SIMPLEX DIFFUSION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion models have revolutionized generative modeling for continuous data through the gradual refinement of a belief state. This iterative refinement has not yet carried over to *discrete* diffusion models, which discard uncertainty at intermediate steps through categorical sampling (*information collapse*). We propose *Simplex Diffusion Models* (SDMs), a framework that lifts the diffusion process to the probability simplex to represent beliefs over categories. SDMs admit probability paths with closed-form reverse transitions and can be trained with a simple cross-entropy loss. Contrary to earlier proposals such as Dirichlet Flow Matching which requires integrating an ordinary differential equation, we introduce a DDIM-like sampler with a tunable level of stochasticity. Because SDMs operate on samples on the simplex, they can carry uncertainty across denoising steps, which mitigates information collapse. On OpenWebText, SDMs are competitive with strong discrete diffusion baselines, achieving 17.0 GenPPL at 5.46 unigram entropy in 64 sampling steps, close to real validation data. Even without self-conditioning (SC), SDMs outperform masked and uniform diffusion (with SC or predictor-corrector sampling) on code generation (TinyGSM, $T = 0.1$; 49.0% vs. 45.8%). Distilled down to 8 steps, SDMs solve 32.1% of GSM8K problems, more than distilled discrete diffusion models with 128 steps (21.4%).

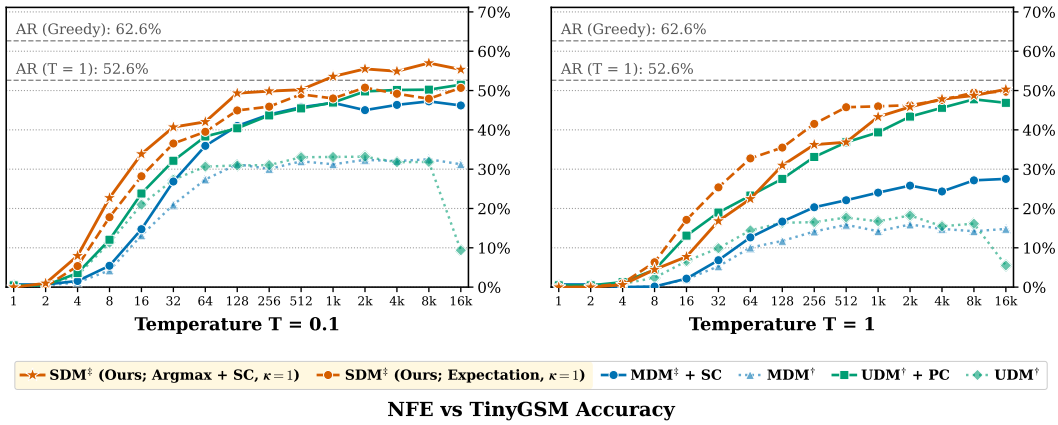

Figure 1: **SDMs outperform masked and uniform diffusion on TinyGSM.** *Left:* Accuracy as a function of the number of function evaluation (NFE) at low temperature ($T = 0.1$). *Right:* Accuracy against NFEs at $T = 1$. Without self-conditioning (SC), SDMs with the expected embedding outperform masked diffusion with SC (MDM + SC) and uniform diffusion with predictor-corrector sampling (UDM + PC). At 512 NFE, SDMs reach 49.0% at $T = 0.1$, vs. 45.8% for MDM‡ + SC, and 45.8% at $T = 1$, vs. 36.8% for UDM† + PC. With SC, SDM (Argmax + SC) is the best diffusion variant at $T = 0.1$ from 32 NFE onwards, and solves 57% of problems with 8k NFEs. Autoregressive models with greedy decoding remain the strongest (62.6%). All SDMs use churn $\kappa = 1$ and the adaptive sampling time grid (Section O.1), and we compare with the best baseline configurations (Section 5.2). †/‡: trained with the uniform/adaptive time sampler. The accuracy drop for UDM at 16k is consistent with typical floating-point rounding errors, which can accumulate at very small step sizes (Karras et al., 2022).

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

1 INTRODUCTION

Denoising diffusion models (Ho et al., 2020; Song et al., 2021b) are state-of-the-art generative models in continuous domains (Ho et al., 2022; Rombach et al., 2022; Ramesh et al., 2022; Watson et al., 2023). However, a vast portion of real-world data—including text, biological sequences, and graph structures—is discrete, and extending diffusion to those modalities remains challenging (Austin et al., 2021; Hoogeboom et al., 2021; Campbell et al., 2022).

Existing diffusion models for categorical data generally fall into three categories. First, *categorical diffusion models* such as masked or uniform diffusion models (Sahoo et al., 2024; Shi et al., 2024; Ou et al., 2025) define the corruption process directly on sequences of discrete tokens. While theoretically sound, they suffer from *information collapse*. Indeed, as intermediate states are discrete, the denoiser's uncertainty is discarded at each update. Hence, these models rely on self-conditioning (Chen et al., 2022) or loopholing (Jo et al., 2026) to carry beliefs across sampling steps.

Second, *continuous relaxations* diffuse one-hot vectors (Lee et al., 2026; Roos et al., 2026; Potaptchik et al., 2026), learned embeddings (Dieleman et al., 2022; Gulrajani & Hashimoto, 2023; Deschenaux & Gulcehre, 2026; Chemseddine et al., 2026; Yang et al., 2026), or frozen embeddings (Hu et al., 2026; Shen et al., 2026). However, the identity of the clean category tends to be destroyed abruptly, within a short window during the forward process, and this worsens as the dimension of the encoding grows (Pynadath et al., 2026a; Shabalin et al., 2026).

Finally, a third line of work diffuses directly on the probability simplex, whose elements are distributions over categories (Richemond et al., 2022; Cheng et al., 2024; Davis et al., 2024; Stark et al., 2024; Cheng et al., 2025; Williams et al., 2026; Boget & Kalousis, 2026; Chandra et al., 2026). However, they typically sample by numerically integrating an ODE or SDE: Dirichlet Flow Matching (DFM; Stark et al., 2024), the closest to our work, scales poorly to large vocabularies (Table 4). While the concurrent Simplax model (Sakurai et al., 2026) does not rely on numerical integration, their sampling procedure operates on categorical samples rather than the continuous representation itself.

Contributions. We propose Simplex Diffusion Models (SDMs), whose intermediate states are distributions over tokens.

1. Similar to DFM, SDMs use a Dirichlet forward process, but parameterized so that its mean and concentration are set separately. This path admits closed-form reverse transitions, inducing a DDIM-style sampler (Song et al., 2021a), with a *churn* parameter that controls the sampling stochasticity. SDMs are trained with a simple cross-entropy objective. Because SDMs operate on samples on the simplex, they carry uncertainty across denoising steps, mitigating information collapse.

2. We show that SDMs unify discrete and continuous diffusions. A category sampled from the diffused simplex state has the same marginal distribution as if it were corrupted by discrete diffusion (Proposition 3.1). Viewing the inverse Dirichlet concentration as a *temperature*, SDMs reduce to discrete diffusion at high temperature, and follow a deterministic path with Gaussian fluctuations at low temperature (Proposition 4.1 and Section E.7). Finally, in the zero-churn regime, the bridge admits a continuous-time representation driven by a pure-jump additive process (Section H).

3. Empirically, even without self-conditioning (SC), **SDMs outperform masked and uniform diffusion** (using either SC or predictor-corrector sampling) on code generation (TinyGSM, $T = 0.1$; 49.0% vs. 45.8%; Figure 1). Distilled down to 8 steps, SDMs solve 32.1% of GSM8K problems, more than distilled discrete diffusion models with 128 steps (21.4%; Table 5). On Sudoku, SDMs with SC match the best discrete diffusion model (99.1% vs. 99.3%). On OpenWebText, SDMs are competitive with UDM and MDMs with all those methods achieving a GenPPL/Entropy frontier that matches real data after logit shaping interventions. This result calls into question the validity of OpenWebText in assessing the unconditional text generation capabilities of small scale models. We also validate the flexibility of our approach by achieving competitive performance on quality and diversity metrics for molecule generation, see Section P.7.

2 BACKGROUND

Notation. Let $\mathcal{X} = \{1, \dots, N\}$ be the finite categorical vocabulary. Let Δ_N denote the probability simplex on $\mathbb{R}^N$, and $\mathbf{1} = (1, \dots, 1)^\top$ be the all-ones vector. For $i \in \mathcal{X}$, let $e_i \in \Delta_N$ be the standard basis (one-hot) vector corresponding to state i, and let $\odot$ denote the Hadamard product. Let $t \mapsto \alpha_t$ be non-increasing on $[0, 1]$ with $\alpha_0 = 1$, $\alpha_1 = 0$, and $0 < \alpha_t < 1$, for $t \in (0, 1)$. We denote by $\pi = (\pi_1, \dots, \pi_N) \in \Delta_N$ a reference prior distribution. We denote by Dir, Beta, Gamma, Ber and Cat the Dirichlet, Beta, Gamma, Bernoulli and Categorical distributions respectively. Basic facts about these distributions are recalled in Section A.

Discrete Diffusion Models. Discrete Diffusion models (DDMs; Sohl-Dickstein et al., 2015; Austin et al., 2021; Campbell et al., 2022; Sahoo et al., 2024; Shi et al., 2024; Ou et al., 2025) are generative models over discrete spaces. DDMs define a conditional corruption process

$$p_{t|0}(x_t|x_0) = \mathrm{Cat}\left(x_t; \alpha_t e_{x_0} + (1 - \alpha_t)\pi\right), \tag{1}$$

that induces marginal distributions $(p_t)_{t\in[0,1]}$, given by $p_t(x_t) = \sum_{x_0} p_{t|0}(x_t|x_0)p_0(x_0)$, bridging $p_0 = p_{\text{data}}$ to $p_1 = \pi$. The generative process is defined in terms of transitions $p_{s|t}$ with $s \leqslant t$. The transition $p_{s|t}$ is *compatible* if $p_s(x_s) = \sum_{x_t} p_{s|t}(x_s|x_t)p_t(x_t)$. Thus, one can generate $x_0 \sim p_{\text{data}}$ by applying $p_{s|t}$ for n iterations, starting from $x_1 \sim p_1$. For a time grid $0 = t_0 < t_1 < \ldots < t_n = 1$,

$$p_0(x_0) = \sum_{(x_{t_1}, \dots, x_{t_{n-1}}, x_1)} p_1(x_1) \prod_{i=1}^n p_{t_{i-1}|t_i}(x_{t_{i-1}}|x_{t_i}).$$

While using *marginal* transitions $p_{s|t}(x_s|x_t) = \sum_{x_0} p_{s|0,t}(x_s|x_0, x_t)p_{0|t}(x_0|x_t)$ is possible, recent work instead defines *bridge* (approximate) transitions $\hat{p}^\theta_{s|t}(x_s|x_t) := p_{s|0,t}(\cdot|\mathbf{x}_\theta(t, x_t), x_t)$, where $\mathbf{x}_\theta(t, x_t) : [0, 1] \times \mathcal{X} \to \Delta_N$ is learned (Gourevitch et al., 2026). One obtains θ by maximizing an expected Evidence Lower Bound (ELBO). See Section N.1 for more details.

Dirichlet Flow Matching. Dirichlet Flow Matching (DFM; Stark et al., 2024) extends Flow Matching (FM; Lipman et al., 2023) to the simplex Δ_N to model categorical data. The data distribution $p_{\text{data}} = (p_{\text{data},1}, \cdots, p_{\text{data},N})$ on $\mathcal{X}$ is first lifted to an atomic measure on the simplex which we also denote as p_{data} in a slight abuse of notation for $P \in \Delta_N$

$$p_{\text{data}}(P) = \sum_{i=1}^N p_{\text{data},i}\delta_{e_i}(P). \tag{2}$$

Stark et al. (2024) define a probability path $(p_t)_{t\in[0,1]}$ between the distribution $p_0 = p_{\text{data}}$ (at time $t = 0$) and the uniform Dirichlet prior $p_1 = \mathrm{Dir}(\cdot; \mathbf{1})$ (at $t = 1$) using

$$p_t(P_t) = \sum_{P_0} p_{t|0}(P_t \mid P_0)p_{\text{data}}(P_0) \quad \text{where} \quad p_{t|0}(P_t|P_0) = \mathrm{Dir}\big(P_t; \mathbf{1} + h_t P_0\big),$$

with $t \mapsto h_t$ a decreasing function with $h_1 = 0$ and $\lim_{t\to 0} h_t = \infty$. A velocity field $u_t(P_t)$ ensuring (approximately) that $\mathrm{d}P_t = u_t(P_t)\mathrm{d}t$ satisfies $P_t \sim p_t$ for $P_1 \sim p_1$, hence $P_0 \sim p_{\text{data}}$, is then learned through cross-entropy; see Section N.4 for details.

3 SIMPLEX DIFFUSION

3.1 PROBABILITY PATH AND BRIDGE

As in Dirichlet flow matching (Stark et al., 2024), we build a probability path on the simplex by considering $P_0 \sim p_{\text{data}}$, see (2), and Dirichlet distributions to define $p_{t|0}(P_t|P_0)$.

Probability Path. Let $t \mapsto c_t$ be a positive function and $t \mapsto \alpha_t$ a non-increasing function such that $\alpha_0 = 1$ and $\alpha_1 = 0$. We consider the corruption mechanism defined by

$$p_{t|0}(P_t|P_0) = \mathrm{Dir}(P_t; \beta_t(P_0, \pi)), \qquad \beta_t(P_0, \pi) = c_t(\alpha_t P_0 + (1 - \alpha_t)\pi), \tag{3}$$

where we recall that $\pi \in \Delta_N$ is a reference prior distribution on $\mathcal{X}$ chosen by the user[1]. In particular $p_{1|0}(P_1|P_0) = \mathrm{Dir}(P_1; c_1\pi)$ is independent of P_0. We have $\mathbb{E}[P_t|P_0] = (1 - \alpha_t)\pi + \alpha_t P_0 := \bar{P}_t$ and $\mathrm{Cov}[P_t|P_0] = (\mathrm{diag}(\bar{P}_t) - \bar{P}_t\bar{P}_t^\top)/(c_t + 1)$. We refer to c_t as the concentration parameter.

[1] In practice, we will use a distribution π with full support on $\mathcal{X}$. Otherwise, this would mean that for data tokens x_0 not in the support of π, any sample P_t from $p_{t|0}(P_t|P_0 = e_{x_0})$ would reveal x_0 almost surely for all $t < 1$. The same argument rules out using a simplex version of masked diffusions.

This forward model on the simplex induces a forward model on the discrete state-space commonly used in discrete diffusions/flow matching (Campbell et al., 2024; Sahoo et al., 2024; Shi et al., 2024).

Proposition 3.1 (Induced forward): *Let $t \in [0,1]$ and P_t satisfying (3). In addition, let x_t be such that $p_{t|0}(x_t|P_t, P_0) = P_{t,x_t}$. Then, recalling that $P_0 = e_{x_0}$, we have that*

$$p_{t|0}(x_t|P_0) = \alpha_t \delta_{x_0}(x_t) + (1-\alpha_t)\pi_{x_t}.$$

We now propose an alternative representation of (3) enabling us to sample easily from $p_{t|0}(P_t \mid P_0)$.

Proposition 3.2 (Interpolation path): *Let $t \in [0,1]$. Let $W_t \sim \mathrm{Beta}\left(c_t\alpha_t, c_t(1-\alpha_t)\right)$ and let $V_t \sim \mathrm{Dir}(c_t(1-\alpha_t)\pi)$ be independent of W_t. Define*

$$P_t = W_t P_0 + (1-W_t)V_t. \tag{4}$$

Then the conditional density of P_t given P_0 is given by $p_{t|0}(P_t \mid P_0) = \mathrm{Dir}\left(P_t; \beta_t(P_0, \pi)\right)$.

Equation (4) shows that, for $t \in (0,1)$, P_t is a random convex combination (with interpolation weight $W_t \in (0,1)$) of a data sample P_0 and a noise sample V_t.

The concentration parameter c_t acts inversely to temperature: larger values concentrate probability mass around the mean, whereas smaller values push it toward the simplex vertices. Specializing momentarily to $c_t = \varepsilon/(1-\alpha_t)$ for $t > 0$, we interpret ε as an *inverse temperature*. In Section 4, we formally prove that as $\varepsilon \to 0$ (high temperature), the conditional distribution of $P_t \mid P_0$ concentrates entirely on the vertices. Hence *discrete diffusion models emerge as a limiting case of Simplex Diffusion Models*. Conversely, as $\varepsilon \to \infty$, the distribution concentrates on the mean, $\alpha_t P_0 + (1-\alpha_t)\pi$. Figure 6 illustrates these regimes across varying $\varepsilon > 0$ and $t \in [0,1]$.

Backward Bridges. We now return to a general concentration schedule $(c_t)_{t\in[0,1]}$. In order to obtain a generative model, in the spirit of DDIM (Song et al., 2021a), we need to identify a conditional distribution $p_{s|0,t}$ satisfying the following backward compatibility condition for any $s, t \in [0,1]$ with $s < t$ and $P_s \in \Delta_N$

$$p_{s|0}(P_s|P_0) = \int_{\Delta_N} p_{s|0,t}(P_s|P_0, P_t) p_{t|0}(P_t|P_0) \mathrm{d}P_t. \tag{5}$$

Leveraging classical properties of Dirichlet distributions, we propose such a backward bridge model. For $t \in (0,1]$, write $\beta_t(P_0,\pi) = a_t P_0 + b_t \pi$ so that $a_t = c_t\alpha_t$ and $b_t = c_t(1-\alpha_t)$. The quantity b_t can be thought of as the concentration associated to π at time t. Let $r_{s,t} = \min\{1, b_s/b_t\}$.

Proposition 3.3 (Simplex Transition): *Let $\kappa \in [0,1)$ and $s, t \in (0,1]$ with $s < t$. Denote $\rho^\kappa_{s,t} = (1-\kappa) r_{s,t}$. For any $P_0, P_s, P_t \in \Delta_N$, consider $P^\kappa_{s,t} = (P^\kappa_{s,t,1}, \cdots, P^\kappa_{s,t,N})$, where*

$$P^\kappa_{s,t,i} = \frac{B_i P_{t,i}}{\sum_{j=1}^N B_j P_{t,j}} \tag{6}$$

with independent $B_i \sim \mathrm{Beta}\left(\rho^\kappa_{s,t}\beta_t(P_0,\pi)_i, (1-\rho^\kappa_{s,t})\beta_t(P_0,\pi)_i\right)$, for $i = 1, \cdots, N$. Moreover, let $W^\kappa_{s,t}$ and $V^\kappa_{s,t}$ be given by

$$W^\kappa_{s,t} \sim \mathrm{Beta}\left(\rho^\kappa_{s,t} c_t,\ c_s - \rho^\kappa_{s,t} c_t\right), \qquad V^\kappa_{s,t} \sim \mathrm{Dir}\left(\beta_s(P_0,\pi) - \rho^\kappa_{s,t}\beta_t(P_0,\pi)\right), \tag{7}$$

the random variables $(B_i, W^\kappa_{s,t}, V^\kappa_{s,t})$ being all independent. Finally let

$$P_s = W^\kappa_{s,t} P^\kappa_{s,t} + (1 - W^\kappa_{s,t}) V^\kappa_{s,t}. \tag{8}$$

Then the induced transition kernel denoted $p_{s|0,t}$ satisfies the compatibility condition (5). For the limiting case $\kappa = 1$, we define $p_{s|0,t}(P_s|P_0, P_t) = p_{s|0}(P_s|P_0)$.

The simplex transition outputs a new state P_s in (8) which is a random convex combination of an innovation term $V^\kappa_{s,t}$ and a thinned version $P^\kappa_{s,t}$ of the current state P_t, with interpolation weight

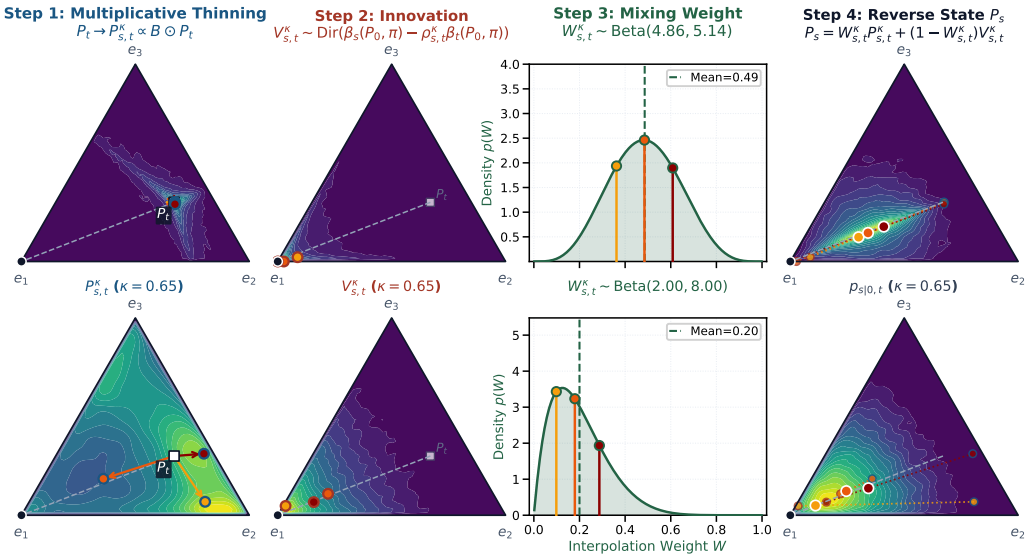

Figure 2: We describe the different stages of one backward step of the inference process from left to right. Here we assume that at the current step, $P_0 \sim \hat{P}_\theta(t, P_t)$ is such that $P_0 = e_1$. The first line corresponds to a low churn $\kappa = 0.15$ and the second line corresponds to a high churn of $\kappa = 0.65$. We consider an interpolation schedule $\alpha_t = 1 - t$ and a concentration schedule $c_t = \varepsilon/(1 - \alpha_t)$ with $\varepsilon = 4.0$. In what follows $s = 0.40$ and $t = 0.70$. In each panel the dots with colors yellow, orange, red correspond to three different samples from the current step. In the first step, using the churn $\kappa \in [0, 1]$, we sample $P^\kappa_{s,t}$. This corresponds to a *multiplicative* noising step around P_t, with a higher churn corresponding to a higher noise. In the second step, we sample from the *innovation* variable $V^\kappa_{s,t}$, which is a Dirichlet random variable concentrated around the prediction P_0. In the third step we sample from a Beta random variable $W^\kappa_{s,t}$ which corresponds to the mixing weight in the interpolation between the thinned variable $P^\kappa_{s,t}$ and the innovation $V^\kappa_{s,t}$. Note that for high churns the mixing weight $W^\kappa_{s,t}$ is close to 0, meaning that we favor the innovation random variable $V^\kappa_{s,t}$. Finally in the fourth step, we visualize the interpolation between the innovation $V^\kappa_{s,t}$ and the thinned variable $P^\kappa_{s,t}$ using the sampled Beta mixing weight $W^\kappa_{s,t}$.

$W^\kappa_{s,t}$. Indeed for $P_t \sim \mathrm{Dir}(\beta_t(P_0, \pi))$, one can show that $P^\kappa_{s,t} \sim \mathrm{Dir}(\rho^\kappa_{s,t}\beta_t(P_0, \pi))$. The *churn* parameter $\kappa \in [0, 1]$ controls how much additional randomness is introduced in the reverse transition at inference time. We present those different steps in Figure 2.

When $\kappa = 0$, we retain the largest fraction of the current state that is compatible with the prescribed concentration schedule. In this case $\rho^0_{s,t} = r_{s,t}$, $W^0_{s,t} \sim \mathrm{Beta}\left(r_{s,t}c_t,\, c_s - r_{s,t}c_t\right)$ and $V^0_{s,t} \sim \mathrm{Dir}\left(\beta_s(P_0, \pi) - r_{s,t}\beta_t(P_0, \pi)\right)$. The corresponding update is

$$P_s = W^0_{s,t}P^0_{s,t} + (1 - W^0_{s,t})V^0_{s,t}. \tag{9}$$

For the concentration $c_t = \varepsilon/(1 - \alpha_t)$, we have $b_t = c_t(1 - \alpha_t) = \varepsilon$, hence $r_{s,t} = 1$. In that case $P^0_{s,t} = P_t$ and $V^0_{s,t} = e_{x_0}$, so (9) reduces to $P_s = W^0_{s,t}P_t + (1 - W^0_{s,t})e_{x_0}$. If $\varepsilon \to +\infty$, (9) becomes

$$P_s = \frac{1 - \alpha_s}{1 - \alpha_t}P_t + \frac{\alpha_s - \alpha_t}{1 - \alpha_t}e_{x_0},$$

which is the exact deterministic DDIM rule. We refer to Proposition 4.1 for a theoretical analysis of those temperature limits and to Section F for a full general framework encompassing those DDIM-like update rules. Equipped with a compatible backward transition mechanism, we can now define our training and inference algorithms.

3.2 Training and Inference

Define $0 = t_0 < \cdots < t_M = 1$. At inference time, we will follow a DDIM approach (Song et al., 2021a): ideally we would generate data by starting from $P_{t_M} \sim p_1(\cdot)$ and $P_{t_k} \sim p_{t_k|t_{k+1}}(\cdot|P_{t_{k+1}})$

for $k = M-1, ..., 0$, where for $0 \leqslant s < t \leqslant 1$

$$p_{s|t}(P_s|P_t) = \int p_{s|0,t}(P_s|P_0, P_t) p_{0|t}(P_0|P_t) \mathrm{d}P_0, \tag{10}$$

with $p_{0|t}(P_0|P_t)$ the posterior distribution of P_0 given P_t. Here $p_1(P_1) = \int p_{1|0}(P_1 \mid P_0) p(P_0) \mathrm{d}P_0 = \mathrm{Dir}(P_1; c_1\pi)$. If we had access to the true backward transitions (10), then this procedure would return samples from P_0, hence from the data distribution. For each time step $k = M-1, \ldots, 0$, we sample $P_{t_k} \sim p_{t_k|0,t_{k+1}}(\cdot \mid P_0, P_{t_{k+1}})$ where $P_0 = e_{\widetilde{x}_0}$ for $\widetilde{x}_0 \sim \hat{P}_\theta(t_{k+1}, P_{t_{k+1}})$; i.e., we approximate $p_{0|t}(P_0|P_t)$ by $p^\theta_{0|t}(P_0|P_t) = \sum_{i=1}^N \hat{P}_{\theta,i}(t, P_t)\delta_{e_i}(P_0)$. To learn the denoiser $\hat{P}_\theta$ from P_t, we minimize the cross entropy loss

$$\mathcal{L}(\theta) = -\mathbb{E}_{t\sim w(t), x_0 \sim p_{\text{data}}, P_0 = e_{x_0}, P_t \sim \mathrm{Dir}(\beta_t(P_0,\pi))}[\log\left(\hat{P}_\theta(t, P_t)_{x_0}\right)], \tag{11}$$

as $\sum_{i=1}^N P_{0,i} \log\left(\hat{P}_\theta(t, P_t)_i\right) = \log\left(\hat{P}_\theta(t, P_t)_{x_0}\right)$. In Section G, we show that this loss corresponds to a negative evidence lower bound (ELBO) if $w(t)$ is selected as the uniform distribution on the set $\{t_1, ..., t_M\}$. Given a token embedding matrix $E \in \mathbb{R}^{N\times d}$, the denoiser $\hat{P}_\theta(t, P_t)$ embeds the simplex state P_t either via its expected embedding $P_t^\top E$ or, to avoid this dense matrix product on large vocabularies, via the most likely token embedding $E_{\mathrm{argmax}\, P_t}$; see Section O.2 for details. We summarize the training of Simplex Diffusion Models in Algorithm 1 and inference in Algorithm 2. These algorithms are described for a single token for the sake of simplicity. The extension to multiple tokens is described in Section C.1.

4 Concentration Properties

4.1 Temperature Limits

In Section 3, we briefly discussed how the distribution of P_t behaves in both the high and low temperature settings $c_t = \varepsilon/(1-\alpha_t)$. We show it here formally.

Proposition 4.1 (Temperature limits): *Let $c_t = \varepsilon/(1-\alpha_t)$ and assume that $P_0 = e_{x_0}$.*

(i) High-temperature limit. *Fix $t \in (0,1)$. As $\varepsilon \to 0$,*

$$P_t \xrightarrow{d} B_t P_0 + (1 - B_t)V,$$

where $B_t \sim \mathrm{Ber}(\alpha_t)$ and $V \sim \sum_{i=1}^N \pi_i \delta_{e_i}$, independently. Thus, P_t is supported on the vertices of Δ_N. Moreover, fix $0 < s < t < 1$ and consider the $\kappa = 0$ backward bridge (9). For every fixed $P_t \in \Delta_N$, as $\varepsilon \to 0$,

$$P_s \xrightarrow{d} W^0_{s,t} P_t + \left(1 - W^0_{s,t}\right) e_{x_0}, \qquad W^0_{s,t} \sim \mathrm{Ber}\left(\frac{1-\alpha_s}{1-\alpha_t}\right).$$

Thus, the limiting transition remains supported on simplex vertices.

(ii) Low-temperature limit. *Fix $t \in (0,1)$. As $\varepsilon \to \infty$,*

$$P_t \xrightarrow{\mathbb{P}} \bar{P}_t := \alpha_t P_0 + (1-\alpha_t)\pi. \tag{12}$$

Proposition 4.1 shows that the inverse temperature interpolates between two very different regimes. At high temperature, the simplex-valued state collapses onto the vertices, and with $\kappa = 0$ it reduces the reverse bridge to a transition between the current and clean vertices. Thus the discrete-diffusion interpretation is recovered for both the probability path and the backward bridge. At low temperature, by contrast, the randomness disappears and the forward state interpolates deterministically between P_0 and π. However, the deterministic limit in (12) only describes the first-order behavior as $\varepsilon \to \infty$. In Section E.7 we identify a Gaussian fluctuation limit result, drawing connections between the limit $\varepsilon \to +\infty$ and Gaussian diffusion models.

4.2 Concentration Schedule via Normalized Variance

While Proposition 4.1 establishes that SDMs interpolate between discrete diffusion (high temperature) and the mean interpolation (low temperature), it remains unclear how to select $t \mapsto c_t$ in practice. Here we focus on the setting where $\pi = \mathbf{1}/N$, i.e., the uniform prior. Rather than tuning an unbounded $c_t \in (0, \infty)$, we define c_t in terms of the fraction $t \mapsto \nu_t \in (0, 1)$ of the maximal total variance (TVar), defined by

$$\mathrm{TVar}(t) := \mathrm{Tr}\left[\mathrm{Cov}\left(P_t \mid P_0\right)\right] = \frac{1}{c_t + 1}\left[1 - \|\alpha_t P_0 + (1 - \alpha_t)\pi\|_2^2\right] = \frac{N-1}{N}\frac{1-\alpha_t^2}{c_t+1}.$$

Because $c_t \in (0, \infty)$, the total variance is bounded above by $\frac{N-1}{N}(1 - \alpha_t^2)$. Setting $\nu_t := \frac{1}{c_t+1}$ (equivalently $c_t = \frac{1}{\nu_t} - 1$), the total variance is exactly the fraction ν_t of this bound. In practice, we experiment with two variants (one constant, one piecewise linear):

$$\nu_t^{\text{cst.}} = \nu_0, \qquad \nu_t^{\text{cst.-lin}} = \begin{cases} \nu_0, & \text{if } t < \ell, \\ \nu_0 + \frac{t-\ell}{1-\ell}(\nu_1 - \nu_0), & \text{if } t \geqslant \ell. \end{cases}$$

This parameterization is more interpretable than the raw value of c_t. At $t = 1$, setting $c_1 = N$ induces a uniform distribution on the simplex, and a larger c_1 concentrates around π. However, for $t < 1$, the mean $\alpha_t P_0 + (1 - \alpha_t)\pi$ moves towards the vertex P_0, and a large c_t collapses P_t onto the mean, which reveals the clean token almost deterministically. In contrast, TVar operates on a bounded interval with clear extrema: it approaches its upper bound when P_t collapses onto the vertices, and vanishes when P_t collapses onto its mean ($c_t \to \infty$).

5 Experiments

We compare SDMs with autoregressive models (AR), masked and uniform diffusion models (MDMs, UDMs), and Flow Language Models (FLM) and Hyperspherical Flows (𝕊-FLM). We evaluate on Sudoku (Alp, 2024; Ben-Hamu et al., 2025; Kim et al., 2026b), code generation on TinyGSM (Liu et al., 2023a; Kim et al., 2026a), language modeling on OpenWebText (OWT) (Gokaslan & Cohen, 2019), language understanding; Section O.7, and molecule generation following GenMol (Lee et al., 2025; Section O.8). Prior work mostly compares models by Generative Perplexity (Gen. PPL) on OWT, which does not always correlate with downstream performance such as functional correctness in code (Deschenaux & Gulcehre, 2024; Feng et al., 2025; Franca & Tong, 2026; Veličković et al., 2026). Thus, we primarily compare models on TinyGSM.

Table 1: **Accuracy (%)** on Sudoku in 180 steps, varying the sampling schedule. †trained with uniform time sampler, ‡trained with adaptive time sampler. For each column, we underline the best and **bold** the highest accuracy for continuous methods. We show SDM with normalized variance $\nu_t^{\text{cst.-lin.}}$, with $\nu_0 = 0.4$, $\nu_1 = 0.75$, $\ell = 0.8$. Values are mean ± std over 5 seeds.

Model	Linear	Cosine
Discrete		
AR (Greedy)	$3.1_{\pm 0.5}$	–
MDM‡	$69.4_{\pm 4.8}$	$69.7_{\pm 4.0}$
MDM‡ (+SC)	$94.7_{\pm 0.8}$	$\underline{99.3}_{\pm 0.2}$
UDM‡	$82.1_{\pm 1.7}$	$82.4_{\pm 1.7}$
UDM† (+SC)	$97.8_{\pm 1.7}$	$98.1_{\pm 1.2}$
Continuous		
FLM‡	$75.3_{\pm 3.4}$	$74.9_{\pm 3.5}$
𝕊-FLM‡	$87.3_{\pm 1.5}$	$87.3_{\pm 1.7}$
SDM† ($\kappa = 1.0$)	$88.4_{\pm 2.2}$	$88.9_{\pm 2.4}$
SDM† (+SC; $\kappa = 1.0$)	$\underline{\mathbf{99.1}}_{\pm 0.2}$	$\mathbf{98.7}_{\pm 0.3}$

5.1 Reasoning on Sudoku

Experimental Setup. We train on 200k grids with 30/81 clues revealed and evaluate on 5k unseen examples. We train a modified Diffusion Transformer (DiT) (Peebles & Xie, 2023) as in Lou et al. (2024) with embedding dimension 512, 8 layers and 8 attention heads (28.6M parameters) for 50k steps. We use 180 sampling steps for the diffusion variants. See Section O.4 for further details.

Results. SDMs are the strongest continuous method on Sudoku and match the discrete diffusion models (Table 1). Without self-conditioning (SC), SDMs outperform MDMs and UDMs with ancestral sampling and FLMs, and match 𝕊-FLMs. With SC, SDMs reach 99.1%, outperforming the continuous baselines and within 0.2 points of the best result overall. SDMs also outperform Dirichlet Flow Matching (DFM) by a wide margin (76.7%; Table 3). Setting $\kappa = 1$ works best. Without

SC, increasing κ from 0 to 1 improves the accuracy from 73.9% to 88.4% (Tables 21 to 23). With SC, SDMs perform similarly across churns. Training with the adaptive time sampler (Section O.1) generally improves MDMs, UDMs, and SDMs at low churn.

5.2 Code Generation on TinyGSM

Experimental Setup. We train on TinyGSM, a dataset of 11.8M synthetic math word problems with executable Python solutions, and evaluate on the GSM8K test set by executing one generated solution per problem. We tokenize with the SmolLM tokenizer, and use a context length of 512. We train a 12-layer DiT with hidden dimension 768 and 12 attention heads (167.9M parameters) for 250k steps ($\approx 77\%$ for SC variants, to match the training FLOPs; Section N.3) with a batch size of 512. See Section O.5 for further details. FLM, $\mathbb{S}$-FLM and DFM perform worse than the discrete baselines on TinyGSM, so we defer them to Section P.

Results. We compare SDMs against MDMs and UDMs with the ancestral sampler, with a Predictor-Corrector (PC) sampler (Section N.2) and with Self-Conditioning (SC; Section N.3). Since the expected embedding is costly with a large vocabulary, we also train SDMs that embed only the token $\operatorname{argmax} P_t$, optionally combined with SC (Argmax + SC; Section O.2). With the expected embedding and without SC, SDMs outperform all diffusion baselines at 512 steps (Figure 1), reaching 45.8% at $T = 1$ (vs. 36.8% for UDM + PC) and 49.0% at $T = 0.1$ (vs. 45.8% for MDM + SC). With Argmax + SC, SDMs perform best at $T = 0.1$ from 32 function evaluations (NFE) onward and reach 57.0% at 8k NFE (50.2% at 512 NFE).

Time Schedule, Churn and Blockwise Generation. We train with an adaptive time sampler that upweights noise levels where the model learns most, using a density fitted to the training loss (Section O.1), and derive the sampling time grid from the same density (Section O.1, Dieleman et al., 2022; Raya et al., 2026). With the expected embedding, this adaptive schedule improves SDMs ($\nu_0 = 0.2$, $\nu_1 = 0.5$, $\ell = 0.2$, $\kappa = 1$) over the linear grid by 9–11 points at 512 steps and 18–19 points at 64 steps; it does not help the Argmax inputs (e.g. Argmax + SC at $T = 1$ and 512 steps: 40.3% adaptive vs. 43.7% linear for $\nu_0 = 0.2$, $\nu_1 = 0.75$, $\ell = 0.2$) (Tables 29 to 32). The adaptive time sampler can improve MDM + SC but not UDM + PC, so we report MDM + SC trained with the adaptive time sampler and UDM + PC trained with the uniform time sampler (Section P.2). As on Sudoku, SDMs perform best with $\kappa = 1$. Finally, full-sequence SDMs outperform a block-autoregressive variant (Han et al., 2023; Arriola et al., 2025) with block size 32 at matched NFE, despite its stronger left-to-right bias (Figure 12).

Distillation and Auto-Guidance. We distill the SDM with the expected embedding (no SC; $\nu_t^{\text{cst.-lin.}}$ with $\nu_0 = 0.2$, $\nu_1 = 0.5$, $\ell = 0.2$) into a few-step generator (Section J). With 8 NFEs, the distilled SDM solves 32.1% of the problems at $T = 1$, more than IDLM (Li et al., 2026) with 128 NFEs (21.4%; Table 5) and the undistilled SDM with 8 NFEs ($\approx 6\%$; Figure 13). The accuracy improves up to $\approx 40\%$ at 128 NFEs and then plateaus. Distillation reduces the diversity, which we measure using the *Abstract Syntax Trees* (AST) of the generated programs (Section O.5). The AST diversity score decreases from 35.3 for the undistilled SDM with 512 NFEs (40.0 with 8 NFEs) to 17.5 for the distilled SDM with 8 NFEs at $T = 1$ (higher is more diverse; Table 33). See Section P.3 for more details. Finally, guiding SDMs away from a weaker checkpoint (Auto-Guidance; Karras et al., 2024) improves accuracy in all settings we tested, mostly in the low churn and high temperature regime (Section P.4).

5.3 Language Modeling on OpenWebText (OWT)

Experimental Setup. We train on OWT (Gokaslan & Cohen, 2019) for unconditional language modeling, and evaluate the generation quality via Generative Perplexity (GenPPL) under a pre-trained GPT-2-large model (Radford et al., 2019) alongside unigram entropy (H_1). We tokenize with the GPT-2 tokenizer, and use a context length of 1024. We train a 12-layer DiT with hidden dimension 768 and 12 attention heads (167.9M parameters) for 1M steps with a batch size of 512.

