation (3); for Sudoku, the policy is trained for 24K steps after 115K denoiser pretraining steps, and for 3-SAT, the policy is trained for 7.5K steps after 58.5K denoiser pretraining steps. In the *joint-training* regime, we optimize the denoiser and policy together with the policy-aware weighted objective from Equation (4).

Deterministic and stochastic decoding. Table 6 includes the accuracies for both deterministic and stochastic decoding (which were used to compute differences for Figure 3). Without adding noise the heuristics perform similarly and learned policy achieves best results on both decoding settings.

Table 6: Deterministic and stochastic decoding accuracy (%) on combinatorial tasks.

	Sudoku		3-SAT	
Ordering	**Det.**	**Stoch.**	**Det.**	**Stoch.**
Policy-only				
High conf.	92.63	18.71 ± 0.16	88.6	86.87 ± 0.76
Margin	92.18	91.53 ± 0.04	88.6	88.33 ± 0.15
Policy	**93.79**	**93.75 ± 0.20**	**89.0**	**89.00 ± 0.00**
Joint training				
Base model	92.63	18.71 ± 0.16	88.6	86.87 ± 0.76
High conf.	92.73	19.44 ± 0.03	88.7	88.80 ± 0.26
Margin	90.96	90.23 ± 0.03	87.1	88.67 ± 0.15
Policy	**93.84**	**93.52 ± 0.11**	**89.4**	**90.57 ± 0.06**

Exact likelihood. For the deterministic combinatorial samplers, we compute held-out negative log-likelihood using DUEL (Turok et al., 2026), as an additional evaluation metric. As shown in Table 7, the learned policy gives the lowest NLL in both training regimes and on both tasks, demonstrating that the policy training better captures the dataset distribution (in addition to having better accuracy). We do not apply DUEL to DPLM because its token sampling is stochastic and its ordering scores depend on the sampled tokens, whereas DUEL requires deterministic position selection.

Table 7: Held-out NLL under deterministic decoding. 3-SAT uses 10 denoising steps; Sudoku uses 20. Lower is better.

Ordering	**3-SAT NLL**	**Sudoku NLL**
Policy-only		
Entropy	0.028561	0.003708
High conf.	0.028587	0.003711
Margin	0.028613	0.004185
Random	0.030919	0.063625
Policy	**0.027888**	**0.003323**
Joint training		
High conf.	0.018856	0.003649
Margin	0.025383	0.004445
Policy	**0.018210**	**0.003056**

A.2 PROTEIN / DPLM EXPERIMENTS

A.2.1 DPLM SETUP

Evaluation setup. For protein experiments, we adapt the same policy parameterization to DPLM-150M (Wang et al., 2024b). The policy again takes per-position confidence scores together with final-layer hidden states and predicts routing logits over masked positions. Our DPLM setup supports both policy-only adaptation, where the pretrained backbone is frozen and only the lightweight policy head is trained, and joint training of the backbone and policy through the policy-aware weighted objective from Equation (4). In the joint DPLM, we initialize from the released DPLM-150M checkpoint of Wang et al. (2024b)[1], unfreeze the backbone, and fine-tune with the adaptive DPLM training configuration using a reduced learning rate (4×10^{-5}) on A100 and H100 GPUs. For the DPLM results summarized in Table 1 and visualized in Figure 1, we generate proteins at sequence lengths $\{100, 200, 300, 400, 500\}$; pLDDT is averaged across three random seeds after averaging within seed over sequence lengths, and we additionally report inner-TM diversity in the corresponding diversity plots. For the protein sequence generation comparison in Table 2, we follow Peng et al. (2025a) and generate 100 sequences at lengths 200, 300, ..., 800 and report pLDDT, pTM, pAE, foldability, token entropy, and sequence diversity. The heuristic baselines are high-confidence decoding and margin decoding, matching the orderings used in the combinatorial experiments.

A.2.2 PROTEIN SEQUENCE GENERATION METRICS

For protein evaluation, we fold each generated sequence with ESMFold (Lin et al., 2022) and use the predicted structures to assess quality, foldability, and diversity. For the joint comparison in Table 2, we report three structural confidence metrics, a composite foldability score, and two diversity statistics. The per-sequence structural metrics are:

$$\mathrm{pLDDT}(i) = 100 \times \mathbb{E}_{j \in \mathcal{N}(i)} \left[1 - \frac{|d_{ij}^{\mathrm{pred}} - d_{ij}^{\mathrm{true}}|}{d_{ij}^{\mathrm{true}}} \right], \tag{5}$$

$$\mathrm{pTM} = \max_{u} \frac{1}{L} \sum_{i=1}^{L} \frac{1}{1 + \left(d_{i,u(i)} / d_0(L) \right)^2}, \tag{6}$$

