Coupling Models for One-Step Discrete Generation

Fred Zhangzhi Peng*
Duke University

Avishek Joey Bose
Imperial College London

Anru R. Zhang
Duke University

Alexander Tong
AITHYRA

Abstract

Generative modeling over discrete structures underpins applications across deep learning, from biological sequence design and code generation to large language models, yet generation often remains sequential, relying on autoregressive decoding or iterative refinement. In this work, we introduce *Coupling Models* (Coupling Models), a one-step discrete generative model that learns a direct coupling between discrete sequences and Gaussian latents. Unlike recent distillation methods that compress a pretrained multi-step sampler into a few steps, *Coupling Model* trains a purpose-built decoder to invert this coupling and generate samples in a single step. The model also avoids complex continuous flows over the simplex and hand-specified data-to-noise couplings. Empirically, *Coupling Model* improves the strongest one-step baselines in each domain: it reduces LM1B text-generation perplexity by 33% at its lowest-perplexity operating point, Fly Brain enhancer-design FBD by 18%, and MNIST-Binary FID by 46%. These results suggest that effective one-step discrete generation depends strongly on how data and noise are coupled before decoding. Code is available at https://github.com/pengzhangzhi/Coupling-Models.

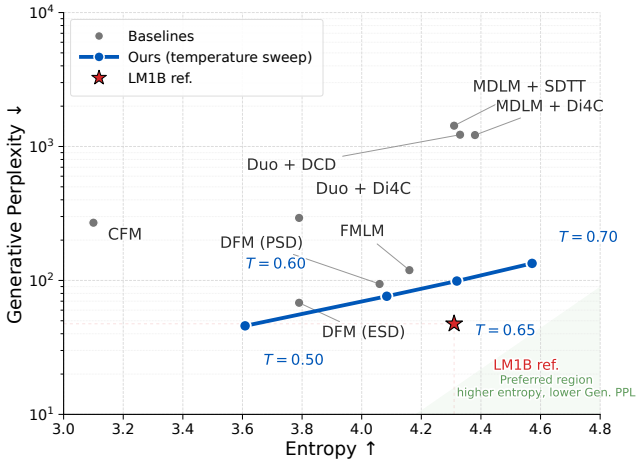

Figure 1: **Toward a one-step language generation quality–diversity frontier.** On LM1B, *Coupling Model* shifts the one-step entropy–perplexity frontier toward the desired low-perplexity, high-entropy region: it improves over prior one-step baselines in generative perplexity while preserving entropy close to the real-data reference.

*Correspondence to: zhangzhi.peng@duke.edu

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

Discrete generation is central to language modeling, code generation, biological sequence design, and discrete image modeling [Brown et al., 2020a, Dauparas et al., 2022, Peng et al., 2025a, Stärk et al., 2024, Davis et al., 2024]. The strongest models in these domains still rely on inference procedures that use multiple model evaluations to represent complex joint structure. For instance, autoregressive models factorize the sequence distribution into next-token conditionals and remain the dominant paradigm for modern large language models [Vaswani et al., 2017, Brown et al., 2020b, Touvron et al., 2023, Yang et al., 2024]. Although this factorization is expressive and scalable, generation is inherently serial. Discrete diffusion, masked refinement, and discrete flow models offer a more parallel alternative by updating many positions simultaneously, but they also require repeated steps to achieve higher levels of coherence [Austin et al., 2021, Sahoo et al., 2024, Lou et al., 2024, Zheng et al., 2024, Nie et al., 2025, Gong et al., 2025, Schiff et al., 2025a, Gat et al., 2024, Cheng et al., 2024, Stärk et al., 2024, Davis et al., 2024].

The search for faster generation motivates the more extreme setting of *true one-step* discrete generation: a single prior sample coupled with a single model forward pass to produce the full sequence. A successful one-step generator would remove the sequential bottleneck of autoregressive decoding and the repeated refinement cost of discrete diffusion-style models. Toward this goal, recent work has made the regime more feasible through consistency, distillation, self-distillation, or rectification [Sahoo et al., 2025, Yoo et al., 2025, Deschenaux and Gulcehre, 2025, Chen et al., 2025, Zhu et al., 2025, Monsefi et al., 2025, Zheng et al., 2026, Zhang et al., 2026, Bartosh et al., 2026]. Another line improves the low-step trajectory itself through better source–target couplings and discrete flow constructions [Gat et al., 2024, Park et al., 2026]. A third line moves generation into continuous or simplex-valued surrogate spaces, where categorical variables can be modeled through smoother geometric structure [Cheng et al., 2024, Stärk et al., 2024, Davis et al., 2024].

A particularly active recent direction is based on *flow maps*. Rather than simulating a full flow or denoising trajectory, flow-map methods learn endpoint maps that directly predict terminal states. Recent discrete-generation work adapts this idea to categorical data and language. For instance, Discrete Flow Maps [Potaptchik et al., 2026] formulates discrete flow maps by aligning endpoint-map training with the geometry of the probability simplex. FMLM [Lee et al., 2026] learns continuous denoising flows over one-hot token embeddings and distills them into flow maps for efficient language generation. CFM [Roos et al., 2026] defines categorical flow maps toward the simplex and trains them with self-distillation and endpoint-consistency objectives. Although these methods differ in parameterization, objectives, and empirical domains, they share a common premise: fast discrete generation can be obtained by learning better endpoint maps for continuous or simplex-valued generative trajectories. This progress raises a basic question:

Q. *Is trajectory compression the most natural route to one-step discrete generation?*

We argue that the difficulty of one-step discrete generation is not only an optimization problem, but also a representation problem. A direct one-step categorical decoder must recover global cross-token dependence in a single parallel prediction, without the benefit of autoregressive factorization or iterative denoising. As the number of refinement steps drops, the model has fewer opportunities to repair local decisions using global context, and quality can degrade even when the underlying multi-step model is capable [Sahoo et al., 2025]. The central issue is therefore not only how to shorten an existing trajectory, but also what representation makes the one-step map easier to learn in the first place.

Present work. We propose Coupling Models (Coupling Models), a coupling-based construction for one-step discrete generation. Instead of distilling an iterative generator or learning an endpoint map over a discrete or simplex-valued trajectory, we first learn an easy-to-sample coupling between discrete sequences and Gaussian latents. This coupling induces paired supervision between Gaussian latents and target sequences. We then train a parallel decoder to invert the coupling and emit all output tokens in one forward pass. At inference time, generation requires one step: draw $z \sim \mathcal{N}(0, I)$ and generate.

We evaluate *Coupling Model* across three regimes: (1) controlled binary image modeling on MNIST-Binary, (2) biological sequence design on DNA enhancers, and (3) open-ended language generation on LM1B. In the strict one-step setting, *Coupling Model* improves the best available one-step comparisons across all three domains. On MNIST-Binary, it reduces FID to 5.50, outperforming the strongest one-step baselines in Table 1. On DNA enhancer generation, it improves over the best one-step baseline, distilled Dirichlet FM [Stärk et al., 2024], reducing FB FBD from 15.8 to 12.9 ± 0.3 in Table 2.

On LM1B, it improves over the flow-map-based methods CFM [Roos et al., 2026], Discrete Flow Maps [Potaptchik et al., 2026], and FMLM [Lee et al., 2026] in generative perplexity while maintaining entropy near the real-data reference. The central language-generation result is that *Coupling Model* does not obtain lower perplexity by collapsing diversity; it shifts the one-step entropy–perplexity frontier toward the desired low-perplexity, high-entropy region, as depicted in Figure 1. Finally, we demonstrate that beyond unconditional sampling, the same latent interface makes guidance more direct and efficient. More precisely, classifier-free guidance acts on the final logits, classifier guidance optimizes the latent variable, and reward fine-tuning updates the generator, avoiding the long stochastic control path required by multi-step denoising. Empirically, this shorter control path improves the quality–efficiency tradeoff relative to the masked-diffusion baseline while using far fewer function evaluations.

2 Related Work

Autoregressive, diffusion, and flow-based discrete generation. Discrete generation is commonly approached through inference procedures that use multiple model evaluations to represent complex joint structure. Autoregressive models factorize the sequence distribution into conditional next-token predictions and remain the dominant paradigm for language and sequence modeling [Vaswani et al., 2017, Brown et al., 2020b, Touvron et al., 2023, Yang et al., 2024]. This factorization is expressive and scalable, but generation is inherently serial. Discrete diffusion and masked-refinement models update many positions in parallel, but recover global dependence through repeated denoising or replacement steps [Austin et al., 2021, Sahoo et al., 2024, Lou et al., 2024, Zheng et al., 2024, Nie et al., 2025, Gong et al., 2025, Schiff et al., 2025a]. Related discrete flow and simplex-based methods formulate generation through transport or flow matching over categorical variables or continuous relaxations of the simplex [Gat et al., 2024, Cheng et al., 2024, Stärk et al., 2024, Davis et al., 2024]. Together, these methods define the main speed-quality tradeoff studied in this paper: strong discrete generation typically relies on either serial factorization or iterative refinement. A more detailed discussion appears in Appendix A.

Fast one-step and few-step generation. Recent work has made one-step and few-step discrete generation increasingly practical. Distillation, consistency, self-distillation, and rectification methods compress or improve multi-step generators so that fewer model evaluations are required at test time [Sahoo et al., 2025, Yoo et al., 2025, Deschenaux and Gulcehre, 2025, Chen et al., 2025, Zhu et al., 2025, Monsefi et al., 2025, Zheng et al., 2026, Zhang et al., 2026, Bartosh et al., 2026]. Coupling-based methods such as PAIRFLOW [Park et al., 2026] improve the low-step coupling itself by constructing better source-target couplings [Park et al., 2026]. A particularly relevant line is flow-map-based generation: Discrete Flow Maps [Potaptchik et al., 2026], FMLM [Lee et al., 2026], and CFM [Roos et al., 2026] learn endpoint maps for continuous or simplex-valued trajectories, enabling faster generation by bypassing long integration or denoising chains [Potaptchik et al., 2026, Lee et al., 2026, Roos et al., 2026]. Our method targets the same one-step regime but differs in construction. Rather than distilling an iterative generator, rectifying a discrete trajectory, or learning an endpoint map along a simplex-valued path, we first learn a coupling between continuous sequence representations and Gaussian latents. The resulting latent-sequence pairs provide supervised training data for a parallel one-step decoder. BiFlow and NFM are related in their use of normalizing flows to connect data and Gaussian noise bidirectionally, but both operate in a continuous setting, whereas we use the learned coupling to induce latent–sequence supervision for one-step discrete decoding [Lu et al., 2026, Berthelot et al., 2026]. Our method is best viewed as a coupling-based route to one-step discrete generation, rather than another trajectory-compression method.

3 Method

We study one-step generation of a discrete sequence $x = (x_1, \dots, x_T)$, where $x_t \in \{1, \dots, V\}$. The main challenge is not only computational, but representational. A direct one-step categorical decoder predicts one distribution per position and samples all positions in parallel. Without an additional shared variable or refinement process, this induces a factorized sequence law, $p(x) = \prod_{t=1}^{T} p_t(x_t)$, which cannot represent general cross-token dependence. Autoregressive models avoid this barrier through sequential conditioning, while diffusion and masked-refinement models use multiple denoising steps to exchange information across positions. The strict one-step setting removes both

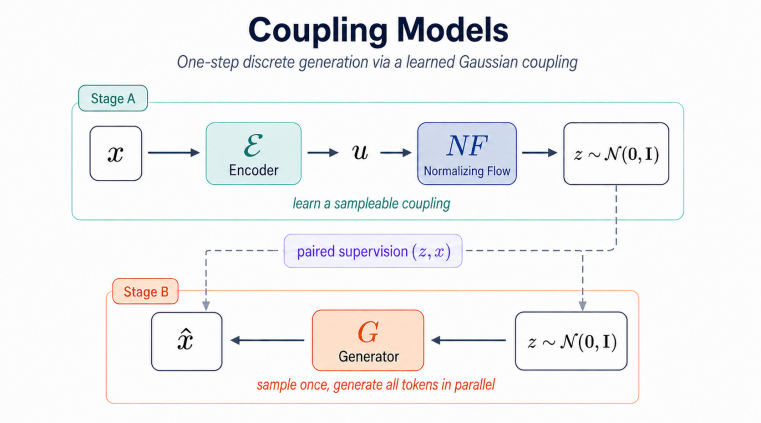

Figure 2: **Overview of the Coupling Model.** Stage A learns a coupling from discrete sequences to Gaussian latents. Stage B trains a parallel decoder on the induced pairs, enabling generation from a single model evaluation.

mechanisms. Thus, the central question is what representation can carry global sequence-level information into a single parallel categorical emission.