Results. We show that after introducing several improvements benefiting most discrete diffusion models, namely MDMs, UDMs and SDMs, we have that SDMs achieve the same improved Pareto

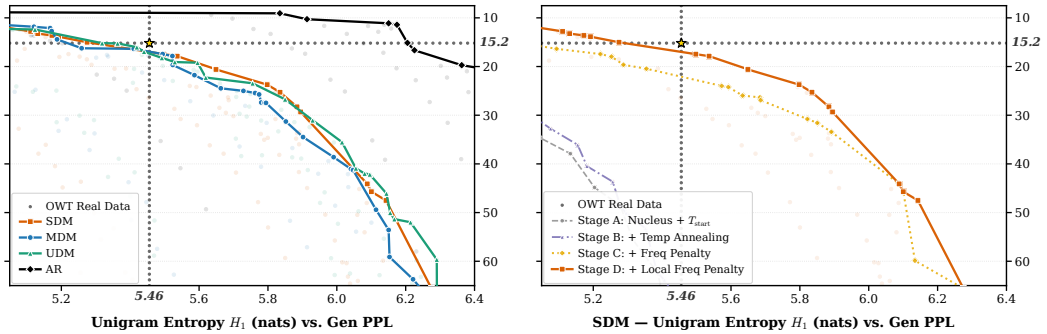

Figure 3: **OWT Pareto Frontiers of Entropy H_1 vs. GPT-2 GenPPL.** Dotted crosshairs and (★) denote measured OpenWebText validation data ($H_1 = 5.46$ nats, Gen PPL $= 15.2$). **Left:** Head-to-head Pareto frontiers comparing 64-step **SDM**, **MDM**, and **UDM** against 1024-step **AR** after sampler interventions. **Right:** Progressive ablation on 64-step **SDM** showing gains from **Stage A** (nucleus top-p) → **Stage B** (power-law logit temperature annealing) → **Stage C** (sequence frequency penalty) → **Stage D** (local frequency penalty).

frontier as those strong baselines. Furthermore, we show that similar interventions also benefit AR models, yielding a Pareto frontier that includes OWT real validation data. These results highlight that heuristic optimization can surpass the empirical validation point of OWT through simple logit shaping, thereby illustrating the limitation of generative frontiers (Pynadath et al., 2026b). We briefly describe the four *logit shaping* interventions we introduce. First, we consider top-p sampling as in Holtzman et al. (2019). Second we introduce a power-law logit temperature annealing during sampling similarly to DiffusionGemma Team et al. (2026); Chang et al. (2022). The third intervention which is by far the most impactful is a *sequence frequency penalty*, see (68) and (69) for more details. Note that this intervention benefits all experimental setups including MDM, UDM, SDM and AR. Finally, SDM is the only model to benefit from a *local frequency penalty* which bridges the gap between SDMs and state-of-the-art MDM and UDM for text generation, achieving a GenPPL/Entropy trade-off that is close to real validation data (GenPPL/Entropy: 17.0/5.46). More details on the interventions and additional results can be found in Section P.5. Our main results including the Pareto frontiers for all models we consider are presented in Figure 3. Non-cherry picked samples are shown in Section Q. Finally, in Section P.6, we present additional non-generative evaluation results on language understanding by evaluating our models on ARC-easy (Clark et al., 2018), PIQA (Bisk et al., 2020) and HellaSwag (Zellers et al., 2019).

6 Conclusion

This paper introduces Simplex Diffusion Models (SDMs) for discrete data. Unlike standard discrete diffusion models, SDMs maintain a continuous distributional belief state throughout the generative process, circumventing information collapse even without self-conditioning, with which SDMs remain complementary. Our proposed generative model introduces a flexible DDIM-type sampler that does not require discretizing an ODE or SDE.

When compared to strong baselines, SDMs are competitive on unconditional text generation and molecular synthesis, and state-of-the-art on coding. Nevertheless, our experiments remain limited to small models, and whether these conclusions remain valid at scale is an open question. Further experimentation is needed to assess the viability of the approach on large-scale language generation tasks.

More broadly, SDMs suggest that there is a false dichotomy between discrete and continuous diffusion language models. Discrete Diffusion Models emerge at the high temperature limit, and with the Argmax input (Section O.2), the denoiser processes a discrete sequence, while the generative process itself remains continuous on the simplex.

AI USE STATEMENT

We acknowledge the use of AI models during the preparation of this manuscript. AI tools were used to assist in the writing of the manuscript, the derivation of the proofs and the writing of the code. Experiments were run with the use of agentic tools. The main ideas were all introduced without relying on AI tools. Interpretation of the results and first writing of the manuscript were entirely done with no LLM assistance. The TinyGSM dataset is fully generated by GPT-3.5 (as explained by Liu et al., 2023a). The authors take full responsibility for all content, scientific claims, citations, and conclusions presented in this work.

REPRODUCIBILITY STATEMENT

To ensure the full reproducibility of our theoretical and empirical results, we provide comprehensive documentation across the paper and its supplementary material:

- **Theoretical Claims and Proofs:** Complete mathematical proofs for all propositions (Propositions 3.1 to 3.3 and 4.1) and extended results, including Gaussian fluctuation limits (Section E.7), the general DDIM framework (Section F), the variational lower bound (Section G), continuous-time pure-jump representations (Section H), the Brownian bridge perspective (Section I.2), and Simplex DMD distillation (Section J), are provided in Sections E to H, I.2 and J, with all mathematical assumptions explicitly stated.
- **Algorithms and Pseudocode:** Self-contained algorithmic pseudocode is detailed in Algorithm 1 (Training), Algorithm 2 (Inference), Algorithm 3 (Accelerated Marsaglia–Tsang Gamma Sampling; see also Section D), and Algorithm 4 (Simplex Distribution Matching Distillation), along with multi-token and classifier-free guidance extensions in Sections C.1 and C.2.
- **Model Architectures and Hyperparameters:** Detailed neural network architectures (Diffusion Transformer configurations, layer counts, embedding dimensions, attention heads), optimizer parameters (Adam/AdamW learning rates, warmup, EMA decay, gradient clipping), adaptive time schedules (Section O.1), denoiser input parameterizations (Section O.2), and variance schedules (ν_t; Section 4.2) are specified in Sections O.1, O.2, O.4, O.5, O.7 and O.8.
- **Experimental Details and Full Sweeps:** Full experimental ablations and hyperparameter sweeps are documented in Section P and in the full result tables (Tables 21 to 23 and 29 to 32), including exact mean and standard deviation values across random seeds.

REFERENCES

Sarah Alamdari, Nitya Thakkar, Rianne van den Berg, Alex Lu, Nicolo Fusi, Ava Amini, and Kevin Yang. Protein generation with evolutionary diffusion: Sequence is all you need. In *NeurIPS Generative AI and Biology (GenBio) Workshop*, 2023.

Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *Journal of Machine Learning Research*, 2025.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. Smollm2: When smol goes big – data-centric training of a small language model. *arXiv preprint arXiv:2502.02737*, 2025.

Ali Alp. Sudoku puzzle generator. https://github.com/alicommit-malp/sudoku, 2024.

Marianne Arriola, Aaron Gokaslan, Justin T. Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *International Conference on Learning Representations*, 2025.

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*, 2021.

Pavel Avdeyev, Chenlai Shi, Yuhao Tan, Kseniia Dudnyk, and Jian Zhou. Dirichlet diffusion score model for biological sequence generation. In *International Conference on Machine Learning*, 2023.

Jack Baker, Paul Fearnhead, Emily Fox, and Christopher Nemeth. Large-scale stochastic sampling from the probability simplex. *Advances in Neural Information Processing Systems*, 31, 2018.

Georgios Batzolis, Mark Girolami, and Luca Ambrogioni. CoBit: Language modeling with bitstream diffusion. *arXiv preprint arXiv:2605.07013*, 2026.

Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas Nolte, and Brian Karrer. Accelerated sampling from masked diffusion models via entropy bounded unmasking. In *Advances in Neural Information Processing Systems*, 2025.

Joe Benton, Yuyang Shi, Valentin De Bortoli, George Deligiannidis, and Arnaud Doucet. From denoising diffusions to denoising Markov models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 86(2):286–301, 2024.

G Richard Bickerton, Gaia V Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L Hopkins. Quantifying the chemical beauty of drugs. *Nature chemistry*, 4(2):90–98, 2012.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020.

Yoann Boget and Alexandros Kalousis. Unrestrained simplex denoising for discrete data: A non-Markovian approach applied to graph generation. *arXiv preprint arXiv:2603.28572*, 2026.

Bastian Boll, Daniel Gonzalez-Alvarado, and Christoph Schnörr. Generative modeling of discrete joint distributions by e-geodesic flow matching on assignment manifolds. *arXiv preprint arXiv:2402.07846*, 2024.

Bastian Boll, Daniel Gonzalez-Alvarado, Stefania Petra, and Christoph Schnörr. Generative assignment flows for representing and learning joint distributions of discrete data. *Journal of Mathematical Imaging and Vision*, 67(3):34, 2025.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

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

Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. In *International Conference on Machine Learning*, 2024.

Nuria Alina Chandra, Yucen Lily Li, Alan N Amin, Alex Ali, Joshua Rollins, Sebastian W Ober, Aniruddh Raghu, and Andrew Gordon Wilson. A unification of discrete, Gaussian, and simplicial diffusion. In *International Conference on Learning Representations*, 2026.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. MaskGIT: Masked generative image transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Jannis Chemseddine, Gregor Kornhardt, and Gabriele Steidl. Spherical flows for sampling categorical data. *arXiv preprint arXiv:2605.05629*, 2026.

Ricky TQ Chen and Yaron Lipman. Flow matching on general geometries. In *International Conference on Learning Representations*, 2024.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, 2018.

Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. In *International Conference on Learning Representations*, 2022.

Yuxin Chen, Chumeng Liang, Hangke Sui, Ruihan Guo, Chaoran Cheng, Jiaxuan You, and Ge Liu. LangFlow: Continuous diffusion rivals discrete in language modeling. *arXiv preprint arXiv:2604.11748*, 2026.

Chaoran Cheng, Jiahan Li, Jian Peng, and Ge Liu. Categorical flow matching on statistical manifolds. In *Advances in Neural Information Processing Systems*, 2024.

Chaoran Cheng, Jiahan Li, Jiajun Fan, and Ge Liu. α-flow: A unified framework for continuous-state discrete flow matching models. *arXiv preprint arXiv:2504.10283*, 2025.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

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

Maurice G Cox. The numerical evaluation of B-splines. *IMA Journal of Applied Mathematics*, 10 (2):134–149, 1972.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, 2013.

Oscar Davis, Samuel Kessler, Mircea Petrache, İsmail İ Ceylan, Michael Bronstein, and Avishek J Bose. Fisher flow matching for generative modeling over discrete data. In *Advances in Neural Information Processing Systems*, 2024.

Carl de Boor. On calculating with b-splines. *Journal of Approximation Theory*, 6(1):50–62, 1972.

Valentin De Bortoli, Emile Mathieu, Michael Hutchinson, James Thornton, Yee Whye Teh, and Arnaud Doucet. Riemannian score-based generative modelling. In *Advances in Neural Information Processing Systems*, 2022.

Justin Deschenaux and Caglar Gulcehre. Promises, outlooks and challenges of diffusion language modeling. *arXiv preprint arXiv:2406.11473*, 2024.

Justin Deschenaux and Caglar Gulcehre. Beyond autoregression: Fast LLMs via self-distillation through time. In *International Conference on Learning Representations*, volume 2025, pp. 5007–5045, 2025.

Justin Deschenaux and Caglar Gulcehre. Language modeling with hyperspherical flows. *arXiv preprint arXiv:2605.11125*, 2026.

Justin Deschenaux, Caglar Gulcehre, and Subham Sekhar Sahoo. The diffusion duality, chapter ii: ψ-samplers. In *International Conference on Learning Representations*, 2026a.

Justin Deschenaux, Lan Tran, and Caglar Gulcehre. Partition generative modeling: Masked modeling without masks. In *International Conference on Learning Representations*, volume 2026, pp. 132963–132988, 2026b.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, 2019.

Luc Devroye. Random variate generation in one line of code. In *Proceedings of the 28th Conference on Winter Simulation*, pp. 265–272, 1996.

Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, Curtis Hawthorne, Rémi Leblond, Will Grathwohl, and Jonas Adler. Continuous diffusion for categorical data. *arXiv preprint arXiv:2211.15089*, 2022.

DiffusionGemma Team, Adrien Ali Taïga, James Assiene, Daniele Calandriello, Rahma Chaabouni, João Gante, Tamara von Glehn, Nate Keating, Chris Knutsen, Martin Kukla, et al. DiffusionGemma technical report. *arXiv preprint arXiv:2608.00146*, 2026.

Ian Dunn and David Ryan Koes. Mixed continuous and categorical flow matching for 3D de novo molecule generation. *arXiv preprint arXiv:2404.19739*, 2024.

Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of Cheminformatics*, 1:1–11, 2009.

Nima Fathi, Torsten Scholak, and Pierre-Andre Noel. Unifying autoregressive and diffusion-based sequence generation. In *Conference on Language Modeling*, 2025.

Guhao Feng, Yihan Geng, Jian Guan, Wei Wu, Liwei Wang, and Di He. Theoretical benefit and limitation of diffusion language model. In *Advances in Neural Information Processing Systems*, 2025.

Nic Fishman, Leo Klarner, Emile Mathieu, Michael Hutchinson, and Valentin De Bortoli. Metropolis sampling for constrained diffusion models. In *Advances in Neural Information Processing Systems*, 2023.

Nic Fishman, Leo Klarner, Valentin De Bortoli, Emile Mathieu, and Michael Hutchinson. Diffusion models for constrained domains. *Transactions on Machine Learning Research*, 2024.

Griffin Floto, Thorsteinn Jonsson, Mihai Nica, Scott Sanner, and Eric Zhengyu Zhu. Diffusion on the probability simplex. In *International Conference on Machine Learning Worshop on Sampling and Optimization in Discrete Space*, 2023.

Antonio Franca and Alexander Tong. Hacking generative perplexity: Why unconditional text evaluation needs distributional metrics. *arXiv preprint arXiv:2606.08417*, 2026.

Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

Samson Gourevitch, Yazid Janati, Dario Shariatian, Umut Simsekli, Eric Moulines, Eric P. Xing, and Alain Durmus. Uniform diffusion models revisited: Leave-one-out denoiser and absorbing state reformulation. *arXiv preprint arXiv:2605.22765*, 2026.

Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. In *International Conference on Learning Representations*, 2018.

Will Grathwohl, Kevin Swersky, Milad Hashemi, David Duvenaud, and Chris Maddison. Oops i took a gradient: Scalable sampling for discrete distributions. In *International Conference on Machine Learning*, 2021.

Dylan Greaves. Extended one-liners for the Beta, Gamma, and Dirichlet distributions with shape parameters below one. *arXiv preprint arXiv:2604.11199*, 2026.

Ishaan Gulrajani and Tatsunori B Hashimoto. Likelihood-based diffusion language models. In *Advances in Neural Information Processing Systems*, 2023.

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 (Volume 1: Long Papers)*, pp. 11575–11596, 2023.

Ulrich G Haussmann and Etienne Pardoux. Time reversal of diffusions. *The Annals of Probability*, 14(3):1188–1205, 1986.

Doron Haviv, Aram-Alexandre Pooladian, Dana Pe'er, and Brandon Amos. Wasserstein flow matching: Generative modeling over families of distributions. In *International Conference on Machine Learning*, 2025.

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

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020.

Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, and Tim Salimans. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

Ari Holtzman, Peter West, Vered Shwartz, Yejin Choi, and Luke Zettlemoyer. Surface form competition: Why the highest probability answer isn't always right. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 7038–7051, 2021.

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*, volume 34, pp. 12454–12465, 2021.

Emiel Hoogeboom, David Ruhe, Jonathan Heek, Thomas Mensink, and Tim Salimans. Beyond single tokens: Distilling discrete diffusion models via discrete MMD. *arXiv preprint arXiv:2603.20155*, 2026.

Keya Hu, Linlu Qiu, Yiyang Lu, Hanhong Zhao, Tianhong Li, Yoon Kim, Jacob Andreas, and Kaiming He. ELF: Embedded language flows. *arXiv preprint arXiv:2605.10938*, 2026.

Chin-Wei Huang, Milad Aghajohari, Joey Bose, Prakash Panangaden, and Aaron C Courville. Riemannian diffusion models. In *Advances in Neural Information Processing Systems*, 2022.

Jaehyeong Jo and Sung Ju Hwang. Continuous diffusion model for language modeling. In *Advances in Neural Information Processing Systems*, 2025.

Mingyu Jo, Jaesik Yoon, Justin Deschenaux, Caglar Gulcehre, and Sungjin Ahn. Loopholing discrete diffusion: Deterministic bypass of the sampling wall. In *International Conference on Learning Representations*, 2026.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems*, 2022.

Tero Karras, Miika Aittala, Tuomas Kynkäänniemi, Jaakko Lehtinen, Timo Aila, and Samuli Laine. Guiding a diffusion model with a bad version of itself. In *Advances in Neural Information Processing Systems*, 2024.

Jaeyeon Kim, Jonathan Geuter, David Alvarez-Melis, Sham M. Kakade, and Sitan Chen. Stop training for the worst: Progressive unmasking accelerates masked diffusion training. In *International Conference on Machine Learning*, 2026a.

Jaeyeon Kim, Seunggeun Kim, Taekyun Lee, David Z. Pan, Hyeji Kim, Sham Kakade, and Sitan Chen. Fine-tuning masked diffusion for provable self-correction. In *International Conference on Machine Learning*, 2026b.

Diederik Kingma and Ruiqi Gao. Understanding diffusion objectives as the ELBO with simple data augmentation. In *Advances in Neural Information Processing Systems*, 2023.

Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. In *Advances in Neural Information Processing Systems*, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Nikita Kornilov, David Li, Tikhon Mavrin, Aleksei Leonov, Nikita Gushchin, Evgeny Burnaev, Iaroslav Koshelev, and Alexander Korotin. Universal inverse distillation for matching models with real-data supervision (no gans). In *International Conference on Learning Representations*, 2026.

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. *arXiv preprint arXiv:2602.16813*, 2026.

Seul Lee, Karsten Kreis, Srimukh Prasad Veccham, Meng Liu, Danny Reidenbach, Saee Paliwal, Weili Nie, and Arash Vahdat. Genmol: A drug discovery generalist with discrete diffusion. In *International Conference on Machine Learning*, 2025.

Christian Léonard. Some properties of path measures. In *Séminaire de Probabilités XLVI*, pp. 207–230. Springer, 2014.

Jose Lezama, Tim Salimans, Lu Jiang, Huiwen Chang, Jonathan Ho, and Irfan Essa. Discrete predictor-corrector diffusion models for image synthesis. In *International Conference on Learning Representations*, 2023.

David Li, Nikita Gushchin, Dmitry Abulkhanov, Eric Moulines, Ivan Oseledets, Maxim Panov, and Alexander Korotin. IDLM: Inverse-distilled diffusion language models. In *International Conference on Machine Learning*, 2026.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

Bingbin Liu, Sebastien Bubeck, Ronen Eldan, Janardhan Kulkarni, Yuanzhi Li, Anh Nguyen, Rachel Ward, and Yi Zhang. TinyGSM: achieving $> 80\%$ on GSM8k with small language models. *arXiv preprint arXiv:2312.09241*, 2023a.

Guan-Horng Liu, Tianrong Chen, Evangelos Theodorou, and Molei Tao. Mirror diffusion models for constrained and watermarked generation. In *Advances in Neural Information Processing Systems*, 2023b.

Sulin Liu, Juno Nam, Andrew Campbell, Hannes Stärk, Yilun Xu, Tommi Jaakkola, and Rafael Gómez-Bombarelli. Think while you generate: Discrete diffusion with planned denoising. In *International Conference on Learning Representations*, 2025.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023c.

Yue Liu, Yuzhong Zhao, Zheyong Xie, Qixiang Ye, Jianbin Jiao, Yao Hu, Shaosheng Cao, and Yunfan Liu. Balancing understanding and generation in discrete diffusion models. In *International Conference on Machine Learning*, 2026.

Aaron Lou and Stefano Ermon. Reflected diffusion models. In *International Conference on Machine Learning*, 2023.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion language modeling by estimating the ratios of the data distribution. In *International Conference on Machine Learning*, 2024.

Eugene Lukacs. A characterization of the gamma distribution. *The Annals of Mathematical Statistics*, 26(2):319–324, 1955.

Weijian Luo, Tianyang Hu, Shifeng Zhang, Jiacheng Sun, Zhenguo Li, and Zhihua Zhang. Diff-instruct: A universal approach for transferring knowledge from pre-trained diffusion models. In *Advances in Neural Information Processing Systems*, 2023.

Rabeeh Karimi Mahabadi, Hamish Ivison, Jaesung Tae, James Henderson, Iz Beltagy, Matthew E Peters, and Arman Cohan. Tess: Text-to-text self-conditioned simplex diffusion. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2347–2361, 2024.

George Marsaglia and Wai Wan Tsang. A simple method for generating Gamma variables. *ACM Transactions on Mathematical Software*, 26(3):363–372, 2000.

Kai Wang Ng, Guo-Liang Tian, and Man-Lai Tang. *Dirichlet and Related Distributions: Theory, Methods and Applications*. John Wiley & Sons, 2011.

Emmanuel Noutahi, Cristian Gabellini, Michael Craig, Jonathan S. C. Lim, and Prudencio Tossou. Gotta be safe: a new framework for molecular design. *Digital Discovery*, 3(4):796–804, 04 2024.

Andrey Okhotin, Dmitry Molchanov, Vladimir Arkhipkin, Grigory Bartosh, Viktor Ohanesian, Aibek Alanov, and Dmitry P Vetrov. Star-shaped denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2023.

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. In *International Conference on Learning Representations*, 2025.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Peter Potaptchik, Jason Yim, Adhi Saravanan, Peter Holderrieth, Eric Vanden-Eijnden, and Michael S. Albergo. Discrete flow maps. *arXiv preprint arXiv:2604.09784*, 2026.

Patrick Pynadath, Jiaxin Shi, and Ruqi Zhang. Candi: Hybrid discrete-continuous diffusion models. In *International Conference on Machine Learning*, 2026a.

Patrick Pynadath, Jiaxin Shi, and Ruqi Zhang. Generative frontiers: Why evaluation matters for diffusion language models. *arXiv preprint arXiv:2604.02718*, 2026b.

Yiming Qin, Manuel Madeira, Dorina Thanou, and Pascal Frossard. Defog: Discrete flow matching for graph generation. In *International Conference on Machine Learning*, 2025.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. URL https://api.semanticscholar.org/CorpusID:160025533.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

Gabriel Raya, Bac Nguyen, Georgios Batzolis, Yuhta Takida, Dejan Stancevic, Naoki Murata, Chieh-Hsin Lai, Yuki Mitsufuji, and Luca Ambrogioni. Noise scheduling as information-guided allocation in diffusion training. *arXiv preprint arXiv:2602.18647*, 2026.

Pierre H Richemond, Sander Dieleman, and Arnaud Doucet. Categorical SDEs with simplex diffusion. *arXiv preprint arXiv:2210.14784*, 2022.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Daan Roos, Oscar Davis, Floor Eijkelboom, Michael M. Bronstein, Max Welling, İsmail İlkan Ceylan, Luca Ambrogioni, and Jan-Willem van de Meent. Categorical flow maps. In *International Conference on Machine Learning*, 2026.

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

Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin Chiu, and Volodymyr Kuleshov. The diffusion duality. In *International Conference on Machine Learning*, 2025.

Subham Sekhar Sahoo, Jean-Marie Lemercier, Zhihan Yang, Justin Deschenaux, Jingyu Liu, John Thickstun, and Ante Jukic. Scaling beyond masked diffusion language models. In *International Conference on Machine Learning*, 2026.

Jinya Sakurai, Patrick Pynadath, Satoshi Hayakawa, Jaehong Yoon, Xulei Yang, Nancy F Chen, and Xun Xu. Simplex relaxation for discrete diffusion. *arXiv preprint arXiv:2608.10615*, 2026.

Tim Salimans, Thomas Mensink, Jonathan Heek, and Emiel Hoogeboom. Multistep distillation of diffusion models via moment matching. In *Advances in Neural Information Processing Systems*, 2024.

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

Eugene Seneta. The early years of the variance-gamma process. *Advances in Mathematical Finance*, pp. 3–19, 2007.

Alexander Shabalin, Simon Elistratov, Viacheslav Meshchaninov, Ildus Sadrtdinov, and Dmitry Vetrov. Why Gaussian diffusion models fail on discrete data? In *Third Conference on Language Modeling*, 2026.

Junzhe Shen, Jieru Zhao, Ziwei He, and Zhouhan Lin. CoDAR: Continuous diffusion language models are more powerful than you think. *arXiv preprint arXiv:2603.02547*, 2026.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. In *Advances in Neural Information Processing Systems*, 2024.

Yuyang Shi, Valentin De Bortoli, Andrew Campbell, and Arnaud Doucet. Diffusion Schrödinger bridge matching. In *Advances in Neural Information Processing Systems*, 2023.

Richard Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The Annals of Mathematical Statistics*, 35(2):876–879, 1964.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021b.

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. In *Advances in Neural Information Processing Systems*, 2025.

Hannes Stark, 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*, 2024.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568(C), 2024.

Haoran Sun, Hanjun Dai, Bo Dai, Haomin Zhou, and Dale Schuurmans. Discrete Langevin samplers via Wasserstein gradient flow. In *International Conference on Artificial Intelligence and Statistics*, 2023.

Simon Tavaré. Line-of-descent and genealogical processes, and their applications in population genetics models. *Theoretical Population Biology*, 26(2):119–164, 1984.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Petar Veličković, Federico Barbero, Christos Perivolaropoulos, Simon Osindero, and Razvan Pascanu. Perplexity cannot always tell right from wrong. *arXiv preprint arXiv:2601.22950*, 2026.

Clément Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. DiGress: Discrete denoising diffusion for graph generation. In *International Conference on Learning Representations*, 2023.

Cedric Villani. *Topics in Optimal Transportation*. American Mathematical Society, 2003.

Dimitri von Rütte, Janis Fluri, Yuhui Ding, Antonio Orvieto, Bernhard Schölkopf, and Thomas Hofmann. Generalized interpolating discrete diffusion. In *International Conference on Machine Learning*, 2025.

Dimitri von Rütte, Janis Fluri, Omead Pooladzandi, Bernhard Schölkopf, Thomas Hofmann, and Antonio Orvieto. Scaling behavior of discrete diffusion language models. In *International Conference on Learning Representations*, 2026.

Guanghan Wang, Yair Schiff, Subham Sahoo, and Volodymyr Kuleshov. Remasking discrete diffusion models with inference-time scaling. In *Advances in Neural Information Processing Systems*, 2025.

Linxuan Wang, Ziyi Wang, Yikun Bai, Wei Deng, Guang Lin, and Qifan Song. Generalized discrete diffusion with self-correction. In *International Conference on Machine Learning*, 2026.

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

Bernardo Williams, Victor M Yeom-Song, Marcelo Hartmann, and Arto Klami. Simplex-to-Euclidean bijections for categorical flow matching. In *Artificial Intelligence and Statistics*, 2026.

Jin Xu, Xiaojiang Liu, Jianhao Yan, Deng Cai, Huayang Li, and Jian Li. Learning to break the loop: Analyzing and mitigating repetitions for neural text generation, 2022. *arXiv preprint arXiv:2206.02369*, 2022.

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

Zhihan Yang, Wei Guo, Shuibai Zhang, Subham Sekhar Sahoo, Yongxin Chen, Arash Vahdat, Morteza Mardani, and John Thickstun. Continuous diffusion scales competitively with discrete diffusion for language. *arXiv preprint arXiv:2605.18530*, 2026.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, 2019.

Jinwei Zhang, Dimitri von Rütte, Yuhui Ding, and Thomas Hofmann. Denoising is not the end: Discrete diffusion language models with self-correction. In *Workshop on Latent & Implicit Thinking – Going Beyond CoT Reasoning*, 2026.

Kaizhong Zhang and Dennis Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262, 1989.

Yixiu Zhao, Jiaxin Shi, Feng Chen, Shaul Druckmann, Lester Mackey, and Scott Linderman. Informed correctors for discrete diffusion models. In *Advances in Neural Information Processing Systems*, 2025.

ORGANIZATION OF THE SUPPLEMENTARY

Part I: Algorithms and Implementation Details

A Beta, Dirichlet and Gamma Distributions **23**

A.1 Basic Definitions 23

A.2 Some Useful Properties of Dirichlet Distributions 23

B Training and Inference Algorithms **24**

B.1 Training 24

B.2 Inference 24

C Several Extensions **26**

C.1 Extension to Multiple Tokens 26

C.2 Classifier-Free Guidance 26

D Fast Sampling of Dirichlet Random Variables **27**

Part II: Theory and Generalizations

E Proofs and Additional Results **29**

E.1 Proof of Proposition 3.1 29

E.2 Proof of Proposition 3.2 29

E.3 Proof of Proposition 3.3 29

E.4 A Representation of P_t 30

E.5 Illustration and Temperature Limits for Proposition E.1 31

E.6 Proof of Proposition 4.1 33

E.7 Gaussian Fluctuations 35

F Generalization of DDIM **37**

F.1 General State-Space DDIM 37

F.2 Simplex Transitions 38

F.3 Gaussian Transition 40

F.4 Categorical Transition 41

F.5 Summary Table 43

G Variational Lower Bound **43**

H A Pure-Jump Representation **45**

H.1 Reminders on Lévy and Additive Processes 45

H.2 Continuous-Time Representation When $\kappa = 0$ . . . 46
H.3 No Stochastically Continuous Markov Embedding for Fixed $\kappa > 0$ . . . 48

I A Bridge Perspective **49**
I.1 High Level Intuition . . . 50
I.2 Bridge for the Beta-Clock Process . . . 51
I.3 Optimal Transport Perspective . . . 53

J Distillation **54**
J.1 Simplex Distribution Matching Distillation . . . 54
J.2 Link with Inverse Distilled Language Models . . . 56

K Extension to Birkhoff Polytope **58**
K.1 Birkhoff Polytope and Sinkhorn Theorem . . . 58
K.2 The Sinkhorn–Gamma Interpolating Path . . . 59
K.3 Induced Row-Wise Categorical Corruption . . . 60

L A Particle Interpretation **60**
L.1 Wright–Fisher Generation . . . 60
L.2 Simplex Transition as an Approximated Wright–Fisher Model. . . . 61
L.3 Limiting Process . . . 62

Part III: Connection with the Literature

M Extended Related Work **64**
M.1 General overview . . . 64
M.2 Simplex Relaxation for Discrete Diffusion . . . 65

N Extended Background **66**
N.1 Discrete Diffusion Models . . . 66
N.2 Predictor-Corrector Sampler . . . 67
N.3 Self-Conditioning and Loopholing . . . 68
N.4 Dirichlet Flow Matching . . . 68

Part IV: Experimental Setup, Ablations and Results

O Extended Experimental Details **70**
O.1 Time Samplers and Sampling Schedules . . . 70
O.2 Denoiser Input for SDMs . . . 70
O.3 Dirichlet Flow Matching Baseline . . . 71

O.4 Sudoku 73
O.5 TinyGSM 73
O.6 OpenWebText 74
O.7 Language Understanding 74
O.8 Unconditional molecular generation 75

P Additional Experimental Results **77**
P.1 Sudoku 77
P.2 TinyGSM 77
P.3 Distillation on TinyGSM 81
P.4 Auto-Guidance on TinyGSM 81
P.5 OpenWebText 82
P.6 Language Understanding 86
P.7 Unconditional molecular generation 87

Q Additional OWT Samples **92**

R Full Result Tables **99**
R.1 Sudoku 99
R.2 TinyGSM 103

Part I: Algorithms and Implementation Details

A Beta, Dirichlet and Gamma Distributions

A.1 Basic Definitions

For sake of completeness, we recall elementary facts about the Beta, Dirichlet and Gamma distributions. The Beta distribution is a distribution on $[0,1]$. Given $a, b > 0$, the associated Beta density $\mathrm{Beta}(x; a, b)$ is given for any $x \in [0,1]$ by

$$\mathrm{Beta}(x; a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1}(1-x)^{b-1},$$

where $\Gamma(\cdot)$ is the Gamma function. The Dirichlet distribution is a distribution on the simplex Δ_N. Given $\beta = (\beta_1, \ldots, \beta_N)$, the associated Dirichlet density $\mathrm{Dir}(P; \beta)$ is given for any $P \in \Delta_N$ by

$$\mathrm{Dir}(P; \beta) = \frac{\Gamma(\sum_{i=1}^N \beta_i)}{\prod_{i=1}^N \Gamma(\beta_i)} \prod_{i=1}^N p_i^{\beta_i - 1},$$

We have

$$\mathbb{E}[P] = \bar{P} = \left(\frac{\beta_1}{C}, \ldots, \frac{\beta_N}{C}\right), \quad \mathrm{Cov}[P] = \frac{\mathrm{diag}(\bar{P}) - \bar{P}\bar{P}^\top}{C+1}, \tag{13}$$

where $C = \sum_{i=1}^N \beta_i$ the concentration parameter of the Dirichlet distribution. The Gamma distribution is a distribution on $[0, \infty)$ defined for parameters $\alpha, \beta > 0$ by

$$\mathrm{Gamma}(x; \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} \exp(-\beta x).$$

We will use extensively in practice the following fact. For N independently distributed Gamma random variables $X_1 \sim \mathrm{Gamma}(\alpha_1, 1), \cdots, X_N \sim \mathrm{Gamma}(\alpha_N, 1)$, we have

$$P = \left(\frac{X_1}{\sum_{i=1}^N X_i}, \cdots, \frac{X_N}{\sum_{i=1}^N X_i}\right) \sim \mathrm{Dir}(\alpha_1, \cdots, \alpha_N).$$

We will use the following convention throughout. Whenever a Gamma/Beta/Dirichlet parameter lies on the boundary (for instance, some Dirichlet coordinates are zero, or a formula is obtained as the limit $t \downarrow 0$), the corresponding distribution is understood in the natural degenerate/weak-limit sense. In particular, Dirichlet distributions with some zero coordinates are supported on the corresponding face of the simplex, Beta distributions with a zero parameter are degenerate at 0 or 1, and formulas involving $t = 0$ are interpreted as limits of the same formulas for $t > 0$.

A.2 Some Useful Properties of Dirichlet Distributions

We now recall two simple results for Dirichlet distributions. To keep the paper self-contained, we provide proofs below without any claim of originality.

Proposition A.1 (Sum of Dirichlet random variables): *Let $\alpha, \beta \in \mathbb{R}^N$ with $\beta_i \geqslant \alpha_i \geqslant 0$ for any $i \in \{1, \ldots, N\}$ and assume that $X \sim \mathrm{Dir}(\alpha)$. Denote $\gamma = \beta - \alpha$ and let $W \sim \mathrm{Beta}(\sum_{i=1}^N \alpha_i, \sum_{i=1}^N \gamma_i)$ independent of X. Let $V \sim \mathrm{Dir}(\gamma)$, independent of X and W. Finally, let Y be given by*

$$Y = WX + (1-W)V.$$

Then, we have that $Y \sim \mathrm{Dir}(\beta)$.

Proof. Using the Gamma characterization of Dirichlet random variables, there exist independent random variables $(Z_i)_{i=1}^N$ such that $Z_i \sim \text{Gamma}(\alpha_i, 1)$ and $X = \left(Z_i / \sum_{j=1}^N Z_j\right)_{i=1}^N$. Similarly, there exist independent random variables $(Z'_i)_{i=1}^N$ such that $Z'_i \sim \text{Gamma}(\gamma_i, 1)$ and $V = \left(Z'_i / \sum_{j=1}^N Z'_j\right)_{i=1}^N$. We denote $\bar{Z} = \sum_{i=1}^N Z_i$, $\bar{Z}' = \sum_{i=1}^N Z'_i$, and $W = \bar{Z}/(\bar{Z} + \bar{Z}')$. Note that $\bar{Z}$ and $\bar{Z}'$ are independent of each other and independent of V and X. The random variable W is independent of X and V by Lukacs's Proportion-Sum Independence Theorem, see Ng et al. (2011). In addition, $\bar{Z} \sim \text{Gamma}(\sum_{i=1}^N \alpha_i, 1)$ and $\bar{Z}' \sim \text{Gamma}(\sum_{i=1}^N \gamma_i, 1)$ using the summation property of independent Γ distributions. Now using the characterization of Beta distributions with Gamma distributions, we get that $W \sim \text{Beta}(\sum_{i=1}^N \alpha_i, \sum_{i=1}^N \gamma_i)$. Finally, we have that

$$Y = WX + (1-W)V = \left(\frac{Z_i + Z'_i}{\sum_{j=1}^N (Z_j + Z'_j)}\right)_{i=1}^N .$$

We conclude the proof using the summation property of independent Γ random variables and the Γ characterization of Dirichlet random variables. □

Proposition A.2 (Dirichlet thinning): *Let $X \sim \text{Dir}(\alpha)$ with $\alpha \in \mathbb{R}^N$ with $\alpha_i \geqslant 0$ for any $i \in \{1, \dots, N\}$. The construction has to be understood on the supporting face $I = \{i : \alpha_i > 0\}$; coordinates outside I are fixed at zero and omitted from the normalization. For $\rho \in (0, 1]$ and $i \in I$, let $B_i \sim \text{Beta}(\rho\alpha_i, (1-\rho)\alpha_i)$ be mutually independent random variables, also independent of X, and define Y by*

$$Y_i = \frac{B_i X_i}{\sum_{j \in I} B_j X_j} \quad \text{for } i \in I, \qquad Y_i = 0 \quad \text{for } i \notin I.$$

Then $Y \sim \text{Dir}(\rho\alpha)$.

Proof. By restricting to the supporting face and relabelling its coordinates, it is enough to consider $\alpha_i > 0$ for every $i \in \{1, \dots, N\}$. We sample mutually independent variables $Z_i \sim \Gamma(\alpha_i, 1)$ and $B_i \sim \text{Beta}(\rho\alpha_i, (1-\rho)\alpha_i)$, and define $X = (Z_i / \sum_{j=1}^N Z_j)_{i=1}^N$. This defines the joint distribution of $(X, (B_i)_{i=1}^N)$. For any $i \in \{1, \dots, N\}$, let $Z'_i = Z_i B_i$ and we have that

$$Y = \left(\frac{B_i X_i}{\sum_{j=1}^N B_j X_j}\right)_{i=1}^N = \left(\frac{B_i Z_i}{\sum_{j=1}^N B_j Z_j}\right)_{i=1}^N = \left(\frac{Z'_i}{\sum_{j=1}^N Z'_j}\right)_{i=1}^N .$$

We now use the fact that for independent random variables $A \sim \text{Beta}(b, a-b)$ and $B \sim \text{Gamma}(a, 1)$ then $AB \sim \text{Gamma}(b, 1)$, using Lukacs's Proportion-Sum Independence Theorem, see Ng et al. (2011). So for any $i \in \{1, \dots, N\}$, with $A = B_i$, $B = Z_i$, $a = \alpha_i$ and $b = \rho\alpha_i$. It follows that that $(Z'_i)_{i=1}^N$ is a collection of independent random variables such that $Z'_i \sim \text{Gamma}(\rho\alpha_i, 1)$. Therefore, using once again the Gamma representation of Dirichlet random variables, $Y \sim \text{Dir}(\rho\alpha)$. □

B Training and Inference Algorithms

B.1 Training

The training algorithm is detailed in Algorithm 1.

B.2 Inference

The inference algorithm is described in Algorithm 2. For readability, this is presented in the case $0 < \rho_{s,t}^{\kappa} < 1$ with non-degenerate Beta and Dirichlet parameters (boundary cases are understood by continuity).

Algorithm 1 Training of Simplex Diffusion Model

Require: Data distribution p_{data}, Reference distribution $\pi \in \Delta_N$ (e.g., Uniform), Schedule $(\alpha_t)_{t\in[0,1]}$, Concentration $(c_t)_{t\in(0,1]}$, time-sampling density $w(t)$, and Model $\hat{P}_\theta : \Delta_N \times [0,1] \to \Delta_N$

while not converged **do**
  1. Sample Data and Time
  Sample $x_0 \sim p_{\text{data}}$ ▷ $x_0 \in \{1, \dots, N\}$
  Sample time $t \sim w(t)$ ▷ Uniform by default
  2. Compute Corruption Parameters
  $P_0 \leftarrow e_{x_0}$ ▷ $P_0 \in \{0,1\}^N$; i.e., $P_0 = \text{OneHot}(x_0)$
  $\beta_t \leftarrow c_t\,(\alpha_t P_0 + (1-\alpha_t)\pi)$ ▷ Dirichlet parameters $\beta_t \in \mathbb{R}_+^N$
  3. Sample Noisy Simplex State P_t
  Sample $P_t \sim \text{Dir}(\beta_t)$
  4. Predict and Optimize
  $\hat{P}_0 \leftarrow \hat{P}_\theta(t, P_t)$ ▷ Model prediction (Softmax output)
  $\mathcal{L} \leftarrow \text{CrossEntropy}(x_0, \hat{P}_0) = -\log(\hat{P}_{0,x_0})$
  $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$ ▷ Gradient descent step
end while

Algorithm 2 Inference for Simplex Diffusion Model

Require: Trained Model $\hat{P}_\theta$, Prior π, Schedule $(\alpha_t)_{t\in[0,1]}$, Concentration $(c_t)_{t\in[0,1]}$, Churn $\kappa \in [0,1]$, Number of steps M, Time sequence $0 = t_0 < \cdots < t_M = 1$.