$$\mathrm{pAE}(i,j) = \mathbb{E}\left[\left\| x_i^{\mathrm{pred}} - x_j^{\mathrm{true}} \right\|_2 \right]. \tag{7}$$

pLDDT measures local per-residue confidence, pTM measures global structural similarity and is adapted from the TM-score (Zhang & Skolnick, 2004), and pAE measures expected alignment error across residue pairs. We report pLDDT averaged over residues and pAE averaged over residue pairs; higher pLDDT and pTM are better, while lower pAE is better. We also report a binary *foldability* rate, defined as the fraction of generated sequences satisfying $\mathrm{pLDDT} > 80$, $\mathrm{pTM} > 0.7$, and $\mathrm{pAE} < 10$ (as in Peng et al. (2025a)).

To assess diversity and possible mode collapse, we report token entropy, sequence diversity, and inner-TM. Token entropy and sequence diversity are defined as:

$$H = -\sum_{a \in \mathcal{A}} p(a) \log p(a), \tag{8}$$

$$\mathrm{Id}\left(x^{(m)}, x^{(n)} \right) = \frac{1}{L} \sum_{i=1}^{L} \mathbf{1}\left[x_i^{(m)} = x_i^{(n)} \right], \tag{9}$$

$$\mathrm{Diversity} = 1 - \frac{2}{B(B-1)} \sum_{1 \le m < n \le B} \mathrm{Id}\left(x^{(m)}, x^{(n)} \right). \tag{10}$$

Here, $\mathcal{A}$ is the set of amino acids observed in the generated set, $p(a)$ is the empirical frequency of amino acid a, and B is the number of generated sequences being compared. Higher entropy and diversity indicate richer amino-acid usage and lower sequence-level collapse. We additionally report inner-TM, defined as the average pairwise TM-score between predicted structures in the generated set; lower inner-TM indicates that the sampled proteins are more structurally diverse.

[1] https://huggingface.co/airkingbd/dplm_150m

A.3 Language and Code Generation

Open-dLLM. We use the Open-dLLM-0.5B code model (Peng et al., 2025b) and train only the policy. Policy training uses FineCode (Peng et al., 2025b), the code corpus used by Open-dLLM. Each document is tokenized with the Open-dCoder-0.5B-no-PL tokenizer, terminated with an end-of-sequence token, and divided into chunks of at most 4,096 tokens without combining different documents. We evaluate pass@1 on HumanEval (Chen et al., 2021), HumanEval+ (Liu et al., 2023), MBPP (Austin et al., 2021b), and MBPP+ (Liu et al., 2023).

LLaDA. We train the policy for 2,000 optimizer steps on a 100M-token mixture comprising 60% FineWeb-Edu (Penedo et al., 2024), 20% OpenWebMath (Paster et al., 2023), and 20% FineCode (Peng et al., 2025b), with a separate 5M-token validation split. Because LLaDA does not release its pretraining data, this mixture is used only for policy training. We evaluate LLaDA-8B (Nie et al., 2025) on HumanEval (Chen et al., 2021), BBH (Suzgun et al., 2022), CoQA (Reddy et al., 2019), WMT16 English–Romanian (Bojar et al., 2016), MBPP (Austin et al., 2021b), and XSum (Narayan et al., 2018). Generation length, denoising steps, and block length are respectively 512/512/512 for HumanEval; 256/256/256 for BBH and MBPP; and 64/64/64 for CoQA, WMT16, and XSum. In the reduced-step evaluation, the denoising budgets are divided by four while generation and block lengths are fixed at 512/128/512, 256/64/256, and 64/16/64, respectively. High-confidence ordering and the learned policy use identical generation settings. We evaluate using pass@1 task completion for HumanEval and MBPP, exact answer match for BBH, F1-score for CoQA, BLEU (Papineni et al., 2002) for WMT16, and ROUGE-L (Lin, 2004) for XSum.

For both open-dLLM and LLaDA tasks, we report the best result obtained from a hyperparameter sweep over inference settings (eg. token sampling temperature, position-sampling temperature, heuristic).

B Additional Policy Architecture Results

B.1 Transformer Policy Variant

In the policy-only setting, we also evaluate a transformer-based alternative to the per-token MLP. The score-only variant projects each confidence score to dimension 128 before a single transformer encoder layer. The score-and-hidden variant additionally incorporates the denoiser hidden states, allowing the policy to exchange information across positions before selecting an ordering.

Table 8: Policy architecture comparison on Sudoku with deterministic decoding and T=20 steps. Higher is better; best in **bold**.