3.1 Coupling Model for one-step generation

Coupling Model addresses this question by learning a continuous coupling between discrete sequences and Gaussian latents. The final decoder still emits token distributions in parallel, but it conditions all positions on a shared latent z. Hence, although $G_\theta(x \mid z) = \prod_t G_\theta(x_t \mid z)$ factorizes conditioned on z, the marginal $p_\theta^{\text{gen}}(x) = \int G_\theta(x \mid z) p_Z(z)\, dz$ is a latent mixture that can carry global dependence.

The method has two stages. Stage A maps each training sequence to a continuous representation and uses a likelihood-based normalizing flow to align the induced latent marginal with $p_Z = \mathcal{N}(0, I)$. This yields paired supervision (z, x), where z is informative about x and sampleable from the inference prior. Stage B trains a parallel decoder to invert these pairs, so inference reduces to drawing $z \sim \mathcal{N}(0, I)$ and decoding all tokens in one forward pass. Unlike flow-map compression, *Coupling Model* does not learn an endpoint map along a discrete or simplex-valued trajectory; it first constructs a sampleable Gaussian coupling and then learns the one-step inverse map. Appendix C formalizes the expressivity barrier for direct token-wise one-step decoders.

Stage A: Discrete sequence → Gaussian. Stage A constructs the latent–sequence coupling used by the one-step decoder. Since the sequence x is discrete, we do not fit a continuous density model directly on tokens. Instead, we first encode x into a continuous representation u, and then transform this representation into a Gaussian latent z using a normalizing flow. This separates two roles: the encoder space should preserve the information needed to reconstruct the sequence, while the flow space should match the simple prior used for sampling.

Concretely, we parameterize a reparameterized encoder $q_\psi(u \mid x)$ by $u = E_\psi(x, \epsilon), \epsilon \sim \mathcal{N}(0, I)$, and map the resulting representation to the sampling latent through an invertible normalizing flow, $z = \mathrm{NF}_\phi(u), p_Z = \mathcal{N}(0, I)$. A lightweight decoder D reconstructs the original sequence from u. The Stage A objective is

$$\begin{aligned}\mathcal{L}_A = \lambda_{\text{rec}} \underbrace{\left[-\sum_{t=1}^{T} \log p_D(x_t \mid u) \right]}_{\mathcal{L}_{\text{rec}}} + \lambda_{\text{KL}} \underbrace{\mathrm{KL}(q_\psi(u \mid x) \,\|\, \mathcal{N}(0, I))}_{\mathcal{L}_{\text{KL}}} \\ + \lambda_{\text{flow}} \underbrace{\left[-\log p_Z(\mathrm{NF}_\phi(u)) - \log \left| \det \frac{\partial \mathrm{NF}_\phi(u)}{\partial u} \right| \right]}_{\mathcal{L}_{\text{flow}}}.\end{aligned} \tag{1}$$

Each term enforces a distinct property of the coupling. The reconstruction loss prevents the continuous representation from discarding sequence-level information. The KL term regularizes the encoder

distribution and stabilizes the continuous representation space. The flow likelihood then aligns the induced latent marginal with the Gaussian prior used at inference. Thus, Stage A is not used as an exact likelihood model for the discrete sequence itself. Its purpose is to produce paired supervision

$$(z, x), \qquad z = \mathrm{NF}_\phi(E_\psi(x, \epsilon)),$$

where z is both informative about x and approximately distributed as the sampling prior. These pairs are the training data for Stage B.

Stage B: Gaussian → Discrete sequence. After Stage A, we freeze E_ψ, D, and NF_ϕ and construct paired data $\mathcal{D}_Z = \{(z^{(i)}, x^{(i)})\}_{i=1}^N$ with $z^{(i)} = \mathrm{NF}_\phi(E_\psi(x^{(i)}, \epsilon^{(i)}))$. We train a parallel decoder G_θ on these pairs:

$$\begin{aligned} \ell = G_\theta(z) \in \mathbb{R}^{T\times V}, \qquad G_\theta(x \mid z) = \prod_{t=1}^{T} \mathrm{softmax}(\ell_t)[x_t], \\ \mathcal{L}_B(\theta) = -\mathbb{E}_{(z,x)\sim\mathcal{D}_Z} \log G_\theta(x \mid z). \end{aligned} \tag{2}$$

Although the categorical emission factorizes across positions given z, the generated marginal does not reduce to an unconditional product distribution: $p_\theta^{\mathrm{gen}}(x) = \int G_\theta(x \mid z) p_Z(z)\, dz$ is a latent mixture of parallel decoders. The global dependence is therefore carried by the coupling-induced latent variable, while the decoder performs the final conditional token emission. At inference time, we do not run Stage A: we draw $z \sim \mathcal{N}(0, I)$, compute $\ell = G_\theta(z)$, and sample $\hat{x}_t \sim \mathrm{Cat}(\mathrm{softmax}(\ell_t/\tau))$ independently across positions. Training proceeds in two passes. We first optimize Eq. (1) on the training set, freeze the encoder and flow, materialize $\mathcal{D}_Z$, and then optimize Eq. (2). The full training and sampling procedure is given in Appendix B.

Why latent matching matters for one-step decoding. A VAE-style encoder already produces latent–sequence pairs, so one may ask why the flow component is necessary. The issue is that Stage B is trained on the latent marginal induced by Stage A, but generation samples from the Gaussian prior. If these two latent distributions differ, the one-step decoder is used off distribution at inference. The flow therefore plays a specific role: it aligns the Stage B training latents with the sampling prior. This train–test alignment is the main reason for fitting a flow after the continuous encoder: the decoder is trained on the same latent marginal from which it will be sampled at inference.

Proposition 3.1: Latent matching controls one-step generation error

Let $q_\phi(z, x)$ be the latent–sequence joint induced by Stage A, and let $p_Z = \mathcal{N}(0, I)$ be the inference prior. For any decoder $G_\theta(x \mid z)$, the generated distribution $p_\theta^{\mathrm{gen}}(x) = \int G_\theta(x \mid z) p_Z(z)\, dz$ satisfies

$$\mathrm{TV}(p_{\mathrm{data}}, p_\theta^{\mathrm{gen}}) \le \mathbb{E}_{z\sim q_\phi(z)} \mathrm{TV}(q_\phi(\cdot \mid z), G_\theta(\cdot \mid z)) + \mathrm{TV}(q_\phi(z), p_Z(z)) \tag{3}$$

Moreover, if

$$\mathbb{E}_{z\sim q_\phi(z)} \mathrm{KL}(q_\phi(\cdot \mid z) \,\|\, G_\theta(\cdot \mid z)) \le \varepsilon_{\mathrm{dec}}, \qquad \mathrm{KL}(q_\phi(z) \,\|\, p_Z(z)) \le \varepsilon_{\mathrm{flow}},$$

then

$$\mathrm{TV}(p_{\mathrm{data}}, p_\theta^{\mathrm{gen}}) \le \sqrt{\frac{\varepsilon_{\mathrm{dec}}}{2}} + \sqrt{\frac{\varepsilon_{\mathrm{flow}}}{2}}. \tag{4}$$

Consequently, if Stage B matches $q_\phi(x \mid z)$ and Stage A matches $q_\phi(z)$ to p_Z, then one-step sampling recovers the data distribution.

Proposition 3.1 identifies the two quantities that the two-stage procedure must control. The first term is the decoding error optimized by Eq. (2); the second is the latent mismatch optimized by the Stage A flow objective. Thus, the flow is not introduced merely as a richer VAE prior. It makes the latent–sequence coupling sampleable by aligning the latents used for supervised decoder training with the Gaussian latents used at inference. Appendix D gives the full formal statement, including the KL-based bound and proof.

3.2 Few-step extension

The coupling construction is not restricted to a direct one-step decoder. Since Stage A produces paired latent–sequence supervision $\mathcal{D}_Z = \{(z^{(i)}, x^{(i)})\}_{i=1}^N$, Stage B can be instantiated by any conditional

discrete generator. We therefore also consider a few-step variant that replaces the one-step decoder G_θ with a latent-conditioned masked denoiser H_θ. This gives a direct extension from one-step decoding to masked diffusion sampling while preserving the same learned Gaussian latent space.

During training, H_θ receives a partially masked sequence together with its Stage A latent and predicts the clean tokens at the masked positions. The latent therefore supplies global sequence-level information, while the masked state provides the local refinement context that a direct one-step decoder lacks. For each pair $(z, x) \sim \mathcal{D}_Z$, we sample a masking time $t \sim \mathrm{Unif}(0, 1)$ and construct a partially masked sequence x_t by masking each position independently with probability t. Specifically, we sample $m_i \sim \mathrm{Bernoulli}(t)$ for each position i, set $(x_t)_i = \mathsf{M}$ if $m_i = 1$ and $(x_t)_i = x_i$ otherwise, where M is the mask token. The denoiser $H_\theta(x_t, z)$ is conditioned only on the masked sequence x_t and the Gaussian latent z, and predicts logits over the vocabulary at all positions. We train it with the masked-token cross-entropy objective

$$\mathcal{L}_{\mathrm{MDM}}(\theta) = -\mathbb{E}_{(z,x)\sim\mathcal{D}_Z,\, t\sim\mathrm{Unif}(0,1),\, m}\left[\frac{1}{\sum_{i=1}^{T} m_i}\sum_{i:m_i=1}\log p_\theta(x_i \mid x_t, z)\right]. \tag{5}$$

The training procedure is summarized in Algorithm 2. At inference time, we sample one latent $z \sim \mathcal{N}(0, I)$, keep it fixed throughout the trajectory, and run P2-self masked sampling [Peng et al., 2025a]. With one reveal step, this reduces to a latent-conditioned parallel decode; with more steps, the sampler can revise low-confidence positions while conditioning on the same global latent. The full masked-token objective and sampling algorithm are given in Appendix E.

3.3 Guided one-step generation

Guidance is a central tool in modern generative modeling, as it enables controllable generation toward a target condition or property [Dhariwal and Nichol, 2021, Ho and Salimans, 2022]. In continuous models, such guidance can often be applied directly through the model output. In discrete generation, however, the situation is more difficult because sampling is non-differentiable. This difficulty becomes especially pronounced in current multi-step discrete generators, where guidance or reward signals must be propagated through an entire trajectory of stochastic transitions [Sahoo et al., 2024, Schiff et al., 2025b, Nisonoff et al., 2025, Wang et al., 2025a]. One-step discrete generators avoid this issue by acting on the final predictive distribution, rather than being propagated through multiple discrete updates. This shorter control path is empirically effective: Section 4.3 shows that one-step guidance improves the quality–efficiency tradeoff relative to the multi-step MDM baseline, and Figure 4 summarizes both the guided comparison and the CFG sweep. We use three standard mechanisms in this one-step setting.

Classifier-free guidance. Following classifier-free guidance [Ho and Salimans, 2022], we train $G_\theta(z, y)$ with conditioning dropout and combine conditional and unconditional logits as

$$\ell^{\mathrm{cfg}} = \ell^u + s(\ell^c - \ell^u), \qquad \ell^c = G_\theta(z, y), \qquad \ell^u = G_\theta(z, \varnothing), \tag{6}$$

which biases the final token distributions toward the requested condition without changing the latent sample.

Classifier guidance. Following standard classifier-guidance methods [Dhariwal and Nichol, 2021, Schiff et al., 2025b, Nisonoff et al., 2025], we keep G_θ fixed and optimize z by gradient ascent on a differentiable relaxation of the decoded sequence before the single final sample:

$$z^{k+1} = z^k + \eta\nabla_{z^k} R\big(\rho(G_\theta(z^k, y)), y\big), \tag{7}$$

which changes the latent draw rather than the decoder parameters.

Reward fine-tuning. For reward fine-tuning [Wang et al., 2025a], we update the generator with a reward term and an anchor penalty to keep it close to the pretrained model:

$$\mathcal{L}_{\mathrm{FT}}(\theta) = -\lambda_{\mathrm{reward}}\mathbb{E}_{z,y} R(\rho(G_\theta(z, y)), y) + \lambda_{\mathrm{anchor}}\mathcal{R}(\theta, \theta_0). \tag{8}$$

These mechanisms act at different places in the one-step pipeline: logits for CFG, the latent variable for classifier guidance, and model parameters for reward fine-tuning. In all cases, the relaxation is needed only at the final prediction layer, rather than through a denoising trajectory.

Table 1: **Unconditional generation on MNIST-Binary.** We report FID at a specified sampling budget (Steps).