1. Initialize
Sample $P_1 \sim \text{Dir}(c_1\pi)$ ▷ Sample from scaled prior
2. Reverse Generative Loop
for $k = M, M-1, \dots, 2$ **do**
  $t \leftarrow t_k, \qquad s \leftarrow t_{k-1}$
  a. Sample a Clean Vertex
  Sample $\widetilde{x}_0 \sim \text{Cat}(\hat{P}_\theta(t, P_{t_k}))$
  $P_0 \leftarrow e_{\widetilde{x}_0}$
  b. Compute $p_{s|0,t}(P_s \mid P_0, P_t)$ **Parameters**
  $r_{s,t} \leftarrow \min\left\{1, \dfrac{c_s(1-\alpha_s)}{c_t(1-\alpha_t)}\right\}$
  $\rho \leftarrow (1-\kappa) r_{s,t}$
  $a_W \leftarrow \rho c_t$
  $b_W \leftarrow c_s - \rho c_t$
  $\beta_V \leftarrow \beta_s(P_0, \pi) - \rho\beta_t(P_0, \pi)$
  c. Sample Variables
  Sample $W \sim \text{Beta}(a_W, b_W)$ and $V \sim \text{Dir}(\beta_V)$
  d. Compute $P_{t_k}^\kappa$
  for $i \in \{1, \dots, N\}$ **do**
    Sample $B_i \sim \text{Beta}(\rho\beta_t(P_0,\pi)_i, (1-\rho)\beta_t(P_0,\pi)_i)$
  end for
  $P_{t_k}^\kappa \leftarrow \dfrac{B \odot P_{t_k}}{\sum_{i=1}^N B_i P_{t_k,i}}$ ▷ Element-wise mult. & normalize
  e. Update State
  $P_{t_{k-1}} \leftarrow W P_{t_k}^\kappa + (1-W)V$
end for
3. Endpoint
Sample $x_{\text{data}} \sim \text{Cat}(\hat{P}_\theta(t_1, P_{t_1}))$
$P_{t_0} \leftarrow e_{x_{\text{data}}}$
return x_{data}

C Several Extensions

C.1 Extension to Multiple Tokens

So far, we have presented Simplex Diffusion Models for a single token taking values in $\mathcal{X} = \{1, \dots, N\}$. We now discuss how to extend the construction to sequences $x_0 = (x_0^1, \dots, x_0^L) \in \mathcal{X}^L$. To obtain a scalable solution, we consider a product of simplices and represent the state at time t by

$$P_t = (P_t^1, \dots, P_t^L) \in (\Delta_N)^L, \qquad P_0^\ell = e_{x_0^\ell}, \quad \ell \in \{1, \dots, L\}.$$

We then define the forward process independently across positions:

$$p_{t|0}(P_t \mid P_0) = \prod_{\ell=1}^{L} \mathrm{Dir}\big(P_t^\ell; \beta_t(P_0^\ell, \pi)\big), \qquad \beta_t(P_0^\ell, \pi) = c_t\big(\alpha_t P_0^\ell + (1-\alpha_t)\pi\big).$$

The denoiser is now a joint sequence model

$$\hat{P}_\theta : (\Delta_N)^L \times [0,1] \to (\Delta_N)^L, \qquad \hat{P}_\theta(t, P_t) = (\hat{P}_0^1, \dots, \hat{P}_0^L),$$

where each $\hat{P}_0^\ell \in \Delta_N$ is the predicted clean distribution at position ℓ. Importantly, although the forward and reverse kernels factorize across positions, each $\hat{P}_0^\ell$ can depend on the full noisy sequence P_t.

Training is performed with the sequence-level extension of the cross-entropy objective. At inference time, Proposition 3.3 is applied independently at each position after sampling a clean token from each predicted categorical distribution. More precisely, for any $0 < s < t \leqslant 1$, we first draw, conditionally independently across positions,

$$\widetilde{x}_0^\ell \sim \mathrm{Cat}(\hat{P}_0^\ell), \qquad \widetilde{P}_0^\ell = e_{\widetilde{x}_0^\ell}, \qquad \ell \in \{1, \dots, L\}.$$

We then sample independently across positions

$$W_{s,t,\ell}^\kappa \sim \mathrm{Beta}\left(\rho_{s,t}^\kappa c_t,\, c_s - \rho_{s,t}^\kappa c_t\right), \qquad V_{s,t,\ell}^\kappa \sim \mathrm{Dir}\left(\beta_s(\widetilde{P}_0^\ell, \pi) - \rho_{s,t}^\kappa \beta_t(\widetilde{P}_0^\ell, \pi)\right),$$

where

$$\rho_{s,t}^\kappa = (1-\kappa) r_{s,t}, \qquad r_{s,t} = \min\left\{1, \frac{c_s(1-\alpha_s)}{c_t(1-\alpha_t)}\right\}.$$

For each $i \in \{1, \dots, N\}$, we also sample

$$B_i^\ell \sim \mathrm{Beta}\left(\rho_{s,t}^\kappa\, \beta_t(\widetilde{P}_0^\ell, \pi)_i, (1-\rho_{s,t}^\kappa)\, \beta_t(\widetilde{P}_0^\ell, \pi)_i\right).$$

We then define

$$(P_{s,t}^\ell)_i^\kappa = \frac{B_i^\ell P_{t,i}^\ell}{\sum_{j=1}^N B_j^\ell P_{t,j}^\ell}, \qquad P_s^\ell = W_{s,t,\ell}^\kappa (P_{s,t}^\ell)^\kappa + (1 - W_{s,t,\ell}^\kappa) V_{s,t,\ell}^\kappa.$$

When $\rho_{s,t}^\kappa = 1$, we set $(P_{s,t}^\ell)^\kappa = P_t^\ell$ directly and omit the B_i^ℓ variables.

This yields the factorized mixture reverse transition

$$p_{s|t}^\theta(P_s \mid P_t) = \prod_{\ell=1}^{L} \left\{ \sum_{j=1}^{N} \hat{P}_{0,j}^\ell\, p_{s|0,t}(P_s^\ell \mid P_0^\ell = e_j, P_t^\ell) \right\}.$$

After the final positive-time step, we compute $(\hat{P}_0^1, \dots, \hat{P}_0^L) = \hat{P}_\theta(t_1, P_{t_1})$ and sample $x_{\mathrm{data}}^\ell \sim \mathrm{Cat}(\hat{P}_0^\ell)$ independently across positions conditional on the prediction.

This factorized extension preserves the simplicity of the single-token construction while allowing the neural predictor $\hat{P}_\theta$ to exploit full sequence context.

C.2 Classifier-Free Guidance

We now describe how to incorporate classifier-free guidance into the generative reverse process for the single-token case. The extension to multiple tokens is straightforward. Let $c \in \mathcal{C}$ denote an external condition (e.g., class label, text prompt, side information). We replace the predictor $\hat{P}_\theta(t, P_t)$ by a conditional predictor $\hat{P}_\theta(t, c, P_t)$.

Training. During training, we sample a dropped condition $\tilde{c}$ according to

$$\tilde{c} = \begin{cases} c, & \text{with probability } 1 - p_{\text{drop}}, \\ \varnothing, & \text{with probability } p_{\text{drop}}, \end{cases}$$

where $\varnothing$ denotes the null condition. We then optimize the same objective as before, replacing c by $\tilde{c}$. This gives

$$L_{\text{CFG}}(\theta) = \mathbb{E}_{t,x_0,P_t,\tilde{c}}\left[-\log \hat{P}_\theta(t, \tilde{c}, P_t)_{x_0}\right].$$

Guided Plug-In Prediction. At inference time, given P_t and a target condition c, we compute the unconditional and conditional predictions

$$\hat{P}_0^{\text{u}} = \hat{P}_\theta(t, \varnothing, P_t), \qquad \hat{P}_0^{\text{c}} = \hat{P}_\theta(t, c, P_t).$$

For a guidance scale $w \geqslant 0$, we define the guided predictor $\tilde{P}_0 \in \Delta_N$ by

$$\tilde{P}_{0,i} = \frac{(\hat{P}_{0,i}^{\text{u}})^{1-w}(\hat{P}_{0,i}^{\text{c}})^{w}}{\sum_{j=1}^{N}(\hat{P}_{0,j}^{\text{u}})^{1-w}(\hat{P}_{0,j}^{\text{c}})^{w}}, \qquad i \in \{1, \dots, N\}.$$

The cases $w = 0$, $w = 1$, and $w > 1$ correspond respectively to unconditional sampling, conditional sampling, and classifier-free guidance.

Guided Reverse Transition. Classifier-free guidance is implemented by replacing the sample P_0 from $\hat{P}_0$ in Proposition 3.3 and Algorithm 2 by one from its guided version $\tilde{P}_0$.

D Fast Sampling of Dirichlet Random Variables

In order to sample from Dirichlet distribution efficiently one leverage the representation of Dirichlet distributions in terms of Gamma distributions. More precisely, for $Y \sim \text{Dir}(\alpha)$ with $\alpha = (\alpha_1, \dots, \alpha_N)$ can be efficiently obtained by defining

$$Y = \left(\frac{X_1}{\sum_{j=1}^{N} X_j}, \dots, \frac{X_N}{\sum_{j=1}^{N} X_j}\right),$$

with $X_i \sim \text{Gamma}(\alpha_i, 1)$. Hence in order to sample efficiently from a Dirichlet distribution one must efficiently sample from a Gamma distribution. In order to do so, we leverage the Marsaglia–Tsang algorithm, Marsaglia & Tsang (2000); see also Algorithm 3. However, the current implementation of the Gamma sampler in JAX is either slow or approximate Bradbury et al. (2018), see `https://github.com/jax-ml/jax/issues/38141`.

In order to generate a Gamma random variable of shape $(d_1, \dots, d_n)$, the current official implementation splits an original key $d = \prod_{i=1}^{n} d_i$ times. Then it proceeds in using `vmap` on those d dimensions. The body of this parallelised function also contains key splitting. This incurs large memory overhead which makes the whole function slow. Instead, we propose an exact implementation which does not rely on `vmap` and therefore do not require creating many keys which significantly reduce the memory overhead. We reproduce a minimal example of this overhead as well as our solution below and provide some performance benchmark on CPU hardware, see Figure 4. All benchmarks were conducted on an AMD EPYC 7B13 processor (32 physical cores, 2.45GHz base clock, 128MB L3 cache) with 117 GB of DDR4 RAM using JAX and the XLA CPU backend in single precision (`float32`).

Finally, we highlight that a recent approach by Greaves (2026) show that Gamma distributions can be generated as extended one-liner, thereby resolving a conjecture of Devroye (1996). In practice, the approach leverages a Generalized Acceptance-Complement procedure with a Gaussian proposal and a triangular symmetric complement. In particular, the proposed sampler only requires three draws of independent random variables. As a consequence, this new sampler i) decreases drastically the number of calls to pseudo random number generators ii) bypasses the need of a while loop to sample from the distribution. Early benchmarks of this approach forecast a sampling speed-up of 1.5x over our fast implementation of the Marsaglia–Tsang algorithm.

Algorithm 3 Marsaglia & Tsang Gamma Sampler

Require: PRNG Key, Shape parameter $\alpha > 0$
Ensure: Sample $x \sim \text{Gamma}(\alpha, 1)$
1. Boosting for Small Alpha
if $\alpha < 1$ **then**
  $\alpha' \leftarrow \alpha + 1$ ▷ Boost variance for stability
  is_small $\leftarrow$ True
else
  $\alpha' \leftarrow \alpha$
  is_small $\leftarrow$ False
end if
2. Compute Constants
$d \leftarrow \alpha' - \frac{1}{3}$
$c \leftarrow \frac{1}{\sqrt{9d}}$
3. Rejection Sampling Loop
while sample not accepted **do**
  a. Generate Candidates
  Sample $z \sim \mathcal{N}(0, 1)$ ▷ Standard Normal
  Sample $u \sim \mathcal{U}(0, 1)$ ▷ Uniform
  $v \leftarrow 1 + c \cdot z$
  if $v \leqslant 0$ **then**
    continue ▷ Reject negative support
  end if
  $v \leftarrow v^3$
  $x_{\text{prop}} \leftarrow d \cdot v$ ▷ Proposed sample
  b. Acceptance Checks
  ▷ Squeeze Test (Fast Check)
  if $u < 1 - 0.0331 \cdot z^4$ **then**
    break
  end if
  ▷ Log-Likelihood Test (Exact Check)
  if $\log(u) < 0.5z^2 + d(1 - v + \log(v))$ **then**
    break
  end if
end while
4. Final Correction
if is_small is True **then**
  Sample $u_{\text{boost}} \sim \mathcal{U}(0, 1)$
  $x \leftarrow x_{\text{prop}} \cdot u_{\text{boost}}^{1/\alpha}$ ▷ Apply boosting correction
else
  $x \leftarrow x_{\text{prop}}$
end if
return x

(a) JAX Native Pattern (`vmap` over scalar PRNG)

```python
@partial(jax.jit, static_argnames=("N", "d"))
def jax_native_pattern(key, N: int, d: int):
  keys = jax.random.split(key, N)
  def _body(k):
    k1, k2 = jax.random.split(k, 2)
    z = jax.random.normal(k1, (d,))
    u = jax.random.uniform(k2, (d,))
    return z, u
  return jax.vmap(_body)(keys)
```

(b) Vectorized Pattern (Counter-based PRNG)

```python
@partial(jax.jit, static_argnames=("N", "d"))
def vectorized_pattern(key, N: int, d: int):
  k1, k2 = jax.random.split(key, 2)
  z = jax.random.normal(k1, (N, d))
  u = jax.random.uniform(k2, (N, d))
  return z, u
```

Figure 4: Comparison of PRNG generation patterns in JAX. Pattern (a) derives per-element keys inside `vmap`, creating 120 MB of DRAM traffic across Threefry rounds. Pattern (b) splits a single scalar key in CPU registers and generates $N \times d$ values via counter indexing, achieving a $3.8\times$ speedup with $N = 10^6$ and $d = 1$.

Part II: Theory and Generalizations

E Proofs and Additional Results

E.1 Proof of Proposition 3.1

We have that for any $t \in [0, 1]$

$$\begin{aligned} p_{t|0}(x_t|P_0) &= \int_{\Delta_N} p_{t|0}(x_t|P_t, P_0) p_{t|0}(P_t|P_0) \mathrm{d}P_t \\ &= \int_{\Delta_N} P_{t,x_t} p_{t|0}(P_t|P_0) \mathrm{d}P_t = \mathbb{E}_{P_t|P_0}[P_{t,x_t}]. \end{aligned}$$

From (13), the mean of the Dirichlet distribution is given for any $x_t \in \mathcal{X}$ by

$$\mathbb{E}_{P_t|P_0}[P_{t,x_t}] = \alpha_t \delta_{x_0}(x_t) + (1 - \alpha_t)\pi_{x_t},$$

which concludes the proof.

E.2 Proof of Proposition 3.2

Fix $t \in (0, 1)$. Recall that $P_0 = e_{x_0}$, which, under our boundary convention, may be viewed as the degenerate Dirichlet random variable $P_0 \sim \mathrm{Dir}(c_t \alpha_t e_{x_0})$. Let $W_t \sim \mathrm{Beta}\left(c_t\alpha_t, c_t(1 - \alpha_t)\right)$ and $V_t \sim \mathrm{Dir}\left(c_t(1 - \alpha_t)\pi\right)$ independently, and define

$$P_t = W_t P_0 + (1 - W_t)V_t.$$

The result is then an immediate application of Proposition A.1 with

$$\alpha \leftarrow c_t\alpha_t e_{x_0}, \qquad \beta \leftarrow c_t\left(\alpha_t e_{x_0} + (1 - \alpha_t)\pi\right), \qquad \gamma = \beta - \alpha \leftarrow c_t(1 - \alpha_t)\pi.$$

Hence

$$P_t \sim \mathrm{Dir}\left(c_t\left(\alpha_t P_0 + (1 - \alpha_t)\pi\right)\right) = \mathrm{Dir}\left(\beta_t(P_0, \pi)\right),$$

which concludes the proof.

E.3 Proof of Proposition 3.3

Since $\alpha_s \geqslant \alpha_t$, the definition of $r_{s,t}$ ensures that

$$\gamma^{\kappa}_{s,t}(P_0, \pi) = \beta_s(P_0, \pi) - \rho^{\kappa}_{s,t}\beta_t(P_0, \pi) \geqslant 0$$

coordinatewise. We write $\rho = \rho^{\kappa}_{s,t}$.

We first apply Proposition A.2. Since $P_t \mid P_0 \sim \mathrm{Dir}\big(\beta_t(P_0,\pi)\big)$ and $B_i \sim \mathrm{Beta}\left(\rho\beta_t(P_0,\pi)_i, (1-\rho)\beta_t(P_0,\pi)_i\right)$ independently, the vector defined in (6) satisfies

$$P^\kappa_{s,t} \mid P_0 \sim \mathrm{Dir}\left(\rho\beta_t(P_0,\pi)\right).$$

By construction, $V^\kappa_{s,t} \sim \mathrm{Dir}\left(\gamma^\kappa_{s,t}(P_0,\pi)\right)$ independently of $P^\kappa_{s,t}$, and $W^\kappa_{s,t} \sim \mathrm{Beta}\left(\sum_i \rho\beta_t(P_0,\pi)_i, \sum_i \gamma^\kappa_{s,t}(P_0,\pi)_i\right)$. Since $\sum_i \rho\beta_t(P_0,\pi)_i = \rho c_t$ and $\sum_i \gamma^\kappa_{s,t}(P_0,\pi)_i = c_s - \rho c_t$, this is exactly the distribution of $W^\kappa_{s,t}$ in (7).

We can therefore apply Proposition A.1 with $X \leftarrow P^\kappa_{s,t}$, $\alpha \leftarrow \rho\beta_t(P_0,\pi)$ and $\beta \leftarrow \beta_s(P_0,\pi)$ which gives

$$P_s = W^\kappa_{s,t}P^\kappa_{s,t} + (1 - W^\kappa_{s,t})V^\kappa_{s,t} \sim \mathrm{Dir}\big(\beta_s(P_0,\pi)\big).$$

This is exactly the compatibility condition (5).

E.4 A Representation of P_t

We provide here an explicit representation of P_t under exact DDIM sampling.

Proposition E.1 (P_t as a random convex combination): *Consider $c_t = \varepsilon/(1-\alpha_t)$. Let $\kappa = 0$ and $(t_i)_{i=0}^M$ with $M \in \mathbb{N}$ such that $0 = t_0 < \cdots < t_M = 1$. Let $P_{t_M} \sim p_{t_M}$. For $i = M-1, \ldots, 0$, conditionally on $P_{t_{i+1}}$, sample $P^i_0 \sim p_{0|t_{i+1}}(\cdot \mid P_{t_{i+1}})$, write $P^i_0 = e_{x^i_0}$, and then sample P_{t_i} from $p_{t_i|0,t_{i+1}}(\cdot \mid P^i_0, P_{t_{i+1}})$, equivalently using (9). For any $i \in \{0,\ldots,M\}$, we have $P_{t_i} \sim p_{t_i}$ and*

$$P_{t_i} = \sum_{j=i}^{M-1} L_{i,j} e_{x^j_0} + L_{i,M} P_{t_M}, \tag{14}$$

with $L_i = (L_{i,j})_{j=i}^M$ and $L_i \sim \mathrm{Dir}\left(c_{t_i} - c_{t_{i+1}}, \ldots, c_{t_{M-1}} - c_{t_M}, c_{t_M}\right)$.

Equation (14) shows that P_{t_i} is a random convex combination of the successive posterior clean-state draws $(x^j_0)_{j=i}^{M-1}$ and the initial noise P_{t_M}. For the plug-in transitions, this exact representation need not hold, but it motivates the interpretation of P_t as a belief-memory state. In Section E.5, we investigate the high and low temperature limits of this representation.

Proof. We first verify the marginal claim. It holds at $i = M$ by assumption. If $P_{t_{i+1}} \sim p_{t_{i+1}}$ and $P^i_0 \sim p_{0|t_{i+1}}(\cdot \mid P_{t_{i+1}})$, then $(P^i_0, P_{t_{i+1}})$ has joint law $p(P_0)p_{t_{i+1}|0}(P_{t_{i+1}} \mid P_0)$. Integrating the exact kernel $p_{t_i|0,t_{i+1}}$ and using the compatibility condition (5) gives $P_{t_i} \sim p_{t_i}$. Backward induction therefore proves $P_{t_i} \sim p_{t_i}$ for every i.

The proof relies on the stick-breaking property of the Dirichlet distribution (a direct consequence of Proposition A.1). If a random vector $X \sim \mathrm{Dir}(\alpha_1,\ldots,\alpha_K)$ and a random variable $W \sim \mathrm{Beta}(\sum_{k=1}^K \alpha_k, \alpha_0)$ are independent, then the augmented vector $(1-W, WX_1, \ldots, WX_K)$ is distributed as $\mathrm{Dir}(\alpha_0,\alpha_1,\ldots,\alpha_K)$. We proceed by backward induction on i, from $i = M-1$ down to 0. To simplify notation, we define the parameter sequence $\gamma_i = \frac{\varepsilon}{1-\alpha_{t_i}} - \frac{\varepsilon}{1-\alpha_{t_{i+1}}}$ for $i \in \{0,\ldots,M-1\}$, and $\gamma_M = \frac{\varepsilon}{1-\alpha_{t_M}}$. We start with $i = M-1$. By (9), we have:

$$P_{t_{M-1}} = W^0_{t_{M-1},t_M} P_{t_M} + (1 - W^0_{t_{M-1},t_M}) e_{x^{M-1}_0},$$

where $W^0_{t_{M-1},t_M} \sim \mathrm{Beta}\left(\gamma_M, \gamma_{M-1}\right)$. The vector $(1 - W^0_{t_{M-1},t_M}, W^0_{t_{M-1},t_M})$ follows a Dirichlet distribution with parameters (γ_{M-1}, γ_M). Setting $L_{M-1,M-1} = 1 - W^0_{t_{M-1},t_M}$ and $L_{M-1,M} = W^0_{t_{M-1},t_M}$ satisfies the proposition for the base case. Now assume the proposition holds for some $i+1 \leqslant M-1$; i.e.,

$$P_{t_{i+1}} = \sum_{j=i+1}^{M-1} L_{i+1,j} e_{x^j_0} + L_{i+1,M} P_{t_M},$$

where $L^\varepsilon_{i+1} \sim \mathrm{Dir}(\gamma_{i+1},\ldots,\gamma_M)$. Note that the sum of these concentration parameters is exactly $\sum_{k=i+1}^M \gamma_k = \frac{\varepsilon}{1-\alpha_{t_{i+1}}}$. So using Proposition 3.3 and applying the backward transition to step i, we

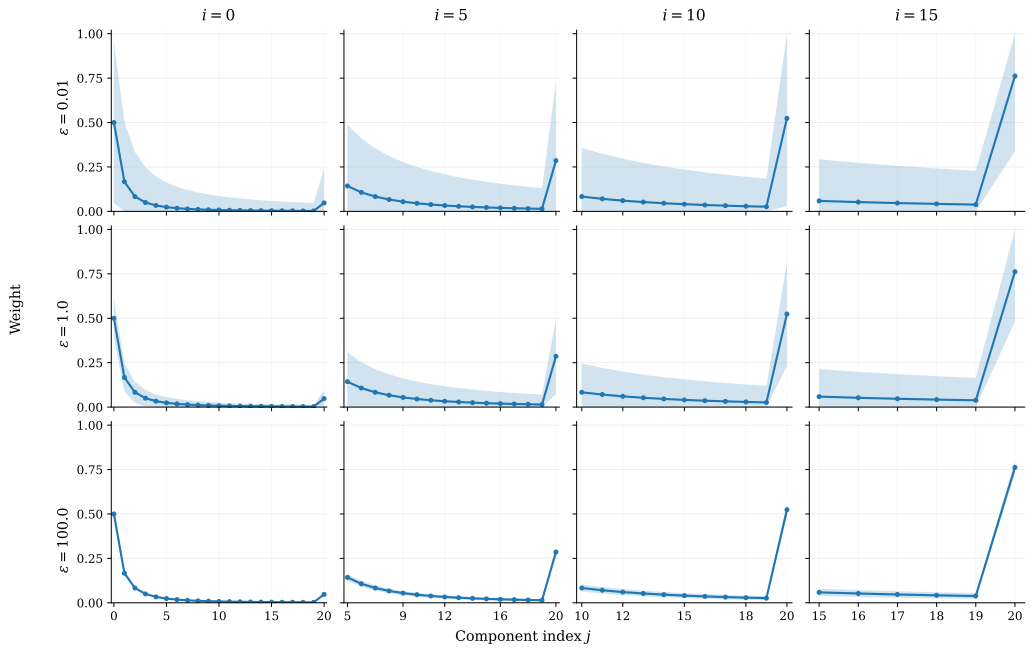

Figure 5: Visualization of the self-conditioning weights $L_{i,j}$ from Proposition E.1. Each panel shows the exact mean profile $\mathbb{E}[L_{i,j}]$ (solid line) together with a ± 1 standard-deviation band for the Dirichlet law in Proposition E.1, for $\varepsilon \in \{0.01, 1, 100\}$ and $i \in \{0, 5, 10, 15\}$. The mean curve is independent of ε.

have:

$$P_{t_i} = W^0_{t_i,t_{i+1}} P_{t_{i+1}} + (1 - W^0_{t_i,t_{i+1}}) e_{x_0^i},$$

where $W^0_{t_i,t_{i+1}} \sim \text{Beta}\left(\frac{\varepsilon}{1-\alpha_{t_{i+1}}}, \gamma_i\right) = \text{Beta}\left(\sum_{k=i+1}^{M} \gamma_k, \gamma_i\right)$. Therefore, we get

$$P_{t_i} = (1 - W^0_{t_i,t_{i+1}}) e_{x_0^i} + \sum_{j=i+1}^{M-1} \left(W^0_{t_i,t_{i+1}} L_{i+1,j}\right) e_{x_0^j} + \left(W^0_{t_i,t_{i+1}} L_{i+1,M}\right) P_{t_M}.$$

We identify the new weights L_i as follows: $L_{i,i} = 1 - W^0_{t_i,t_{i+1}}$ and $L_{i,j} = W^0_{t_i,t_{i+1}} L_{i+1,j}$ for $j \in \{i+1, \ldots, M\}$. Because $L_{i+1} \sim \text{Dir}(\gamma_{i+1}, \ldots, \gamma_M)$ and $W^0_{t_i,t_{i+1}} \sim \text{Beta}\left(\sum_{k=i+1}^{M} \gamma_k, \gamma_i\right)$, applying the stick-breaking property guarantees that the augmented vector L_i is distributed as $\text{Dir}(\gamma_i, \gamma_{i+1}, \ldots, \gamma_M)$. This concludes the inductive step and the proof. □

E.5 Illustration and Temperature Limits for Proposition E.1

We first present in Figure 5 the self-conditioning weights $L^{\varepsilon}_{i,j}$ appearing in Proposition E.1 for various ε and index i.

Let us now investigate the high and low temperature limits of Proposition E.1.

Corollary E.1 (Limiting behavior of intrinsic self-conditioning): *Let $L_i = (L_{i,j})_{j=i}^M$ be the Dirichlet-distributed weight vector defined in Proposition E.1. We define the unscaled parameters $c_j = \frac{1}{1-\alpha_{t_j}} - \frac{1}{1-\alpha_{t_{j+1}}}$ for $j \in \{i, \dots, M-1\}$ and $c_M = \frac{1}{1-\alpha_{t_M}}$. Let $C_i = \sum_{j=i}^M c_j = \frac{1}{1-\alpha_{t_i}}$. We have the following limiting cases.*

1. ***Low-temperature limit (Deterministic flow):** As $\varepsilon \to \infty$, L_i converges in distribution to a deterministic vector:*
$$L_i \xrightarrow{d} \bar{L}_i, \qquad \text{where} \quad \bar{L}_{i,j} = \frac{c_j}{C_i}.$$
2. ***High-temperature limit (Discrete jumps):** As $\varepsilon \to 0$, L_i converges in distribution to a categorical distribution over the canonical basis $(u_j)_{j=i}^M$ of the simplex over indices $j \in \{i, ..., M\}$:*
$$L_i \xrightarrow{d} u_{J_i}, \qquad \text{where} \quad J_i \sim \text{Cat}\left(\left(\frac{c_j}{C_i}\right)_{j=i}^M\right).$$

Therefore, we recover the discrete diffusion model behavior when $\varepsilon \to 0$ as in that case only one component $j^\star \in \{i, \dots, M\}$ is non-zero. This means that all contributions at steps other than $t_{j^\star}$ are neglected. This is not the case when $\varepsilon > 0$ and in the limit $\varepsilon \to +\infty$, the mixing of the clean-state draws is deterministic.

Proof. Recall that Proposition E.1 assumes the schedule $c_t = \varepsilon/(1-\alpha_t)$, and gives

$$L_i \sim \text{Dir}\left(c_{t_i} - c_{t_{i+1}}, \dots, c_{t_{M-1}} - c_{t_M}, c_{t_M}\right).$$

In terms of the unscaled parameters $\tilde{c}_j$ of the statement, namely $\tilde{c}_j = \frac{1}{1-\alpha_{t_j}} - \frac{1}{1-\alpha_{t_{j+1}}}$ for $j \in \{i, \dots, M-1\}$ and $\tilde{c}_M = \frac{1}{1-\alpha_{t_M}}$, we have

$$c_{t_j} - c_{t_{j+1}} = \varepsilon\, \tilde{c}_j \quad (j \leqslant M-1), \qquad c_{t_M} = \varepsilon\, \tilde{c}_M,$$

so that $L_i \sim \text{Dir}\,(\varepsilon \tilde{c}_i, \dots, \varepsilon \tilde{c}_M)$. Since $u \mapsto \alpha_u$ is non-increasing, each $\tilde{c}_j \geqslant 0$, and the total concentration telescopes:

$$\sum_{j=i}^M \varepsilon\, \tilde{c}_j = \varepsilon\, C_i = \frac{\varepsilon}{1-\alpha_{t_i}} = c_{t_i} \in (0, \infty) \qquad \text{for } i \geqslant 1.$$

Writing $\bar{L}_i = (\tilde{c}_j / C_i)_{j=i}^M \in \Delta_{M-i+1}$, which does not depend on ε, this reads

$$L_i \sim \text{Dir}\left((\varepsilon C_i)\, \bar{L}_i\right).$$

So the following results follow.

(1) Low-temperature limit. As $\varepsilon \to \infty$ we have $\varepsilon C_i \to \infty$, so $L_i \to \bar{L}_i$ in probability, hence in distribution. Explicitly, $\mathbb{E}[L_{i,j}] = \tilde{c}_j / C_i$ for every ε, while

$$\text{Var}\,[L_{i,j}] = \frac{(\tilde{c}_j/C_i)\,(1 - \tilde{c}_j/C_i)}{\varepsilon C_i + 1} \xrightarrow[\varepsilon \to \infty]{} 0.$$

(2) High-temperature limit. As $\varepsilon \to 0$ we have $\varepsilon C_i \to 0$, so we obtain

$$L_i \xrightarrow{d} \sum_{j=i}^M \frac{\tilde{c}_j}{C_i}\, \delta_{u_j},$$

where $(u_j)_{j=i}^M$ denotes the canonical basis of $\mathbb{R}^{M-i+1}$. Equivalently, $L_i \xrightarrow{d} u_{J_i}$ with $J_i \sim \text{Cat}\left((\tilde{c}_j/C_i)_{j=i}^M\right)$. Indices j with $\tilde{c}_j = 0$, which occur when α is constant on $[t_j, t_{j+1}]$, satisfy $L_{i,j} = 0$ almost surely and receive zero mass in the limit, consistently with the statement. □

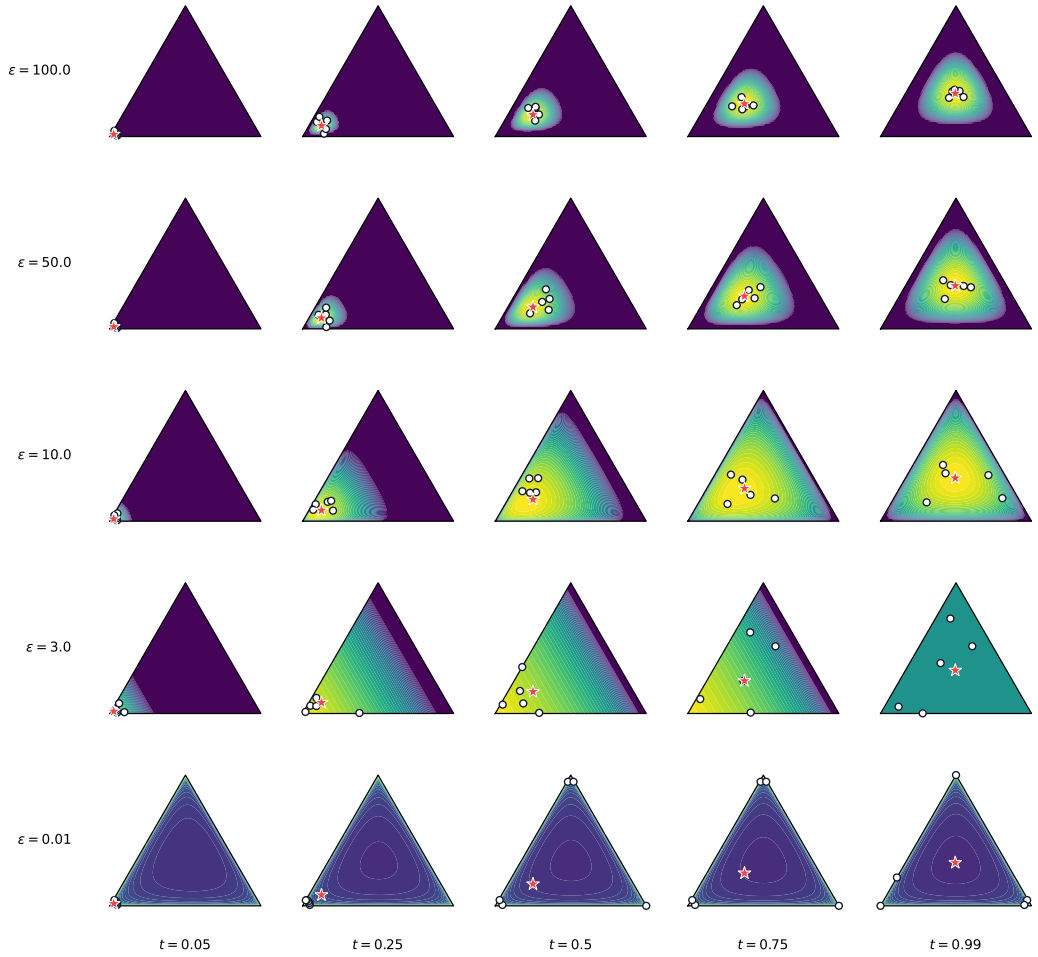

Figure 6: Each figure corresponds to the density of $p_{t|0}(P_t|P_0) = \mathrm{Dir}(c_t(\alpha_t P_0 + (1-\alpha_t)\pi))$, with π the uniform distribution and $c_t = \varepsilon/(1-\alpha_t)$ for different values of $t \in [0,1]$ and $\varepsilon > 0$. We also display the mean (red star) of $p_{t|0}$, i.e., $\mathbb{E}[P_t|P_0] = \alpha_t P_0 + (1-\alpha_t)\pi$ and observe that this quantity is independent of ε. Finally, we also display 5 samples (white dots) of $p_{t|0}$. Note that as $\varepsilon \to 0$ the samples concentrate on the vertices of the simplex, thereby illustrating the convergence of SDMs to their discrete diffusion counterpart.

E.6 Proof of Proposition 4.1

In this section, we investigate the temperature limits of SDMs. The complete design space is summarized in Figure 7. We recall that we set $c_t = \varepsilon/(1-\alpha_t)$.

We use the representation from Proposition 3.2,

$$P_t = W_t P_0 + (1 - W_t)\, V_t,$$

where

$$W_t \sim \mathrm{Beta}(\varepsilon h_t, \varepsilon), \qquad V_t \sim \mathrm{Dir}(\varepsilon\pi), \qquad h_t = \frac{\alpha_t}{1-\alpha_t},$$

and W_t and V_t are independent.

High-Temperature Limit. For fixed $t \in (0,1)$, standard results for the Beta and Dirichlet distributions give

$$W_t \xrightarrow{d} B_t, \qquad B_t \sim \mathrm{Ber}(\alpha_t),$$

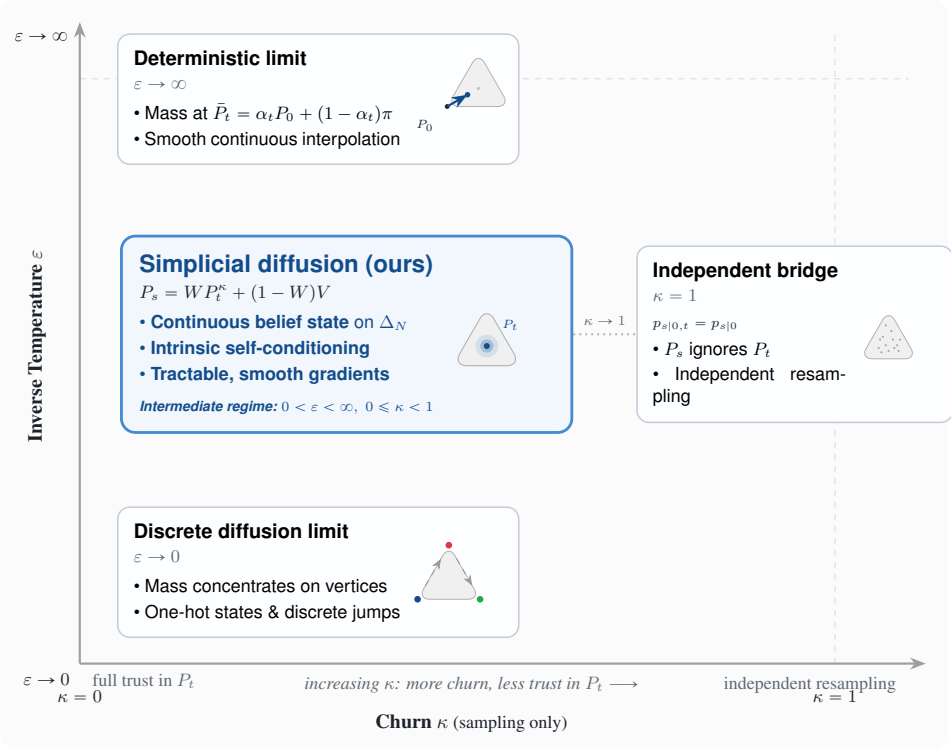

Figure 7: The design space of Simplex Diffusion Models (SDMs). The forward process is controlled by the inverse temperature ε: as $\varepsilon \to 0$, the simplex mass concentrates on vertices and recovers discrete-diffusion behavior; as $\varepsilon \to \infty$, the forward process concentrates around the deterministic interpolation $\bar{P}_t = \alpha_t P_0 + (1-\alpha_t)\pi$. The reverse process is controlled by the churn parameter κ: $\kappa = 0$ fully trusts the current belief P_t, while $\kappa = 1$ yields the independent bridge $p_{s|0,t} = p_{s|0}$. SDMs operate in the intermediate regime, maintaining a continuous belief state throughout sampling.

and

$$V_t \xrightarrow{d} V, \qquad V \sim \sum_{i=1}^{N} \pi_i \, \delta_{e_i}.$$

The limiting variables are independent. Hence, by the continuous mapping theorem,

$$P_t \xrightarrow{d} B_t P_0 + (1 - B_t)V.$$

Since $P_0 = e_{x_0}$ and V is supported on the canonical basis vectors, the limiting law is supported on the vertices of Δ_N.

For $\kappa = 0$, the backward transition is

$$P_s = W^0_{s,t} P_t + \left(1 - W^0_{s,t}\right) e_{x_0},$$

with

$$W^0_{s,t} \sim \mathrm{Beta}\left(\frac{\varepsilon}{1-\alpha_t}, \frac{\varepsilon}{1-\alpha_s} - \frac{\varepsilon}{1-\alpha_t}\right).$$

Applying again the small-concentration Beta limit gives

$$W^0_{s,t} \xrightarrow{d} \mathrm{Ber}\left(\frac{1-\alpha_s}{1-\alpha_t}\right).$$

Therefore, for every fixed $P_t \in \Delta_N$, we have in the limit

$$P_s \xrightarrow{d} W^0_{s,t} P_t + \left(1 - W^0_{s,t}\right) e_{x_0}, \qquad W^0_{s,t} \sim \mathrm{Ber}\left(\frac{1-\alpha_s}{1-\alpha_t}\right).$$

In particular, when P_t is a simplex vertex, the limiting transition is supported on simplex vertices.

Low-Temperature Limit. For fixed $t \in (0,1)$, standard results give

$$W_t \xrightarrow{\mathbb{P}} \frac{h_t}{1+h_t} = \alpha_t,$$

and

$$V_t \xrightarrow{\mathbb{P}} \pi.$$

Using once again the representation of P_t and the continuous mapping theorem,

$$P_t \xrightarrow{\mathbb{P}} \alpha_t P_0 + (1-\alpha_t)\pi,$$

which concludes the proof.

E.7 Gaussian Fluctuations

The next result identifies the fluctuations around the deterministic path at scale $\varepsilon^{-1/2}$. For this result, we introduce the tangent space of the simplex $\mathrm{T}\Delta_N = \{x \in \mathbb{R}^N,\ \mathbf{1}^\top x = 0\}$.

Proposition E.2 (Gaussian fluctuation limit): *Let $U \in \mathbb{R}^{N\times(N-1)}$ be an orthonormal basis for the tangent space $\mathrm{T}\Delta_N$. Let $t \in (0,1]$ and let $P_t \sim \mathrm{Dir}(\beta_t(P_0,\pi))$ with $\pi_i > 0$ for all i. Let $X_0 = \sqrt{\varepsilon}U^\top(P_0-\pi)$ and $X_t = \sqrt{\varepsilon}U^\top(P_t-\pi)$. We have*

$$X_t = \alpha_t X_0 + (1-\alpha_t)\xi_t^\varepsilon,$$

where $\xi_t^\varepsilon = \frac{\sqrt{\varepsilon}}{1-\alpha_t}U^\top(P_t - \bar{P}_t)$. Define $\Sigma_t := U^\top\left[\mathrm{diag}(\pi) - \pi\pi^\top + \alpha_t(P_0-\pi)(P_0-\pi)^\top\right]U$. As $\varepsilon \to \infty$,

$$\xi_t^\varepsilon \xrightarrow{d} \mathcal{N}(0,\Sigma_t).$$

In addition, the spectrum of Σ_t is contained in $[\min_i \pi_i, 2]$ for all $t \in (0,1]$.