Policy architecture	Accuracy
Per-token MLP (scores + hidden)	**90.82%**
Score Transformer (scores only)	89.84%
Score + Hidden Transformer (scores + hidden)	90.23%

The per-token MLP remains the strongest architecture in Table 8. Adding hidden states improves the transformer variant, but cross-position attention provides no further gain over the simpler MLP on Sudoku. This suggests that the denoiser hidden states already contain sufficient contextual information for the policy to make effective token-level decisions.

C Relation of Objective to ELBO Bounds

We can derive our objective Equation (3) as an approximation to an ELBO bound.

In particular, we make use of Proposition 3.2 from Peng et al. (2025a). This gives an ELBO-style objective for masked diffusion decoding under an unmasking policy q^ϕ. Rewritten in the notation of this paper, the proposition can be stated as follows.

Proposition 1 (Proposition 3.2 of (Peng et al., 2025a))**.** *Let $q^\phi(i \mid x_0, x_t)$ be a policy over masked positions at time t, and let $p_{\theta,\phi}(x_0)$ denote the marginal probability of x_0 under the unmasking process obtained by decoding according to this policy. For each time t, let $r_t^\phi(\cdot \mid x_0)$ denote the*

distribution of the partially masked sequence x_t under the policy-guided unmasking process. Then

$$\log p_{\theta,\phi}(x_0) \geq \mathcal{E}_1^{\theta,\phi}(x_0) + \mathcal{E}_2^{\theta,\phi}(x_0), \tag{11}$$

where

$$\mathcal{E}_1^{\theta,\phi}(x_0) = \mathbb{E}_{t\sim\mathcal{U}[0,1]}\left[c(\alpha_t)\mathbb{E}_{x_t\sim r_t^\phi(\cdot|x_0)}\left[\sum_{i=1}^{L} q^\phi(i \mid x_0, x_t)\,\mathrm{CE}(x_0^i, \mu^\theta(x_t)[i,\cdot])\right]\right], \tag{12}$$

and

$$\mathcal{E}_2^{\theta,\phi}(x_0) = -\mathbb{E}_{t\sim\mathcal{U}[0,1]}\left[c(\alpha_t)\mathbb{E}_{x_t\sim r_t^\phi(\cdot|x_0)}\left[\sum_{i=1}^{L} q^\phi(i \mid x_0, x_t)\,\log\frac{q^\phi(i \mid x_0, x_t)}{F_{\theta,\phi}(x_t, x_0^i, i)}\right]\right]. \tag{13}$$

Here $c(\alpha_t)$ is some constant depending on the schedule, and $F_{\theta,\phi}(x_t, x_0^i, i)$ is an auxiliary distribution:

$$F_{\theta,\phi}(x_t, x_0^i, i) = \mathbb{E}_{z\sim\mu^\theta(x_t)}\left[q^\phi(i \mid z^{-i,x_0^i}, x_t)\right], \tag{14}$$

z^{-i,x_0^i} is z with token i set to value x_0^i.

To simplify for our setting, since x_0 is sampled from the denoiser, we can consider the policy $q^\phi(i \mid x_0 \sim \mu^\theta(x_t), x_t)$ as being conditioned on only x_t, so that $q^\phi(i \mid x_t, x_0) = q^\phi(i \mid x_t)$. With this notation, we note that the first term $\mathcal{E}_1$ is very similar to the negative of the policy-weighted loss in Equation (3) evaluated along policy-induced paths $r_t^\phi(\cdot \mid x_0)$ rather than the standard noising process $p(x_t \mid x_0)$.

The quantity $F_{\theta,\phi}(x_t, x_0^i, i)$ in the second term evaluates the policy probability of unmasking position i to have token x_0^i (marginalizing over possible denoiser randomness). Explicitly evaluating the $\mathcal{E}_2$ term requires multiple passes of the policy module.

From the above ELBO, if we make the following approximations, then we can recover our objective in Equation (3):

- sample x_t from the noising distribution $p(x_t \mid x_0)$ rather than the policy-induced distribution $r_t^\phi(\cdot \mid x_0)$, since the former is tractable as simple. Peng et al. (2025a) also make this simplification.
- we omit the second term $\mathcal{E}_2^{\theta,\phi}(x_0)$, which involves a term $F_{\theta,\phi}(x_t, x_0^i, i)$ that is difficult to compute. This term may be omitted when fixing the policy (such as when it is obtained from a heuristic). By contrast, we learn our policy, but argue that omitting $\mathcal{E}_2$ still yields a reasonable objective for training the policy, which we reinforce empirically.

Given these approximations, we obtain our policy weighted objective as a surrogate of the policy-induced ELBO. We use this objective for both policy-only training (in Equation (3)), as well as joint training of the policy and denoiser (in Equation (4)).