One-step

Method	Steps	FID↓
MDLM [Sahoo et al., 2024]	1	204.64
UDLM [Schiff et al., 2025a]	1	130.57
PAIRFLOW [Park et al., 2026]	1	40.58
UDLM [Schiff et al., 2025a] + DCD [Sahoo et al., 2025]	1	53.84
PAIRFLOW [Park et al., 2026] + DCD [Sahoo et al., 2025]	1	19.51
UDLM [Schiff et al., 2025a] + ReDi [Yoo et al., 2025]	1	18.44
PAIRFLOW [Park et al., 2026] + ReDi [Yoo et al., 2025]	1	12.90
CFM [Roos et al., 2026]	1	10.10
Coupling Model (Ours)	1	**5.50**

Few-step

Method	Steps	FID↓
MDLM [Sahoo et al., 2024]	2	159.26
MDLM [Sahoo et al., 2024]	4	103.74
UDLM [Schiff et al., 2025a]	2	42.54
UDLM [Schiff et al., 2025a]	4	11.25
PAIRFLOW [Park et al., 2026]	2	15.61
PAIRFLOW [Park et al., 2026]	4	8.50
CFM [Roos et al., 2026]	2	8.70
CFM [Roos et al., 2026]	4	7.80

Multi-step

Method	Steps	FID↓
SFM w/ OT [Cheng et al., 2024]	300	4.62
DirichletFM [Stärk et al., 2024]	–	77.35
DDSM [Avdeyev et al., 2023]	100	7.79
D3PM [Austin et al., 2021]	1000	67.36
DFM [Gat et al., 2024]	–	34.42
MDLM [Sahoo et al., 2024]	64	7.01
UDLM [Schiff et al., 2025a]	64	5.01
PAIRFLOW [Park et al., 2026]	64	5.17

Differentiable guidance through final-layer relaxations. Classifier guidance and reward fine-tuning require gradients through the generated sequence, but discrete sampling is not differentiable. We use two simple surrogate paths. The first replaces the sampled sequence by a soft relaxation of the final logits, using a softmax relaxation for categorical outputs to give stable low-variance gradients. The second uses a Gumbel-Softmax or straight-through relaxation, which more closely follows the categorical sampling path at the cost of noisier gradients. In both cases, the relaxation is used only to compute guidance or fine-tuning gradients; the final sample is still drawn discretely from the guided logits. Full procedural details are given in Appendix F.1.

4 Experiments

We evaluate whether *Coupling Model* provides a practical route to one-step discrete generation across three established benchmarks: MNIST-Binary unconditional image generation, Fly Brain enhancer generation, and LM1B unconditional text generation. These tasks span controlled binary images, structured biological sequences, and open-ended language, allowing us to test whether the same coupling-based construction remains useful across qualitatively different discrete domains. The main paper keeps the benchmark setup concise but explicit, while Appendix G collects the full implementation details. In particular, Appendix G.1, Appendix G.2, and Appendix G.3 provide the complete training and evaluation protocols for the three tasks discussed below.

4.1 Coupling Model improves one-step generation across domains

We first evaluate the strict one-step setting, where generation consists of one Gaussian draw followed by one parallel decode. Tables 1 and 2 report the image and DNA results, while Figure 1 reports the LM1B entropy–perplexity tradeoff. Across all three settings, *Coupling Model* improves the available one-step baselines under the corresponding evaluation protocol.

MNIST-Binary setup. MNIST-Binary is a controlled discrete image-generation benchmark in which each binarized 28×28 image is treated as a discrete grid. We follow the unconditional-generation protocol used in CFM [Roos et al., 2026] and report FID as the primary sample-quality metric. Our model uses the two-stage *Coupling Model* pipeline: Stage A learns a latent coupling on a 7×7 latent grid with 16 channels, and Stage B trains a one-step binary decoder. The full benchmark configuration is reported in Appendix G.1.

Table 1 shows the strongest result for the *Coupling Model* model. With one decoding step, it achieves FID 5.50, improving over all one-step baselines in the comparison, including CFM [Roos et al., 2026] at 10.10 and PAIRFLOW [Park et al., 2026]+ReDi [Yoo et al., 2025] at 12.90. The result is also

competitive with several few-step and multi-step baselines, while preserving true one-step sampling. Figure 5 shows random samples from the trained generator.

Fly Brain enhancer setup. For DNA enhancer generation, we follow the Fly Brain benchmark from DFM [Gat et al., 2024]. Sequences have length 500 over $\{A, C, G, T\}$, and the benchmark contains 81 target classes. We report FBD, the Fréchet distance in the frozen classifier embedding space used by prior work. The corresponding training and evaluation details are in Appendix G.2.

Table 2: **Fly Brain enhancer generation.**

Model	Steps	FB FBD ↓
Reference		
Random Sequence	–	876
Multi-step		
Linear FM [Stärk et al., 2024]	100	15.0
Dirichlet FM [Stärk et al., 2024]	100	15.2
SLM [Song et al., 2025]	100	4.4 ± 0.2
Fisher-Flow [Davis et al., 2024]	100	3.8 ± 0.3
Autoregressive [Stärk et al., 2024]	500	25.2
One-step		
Dirichlet FM (dist.) [Stärk et al., 2024]	1	15.8
***Coupling Model* (Ours)**	**1**	**12.9 ± 0.3**

Table 2 shows that *Coupling Model* also transfers to structured biological sequences. It reduces Fly Brain FBD from 15.8 for the distilled one-step Dirichlet FM baseline to 12.9 ± 0.3. Stronger 100-step simplex diffusion and flow methods, especially SLM [Song et al., 2025] and Fisher-Flow [Davis et al., 2024], still perform better. We therefore view this result as evidence that *Coupling Model* is meaningful beyond binary images, while leaving a clear gap to stronger iterative generators in harder biological sequence settings.

LM1B setup. For LM1B, we use Figure 1 as the main one-step result, and Table 6 lists the raw numbers behind that tradeoff. We follow the FLM evaluation protocol and report generative perplexity under a GPT-2 Large scorer together with unigram entropy on 1,024 generated samples. Gen-PPL measures sample quality under an external language model, while entropy separates diverse generation from low-entropy degenerate samples. *Coupling Model* uses a frozen Qwen2.5-0.5B encoder in Stage A and a one-step Transformer decoder in Stage B; the full text-generation setup is given in §G.3.

At the lowest temperature in the sweep, *Coupling Model* reaches Gen-PPL 45.82 with entropy 3.6088, which is slightly better in perplexity than the LM1B reference point at Gen-PPL 47.47, though with lower entropy than the reference entropy of 4.31. As the temperature increases to 0.60 and 0.65, entropy rises to 4.0841 and 4.3192, while Gen-PPL remains at 76.19 and 98.96, respectively. Compared with one-step baselines such as FMLM [Lee et al., 2026] at 119.34 Gen-PPL and 4.16 entropy, or DFM [Gat et al., 2024] (ESD) at 68.11 Gen-PPL and 3.79 entropy, the sweep shows a cleaner tradeoff between quality and diversity than the prior one-step points. This result is weaker than the MNIST-Binary case in absolute quality, but it shows that the same *Coupling Model* construction remains credible for open-ended text generation.

4.2 Few-step refinement

The one-step results test whether *Coupling Model* can support direct parallel decoding. We next ask whether the same learned Gaussian latent space remains useful when the generator is allowed a small number of masked-refinement steps. Figure 3 compares LM1B Gen-PPL at 2, 4, and 8 sampling steps under the same evaluation protocol. *Coupling Model* draws one Gaussian latent $z \sim \mathcal{N}(0, I)$ and keeps it fixed throughout the masked-refinement trajectory. The additional steps refine local token decisions, while global sequence-level information is supplied by the continuous Gaussian latent. *Coupling Model* achieves the best Gen-PPL among the compared low-step methods at all three budgets. At 2 steps, it reaches Gen-PPL 66.2, improving over the strongest baseline DFM-ESD at 76.6 and DFM-PSD at 86.5. At 4 steps, it obtains 69.3, still slightly better than DFM-ESD at 71.2 and substantially better than DFM-PSD at 78.5 and FMLM at 97.6. At 8 steps, it reaches its best score, 64.3, improving over DFM-ESD at 65.4, DFM-PSD at 69.5, and FMLM at 84.4. Other low-step

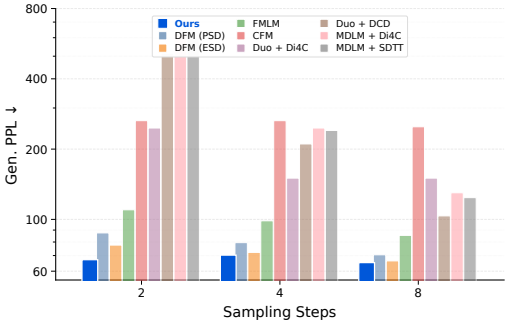

Figure 3: ***Coupling Model* remains competitive for few-step LM1B generation.**

Model	Guidance	Params (M)	FID ↓	Acc. ↑	NFE ↓	Sec/1k ↓
MDM	CFG	73.74	8.72	**99.8**	512	356.82
MDM	Classifier	73.74	7.95	97.2	256	244.68
MDM	Reward FT	73.74	8.82	99.7	256	203.04
Ours	CFG	**24.07**	5.64	99.1	2	1.72
Ours	Classifier	**24.07**	5.26	98.5	6	13.20
Ours	Reward FT	**24.07**	**5.09**	99.5	**1**	**0.63**

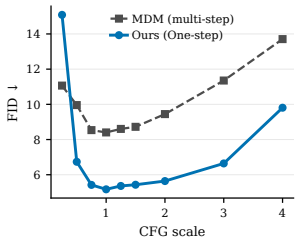

Figure 4: **Guidance is efficient and effective in one-step generation. Left**: *Coupling Model* achieves lower FID than the MDM baseline for CFG, classifier guidance, and reward fine-tuning while using far fewer function evaluations. **Right**: *Coupling Model* shows better quality–guidance tradeoff across scales.

acceleration methods, including CFM, Duo-based distillation, and MDLM-based distillation, remain substantially worse in this low-step regime.

4.3 Efficient one-step guidance

Finally, we test the guidance claim suggested by the method. In multi-step discrete generators, guidance must influence a long trajectory of non-differentiable stochastic token updates. The one-step model instead acts on the final logits, avoiding trajectory-level credit assignment. We compare against a larger 32×32 U-Net masked diffusion model trained on the same MNIST-Binary task with the same data budget. All evaluations use balanced digit labels, 1,000 generated samples, the same reward model, and the same held-out evaluation classifier. Guidance algorithms and hyperparameters are given in Appendix F.

Figure 4 shows that one-step guidance is more efficient across all three mechanisms while matching or improving FID. With classifier-free guidance, *Coupling Model* obtains FID 5.64 and 99.1% class accuracy using 2 function evaluations, compared with FID 8.72 and 99.8% accuracy for MDM using 512. With classifier guidance, *Coupling Model* reaches FID 5.26 and 98.5% accuracy using 6 evaluations, while MDM obtains FID 7.95 and 97.2% accuracy using 256. Reward fine-tuning gives the strongest *Coupling Model* result: FID 5.09, 99.5% accuracy, and 0.63 seconds per 1,000 samples, compared with FID 8.82, 99.7%, and 203.04 seconds for reward-fine-tuned MDM.

5 Conclusion

We presented Coupling Models as a representation-based route to fast discrete generation. Rather than compressing an existing denoising or flow trajectory, the method learns a sampleable coupling between discrete sequences and Gaussian latents, then trains a parallel decoder on the induced latent–sequence pairs. At inference time, generation is one-step: sample $z \sim \mathcal{N}(0, I)$ and decode once. Empirically, *Coupling Model* gives a consistent low-step advantage across MNIST-Binary, DNA enhancer design, and LM1B. The same latent interface also supports direct guidance and few-step masked refinement, suggesting that one-step discrete generation need not be viewed only as the limit of trajectory compression.

Limitations. The main limitation is scale. Although *Coupling Model* is evaluated across images, DNA, and language generation, the experiments use moderate-sized models and do not test frontier-scale language generation, long-context sequence modeling, or large-scale protein design. The results should therefore be read as evidence for a one-step modeling route rather than a final scaling study. Finally, the metrics for language generation, like generative perplexity, are often not well correlated with final sample quality, a phenomenon that is also commonly observed in image generative models with likelihood-based evaluation [Theis et al., 2015, Nalisnick et al., 2018].

Acknowledgments and Disclosure of Funding

We thank Zachary Bezemek for helpful discussions and Antonio Franca for assistance with the LM1B experiments. F.Z.P. and A.R.Z. are partially supported by NIH R01HL169347.