To connect with standard continuous Gaussian diffusion models, we project the scaled displacement of the forward state from the prior π onto an orthonormal basis of the simplex tangent space.

Proposition E.3 (Asymptotic normality and covariance structure): *Assume that π has strictly positive components. Let $t \in (0,1]$ and let $P_t \sim \mathrm{Dir}(\beta_t(P_0,\pi))$. Define $\bar{P}_t = \alpha_t P_0 + (1-\alpha_t)\pi$ and $V_t = (1-\alpha_t)^{-1}(\mathrm{diag}(\bar{P}_t) - \bar{P}_t\bar{P}_t^\top)$. As $\varepsilon \to \infty$, we have*

$$\sqrt{\varepsilon}(P_t - \bar{P}_t) \xrightarrow{d} \mathcal{N}\left(0,(1-\alpha_t)^2 V_t\right).$$

In addition, letting $V_\pi = \mathrm{diag}(\pi) - \pi\pi^\top$, we have

$$V_t = V_\pi + \alpha_t(P_0-\pi)(P_0-\pi)^\top.$$

Proof. For the rest of the proof, we define $\tilde{V}_t = \mathrm{diag}(\bar{P}_t) - \bar{P}_t\bar{P}_t^\top$. Hence, we have that $\tilde{V}_t = (1-\alpha_t)V_t$. We leverage the Gamma representation of the Dirichlet distribution. Let $\mu_t = \pi + h_t P_0$ with $h_t = \frac{\alpha_t}{1-\alpha_t}$, such that the concentration parameters are $\beta_t^\varepsilon = \varepsilon\mu_t$. We can write $P_t \overset{d}{=} Y^\varepsilon/(\mathbf{1}^\top Y^\varepsilon)$, where Y^ε is a vector of independent random variables $Y_i^\varepsilon \sim \mathrm{Gamma}(\varepsilon\mu_{t,i},1)$.

Using the Central Limit Theorem, as $\varepsilon \to \infty$, we have

$$\sqrt{\varepsilon}\left(\frac{Y^\varepsilon}{\varepsilon} - \mu_t\right) \xrightarrow{d} \mathcal{N}(0,\mathrm{diag}(\mu_t)).$$

We apply the multivariate Delta method with $g(x) = \frac{x}{\mathbf{1}^\top x}$. Let $c_t = \mathbf{1}^\top\mu_t = 1 + h_t = \frac{1}{1-\alpha_t}$. Notice that $g(\mu_t) = \frac{\mu_t}{c_t} = \bar{P}_t$. The Jacobian of g at μ_t is given by

$$\nabla g(\mu_t) = \frac{1}{\mathbf{1}^\top\mu_t}\mathrm{Id} - \frac{\mu_t\mathbf{1}^\top}{(\mathbf{1}^\top\mu_t)^2} = \frac{1}{c_t}\left(\mathrm{Id} - \bar{P}_t\mathbf{1}^\top\right).$$

The asymptotic covariance is $\nabla g(\mu_t)\mathrm{diag}(\mu_t)\nabla g(\mu_t)^\top$ which can be simplified in

$$\begin{aligned}\frac{1}{c_t^2}\left(I-\bar{P}_t\mathbf{1}^\top\right)\mathrm{diag}(\mu_t)\left(I-\mathbf{1}\bar{P}_t^\top\right) &= \frac{1}{c_t}\left(I-\bar{P}_t\mathbf{1}^\top\right)\mathrm{diag}(\bar{P}_t)\left(I-\mathbf{1}\bar{P}_t^\top\right)\\ &= \frac{1}{c_t}\left(\mathrm{diag}(\bar{P}_t)-\bar{P}_t\bar{P}_t^\top-\bar{P}_t\bar{P}_t^\top+\bar{P}_t(\mathbf{1}^\top\bar{P}_t)\bar{P}_t^\top\right)\\ &= \frac{1}{c_t}(\mathrm{diag}(\bar{P}_t)-\bar{P}_t\bar{P}_t^\top)\\ &= (1-\alpha_t)\tilde{V}_t = (1-\alpha_t)^2V_t.\end{aligned}$$

To obtain the second part of the proposition, we expand $\tilde{V}_t = \mathrm{diag}(\bar{P}_t) - \bar{P}_t\bar{P}_t^\top$ using $\bar{P}_t = \alpha_t P_0 + (1-\alpha_t)\pi$. Since $\mathrm{diag}(P_0) = P_0P_0^\top$, because $P_0 = e_{x_0}$, we have

$$\begin{aligned}\tilde{V}_t &= \alpha_t\mathrm{diag}(P_0) + (1-\alpha_t)\mathrm{diag}(\pi) - \left[\alpha_t^2P_0P_0^\top + \alpha_t(1-\alpha_t)(P_0\pi^\top+\pi P_0^\top) + (1-\alpha_t)^2\pi\pi^\top\right]\\ &= (\alpha_t-\alpha_t^2)P_0P_0^\top + (1-\alpha_t)\mathrm{diag}(\pi) - \alpha_t(1-\alpha_t)(P_0\pi^\top+\pi P_0^\top) - (1-\alpha_t)^2\pi\pi^\top\\ &= (1-\alpha_t)\left[\mathrm{diag}(\pi) - \pi\pi^\top + \alpha_tP_0P_0^\top - \alpha_t(P_0\pi^\top+\pi P_0^\top) + \alpha_t\pi\pi^\top\right]\\ &= (1-\alpha_t)\left[V_\pi + \alpha_t(P_0-\pi)(P_0-\pi)^\top\right],\end{aligned}$$

which concludes the proof since $\tilde{V}_t = (1-\alpha_t)V_t$. □

Proposition E.4 (Spectral stability of the projected covariance): *Let $U \in \mathbb{R}^{N\times(N-1)}$ be an orthonormal basis for the tangent space $\mathrm{T}\Delta_N$. If π has strictly positive components, the spectrum of $U^\top V_tU$ is uniformly bounded for all $t \in (0,1]$:*

$$0 < \min_i \pi_i \leqslant \lambda_{\min}(U^\top V_tU) \leqslant \lambda_{\max}(U^\top V_tU) \leqslant 2.$$

Proof. Let $v \in \mathbb{R}^{N-1}$ such that $\|v\|_2 = 1$ and $u = Uv$. It follows that $\|u\|_2 = \|v\|_2 = 1$ and $u^\top\mathbf{1} = 0$. Using the decomposition of V_t from Proposition E.3, we get

$$v^\top(U^\top V_tU)v = u^\top V_\pi u + \alpha_t(u^\top(P_0-\pi))^2. \tag{15}$$

We analyze the two terms in (15) separately. First, we have

$$\begin{aligned}u^\top V_\pi u &= \sum_{i=1}^N \pi_iu_i^2 - \left(\sum_i \pi_iu_i\right)^2\\ &= \sum_{i=1}^N \pi_i(u_i-\pi^\top u)^2\\ &\geqslant \min_i \pi_i\left(1+N(\pi^\top u)^2\right) \geqslant \min_i \pi_i,\end{aligned}$$

where we have used that $u^\top u = 1$ and $u^\top\mathbf{1} = 0$. Combining this result and (15), we get that $\lambda_{\min}(U^\top V_tU) \geqslant \min_i \pi_i > 0$.

For the upper-bound, we use that

$$\lambda_{\max}(U^\top V_tU) \leqslant \mathrm{Tr}(U^\top V_tU) \leqslant \mathrm{Tr}(V_t). \tag{16}$$

Using that $P_0 = e_{x_0}$, we have that

$$\mathrm{Tr}(V_\pi) = 1-\|\pi\|^2, \qquad \mathrm{Tr}((P_0-\pi)(P_0-\pi)^\top) = \|P_0-\pi\|^2 = 1-2\pi^\top P_0+\|\pi\|^2.$$

Therefore, combining this result, $\alpha_t \leqslant 1$ and (16) we get that

$$\lambda_{\max}(U^\top V_tU) \leqslant 1-\|\pi\|^2+1-2\pi^\top P_0+\|\pi\|^2 \leqslant 2,$$

which concludes the proof. □

Finally, we conclude this section with the proof of Proposition E.2.

Proposition E.5 (Gaussian fluctuation limit): *Let $U \in \mathbb{R}^{N\times(N-1)}$ be an orthonormal basis for the tangent space $\mathrm{T}\Delta_N$. Let $t \in (0,1]$ and let $P_t \sim \mathrm{Dir}(\beta_t(P_0,\pi))$ with $\pi_i > 0$ for all i. Let $X_0 = \sqrt{\varepsilon}U^\top(P_0-\pi)$ and $X_t = \sqrt{\varepsilon}U^\top(P_t-\pi)$. We have*

$$X_t = \alpha_t X_0 + (1-\alpha_t)\xi_t^\varepsilon,$$

where $\xi_t^\varepsilon = \frac{\sqrt{\varepsilon}}{1-\alpha_t}U^\top(P_t-\bar{P}_t)$. As $\varepsilon\to\infty$, we have

$$\xi_t^\varepsilon \xrightarrow{d} \mathcal{N}\left(0, U^\top V_t U\right).$$

In addition, the spectrum of $U^\top V_t U$ is contained in $[\min_i \pi_i, 2]$ for all $t \in (0,1]$.

Proof. Scaling the perturbation $P_t - \pi = \alpha_t(P_0-\pi) + (P_t - \bar{P}_t)$ by $\sqrt{\varepsilon}U^\top$ yields the linear form. By Proposition E.3, $\sqrt{\varepsilon}(P_t-\bar{P}_t) \xrightarrow{d} \mathcal{N}\left(0,(1-\alpha_t)^2 V_t\right)$. Applying the linear transformation $\frac{1}{(1-\alpha_t)}U^\top$ directly yields

$$\xi_t^\varepsilon \xrightarrow{d} \mathcal{N}\left(0, \frac{1}{(1-\alpha_t)^2}U^\top(1-\alpha_t)^2 V_t U\right) = \mathcal{N}\left(0, U^\top V_t U\right),$$

which concludes the proof. The second part of the proof is a direct consequence of Proposition E.4. □

F Generalization of DDIM

In this section, we provide a general framework for the derivation of Proposition 3.3. Our main goal is to provide a general, affine-state-space, DDIM-style bridge framework (Song et al., 2021a). In fact, we show that the notion of *perturbed state*, *innovation* and *interpolation* random variables can be recovered for categorical, simplex and continuous modalities.

F.1 General State-Space DDIM

Let us consider a standard Borel subset $\mathcal{X}$ of a real vector space and a distribution Q_θ with $\theta \in \Theta$ and $\Theta = \{(a,b) \in [0,\infty)^2 : a+b>0\}$. Arithmetic operations and inequalities on parameters are understood componentwise.

We define $c : \Theta \to \mathbb{R}$ such that $c((a,b)) = a+b$. Next, define the family of distributions B_{c_1,c_2} with c_1, c_2 on $[0,1]$. We assume that we have the following compatibility conditions:

- **Normalization:** For any $c_1, c_2 > 0$, if $W \sim \mathsf{B}_{c_1,c_2}$ then $\mathbb{E}[W] = \frac{c_1}{c_1+c_2}$.
- **Closure:** For any $\theta_1, \theta_2 \in \Theta$, $X_1 \sim \mathsf{Q}_{\theta_1}$, $X_2 \sim \mathsf{Q}_{\theta_2}$ and $W \sim \mathsf{B}_{c(\theta_1),c(\theta_2)}$ mutually independent, we have that $Z = WX_1 + (1-W)X_2 \sim \mathsf{Q}_{\theta_1+\theta_2}$, where the addition is pointwise.

Based on those definitions, we now introduce the *thinning kernel*.

Definition F.1 (Thinning kernel): *Let $\rho \in (0,1)$, $\theta \in \Theta$ and let $X \sim \mathsf{Q}_{\rho\theta}$, $Y \sim \mathsf{Q}_{(1-\rho)\theta}$ and $W \sim \mathsf{B}_{\rho c(\theta),(1-\rho)c(\theta)}$, all independent random variables, and define $Z = WX + (1-W)Y$. We define $\mathsf{T}_{\rho,\theta}(\cdot \mid z) := \mathcal{L}(X \mid Z = z)$, the conditional distribution of X given Z and call it the* thinning kernel.

By the closure property, if $Z \sim \mathsf{Q}_\theta$ and $X \sim \mathsf{T}_{\rho,\theta}(\cdot \mid Z)$, then X has distribution $\mathsf{Q}_{\rho\theta}$. With those definitions we have immediately the following property.

Proposition F.1 (Reverse Bridge): *Let $s, t \in [0,1]$, $\theta_s, \theta_t \in \Theta$, and choose $\rho \in (0,1]$ such that $\theta_s - \rho\theta_t$ is coordinatewise non-negative. If $\theta_s = \rho\theta_t$, we use the boundary convention $W^\rho_{s,t} = 1$ and omit $V^\rho_{s,t}$. Denote $c_t = c(\theta_t)$ and $c_s = c(\theta_s)$. Let $X_t \in \mathcal{X}$ and define $X^\rho_{s,t} \sim \mathsf{T}_{\rho,\theta_t}(\cdot \mid X_t)$, the* perturbed random variable. *Then, let $V^\rho_{s,t} \sim \mathsf{Q}_{\theta_s - \rho\theta_t}$ be the* innovation random variable, *$W^\rho_{s,t} \sim \mathsf{B}_{\rho c_t, c_s - \rho c_t}$ the* interpolation random variable, *independently of $X^\rho_{s,t}$, and*

$$X_s = W^\rho_{s,t} X^\rho_{s,t} + (1 - W^\rho_{s,t}) V^\rho_{s,t}. \tag{17}$$

If $X_t \sim \mathsf{Q}_{\theta_t}$ then $X_s \sim \mathsf{Q}_{\theta_s}$.

Indeed, by definition of the thinning kernel $X^\rho_{s,t} \sim \mathsf{Q}_{\rho\theta_t}$ marginally, and the conclusion follows immediately from the closure property.

We now specialise Proposition F.1 to different settings.

F.2 Simplex Transitions

In the simplicial setting, Proposition 3.3 is an instantiation of Proposition F.1. We consider $P_0 \in \Delta_N$. We define the forward distribution Q_θ as follows: for any $\theta = (a, b)$ with $a > 0$ and $b > 0$ we have

$$\mathsf{Q}_{(a,b)} = \mathrm{Dir}(aP_0 + b\pi), \tag{18}$$

with $\pi \in \Delta_N$. Note that $c = a + b$ is the concentration parameter of the Dirichlet random variable. Similarly, we define the family of mixing distributions B_{c_1,c_2} such that

$$\mathsf{B}_{c_1,c_2} = \mathrm{Beta}(c_1, c_2), \tag{19}$$

with $c_1, c_2 > 0$. Note that $\{\mathsf{B}_{c_1,c_2}\}_{c_1,c_2>0}$ and $\{\mathsf{Q}_\theta\}_{\theta\in\Theta}$ satisfy the normalization property using properties of the Beta random variable, and the closure property using Proposition A.1.

Let us derive the thinning operator in that case. Note that the following proposition can be seen as the reciprocal of Proposition A.2

Proposition F.2 (Dirichlet thinning kernel): *Let $\rho \in (0,1)$, $\theta = (a, b) \in \Theta$ and $\mathsf{T}_{\rho,\theta}$ given by Definition F.1 with forward given by (18) and mixing given by (19). Write $\eta(\theta) := aP_0 + b\pi$. Let $\{B_i\}_{i=1}^N$ be mutually independent random variables such that for any $i \in \{1, \dots, N\}$, $B_i \sim \mathrm{Beta}(\rho\eta_i(\theta), (1-\rho)\eta_i(\theta))$ and define $Y(x) \in \Delta_N$ such that for any $i \in \{1, \dots, N\}$ and $x \in \Delta_N$*

$$Y_i(x) = \frac{B_i x_i}{\sum_{j=1}^N B_j x_j},$$

Then, almost everywhere on Δ_N, $Y(x) \sim \mathsf{T}_{\rho,\theta}(\cdot|x)$.

Proof. We denote $F : \Delta_N \times [0,1]^N \to \Delta_N$ such that for any $p \in \Delta_N$ and $\{b_i\}_{i=1}^N$ with $b_i \in [0,1]$ we have

$$F(p, \{b_i\}_{i=1}^N) = \left\{ \frac{b_i p_i}{\sum_{j=1}^N b_j p_j} \right\}_{i=1}^N.$$

Let $\{B^1_i\}_{i=1}^N$ and $\{B^2_i\}_{i=1}^N$ be mutually independent such that $B^1_i \sim \mathrm{Gamma}(\rho(aP_{0,i} + b\pi_i), 1)$, where $\theta = (a, b)$ and $B^2_i \sim \mathrm{Gamma}((1-\rho)(aP_{0,i} + b\pi_i), 1)$. Next, for any $i \in \{1, \dots, N\}$ we define

$$X_i = \frac{B^1_i}{\sum_{j=1}^N B^1_j}, \qquad Y_i = \frac{B^2_i}{\sum_{j=1}^N B^2_j}.$$

We have that $X \sim \mathrm{Dir}(\rho(aP_0 + b\pi))$ and $Y \sim \mathrm{Dir}((1-\rho)(aP_0 + b\pi))$. In addition, X, Y, S_X and S_Y are independent where

$$S_X = \sum_{j=1}^N B^1_j, \qquad S_Y = \sum_{j=1}^N B^2_j.$$

In particular, defining $W = S_X/(S_X + S_Y)$ we have that $W \sim \mathrm{Beta}(\rho c(\theta), (1-\rho)c(\theta))$ and (X, Y, W) are mutually independent. Therefore, we can define $Z = WX + (1-W)Y$ as in Definition F.1. Note that we have for any $i \in \{1, \dots, N\}$

$$Z_i = \frac{B_i^1 + B_i^2}{\sum_{j=1}^N (B_j^1 + B_j^2)}. \tag{20}$$

Next, we denote $\{B_i^3\}_{i=1}^N$ and $\{B_i^4\}_{i=1}^N$ such that for any $i \in \{1, \dots, N\}$

$$B_i^3 = B_i^1 + B_i^2, \qquad B_i^4 = B_i^1/(B_i^1 + B_i^2).$$

We have that $B_i^3 \sim \mathrm{Gamma}(\eta_i(\theta), 1)$ and $B_i^4 \sim \mathrm{Beta}(\rho\eta_i(\theta), (1-\rho)\eta_i(\theta))$ for any $i \in \{1, \dots, N\}$. We have that $\{B_i^3\}_{i=1}^N$ and $\{B_i^4\}_{i=1}^N$ are independent. Define $B = \{B_i^4\}_{i=1}^N$ and observe that using (20) we have that B and Z are independent. Then, note that

$$B_i Z_i = \frac{B_i^1}{\sum_{j=1}^N (B_j^1 + B_j^2)}, \qquad \sum_{j=1}^N B_j Z_j = \frac{\sum_{j=1}^N B_j^1}{\sum_{j=1}^N (B_j^1 + B_j^2)}.$$

Therefore, we have that for any $i \in \{1, \dots, N\}$

$$X_i = \frac{B_i Z_i}{\sum_{j=1}^N B_j Z_j} = F(Z, B)_i.$$

Since $X = F(Z, B)$ and B is independent of Z, the conditional law of X given $Z = z$ is exactly the law of $F(z, B)$. By Definition F.1, this is $\mathsf{T}_{\rho,\theta}(\cdot \mid z)$, which proves the result (for Q_θ-almost every z). □

Note that Proposition F.2 shows that the thinning procedure we identified in Proposition 3.3 is exactly what we need to define the general thinning kernel when specialised to the simplicial framework. Next, given the forward family (18), the mixing family (19) and the thinning kernel Proposition F.2 we are ready to apply Proposition F.1.

Let us consider $\theta_s = (a_s, b_s)$ and $\theta_t = (a_t, b_t)$ and choose ρ such that $a_s - \rho a_t \geqslant 0$ and $b_s - \rho b_t \geqslant 0$. The first part of Proposition F.1 defines

$$(X_{s,t}^\rho)_i = \frac{B_i X_{t,i}}{\sum_{j=1}^N B_j X_{t,j}}, \tag{21}$$

with $\{B_i\}_{i=1}^N$ mutually independent random variables such that for any $i \in \{1, \dots, N\}$, $B_i \sim \mathrm{Beta}(\rho\beta_t(P_0, \pi)_i, (1-\rho)\beta_t(P_0, \pi)_i)$, according to Proposition F.2. We also have that

$$W_{s,t}^\rho \sim \mathrm{Beta}(\rho c_t, c_s - \rho c_t).$$

Finally, we have that

$$V_{s,t}^\rho \sim \mathrm{Dir}((a_s - \rho a_t)P_0 + (b_s - \rho b_t)\pi). \tag{22}$$

Then, defining X_s as in (17) with

$$X_s = W_{s,t}^\rho X_{s,t}^\rho + (1 - W_{s,t}^\rho)V_{s,t}^\rho, \tag{23}$$

we have that $X_s \sim \mathrm{Dir}(a_s P_0 + b_s \pi)$ if $X_t \sim \mathrm{Dir}(a_t P_0 + b_t \pi)$.

Now, we compare our general procedure to Proposition 3.3. Recall that $\beta_t(P_0, \pi) = c_t(\alpha_t P_0 + (1-\alpha_t)\pi)$. Let $a_t = c_t\alpha_t$ and $b_t = c_t(1-\alpha_t)$ and we define $\theta_t = (a_t, b_t)$. Note that $\mathrm{Dir}(\beta_t(P_0, \pi)) = \mathrm{Dir}(a_t P_0 + b_t \pi)$. We have that $c(\theta_t) = c_t$. The admissibility condition is $\rho a_t \leqslant a_s$ and $\rho b_t \leqslant b_s$. Since $a_u/b_u = \alpha_u/(1-\alpha_u)$ is non-increasing in u, the first inequality is implied by the second for $s < t$; hence the largest admissible value is $r_{s,t} = \min\{1, b_s/b_t\}$. Let $\rho = \rho_{s,t}^\kappa = (1-\kappa)r_{s,t}$. So, using (3), we have that $\mathsf{Q}_{\theta_t} = p_{t|0}$. We have that (21) corresponds to (6), while (22) corresponds to (7) and (23) corresponds to (8). Therefore, we recover Proposition 3.3 and its conclusion.

F.3 Gaussian Transition

In the Gaussian setting we can also obtain the standard DDIM-style transition step Song et al. (2021a) using Proposition F.1. We define the forward kernel Q_θ as follows: for any $a, b > 0$ we define

$$\mathsf{Q}_\theta = \mathcal{N}\left(\frac{ax_0}{c}, \frac{b\mathrm{Id}}{c^2}\right), \tag{24}$$

with $\Theta = \{(a, b) \in [0, \infty)^2 : \ a + b > 0\}$ and $c = a + b$. Note that one can recover the classical flow matching or Rectified Flow (Lipman et al., 2023; Liu et al., 2023c) by setting $a_t = \alpha_t/(1 - \alpha_t)$ and $b_t = 1$. Indeed, in that case we have $a_t + b_t = 1/(1-\alpha_t)$ and therefore we have setting $\theta_t = (a_t, b_t)$

$$\mathsf{Q}_{\theta_t} = \mathcal{N}(\alpha_t x_0, (1 - \alpha_t)^2 \mathrm{Id}). \tag{25}$$

Similarly, we define the family of mixing distributions B_{c_1,c_2} such that for any $c_1, c_2 > 0$

$$\mathsf{B}_{c_1,c_2} = \delta_{c_1/(c_1+c_2)}.$$

The normalization property is immediate. For the closure property, we note that for any $\theta_1, \theta_2 \in \Theta$, $X_1 \sim \mathsf{Q}_{\theta_1}$, $X_2 \sim \mathsf{Q}_{\theta_2}$ and $W \sim \mathsf{B}_{c(\theta_1),c(\theta_2)}$ we have that $Z = WX_1 + (1 - W)X_2$ is Gaussian. We have that

$$\mathbb{E}[Z] = \frac{a_1}{c_1}\frac{c_1}{c_1 + c_2} + \frac{a_2}{c_2}\frac{c_2}{c_1 + c_2} = \frac{a_1 + a_2}{c_1 + c_2}.$$

Similarly we have that

$$\mathrm{Cov}[Z] = \frac{b_1}{c_1^2}\frac{c_1^2}{(c_1 + c_2)^2} + \frac{b_2}{c_2^2}\frac{c_2^2}{(c_1 + c_2)^2} = \frac{b_1 + b_2}{(c_1 + c_2)^2},$$

which concludes the proof. Then, we can derive the thinning kernel in the Gaussian setting.

Proposition F.3 (Gaussian thinning kernel): *Let $\rho \in (0, 1)$, $\theta = (a, b) \in \Theta$ and $\mathsf{T}_{\rho,\theta}$ given by Definition F.1 with forward given by (24) and mixing given by (25). Then, we have that for any $x \in \mathbb{R}^d$*

$$\mathsf{T}_{\rho,\theta}(\cdot|x) = \mathcal{N}\left(x, \frac{(1 - \rho)b}{\rho c^2}\mathrm{Id}\right).$$

Note that when $\rho = 1$ we have that $\mathsf{T}_{\rho,\theta}(\cdot|x) = \delta_x$ as in Proposition F.2.

Proof. Let $\rho \in (0, 1)$, $\theta \in \Theta$ and let $X \sim \mathsf{Q}_{\rho\theta}$, $Y \sim \mathsf{Q}_{(1-\rho)\theta}$ and $W \sim \mathsf{B}_{\rho c(\theta),(1-\rho)c(\theta)}$, all independent random variables and define $Z = WX + (1 - W)Y$. Next, let $R = X - Z$, we have that

$$\begin{aligned}\mathrm{Cov}[R, Z] &= \mathrm{Cov}[X, Z] - \mathrm{Cov}[Z] \\ &= \rho\,\mathrm{Cov}[X] - \mathrm{Cov}[Z] \\ &= \rho\frac{b}{\rho c^2}\mathrm{Id} - \frac{b}{c^2}\mathrm{Id} = 0.\end{aligned}$$

Since R and Z are Gaussian we have that R and Z are independent. Now, we have that

$$\mathbb{E}[R] = \frac{\rho a}{\rho c} - \frac{a}{c} = 0$$

In addition, we have that

$$\mathrm{Cov}[R] = \mathrm{Cov}[X] + \mathrm{Cov}[Z] - 2\,\mathrm{Cov}[X, Z] = \frac{b}{\rho c^2}\mathrm{Id} + \frac{b}{c^2}\mathrm{Id} - 2\frac{b}{c^2}\mathrm{Id} = \frac{b(1 - \rho)}{\rho c^2}\mathrm{Id}.$$

We conclude the proof upon using that $X = Z + R$. □

Next, given the forward family (24), the mixing family (25) and the thinning kernel Proposition F.3 we are ready to apply Proposition F.1. Let us consider $\theta_s = (a_s, b_s)$ and $\theta_t = (a_t, b_t)$ and choose ρ such that $a_s - \rho a_t \geqslant 0$ and $b_s - \rho b_t \geqslant 0$. The first part of Proposition F.1 defines

$$X^\rho_{s,t} = X_t + \sqrt{\frac{(1-\rho)b_t}{\rho(a_t+b_t)^2}} Z_1,$$

with $Z_1 \sim \mathcal{N}(0, \mathrm{Id})$ a d-dimensional Gaussian random variable with identity covariance matrix. We also have that

$$W^\rho_{s,t} = \frac{\rho(a_t+b_t)}{a_s+b_s}.$$

Finally, we have that

$$V^\rho_{s,t} = \frac{a_s - \rho a_t}{(a_s+b_s)-\rho(a_t+b_t)} x_0 + \frac{\sqrt{b_s - \rho b_t}}{(a_s+b_s)-\rho(a_t+b_t)} Z_2,$$

with $Z_2 \sim \mathcal{N}(0, \mathrm{Id})$ a d-dimensional Gaussian random variable with identity covariance matrix independent from Z_1. Then, defining X_s as in (17) with

$$X_s = W^\rho_{s,t} X^\rho_{s,t} + (1 - W^\rho_{s,t}) V^\rho_{s,t},$$

we have that $X_s \sim \mathcal{N}\left(\frac{a_s x_0}{c_s}, \frac{b_s \mathrm{Id}}{c_s^2}\right)$ if $X_t \sim \mathcal{N}\left(\frac{a_t x_0}{c_t}, \frac{b_t \mathrm{Id}}{c_t^2}\right)$, where we recall that $c_s = a_s + b_s$ and $c_t = a_t + b_t$.

Let us specialize the above derivations to the case of flow matching or Rectified Flow (Lipman et al., 2023; Liu et al., 2023c) by setting $a_t = \alpha_t/(1-\alpha_t)$ and $b_t = 1$.

In that case, we have that

$$X^\rho_{s,t} = X_t + (1-\alpha_t)\sqrt{\frac{1-\rho}{\rho}} Z_1.$$

In addition, we have that

$$W^\rho_{s,t} = \rho \frac{1-\alpha_s}{1-\alpha_t}.$$

Finally, we get that

$$V^\rho_{s,t} = \frac{\alpha_s - \rho\alpha_t - (1-\rho)\alpha_s\alpha_t}{\rho\alpha_s - \alpha_t + 1 - \rho} x_0 + \frac{(1-\alpha_s)(1-\alpha_t)\sqrt{1-\rho}}{\rho\alpha_s - \alpha_t + 1 - \rho} Z_2.$$

One can observe that if $\rho = 1$ we have $X^\rho_{s,t} = X_t$ and $V^\rho_{s,t} = x_0$. Similarly, if $\rho = 0$ we have that

$$V^\rho_{s,t} = \alpha_s x_0 + (1-\alpha_s) Z_2,$$

and therefore $V^\rho_{s,t} \sim \mathsf{Q}_{\theta_s}$.

F.4 Categorical Transition

In the categorical setting we can also obtain the remasking transition step Wang et al. (2025) using Proposition F.1. In this subsection we identify categorical states with their one-hot vectors. One way to obtain those results is to consider $(\varepsilon a, \varepsilon b)$ in the simplicial setting and let $\varepsilon \to 0$. However, for completeness, we derive the categorical framework from scratch.

We define the forward kernel Q_θ as follows: for any $a, b > 0$ let

$$\mathsf{Q}_\theta = \mathrm{Cat}\left(\frac{a x_0 + b\pi}{a+b}\right), \tag{26}$$

with $\Theta = \{(a,b) \in [0,\infty)^2 : a + b > 0\}$ and $c = a + b$. Note that by setting $\theta_t = (a_t, b_t)$ with $a_t = \alpha_t/(1-\alpha_t)$ and $b_t = 1$ we recover the original forward transition probability of discrete diffusion models Sohl-Dickstein et al. (2015); Austin et al. (2021); Campbell et al. (2022); Sahoo et al. (2024); Shi et al. (2024); Ou et al. (2025). Note that another possible choice would be $a_t = \alpha_t$ and $b_t = 1 - \alpha_t$, however this would yield a different mixing kernel.

We define the family of mixing distributions B_{c_1,c_2} such that for any $c_1, c_2 > 0$

$$\mathsf{B}_{c_1,c_2} = \mathrm{Ber}\left(\frac{c_1}{c_1+c_2}\right). \tag{27}$$

The normalization property is immediate. For the closure property, we note that for any $\theta_1, \theta_2 \in \Theta$, $X_1 \sim \mathsf{Q}_{\theta_1}$, $X_2 \sim \mathsf{Q}_{\theta_2}$ and $W \sim \mathsf{B}_{c(\theta_1),c(\theta_2)}$ we have

$$\begin{aligned}\mathbb{P}(Z = e_k) &= \frac{c_1}{c_1+c_2}\frac{a_1x_0 + b_1\pi}{c_1} + \frac{c_2}{c_1+c_2}\frac{a_2x_0+b_2\pi}{c_2} \\ &= \frac{(a_1+a_2)x_0 + (b_1+b_2)\pi}{c_1+c_2},\end{aligned}$$

which concludes the proof. Then, we can derive the thinning kernel in the categorical setting.

Proposition F.4 (Categorical thinning kernel): *Let $\rho \in (0,1)$, $\theta = (a,b) \in \Theta$ and $\mathsf{T}_{\rho,\theta}$ given by Definition F.1 with forward given by (26) and mixing given by (27). Then, we have that for any $x \in \{e_1, \dots, e_N\}$*

$$\mathsf{T}_{\rho,\theta}(\cdot|x) = \mathrm{Cat}\left(\rho x + (1-\rho)\frac{ax_0 + b\pi}{a+b}\right).$$

Note that when $\rho = 1$ we have that $\mathsf{T}_{\rho,\theta}(\cdot|x) = \delta_x$ as in Proposition F.2 and Proposition F.3.

Proof. Let $\rho \in (0,1)$, $\theta \in \Theta$ and let $X \sim \mathsf{Q}_{\rho\theta}$, $Y \sim \mathsf{Q}_{(1-\rho)\theta}$ and $W \sim \mathsf{B}_{\rho c(\theta),(1-\rho)c(\theta)}$, all independent random variables and define $Z = WX + (1-W)Y$. We have that for any $i, j \in \{1, \dots, N\}$

$$\begin{aligned}\mathbb{P}(Z = e_i, X = e_j) &= \mathbb{P}(X = e_j)\mathbb{P}(Z = e_i | X = e_j) \\ &= \frac{ax_{0,j} + b\pi_j}{a+b}\left(\rho\delta_i(j) + (1-\rho)\frac{ax_{0,i}+b\pi_i}{a+b}\right).\end{aligned}$$

In addition, we have that

$$\mathbb{P}(Z = e_i) = \frac{ax_{0,i} + b\pi_i}{a+b}.$$

Therefore, we get that

$$\mathbb{P}(X = e_j | Z = e_i) = (1-\rho)\frac{ax_{0,j}+b\pi_j}{a+b} + \rho\frac{ax_{0,j}+b\pi_j}{ax_{0,i}+b\pi_i}\delta_i(j).$$

This can further simplified in

$$\mathbb{P}(X = e_j | Z = e_i) = \rho\delta_i(j) + (1-\rho)\frac{ax_{0,j}+b\pi_j}{a+b},$$

which concludes the proof. □

Next, given the forward family (26), the mixing family (27) and the thinning kernel Proposition F.4 we are ready to apply Proposition F.1. Let us consider $\theta_s = (a_s, b_s)$ and $\theta_t = (a_t, b_t)$ and choose ρ such that $a_s - \rho a_t \geqslant 0$ and $b_s - \rho b_t \geqslant 0$. The first part of Proposition F.1 defines

$$X^\rho_{s,t} \sim \mathrm{Cat}\left(\rho X_t + (1-\rho)\frac{a_t x_0 + b_t\pi}{a_t+b_t}\right),$$

We also have that

$$W^\rho_{s,t} \sim \mathrm{Ber}\left(\rho\frac{a_t+b_t}{a_s+b_s}\right).$$

Finally, we have that

$$V^\rho_{s,t} \sim \mathrm{Cat}\left(\frac{a_s-\rho a_t}{a_s+b_s-\rho(a_t+b_t)}x_0 + \frac{b_s-\rho b_t}{a_s+b_s-\rho(a_t+b_t)}\pi\right),$$

Then, defining X_s as in (17) with

$$X_s = W^\rho_{s,t} X^\rho_{s,t} + (1 - W^\rho_{s,t}) V^\rho_{s,t},$$

we have that $X_s \sim \mathrm{Cat}(\frac{a_s x_0 + b_s \pi}{a_s + b_s})$ if $X_t \sim \mathrm{Cat}(\frac{a_t x_0 + b_t \pi}{a_t + b_t})$.

Let us now specialize the above to $a_t = \alpha_t/(1-\alpha_t)$ and $b_t = 1$. In that case, we have that

$$X^\rho_{s,t} \sim \mathrm{Cat}(\rho x_t + (1-\rho)(\alpha_t x_0 + (1-\alpha_t)\pi)).$$

In addition, we have that

$$W^\rho_{s,t} \sim \mathrm{Ber}\left(\rho \frac{1-\alpha_s}{1-\alpha_t}\right).$$

Finally, we have that

$$V^\rho_{s,t} \sim \mathrm{Cat}\left(\frac{\alpha_s - \rho\alpha_t - (1-\rho)\alpha_s\alpha_t}{\rho\alpha_s - \alpha_t + 1 - \rho} x_0 + \frac{(1-\rho)(1-\alpha_t)(1-\alpha_s)}{\rho\alpha_s - \alpha_t + 1 - \rho}\pi\right).$$

In the specific case where π is a one hot masking vector M, we can draw an explicit connection between the remasking parameter of Wang et al. (2025) and our parameter ρ.

Proposition F.5 (Connection with remasking): *If $x_t \neq \mathrm{M}$ then we have that*

$$X_s \sim \mathrm{Cat}\left((\alpha_s + \rho^2(1-\alpha_s))x_0 + (1-\rho^2)(1-\alpha_s)\mathrm{M}\right).$$

If $x_t = \mathrm{M}$, then we have that

$$X_s \sim \mathrm{Cat}\left(\left(\alpha_s - \rho^2\frac{\alpha_t(1-\alpha_s)}{1-\alpha_t}\right)x_0 + \left(1 - \alpha_s + \rho^2\frac{\alpha_t(1-\alpha_s)}{1-\alpha_t}\right)\mathrm{M}\right).$$

Setting $\sigma_t = (1-\rho^2)(1-\alpha_s)$[a], we get that if $x_t \neq \mathrm{M}$ then we have that

$$X_s \sim \mathrm{Cat}\left((1-\sigma_t)x_0 + \sigma_t \mathrm{M}\right).$$

If $x_t = \mathrm{M}$, then we have that

$$X_s \sim \mathrm{Cat}\left(\frac{\alpha_s - (1-\sigma_t)\alpha_t}{1-\alpha_t}x_0 + \frac{1 - \alpha_s - \sigma_t\alpha_t}{1-\alpha_t}\mathrm{M}\right).$$

[a] Following to the convention of Wang et al. (2025) we do not highlight the dependency with respect to s.

F.5 Summary Table

In Table 2, we summarize our findings. In particular, we highlight the perturbation, the interpolation weights and the innovation random variables in each of the three settings discussed above (simplicial, Gaussian and categorical).

G Variational Lower Bound

We derive a variational lower bound for the inference procedure described in Section 3.2.

Fix a grid $0 = t_0 < t_1 < \cdots < t_M = 1$ and write $P_k = P_{t_k}$. For $j \in \mathcal{X}$ and $k = 1, \ldots, M-1$, define

$$R^j_k(P_k \mid P_{k+1}) = p_{t_k|0,t_{k+1}}(P_k \mid e_j, P_{k+1}).$$

The learned reverse kernel is

$$K^\theta_k(P_k \mid P_{k+1}) = \sum_{j=1}^N \hat{P}_{\theta,j}(t_{k+1}, P_{k+1}) R^j_k(P_k \mid P_{k+1}), \tag{28}$$

and the endpoint decoder is $p_\theta(x \mid P_1) = \hat{P}_\theta(t_1, P_1)_x$.

Table 2: **Unified DDIM Framework across Modalities.** Comparison of the abstract variables from Proposition F.1 instantiated in the Simplex, Gaussian (Flow Matching), and Categorical (ReMDM) settings with canonical parametrization $a_t = \alpha_t/(1-\alpha_t)$ and $b_t = 1$, yielding $c_t = 1/(1-\alpha_t)$.

Component	General Framework	Simplex (Δ_N)	Gaussian ($\mathbb{R}^d$)	Categorical
Space $\mathcal{X}$	Abstract $\mathcal{X}$	Simplex Δ_N	Euclidean $\mathbb{R}^d$	Discrete $\{e_1, \dots, e_N\}$
Forward Q_θ	$\mathsf{Q}_{(a,b)}$	$\mathrm{Dir}(ax_0 + b\pi)$	$\mathcal{N}\left(\frac{a}{c}x_0, \frac{b}{c^2}\mathrm{Id}\right)$	$\mathrm{Cat}\left(\frac{ax_0+b\pi}{c}\right)$
Capacity $c(\theta)$	$a+b$	Concentration c_t	$\frac{1}{1-\alpha_t}$	$\frac{1}{1-\alpha_t}$
Mixing B_{c_1,c_2}	Law on $[0,1]$	$\mathrm{Beta}(c_1, c_2)$	$\delta_{c_1/(c_1+c_2)}$	$\mathrm{Ber}\left(\frac{c_1}{c_1+c_2}\right)$
Perturbed $X^\rho_{s,t}$	$\mathsf{T}_{\rho,\theta_t}(\cdot \mid X_t)$	$\frac{B \odot X_t}{\langle B, X_t\rangle}$	$X_t + (1-\alpha_t)\sqrt{\frac{1-\rho}{\rho}}Z_1$	$\mathrm{Cat}(\rho X_t + (1-\rho)\mathbb{E}[X_t])$
Interpolation $W^\rho_{s,t}$	$\mathsf{B}_{\rho c_t, c_s - \rho c_t}$	$\mathrm{Beta}(\rho c_t, c_s - \rho c_t)$	$\rho\frac{1-\alpha_s}{1-\alpha_t}$	$\mathrm{Ber}\left(\rho\frac{1-\alpha_s}{1-\alpha_t}\right)$
Mean $\mathbb{E}[W^\rho_{s,t}]$	$\rho\frac{c_t}{c_s}$	$\rho\frac{c_t}{c_s} = \rho\frac{1-\alpha_s}{1-\alpha_t}$	$\rho\frac{1-\alpha_s}{1-\alpha_t}$	$\rho\frac{1-\alpha_s}{1-\alpha_t}$
Innovation $V^\rho_{s,t}$	$\mathsf{Q}_{\theta_s - \rho\theta_t}$	$\mathrm{Dir}\big((a_s - \rho a_t)x_0 + (b_s - \rho b_t)\pi\big)$	$\mathcal{N}(\mu_{\text{innov}}, \sigma^2_{\text{innov}}\mathrm{Id})$	$\mathrm{Cat}\big(p_{\text{innov}}x_0 + (1-p_{\text{innov}})\pi\big)$

For any $x \in \mathcal{X}$, define the bridge path law

$$Q_x(P_{1:M}) := \mathrm{Dir}(P_M; c_1\pi) \prod_{k=1}^{M-1} R^x_k(P_k \mid P_{k+1}).$$

Because $\alpha_1 = 0$, the terminal marginal is $\mathrm{Dir}(c_1\pi) = \mathrm{Dir}(\beta_{t_M}(e_x, \pi))$. One can easily check by backward induction that

$$P_k \sim \mathrm{Dir}(\beta_{t_k}(e_x, \pi)) \quad \text{under } Q_x, \qquad k = 1, \dots, M. \tag{29}$$

Proposition G.1 (Cross-Entropy ELBO): *Let $p_\theta(x)$ denote the marginal likelihood of the fixed-grid sampler with terminal law $\mathrm{Dir}(c_1\pi)$, reverse kernels (28), and endpoint decoder $\hat{P}_\theta(t_1, P_1)$. For every $x_0 \in \mathcal{X}$,*

$$\log p_\theta(x_0) \geqslant \mathcal{L}_{\mathrm{ELBO}}(\theta; x_0),$$

where

$$\mathcal{L}_{\mathrm{ELBO}}(\theta; x_0) = \sum_{k=1}^{M} \mathbb{E}_{P_k \sim \mathrm{Dir}(\beta_{t_k}(e_{x_0}, \pi))} \left[\log \hat{P}_\theta(t_k, P_k)_{x_0}\right]. \tag{30}$$

Proof. Since (28) is a nonnegative mixture, for every P_{k+1},

$$K^\theta_k(P_k \mid P_{k+1}) \geqslant \hat{P}_\theta(t_{k+1}, P_{k+1})_{x_0} R^{x_0}_k(P_k \mid P_{k+1}).$$

Retaining this single component at every reverse step gives

$$p_\theta(x_0) \geqslant \mathbb{E}_{Q_{x_0}} \left[\prod_{k=1}^{M} \hat{P}_\theta(t_k, P_k)_{x_0}\right].$$

Taking logarithms and applying Jensen's inequality,

$$\begin{aligned} \log p_\theta(x_0) &\geqslant \log \mathbb{E}_{Q_{x_0}} \left[\prod_{k=1}^{M} \hat{P}_\theta(t_k, P_k)_{x_0}\right] \\ &\geqslant \sum_{k=1}^{M} \mathbb{E}_{Q_{x_0}} \left[\log \hat{P}_\theta(t_k, P_k)_{x_0}\right]. \end{aligned}$$

Using the marginals (29) proves (30). □

Thus, sampling k uniformly from the inference grid and then sampling $P_k \sim \mathrm{Dir}(\beta_{t_k}(e_{x_0}, \pi))$ gives an unbiased estimator of $-\mathcal{L}_{\mathrm{ELBO}}(\theta; x_0)/M$.

Alternative bound. Keeping the full mixture before applying Jensen gives a tighter but generally intractable mixture-KL bound, which we do not consider further.

H A Pure-Jump Representation

Throughout this section, we restrict ourselves to the concentration schedule $c_t = \varepsilon/(1-\alpha_t)$. Under this parameterization,

$$\beta_t(P_0, \pi) = \varepsilon\,(\pi + h_t P_0)\,, \qquad h_t = \frac{\alpha_t}{1-\alpha_t}.$$

H.1 Reminders on Lévy and Additive Processes

Let $(Z_t)_{t\geqslant 0}$ be a stochastic process. We say that $(Z_t)_{t\geqslant 0}$ is an *additive process* (or inhomogeneous Lévy process) if $Z_0 = 0$, for any $n \in \mathbb{N}$, $t_0 \leqslant \cdots \leqslant t_n$, the increments $(Z_{t_{i+1}} - Z_{t_i})_{i=0}^{n-1}$ are mutually independent random variables, and for any $c > 0$ and $t \geqslant 0$, $\lim_{h\to 0} \mathbb{P}(|Z_{t+h} - Z_t| \geqslant c) = 0$, i.e., $(Z_t)_{t\geqslant 0}$ is continuous in probability. Note that contrary to classical definitions of Lévy processes, we do not assume that the process has stationary increments.

Assume that $(Z_t)_{t\geqslant 0}$ is an additive process. Then, when it is well-defined, there exists a cumulant generating function η such that for any $t \geqslant 0$ and $\theta \in \mathbb{R}$

$$\mathbb{E}[\exp(\theta Z_t)] = \exp\left[\int_0^t \eta_s(\theta)\mathrm{d}s\right].$$

Note that, to simplify the exposition, we consider the Moment Generating Function (MGF) here and not the characteristic function, which is the usual introduction of the Lévy–Khintchine formula. When it is well-defined, there exists (b_t, σ_t, ν_t) for any $t \geqslant 0$ such that for any $\theta \in \mathbb{R}$

$$\eta_t(\theta) = b_t\theta + \frac{\sigma_t^2\theta^2}{2} + \int_{\mathbb{R}} (\exp(\theta x) - 1 - \theta x \mathbf{1}_{[-1,1]}(x))\nu_t(\mathrm{d}x).$$

The term $\theta x \mathbf{1}_{[-1,1]}(x)$ is known as a *compensator*. If $\sigma_t = 0$ and $\int_{|x|\leqslant 1} |x|\nu_t(\mathrm{d}x) < +\infty$ (with the corresponding time-integrability condition on compact intervals), then the jump part has finite variation. After absorbing the compensator into the drift, one may write, whenever the expression is well defined, for any $t \geqslant 0$ and $\theta \in \mathbb{R}$,

$$\eta_t(\theta) = b_t\theta + \int_{\mathbb{R}} (\exp(\theta x) - 1)\nu_t(\mathrm{d}x).$$

For any $s, t \geqslant 0$ with $s \leqslant t$ and $\theta \in \mathbb{R}$, we define

$$M_{s,t}(\theta) = \mathbb{E}[\exp(\theta(Z_t - Z_s))].$$

H.2 Continuous-Time Representation When $\kappa = 0$

Proposition H.1 (Pure-jump representation): *Let $\varepsilon > 0$. Assume $t \mapsto \alpha_t$ is continuously differentiable and that $\pi_i > 0$ for all $i \in \{1, \dots, N\}$, so that $P_1 \sim \mathrm{Dir}(\varepsilon\pi)$ has strictly positive coordinates almost surely. Define*

$$a(t) := \frac{\varepsilon}{1 - \alpha_{1-t}}, \qquad t \in [0, 1).$$

Then there exists a pure-jump additive process $(Z_t)_{t\in[0,1)}$ with $Z_0 = 0$, taking values in $(-\infty, 0]$, such that the process

$$\widetilde{P}_{1-t} = \big(1 - e^{Z_t}\big)P_0 + e^{Z_t}P_1, \qquad t \in [0, 1),$$

has the same terminal law and the same reverse transition kernels as (9). In particular, $(\widetilde{P}_t)_{t\in(0,1]}$ has the same finite-dimensional distributions as the reverse process implied by Proposition 3.3. Moreover, the increments of Z satisfy, for any $0 \leqslant s < t < 1$,

$$e^{Z_t - Z_s} \sim \mathrm{Beta}\big(a(s), a(t) - a(s)\big),$$

and Z is an additive process with local Lévy densities

$$\nu_t(\mathrm{d}x) = \frac{-\varepsilon\alpha'_{1-t}}{(1 - \alpha_{1-t})^2}\,\mathbf{1}_{(-\infty,0)}(x)\,\frac{\exp\Big(\frac{\varepsilon x}{1-\alpha_{1-t}}\Big)}{1 - \exp(x)}\,\mathrm{d}x, \qquad t \in [0, 1).$$

Proof. For simplicity, write

$$W_{s,t} := W^0_{s,t}.$$

For $0 \leqslant s < t < 1$, let $\mu_{s,t}$ denote the law of

$$\log W_{1-t,\,1-s}.$$

Since, by Proposition 3.3 for $\kappa = 0$, we get

$$W_{1-t,\,1-s} \sim \mathrm{Beta}\big(a(s),\, a(t) - a(s)\big),$$

we have, for every $\theta \geqslant 0$,

$$M_{s,t}(\theta) := \int e^{\theta x}\,\mu_{s,t}(dx) = \mathbb{E}\left[\big(W_{1-t,\,1-s}\big)^{\theta}\right] = \frac{\Gamma(a(s) + \theta)\Gamma(a(t))}{\Gamma(a(s))\Gamma(a(t) + \theta)}.$$

Hence, for $0 \leqslant r < s < t < 1$,

$$M_{r,t}(\theta) = M_{r,s}(\theta)M_{s,t}(\theta),$$

so that $\mu_{r,t} = \mu_{r,s} * \mu_{s,t}$. Moreover,

$$W_{1-t,\,1-s} \xrightarrow[t\downarrow s]{\mathbb{P}} 1,$$

thus $\mu_{s,t} \Rightarrow \delta_0$ as $t \downarrow s$. By the Kolmogorov extension theorem, there exists a process $(Z_t)_{t\in[0,1)}$ with $Z_0 = 0$, independent increments, and

$$Z_t - Z_s \sim \mu_{s,t}, \qquad 0 \leqslant s < t < 1.$$

Since $\mu_{s,t} \Rightarrow \delta_0$ as $t \downarrow s$, the process is continuous in probability, hence additive. In particular,

$$e^{Z_t - Z_s} \sim \mathrm{Beta}\big(a(s),\, a(t) - a(s)\big), \qquad 0 \leqslant s < t < 1.$$

Now let $P_1 \sim \mathrm{Dir}(\varepsilon\pi)$, independent of Z, and define

$$\widetilde{P}_{1-t} = (1 - e^{Z_t})P_0 + e^{Z_t}P_1, \qquad t \in [0, 1).$$

Then, for $0 \leqslant s < t < 1$,

$$\widetilde{P}_{1-t} - P_0 = e^{Z_t - Z_s}\big(\widetilde{P}_{1-s} - P_0\big),$$

that is,

$$\widetilde{P}_{1-t} = e^{Z_t - Z_s}\widetilde{P}_{1-s} + \big(1 - e^{Z_t - Z_s}\big)P_0.$$

Since

$$e^{Z_t - Z_s} \sim \mathrm{Beta}\big(a(s),\, a(t) - a(s)\big),$$

this is exactly the $\kappa = 0$ reverse kernel in Proposition 3.3 with

$$s' = 1 - t, \qquad t' = 1 - s.$$

Also,

$$\widetilde{P}_1 = P_1 \sim \mathrm{Dir}(\varepsilon\pi).$$

Therefore $(\widetilde{P}_t)_{t\in(0,1]}$ has the same terminal distribution and the same reverse transition kernels as the reverse process implied by Proposition 3.3. Hence it has the same finite-dimensional distributions.

Finally, starting from the expression for $M_{s,t}$, we obtain

$$\log M_{s,t}(\theta) = \int_s^t a'(u)\big[\psi(a(u)) - \psi(a(u) + \theta)\big]\, du.$$

Using the identity

$$\psi(a) - \psi(a + \theta) = \int_{-\infty}^{0} \big(e^{\theta x} - 1\big)\frac{e^{ax}}{1 - e^x}\, dx, \qquad a > 0,\ \theta \geqslant 0,$$

we get

$$\log \mathbb{E}\Big[e^{\theta(Z_t - Z_s)}\Big] = \int_s^t \int_{(-\infty,0)} \big(e^{\theta x} - 1\big)\, \nu_u(dx)\, du,$$

with

$$\nu_u(dx) = -\frac{\varepsilon\alpha'_{1-u}}{(1 - \alpha_{1-u})^2}\mathbf{1}_{(-\infty,0)}(x)\frac{\exp\Big(\frac{\varepsilon x}{1-\alpha_{1-u}}\Big)}{1 - \exp(x)}\, dx.$$

As $x \uparrow 0$,

$$\frac{e^{a(u)x}}{1 - e^x} \sim \frac{1}{|x|},$$

so

$$\int_{(-1,0)} |x|\, \nu_u(dx) < \infty.$$

As $x \to -\infty$, the density decays exponentially, while as $x \uparrow 0$ it behaves as $a'(u)/|x|$. Hence

$$\int_{\mathbb{R}} (1 \wedge |x|)\, \nu_u(\mathrm{d}x) < \infty.$$

Moreover, these quantities are integrable in u on every compact interval $[0, T] \subset [0, 1)$. Thus Z admits a pure-jump version of locally finite variation, with negative jumps only. □

H.3 No Stochastically Continuous Markov Embedding for Fixed $\kappa > 0$

Proposition H.2 (Inexistence of limit): *Fix $\varepsilon > 0$, $\kappa \in (0,1)$, and $t \in (0,1)$. Assume $N \geqslant 2$, that $\pi_i > 0$ for all $i \in \{1,\dots,N\}$, and let $P_0 = e_{x_0}$. For $P_t \in \mathrm{ri}(\Delta_N)$, denote by $p_{s|0,t}(\cdot \mid P_0, P_t)$ the reverse kernel from Proposition 3.3. Then, as $s \uparrow t$,*

$$p_{s|0,t}(\cdot \mid P_0, P_t) \xrightarrow{d} K_t^{\kappa}(\cdot \mid P_0, P_t),$$

where $K_t^{\kappa}(\cdot \mid P_0, P_t)$ is the law of

$$W_t^{\kappa} P_t^{\kappa} + \big(1 - W_t^{\kappa}\big) V_t^{\kappa},$$

with

$$W_t^{\kappa} \sim \mathrm{Beta}\left(\frac{(1-\kappa)\varepsilon}{1-\alpha_t}, \frac{\kappa\varepsilon}{1-\alpha_t}\right), \qquad V_t^{\kappa} \sim \mathrm{Dir}\big(\kappa\beta_t(P_0,\pi)\big),$$

and

$$(P_t^{\kappa})_i = \frac{B_i P_{t,i}}{\sum_{j=1}^N B_j P_{t,j}}, \qquad B_i \sim \mathrm{Beta}\big((1-\kappa)\,\beta_t(P_0,\pi)_i, \kappa\,\beta_t(P_0,\pi)_i\big), \quad i \in \{1,\dots,N\}.$$

Here the random variables $((B_i)_{i=1}^N, V_t^{\kappa}, W_t^{\kappa})$ are mutually independent.
In particular, $K_t^{\kappa}(\cdot \mid P_0, P_t) \neq \delta_{P_t}$. Consequently, for fixed $\kappa > 0$, the family of reverse kernels from Proposition 3.3 cannot arise from a stochastically continuous Markov process on Δ_N. In particular, the continuous-time jump representation of the previous subsection is specific to the case $\kappa = 0$.

Proof. Conditionally on P_0 and P_t, Proposition 3.3 gives

$$P_s = F_t\big(W_{s,t}^{\kappa},\, V_{s,t}^{\kappa},\, B\big)\,,$$

where

$$F_t(w,v,b) = w\left(\frac{b_i P_{t,i}}{\sum_{j=1}^N b_j P_{t,j}}\right)_{i=1}^N + (1-w)v, \qquad B = (B_1,\dots,B_N),$$

and

$$B_i \sim \mathrm{Beta}\big((1-\kappa)\beta_t(P_0,\pi)_i,\, \kappa\beta_t(P_0,\pi)_i\big), \qquad i \in \{1,\dots,N\}.$$

Since $\pi_i > 0$ for all i and $\kappa \in (0,1)$, the Beta parameters above are strictly positive, so $B_i \in (0,1)$ almost surely. As $P_t \in \mathrm{ri}(\Delta_N)$, where ri denotes the relative interior, we have $P_{t,i} > 0$ for all i, hence

$$\sum_{j=1}^N b_j P_{t,j} > 0 \qquad \text{for all } b \in (0,1)^N.$$

Therefore F_t is continuous on $(0,1) \times \Delta_N \times (0,1)^N$.

Now, as $s \uparrow t$,

$$W_{s,t}^{\kappa} \xrightarrow{d} W_t^{\kappa} \sim \mathrm{Beta}\left(\frac{(1-\kappa)\varepsilon}{1-\alpha_t}, \frac{\kappa\varepsilon}{1-\alpha_t}\right),$$

because the Beta parameters converge to

$$\frac{(1-\kappa)\varepsilon}{1-\alpha_t}, \qquad \frac{\varepsilon}{1-\alpha_t} - \frac{(1-\kappa)\varepsilon}{1-\alpha_t} = \frac{\kappa\varepsilon}{1-\alpha_t}.$$

Similarly,

$$\kappa\varepsilon\pi + \left(\frac{\varepsilon\alpha_s}{1-\alpha_s} - \frac{(1-\kappa)\varepsilon\alpha_t}{1-\alpha_t}\right) P_0 \longrightarrow \kappa\varepsilon\pi + \kappa\frac{\varepsilon\alpha_t}{1-\alpha_t} P_0 = \kappa\beta_t(P_0,\pi),$$

so

$$V_{s,t}^{\kappa} \xrightarrow{d} V_t^{\kappa} \sim \mathrm{Dir}\big(\kappa\beta_t(P_0,\pi)\big).$$

Since the law of B does not depend on s and B is independent of $(W^\kappa_{s,t}, V^\kappa_{s,t})$, we obtain the joint convergence

$$\left(W^\kappa_{s,t}, V^\kappa_{s,t}, B\right) \xrightarrow{d} (W^\kappa_t, V^\kappa_t, B)\,.$$

By the continuous mapping theorem,

$$P_s \xrightarrow{d} F_t(W^\kappa_t, V^\kappa_t, B)\,,$$

which is exactly the kernel

$$K^\kappa_t(\cdot \mid P_0, P_t).$$

It remains to show that this limit kernel is not the identity. Since $P_t \in \mathrm{ri}(\Delta_N)$, the map

$$b \longmapsto \left(\frac{b_i P_{t,i}}{\sum_{j=1}^N b_j P_{t,j}}\right)_{i=1}^N$$

is non-constant on $(0,1)^N$, and each B_i has a non-degenerate Beta law. Hence P^κ_t is non-degenerate conditional on P_t. Since

$$W^\kappa_t \in (0,1) \qquad \text{almost surely},$$

the conditional law of

$$W^\kappa_t P^\kappa_t + \left(1 - W^\kappa_t\right) V^\kappa_t$$

given (W^κ_t, V^κ_t) is a non-degenerate affine image of P^κ_t. Therefore

$$K^\kappa_t(\cdot \mid P_0, P_t) \neq \delta_{P_t}.$$

Finally, the small-time transition kernels of any stochastically continuous Markov process must converge weakly to the Dirac mass at the current state. Since this fails here for fixed $\kappa > 0$, the family of reverse kernels from Proposition 3.3 cannot arise from a stochastically continuous Markov process on Δ_N. In particular, the continuous-time jump representation of the previous subsection is specific to the case $\kappa = 0$. □

I A Bridge Perspective

In this section, we show that (4) can arise as the bridge representation of a time-changed Brownian process. The simplex-valued component considered below is not Markov in general, but becomes Markov after augmenting the state with its stochastic clock. This construction is useful to draw connections with Optimal Transport (OT) (Villani, 2003) and provides a corresponding Schrödinger Bridge perspective on the augmented state space (Léonard, 2014; Shi et al., 2023).

We start by recalling (4), i.e.

$$P_t = W_t P_0 + (1 - W_t) V_\pi,$$

where $\varepsilon > 0$, $t \in [0,1]$ and independent $W_t \sim \mathrm{Beta}\,(\varepsilon h_t, \varepsilon)$, $V_\pi \sim \mathrm{Dir}(\varepsilon\pi)$.

In order to clarify the rest of our exposition, we briefly revisit the Euclidean setting, adapting the presentation of Albergo et al. (2025).

Let $x_0, x_1 \in \mathbb{R}^d$, we consider X_t for $t \in [0,1]$ given by

$$X_t = (1-t)x_0 + t x_1 + \sqrt{\varepsilon t(1-t)}Z, \tag{31}$$

where $Z \sim \mathcal{N}(0, \mathrm{Id})$ is a d-dimensional Gaussian random variable. It can be easily shown that the following process $(X_t)_{t\in[0,1]}$ admits the same marginal distributions as (31): first consider a (scaled) d-dimensional Brownian motion $(\sqrt{\varepsilon}B_t)_{t\in[0,1]}$. Then, we can define X_t as B_t conditioned on $B_0 = x_0$ and $B_1 = x_1$. Hence, $(X_t)_{t\in[0,1]}$ is a scaled Brownian bridge starting at x_0 and ending at x_1 at time $t = 1$.

From this perspective, we can draw a link with Optimal Transport (OT). Let $x_0 \sim \pi^0$ and $x^1 \sim \pi^1$ independently, with π^0 and π^1 two probability measures defined on $\mathbb{R}^d$. Let us denote $\mathbb{Q}$ the path measure associated with $(\sqrt{\varepsilon}B_t)_{[t\in[0,1]}$. Next, define the sequence of path measures $(\mathbb{P}^n)_{n\in\mathbb{N}}$ such that $\mathbb{P}^0 = (\pi^0 \otimes \pi^1)\mathbb{Q}_{|0,1}$. This notation means that given $X_0 \sim \pi^0$ and $X_1 \sim \pi^1$ independently,

we sample $(X_t)_{t\in[0,1]}$ according to the (scaled) Brownian bridge, i.e., we can access X_t using (31). For any $n \in \mathbb{N}$, we define $\mathbb{P}^{2n+1}$ as the Markov projection of $\mathbb{P}^n$ and $\mathbb{P}^{2n+2} = \mathbb{P}^{2n+1}_{0,1}\mathbb{Q}_{|0,1}$. We refer to Shi et al. (2023) for details about this procedure called Iterative Markovian Fitting (IMF), which can be seen as the dual of the well known Sinkhorn algorithm (Cuturi, 2013).

Under some assumptions, we have that $\lim_{n\to+\infty} \mathbb{P}^n = \mathbb{P}^\star$ with $\mathbb{P}^\star$ as the Schrödinger Bridge, i.e.

$$\mathbb{P}^\star = \mathrm{argmin}_{\mathbb{P}} \mathrm{KL}(\mathbb{P}|\mathbb{Q}),$$

where the argmin is taken over Markov path measures such that $\mathbb{P}_0 = \pi^0$ and $\mathbb{P}_1 = \pi^1$. It can be shown that $\mathbb{P}^\star_{0,1}$ is in fact given by

$$\mathbb{P}^\star_{0,1} = \mathrm{argmin}_{\mathbb{P}_{0,1}} \left[-\int \log(\mathbb{Q}_{1|0}(x_1|x_0)) \mathrm{d}\mathbb{P}_{0,1}(x_0, x_1) - \mathrm{H}(\mathbb{P}_{0,1}) \right],$$

where H is the differential entropy defined by $\mathrm{H}(\mathbb{P}_{0,1}) = -\int \log(\mathbb{P}_{0,1}) \mathrm{d}\mathbb{P}_{0,1}$. In our case, since $\mathbb{Q}$ is a scaled Brownian motion, we get that

$$\mathbb{P}^\star_{0,1} = \mathrm{argmin}_{\mathbb{P}_{0,1}} \left[\int \|x_0 - x_1\|^2 \mathrm{d}\mathbb{P}_{0,1}(x_0, x_1) - 2\varepsilon\, \mathrm{H}(\mathbb{P}_{0,1}) \right]. \tag{32}$$

In particular, we have that Optimal Transport cost is fully determined by the forward process $\mathbb{Q}$, here the scaled Brownian motion. Hence, we have shown that in the Euclidean setting the following holds:

- *Stochastic interpolant:* there exists an interpolant between x_0 and x_1 which converges to the classical flow matching bridge as an hyperparameter goes to zero.
- *Brownian bridge:* this interpolant can be realised as the bridge of a Markov measure, in the Euclidean case a scaled Brownian motion.
- *Optimal Transport:* iterating on the associated bridge matching procedure, we show that the obtained coupling between the target measures π^0 and π^1 converges to the solution of the (regularized) Optimal Transport problem with ℓ_2^2 cost. The regularization is given by the scale of the Brownian motion.

In what follows, we aim at providing a similar description of interpolants in the setting of simplicial models. In particular, we want to answer the following questions:

- Can we find a stochastic interpolant with a parameter which plays the same role as ε in the Euclidean setting?
- Can we realise this interpolant as the bridge of a stochastic process, possibly after a Markovian augmentation?
- What is the associated Optimal Transport problem?

We begin with Section I.1 to give some high level intuition about the stochastic process we consider. Our main result regarding the interpolant is given in Section I.2.

I.1 High Level Intuition

In this section, we provide some intuition about the process we introduce in Section I.2. We do not claim full rigour but rather emphasize the main ideas that lead to the definition of the stochastic time-changed process.

First of all, let us analyze the bridge requirement. We require

$$\lim_{c\to+\infty} P_t = (1 - W_t)P_0 + W_t P_1,$$

where $W_t \sim \mathrm{Beta}\,(\varepsilon h_t, \varepsilon)$, h_t is a strictly increasing schedule with $h_0 = 0$ and $\lim_{t\to 1} h_t = +\infty$, and c is some hyperparameter similar to the scale in the case of the scaled Brownian motion. In the Brownian motion case, we have

$$\lim_{\varepsilon\to 0} X_t = (1 - t)x_0 + tx_1.$$

In contrast to the Brownian motion case, we want a process such that, when we take the limit with respect to the hyperparameter $c \to +\infty$, the interpolation is *still* stochastic. Intuitively if we identify t and W_t this means that the time itself must be stochastic. All of the above considerations guide us towards considering a process defined in a *random medium*.

To do so we define a stochastic time-change using a *Gamma subordinator* and an auxiliary *random medium* variable. More precisely, let $S \sim \Gamma(\varepsilon, 1)$ be an auxiliary variable drawn at time $t = 0$, and let $(G_t)_{t\in[0,1]}$ be an independent Gamma subordinator with increasing shape function $\gamma(t) = \varepsilon h_t$, such that $G_t \sim \Gamma(\varepsilon h_t, 1)$. We define the stochastic time-change $\tau_t \in [0, 1]$ by

$$\tau_t = \frac{G_t}{S + G_t}.$$

By Lukacs' proportion-sum theorem (Lukacs, 1955), $\tau_t \sim \mathrm{Beta}(\varepsilon h_t, \varepsilon)$. Next, we define the process $(P_t)_{t\in[0,1]}$ such that for any $t \in [0, 1]$

$$P_t = P_0 + \frac{1}{\sqrt{c}} B_{\tau_t},$$

where $c > 0$ and $(B_u)_{u\in[0,1]}$ is a d-dimensional Brownian motion independent of (S, G_t). The process $(P_t)_{t\in[0,1]}$ is in general not Markov: the current value P_t does not determine the current stochastic clock τ_t, while the law of future increments depends on τ_t. However, the augmented process $(\tau_t, P_t)_{t\in[0,1]}$ is Markov. Thus, whenever we refer below to a Markov representation, we are talking about the augmented representation. Note that the process $(P_t)_{t\in[0,1]}$ does not necessarily live on the simplex. Nevertheless we will show that it has the correct bridge property. Since $\tau_t \in [0, 1]$ is a bounded stochastic time-change driven by a Gamma subordinator in a random medium, this process can be seen as a modification of the Variance-Gamma process originally introduced in the finance community; see (Seneta, 2007) for a retrospective.

I.2 Bridge for the Beta-Clock Process

We begin with the following lemma which gives the relevant densities of $(P_t)_{t\in[0,1]}$.

Lemma I.1 (Conditional density representation): *Let $(B_u)_{u\in[0,1]}$ be a standard d-dimensional Brownian motion, let $S \sim \Gamma(\varepsilon, 1)$ be an auxiliary variable, and let $(G_t)_{t\in[0,1]}$ be an independent Gamma subordinator with increasing shape function $\gamma(t) = \varepsilon h_t$, such that $G_t \sim \Gamma(\varepsilon h_t, 1)$. Define $\tau_t = G_t/(S + G_t)$ and let $(P_t)_{t\in[0,1]}$ be defined by*

$$P_t = P_0 + \frac{1}{\sqrt{c}} B_{\tau_t}, \qquad t \in [0, 1].$$

Then, for any $t \in (0, 1)$:

1. *The forward marginal density of P_t given P_0 is*

$$p(P_t \mid P_0) = \int_0^1 \mathrm{Beta}(u;\, \varepsilon h_t,\, \varepsilon)\, \varphi_{P_0,(u/c)\mathrm{Id}}(P_t)\, \mathrm{d}u, \tag{33}$$

   where $\varphi_{\mu,\Sigma}$ is the Gaussian density with mean μ and covariance matrix Σ.

2. *For any $u \in (0, 1)$, conditioned on $\tau_t = u$, the conditional density from P_t to $P_1 = P(1)$ is*

$$p(P_1 \mid P_t, \tau_t = u) = \varphi_{P_t,((1-u)/c)\mathrm{Id}}(P_1). \tag{34}$$

3. *The marginal density of $P_1 = P(1)$ given P_0 is*

$$p(P_1 \mid P_0) = \varphi_{P_0,(1/c)\mathrm{Id}}(P_1). \tag{35}$$

Proof. 1. Conditional on $\tau_t = u$, the increment is $P_t - P_0 = \frac{1}{\sqrt{c}} B_u \sim \mathcal{N}(0, \frac{u}{c}\mathrm{Id})$, so $p(P_t \mid P_0, \tau_t = u) = \varphi_{P_0,(u/c)\mathrm{Id}}(P_t)$. Marginalising over $\tau_t \sim \mathrm{Beta}(\varepsilon h_t, \varepsilon)$ yields (33).

2. Since $\tau_1 = 1$ deterministically, we have $P_1 - P_t = \frac{1}{\sqrt{c}}(B_1 - B_u)$. By the independent increment property of Brownian motion, $B_1 - B_u \sim \mathcal{N}(0, (1-u)\mathrm{Id})$ is independent of $(B_v)_{v \leqslant u}$ and of (S, G_t). Hence $P_1 - P_t \sim \mathcal{N}(0, \frac{1-u}{c}\mathrm{Id})$, which gives (34).

3. At $t = 1$, $\tau_1 = 1$ deterministically. Therefore $P_1 - P_0 = \frac{1}{\sqrt{c}} B_1 \sim \mathcal{N}(0, \frac{1}{c}\mathrm{Id})$, yielding (35). Note that no random clock appears at the terminal time.

□

We now provide a useful bridge representation.

Proposition I.1 (Bridge representation): *Let $c > 0$. Conditionally on P_0, P_1, the process admits the representation*

$$P_t \overset{d}{=} (1-\tau_t)P_0 + \tau_t P_1 + \frac{1}{\sqrt{c}}\,\mathbb{B}_{\tau_t}, \qquad t \in [0,1], \tag{36}$$

where $(\mathbb{B}_u)_{u\in[0,1]}$ is a standard d-dimensional Brownian bridge, independent of the stochastic clock $(\tau_t)_{t\in[0,1]}$.
In particular, for any $t \in (0,1)$ and any $P_0, P_1 \in \mathbb{R}^d$,

$$p(P_t \mid P_0, P_1) = \int_0^1 \mathrm{Beta}(u; \varepsilon h_t, \varepsilon)\, \varphi_{\mu_u, \frac{u(1-u)}{c}\mathrm{Id}}(P_t)\, \mathrm{d}u, \tag{37}$$

where $\mu_u = (1-u)P_0 + uP_1$.

Proof. Recall that

$$P_t = P_0 + \frac{1}{\sqrt{c}} B_{\tau_t}, \qquad P_1 = P_0 + \frac{1}{\sqrt{c}} B_1,$$

where $(B_u)_{u\in[0,1]}$ is a standard d-dimensional Brownian motion independent of the stochastic clock $(\tau_t)_{t\in[0,1]}$.

Introduce the standard Brownian bridge

$$\mathbb{B}_u := B_u - uB_1, \qquad u \in [0,1].$$

Then, we have

$$B_u = uB_1 + \mathbb{B}_u.$$

Moreover, $(\mathbb{B}_u)_{u\in[0,1]}$ is independent of B_1. Indeed, $(B_1, \mathbb{B})$ is jointly Gaussian and, for every $u \in [0,1]$,

$$\mathrm{Cov}(\mathbb{B}_u, B_1) = \mathrm{Cov}(B_u - uB_1, B_1) = u\mathrm{Id} - u\mathrm{Id} = 0.$$

Since the (stochastic) clock is independent of the Brownian motion, $(\tau_t)_{t\in[0,1]}$ is also independent of $(B_1, \mathbb{B})$.

Conditioning on (P_0, P_1) is therefore equivalent to fixing

$$B_1 = \sqrt{c}\,(P_1 - P_0),$$

while leaving the laws of both $\mathbb{B}$ and τ unchanged and independent. Hence (36) follows from

$$\begin{aligned} P_t &= P_0 + \frac{1}{\sqrt{c}}\,(\tau_t B_1 + \mathbb{B}_{\tau_t}) \\ &= (1-\tau_t)P_0 + \tau_t P_1 + \frac{1}{\sqrt{c}}\mathbb{B}_{\tau_t}. \end{aligned}$$

Finally, conditionally on $\tau_t = u$, $\mathbb{B}_u \sim \mathcal{N}\big(0, u(1-u)\mathrm{Id}\big)$ and $\tau_t \sim \mathrm{Beta}(\varepsilon h_t, \varepsilon)$. Therefore

$$P_t \mid (P_0, P_1, \tau_t = u) \sim \mathcal{N}\left((1-u)P_0 + uP_1,\ \frac{u(1-u)}{c}\mathrm{Id}\right).$$

Integrating with respect to the law of τ_t gives (37). □

We are now ready to state our main bridge concentration result.

Theorem I.1 (Bridge concentration): *For any $t \in (0,1)$, we have as $c \to \infty$*

$$P_t \mid P_0, P_1 \xrightarrow{d} (1 - W_t)\, P_0 + W_t\, P_1, \qquad W_t \sim \mathrm{Beta}(\varepsilon h_t,\, \varepsilon),$$

where W_t is independent of the endpoints (P_0, P_1), and satisfies $\mathbb{E}[W_t] = \frac{h_t}{1+h_t}$.

Proof. Let $\Phi \in \mathrm{C}^\infty(\mathbb{R}^d, \mathbb{R})$ be bounded and uniformly continuous. Using Section I.2, we have

$$\mathbb{E}[\Phi(P_t) \mid P_0, P_1] = \int_0^1 \mathrm{Beta}(u; \varepsilon h_t, \varepsilon) \int_{\mathbb{R}^d} \Phi(P)\, \varphi_{\mu_u, \frac{u(1-u)}{c}\mathrm{Id}}(P) \mathrm{d}P\, \mathrm{d}u.$$

Denote $\sigma_u^2 = \frac{u(1-u)}{c}$. Since $u(1-u) \leqslant \frac{1}{4}$ for all $u \in [0,1]$, we have

$$\sigma_u^2 \leqslant \frac{1}{4c} \xrightarrow{c\to\infty} 0.$$

Next, denote by $\omega_\Phi(\delta) = \sup\{|\Phi(P) - \Phi(Q)|\,;\, \|P - Q\|_2 \leqslant \delta\}$ the modulus of continuity of Φ. For any $u \in (0,1)$, we have

$$\left| \int_{\mathbb{R}^d} \Phi(P)\, \varphi_{\mu_u, \sigma_u^2 \mathrm{Id}}(P) \mathrm{d}P - \Phi(\mu_u) \right| \leqslant \mathbb{E}_{Z\sim\mathcal{N}(0,\mathrm{Id})}\left[\omega_\Phi(\sigma_u \|Z\|_2)\right].$$

For any fixed truncation $R > 0$, we bound

$$\mathbb{E}_Z\left[\omega_\Phi(\sigma_u \|Z\|_2)\right] \leqslant \omega_\Phi\left(\frac{R}{2\sqrt{c}}\right) + 2\|\Phi\|_\infty \frac{d}{R^2}.$$

This bound is uniform in u. Therefore, integrating with respect to $\mathrm{Beta}(u; \varepsilon h_t, \varepsilon)$:

$$\left| \mathbb{E}[\Phi(P_t) \mid P_0, P_1] - \int_0^1 \mathrm{Beta}(u; \varepsilon h_t, \varepsilon)\, \Phi(\mu_u)\, \mathrm{d}u \right| \leqslant \omega_\Phi\left(\frac{R}{2\sqrt{c}}\right) + 2\|\Phi\|_\infty \frac{d}{R^2}.$$

Taking $c \to \infty$ and then $R \to \infty$ yields

$$\begin{aligned} \lim_{c\to\infty} \mathbb{E}[\Phi(P_t) \mid P_0, P_1] &= \int_0^1 \mathrm{Beta}(u; \varepsilon h_t, \varepsilon)\, \Phi\left((1-u)P_0 + uP_1\right) \mathrm{d}u \\ &= \mathbb{E}\left[\Phi\left((1 - W_t^\varepsilon)P_0 + W_t^\varepsilon P_1\right)\right], \end{aligned}$$

where $W_t^\varepsilon \sim \mathrm{Beta}(\varepsilon h_t, \varepsilon)$. By the Portmanteau theorem, this establishes weak convergence, which concludes the proof. □

I.3 Optimal Transport Perspective

Although $(P_t)_{t\in[0,1]}$ is not Markov in general, the augmented process $(\tau_t, P_t)_{t\in[0,1]}$ is Markov. Since $\tau_0 = 0$ and $\tau_1 = 1$ are deterministic, its endpoint kernel on the P-coordinates is simply $p(P_1 \mid P_0)$. Hence, the associated Schrödinger/entropic Optimal Transport problem is governed by the cost $-\log p(P_1 \mid P_0)$. For our process, we have the following result.

Proposition I.2 (Terminal density): *Let $c > 0$. For any $P_0, P_1 \in \mathbb{R}^N$, we have*

$$p(P_1 \mid P_0) = \varphi_{P_0, (1/c)\mathrm{Id}}(P_1) = \frac{c^{N/2}}{(2\pi)^{N/2}} \exp\left(-\frac{c}{2}\|P_1 - P_0\|_2^2\right).$$

In particular,

$$-\log p(P_1 \mid P_0) = \frac{c}{2}\|P_1 - P_0\|_2^2 + \frac{N}{2}\log\frac{2\pi}{c}.$$

Proof. This follows directly from Lemma I.1, part 3: since $\tau_1 = 1$ deterministically, $P_1 - P_0 = \frac{1}{\sqrt{c}} B_1 \sim \mathcal{N}(0, \frac{1}{c}\mathrm{Id})$. □

Therefore, we have that

$$\mathbb{P}^{\star}_{0,1} = \operatorname{argmin}_{\mathbb{P}_{0,1}} \left[-\int \log(p_{1|0}(P_1|P_0)) \mathrm{d}\mathbb{P}_{0,1}(P_0, P_1) - \mathrm{H}(\mathbb{P}_{0,1}) \right]$$
$$= \operatorname{argmin}_{\mathbb{P}_{0,1}} \left[\frac{c}{2} \int \|P_0 - P_1\|_2^2 \mathrm{d}\mathbb{P}_{0,1}(P_0, P_1) - \mathrm{H}(\mathbb{P}_{0,1}) \right].$$

The ground cost that arises with our representation is the squared ℓ_2 distance $\|P_0 - P_1\|_2^2$, also known as the Brier divergence when $P_0, P_1 \in \Delta_N$. This is a direct analogue of (32) in the Euclidean setting. The scale parameter c plays the role of the inverse temperature.

Remark I.1. In addition to the terminal density, the forward marginal density $p(P_t \mid P_0)$ for $t \in (0, 1)$ is given by the Beta mixture of Gaussians from Lemma I.1:

$$p(P_t \mid P_0) = \int_0^1 \mathrm{Beta}(u;\, \varepsilon h_t,\, \varepsilon)\, \varphi_{P_0, (u/c)\mathrm{Id}}(P_t)\, \mathrm{d}u.$$

This mixture does not simplify to a closed-form expression in terms of classical special functions, in contrast with the standard Variance-Gamma process whose transition density involves the modified Bessel function K_ν.

J Distillation

J.1 Simplex Distribution Matching Distillation

Throughout this section, we restrict attention to time pairs for which the Dirichlet parameters below are strictly positive.