References

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. doi: 10.48550/arXiv.2107.03006. URL https://arxiv.org/abs/2107.03006. 2, 3, 7, 16

Pavel Avdeyev, Chenlai Shi, Yuhao Tan, Kseniia Dudnyk, and Jian Zhou. Dirichlet diffusion score model for biological sequence generation. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pages 1276–1301. PMLR, 2023. doi: 10.48550/arXiv.2305.10699. URL https://arxiv.org/abs/2305.10699. 7

Grigory Bartosh, Teodora Pandeva, Sushrut Karmalkar, and Javier Zazo. Forward-learned discrete diffusion: Learning how to noise to denoise faster. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=45EtKUdgbJ. 2, 3, 16

David Berthelot, Tianrong Chen, Jiatao Gu, Marco Cuturi, Laurent Dinh, Bhavik Chandna, Michal Klein, Josh Susskind, and Shuangfei Zhai. The coupling within: Flow matching via distilled normalizing flows. *arXiv preprint arXiv:2603.09014*, 2026. 3

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020a. 2

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners, 2020b. URL https://arxiv.org/abs/2005.14165. 2, 3, 16

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*, 2022. doi: 10.48550/arXiv.2202.04200. URL https://arxiv.org/abs/2202.04200. 16

Huiwen Chang, Han Zhang, Jarred Barber, AJ Maschinot, Jose Lezama, Lu Jiang, Ming-Hsuan Yang, Kevin Murphy, William T. Freeman, Michael Rubinstein, Yuanzhen Li, and Dilip Krishnan. Muse: Text-to-image generation via masked generative transformers. In *Proceedings of the 40th International Conference on Machine Learning*, 2023. doi: 10.48550/arXiv.2301.00704. URL https://arxiv.org/abs/2301.00704. 16

Tianqi Chen, Shujian Zhang, and Mingyuan Zhou. DLM-One: Diffusion language models for one-step sequence generation, 2025. URL https://arxiv.org/abs/2506.00290. 2, 3, 16

Ting Chen, Ruixiang Zhang, and Geoffrey E. Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning, 2022. URL https://arxiv.org/abs/2208.04202. 16

Chaoran Cheng, Jiahan Li, Jian Peng, and Ge Liu. Categorical flow matching on statistical manifolds. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. doi: 10.48550/arXiv.2405.16441. URL https://arxiv.org/abs/2405.16441. 2, 3, 7, 16

Justas Dauparas, Ivan Anishchenko, Nathaniel Bennett, Hua Bai, Robert J. Ragotte, Lukas F. Milles, Basile I. M. Wicky, Alexis Courbet, Rob J. de Haas, Neville Bethel, Philip J. Y. Leung, Timothy F. Huddy, Samuel Pellock, Doug Tischer, Francis Chan, Brian Koepnick, Hahn Nguyen, Alex Kang, B. Sankaran, Asim K. Bera, Neil P. King, and David Baker. Robust deep learning-based protein sequence design using proteinmpnn. *Science*, 378(6615):49–56, 2022. doi: 10.1126/science.add2187. 2

Oscar Davis, Samuel Kessler, Mircea Petrache, İsmail İlkan Ceylan, Michael Bronstein, and Avishek Joey Bose. Fisher flow matching for generative modeling over discrete data, 2024. URL https://arxiv.org/abs/2405.14664. 2, 3, 8, 16

Justin Deschenaux and Caglar Gulcehre. Beyond autoregression: Fast LLMs via self-distillation through time. In *International Conference on Learning Representations*, 2025. doi: 10.48550/arXiv.2410.21035. URL https://openreview.net/forum?id=uZ5K4HeNwd. 2, 3, 16

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *Advances in Neural Information Processing Systems*, 2021. 6

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=GTDKo3Sv9p. 2, 3, 7, 8, 16

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6112–6121, 2019. doi: 10.18653/v1/D19-1633. URL https://aclanthology.org/D19-1633/. 16

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. DiffuSeq: Sequence to sequence text generation with diffusion models. In *International Conference on Learning Representations*, 2023a. doi: 10.48550/arXiv.2210.08933. URL https://openreview.net/forum?id=jQj-_rLVXsj. 16

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. DiffuSeq-v2: Bridging discrete and continuous text spaces for accelerated seq2seq diffusion models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9868–9875, 2023b. doi: 10.18653/v1/2023.findings-emnlp.660. URL https://aclanthology.org/2023.findings-emnlp.660/. 16

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, Hao Peng, and Lingpeng Kong. Scaling diffusion language models via adaptation from autoregressive models. In *International Conference on Learning Representations*, 2025. doi: 10.48550/arXiv.2410.17891. URL https://openreview.net/forum?id=j1tSLYKwg8. 2, 3, 16

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1l8BtlCb. 16

Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. Vector quantized diffusion model for text-to-image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. doi: 10.48550/arXiv.2111.14822. URL https://arxiv.org/abs/2111.14822. 16

Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. SSD-LM: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pages 11575–11596, 2023. doi: 10.18653/v1/2023.acl-long.647. URL https://aclanthology.org/2023.acl-long.647/. 16

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 6

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. In *Advances in Neural Information Processing Systems*, 2021. doi: 10.48550/arXiv.2102.05379. URL https://proceedings.neurips.cc/paper/2021/hash/67d96d458abdef21792e6d8e590244e7-Abstract.html. 16

Emiel Hoogeboom, Alexey A. Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. Autoregressive diffusion models. In *International Conference on Learning Representations*, 2022. doi: 10.48550/arXiv.2110.02037. URL https://openreview.net/forum?id=Lm8T39vLDTE. 16

Vincent Tao Hu and Björn Ommer. [MASK] is all you need, 2024. URL https://arxiv.org/abs/2412.06787. 16

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017. 24

Chanhyuk Lee, Jaehoon Yoo, Manan Agarwal, Sheel Shah, Jerry Huang, Aditi Raghunathan, Seunghoon Hong, Nicholas M. Boffi, and Jinwoo Kim. One-step language modeling via continuous denoising, 2026. URL https://arxiv.org/abs/2602.16813. 2, 3, 8, 17

Jason Lee, Elman Mansimov, and Kyunghyun Cho. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182, 2018. doi: 10.18653/v1/D18-1149. URL https://aclanthology.org/D18-1149/. 16

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-LM improves controllable text generation. In *Advances in Neural Information Processing Systems*, 2022. doi: 10.48550/arXiv.2205.14217. URL https://openreview.net/forum?id=3s9IrEsjLyk. 16

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. doi: 10.48550/arXiv.2310.16834. URL https://arxiv.org/abs/2310.16834. 2, 3, 16

Justin Lovelace, Varsha Kishore, Chao Wan, Eliot Shekhtman, and Kilian Q. Weinberger. Latent diffusion for language generation. In *Advances in Neural Information Processing Systems*, 2023. doi: 10.48550/arXiv.2212.09462. URL https://openreview.net/forum?id=NKdtztladR. 16

Yiyang Lu, Qiao Sun, Xianbang Wang, Zhicheng Jiang, Hanhong Zhao, and Kaiming He. Bidirectional normalizing flow: From data to noise and back. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026. 3

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2017. 24

Amin Karimi Monsefi, Nikhil Bhendawade, Manuel Rafael Ciosici, Dominic Culver, Yizhe Zhang, and Irina Belousova. FS-DFM: Fast and accurate long text generation with few-step diffusion language models, 2025. URL https://arxiv.org/abs/2509.20624. 2, 3, 16

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? *arXiv preprint arXiv:1810.09136*, 2018. 9

Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. Scaling up masked diffusion models on text. In *International Conference on Learning Representations*, 2025. doi: 10.48550/arXiv.2410.18514. URL https://openreview.net/forum?id=WNvvwK0tut. 2, 3, 16

Hunter Nisonoff, Junhao Xiong, Stephan Allenspach, and Jennifer Listgarten. Unlocking guidance for discrete state-space diffusion and flow models. In *International Conference on Learning Representations*, 2025. 6

Mingue Park, Jisung Hwang, Seungwoo Yoo, Kyeongmin Yeo, and Minhyuk Sung. Pairflow: Closed-form source-target coupling for few-step generation in discrete flow models. In *International Conference on Learning Representations (ICLR)*, 2026. doi: 10.48550/arXiv.2512.20063. URL https://arxiv.org/abs/2512.20063. 2, 3, 7, 16

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, 2025a. URL https://arxiv.org/abs/2502.03540. 2, 6

Fred Zhangzhi Peng, Zachary Bezemek, Jarrid Rector-Brooks, Shuibai Zhang, Anru R. Zhang, Michael M. Bronstein, Avishek Joey Bose, and Alexander Tong. Planner aware path learning in diffusion language models training, 2025b. URL https://arxiv.org/abs/2509.23405. 16

Peter Potaptchik, Jason Yim, Adhi Saravanan, Peter Holderrieth, Eric Vanden-Eijnden, and Michael S. Albergo. Discrete flow maps, 2026. URL https://arxiv.org/abs/2604.09784. 2, 3, 17

Daan Roos, Oscar Davis, Floor Eijkelboom, Michael Bronstein, Max Welling, İsmail İlkan Ceylan, Luca Ambrogioni, and Jan-Willem van de Meent. Categorical flow maps, 2026. URL https://arxiv.org/abs/2602.12233. 2, 3, 7, 17, 24

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Mariano Marroquin, Justin T. Chiu, Alexander M. Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems*, 2024. 2, 3, 6, 7, 16

Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin T. Chiu, and Volodymyr Kuleshov. The diffusion duality. In *International Conference on Machine Learning (ICML)*, 2025. doi: 10.48550/arXiv.2506.10892. URL https://arxiv.org/abs/2506.10892. 2, 3, 7, 16

Yair Schiff, Subham Sekhar Sahoo, Hao Phung, Guanghan Wang, Sam Boshar, Hugo Dalla-torre, Bernardo P. de Almeida, Alexander Rush, Thomas Pierrot, and Volodymyr Kuleshov. Simple guidance mechanisms for discrete diffusion models. In *International Conference on Learning Representations (ICLR)*, 2025a. doi: 10.48550/arXiv.2412.10193. URL https://arxiv.org/abs/2412.10193. 2, 3, 7, 16

Yair Schiff, Subham Sekhar Sahoo, Hao Phung, Guanghan Wang, Sam Boshar, Hugo Dalla-Torre, Bernardo P. de Almeida, Alexander M. Rush, Thomas Pierrot, and Volodymyr Kuleshov. Simple guidance mechanisms for discrete diffusion models. In *International Conference on Learning Representations*, 2025b. 6

Yuxuan Song, Zhe Zhang, Yu Pei, Jingjing Gong, Qiying Yu, Zheng Zhang, Mingxuan Wang, Hao Zhou, Jingjing Liu, and Wei-Ying Ma. Shortlisting model: A streamlined simplex diffusion for discrete variable generation, 2025. URL https://arxiv.org/abs/2508.17345. 8

Hannes Stärk, Bowen Jing, Chenyu Wang, Gabriele Corso, Bonnie Berger, Regina Barzilay, and Tommi Jaakkola. Dirichlet flow matching with applications to dna sequence design. In *International Conference on Machine Learning (ICML)*, 2024. doi: 10.48550/arXiv.2402.05841. URL https://arxiv.org/abs/2402.05841. 2, 3, 7, 8, 16

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. Insertion transformer: Flexible sequence generation via insertion operations. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*. PMLR, 2019. URL https://proceedings.mlr.press/v97/stern19a.html. 16

Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015. 9

Hugo Touvron et al. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288. 2, 3, 16

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. doi: 10.48550/arXiv.1706.03762. URL https://arxiv.org/abs/1706.03762. 2, 3, 16

Chenyu Wang, Masatoshi Uehara, Yichun He, Amy Wang, Avantika Lal, Tommi Jaakkola, Sergey Levine, Aviv Regev, Hanchen Wang, and Tommaso Biancalani. Fine-tuning discrete diffusion models via reward optimization with applications to dna and protein design. In *International Conference on Learning Representations*, 2025a. 6

Xinyou Wang, Zaixiang Zheng, Fei Ye, Dongyu Xue, Shujian Huang, and Quanquan Gu. Diffusion language models are versatile protein learners. In *International Conference on Machine Learning*, 2024. doi: 10.48550/arXiv.2402.18567. URL https://arxiv.org/abs/2402.18567. 16

Xinyou Wang, Zaixiang Zheng, Fei Ye, Dongyu Xue, Shujian Huang, and Quanquan Gu. DPLM-2: A multimodal diffusion protein language model. In *International Conference on Learning Representations*, 2025b. doi: 10.48550/arXiv.2410.13782. URL https://openreview.net/forum?id=5z9GjHgerY. 16

An Yang et al. Qwen2 technical report, 2024. URL https://arxiv.org/abs/2407.10671. 2, 3, 16

Jaehoon Yoo, Wonjung Kim, and Seunghoon Hong. Redi: Rectified discrete flow. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. doi: 10.48550/arXiv.2507.15897. URL https://arxiv.org/abs/2507.15897. 2, 3, 7, 16

Tunyu Zhang, Xinxi Zhang, Ligong Han, Haizhou Shi, Xiaoxiao He, Zhuowei Li, Hao Wang, Kai Xu, Akash Srivastava, Hao Wang, Vladimir Pavlovic, and Dimitris N. Metaxas. T3D: Few-step diffusion language models via trajectory self-distillation with direct discriminative optimization, 2026. URL https://arxiv.org/abs/2602.12262. 2, 3, 16

Haoyang Zheng, Xinyang Liu, Cindy Xiangrui Kong, Nan Jiang, Zheyuan Hu, Weijian Luo, Wei Deng, and Guang Lin. Ultra-fast language generation via discrete diffusion divergence instruct. In *International Conference on Learning Representations*, 2026. doi: 10.48550/arXiv.2509.25035. URL https://arxiv.org/abs/2509.25035. 2, 3, 16

Lin Zheng, Jianbo Yuan, Lei Yu, and Lingpeng Kong. A reparameterized discrete diffusion model for text generation. In *First Conference on Language Modeling*, 2024. doi: 10.48550/arXiv.2302.05737. URL https://openreview.net/forum?id=PEQFHRUFca. 2, 3, 16

Kun Zhou, Yifan Li, Wayne Xin Zhao, and Ji-Rong Wen. Diffusion-NAT: Self-prompting discrete diffusion for non-autoregressive text generation, 2024. URL https://arxiv.org/abs/2305.04044. 16

Yuanzhi Zhu, Xi Wang, Stéphane Lathuilière, and Vicky Kalogeiton. Di[M]O: Distilling masked diffusion models into one-step generator. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025. doi: 10.48550/arXiv.2503.15457. URL https://arxiv.org/abs/2503.15457. 2, 3, 16

Appendix Contents

A Extended Related Work . . . 16
A.1 Autoregressive sequence generation . . . 16
A.2 Iterative discrete diffusion, masked refinement, and discrete flows . . . 16
A.3 One-step and few-step acceleration . . . 16
A.4 Flow-map methods . . . 17
A.5 Position of our method . . . 17
B Training and sampling algorithm . . . 17
C Expressivity barrier of direct one-step token-wise decoders . . . 17
C.1 Why a direct one-step token-wise decoder is limited . . . 17
C.2 Interpretation of the degree-of-freedom gap . . . 18
C.3 How existing model families resolve the trilemma . . . 19
C.4 Connection to our design principle . . . 19
C.5 Scope of the claim . . . 19
D Latent matching and one-step generation error . . . 20
E Few-step latent-conditioned masked diffusion . . . 21
E.1 Training . . . 21
E.2 Sampling . . . 22
F Guidance algorithms . . . 23
F.1 Relaxations and implementation choices . . . 24
G Experimental details . . . 24
G.1 MNIST-Binary details . . . 24
G.2 Fly Brain enhancer-generation details . . . 25
G.3 LM1B text-generation details . . . 26
G.4 Asset licenses and provenance . . . 27
G.5 Compute and implementation notes . . . 28

A Extended Related Work

This appendix expands the concise discussion in Section 2. The main paper groups prior work by inference regime: serial autoregressive generation, iterative diffusion and flow-based generation, and recent attempts to compress discrete generation into one or a few evaluations. Here we provide a more detailed account of those families and clarify how the proposed model construction differs from trajectory-compression and flow-map methods.

A.1 Autoregressive sequence generation

Autoregressive models represent a discrete sequence distribution through a product of conditional next-token distributions. This factorization has become the standard foundation for modern Transformer-based language models [Vaswani et al., 2017, Brown et al., 2020b, Touvron et al., 2023, Yang et al., 2024]. Its main advantage is expressivity: each token is generated while conditioning on all previously generated tokens, allowing the model to represent rich long-range dependencies through a sequence of local decisions. Its main limitation is inference structure. Generation proceeds token by token, so the number of model evaluations scales with sequence length. Non-autoregressive and insertion-based sequence models replace strict left-to-right decoding with parallel prediction, iterative refinement, or flexible insertion orders [Gu et al., 2018, Lee et al., 2018, Ghazvininejad et al., 2019, Stern et al., 2019]. These models are an important precursor to the speed-quality tradeoff studied here, but they still rely on refinement or specialized decoding structure rather than a single latent-to-sequence map.

A.2 Iterative discrete diffusion, masked refinement, and discrete flows

Discrete diffusion and masked-refinement models provide a more parallel alternative to autoregressive decoding. Instead of generating tokens sequentially, these models update many positions at once and recover global structure through repeated denoising, masking, or replacement steps [Austin et al., 2021, Hoogeboom et al., 2021, 2022, Chen et al., 2022, Sahoo et al., 2024, Lou et al., 2024, Zheng et al., 2024, Schiff et al., 2025a]. Text diffusion models have developed both continuous-token and discrete-token variants, including embedding-space, simplex-based, latent, and non-autoregressive formulations [Li et al., 2022, Gong et al., 2023a, Han et al., 2023, Lovelace et al., 2023, Gong et al., 2023b, Zhou et al., 2024, Nie et al., 2025, Gong et al., 2025, Peng et al., 2025b]. This improves parallelism across token positions, but the model must still be evaluated many times at inference. Discrete flow and simplex-based methods pursue related goals through transport or flow-matching objectives over categorical variables or their continuous relaxations [Gat et al., 2024, Cheng et al., 2024, Stärk et al., 2024, Davis et al., 2024]. Related masked-token generators in vision and vector-quantized image models also use iterative parallel decoding over discrete codebooks [Gu et al., 2022, Chang et al., 2022, 2023, Hu and Ommer, 2024]. In biological sequence modeling, diffusion language models have been adapted to protein generation and multimodal protein representations [Wang et al., 2024, 2025b]. These methods show that strong sample quality can be obtained when multiple denoising steps are allowed. They also sharpen the central question of this paper: whether one-step generation can recover useful global structure without relying on a long refinement trajectory.

A.3 One-step and few-step acceleration

A growing line of work asks whether the inference cost of iterative discrete generators can be compressed into the one-step or few-step regime. Distillation and consistency methods train a compressed sampler or student model to imitate a stronger multi-step process, reducing the number of model evaluations required at test time [Sahoo et al., 2025, Deschenaux and Gulcehre, 2025, Zhu et al., 2025, Chen et al., 2025]. Rectification methods improve the low-step trajectory itself by reducing the error induced when an iterative process is aggressively shortened [Yoo et al., 2025, Monsefi et al., 2025, Zheng et al., 2026, Zhang et al., 2026, Bartosh et al., 2026]. Coupling-based methods provide another route: PAIRFLOW [Park et al., 2026] constructs source-target couplings for few-step discrete flow models and shows that the choice of coupling can strongly affect low-step performance [Park et al., 2026]. These approaches demonstrate that fast discrete generation is feasible, but their central object is still a compressed, rectified, or otherwise improved generative trajectory.

A.4 Flow-map methods

Flow-map methods are especially close to the setting studied in this paper. In continuous generative modeling, a flow map directly predicts a later or terminal state of a generative trajectory, thereby avoiding expensive numerical integration. Recent work adapts this idea to discrete data by operating over continuous relaxations, one-hot embeddings, or probability-simplex representations. Discrete Flow Maps [Potaptchik et al., 2026] formulates discrete flow maps with objectives aligned to simplex geometry [Potaptchik et al., 2026]. FMLM [Lee et al., 2026] learns continuous flows over one-hot token embeddings and distills them into flow maps for efficient language generation [Lee et al., 2026]. CFM [Roos et al., 2026] defines categorical flow maps toward the simplex and trains them with self-distillation and endpoint-consistency objectives [Roos et al., 2026]. Although these methods differ in parameterization and loss design, they share a common premise: fast discrete generation can be obtained by learning better endpoint maps for continuous or simplex-valued trajectories.

A.5 Position of our method

Our method targets the same practical regime as recent one-step and few-step generators, but it uses a different construction. We do not distill a multi-step teacher, rectify a discrete trajectory, or learn an endpoint map along a simplex-valued path. Instead, we first learn a coupling between discrete sequences and Gaussian latents. This coupling induces paired supervision between sampleable latents and target sequences. A parallel decoder is then trained as a supervised inverse map from Gaussian latents to discrete outputs. The distinction is therefore not only architectural but also supervisory: flow-map and distillation methods inherit supervision from a generative trajectory, whereas our decoder is trained on latent-sequence pairs created by a learned sequence–noise coupling.

B Training and sampling algorithm

Algorithm 1 Two-stage training and one-step sampling

Input: training set $\mathcal{D} = \{x^{(i)}\}_{i=1}^{N}$
for Stage A updates **do**
  sample a minibatch $x \sim \mathcal{D}$ and noise $\epsilon \sim \mathcal{N}(0, I)$
  $u \leftarrow E_\psi(x, \epsilon)$
  $\hat{x}^A \leftarrow D(u), \quad z \leftarrow \mathrm{NF}_\phi(u)$
  update (ψ, ϕ, D) by minimizing $\mathcal{L}_A$ in Eq. (1)
end for
freeze E_ψ, D, and NF_ϕ
construct $\mathcal{D}_Z = \{(\mathrm{NF}_\phi(E_\psi(x^{(i)}, \epsilon^{(i)})), x^{(i)})\}_{i=1}^{N}$
for Stage B updates **do**
  sample a minibatch $(z, x) \sim \mathcal{D}_Z$
  update G_θ by minimizing $\mathcal{L}_B$ in Eq. (2)
end for
Sampling: draw $z \sim \mathcal{N}(0, I)$, compute $\ell \leftarrow G_\theta(z)$, and sample $\hat{x}_t \sim \mathrm{Cat}(\mathrm{softmax}(\ell_t/\tau))$

C Expressivity barrier of direct one-step token-wise decoders

This appendix expands the short motivation in Section 3. There, we argued that the difficulty of one-step discrete generation is not merely an optimization issue, but also a representational one: a direct one-step categorical decoder must recover global cross-token dependence in a single parallel prediction. Here we make that point precise, provide a complete proof of the expressivity barrier underlying Eq. (9), give additional examples, and clarify how this perspective relates to existing model families and to our method.

C.1 Why a direct one-step token-wise decoder is limited

In the main text, the key observation was that a direct one-step discrete decoder typically predicts one categorical distribution per position and then samples all tokens independently, yielding a factorized

sequence law

$$p(x) = \prod_{i=1}^{L} p_i(x_i). \tag{9}$$

We now formalize the family induced by this design.

Let $\mathcal{X} = [V]^L$ denote the discrete sequence space with vocabulary size V and sequence length L, and let $\Delta(\mathcal{X})$ denote the simplex of all probability distributions over $\mathcal{X}$. Consider the family

$$\mathcal{F}_{\text{fact}} = \left\{ q \in \Delta(\mathcal{X}) \ : \ q(x) = \prod_{i=1}^{L} q_i(x_i), \ \ q_i \in \Delta([V]) \right\}. \tag{10}$$

This is exactly the distribution family induced by a direct one-step token-wise decoder with independent per-position sampling.

The main-text claim can now be stated precisely: the family in Eq. (10) is too small to represent a general joint law over sequences.

Proposition 1 (Expressivity barrier of direct one-step token-wise decoders)**.** *For any $V, L \geq 2$, the factorized family $\mathcal{F}_{\text{fact}}$ is a strict subset of $\Delta([V]^L)$. Equivalently, a direct one-step token-wise discrete decoder is not universally expressive over sequence distributions.*

Proof. We prove the claim in two complementary ways.

First, consider the degrees of freedom. The full sequence distribution lives in the simplex $\Delta([V]^L)$, whose dimension is

$$\dim \Delta([V]^L) = V^L - 1. \tag{11}$$

By contrast, each factor $q_i \in \Delta([V])$ contributes only $V - 1$ free parameters, so the factorized family in Eq. (10) has dimension

$$\dim \mathcal{F}_{\text{fact}} = L(V - 1). \tag{12}$$

For all $V, L \geq 2$,

$$L(V - 1) < V^L - 1. \tag{13}$$

Therefore, $\mathcal{F}_{\text{fact}}$ cannot coincide with the full sequence simplex. In other words, the family induced by direct one-step token-wise decoding is a strict low-dimensional subset of all discrete sequence distributions.