Distribution Matching Distillation. We consider a multistep and simplicial version of Distribution Matching Distillation (DMD) (Luo et al., 2023; Yin et al., 2024). We introduce a generator $G : \mathcal{P} \times [0, 1] \times \Delta_N \times \mathcal{U} \to \Delta_N$, where $\mathcal{U}$ is a noise space and $\mathcal{P}$ is a real vector space which represents the parameter space of the parametric function G_η. The main goal of the distillation method we describe below is to find G_η such that, for some random variable $U \sim \mathbb{P}_U$ taking values in $\mathcal{U}$, we have $G_\eta(t, P_t, U) \sim \hat{p}_{0|t}(\cdot|P_t)$, where $p_{0|t}$ is given by Bayes's rule, (3) and

$$\int_{\Delta_N} p_t(P_t) p_{0|t}(P_0|P_t) \mathrm{d}P_t = \int_{\Delta_N} p_t(P_t) \hat{p}_{0|t}(P_0|P_t) \mathrm{d}P_t.$$

In particular, $G_\eta(t, P_t, U)$ should take values on the vertices of Δ_N, since its prior distribution $p_{\text{data}}(P) = \sum_{i=1}^N p_{\text{data},i} \delta_{e_i}(P)$ (see (2)) only has mass on the vertices. In what follows, we make no such assumption on the form of G_η and only ensure that G_η takes values in Δ_N.

We denote $\mathrm{d}G_\eta(t, P_t, U) : \mathcal{P} \to \mathbb{R}^N$ the derivative of G_η with respect to η, evaluated at (η, t, P_t, U). Similarly, we denote $\mathrm{D}G_\eta(t, P_t, U)$ such that for any $h \in \mathcal{P}$

$$\mathrm{d}G_\eta(t, P_t, U)(h) = \mathrm{D}G_\eta(t, P_t, U)h.$$

We recall that for every $t \in [0, 1]$, we have that P_t is given by (3). For notational simplicity, we define $\mathbb{P}_t$ the distribution of P_t with $P_t \sim p_{t|0}(\cdot \mid P_0)$ and $P_0 = e_{x_0}$ with $x_0 \sim p_{\text{data}}$. More precisely for any test function $f \in \mathrm{C}^\infty(\Delta_N, \mathbb{R})$ we have that

$$\int_{\Delta_N} f(P)\, \mathrm{d}\mathbb{P}_t(P) = \int_{\Delta_N} \int_{\Delta_N} f(P_t)\, \mathrm{d}p_{t|0}(P_t \mid P_0)\, \mathrm{d}p_{\text{data}}(P_0).$$

Similarly, we denote $\mathbb{P}_t^{\eta,s}$ the distribution of P_t with $P_t \sim p_{t|0}(\cdot \mid P_0^s)$ where $P_0^s = G_\eta(s, P_s, U)$ where $P_s \sim p_{s|0}(\cdot \mid P_0)$ and $P_0 \sim p_{\text{data}}(\cdot)$. More precisely for any test function $f \in \mathrm{C}^\infty(\Delta_N, \mathbb{R})$ we have that

$$\int_{\Delta_N} f(P)\, \mathrm{d}\mathbb{P}_t^{\eta,s}(P)$$
$$= \int_{\Delta_N} \int_{\Delta_N} \int_{\mathcal{U}} \int_{\Delta_N} \int_{\Delta_N} f(P_t)\, \mathrm{d}p_{t|0}(P_t \mid P_0^s)\, \mathrm{d}\delta_{G_\eta(s,P_s,U)}(P_0^s)\, \mathrm{d}\mathbb{P}_U(U)\, \mathrm{d}p_{s|0}(P_s \mid P_0)\, \mathrm{d}p_{\text{data}}(P_0).$$

We denote by $\mathbb{P}^{\eta,s}_{0|t}$ the conditional distribution of P^s_0 given P_t under this construction, and by $\mathbb{P}_{0|t}$ the conditional distribution of P_0 given P_t under the data-forward joint distribution.

We consider the following distribution matching loss

$$\mathcal{L}(\eta) = \int_0^1 \int_0^1 \mathrm{KL}(\mathbb{P}^{\eta,s}_t | \mathbb{P}_t) \mathrm{d}\mathbb{Q}(s,t). \tag{38}$$

Note that in the original setting of DMD (Luo et al., 2023; Yin et al., 2024), we have that $\mathrm{d}\mathbb{Q}(s,t) = \delta_1(s)\mathrm{d}\mathbb{Q}(t)$. In the multistep regime however, we consider a general distribution $\mathbb{Q}$. In what follows, we are going to first derive the gradient of the loss function and then give an equivalent expression for (38) which can be readily implemented.

First, we recall the forward process

$$p_{t|0}(P_t|P_0) = \mathrm{Dir}(P_t; \beta_t(P_0,\pi)), \qquad \beta_t(P_0,\pi) = a_t P_0 + b_t \pi = c_t(\alpha_t P_0 + (1-\alpha_t)\pi),$$

where $a_t = c_t\alpha_t$ and $b_t = c_t(1-\alpha_t)$, as in Section 3. In particular, using the fact that a Dirichlet random variable can be generated using Gamma random variables (see Section A), then for any $t \in [0,1]$ there exists F_t such that $F_t(P_0, V)$ has distribution $p_{t|0}(\cdot|P_0)$, with $V = \{V_i\}_{i=1}^N$ which represents N independent uniform random variables used to generate the N Gamma random variates. In addition, F_t is differentiable with respect to P_0. We denote $\mathrm{d}F_t(P_0,V) : \mathbb{R}^N \to \mathbb{R}^N$ the differential of F_t with respect to P_0 evaluated at (P_0, V). Finally, we introduce $\mathrm{D}F_t(P_0,V) \in \mathbb{R}^{N\times N}$ such that for any $h \in \mathbb{R}^N$ we have

$$\mathrm{d}F_t(P_0,V)(h) = \mathrm{D}F_t(P_0,V)h.$$

Surrogate Loss. While (38) is a valid loss, it is not tractable. However, leveraging stop-gradient techniques and an equivalent of Tweedie's identity in the case of Dirichlet distributions, we will be able to derive a surrogate tractable loss with identical gradients in Proposition J.1. We start with the following lemma.

Lemma J.1 (Chain rule): *Let $s,t \in [0,1]$, $P_s \in \Delta_N$, $U \sim \mathbb{P}_U$ and $V = \{V_i\}_{i=1}^N$, N independent uniform random variables. Let $f : \Delta_N \to \mathbb{R}$ be a differentiable function. Define $g : \mathcal{P} \to \mathbb{R}$ such that $g(\eta) = f(F_t(G_\eta(s,P_s,U),V))$. We have*

$$\nabla_\eta g(\eta) = \mathrm{D}G_\eta(s,P_s,U)^\top \mathrm{D}F_t(G_\eta(s,P_s,U),V)^\top \nabla f(F_t(G_\eta(s,P_s,U),V)).$$

Next, we consider the following lemma which shows that Dirichlet distribution also enjoys some form of Tweedie's identity.

Lemma J.2 (Tweedie meets Dirichlet): *Let p_X be a distribution over Δ_N and $p_{Y|X}$ a Dirichlet distribution with parameter $\alpha(X) \in (0,+\infty)^N$. Then, we have*

$$\nabla \log p_Y(y) = \left\{ \frac{\mathbb{E}[\alpha_i(X)\,|Y] - 1}{y_i} \right\}_{i=1}^N.$$

Proof. We have that p_X is a distribution over Δ_N and $p_{Y|X}$ is a Dirichlet distribution with parameter $\alpha(X) \in (0,+\infty)^N$. We have that for any $y \in \Delta_N$

$$p_{Y|X}(y|X) = \frac{\Gamma(\sum_{i=1}^N \alpha_i(X))}{\prod_{i=1}^N \Gamma(\alpha_i(X))} \prod_{i=1}^N y_i^{\alpha_i(X)-1}.$$

Then, we have that

$$\nabla_y \log p_{Y|X}(y|X) = \{(\alpha_i(X)-1)/y_i\}_{i=1}^N.$$

In addition, we have that

$$\nabla \log p_Y(y) = \int \nabla_y \log p_{Y|X}(y|x)\; p_{X|Y}(x|y)\mathrm{d}x.$$

Finally, we get that

$$\nabla \log p_Y(y) = \left\{ \frac{\mathbb{E}[\alpha_i(X)\,|Y] - 1}{y_i} \right\}_{i=1}^{N}.$$

□

Combining the standard pathwise gradient identity for the KL divergence with Lemma J.1 and Lemma J.2, we obtain the following proposition.

Proposition J.1. *Let $\mathcal{L}(\eta)$ be given by (38). Then, we have that*

$$\begin{aligned}\nabla_\eta \mathcal{L}(\eta) = \int a_t \, \mathrm{D}G_\eta(s, P_s, U)^\top \mathrm{D}F_t(G_\eta(s, P_s, U), V)^\top \\ \times \frac{\mathbb{E}_{\mathbb{P}^{\eta,s}_{0|t}}[P_0^s | F_t(G_\eta(s, P_s, U), V)] - \mathbb{E}_{\mathbb{P}_{0|t}}[P_0 | F_t(G_\eta(s, P_s, U), V)]}{F_t(G_\eta(s, P_s, U), V)} \\ \times \mathrm{d}\mathbb{Q}(s,t)\mathrm{d}p_{\text{data}}(P_0)\mathrm{d}p_{s|0}(P_s|P_0)\mathrm{d}\mathbb{P}_U(U)\mathrm{d}V,\end{aligned}$$

where the division is coordinate-wise.

We can therefore find an equivalent loss to (38) which is tractable and yields the same gradients.

Proposition J.1 (Surrogate loss): *Let $\hat{\mathcal{L}}(\eta)$ be given by*

$$\begin{aligned}\hat{\mathcal{L}}(\eta) = \int a_t \, \log(F_t(G_\eta(s, P_s, U), V))^\top \\ \mathrm{sg}\left(\mathbb{E}_{\mathbb{P}^{\eta,s}_{0|t}}[P_0^s | F_t(G_\eta(s, P_s, U), V)] - \mathbb{E}_{\mathbb{P}_{0|t}}[P_0 | F_t(G_\eta(s, P_s, U), V)]\right) \\ \times \mathrm{d}\mathbb{Q}(s,t)\mathrm{d}p_{\text{data}}(P_0)\mathrm{d}p_{s|0}(P_s|P_0)\mathrm{d}\mathbb{P}_U(U)\mathrm{d}V.\end{aligned}$$

Then $\hat{\mathcal{L}}$ and $\mathcal{L}$ have the same gradients.

Algorithm and Links with Literature. Leveraging the loss given in Proposition J.1, we consider an algorithm to learn a distilled model. The full algorithm is given in Algorithm 4. It is apparent that Algorithm 4 follows the same pattern as many existing distillation algorithms such as DMD (Yin et al., 2024) but also Multistep Moment matching Distillation (MMD) (Salimans et al., 2024). In fact, our algorithm can be interpreted as a simplicial extension of the MMD algorithm. In both cases, we train a *generator* network G_η as well as an *auxiliary* denoiser approximating $\mathbb{E}_{\mathbb{P}^{\eta,s}_{0|t}}[P_0^s | P_t]$. Another line of related work is Universal Distillation Matching (UDM) and its discrete counterpart Inverse Distillation Language Model (IDLM) (Kornilov et al., 2026; Li et al., 2026). While closely related to our method, we highlight the key differences between these methods and our contribution in Section J.2.

J.2 Link with Inverse Distilled Language Models

In this section, we draw connections between the Simplex DMD framework and the Inverse Distilled Language Model (IDLM) approach (Kornilov et al., 2026; Li et al., 2026). We show that both methods share a common structure but differ in how the gradient of the generative loss is handled, with the Simplex DMD framework resolving a fundamental difficulty encountered in IDLM.

IDLM Generative Loss. Adapting the formulation of Kornilov et al. (2026); Li et al. (2026) to the notation of the present paper, the IDLM generative loss can be written as

$$\begin{aligned}\mathcal{L}^{\mathrm{G}}(\eta) = \int G_\eta(s, P_s, U)^\top \left(\log D_\phi(F_t(G_\eta(s, P_s, U), V),\ s, t) - \log D_{\text{teach}}(F_t(G_\eta(s, P_s, U), V),\ t)\right) \\ \times \mathrm{d}\mathbb{Q}(s,t)\mathrm{d}p_{\text{data}}(P_0)\mathrm{d}p_{s|0}(P_s|P_0)\mathrm{d}\mathbb{P}_U(U)\mathrm{d}V,\end{aligned}$$

where D_ϕ plays the role of the auxiliary model and D_{teach} the teacher model from Li et al. (2026). The generator G_η appears *twice* in this expression: as the denoised sample $P_0^s = G_\eta(s, P_s, U)$ and

Algorithm 4 Simplex DMD Training

Require: Teacher denoiser $D_{\text{teach}}(P_t, t) \approx \mathbb{E}_{\mathbb{P}_{0|t}}[P_0 \mid P_t]$
Require: Generator $G_\eta(s, P_s, U)$, auxiliary denoiser $D_\phi(P_t, s, t) \approx \mathbb{E}_{\mathbb{P}^{\eta,s}_{0|t}}[P_0^s \mid P_t]$
Require: Data distribution p_{data}, noise distribution $\mathbb{P}_U$, time distribution $\mathbb{Q}$
Require: Forward process parameters $(\alpha_t)_{t\in[0,1]}$, $(c_t)_{t\in[0,1]}$, and π, with $a_t = c_t\alpha_t$
Require: Learning rates $\text{lr}_\eta, \text{lr}_\phi$
repeat
  1. Sample data and times
  $x_0 \sim p_{\text{data}}, \quad P_0 \leftarrow e_{x_0}$
  $(s, t) \sim \mathbb{Q}$
  2. Teacher forward to time s ▷ No gradient through this step
  for $i = 1, \ldots, N$ **do**
    $G_i^{(s)} \leftarrow \textsc{GammaSample}(\beta_s(P_0, \pi)_i)$ ▷ Algorithm 3
  end for
  $P_s \leftarrow (G_1^{(s)}, \ldots, G_N^{(s)})/\sum_j G_j^{(s)}$
  3. Generator (one denoising step)
  $U \sim \mathbb{P}_U$
  $P_0^s \leftarrow G_\eta(s, P_s, U)$
  4. Student forward to time t ▷ Differentiable through P_0^s
  for $i = 1, \ldots, N$ **do**
    $G_i^{(t)} \leftarrow \textsc{GammaSample}(\beta_t(P_0^s, \pi)_i)$ ▷ Algorithm 3
  end for
  $P_t \leftarrow (G_1^{(t)}, \ldots, G_N^{(t)})/\sum_j G_j^{(t)}$
  5. Score difference (stopped gradients)
  $\Delta \leftarrow \text{sg}\big(D_\phi(P_t, s, t)\big) - \text{sg}\big(D_{\text{teach}}(P_t, t)\big)$
  6. Generator update
  $\hat{\mathcal{L}}_\eta \leftarrow a_t \sum_{k=1}^{N} \log(P_{t,k}) \cdot \Delta_k$ ▷ Gradient flows only through $\log P_t$
  $\eta \leftarrow \eta - \text{lr}_\eta \cdot \nabla_\eta \hat{\mathcal{L}}_\eta$
  7. Auxiliary denoiser update
  $\hat{\mathcal{L}}_\phi \leftarrow -\sum_{k=1}^{N} \log(D_\phi(\text{sg}(P_t),\ s, t))_k \cdot \text{sg}(P_0^s)_k$
  $\phi \leftarrow \phi - \text{lr}_\phi \cdot \nabla_\phi \hat{\mathcal{L}}_\phi$
until converged

as the input to the forward reparameterization $P_t = F_t(G_\eta(s, P_s, U), V)$. As a result, the gradient of $\mathcal{L}^{\mathrm{G}}$ with respect to η splits into two terms:

$$\nabla_\eta \mathcal{L}^{\mathrm{G}}(\eta) = \int \mathrm{D}G_\eta^\top \underbrace{\big(\log D_\phi(P_t, s, t) - \log D_{\mathrm{teach}}(P_t, t)\big)}_{\text{direct gradient}} \,\mathrm{d}(\cdots) \tag{39}$$

$$+ \int \mathrm{D}G_\eta^\top \,\mathrm{D}F_t^\top \underbrace{\left(\frac{\partial \log D_\phi}{\partial P_t} - \frac{\partial \log D_{\mathrm{teach}}}{\partial P_t}\right)^\top G_\eta}_{\text{indirect gradient}} \,\mathrm{d}(\cdots). \tag{40}$$

As discussed in Li et al. (2026); Hoogeboom et al. (2026), the indirect gradient (40) is empirically high-variance, and current implementations discard it entirely. The resulting update, however, is no longer the gradient of any well-defined objective.

Comparison with Simplex DMD. By contrast, the surrogate loss $\hat{\mathcal{L}}(\eta)$ of Proposition J.1 yields a *single* gradient term

$$\nabla_\eta \hat{\mathcal{L}}(\eta) = \int a_t \,\mathrm{D}G_\eta^\top \,\mathrm{D}F_t^\top \frac{\mathrm{sg}\big(D_\phi(P_t, s, t) - D_{\mathrm{teach}}(P_t, t)\big)}{P_t} \,\mathrm{d}(\cdots), \tag{41}$$

which is the *exact* gradient of the KL divergence $\mathcal{L}(\eta)$ defined in (38). We highlight several structural differences with the IDLM gradient (39)–(40).

1. *Single gradient path.* In (41), the gradient flows exclusively through the forward map F_t. There is no direct term because the DMD loss is derived from $\mathrm{KL}(\mathbb{P}_t^{\eta,s} \| \mathbb{P}_t)$ via the Stein score identity for pushforward measures, which produces a gradient that is entirely of the indirect type.
2. *Exact gradient with stop-gradient.* The sg operator on Δ in (41) is not a heuristic approximation: Proposition J.1 guarantees $\nabla_\eta \hat{\mathcal{L}} = \nabla_\eta \mathcal{L}$. In contrast, discarding the indirect gradient in IDLM is an approximation without such a guarantee.
3. *Auxiliary model.* The IDLM auxiliary loss $-\int P_0^{s,\top} \log D_\phi(P_t, s, t) \,\mathrm{d}(\cdots)$ is a cross-entropy between the generator output and the auxiliary model. This is identical in structure to the auxiliary denoiser loss in Step 7 of Algorithm 4, confirming the correspondence between the auxiliary model of Li et al. (2026) and the auxiliary denoiser D_ϕ.
4. *Differentiability.* In the simplicial framework, F_t is differentiable via the Γ-reparameterization (Algorithm 3), so the Jacobian $\mathrm{D}F_t$ in (41) can be computed exactly by automatic differentiation. In discrete models (Li et al., 2026), the forward corruption is non-differentiable, necessitating biased approximations. This is one of the key advantages of the simplicial formulation.

K Extension to Birkhoff Polytope

We propose here to adapt simplicial diffusions to the generation of permutations. In that setting one must replace the simplex Δ_N with the Birkhoff polytope $\mathcal{B}_N$. In this section, we only define the forward and do not make any claim regarding a scalable methodology.

K.1 Birkhoff Polytope and Sinkhorn Theorem

The *Birkhoff polytope* $\mathcal{B}_N$ is the convex hull of all $n \times n$ permutation matrices, characterized as the set of doubly stochastic matrices. In practice, we define

$$\mathcal{B}_N = \big\{P \in \mathbb{R}_+^{N \times N} : P\mathbf{1} = \mathbf{1},\ P^\top \mathbf{1} = \mathbf{1}\big\},$$

where $\mathbb{R}_+ = [0, +\infty)$ and $\mathbf{1} = (1, \ldots, 1)^\top \in \mathbb{R}^N$. The set of extreme points of $\mathcal{B}_N$ consists of the permutation matrices:

$$\mathcal{P}_N := \big\{P \in \{0,1\}^{N \times N} : P\mathbf{1} = \mathbf{1},\ P^\top \mathbf{1} = \mathbf{1}\big\}.$$

A permutation $\sigma_0 \in \mathfrak{S}_N$ is represented by the matrix $P_0 \in \mathcal{P}_N$ whose entries are $P_{0,ij} = \delta_{\sigma_0(i)}(j)$. In particular, the i-th row of P_0 is the one-hot standard basis vector:

$$P_{0,i} = e_{\sigma_0(i)} \in \Delta_N, \qquad i \in \{1, \dots, N\},$$

where $\Delta_N = \{p \in \mathbb{R}_+^N : \sum_{j=1}^N p_j = 1\}$ denotes the probability simplex on $\{1, \dots, N\}$. We fix a reference doubly stochastic matrix $\pi \in \mathrm{ri}(\mathcal{B}_N)$, where ri denotes the relative interior. Note that π has strictly positive entries $\pi_{ij} > 0$ (e.g., the uniform matrix $\pi = \frac{1}{N}\mathbf{1}\mathbf{1}^\top$), and denote its i-th row by $\pi_i \in \Delta_N$. To map strictly positive matrices to the Birkhoff polytope, we rely on the classical theorem of Sinkhorn (Sinkhorn, 1964).

Theorem K.1 (Sinkhorn's Theorem): *For any strictly positive matrix $M \in (0, +\infty)^{N \times N}$, there exist strictly positive diagonal matrices $D_1, D_2 \in (0, +\infty)^{N \times N}$ such that*

$$P = D_1 M D_2 \in \mathcal{B}_N.$$

Moreover, P is unique and D_1 and D_2 are unique up to a positive scalar.

In what follows, we denote by $S(M) := D_1 M D_2 \in \mathcal{B}_N$ the unique doubly stochastic scaling of M. As a consequence of Section K.1 we have

$$S(AMB) = S(M), \tag{42}$$

for any diagonal matrices $A, B \in (0, +\infty)^{n \times n}$.

K.2 The Sinkhorn–Gamma Interpolating Path

Let $t \mapsto \alpha_t \in [0, 1]$ be a decreasing noise schedule on $[0, 1]$ such that $\alpha_0 = 1$ and $\alpha_1 = 0$, similar to Section 3 and define

$$h_t = \frac{\alpha_t}{1 - \alpha_t}, \qquad t \in (0, 1).$$

For a concentration parameter $\varepsilon > 0$, we finally define $\beta_t^\varepsilon(P_0, \pi) \in \mathbb{R}_+^{N \times N}$ by

$$\beta_t^\varepsilon(P_0, \pi) = \varepsilon(\pi + h_t P_0),$$

with i-th row $\beta_t^\varepsilon(P_0, \pi)_i = \varepsilon(\pi_i + h_t P_{0,i}) \in \mathbb{R}_+^n$. The *Sinkhorn–Gamma forward process* defines $P_t \in \mathcal{B}_N$ for any $t \in [0, 1]$ by first generating a positive random matrix G_t with independent Gamma entries, i.e., we have for any $i, j \in \{1, \dots, N\}$

$$G_{t,ij} \sim \mathrm{Gamma}\big(\beta_t^\varepsilon(P_0, \pi)_{ij},\, 1\big),$$

independently for any $i, j \in \{1, \dots, N\}$. Then we project G_t to the Birkhoff polytope using the Sinkhorn operator

$$P_t = S(G_t) \in \mathcal{B}_N.$$

Alternatively, one may consider the auxiliary row-stochastic matrix obtained by row-normalizing G_t

$$Q_t = \mathrm{diag}(G_t \mathbf{1})^{-1} G_t \in (\Delta_N)^N. \tag{43}$$

The following lemma establishes that row-normalization does not affect the projected state, and that the rows of Q_t are independent Dirichlet random variables.

Lemma K.1 (Gamma-Dirichlet-Sinkhorn): *Let $t \in [0, 1]$ and G_t, Q_t, P_t be defined as above. Then, we have that the rows $Q_{t,1}, \dots, Q_{t,n}$ are mutually independent and distributed as*

$$Q_{t,i} \sim \mathrm{Dir}\big(\beta_t^\varepsilon(P_0, \pi)_i\big), \qquad i \in \{1, \dots, N\}.$$

In addition, we have that $P_t = S(Q_t) = S(G_t)$.

Proof. Fix $i \in \{1, \dots, N\}$. We have that $G_{t,i,1}, \dots, G_{t,i,N}$ are independent Gamma random variables $\Gamma(\beta_t^\varepsilon(P_0, \pi)_{i,j}, 1)$ for $j \in \{1, \dots, N\}$. Using the Gamma characterization of Dirichlet distribution, we obtain that

$$Q_{t,i} = \frac{G_{t,i}}{\sum_{j=1}^N G_{t,ij}}$$

follows the Dirichlet distribution $\mathrm{Dir}\big(\beta_t^\varepsilon(P_0,\pi)_i\big)$. Mutual independence across rows follows from the independence of the Gamma variables across different rows. For the second claim, note that $G_t = D_t Q_t$ with $D_t = \mathrm{diag}(G_t \mathbf{1}) \in \mathbb{R}_{>0}^{N\times N}$. Since D_t is a positive diagonal matrix, the diagonal invariance of the Sinkhorn operator (42) directly implies

$$S(G_t) = S(D_t Q_t) = S(Q_t),$$

which concludes the proof. □

K.3 Induced Row-Wise Categorical Corruption

We can now examine the marginal row-wise law induced by the auxiliary matrix Q_t, which we recall is defined via (43).

Proposition K.1 (Induced discrete forward): *For each row $i \in \{1,\dots,N\}$, let $x_t^i \in \{1,\dots,N\}$ be a categorical random variable sampled from the i-th row of Q_t, i.e., we have for any $i \in \{1,\dots,N\}$*

$$x_t^i \mid Q_t, P_0 \sim \mathrm{Cat}(Q_{t,i}).$$

Then, we have for any $i \in \{1,\dots,N\}$

$$p_{t|0}(x_t^i \mid P_0) = \alpha_t\, \delta_{x_0(i)}(x_t^i) + (1-\alpha_t)\, \pi_i(x_t^i).$$

Proof. Fix $i \in \{1,\dots,N\}$ and $j \in \{1,\dots,N\}$. We have

$$p_{t|0}(x_t^i = j \mid P_0) = \mathbb{E}\big[P(x_t^i = j \mid Q_t, P_0) \,\big|\, P_0\big] = \mathbb{E}[Q_{t,ij} \mid P_0].$$

By Lemma K.1, $Q_{t,i} \sim \mathrm{Dir}\big(\beta_t^\varepsilon(P_0,\pi)_i\big)$ conditionally on P_0. Hence, we have

$$\mathbb{E}[Q_{t,ij} \mid P_0] = \frac{\beta_t^\varepsilon(P_0,\pi)_{ij}}{\sum_{\ell=1}^N \beta_t^\varepsilon(P_0,\pi)_{i\ell}} = \frac{\varepsilon\big(\pi_{ij} + h_t P_{0,ij}\big)}{\varepsilon \sum_{\ell=1}^N \big(\pi_{i\ell} + h_t P_{0,i\ell}\big)}.$$

Using that $\pi_i \in \Delta_N$ and $P_{0,i} \in \Delta_N$, we have

$$\sum_{\ell=1}^N \pi_{i\ell} + h_t \sum_{\ell=1}^N P_{0,i\ell} = 1 + h_t.$$

Substituting $h_t = \alpha_t/(1-\alpha_t)$, we have $(1+h_t)^{-1} = 1-\alpha_t$ and $h_t/(1+h_t) = \alpha_t$. Consequently,

$$\mathbb{E}[Q_{t,ij} \mid P_0] = (1-\alpha_t)\pi_{ij} + \alpha_t P_{0,ij} = \alpha_t\, \delta_{x_0(i)}(j) + (1-\alpha_t)\pi_{ij},$$

which concludes the proof. □

L A Particle Interpretation

Inference in SDMs is carried out by sampling from a reverse-time simplex-valued stochastic process. We show here that this process is related to the Wright-Fisher process, a standard model of population genetics. In particular, in Proposition L.1 we show that one step of the SDM can be approximated by a Wright-Fisher generation step and in Theorem L.1, we show that in the large population size limit $K \to +\infty$ we recover the original simplicial transition from SDMs. We first recall some elementary facts about the Wright-Fisher model.

L.1 Wright–Fisher Generation

Consider a population of fixed size $K \in \mathbb{N}$ with $K > 0$ whose individuals take values in $\{1,\dots,N\}$. We associate to this population the count vector $c = (c_1,\dots,c_N)$, where c_i is the number of individuals having type i so $\sum_{i=1}^N c_i = K$. In a Wright–Fisher model, this population evolves according to a resampling-mutation process that we describe below.

First, K offspring are sampled independently, with replacement, from the current population as follows. We associate to each type i a positive weight b_i indicating its propensity to generate offspring, then the probability that a chosen parent has type i is

$$r_i(c, b) = \frac{b_i c_i}{\sum_{j=1}^N b_j c_j}. \tag{44}$$

After this resampling step, these individuals mutate: conditional on a parent of type j, the offspring has type i with probability M_{ji}. Hence, the type distribution of one offspring is

$$q_i = \sum_{j=1}^N r_j(c, b) M_{ji}. \tag{45}$$

As the K offspring are conditionally independent, the corresponding new count vector C' satisfies

$$C' \mid c, b, M \sim \mathrm{Mult}(K, q),$$

where we recall that $C' = (C'_1, \dots, C'_N)$ and for any $c = (c_1, \dots, c_N) \in \mathbb{N}^N$ such that $\sum_{i=1}^N c_i = K$ we have

$$\mathbb{P}(C' = c) = \frac{K!}{\prod_{i=1}^N c_i!} \prod_{i=1}^N q_i^{c_i}.$$

L.2 Simplex Transition as an Approximated Wright–Fisher Model.

In what follows, we let $s, t \in [0, 1]$ with $s \leqslant t$. The transition presented in Proposition 3.3 given for any $i \in \{1, \dots, N\}$ by

$$P^{\kappa}_{t,i} = \frac{B_i P_{t,i}}{\sum_{j=1}^N B_j P_{t,j}}, \qquad P_s = W P^{\kappa}_t + (1 - W)V, \tag{46}$$

where (B, W, V) are Beta/Dirichlet random variables defined in Proposition 3.3.

Now assume that P_t is given by the empirical frequencies of a population, $P_t \approx \mu^K_t := C_t/K$, and that $C_t = c$. So the first part of (46) can be reinterpreted as

$$R^K_{t,i} = \frac{B_i(c_i/K)}{\sum_j B_j(c_j/K)} = \frac{B_i c_i}{\sum_j B_j c_j}. \tag{47}$$

Comparing to (44), we see that in this model $b_i = B_i$ is the propensity of type i to generate offspring.

Next we consider the mutation

$$M_{ji} = W \mathbf{1}_{\{i=j\}} + (1 - W)V_i; \tag{48}$$

i.e., after selecting a parent of type j, the offspring is given the same type as its parent with probability W and a new type is sampled from V otherwise. Combining (47) and (48) in (45) we define for any $i \in \{1, \dots, N\}$

$$Q^K_{s,i} = \sum_{j=1}^N R^K_{t,j} M_{ji} = W R^K_{t,i} + (1 - W)V_i.$$

This is exactly the second part of (46), i.e., $P_s = Q^K_s$ and corresponds to the total offspring evolution (45). Therefore, we have the following result.

Proposition L.1 (Particles Wright–Fisher generation): *Let $s, t \in [0, 1]$ with $s \leqslant t$ and let $C_t \in \mathbb{N}^N$ with $\sum_{i=1}^N C_{t,i} = K$ be a count vector of a population of size $K \in \mathbb{N}$ with $K > 0$, with empirical frequencies $P_t = C_t/K$. Let (B, W, V) be sampled according to Proposition 3.3, and let $P_s \in \Delta_N$ be defined in (46) (or Proposition 3.3). Then, one Wright–Fisher generation with parent counts C_t, propensity weights $b = B$, and mutation matrix M given by (48) yields an offspring count vector C_s which satisfies*

$$C_s \mid (C_t, B, W, V) \sim \mathrm{Mult}(K, P_s).$$

L.3 Limiting Process

Now we can check that the process with $K \in \mathbb{N}$ with $K > 0$ particles converge in distribution to the inference process used by Simplex Diffusion Models as $K \to \infty$. We first present a result that follows from standard properties of the multinomial distribution.

Proposition L.2 (Finite-population error): *Let $K \in \mathbb{N}$ with $K > 0$ and $s, t \in [0,1]$ with $s \leqslant t$. Let $\mu_s^K = C_s/K$ and $\overline{Q}_s^K = \mathbb{E}[Q_s^K \mid C_t]$. Conditional on (C_t, B, W, V),*

$$\begin{aligned}
\mathbb{E}[\mu_s^K \mid C_t, B, W, V] &= Q_s^K, \\
\mathrm{Cov}(\mu_s^K \mid C_t, B, W, V) &= \frac{1}{K}\left\{\mathrm{diag}(Q_s^K) - Q_s^K (Q_s^K)^\top\right\}, \\
\mathbb{E}\big[\|\mu_s^K - Q_s^K\|_2^2 \mid C_t, B, W, V\big] &= \frac{1 - \|Q_s^K\|_2^2}{K} \leqslant \frac{1}{K}, \\
\mathbb{P}(C_{s,i} = 0 \mid C_t, B, W, V) &= (1 - Q_{s,i}^K)^K.
\end{aligned}$$

Moreover,

$$\begin{aligned}
\mathrm{Cov}(\mu_s^K \mid C_t) = \frac{1}{K}\left\{\mathrm{diag}(\overline{Q}_s^K) - \overline{Q}_s^K (\overline{Q}_s^K)^\top\right\} \\
+ \left(1 - \frac{1}{K}\right) \mathrm{Cov}(Q_s^K \mid C_t).
\end{aligned}$$

Now we can obtain the following weak convergence result.

Theorem L.1 (Multistep particle approximation): *Let $\varepsilon > 0$, $\kappa \in [0,1)$ and $0 = t_0 < t_1 < \cdots < t_M = 1$. Assume that $p \mapsto \widehat{P}_\theta(t_m, p)$ is continuous on Δ_N for all t_i. Let $(P_{t_M}, \ldots, P_{t_1}, X_{\mathrm{out}})$ denote the joint distribution of the inference sampler and let $(C_{t_M}/K, \ldots, C_{t_1}/K, X_{\mathrm{out}}^K)$ be the particle approximation described above. Then, we have as $K \to \infty$*

$$\left(C_{t_M}/K, \ldots, C_{t_1}/K, X_{\mathrm{out}}^K\right) \xrightarrow{\mathrm{d}} \left(P_{t_M}, \ldots, P_{t_1}, X_{\mathrm{out}}\right),$$

where $X_{\mathrm{out}}^K \mid C_{t_1} \sim \mathit{Cat}\left(\widehat{P}_\theta(t_1, C_{t_1}/K)\right)$ and $X_{\mathrm{out}} \sim \mathit{Cat}\left(\widehat{P}_\theta(t_1, P_1)\right)$.

Proof. We proceed by backward induction. At $t_M = 1$, the initialization is through

$$P_{t_M} \sim \mathrm{Dir}(\varepsilon\pi), \qquad C_{t_M} \mid P_{t_M} \sim \mathrm{Mult}(K, P_{t_M}).$$

This gives, conditional on P_{t_M}, $C_{t_M}/K \xrightarrow{P} P_{t_M}$ in probability. Now assume the empirical state at time t_m converges to the corresponding state P_{t_m}. By continuity of $p \mapsto \widehat{P}_\theta(t_m, p)$, the parameters of the Beta–Dirichlet variables defining the step $t_m \to t_{m-1}$ converge to those of the continuous plug-in transition. Hence, by the continuous mapping theorem, $Q_{t_{m-1}}^K \xrightarrow{\mathrm{d}} P_{t_{m-1}}$.

Now, we need to control the error introduced by replacing $Q_{t_{m-1}}^K$ by a population of K particles. From Proposition L.1 $C_{t_{m-1}} \sim \mathrm{Mult}(K, Q_{t_{m-1}}^K)$. Proposition L.2 then gives

$$\mathbb{E}\left[\left\|\frac{C_{t_{m-1}}}{K} - Q_{t_{m-1}}^K\right\|_2^2 \middle| C_{t_m}, B, W, V\right] \leqslant \frac{1}{K}.$$

So, $C_{t_{m-1}}/K - Q_{t_{m-1}}^K \to 0$ in probability, and Slutsky's theorem yields $C_{t_{m-1}}/K \xrightarrow{d} P_{t_{m-1}}$. We can iterate this argument of all the time steps. □

Link with (Chandra et al., 2026). So far, we have described a Wright–Fisher point of view on the *inference* of SDMs. In fact, one can also approximate the forward process $P_t \mid P_0$ with a Wright–Fisher one-step generation process. We recall that we consider the corruption mechanism for any

$t \in [0,1]$ given by

$$p_{t|0}(P_t \mid P_0) = \mathrm{Dir}\big(P_t; \beta_t(P_0,\pi)\big), \qquad \beta_t(P_0,\pi) = c_t\big(\alpha_t P_0 + (1-\alpha_t)\pi\big) = b_t\big(\pi + h_t P_0\big),$$

where

$$b_t = c_t(1-\alpha_t), \qquad h_t = \frac{\alpha_t}{1-\alpha_t}.$$

Now, recall that $P_0 = e_{x_0}$ and define $K \in \mathbb{N}$ with $K > 0$. Then, we have that $P_0 = C_0/K$, where $C_0 = (0,\dots,0,K,0,\dots,0)$ and only the x_0-th coordinate is non-zero.

In that case, one can define the particle approximation of the forward process by first sampling $P_t \sim \mathrm{Dir}\big(\beta_t(P_0,\pi)\big)$ and then $C_t \sim \mathrm{Mult}(K, P_t)$.

Now, Chandra et al. (2026) also introduce a particle approximation of simplex-based approaches such as the one of Avdeyev et al. (2023); see (Chandra et al., 2026, Algorithm 4). First, sample $m \sim A(\psi,\tau_t)$ with $\tau_t = -\log(\alpha_t)$ (see (Chandra et al., 2026, p. 3)), $\psi > 0$, and

$$A(\psi,\tau_t)(m) = \sum_{k=m}^{C} \exp[-k(k+\psi-1)\tau_t/2](-1)^{k-m}\frac{(2k+\psi-1)(m+\psi)_{(k-1)}}{m!(k-m)!}\frac{(C-k+1)_{(k)}}{(C+\psi)_{(k)}}, \tag{49}$$

where $x_{(k)} = \Gamma(x+k)/\Gamma(x)$ (Tavaré, 1984). Next, let $P_t \sim \mathrm{Dir}\big(\psi\pi + mP_0\big)$ and finally let $C_t \sim \mathrm{Mult}(K, P_t)$.

The two constructions therefore have the same Dirichlet form. At any fixed time t, b_t plays the role of ψ, while the deterministic coefficient $h_t = \alpha_t/(1-\alpha_t)$ multiplying P_0 in our model corresponds to the random coefficient m/ψ in Chandra et al. (2026). Note that m is itself stochastic and follows the lineage distribution (49). For the canonical choice $c_t = \varepsilon/(1-\alpha_t)$, we have $b_t \equiv \varepsilon$, and this reduces to the time-independent identification $\psi = \varepsilon$ used previously. Letting $K \to +\infty$, $A(\psi,\tau_t)(m)$ converges to a distribution on $\mathbb{N}$ (which can be made explicit; see (Tavaré, 1984) or (Chandra et al., 2026, Lemma E.10)).

In particular, while the use of $A(\psi,\tau_t)$ makes explicit the connection with the Wright–Fisher Stochastic Differential Equation (SDE), a continuous-time limit might not exist in our case. We bypass the complexities of introducing A by simply defining a backward transition probability which satisfies compatibility rules; see Proposition 3.3.

Part III: Connection with the Literature

M Extended Related Work

M.1 General overview

In this section, we present an overview of simplex diffusion models and their applications to discrete data generation. For discrete diffusions, we refer the reader to the original paper of Austin et al. (2021) and more recent extensions (Campbell et al., 2022; Benton et al., 2024; Lou et al., 2024; Sahoo et al., 2024; Shi et al., 2024).

Several simplex diffusion models are obtained by defining a forward stochastic process directly on the simplex. Then, relying on tools from time-reversal (Haussmann & Pardoux, 1986), similar to Song et al. (2021b) in Euclidean state spaces, one can define a generative process. In Richemond et al. (2022), inspired by Baker et al. (2018), the forward process corresponds to a set of N Cox–Ingersoll–Ross processes that converge to a Gamma distribution and to a Dirichlet distribution after renormalization. Floto et al. (2023) consider another forward process based on a softmax transformation of an Ornstein–Uhlenbeck process. One of the main advantages of Floto et al. (2023) is that for any $t \in [0, 1]$ the distribution $p_{t|0}$ is a logistic normal distribution which is easy to sample. Dirichlet Diffusion Score Models (DDSMs) (Avdeyev et al., 2023) consider instead a Jacobi diffusion process as a forward process that converges to a Beta distribution. By then leveraging a stick-breaking construction, the authors obtain a forward process converging to a Dirichlet distribution. Benton et al. (2024) instead use the Wright-Fisher diffusion as a forward process, a diffusion on the simplex originating from genetics. Recently, in Chandra et al. (2026), it was shown that discrete, continuous and Wright-Fisher diffusions could arise from a similar particle perspective, where the Wright-Fisher diffusion is obtained as a limit where the number of particles goes to infinity and we consider reproduction within the particles' evolution.