Second, it is useful to exhibit a concrete distribution that lies outside this family. Consider the binary length-two case $V = 2$, $L = 2$, and define

$$p^\star(0,0) = \tfrac{1}{2}, \qquad p^\star(1,1) = \tfrac{1}{2}, \qquad p^\star(0,1) = p^\star(1,0) = 0. \tag{14}$$

This distribution places all probability mass on the two perfectly correlated outcomes. Suppose, for contradiction, that $p^\star \in \mathcal{F}_{\text{fact}}$. Then there exist $q_1, q_2 \in \Delta([2])$ such that

$$p^\star(x_1, x_2) = q_1(x_1) q_2(x_2). \tag{15}$$

Any such factorized distribution must satisfy

$$p^\star(0,0)\, p^\star(1,1) = p^\star(0,1)\, p^\star(1,0), \tag{16}$$

because both sides are equal to $q_1(0)q_1(1)q_2(0)q_2(1)$. But for the target distribution in Eq. (14),

$$p^\star(0,0)\, p^\star(1,1) = \tfrac{1}{4}, \qquad p^\star(0,1)\, p^\star(1,0) = 0, \tag{17}$$

which is impossible. Hence $p^\star \notin \mathcal{F}_{\text{fact}}$.

Together, the dimension argument and the explicit counterexample establish the proposition. □

C.2 Interpretation of the degree-of-freedom gap

The main text summarized the above result by noting that a general sequence distribution has $V^L - 1$ degrees of freedom, whereas the factorized family in Eq. (9) has only $L(V - 1)$. This gap is the mathematical form of the representational barrier.

The key point is that the problem is structural rather than pathological. The family in Eq. (10) excludes any target law with nontrivial cross-token dependency. This includes simple correlated

patterns such as Eq. (14), but also more realistic structures such as repeated motifs, long-range constraints, parity-type dependencies, and other globally coordinated sequence patterns. Thus, before considering optimization, architecture, or training objective, a direct one-step token-wise discrete decoder is already too weak to model the kind of dependency structure that discrete sequence data typically exhibit.

This also explains why the factorized decoder is simultaneously attractive and limited. It is attractive because its output size grows only linearly with sequence length and vocabulary size. It is limited because linear scaling in L cannot directly parameterize an exponentially large joint probability space.

C.3 How existing model families resolve the trilemma

The main text then used this barrier to explain how existing model families resolve the trilemma.

Autoregressive models retain *native discrete modeling* and expressive joint structure, but sacrifice *parallel one-step generation*. Their key advantage is that they do not attempt to represent the full sequence law through a single factorized prediction of the form in Eq. (9). Instead, they decompose the joint distribution into a sequence of conditional distributions, allowing dependency structure to accumulate through the generation order.

Masked diffusion models make a closely related tradeoff. They also remain in the native discrete space and typically use local token-wise predictions, but they do not rely on a single direct one-step decode. Instead, they recover expressive joint behavior through multiple denoising or refinement steps. In this sense, they preserve native discreteness and local parameterization, but give up the one-step requirement.

At the opposite extreme, exact one-step native discrete generation can be obtained by explicitly parameterizing the full joint distribution over the entire sequence space. However, this requires a number of outputs or equivalent representational complexity that scales as $O(V^L)$, which becomes computationally prohibitive for realistic sequence lengths.

Continuous-relaxation methods such as simplex flow and Fisher-Flow resolve the tension differently. Rather than directly parameterizing a discrete sequence law in one shot, they embed categorical data into a continuous surrogate geometry and perform transport in that continuous space. This preserves local parameterization and can support transport-style parallel generation, but it no longer remains within the class of direct native-discrete one-step decoders covered by Proposition 1.

C.4 Connection to our design principle

The main-text design principle now follows directly from the above discussion. The obstacle is not that a single forward pass is inherently incapable of producing a complex discrete sequence. Rather, the obstacle is that a *direct* one-step token-wise categorical decoder is too weak to represent a general joint law over sequence space.

Our method therefore avoids assigning that burden to the final discrete decoder. Instead, we first learn a deterministic coupling between sequences and a continuous Gaussian latent space using a normalizing flow encoder trained by maximum likelihood. This coupling absorbs global dependency structure into a continuous representation. We then train a generator to decode from Gaussian noise to the target sequence in a single forward pass.

From the perspective of the trilemma, our method preserves one-step generation at inference time, but escapes the barrier in Proposition 1 by giving up direct native-discrete one-shot parameterization and replacing it with a learned continuous surrogate space. This is precisely why the proposition does not contradict our modeling approach: our method is deliberately designed to fall outside the restrictive factorized family in Eq. (10).

C.5 Scope of the claim

Finally, we clarify the scope of the result.

Proposition 1 is not a universal impossibility theorem for all one-step generators. Rather, it is a precise structural barrier for a broad and practically important class of models: direct one-step token-wise discrete decoders with independent per-position sampling.

Accordingly, the proposition does *not* imply that one-step discrete generation is impossible in full generality. A full-joint decoder over $[V]^L$ can represent any target law exactly, but is typically intractable because its complexity grows exponentially in sequence length. Likewise, the proposition does *not* apply to methods that introduce a learned continuous surrogate space or coupling-based latent representation before decoding. Such methods step outside the class in Eq. (10), which is exactly how they can preserve one-step generation without resorting to an explicit full-joint discrete parameterization.

For this reason, we view the trilemma not as a blanket impossibility statement for one-step generation, but as a useful lens for understanding why direct one-step native-discrete decoding is difficult, why iterative discrete models remain effective, and why a learned continuous coupling provides a natural alternative.

D Latent matching and one-step generation error

We give the formal version of Proposition 3.1. Let $p_{\mathrm{data}}(x)$ be the data distribution. Let $q_\psi(u \mid x)$ be the Stage A encoder distribution and let $T_\phi = \mathrm{NF}_\phi$ denote the learned flow map. Define the Stage A joint distribution $q_\phi(z, x)$ by

$$x \sim p_{\mathrm{data}}, \qquad u \sim q_\psi(u \mid x), \qquad z = T_\phi(u).$$

Let $q_\phi(z)$ and $q_\phi(x \mid z)$ denote the corresponding latent marginal and conditional sequence distribution. Let $p_Z = \mathcal{N}(0, I)$ be the prior used for one-step sampling, and define

$$p_\theta^{\mathrm{gen}}(x) = \int G_\theta(x \mid z) p_Z(z)\, dz.$$

Proposition 2 (Latent matching controls one-step generation error)**.** *For any decoder $G_\theta(x \mid z)$,*

$$\mathrm{TV}(p_{\mathrm{data}}, p_\theta^{\mathrm{gen}}) \le \mathbb{E}_{z\sim q_\phi(z)} \mathrm{TV}(q_\phi(\cdot \mid z), G_\theta(\cdot \mid z)) + \mathrm{TV}(q_\phi(z), p_Z(z))\,. \tag{18}$$

If

$$\mathbb{E}_{z\sim q_\phi(z)} \mathrm{KL}(q_\phi(\cdot \mid z) \,\|\, G_\theta(\cdot \mid z)) \le \varepsilon_{\mathrm{dec}}$$

and

$$\mathrm{KL}(q_\phi(z) \,\|\, p_Z(z)) \le \varepsilon_{\mathrm{flow}},$$

then

$$\mathrm{TV}(p_{\mathrm{data}}, p_\theta^{\mathrm{gen}}) \le \sqrt{\frac{\varepsilon_{\mathrm{dec}}}{2}} + \sqrt{\frac{\varepsilon_{\mathrm{flow}}}{2}}. \tag{19}$$

In particular, if $q_\phi(z) = p_Z$ and $G_\theta(\cdot \mid z) = q_\phi(\cdot \mid z)$ for q_ϕ-almost every z, then $p_\theta^{\mathrm{gen}} = p_{\mathrm{data}}$.

Proof. Let

$$p_\theta^q(x) = \int G_\theta(x \mid z)\, q_\phi(z)\, dz \tag{20}$$

denote the distribution obtained by decoding from the Stage A latent marginal rather than from the inference prior. By the triangle inequality,

$$\mathrm{TV}(p_{\mathrm{data}}, p_\theta^{\mathrm{gen}}) \le \mathrm{TV}(p_{\mathrm{data}}, p_\theta^q) + \mathrm{TV}(p_\theta^q, p_\theta^{\mathrm{gen}})\,. \tag{21}$$

For the first term, the Stage A joint has marginal

$$p_{\mathrm{data}}(x) = \int q_\phi(x \mid z)\, q_\phi(z)\, dz. \tag{22}$$

Therefore,

$$\mathrm{TV}(p_{\mathrm{data}}, p_\theta^q) = \frac{1}{2} \sum_x \left| \int (q_\phi(x \mid z) - G_\theta(x \mid z))\, q_\phi(z)\, dz \right| \tag{23}$$

$$\le \int \frac{1}{2} \sum_x |q_\phi(x \mid z) - G_\theta(x \mid z)|\, q_\phi(z)\, dz \tag{24}$$

$$= \mathbb{E}_{z\sim q_\phi(z)} \mathrm{TV}(q_\phi(\cdot \mid z), G_\theta(\cdot \mid z))\,. \tag{25}$$

For the second term, $G_\theta(\cdot \mid z)$ is a Markov kernel from latents to sequences. Total variation cannot increase under a Markov kernel, so

$$\mathrm{TV}(p_\theta^q, p_\theta^{\mathrm{gen}}) \le \mathrm{TV}(q_\phi(z), p_Z(z))\,. \tag{26}$$

Combining these two bounds proves Eq. (3).

For the KL-based bound, Pinsker's inequality gives

$$\mathrm{TV}(q_\phi(\cdot \mid z), G_\theta(\cdot \mid z)) \le \sqrt{\frac{1}{2}\mathrm{KL}(q_\phi(\cdot \mid z) \,\|\, G_\theta(\cdot \mid z))}. \tag{27}$$

Taking expectation over $z \sim q_\phi(z)$ and applying Jensen's inequality,

$$\mathbb{E}_{z\sim q_\phi(z)}\mathrm{TV}(q_\phi(\cdot \mid z), G_\theta(\cdot \mid z)) \le \mathbb{E}_{z\sim q_\phi(z)}\sqrt{\frac{1}{2}\mathrm{KL}(q_\phi(\cdot \mid z) \,\|\, G_\theta(\cdot \mid z))} \tag{28}$$

$$\le \sqrt{\frac{1}{2}\mathbb{E}_{z\sim q_\phi(z)}\mathrm{KL}(q_\phi(\cdot \mid z) \,\|\, G_\theta(\cdot \mid z))} \tag{29}$$

$$\le \sqrt{\frac{\varepsilon_{\mathrm{dec}}}{2}}. \tag{30}$$

Finally, applying Pinsker's inequality to the latent marginal gives

$$\mathrm{TV}(q_\phi(z), p_Z(z)) \le \sqrt{\frac{1}{2}\mathrm{KL}(q_\phi(z) \,\|\, p_Z(z))} \le \sqrt{\frac{\varepsilon_{\mathrm{flow}}}{2}}, \tag{31}$$

which proves Eq. (19). The final consistency statement follows by setting both terms in Eq. (3) to zero. □

E Few-step latent-conditioned masked diffusion

This appendix gives the full training and sampling procedures for the few-step extension introduced in Section 3.2. The extension uses the same Stage A latent–sequence pairs as the one-step model. After Stage A is trained and frozen, each training sequence is mapped to a Gaussian latent $z^{(i)} = \mathrm{NF}_\phi(E_\psi(x^{(i)}, \epsilon^{(i)}))$ with $\epsilon^{(i)} \sim \mathcal{N}(0, I)$, which yields the paired dataset $\mathcal{D}_Z = \{(z^{(i)}, x^{(i)})\}_{i=1}^N$. The one-step model trains a direct decoder $G_\theta(z)$ on this paired dataset. The few-step variant instead trains a latent-conditioned masked denoiser $H_\theta(x_t, z)$. Thus, Stage A is unchanged; only the Stage B generator is replaced by a masked diffusion model conditioned on the Gaussian latent.

E.1 Training

For each paired example $(z, x) \sim \mathcal{D}_Z$, we sample a masking time $t \sim \mathrm{Unif}(0, 1)$ and independently mask each position with probability t. Let $m \in \{0, 1\}^T$ be the resulting binary mask, where $m_i = 1$ indicates that position i is masked. The corrupted sequence x_t is obtained by setting $(x_t)_i = \mathsf{M}$ if $m_i = 1$ and $(x_t)_i = x_i$ otherwise. The denoiser receives only the masked sequence and the latent variable, $H_\theta(x_t, z)$, and predicts the clean sequence distribution. The loss is evaluated on the masked positions:

$$\mathcal{L}_{\mathrm{MDM}}(\theta) = -\mathbb{E}_{(z,x)\sim\mathcal{D}_Z,\, t\sim\mathrm{Unif}(0,1),\, m}\left[\frac{1}{\sum_{i=1}^T m_i}\sum_{i:m_i=1}\log p_\theta(x_i \mid x_t, z)\right]. \tag{32}$$

The training procedure is summarized in Algorithm 2.

Algorithm 2 Training latent-conditioned masked diffusion

Input: paired latent–sequence dataset $\mathcal{D}_Z = \{(z^{(i)}, x^{(i)})\}_{i=1}^{N}$
Model: latent-conditioned masked denoiser H_θ
for training updates **do**
    sample a minibatch $(z, x) \sim \mathcal{D}_Z$
    sample masking times $t \sim \mathrm{Unif}(0, 1)$
    sample masks $m_i \sim \mathrm{Bernoulli}(t)$ independently across positions
    resample m if $\sum_{i=1}^{T} m_i = 0$
    construct x_t by replacing positions with $m_i = 1$ by M
    compute logits $\ell \leftarrow H_\theta(x_t, z)$
    update θ by minimizing the masked-token loss in Eq. (32)
end for

E.2 Sampling

At inference time, we first sample a Gaussian latent $z \sim \mathcal{N}(0, I)$ and initialize the sequence as fully masked, $x^{(0)} = (\mathsf{M}, \ldots, \mathsf{M})$. We then run a latent-conditioned P2-self sampler for K iterations. Let $\kappa : [0, 1] \to [0, 1]$ be a monotone unmasking schedule with $\kappa(0) = 0$ and $\kappa(1) = 1$. At iteration i, we set $t_i = i/K$ and choose the number of positions that should remain masked as $\lfloor L(1 - \kappa(t_i)) \rfloor$, where L is the sequence length.

Given the current partially generated sequence $x^{(i-1)}$, the denoiser predicts logits $\ell^{(i)} = H_\theta(x^{(i-1)}, z)$. We sample a candidate clean token for each non-fixed position using the Gumbel-max form of categorical sampling with temperature τ_i, and use the corresponding log-probability as a confidence score. P2-self then plans the next mask set by selecting the lowest-confidence positions to remain masked. This differs from a purely monotone unmasking procedure: positions that were previously masked may be revealed, while positions that were previously revealed may be remasked if their confidence is low. For previously revealed positions, we multiply the confidence score by a remasking strength η. Since the score is a log-probability, larger η makes such positions easier to remask, whereas smaller η makes already revealed tokens more persistent.

Thus, each iteration consists of three operations: propose candidate clean tokens, select the low-confidence positions that should remain masked, and write candidate tokens only into positions that were masked and are no longer selected to remain masked. After the final step, all remaining mask tokens are filled using the last denoiser proposal. The Gaussian latent z is fixed throughout the trajectory and provides global sequence-level conditioning at every denoising step. When $K = 1$, the method reduces to one-step latent-conditioned masked decoding; for $K > 1$, it performs iterative P2-self refinement with possible remasking.

The full procedure is summarized in Algorithm 3.

Algorithm 3 Few-step latent-conditioned P2-self sampling

Require: number of sampling steps K, sequence length L, mask token M, denoiser H_θ, unmasking schedule κ, temperatures $\{\tau_i\}_{i=1}^{K}$, remasking strength η

sample $z \sim \mathcal{N}(0, I)$
initialize $x^{(0)} \leftarrow (\mathsf{M}, \ldots, \mathsf{M})$
set $f_j \leftarrow 1$ if $x_j^{(0)} \neq \mathsf{M}$, and $f_j \leftarrow 0$ otherwise
for $i = 1, \ldots, K$ **do**
  $t_i \leftarrow i/K$
  compute logits $\ell^{(i)} \leftarrow H_\theta(x^{(i-1)}, z)$
  for each position j with $f_j = 0$ **do**
    sample Gumbel noise $g_{j,v} \sim \mathrm{Gumbel}(0, 1)$ for each token v
    $\tilde{\ell}_{j,v}^{(i)} \leftarrow \ell_{j,v}^{(i)}/\tau_i + g_{j,v}$
    propose $\hat{x}_j^{(i)} \leftarrow \arg\max_v \tilde{\ell}_{j,v}^{(i)}$
    score $s_j^{(i)} \leftarrow \log \mathrm{softmax}(\tilde{\ell}_j^{(i)})_{\hat{x}_j^{(i)}}$
  end for
  set $s_j^{(i)} \leftarrow +\infty$ for all fixed positions j with $f_j = 1$
  define previously generated positions $\mathcal{U}^{(i)} \leftarrow \{j : x_j^{(i-1)} \neq \mathsf{M},\ f_j = 0\}$
  rescale $s_j^{(i)} \leftarrow \eta s_j^{(i)}$ for all $j \in \mathcal{U}^{(i)}$
  set $m_i \leftarrow \lfloor L(1 - \kappa(t_i)) \rfloor$
  let $\mathcal{R}^{(i)}$ be the m_i non-fixed positions with the lowest scores $s_j^{(i)}$
  set $x^{(i)} \leftarrow x^{(i-1)}$
  set $x_j^{(i)} \leftarrow \mathsf{M}$ for all $j \in \mathcal{R}^{(i)}$
  define newly revealed positions $\mathcal{A}^{(i)} \leftarrow \{j : x_j^{(i-1)} = \mathsf{M}\} \setminus \mathcal{R}^{(i)}$
  set $x_j^{(i)} \leftarrow \hat{x}_j^{(i)}$ for all $j \in \mathcal{A}^{(i)}$
end for
fill any remaining masked position j by setting $x_j^{(K)} \leftarrow \hat{x}_j^{(K)}$
return $x^{(K)}$

F Guidance algorithms

This appendix gives the procedural details for the guidance mechanisms summarized in Section 3.3. The goal is to specialize standard guidance methods to *Coupling Model*. Since generation consists of sampling $z \sim \mathcal{N}(0, I)$ and decoding once, guidance acts either on the final logits, on the latent variable before decoding, or on the generator parameters during fine-tuning.

Algorithm 4 Classifier-free guidance for one-step generation

Require: generator G_θ, condition y, guidance scale s, temperature τ

sample $z \sim \mathcal{N}(0, I)$
compute conditional logits $\ell^c \leftarrow G_\theta(z, y)$
compute unconditional logits $\ell^u \leftarrow G_\theta(z, \varnothing)$
combine logits $\ell^{\mathrm{cfg}} \leftarrow \ell^u + s(\ell^c - \ell^u)$
sample $\hat{x}_t \sim \mathrm{Cat}(\mathrm{softmax}(\ell_t^{\mathrm{cfg}}/\tau))$ for all positions t in parallel
return $\hat{x}$

Algorithm 5 Latent-space classifier guidance

Require: generator G_θ, reward $R(\cdot, y)$, condition y
Require: step size η, guidance steps K, relaxation ρ, sampling temperature τ

sample $z^0 \sim \mathcal{N}(0, I)$
for $k = 0, \ldots, K - 1$ **do**
  compute logits $\ell^k \leftarrow G_\theta(z^k, y)$
  compute relaxed output $\tilde{x}^k \leftarrow \rho(\ell^k)$
  update $z^{k+1} \leftarrow z^k + \eta \nabla_{z^k} R(\tilde{x}^k, y)$
end for
compute final logits $\ell^K \leftarrow G_\theta(z^K, y)$
sample $\hat{x}_t \sim \mathrm{Cat}(\mathrm{softmax}(\ell_t^K/\tau))$ for all positions t in parallel
return $\hat{x}$

Table 3: Summary of the experimental benchmarks used in the paper.

Benchmark	Data / setting	Metric	Protocol source
MNIST-Binary	Unconditional generation on binarized 28×28 MNIST digits	FID ↓	Follows the comparison protocol used by CFM [Roos et al., 2026]
Fly Brain enhancer generation	Conditional DNA sequence generation on DeepFlyBrain, 81 target classes, sequence length 500	FBD ↓	Follows the enhancer-generation benchmark of Dirichlet Flow Matching
LM1B	Unconditional one-step language generation on LM1B, sequence length 128	Gen-PPL ↓, entropy	Follows the LM1B evaluation style of FLM

Algorithm 6 Reward fine-tuning for one-step generation

Require: pretrained generator G_{θ_0}, trainable generator G_θ
Require: reward $R(\cdot, y)$, relaxation ρ
Require: weights λ_{reward}, λ_{anchor}
initialize $\theta \leftarrow \theta_0$
for fine-tuning updates **do**
  sample condition y and latent $z \sim \mathcal{N}(0, I)$
  compute relaxed output $\tilde{x}_\theta \leftarrow \rho(G_\theta(z, y))$
  compute anchor penalty $\mathcal{R}(\theta, \theta_0)$
  update θ by minimizing $\mathcal{L}_{\text{FT}} = -\lambda_{\text{reward}} R(\tilde{x}_\theta, y) + \lambda_{\text{anchor}} \mathcal{R}(\theta, \theta_0)$
end for
return fine-tuned generator G_θ

F.1 Relaxations and implementation choices

For classifier guidance and reward fine-tuning, the reward gradient must pass through the model output. We use two final-layer relaxations. The soft relaxation sets $\rho(\ell) = \text{softmax}(\ell/\tau_r)$, or the corresponding sigmoid relaxation for binary outputs, and evaluates the reward on these continuous probabilities. This gives a low-variance gradient but introduces a mismatch between relaxed optimization and final discrete sampling. The Gumbel-based relaxation instead uses a Gumbel-Softmax or straight-through estimator [Jang et al., 2017, Maddison et al., 2017], which more closely matches the categorical sampling path but can be noisier.

In all guidance experiments, CFG uses conditioning dropout during training and two generator evaluations at sampling time, one conditional and one unconditional. Latent-space classifier guidance keeps the generator fixed and uses a small number of gradient steps on z before the final decode. Reward fine-tuning initializes from the pretrained generator and uses an anchor penalty, implemented as a logit-level or distribution-level matching loss, to discourage drift from the data distribution while optimizing the reward.

G Experimental details

This appendix collects the implementation details used for the experiments in Section 4. The goal is to make the retained benchmarks easy to reproduce while keeping the main paper focused on the empirical argument. Table 3 summarizes the three tasks and their evaluation protocols, and the following subsections provide benchmark-specific training and evaluation details.

G.1 MNIST-Binary details

We use the standard torchvision MNIST split and binarize each pixel by thresholding the grayscale value at 0.5. Each image is then flattened into a length-784 binary sequence for training, while evaluation samples are reshaped back to 28×28 images. The main paper reports the unconditional setting (`-mode uncond`), and the code also supports a class-conditional variant.

Training uses a two-stage schedule over 200 epochs, with the first half used for Stage A and the second half for Stage B. Stage A trains the binary autoencoder and latent flow, while Stage B freezes

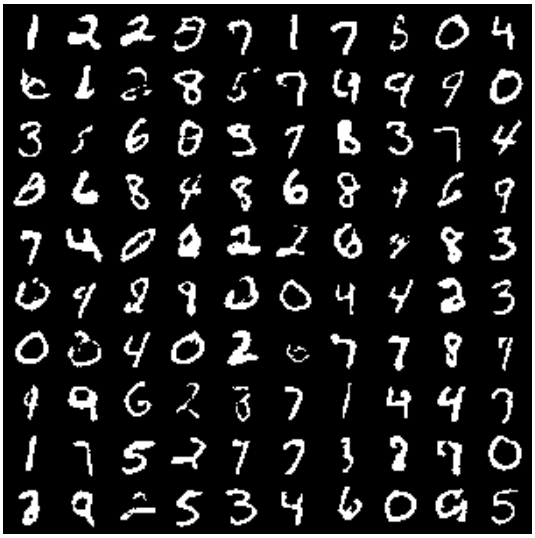

Figure 5: Unconditional MNIST-Binary generations from our model.

Table 4: MNIST-Binary training configuration used for the main result.

Item	Value
Dataset	MNIST, binarized at threshold 0.5
Task	Unconditional one-step generation
Batch size	256
Epochs	200 total (100 Stage A, 100 Stage B)
Optimizer	AdamW
Learning rate	2×10^{-4}
Weight decay	10^{-4}
Latent spatial size	7×7
Latent channels	16
Latent noise std	0.5
Flow width / blocks / layers / heads	128 / 5 / 5 / 4
Generator width / depth	512 / 8
FID evaluation	1,000 generated samples, one step
Random seed	0

them and trains the one-step decoder. We optimize all trainable parameters with AdamW, learning rate 2×10^{-4}, weight decay 10^{-4}, and a cosine decay schedule with one-epoch warmup. We evaluate FID with 1,000 generated samples against the full MNIST training set, using the `torchmetrics` implementation of Fréchet Inception Distance.

G.2 Fly Brain enhancer-generation details

For DNA sequence design, we follow the enhancer benchmark from Dirichlet Flow Matching and retain only the Fly Brain dataset in the paper. The dataset configuration used by our code corresponds to `DeepFlyBrain_data.pkl`, with 81 target classes and sequence length 500 over the alphabet $\{A, C, G, T\}$. The training loader uses the benchmark train split and the default validation split (`valid` rather than `test`). Evaluation uses FBD in the embedding space of the frozen benchmark classifier, matching the protocol implemented for prior sequence-design baselines. Our code also reports a `flow_reverse_fbd` diagnostic for the reverse map of Stage A, but the main paper reports the one-step generator FBD.