Yet another approach is to define diffusion models taking into account the special geometry of the simplex. Doing so, it is then possible to define diffusion models on an associated *statistical manifold* using general techniques from Flow Matching and Riemannian diffusion models (De Bortoli et al., 2022; Huang et al., 2022; Chen & Lipman, 2024). While in Cheng et al. (2024); Davis et al. (2024); Williams et al. (2026), the authors derive the metrics on the simplex using the Fisher–Rao connection, Boll et al. (2024; 2025) consider e-connections, see (Davis et al., 2024, Appendix E.2) for a discussion of the choice of metrics. In Han et al. (2023); Mahabadi et al. (2024); Jo & Hwang (2025), the authors consider a Gaussian diffusion in the space of logits embeddings. Those different approaches, along with the one ignoring the geometry of the simplex and simply performing a linear diffusion in that space (Dunn & Koes, 2024), can be unified using the concept of α-divergence to define the statistical manifold (see Cheng et al., 2025).

Another approach incorporating the geometry of the simplex into the generation process is to consider a *constrained* forward process using either projection or reflection of the original Stochastic Differential Equation (Liu et al., 2023b; Fishman et al., 2024; 2023; Lou & Ermon, 2023).

The closest work to ours is the Dirichlet Flow Matching approach introduced by Stark et al. (2024). In this work, the authors introduced a forward process akin to ours. The main difference between the two approaches arises from inference. While Stark et al. (2024) leverage a flow perspective, our approach is akin to DDIM (Song et al., 2021a). We also provide a different temperature-parameterized process. Note that very recently Boget & Kalousis (2026) have proposed the same forward process as Stark et al. (2024). They introduce an inference procedure which, like ours, allows for stochasticity but assumes $p_{s|0,t}(P_s|P_0, P_t) = p_{s|0}(P_s|P_0)$ as in star-shaped diffusion models (Okhotin et al., 2023) (corresponding to $\kappa = 1$ in our case). Another concurrent work (Simplax) (Sakurai et al., 2026) also uses Dirichlet simplex states, but their denoiser and sampling procedure operate on categorical samples rather than the continuous simplex state itself.

We conclude by mentioning that Haviv et al. (2025) introduce Wasserstein Flow Matching, i.e., define Flow Matching on the space of distributions, similar to the simplicial approach lifting distributions on a given state space to the space of distributions of distributions. Finally, we note that

the recent work of Song et al. (2025) introducing Shortlisting Models (SLMs) also claims a simplex approach as they "aim to preserve the core principle of simplex-based methods, *gradual information growth*". To do so, they operate on the space of probability vectors. Starting from a single full probability vector, they progressively eliminate categories until the final denoised state is one-hot.

M.2 Simplex Relaxation for Discrete Diffusion

In this section, we describe the approach of Sakurai et al. (2026) and compare it with our framework. We first outline how they construct their forward process alongside their simplex relaxation. Next, we discuss their chosen training loss. Finally, we investigate their sampling procedure and show that it can be simplified to bypass the simplex relaxation, thereby highlighting a fundamental difference from Simplex Diffusion Models.

Forward Process. Sakurai et al. (2026) first consider a forward process on the categorical space given for any $t \in [0,1]$ and $x_t, x_0 \in \{1,\dots,N\}$ by

$$p_{t|0}(x_t|x_0) = \mathrm{Cat}(x_t; \alpha_t x_0 + (1-\alpha_t)\pi),$$

where $\pi \in \Delta_N$ is a probability distribution. Note that, as emphasized by Sakurai et al. (2026), one can define a compatible backward bridge transition for this forward rule by defining for any $s,t \in [0,1]$ and $s \leqslant t$ and $x_0, x_s, x_t \in \{1,\dots,N\}$ by

$$p_{s|0,t}(x_s|x_0,x_t) = \mathrm{Cat}\left(x_s; \frac{\left[\frac{\alpha_t}{\alpha_s}x_t + \left(1-\frac{\alpha_t}{\alpha_s}\right)\langle x_t,\pi\rangle \mathbf{1}\right] \odot (\alpha_s x_0 + (1-\alpha_s)\pi)}{\langle x_t, \alpha_t x_0 + (1-\alpha_t)\pi\rangle}\right).$$

We denote $r_{s|0,t}$ the mean of $p_{s|0,t}$. In particular, we have that for any $x_0, x_s \in \{1,\dots,N\}$

$$p_{s|0}(x_s|x_0) = \sum_{x_t=1}^{N} p_{s|0,t}(x_s|x_0,x_t)p_{t|0}(x_t|x_0).$$

One of the main innovations of Sakurai et al. (2026) is to introduce the simplex-valued variable $w_t \in \Delta_N$ for any $t \in [0,1]$ with

$$p(w_t|x_t,x_0) = \mathrm{Dir}(w_t; \eta_t(\alpha_t x_0 + (1-\alpha_t)\pi) + x_t).$$

Training Loss. The training loss they consider is given for a given $s,t \in [0,1]$ with $s \leqslant t$, $x_0 \in \{1,\dots,N\}$ and $w_t \in \Delta_N$

$$\mathcal{L}_{s,t} = \sum q(\tilde{x}_t|w_t)\mathrm{KL}(p_{s|0,t}(x_s|x_0,\tilde{x}_t)|p(x_s|x_\theta(\tilde{x}_t),\tilde{x}_t)).$$

This can be simplified (see (Sakurai et al., 2026, Proposition 5)) in

$$\begin{aligned}\mathcal{L}_{s,t} = &\langle w_t, \log(\alpha_t \hat{x}_\theta(x_t) + (1-\alpha_t)\pi) - \log(\alpha_t x_0 + (1-\alpha_t)\pi)\rangle \\ &+ \langle \rho_{s|0,t}, \log(\alpha_s x_0 + (1-\alpha_s)\pi) - \log(\alpha_s \hat{x}_\theta(x_t) + (1-\alpha_s)\pi)\rangle\end{aligned}$$

where $\rho_{s|0,t}$ is defined by (Sakurai et al., 2026, Equation 11)

$$\begin{aligned}\rho_{s|0,t} = &(\alpha_s x_0 + (1-\alpha_s)\pi) \\ &\odot \left[\frac{\alpha_t}{\alpha_s}(w_t \oslash (\alpha_t x_0 + (1-\alpha_t)\pi)) + \left(1-\frac{\alpha_t}{\alpha_s}\right)\langle w_t, \pi \oslash (\alpha_t x_0 + (1-\alpha_t)\pi)\rangle \mathbf{1}\right].\end{aligned}$$

In contrast, we use a simple cross-entropy loss (11), and discuss a true ELBO in Section G.

Sampling. Once the model is trained, they sample from the model as follows. Let $s,t \in [0,1]$ with $s \leqslant t$. Assume that we have access to a pair (x_t, w_t). Then, they let the denoiser predict $x_\theta(t,x_t)$. Then, they sample $x_s \sim \mathrm{Cat}(\rho_{s|0,t})$ and $w_s \sim \mathrm{Dir}(\eta_s(\alpha_s x_\theta(t,x_t) + (1-\alpha_s)\pi) + x_s)$. In (Sakurai et al., 2026, Proposition 1), it is shown that $x_s \sim p_{s|0,t}(x_s|x_\theta(t,x_t),w_t)$. Therefore, we have that for any test function $f : \{1,\dots,N\} \to \mathbb{R}$

$$\mathbb{E}[f(x_s)|x_t] = \sum f(x_s)\mathbb{E}[p_{s|0,t}(x_s|x_0,w_t)|x_t]p_{0|t}(x_0|x_t),$$

where the expectation is w.r.t. w_t. In addition, we have that

$$\mathbb{E}[p_{s|0,t}(x_s|x_0,w_t)|x_t] = (\alpha_s x_0 + (1-\alpha_s)\pi) \odot \left[\frac{\alpha_t}{\alpha_s}(\mathbb{E}[w_t|x_t] \oslash (\alpha_t x_0 + (1-\alpha_t)\pi)) + \left(1 - \frac{\alpha_t}{\alpha_s}\right)\langle \mathbb{E}[w_t|x_t], \pi \oslash (\alpha_t x_0 + (1-\alpha_t)\pi)\rangle \mathbf{1}\right].$$

We have that

$$\mathbb{E}[w_t|x_t] = \frac{\eta_t}{1+\eta_t}(\alpha_t x_0 + (1-\alpha_t)\pi) + \frac{1}{1+\eta_t}x_t.$$

Therefore, we have that

$$\mathbb{E}[p_{s|0,t}(x_s|x_0,w_t)|x_t] = \frac{\eta_t}{1+\eta_t}(\alpha_s x_0 + (1-\alpha_s)\pi) + \frac{1}{1+\eta_t}r_{s|0,t}.$$

Hence, we get that

$$\mathbb{E}[f(x_s)|x_t] = \sum f(x_s) \sum \left(\frac{\eta_t}{1+\eta_t}p_{s|0}(x_s|x_0) + \frac{1}{1+\eta_t}p_{s|0,t}(x_s|x_0,x_t)\right) p_{0|t}(x_0|x_t).$$

Therefore, we can interpret the transition proposed in Sakurai et al. (2026) as a *pure discrete* backward transition *with remasking* with remasking levels controlled by η_t.

In contrast, in our framework, we do not maintain a discrete state during the generation and instead only track a simplex state and only sample from the terminal simplex state.

N Extended Background

N.1 Discrete Diffusion Models

In this section, we outline the training and sampling procedures for discrete diffusion models. For simplicity, we describe processes over scalar variables, and the extension to sequences is similar to Section C.1. Refer to Austin et al. (2021); Campbell et al. (2022); Sahoo et al. (2024); Shi et al. (2024); Schiff et al. (2025); von Rütte et al. (2025); Gourevitch et al. (2026) for the derivations. Discrete Diffusion Models define a corruption process (1) in terms of a prior $\pi \in \Delta_N$. Prior work mainly focuses on the *absorbing* (or masked) prior $\pi^{\text{mask}} = m$, where m is the one-hot embedding of a special [MASK] token, and the *uniform* prior $\pi^{\text{unif}} = \mathbf{1}/N$. Several works study mixtures of π^{mask} and π^{unif} (Fathi et al., 2025; von Rütte et al., 2025; Liu et al., 2026; Wang et al., 2026; Zhang et al., 2026), or data-dependent priors (Alamdari et al., 2023; Vignac et al., 2023; Qin et al., 2025) but it is not clear whether elaborate priors are necessary at scale for language modeling (Sahoo et al., 2026; von Rütte et al., 2026).

Sampling. Let us refer to $p_{s|0,t}$ as the *bridge* (Gourevitch et al., 2026):

$$\begin{aligned} p_{s|0,t}(x_s \mid x_0, x_t) &= \frac{p_{t|s}(x_t \mid x_s) p_{s|0}(x_s \mid x_0)}{p_{t|0}(x_t \mid x_0)} \\ &= \text{Cat}\left(x_s; \frac{[\alpha_{t|s} e_{x_t} + (1-\alpha_{t|s})(e_{x_t}^\top \pi)\mathbf{1}] \odot [\alpha_s e_{x_0} + (1-\alpha_s)\pi]}{\alpha_t (e_{x_t}^\top e_{x_0}) + (1-\alpha_t)(e_{x_t}^\top \pi)}\right). \end{aligned} \tag{50}$$

where $\alpha_{t|s} = \alpha_t/\alpha_s$. While the bridge (50) is defined for discrete variables x_0, x_s, x_t, it is possible to define transitions $\hat{p}^\theta_{s|t}(x_s|x_t)$ by replacing e_{x_0} with the predictions of a denoiser $\mathbf{x}_\theta(t, x_t)$. Thus, with a slight abuse of notation, one can write $\hat{p}^\theta_{s|t}(x_s|x_t) = p_{s|0,t}(x_s|\mathbf{x}_\theta(t,x_t), x_t)$. After expanding (50) with the absorbing and uniform priors, we find that

$$p^{\text{mask}}_{s|0,t}(x_s \mid x_0, x_t) = \begin{cases} \text{Cat}(x_s; e_{x_0}) & \text{if } x_t = x_0 \\ \text{Cat}\left(x_s; \frac{\alpha_s - \alpha_t}{1-\alpha_t} e_{x_0} + \frac{1-\alpha_s}{1-\alpha_t} m\right) & \text{if } x_t = m, \end{cases}$$

and

$$p^{\text{unif}}_{s|0,t}(x_s \mid x_0, x_t) = \text{Cat}\left(x_s; \frac{N\alpha_t (e_{x_t}^\top e_{x_0}) e_{x_t} + (\alpha_{t|s} - \alpha_t) e_{x_t} + (\alpha_s - \alpha_t) e_{x_0} + D_{s,t}\mathbf{1}/N}{N\alpha_t (e_{x_t}^\top e_{x_0}) + 1 - \alpha_t}\right),$$

where $D_{s,t} := (1-\alpha_{t|s})(1-\alpha_s)$. For a time grid $0 = t_0 < t_1 < \ldots < t_n = 1$, the standard ancestral sampler applies the transition $\hat{p}^\theta_{s|t}(x_s|x_t)$ n times to obtain x_0:

$$x_1 \sim \mathrm{Cat}(\pi), \qquad x_{t_{i-1}} \sim \hat{p}^\theta_{t_{i-1}|t_i}(\cdot \mid x_{t_i}) \quad \text{for } i = n, \ldots, 1.$$

Alternatively, Predictor-Corrector samplers (Grathwohl et al., 2021; Campbell et al., 2022; Lezama et al., 2023; Sun et al., 2023; Campbell et al., 2024; Kim et al., 2026b; Wang et al., 2025; Liu et al., 2025; Zhao et al., 2025; Deschenaux et al., 2026a; Gourevitch et al., 2026) also exist. We describe the variant used in our experiments in Section N.2.

Temperature. All samplers can scale the logits ζ of the denoiser by a temperature $T > 0$, i.e., they use the prediction

$$\mathrm{softmax}(\zeta/T) \tag{51}$$

instead of $\mathrm{softmax}(\zeta)$. $T = 1$ recovers the denoiser prediction, and $T \to 0$ approaches greedy decoding. For SDMs, the scaled prediction replaces $\hat{P}_\theta(t, P_t)$ in the transition of Proposition 3.3.

Training. As for Variational (continuous) Diffusion Models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Kingma et al., 2021; Kingma & Gao, 2023), Discrete Diffusion Models can be trained by minimizing an expected Negative Evidence Lower Bound (NELBO). Specifically, one can first derive an expression for the standard discrete-time expected NELBO with n noise levels:

$$L_n(x_0;\theta) = \mathbb{E}\left[-\log \hat{p}^\theta_{0|t_1}(x_0 \mid x_{t_1}) + \sum_{i=2}^{n} \mathrm{KL}\left(p_{t_{i-1}|0,t_i}(\cdot \mid x_0, x_{t_i}) \,\|\, \hat{p}^\theta_{t_{i-1}|t_i}(\cdot \mid x_{t_i})\right)\right], \tag{52}$$

and find the limit of (52) as $n \to \infty$ to obtain a continuous-time objective. For the absorbing corruption processes, the limit of (52) converges to (Ou et al., 2025; Sahoo et al., 2024; Shi et al., 2024):

$$L_\infty^{\text{mask}}(x_0;\theta) = -\int_0^1 \frac{\alpha_t'}{1-\alpha_t} \mathbb{E}_{p_{t|0}}\left[e_{x_0}^\top \log \mathbf{x}_\theta(t, x_t)\right] \mathrm{d}t.$$

With uniform corruption, (52) converges to (Schiff et al., 2025; Sahoo et al., 2025):

$$\begin{aligned} L_\infty^{\text{unif}}(x_0;\theta) = -\int_0^1 \frac{\alpha_t'}{N\alpha_t} \mathbb{E}_{p_{t|0}} \Bigg[& \frac{N}{\bar{x}_{x_t}} - \frac{N}{e_{x_t}^\top \mathbf{x}_\theta(t, x_t)} \\ & - (\kappa_t \mathbb{I}_{x_t = x_0} + \mathbb{I}_{x_t \neq x_0}) \left(N \log(e_{x_t}^\top \mathbf{x}_\theta(t, x_t)) - \mathbf{1}^\top \log \mathbf{x}_\theta(t, x_t)\right) \\ & - N \frac{\alpha_t}{1-\alpha_t} \left(\log(e_{x_t}^\top \mathbf{x}_\theta(t, x_t)) - \log(e_{x_0}^\top \mathbf{x}_\theta(t, x_t))\right) \mathbb{I}_{x_t \neq x_0} \\ & - \left((N-1)\kappa_t \mathbb{I}_{x_t = x_0} - \frac{1}{\kappa_t} \mathbb{I}_{x_t \neq x_0}\right) \log \kappa_t \Bigg] \mathrm{d}t, \end{aligned}$$

where $\kappa_t := \frac{1-\alpha_t}{N\alpha_t + 1 - \alpha_t}$ and $\mathbb{I}$ is the indicator function.

N.2 Predictor-Corrector Sampler

Our PC baselines for MDMs and UDMs use a *pure* predict-and-renoise sampler. Instead of the ancestral transition $\hat{p}^\theta_{s|t}$, each step (1) samples $\hat{x}_0$ from the plug-in posterior at target time 0, i.e., the bridge (50) with $s = 0$ and e_{x_0} replaced by the denoiser prediction, and (2) re-applies the forward process (1) to $\hat{x}_0$ at the next time. Let $(t_i)_{i=0}^n$ be the sampling time grid (Section O.1), and let $\mathbf{x}_\theta(t, x_t)$ denote the denoiser, with logits scaled by the temperature T of (51). Starting from $x_{t_n} \sim \mathrm{Cat}(\pi)$, we repeat for $i = n, \ldots, 1$:

$$\begin{aligned} &\textbf{(Predict)} \qquad \hat{x}_0 \sim \mathrm{Cat}\big(\mathbf{x}_\theta(t_i, x_{t_i})\big), \\ &\textbf{(Re-noise)} \qquad x_{t_{i-1}} \sim p_{t_{i-1}|0}(\cdot \mid \hat{x}_0) = \mathrm{Cat}\big(\alpha_{t_{i-1}} e_{\hat{x}_0} + (1-\alpha_{t_{i-1}})\pi\big), \end{aligned}$$

independently for each position, and we return $\hat{x}_0$ at the last step ($t_0 = 0$, $\alpha_{t_0} = 1$). The sampler therefore uses the bridge $p_{s|0,t}(x_s \mid x_0, x_t) = p_{s|0}(x_s \mid x_0)$ with the plug-in prediction, as in star-shaped diffusion (Okhotin et al., 2023). Every step discards x_{t_i} except through the prediction $\hat{x}_0$. For MDMs, this means that tokens unmasked at earlier steps can be masked again, which lets the sampler revise earlier choices.

N.3 Self-Conditioning and Loopholing

Because discrete diffusion models operate directly on discrete state spaces, the rich categorical distribution predicted by the denoiser collapses into a single discrete token value at each sampling step. Consequently, sampling discards the uncertainty of the denoiser. Therefore, it is common to resort to Self-Conditioning (SC; Chen et al., 2022) or Loopholing (Jo et al., 2026) to propagate continuous information across sampling steps. We implement SC following Jo et al. (2026).

Architecture. We decompose the denoiser into three parts. An embedding layer $e(\cdot)$ maps the input state x_t to one vector per position. A backbone f_θ maps the embedded input and the time t to the last hidden representation h, which we call the latent. An output head g (a linear projection followed by a softmax) maps h to the predicted distribution over clean tokens. Without SC, the denoiser computes

$$h = f_\theta\big(e(x_t), t\big), \qquad \mathbf{x}_\theta(t, x_t) = g(h).$$

With SC, the denoiser additionally receives a latent h^{prev} and adds it to the input embedding after a LayerNorm $\mathrm{LN}(\cdot)$:

$$h = f_\theta\big(e(x_t) + \mathrm{LN}(h^{\text{prev}}), t\big), \qquad \mathbf{x}_\theta(t, x_t, h^{\text{prev}}) = g(h). \tag{53}$$

Setting $h^{\text{prev}} = 0$ recovers a denoiser without context. Following Jo et al. (2026), we initialize the scale and shift of LN to zero, so that SC initially leaves the input unchanged.

Training. Let $\mathrm{sg}(\cdot)$ denote the stop-gradient operator, which acts as the identity in the forward pass and blocks gradients in the backward pass. With probability $p_{\text{SC}} = 0.9$, we apply SC: a first forward pass with $h^{\text{prev}} = 0$ and without gradient produces a latent h^0, and a second forward pass (53) with $h^{\text{prev}} = \mathrm{sg}(h^0)$ produces the prediction on which we compute the loss. With probability $1 - p_{\text{SC}}$, we use a single forward pass with $h^{\text{prev}} = 0$. This avoids unrolling the sampling trajectory during training.

Matching the Training FLOPs. We count the cost of a forward pass as 1 and that of a forward and backward pass as 3. A training step with SC then costs on average $3 + p_{\text{SC}} \cdot 1 = 3 + 0.9 = 3.9$ forward-equivalents, against 3 without SC. To match the training FLOPs of models trained without SC, we train SC models for a fraction $3/3.9 \approx 0.77$ of the training steps.

Sampling. At the first sampling step, $h^{\text{prev}} = 0$. The latent h computed at step t_i is then passed as h^{prev} to step t_{i-1}. Next to the sampled discrete state, each step thus carries a deterministic continuous state. This adds one LayerNorm and one addition per step, and no extra forward pass.

N.4 Dirichlet Flow Matching

This section contains additional background on Euclidean and Dirichlet Flow Matching. To ensure consistency with the discrete diffusion literature, we denote the noise distribution by p_1 and the empirical data distribution by p_0.

Continuous Normalizing Flows. Continuous Normalizing Flows (CNFs; Chen et al., 2018; Grathwohl et al., 2018) are generative models on $\mathbb{R}^d$ that transport samples from a tractable prior $p_1 = p_{\text{noise}}$ to an unknown data distribution $p_0 = p_{\text{data}}$. To match the rest of the paper, we reverse the usual CNF time convention, in which $t = 0$ is noise. The time-dependent velocity field $u_t^\theta \colon \mathbb{R}^d \to \mathbb{R}^d$ induces a flow map $\phi_t \colon \mathbb{R}^d \to \mathbb{R}^d$ via the Ordinary Differential Equation (ODE):

$$\frac{\mathrm{d}x_t}{\mathrm{d}t} = u_t^\theta(x_t), \quad x_1 \sim p_1, \tag{54}$$

where θ parameterizes a neural network. Integrating (54) from $t = 1$ to $t = 0$ maps approximately noise x_1 to data $x_0 = \phi_0(x_1)$, defining intermediate densities $p_t = [\phi_t]_\sharp p_1$ along the probability path $\{p_t\}_{t\in[0,1]}$.

Flow Matching. Rather than leaving the velocity field u_t^θ unconstrained and optimizing θ through expensive ODE simulation, Flow Matching (FM) constructs a generative process using conditional probability paths $p_{t|0}(x_t \mid x_0)$ and conditional velocity fields $u_{t|0}(x_t \mid x_0)$, conditioned on clean data samples $x_0 \sim p_0$. From this conditional pair, we define

$$p_t(x_t) = \int p_{t|0}(x_t \mid x_0)\, p_0(x_0)\, \mathrm{d}x_0.$$

and

$$u_t(x_t) = \int u_t(x_t \mid x_0)\, p_{0|t}(x_0 \mid x_t)\, \mathrm{d}x_0 = \int u_t(x_t \mid x_0) \frac{p_{t|0}(x_t \mid x_0) p_0(x_0)}{p_t(x_t)} dx_0. \tag{55}$$

It can then be shown that these quantities (see e.g., Lipman et al. (2023)) satisfy

$$\frac{\partial p_t(x)}{\partial t} + \nabla \cdot \big(p_t(x) u_t(x)\big) = 0. \tag{56}$$

Satisfying (56) ensures that integrating an ODE of drift u_t from $t = 1$ to $t = t'$ produces a sample $x_{t'} \sim p_{t'}$.

Extension to the Simplex. Dirichlet Flow Matching (DFM; Stark et al., 2024) extends FM to the simplex Δ_N to model categorical data. In particular, DFM tackles the issue of contracting support on Δ_N by defining conditional probability paths with full support:

$$p_{t|0}(P_t|P_0) = \mathrm{Dir}\big(P_t; \mathbf{1} + h_t P_0\big).$$

Stark et al. (2024) show that when paired with the following conditional velocity field $u_{t|0}$, $(p_{t|0}, u_{t|0})$ satisfy the continuity equation. For P_0 the one-hot embedding e_i of category i,

$$\begin{aligned} u_{t|0}(P_t|P_0) &= C(P_{t,i}, t)(e_i - P_t), \\ C(P_{t,i}, t) &= -\tilde{I}_{P_{t,i}}(t+1, N-1) \frac{\mathcal{B}(t+1, N-1)}{(1-P_{t,i})^{N-1} P_{t,i}^t}, \end{aligned} \tag{57}$$

where $\mathcal{B}$ denotes the beta function, $\tilde{I}_x(a,b) = \frac{\partial}{\partial a} I_x(a,b)$ the partial derivative of the regularized incomplete beta function, and N the number of categories. Like us, Stark et al. (2024) train a denoiser $p_{0|t}^\theta(x_0|x_t) : (\Delta_N)^L \mapsto (\Delta_N)^L$ with Cross-Entropy. However, their sampling dynamics differs, as they marginalize the conditional as in (55), replacing the true posterior by the learned one. Therefore Stark et al. (2024) propose a deterministic sampler, while ours is not, even in the case $\kappa = 0$ (Section 3).

Part IV: Experimental Setup, Ablations and Results

O Extended Experimental Details

O.1 Time Samplers and Sampling Schedules

We distinguish two choices. The *time sampler* is the distribution of t during training: uniform (†) or adaptive (‡). The *sampling schedule* is the time grid used at inference: linear, cosine or adaptive. The two can be combined freely, except that the adaptive schedule requires the adaptive time sampler, since it reuses the loss profile fitted during training.

Motivation. By default, recent discrete diffusion draws the time t uniformly from $t \in [0, 1]$ during training. However, recent *continuous* diffusion language models require *adaptive* time samplers (Dieleman et al., 2022; Pynadath et al., 2026a; Batzolis et al., 2026; Chemseddine et al., 2026; Chen et al., 2026; Deschenaux & Gulcehre, 2026; Lee et al., 2026; Potaptchik et al., 2026; Raya et al., 2026; Roos et al., 2026; Yang et al., 2026) to perform well. During training, we use the derivative of the loss profile $t \mapsto \frac{\mathrm{d}}{\mathrm{d}t} L_t$ as a proxy for where the network learns the most. Assuming that the true loss increases monotonically with t, the derivative $\frac{\mathrm{d}L_t}{\mathrm{d}t} \geqslant 0$ directly defines an importance density $q(t) \propto \frac{\mathrm{d}L_t}{\mathrm{d}t}$.

High-Level Algorithm. We maintain a ring buffer to store recent (t, L_t) pairs. During the first 1k training steps, we start with $t \sim \mathcal{U}(0, 1)$ to fill the buffer. Afterwards, every 50 training steps, we approximate the loss profile using the content of the ring buffer, with either B-splines (Cox, 1972; de Boor, 1972) or with the mean per bucket, and differentiate via finite differences to obtain the empirical density $\hat{q}(t) \propto \frac{\mathrm{d}L_t}{\mathrm{d}t}$. We track an EMA $\hat{q}_{\text{EMA}}$ of the successive densities $\hat{q}$ with momentum 0.9 for stability. Let $\hat{F}_{\text{EMA}}$ denote the CDF obtained by numerically integrating $\hat{q}_{\text{EMA}}$. Finally, we evaluate $\hat{F}_{\text{EMA}}$ on a fine grid of 1k values and store $(\hat{F}_{\text{EMA}}(t), t)$. We evaluate $t = \hat{F}^{-1}_{\text{EMA}}(u)$ with linear interpolation for $u \sim \mathcal{U}(0, 1)$ during training (inverse transform sampling).

Estimating the Loss Profile. Deschenaux & Gulcehre (2026) estimates L_t by fitting a global cubic B-spline with ridge regression. In preliminary experiments, we found that a simpler bucketed piecewise-linear estimator led to stronger performance. Therefore, we partition $[0, 1]$ into $B = 50$ uniform bins, compute the mean loss within each bin independently, and linearly interpolate, placing the estimated means at the center of each bin. We compute the density as $\hat{q}(t) = \max\left(0, \frac{\mathrm{d}L}{\mathrm{d}t}\right)$. While the true $\frac{\mathrm{d}L}{\mathrm{d}t}$ should be non-negative in principle, max removes numerical artifacts that would make the density negative.

Sampling Time Schedules. We define the sampling time grid as $0 = t_0 < t_1 < \cdots < t_n = 1$ and compare three schedules. The first is the linear grid $t_i = \frac{i}{n}$. The second is the cosine grid $t_i = \cos\left(\frac{\pi}{2}\left(1 - \frac{i}{n}\right)\right)$ (Chang et al., 2022; Shi et al., 2024). The third is an adaptive grid that reuses the loss profile fitted during training. Specifically, we take the final CDF $\hat{F}_{\text{EMA}}$ of the adaptive time sampler, stored as a lookup table of 1k points, and place the grid at uniform quantiles:

$$t_i = \hat{F}^{-1}_{\text{EMA}}\left(\tfrac{i}{n}\right), \qquad i = 0, \ldots, n,$$

evaluated by linear interpolation, exactly as during training. Since $\hat{q}_{\text{EMA}} \propto \max(0, \frac{\mathrm{d}L_t}{\mathrm{d}t})$, the grid spends more denoising steps on the noise levels where the loss changes fastest. These are the same noise levels at which the adaptive sampler concentrates training. The grid needs no extra computation or tuning at inference.

O.2 Denoiser Input for SDMs

The SDM denoiser receives a sequence of simplex states $P_t = (P_t^1, \ldots, P_t^L) \in (\Delta_N)^L$ (Section C.1). As in Section N.3, an embedding layer e maps the input to one vector in $\mathbb{R}^d$ per position,

a backbone f_θ maps the embedded sequence and the time t to the last hidden representation h, and an output head g maps h to the predicted distribution $\hat{P}_\theta(t, P_t)$. Let $E \in \mathbb{R}^{N\times d}$ be the input embedding matrix, whose i-th row E_i embeds token $i \in \mathcal{X}$. We consider two embeddings of P_t, each with or without self-conditioning.

Expectation. We feed the expected embedding

$$e(P_t)^\ell = (P_t^\ell)^\top E = \sum_{i=1}^{N} P_{t,i}^\ell \, E_i = \mathbb{E}_{X\sim \mathrm{Cat}(P_t^\ell)}\big[E_X\big]. \tag{58}$$

This input keeps the full belief state: two states with the same most likely token but different uncertainty map to different inputs. Since P_t^ℓ has strictly positive entries almost surely, (58) requires a dense product with E, which costs $2Nd$ FLOPs per token, as much as the output projection. On TinyGSM ($N \approx 49$k, $d = 768$), this adds $\approx 29\%$ to the FLOPs of a forward pass of our 12-layer DiT.

Argmax. To avoid this product, we embed only the most likely token, both during training and at sampling:

$$e(P_t)^\ell = E_{\hat{x}_t^\ell}, \qquad \hat{x}_t^\ell = \mathrm{argmax}_i P_{t,i}^\ell,$$

which is a table lookup, as for MDMs and UDMs. The denoiser then sees only $\hat{x}_t^\ell$, but the sampler is unchanged: the transition of Proposition 3.3 still acts on the full state P_t, so the belief state is still carried from step to step.

Self-Conditioning. We also combine both inputs with SC, as described in Section N.3, with the simplex state P_t in place of x_t: the backbone receives $e(P_t) + \mathrm{LN}(h^{\text{prev}})$. SC gives the denoiser a rich summary of its previous prediction, which complements the argmax input in particular (Argmax + SC).

Which Input Is Used Where. On Sudoku, SDMs use the expectation input, with and without SC (Table 1). On TinyGSM, we report the expectation, Argmax and Argmax + SC inputs (Tables 29 to 32).

O.3 Dirichlet Flow Matching Baseline

To understand the benefits of our simplex based method, we also compared to the Dirichlet Flow Matching (DFM) method, as described in Section N.4. The original implementation provided by the authors at `https://github.com/HannesStark/dirichlet-flow-matching` does not include the ability to run on any of the datasets we experimented on. Furthermore, the original implementation and experiments deal with small vocab sizes (only up to 160 categories) whereas our text experiments use up to 50k tokens. We therefore re-implemented and extended the DFM method to provide a fair comparison to our approach. We made two key changes to DFM. The first was to use the adaptive time sampler during training which we found helped performance on TinyGSM for our method. In the original presentation in Stark et al. (2024), eq (14) presents the corruption distribution as (in the authors' original notation)

$$p_t(\mathbf{x}|\mathbf{x}_1 = \mathbf{e}_i) = \mathrm{Dir}(\mathbf{x}; \alpha = \mathbf{1} + t \cdot \mathbf{e}_i).$$

In the original implementation during training, t is sampled as an exponential random variable with scale parameter α_{scale}, $t \sim \mathrm{Exp}(\frac{1}{\alpha_{\text{scale}}})$. We instead sample t with the adaptive time sampler described in Section O.1.

Secondly, we improved the sampling implementation to handle much larger numbers of categories than the original implementation. Sampling in DFM requires integrating the conditional velocity field (57) towards target vertices e_i on the simplex Δ_N. Writing $b = P_{t,i} \in [0, 1]$ for the coordinate along category i, the velocity is $u_{t|0}(P_t \mid P_0 = e_i) = C(b, t)(e_i - P_t)$ with

$$C(b, t) = -\tilde{I}_b(t + 1, N - 1)\frac{\mathcal{B}(t + 1, N - 1)}{(1 - b)^{N-1}b^t}, \tag{59}$$

where $\mathcal{B}$ and $\tilde{I}$ are defined in Section N.4. Since the first argument is $t+1$, we have $\tilde{I}_b(t+1, N-1) = \frac{\partial}{\partial t} I_b(t + 1, N - 1)$.

In the original implementation of Stark et al. (2024), $\tilde{I}_b(t+1, N-1)$ is calculated through computing $I_b(t+1, N-1)$ at linearly spaced intervals of $b \in [0, 1]$ and $t \in [t_{\text{min}}, t_{\text{max}}]$ and using numerical differences to approximate the gradient. This breaks down for large vocab sizes because if we consider a sample from the uniform prior $\text{Dir}(\mathbf{1})$ then the coordinate b is distributed according to $\text{Beta}(1, N-1)$ which has mean $1/N \approx 2 \times 10^{-5}$. Therefore we need more precision in this b range than a linear spacing of b would provide. The original implementation used $\Delta b = 10^{-3}$ which immediately skips over the high probability region at 2×10^{-5} for large vocab sizes. We instead use a non-uniform geometric discretization grid that is denser around the prior mean $1/N$.

$$b_k = b_{\text{min}} \left(\frac{b_{\text{max}}}{b_{\text{min}}} \right)^{\frac{k}{M-1}}$$

with $M = 1000$, $b_{\text{min}} = 10^{-7}$ and $b_{\text{max}} = 0.999$.

Furthermore, the original implementation uses `scipy.special.betainc` to provide the values of $I_b(t+1, N-1)$. However, for I_b close to 1.0, values quickly round to exactly 1.00 when represented as floats thus giving 0 numerical gradient. We avoid this by using the complementary Beta CDF function `scipy.special.betaincc` to compute gradients in the regime where $I_b > 0.5$ because this keeps numerical values closer to 0.0 where they have more precision.

Finally, as b gets larger in (59) $\tilde{I}_b \to 0$ while $\frac{1}{(1-b)^{N-1}} \to \infty$. These two effects should approximately cancel leaving a well-conditioned $C(b, t)$ however when the terms are represented numerically, underflow and overflow can result in attempting to compute $0 \times \infty$. To avoid this situation, we can reformulate $C(b, t)$ into an exact integral where $(1-b)^{N-1}$ is canceled analytically

$$\begin{aligned}
C(b,t) &= \frac{\mathcal{B}(t+1, N-1)}{(1-b)^{N-1} b^t} \frac{\partial}{\partial t} \left[\frac{1}{\mathcal{B}(t+1, N-1)} \int_b^1 s^t (1-s)^{N-2} ds \right] \\
&= \frac{1}{(1-b)^{N-1} b^t} \left[\int_b^1 s^t (1-s)^{N-2} \ln(s) ds \right. \\
&\qquad \left. - \frac{\frac{\partial}{\partial t} \mathcal{B}(t+1, N-1)}{\mathcal{B}(t+1, N-1)} \int_b^1 s^t (1-s)^{N-2} ds \right] \\
&= \frac{1}{(1-b)^{N-1} b^t} \int_b^1 s^t (1-s)^{N-2} \left[\psi(t+N) - \psi(t+1) + \ln(s) \right] ds \qquad (60) \\
&= \frac{1}{(1-b)^{N-1} b^t} \int_0^1 (b + (1-b)u)^t (1-b)^{N-1} (1-u)^{N-2} \\
&\qquad \times \left[\psi(t+N) - \psi(t+1) + \ln(b + (1-b)u) \right] du \qquad (61) \\
&= \int_0^1 \left(1 + \frac{1-b}{b} u \right)^t (1-u)^{N-2} \left[\psi(t+N) - \psi(t+1) + \ln(b + (1-b)u) \right] du, \qquad (62)
\end{aligned}$$

where $\psi(z) = \frac{\mathrm{d}}{\mathrm{d}z} \ln \Gamma(z) = \frac{\Gamma'(z)}{\Gamma(z)}$ is the digamma function. In the above derivation, we have used in (60) that $\frac{\partial}{\partial t} \ln \mathcal{B}(t+1, N-1) = \psi(t+1) - \psi(t+N)$, and in (61) the substitution $s = b + (1-b)u$, allowing $(1-b)^{N-1}$ to cancel in (62). For $b \gtrsim 0.01$, this integral is relatively well-behaved and can be numerically integrated. We use the numerically integrated value by 48-node Gauss-Legendre quadrature instead of the numerical gradient approach for $b > 0.01$. These three improvements to the DFM sampling algorithm allow us to safely sample at the $50k$ vocabulary size scale.

To verify our implementation, we first re-ran the toy experiment from Stark et al. (2024) where the model is tasked to reproduce a synthetic categorical distribution. We train our re-implemented model with 40 categories, sample it and compute the KL-divergence to the ground truth. We obtain a KL-divergence of 0.03 approximately matching the value from Stark et al. (2024, Figure 4).

We integrate the marginal velocity field with the explicit Euler method in α from $\alpha_{\text{min}} = 1$ to $\alpha_{\text{max}} = 4\alpha_{\text{scale}}$, project onto the simplex after every step, and return the argmax of the final prediction. We use the same number of steps as for the other methods (180 on Sudoku; 64 or 512 on TinyGSM), so one step is one NFE.

O.4 SUDOKU

Data Generation. We generate 9x9 Sudoku puzzles with 30 visible cells using the backtracking generator of Alp (2024). We produce partial grids by iteratively removing cells and verifying that the solution remains unique. We ensure that the training and validation sets share no full grids. We use 200k training and 5k validation puzzles (instead of the 48k/2k split in Deschenaux & Gulcehre (2026); Kim et al. (2026b)) since we observed mild overfitting in preliminary experiments, especially when using the adaptive time sampler. We report the exact-match accuracy on the validation set.

Tokenization. We represent each puzzle as a sequence of 180 tokens drawn from a vocabulary of size 14. We map digits $\{1, \dots, 9\}$ directly to their numerical values, empty cells _ to ID 0, row separators | to ID 10, sequence boundaries [BOS] to ID 11, (unused) padding to ID 12, and the [MASK] to ID 13. To accommodate both autoregressive and diffusion algorithms, we concatenate the unsolved puzzle and its complete solution into a sequence of 180 tokens:

$$\Big(\underbrace{[\text{BOS}],\, 5, 3, _, \dots, 2,\, |, \dots, |,\, _, \dots, 9}_{\text{Prompt / Unsolved Puzzle (90 tokens)}},\; \underbrace{[\text{BOS}],\, 5, 3, 4, \dots, 2,\, |, \dots, |,\, 3, \dots, 9}_{\text{Target / Complete Solution (90 tokens)}} \Big).$$

During training, we only corrupt the target tokens and keep the first half clean as conditioning. During inference, we start from a clean prompt and denoise only the second half.

Architecture and Hyperparameters. We train an 8-layer Diffusion Transformer (DiT) (Peebles & Xie, 2023) with hidden dimension 512, 8 attention heads, an MLP expansion ratio of 4, 1D Rotary Positional Embeddings (Su et al., 2024), and untied input/output embeddings (28.6M total parameters). We apply a 0.1 dropout rate to the output projection (Sahoo et al., 2024). We train with Adam ($\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$), gradient clipping at maximum norm 1.0, and no weight decay. The learning rate warms up linearly to 3×10^{-4} over 2.5k steps and remains constant afterwards. We maintain an Exponential Moving Average (EMA) of the parameters with decay 0.9999 for evaluation for all methods. We implement time conditioning via Adaptive LayerNorm (AdaLN) (Peebles & Xie, 2023). The time-independent variants (AR and absorbing diffusion) receive a constant (zero) vector in place of time-conditioning, to share the exact same architecture with the other approaches. We train for 50k steps with a global batch size of 256 in full 32-bit precision ($\approx 77\%$ of the steps, i.e. 38,462, for SC variants, to match the training FLOPs; Section N.3).