The Fly Brain experiment is trained for 300 epochs. We use the first 150 epochs for Stage A and the remaining 150 for Stage B. The run in `train_enhancer_fb.sh` uses batch size 128, compression ratio 0.05 (which corresponds to latent length 25 for length-500 sequences), latent width 256, dropout 0.1, and flow depth 8 blocks. Parameters not overridden in the launch script use the defaults in `one_step_enhancer.py`; in particular, the optimizer is AdamW with learning rate 2×10^{-4} and

Table 5: Fly Brain enhancer-generation configuration used for the main result.

Item	Value
Dataset	DeepFlyBrain (`fb`)
Sequence length / alphabet	500 / $\{A, C, G, T\}$
Number of target classes	81
Train / eval split	train / valid
Metric	FBD in frozen classifier embedding space
Batch size / eval batch size	128 / 128
Epochs	300 total (150 Stage A, 150 Stage B)
Optimizer	AdamW
Learning rate / weight decay	2×10^{-4} / 10^{-4}
Compression ratio	0.05
Latent length	25
Latent channels	256
Latent noise std	1.0
Encoder width / depth	256 / 8
Decoder depth	8
Generator width / depth	512 / 8
Flow width / blocks / layers / heads	128 / 8 / 4 / 4
Dropout	0.1
β_{KL}	1.0
Alignment losses	Off in the reported run
Random seed	0

weight decay 10^{-4}. No external embedding alignment or Stage B alignment is enabled in the reported configuration.

G.3 LM1B text-generation details

For text generation, we use LM1B tokenized with the `Qwen/Qwen2.5-0.5B` tokenizer. The preprocessing pipeline truncates or pads each sequence to length 128 and stores tokenized training and test splits as memory-mapped NumPy arrays. The underlying dataset loader uses `dvruette/lm1b` with the `~parquet` revision. All text experiments use the same two-stage training pipeline as the rest of the paper, but with a frozen pretrained language-model encoder in Stage A and a Transformer decoder in Stage B.

Stage A uses a frozen Qwen2.5-0.5B encoder, a learned linear projection from 896 to 256 dimensions, a RealNVP flow with 12 coupling layers, and a lightweight Transformer decoder that reconstructs Qwen-tokenized text. The dequantization noise scale is 0.2. Stage A is trained for 200k optimization steps with batch size 128, AdamW, learning rate 3×10^{-4}, weight decay 0.01, and 2k warmup steps. The flow-weight coefficient is annealed from 0.5 to 1.0 over the first 5% of training.

Stage B uses a one-step Transformer generator with width 768, depth 12, 12 attention heads, and feed-forward dimension 2048. It is trained for 400k steps with batch size 32, AdamW, learning rate 2×10^{-4}, weight decay 0.01, 2k warmup steps, cosine restarts every 50k steps, gradient clipping at 1.0, and EMA decay 0.999. In addition to token-level cross entropy, Stage B uses KL distillation from a frozen Qwen2.5-0.5B causal LM with distillation weight 1.0 and temperature 2.0. During training, 30% of updates replace the Stage A latent with pure Gaussian noise scaled by 0.90 to better match one-step prior sampling at inference time.

For evaluation, we generate 1,024 samples from the one-step model, decode them with the Qwen tokenizer, and score the resulting text with GPT-2 Large. We report Gen-PPL and average unigram entropy. Table 6 lists the raw values for the temperature sweep with top-p 0.95 and latent scale 0.8, and Figure 6 repeats the same tradeoff plot for completeness. We also compute an LM1B reference entropy of 4.31 and a reference Gen-PPL using the same decode-and-score pipeline on real LM1B test sentences. Because our generated text is decoded with a Qwen tokenizer before scoring with GPT-2 Large, the absolute Gen-PPL values are tied to this evaluation pipeline; we therefore use the FLM comparison primarily as a relative benchmark.

Table 6: Raw entropy and generative perplexity values used in Figure 1 for one-step LM1B generation. For our method, we report the temperature sweep with fixed top-$p = 0.95$ and latent scale $z_{\text{scale}} = 0.8$. Lower generative perplexity and higher entropy are preferred.

Category	Method	Temperature	Gen. PPL ↓	Entropy ↑
Baseline	Duo + DCD	–	1224.52	4.33
Baseline	Duo + Di4C	–	292.94	3.79
Baseline	MDLM + SDTT	–	1429.48	4.31
Baseline	MDLM + Di4C	–	1217.10	4.38
Baseline	CFM	–	269.72	3.10
Baseline	FMLM	–	119.34	4.16
Baseline	DFM (PSD)	–	94.08	4.06
Baseline	DFM (ESD)	–	68.11	3.79
Ours	Ours	0.50	45.82	3.6088
Ours	Ours	0.60	76.19	4.0841
Ours	Ours	0.65	98.96	4.3192
Ours	Ours	0.70	134.06	4.5716
Reference	LM1B test set	–	47.47	4.31

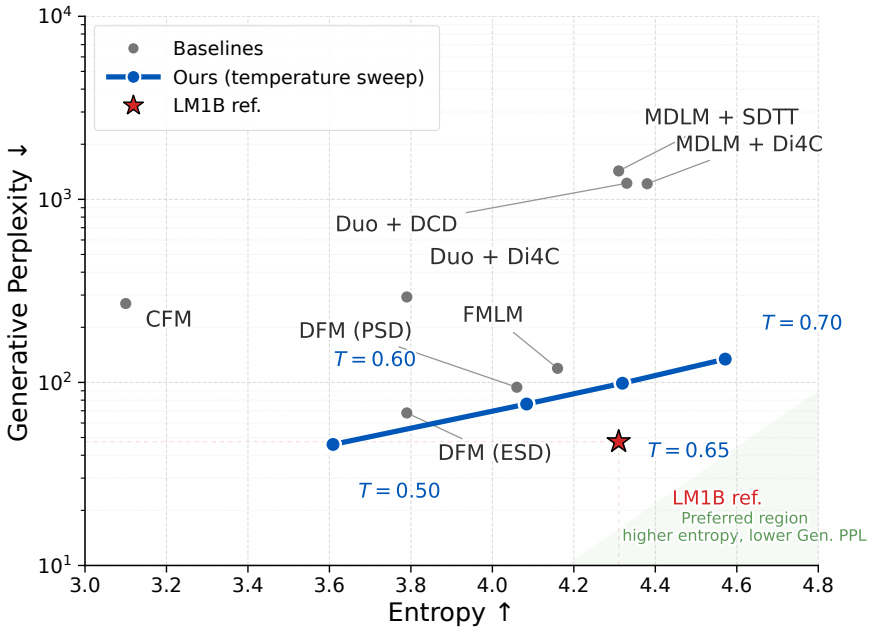

Figure 6: Entropy–perplexity tradeoff for one-step LM1B generation, reproduced from Figure 1. The appendix copy is included with the evaluation details so that the sampling settings and metric definitions can be read alongside the result.

G.4 Asset licenses and provenance

The main assets used in the experiments are the following: MNIST, which we cite as CC BY-SA 3.0; the DeepFlyBrain model, which the Kipoi model page lists as MIT; the Fly Brain / DeepFlyBrain article, which is CC BY 4.0; the associated Zenodo design package, which is distributed under an academic non-commercial software license; LM1B benchmark code/scripts, which are Apache-2.0, while the underlying corpus license is not explicitly stated on the official benchmark page; the GPT-2 model/tokenizer, which uses OpenAI's modified MIT license; and the Qwen2.5-0.5B model/tokenizer, which is Apache-2.0.

Table 7: LM1B configuration used for the reported one-step text result.

Item	Value
Dataset	LM1B (`dvruette/lm1b`, `~parquet` revision)
Tokenizer	Qwen2.5-0.5B tokenizer
Vocabulary size	151,936
Sequence length	128
Stage A encoder	Frozen Qwen2.5-0.5B, hidden size 896
Stage A projection dim	256
Stage A flow	RealNVP, 12 coupling layers, hidden dim 1024
Stage A optimizer / LR / WD	AdamW / 3×10^{-4} / 0.01
Stage A batch size / steps	128 / 200k
Stage B generator	Transformer, width 768, depth 12, heads 12, FFN 2048
Stage B optimizer / LR / WD	AdamW / 2×10^{-4} / 0.01
Stage B batch size / steps	32 / 400k
KD teacher	Frozen Qwen2.5-0.5B causal LM
KD weight / temperature	1.0 / 2.0
Gaussian-latent mixing prob.	0.30
Inference latent scale	0.90
Evaluation samples	1,024
Evaluation settings	temperature 0.8, top-p 0.95, GPT-2 Large scorer
Reported checkpoint	Stage B step 152,038

G.5 Compute and implementation notes

The MNIST and Fly Brain experiments were launched through single-node SLURM jobs requesting one NVIDIA H200 GPU, 32 GB host memory, 12 CPU cores, and a two-day wall-clock limit. The text experiments use the local training code and configuration files described above, but the current draft does not yet include a consolidated summary of text training wall-clock time or total compute. Across benchmarks, we use the default random seed 0 in the released scripts unless otherwise noted.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the paper's actual contributions: a two-stage model for one-step discrete generation and empirical results on MNIST-Binary, Fly Brain, and LM1B. The appendix supplies the expressivity argument for direct one-step token-wise decoders, and the paper also notes the main limitation that stronger iterative baselines remain better in harder settings (Sections 3, 4, Appendix C, and the Conclusion).

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper states that the model is competitive in the one-step regime but still trails the strongest iterative baselines on harder settings, and the Conclusion limits the scale of the current study to moderate-size models and benchmarks. The appendix also clarifies that the text benchmark uses a benchmark-specific evaluation pipeline and that the reported gains should be read in that context (Sections 3, 4, G.3).

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The theoretical result is the expressivity barrier for direct one-step token-wise decoders. The appendix states the assumptions, defines the factorized family, and gives a complete proof by both dimension counting and an explicit counterexample (Appendix C).

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The appendix specifies the datasets, splits, preprocessing, model components, training schedules, optimizers, learning rates, evaluation metrics, sampling settings, and reported checkpoints for the three main benchmarks (Appendix G, especially Sections G.1, G.2, and G.3).

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The paper provides detailed reproduction instructions, but the code and any released checkpoints will be made public only after acceptance rather than being open at submission time. We therefore describe a reproducible route, but not current open access to the full artifact set (Appendix G).

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The appendix gives the dataset splits, preprocessing, training schedules, optimizer settings, latent sizes, flow depths, generator widths, evaluation budgets, and checkpoint choices for all three benchmarks. The remaining implementation details are also summarized in tables for each benchmark (Appendix G).

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The Fly Brain table reports $\pm$ values for the main one-step result, but the paper does not yet define those error bars or report comparable uncertainty information for all main results. We therefore do not claim a full statistical-significance treatment in the current draft.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar rather than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: The appendix specifies the H200-based SLURM setup for MNIST and Fly Brain, but the text experiment section still lacks a consolidated wall-clock and total-compute summary. As written, the paper does not yet provide compute details at the level requested for every experiment (Appendix G.5).

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

Justification: The work uses standard public benchmarks and pretrained models, does not involve human subjects, and does not report any data collection or deployment practice that conflicts with the NeurIPS Code of Ethics. The paper's stated limitations and benchmark scope are consistent with the ethics guidance.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The checklist identifies the main positive and negative impacts: the method may reduce inference cost for discrete generation and help sequence-design workflows, but the same capability can also lower the cost of generating synthetic text or biological sequences for misuse. The paper frames the method as a generative modeling advance rather than a safety mechanism or a deployment system with built-in controls.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [No]

Justification: The paper does not describe gated release, usage restrictions, or safety filters for models or data. That is consistent with the current status of the project, which is a research paper rather than a deployed release.

Guidelines:

- The answer NA means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [No]

    Justification: The paper credits the benchmark sources and now records the licenses/terms of the major assets in Appendix G.4, but the LM1B corpus itself does not have a clearly specified upstream license on the official benchmark page. Because that uncertainty remains, we do not mark the licensing disclosure as fully complete.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not introduce a new dataset or benchmark package as a released asset. It introduces a model and reports experiments on existing public benchmarks.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not include crowdsourcing studies or other human-subjects experiments.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve human-subjects research, so IRB approval is not applicable.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method does not rely on an LLM as an original or non-standard component. The pretrained language models used in the text experiment are standard frozen components for representation extraction and evaluation, not the core methodological contribution.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.