O.5 TINYGSM

Tokenization. We train our models on TinyGSM (Liu et al., 2023a), a dataset containing approximately 11.8M synthetic grade-school math word problems associated with executable Python programs producing the correct answer. We evaluate the models zero-shot on the 1319 test problems of GSM8K (Cobbe et al., 2021) by executing a generated program and verifying its numerical output. Following Deschenaux & Gulcehre (2026), we tokenize TinyGSM with the SmolLM tokenizer (Allal et al., 2025) (49k tokens). Unlike GPT-2, SmolLM pretrains on code, and thus its tokenizer compresses Python programs better. Kim et al. (2026a) originally used the Qwen2 tokenizer (Yang et al., 2024), but its 151k tokens induce huge embedding tables, and therefore we chose SmolLM as a compact alternative.

Architecture and Hyperparameters. We train a 12-layer Diffusion Transformer (DiT) (Peebles & Xie, 2023) with hidden dimension 768, 12 attention heads, an MLP expansion ratio of 4, 1D Rotary Positional Embeddings (Su et al., 2024), and untied input and output embeddings (167.9M total parameters). Following Sahoo et al. (2024), we apply a 0.1 dropout rate to the output projection. We optimize all models using Adam (Kingma & Ba, 2014) ($\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$) without weight decay, clipping gradient norms at 1.0. We linearly warm up the learning rate to 3×10^{-4} over 2,500 steps and hold it constant thereafter. We maintain an Exponential Moving Average (EMA) of the parameters with decay 0.9999 and evaluate the EMA weights across all methods, including the autoregressive baseline. We implement time conditioning via Adaptive LayerNorm (AdaLN) (Peebles & Xie, 2023) with a conditioning dimension of 128. The time-independent models (autoregressive and absorbing discrete diffusion) receive a constant zero vector as time conditioning, to keep the architecture the same across experiments. We train each model for 250k steps with a global

batch size of 512 in full 32-bit floating-point precision ($\approx 77\%$ of the steps for SC variants, to match the training FLOPs; Section N.3). We train all SDMs with the adaptive time sampler (Section O.1). During evaluation, we run 512 sampling steps for all diffusion variants unless stated otherwise.

AST Diversity Score. We measure how structurally different the $K = 5$ programs sampled for each GSM8K problem are by comparing their abstract syntax trees (ASTs). For each sample, we keep the text from the first function definition onward (or the content of a Markdown code block, if present), parse it with Python's `ast` module, and discard samples that do not parse. We then normalize each tree: we remove docstrings, rename user-defined variables, functions and arguments to canonical identifiers within each scope (`VAR0`, `VAR1`, ...) while keeping built-in names such as `sum` or `range`, and replace keyword-argument names and import aliases by fixed placeholders. Each node is labeled by its node type; names also carry their canonical identifier, constants their type and value, and arithmetic operators their type.

For two normalized trees T_a and T_b with $|T_a|$ and $|T_b|$ nodes, we compute the tree edit distance $\mathrm{TED}(T_a, T_b)$ (Zhang & Shasha, 1989) with unit costs for insertion, deletion and relabeling, and define the similarity

$$\mathrm{sim}(T_a, T_b) = \max\Big(0,\, 1 - \frac{\mathrm{TED}(T_a, T_b)}{\max(|T_a|, |T_b|)}\Big) \in [0, 1].$$

For a problem i whose set of parsable samples $\mathcal{V}_i$ has at least two elements, the diversity is one minus the mean pairwise similarity,

$$D_i = 1 - \binom{|\mathcal{V}_i|}{2}^{-1} \sum_{a<b\in\mathcal{V}_i} \mathrm{sim}(T_a, T_b).$$

We report $100 \cdot \frac{1}{|\mathcal{I}|} \sum_{i\in\mathcal{I}} D_i$, where $\mathcal{I}$ is the set of problems with at least two parsable samples; higher is more diverse. *AST Div. (Correct)* is the same score computed only on the samples whose execution returns the reference answer, averaged over the problems with at least two such samples. It measures whether a model finds structurally different *correct* solutions, rather than rewarding diversity that comes from incorrect programs.

O.6 OpenWebText

Tokenization. We evaluate unconditional language modeling on OpenWebText (OWT) (Gokaslan & Cohen, 2019). Following prior work, we tokenize OWT using the GPT-2 tokenizer (Radford et al., 2019) (50257 tokens) and pack documents into sequences of 1024 tokens. Unlike Sudoku and TinyGSM, which require conditional prefix completion, we corrupt all 1024 positions during training and denoise sequences from scratch during inference. We measure the sample quality with the Generative Perplexity (Gen. PPL) under a pretrained GPT-2-large model and the unigram entropy. Specifically, we produce Pareto curves, following Pynadath et al. (2026a).

Architecture and Hyperparameters. We use the same 12-layer DiT architecture as in TinyGSM. We use the exact same optimizer, learning rate schedule, AdaLN time conditioning, EMA decay rate, and global batch size as well. To follow the academic literature, we train each model for 1M steps. We sample the diffusion models with 64 steps, and the AR model token by token (1024 steps).

O.7 Language Understanding

We evaluate our model on language understanding benchmarks such as ARC-easy (Clark et al., 2018), PIQA (Bisk et al., 2020) and HellaSwag (Zellers et al., 2019). The goal of this benchmark is to evaluate the accuracy of a language model to choose the correct continuation when faced with multiple choices for a given context prompt. This is a significant departure from the *sample quality* metrics that we report for Sudoku, TinyGSM and OWT.

We compare our method to GPT-2 (Radford et al., 2019), the retrained LLaMA baseline (Touvron et al., 2023) of von Rütte et al. (2025), Generalized Interpolating Discrete Diffusion (GIDDs) (von Rütte et al., 2025), Masked Diffusion Models (MDMs) as reported in Deschenaux & Gulcehre (2025) and Partition Generative Models (PGMs) (Deschenaux et al., 2026b).

In each scenario we are given a context s of length L_s and possible continuations $c^{(1)}, \dots, c^{(n)}$ ($n = 2$ in the case of PIQA, and $n = 4$ in the case of ARC-easy and HellaSwag). For a continuation $c = (c^1, \dots, c^{L_c})$ of length L_c, we denote by $x = (s, c) \in \mathcal{X}^L$ the full sequence, with $L = L_s + L_c$. For AR models, the score of a continuation is given by

$$S_{\text{AR}}(s, c) = \sum_{\ell=1}^{L_c} \log p_\theta(c^\ell \mid s, c^1, \dots, c^{\ell-1}).$$

In the case of MDMs, the score is given by

$$S_{\text{MDM}}(s, c) = -\sum_{\ell=1}^{L} \int_0^1 \frac{\alpha_t'}{1-\alpha_t} \langle x^\ell, \log \mathbf{x}_\theta(t, x_t)^\ell \rangle \mathrm{d}t. \tag{63}$$

In von Rütte et al. (2025), the integral is discretized on a uniform grid with 128 NFE, whereas in Deschenaux & Gulcehre (2025); Deschenaux et al. (2026b) the authors sample uniformly 1024 times in the interval $[0, 1]$.

In our case, the noising process retains significant information about the initial sample contrary to MDMs. Therefore, we consider an additional regularization term emphasizing that we care about the continuation and not only the whole text plausibility. In practice, we consider

$$S_{\text{SDM}}(s, c) = \sum_{\ell=L_s+1}^{L} \int_0^1 q_t \langle x^\ell, \log x_\theta(t, x_t)^\ell \rangle \mathrm{d}t - w \sum_{\ell=L_s+1}^{L} \int_0^1 q_t \langle \tilde{x}^\ell, \log x_\theta(t, \tilde{x}_t)^\ell \rangle \mathrm{d}t, \tag{64}$$

where $\tilde{x}$ is the same as x except that the context is replaced by `"Answer"`. Given a score S, we define the *accuracy* as follows. Given a context s and possible continuations $c^{(1)}, \dots, c^{(n)}$ for this context with ground-truth $c^{(1)}$, we define the accuracy as $\text{acc} = \delta_1\big(\text{argmax}_{j \in \{1,\dots,n\}} S(s, c^{(j)})\big)$, where we normalize each score by the byte length of its continuation. For each task we report the average accuracy over the test set.

The quantity q_t is the density we consider to reweight the time during training with the ring buffer, see Section O.1 for details. This practice is consistent with the recommendation of Holtzman et al. (2021) who introduced *Domain Conditional Pointwise Mutual Information*. For completeness, we sweep over $w \in [0, 1]$ and remark that $w = 0$ corresponds to disregarding the domain regularization.

We first evaluate the results for a checkpoint obtained while training with the hyper-parameters of Section O.6. However, we note that the training procedure on OWT noises the whole sequence of 1024 tokens. Hence, when we noise only the continuation while keeping the context clean in (64), the model is severely out of distribution. We therefore also train a short adaptation run, which changes how the sequence is corrupted on OWT. To train the adapted model, we initialize the training run with the baseline weights and train the model for 20,000 iterations with global batch size 256 and EMA decay rate 0.999. We consider a learning rate of 1×10^{-4} with cosine decay to 0 after 20,000 iterations. Finally, during the adaptation run, we train on OWT sequences of length 128 and corrupt them as at evaluation: each sequence is split into a clean context and a noised continuation, with lengths (L_s, L_c) drawn from the 1,024 pairs measured on the tokenized evaluation prompts.

O.8 Unconditional molecular generation

Experimental setup. We evaluate unconditional *de novo* molecular generation following the GenMol benchmark protocol (Lee et al., 2025) using the SAFE (Sequential Attachment-based Fragment Embedding) molecular representation (Noutahi et al., 2024). We train on the SAFE-GPT dataset (Noutahi et al., 2024) (v2 version) using the SAFE tokenizer, which has a vocabulary size of $|\mathcal{V}| = 1{,}880$. All sequences are padded or truncated to a fixed length of $L = 256$. In all diffusion experiments, we treat padding tokens as ordinary tokens during forward corruption, reverse denoising, and in the loss, whereas in the autoregressive (AR) baseline, padding tokens are ignored in the cross-entropy loss.

For all experiments, we use a 12-layer BERT (Devlin et al., 2019) transformer ($d_{\text{model}} = 768$, 12 attention heads, MLP dimension 3,072) similar to the 87M-parameter BERT-Base architecture of GenMol (Lee et al., 2025). We use Post-LayerNorm layers as in MaskGIT (Chang et al., 2022), but replace its MLM prediction head with a single zero-initialized linear output layer that is not tied to

the input embeddings. For masked diffusion, similar to Sahoo et al. (2024), we do not condition the network on the diffusion time, while for uniform and Simplex diffusion, we study both variants – with and without time conditioning. Time-conditioned variants use Sinusoidal time embedding with SiLU activation and embedding dimension 128, the time conditioning is provided via adaptive layer normalization (AdaLN). Diffusion models use bidirectional attention, while the AR model uses causal attention. We disable dropout on the attention probabilities and apply a hidden dropout rate of 0.1 to the embedding output and to every attention and MLP sublayer output. For Simplex Diffusion, the denoiser receives the expectation embedding (58) of each simplex state. We do not use self-conditioning in any of the diffusion models. SDM and masked diffusion are trained with cross-entropy objectives, while uniform diffusion is trained with the ELBO objective.

All models are trained for up to 400k iterations with a global batch size of 2,048 using AdamW (lr $= 3 \times 10^{-4}$ with a 2,500-step linear warmup followed by a constant schedule, weight decay 0.0, $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$), global gradient-norm clipping at 1.0, and an EMA of the parameters with decay 0.9999; all evaluations use the EMA parameters. During training, diffusion times are drawn from $[10^{-4}, 1]$: by stratified uniform sampling for MDM and UDM, and by the adaptive time sampler of Section O.1 for SDM.

During training, we monitored the evaluation metrics (see below) and noticed that some runs (including AR ones) exhibited collapse (all the evaluation metrics went to zero, while the loss went up). We therefore used early stopping. For each training run, we select the checkpoint with the highest Quality (see below) metric, computed on 1,000 generated molecules. For this evaluation, every method used the standard sampler (see below) with temperature $T = 1$ and 256 sampling steps, and we used $\kappa = 0$ for Simplex diffusion. The checkpoints were saved every 500 steps. In addition, for Simplex Diffusion, we swept over the concentration schedules: *Constant-linear*($\nu_0 = 0.2, \nu_1 = 0.5, \ell = 0.8$), *Constant-linear*($\nu_0 = 0.2, \nu_1 = 0.75, \ell = 0.8$), *Constant-linear* ($\nu_0 = 0.4, \nu_1 = 0.75, \ell = 0.2$), *Constant*($\nu = 0.5$), *Constant*($\nu = 0.25$), where the *Constant-linear*($\nu_0 = 0.4, \nu_1 = 0.75, \ell = 0.2$) schedule achieved the best Quality metric and is used for all reported Simplex Diffusion evaluations.

Evaluation Metrics. Following Lee et al. (2025), we generate $M = 1{,}000$ molecules from scratch for each of 3 sampling seeds, and report the mean $\pm$ standard deviation over seeds. Generated SAFE strings are decoded using the post-processing from (Lee et al., 2025). We report four metrics: **Validity** (percentage of the M generated sequences that decode to chemically valid molecules), **Uniqueness** (percentage of unique canonical SMILES among valid molecules), **Diversity** (one minus the mean pairwise Tanimoto similarity of Morgan fingerprints with radius 2 and 2,048 bits, over the unique valid molecules), and **Quality** (percentage of generated samples that are simultaneously valid, unique and drug-like. Drug-like molecules are defined as those satisfying quantitative estimate of drug-likeness QED $\geqslant 0.6$ (Bickerton et al., 2012) and synthetic accessibility SA $\leqslant 4$ (Ertl & Schuffenhauer, 2009), respectively, following Lee et al. (2025).).

Sampling and Hyperparameter Sweeps. For the autoregressive (AR) baseline, we generate sequences token-by-token using temperature-scaled softmax sampling with temperature T. For all diffusion models, we evaluate sampling budgets of $\{32, 64, 256, 1024\}$ steps and three sampling schedules: *linear*, *cosine*, and *adaptive* (Section O.1). The detailed results in Tables 14 and 15 use 256 steps.

We use two diffusion sampling strategies: (i) **standard (ordinary) sampling** (Table 14), where all sequence positions at step $t \to s$ are updated solely via the reverse bridge transition kernel using temperature-scaled clean predictions sampled from $\hat{P}_0^\ell = \mathrm{Softmax}(z_\theta^\ell(P_t)/T)$ (with churn $\kappa \in [0, 1]$ in Proposition 3.3 for SDM); (ii) **confidence-based (*conf.*) sampling** (Table 15), where each position $\ell \in \{1, \ldots, L\}$ is scored by its maximum predicted probability $c_\ell = \max_{v \in \mathcal{X}} \hat{P}_{0,v}^\ell$. At step $t \to s$, we commit the union of the $\lfloor \alpha_s^p L \rfloor$ most confident positions (top-fraction schedule with power p) directly to their most likely token $\arg\max_{v \in \mathcal{X}} \hat{P}_{0,v}^\ell$ (a simplex vertex for SDM). The remaining positions follow the standard update (i). Here, α_s is defined in (1) and is evaluated at the target time s of the transition from t to s. Note that the end at the step going from $t \to s$ we have a mixed of clean and noisy tokens in the sample. In the case of UDM and MDM those samples are *in distribution* while in the case of SDM they are severely *out of distribution*. Nevertheless we observe benefits from using this confidence based sampler even without retraining. Additional

Table 3: **Accuracy (%)** on Sudoku in 180 steps comparing Dirichlet Flow Matching and our SDM method. SDMs use $\nu_t^{\text{cst.-lin.}}$ with $\nu_0 = 0.4$, $\nu_1 = 0.75$, $\ell = 0.8$, as in Table 1.

Model	Linear
Continuous	
DFM (re-implemented)	$76.7_{\pm 0.75}$
SDM ($\kappa = 1.0$)	$88.4_{\pm 2.2}$
SDM (+SC) ($\kappa = 1.0$)	$\underline{\mathbf{99.1}}_{\pm 0.2}$

training of the denoiser with such strategy (e.g. training with a fraction α_t^p of positions replaced by clean vertices) could, in principle, further improve Quality and remains an interesting avenue for future work. For MDM, unmasked tokens are never re-masked, and when the top-fraction schedule is active, positions outside the committed set stay masked. For SDM and UDM, the committed set is recomputed at every step.

P Additional Experimental Results

P.1 Sudoku

We report the full Sudoku results for SDMs with $\kappa \in \{0, 0.2, 1\}$ in Tables 21 to 23, for the baselines in Table 24, and for our re-implementation of DFM in Table 3. The full tables are in Section R.1.

Churn and Concentration Schedule. SDMs perform best with $\kappa = 1$. Without SC, every concentration schedule reaches 86–92% at $\kappa = 1$, whereas no schedule exceeds 79.4% at $\kappa = 0$. The best result (91.6%) comes from $\nu_t^{\text{cst.-lin.}}$ with $\nu_0 = 0.2$, $\nu_1 = 0.75$, $\ell = 0.2$. With SC, SDMs are robust to the churn: all three values reach 98.6–99.2%.

Comparison with the Baselines. We compare against SDMs with $\nu_t^{\text{cst.-lin.}}$ with $\nu_0 = 0.4$, $\nu_1 = 0.75$, $\ell = 0.8$ and $\kappa = 1$, as in Table 1. Without SC, SDMs (88.4%) outperform DFM (76.7%) and FLMs (75.3%), and match $\mathbb{S}$-FLMs (87.3%). Among discrete models without SC, only UDMs with predictor-corrector sampling (95.8%) exceed them. With SC, SDMs reach 99.1% ($\kappa = 1$), on par with the best discrete model (MDM with SC, 99.3%).

Adaptive Time Sampler. Without SC, the adaptive time sampler improves ancestral sampling across models: MDMs (60.1% → 69.4%), UDMs (73.7% → 82.1%), FLMs (73.4% → 75.3%), $\mathbb{S}$-FLMs (83.5% → 87.3%) and SDMs at $\kappa = 0$ (73.9% → 77.1%). At $\kappa = 1$, SDMs perform similarly with both samplers (within 3 points). With SC, the adaptive sampler increases the variance across seeds for both SDMs and UDMs, and it degrades predictor-corrector sampling for the discrete baselines. We therefore train SDMs with the uniform time sampler in the main text.

Comparison with DFM. With the improvements of Section O.3, DFM reaches 76.7% on Sudoku, well below SDMs with $\nu_t^{\text{cst.-lin.}}$ with $\nu_0 = 0.4$, $\nu_1 = 0.75$, $\ell = 0.8$ and $\kappa = 1$ (88.4%, and 99.1% with SC; Table 3).

P.2 TinyGSM

Full tables are in Section R.2.

Effect of the Churn κ. Tables 29 to 32 report SDM accuracy for $\kappa \in \{0, 0.2, 1\}$. The tables cover three sampling grids and several concentration schedules (Section 4.2), as well as the three input variants (expected embedding, Argmax, Argmax + SC), two temperatures and two step budgets. Figures 8 to 11 sweep $\kappa \in \{0, 0.1, \ldots, 1\}$ for two concentration schedules. In all configurations of the tables, $\kappa = 1$ gives the highest accuracy. The gain is largest for the expected-embedding input with the adaptive grid. For $\nu^{\text{cst.-lin.}}$ with $\nu_0 = 0.2$, $\nu_1 = 0.5$, $\ell = 0.2$, accuracy rises from 12.6% to 45.8% at $T = 1$ and from 19.3% to 49.0% at $T = 0.1$ (512 steps). The trend is not always

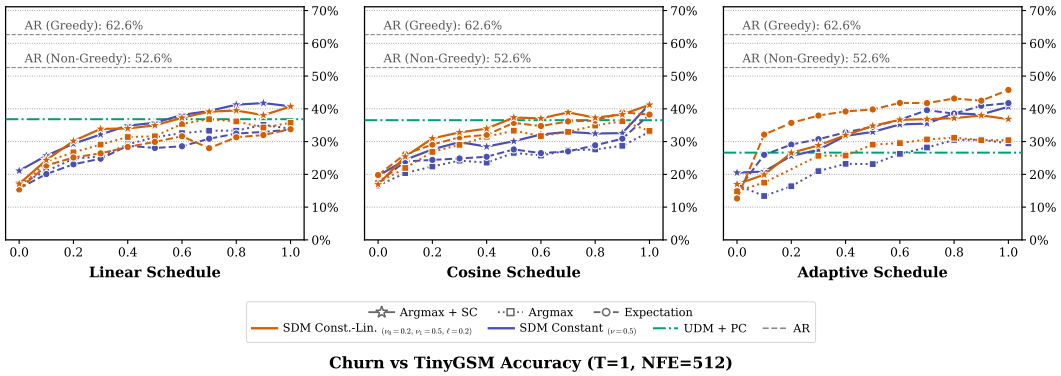

Figure 8: **Churn κ vs. TinyGSM accuracy at $T = 1.0$ with 512 steps** for the *Linear* (left), *Cosine* (center) and *Adaptive* (right) sampling schedules. We compare two concentration schedules, `Const.-Lin.` ($\nu_0 = 0.2, \nu_1 = 0.5, \ell = 0.2$, orange) and `Constant` ($\nu = 0.5$, indigo), and three input variants: *Argmax + SC* ($\star$), *Argmax* ($\blacksquare$) and *Expectation* P_t ($\bullet$). Horizontal lines show AR decoding (greedy: 62.6%; sampling: 52.6%) and UDM with the Predictor-Corrector (PC) sampler. Accuracy generally increases with κ. With the adaptive schedule, `Const.-Lin.` with the expectation input rises from 12.6% at $\kappa = 0$ to **45.8**% at $\kappa = 1$, 9.0 points above the best UDM with PC (36.8%). Exact values for $\kappa \in \{0, 0.2, 1\}$ are in Table 29.

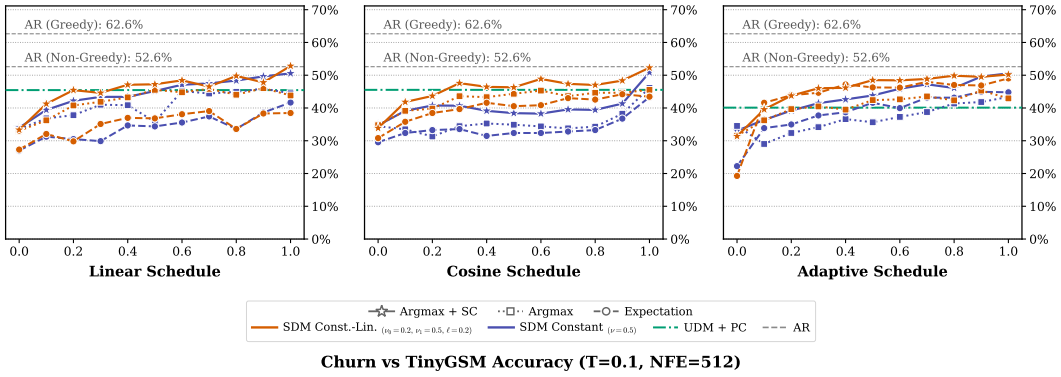

Figure 9: **Churn κ vs. TinyGSM accuracy at $T = 0.1$ with 512 steps** for the *Linear* (left), *Cosine* (center) and *Adaptive* (right) sampling schedules. We compare two concentration schedules, `Const.-Lin.` ($\nu_0 = 0.2, \nu_1 = 0.5, \ell = 0.2$, orange) and `Constant` ($\nu = 0.5$, indigo), and three input variants: *Argmax + SC* ($\star$), *Argmax* ($\blacksquare$) and *Expectation* P_t ($\bullet$). Horizontal lines show AR decoding (greedy: 62.6%; sampling: 52.6%) and UDM with the Predictor-Corrector (PC) sampler. Accuracy generally increases with κ, and *Argmax + SC* is the strongest input under the linear and cosine schedules. At $\kappa = 1$, `Constant` with *Argmax + SC* reaches **50.9**% (cosine), above the best UDM with PC (45.5%). With the adaptive schedule, `Const.-Lin.` with the expectation input rises from 19.3% at $\kappa = 0$ to 49.0% at $\kappa = 1$. Exact values for $\kappa \in \{0, 0.2, 1\}$ are in Table 31.

monotone. In some configurations, $\kappa = 0.2$ is below $\kappa = 0$, mostly with the Argmax inputs at 64 steps (e.g., 15.6% → 10.0% → 22.5% for Argmax, $\nu = 0.5$, cosine grid, $T = 1$).

Blockwise Generation. We compare full-sequence SDMs with a block-autoregressive variant (Han et al., 2023; Arriola et al., 2025) that generates blocks of 32 tokens from left to right, with the linear, cosine or adaptive grid (Figure 12). Both use $\kappa = 1$, with the expected-embedding and Argmax + SC inputs. Full-sequence SDMs are more accurate at every NFE budget, despite the stronger left-to-right bias of block generation.

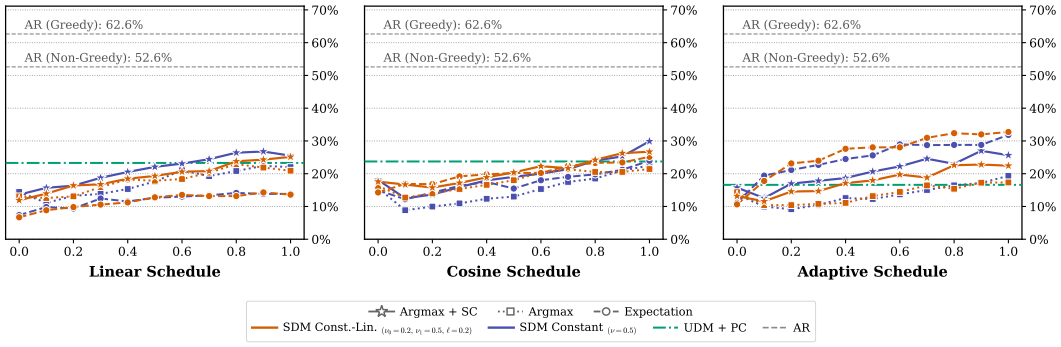

Figure 10: **Churn κ vs. TinyGSM accuracy at $T = 1.0$ with 64 steps** for the *Linear* (left), *Cosine* (center) and *Adaptive* (right) sampling schedules. We compare two concentration schedules, `Const.-Lin.` ($\nu_0 = 0.2, \nu_1 = 0.5, \ell = 0.2$, orange) and `Constant` ($\nu = 0.5$, indigo), and three input variants: *Argmax + SC* ($\star$), *Argmax* ($\blacksquare$) and *Expectation* P_t ($\bullet$). Horizontal lines show AR decoding (greedy: 62.6%; sampling: 52.6%) and UDM with the Predictor-Corrector (PC) sampler. With few steps and high temperature, the expectation input combined with the adaptive schedule and $\kappa = 1$ performs best: `Const.-Lin.` rises from 10.6% at $\kappa = 0$ to **32.7**% at $\kappa = 1$ and `Constant` from 11.2% to 31.8%, compared with 14.7% and 13.8% under the linear schedule and 23.7% for the best UDM with PC. Exact values for $\kappa \in \{0, 0.2, 1\}$ are in Table 30.

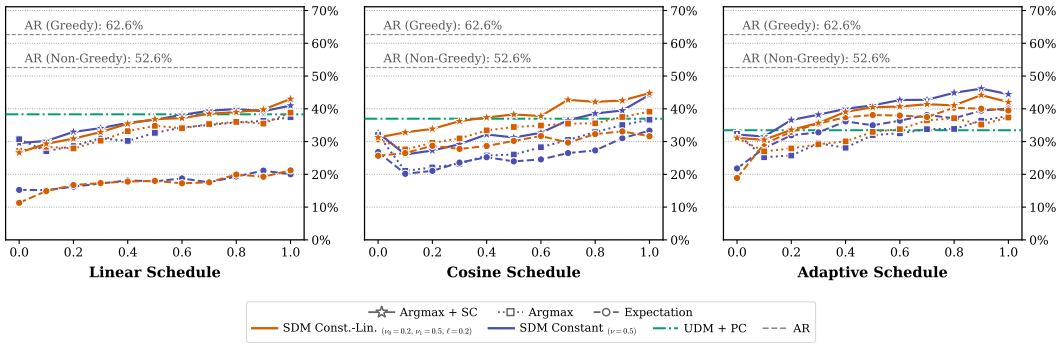

Figure 11: **Churn κ vs. TinyGSM accuracy at $T = 0.1$ with 64 steps** for the *Linear* (left), *Cosine* (center) and *Adaptive* (right) sampling schedules. We compare two concentration schedules, `Const.-Lin.` ($\nu_0 = 0.2, \nu_1 = 0.5, \ell = 0.2$, orange) and `Constant` ($\nu = 0.5$, indigo), and three input variants: *Argmax + SC* ($\star$), *Argmax* ($\blacksquare$) and *Expectation* P_t ($\bullet$). Horizontal lines show AR decoding (greedy: 62.6%; sampling: 52.6%) and UDM with the Predictor-Corrector (PC) sampler. At $\kappa = 1$, *Argmax + SC* reaches up to **44.5**% (`Constant`, adaptive), above the best UDM with PC (38.3%). With the adaptive schedule, the expectation input also improves strongly with churn, from 21.8% to 40.1% (`Constant`) and from 18.9% to 39.5% (`Const.-Lin.`). Exact values for $\kappa \in \{0, 0.2, 1\}$ are in Table 32.

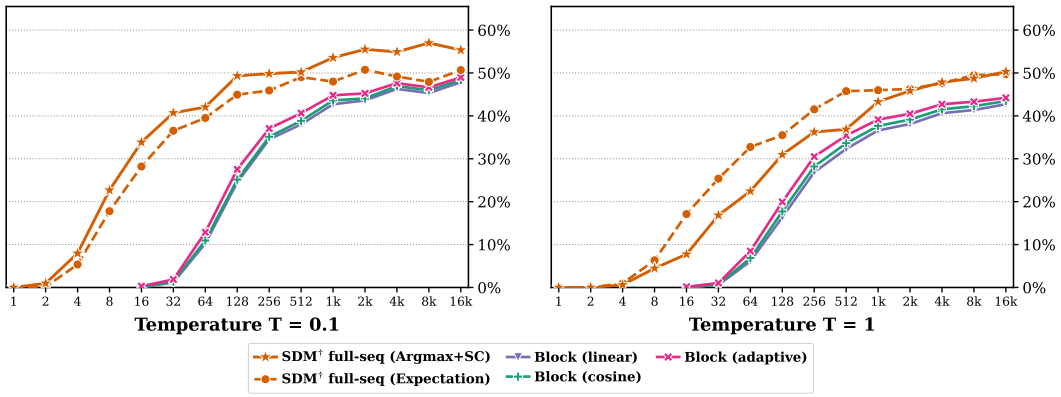

Figure 12: **Full-sequence vs. block-wise SDM generation on TinyGSM.** Accuracy vs. total NFE at $T = 0.1$ (left) and $T = 1$ (right) for block-wise generation with block size 32 under the linear, cosine and adaptive grids, compared with full-sequence SDMs (expected-embedding and Argmax + SC inputs, $\kappa = 1$). Block-wise generation is less accurate at every budget.

Adaptive Grid. The adaptive grid (Section O.1) gives the largest gains for SDMs with the expected-embedding input. These gains are largest at small step budgets: $+18.0$ points at 64 steps and $T = 1$ (Table 30).

Training Sampler and Grid for the Baselines. Tables 25 to 28 report every baseline trained with the uniform (†) and the adaptive (‡) time sampler, and sampled with the linear, cosine and adaptive grids. The adaptive training sampler improves the SC variants in 11 of 12 settings (e.g., UDM + SC from 31.7% to 39.7% at $T = 0.1$ with 512 steps) and FLMs in all four settings. It lowers UDM + PC in all four settings (e.g., from 45.5% to 40.1%), consistent with our Sudoku results. For discrete baselines, the adaptive grid changes accuracy by between -5.8 and $+5.8$ points relative to the linear grid. For every baseline, we sweep the training time sampler (uniform or adaptive), the loss (CE or ELBO for MDMs), the sampler (ancestral, PC, SC) and the sampling grid (linear, cosine, and adaptive when trained with the adaptive sampler), at the same temperatures and step budgets as SDMs, and compare SDMs against the best of these configurations. SDMs have a larger search space, since we additionally sweep the concentration schedule (4–5 per input) and the churn $\kappa \in \{0, 0.2, 1\}$; however, a single configuration ($\nu_0 = 0.2$, $\nu_1 = 0.5$, $\ell = 0.2$, $\kappa = 1$, adaptive grid) is the best Expectation SDM in three of the four settings.

Comparison with DFM. We trained our re-implemented DFM model (Section O.3) on TinyGSM, using the exact same network architecture and training setup as our method. We again find that DFM performs worse than our method (Table 4).

Table 4: **Accuracy (%)** on TinyGSM (Adaptive schedule) comparing Dirichlet Flow Matching and our SDM method across step budgets and logit temperature T. For SDM, we use the expectation input, $\nu_t^{\text{cst.-lin.}}$ ($\nu_0 = 0.2$, $\nu_1 = 0.5$, $\ell = 0.2$) and $\kappa = 1$ (Tables 29 to 32).

Model	64 NFE	512 NFE
$T = 1.0$		
DFM (re-implemented)	5.4	6.1
SDM ($\kappa = 1.0$)	32.7	45.8
$T = 0.1$		
DFM (re-implemented)	12.1	13.3
SDM ($\kappa = 1.0$)	39.5	49.0

Table 5: **Accuracy (%)** of distilled models on TinyGSM across sampling steps ($T = 1$).

Model	8	32	64	128
IDLM-MDLM (Li et al., 2026)	–	12.8	14.9	19.9
IDLM-Duo (Li et al., 2026)	–	15.4	19.0	21.4
SDM (undistilled)	6.3	25.4	32.7	35.5
SDM (distilled)	**32.1**	**37.6**	**37.8**	**39.4**

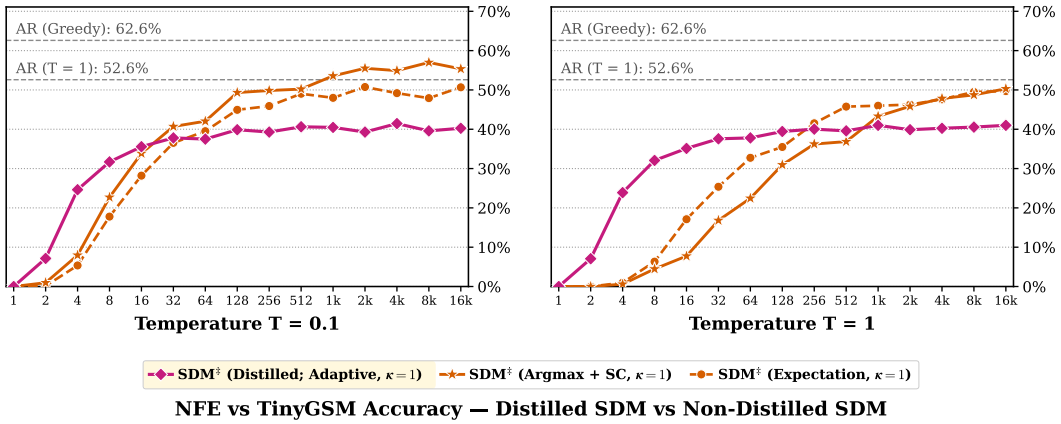

Figure 13: **Distilled vs. undistilled SDMs on TinyGSM** across sampling steps (NFE $\in [1, 2^{14}]$, $\kappa = 1$, adaptive grid) at $T = 0.1$ (left) and $T = 1$ (right). At $T = 1$, the distilled SDM solves 32.1% of the problems with 8 NFEs, against 6.3% for the undistilled SDM with the expected embedding, and 39.4% with 128 NFEs. Beyond 128 NFEs it plateaus at 40–41%, whereas the undistilled SDM keeps improving (45.8% at 512 and 49.7% at 16k NFEs).

P.3 Distillation on TinyGSM

Teacher. We distill the SDM trained with the cross-entropy loss (11), the expected-embedding input (no SC), $\alpha_t = 1 - t$, a uniform prior π, the schedule $\nu_t^{\text{cst.-lin.}}$ with $\nu_0 = 0.2$, $\nu_1 = 0.5$, $\ell = 0.2$ (Section 4.2), and the adaptive time sampler (Section O.1). This is the SDM (Expectation) of Figure 1. We sample distilled models with $\kappa = 1$, the adaptive grid and $T = 1$.

Hyperparameters. We sweep 36 configurations (Table 6) and the sampling churn $\kappa \in \{0, 0.2, 1\}$. We do not hold out a validation split: configurations are evaluated, and the reported one is selected, on the GSM8K test set, so the 32.1% in the main text is a best-of-sweep number. The conclusion does not depend on this selection: with $\kappa = 1$, all 36 configurations reach between 27.4% and 32.1% pass@1 with 8 steps (median 28.9%), above IDLM with 128 steps (21.4%). The generator loss weight λ_{gen} has little to no effect.

Comparison with IDLM. We take the IDLM results from Li et al. (2026) without re-training. They use the same tokenizer, metric (pass@1) and temperature ($T = 1$) as us (Table 5). The remaining differences are the distillation objective, the diffusion process (masked vs. simplex) and the teacher.

Accuracy vs. Steps and Diversity. Figure 13 compares distilled and undistilled SDMs across sampling budgets. Table 33 reports the multi-sample accuracy and AST diversity of the baselines. Distillation trades diversity for speed: with 8 NFEs at $T = 1$, the distilled SDM reaches an AST diversity of 17.5, vs. 35.3 for the undistilled SDM with 512 NFEs and 36.7 for AR.

P.4 Auto-Guidance on TinyGSM

While Classifier-Free Guidance (CFG; Ho & Salimans, 2022) is a standard mechanism for improving sample fidelity in diffusion models, it requires training with conditioning dropout and cannot

Table 6: **Distilled SDMs on TinyGSM** (8 steps, $\kappa = 1$, adaptive grid, $T = 1$, $K = 5$ samples per problem). We sweep the learning rate $\in \{3 \times 10^{-6}, 10^{-5}, 3 \times 10^{-5}\}$, the Adam parameters $(\beta_1, \beta_2) \in \{0, 0.9\} \times \{0.95, 0.999\}$ and the generator loss weight $\lambda_{\text{gen}} \in \{1, 2, 5\}$ (36 configurations), and report the 25 configurations with highest pass@1.

lr	β_1	β_2	λ_{gen}	pass@1 (%) ↑	pass@2 (%) ↑	pass@5 (%) ↑	AST Div. ↑	AST Div. (Correct) ↑
1e-05	0.0	0.95	1.0	**31.9**	**40.0**	48.7	17.5	7.5
1e-05	0.0	0.95	2.0	**31.9**	**40.0**	48.7	17.5	7.5
1e-05	0.0	0.95	5.0	**31.9**	**40.0**	48.7	17.5	7.5
1e-05	0.9	0.999	1.0	31.8	**40.0**	49.6	17.4	7.6
1e-05	0.9	0.999	2.0	31.8	**40.0**	49.6	17.4	7.6
1e-05	0.9	0.999	5.0	31.7	**40.0**	49.6	17.4	7.6
3e-06	0.0	0.999	2.0	30.1	39.6	**49.9**	19.3	**8.7**
3e-06	0.0	0.999	1.0	29.9	39.3	49.8	19.3	**8.7**
3e-06	0.0	0.999	5.0	29.9	39.3	49.8	19.3	**8.7**
1e-05	0.0	0.999	1.0	29.5	38.1	48.5	18.2	8.3
1e-05	0.0	0.999	2.0	29.5	38.1	48.5	18.2	8.3
1e-05	0.0	0.999	5.0	29.5	38.1	48.5	18.2	8.3
3e-05	0.9	0.95	1.0	29.2	37.1	46.6	**20.5**	8.1
3e-05	0.9	0.95	2.0	29.2	37.1	46.6	**20.5**	8.1
3e-05	0.9	0.95	5.0	29.2	37.1	46.6	**20.5**	8.1
3e-06	0.9	0.999	1.0	28.9	38.3	48.7	17.5	8.4
3e-06	0.9	0.999	2.0	28.9	38.2	48.7	17.6	8.3
3e-06	0.9	0.999	5.0	28.9	38.3	48.7	17.5	8.4
3e-05	0.0	0.95	1.0	28.8	37.0	44.9	19.6	8.4
3e-05	0.0	0.95	2.0	28.8	36.9	44.9	19.6	8.2
3e-05	0.0	0.95	5.0	28.8	36.9	44.9	19.6	8.2
3e-05	0.0	0.999	1.0	28.6	35.8	45.3	19.6	8.4
3e-05	0.0	0.999	2.0	28.6	35.8	45.3	19.6	8.5
3e-05	0.0	0.999	5.0	28.6	35.8	45.3	19.6	8.4
3e-05	0.9	0.999	1.0	28.6	36.6	44.7	18.4	8.0

be applied directly to unconditional generation or pre-trained models without dropout tokens. To sharpen generation quality at inference time without modifying the training objective, we adapt *Auto-Guidance* (Karras et al., 2024) to Simplex Diffusion Models (SDMs). Specifically, we guide the fully converged model θ_{final} away from an earlier, unconverged checkpoint θ_K saved during training. More precisely, we modify the logits and probabilities of the Simplex Diffusion Model as follows

$$\tilde{\zeta}(x_t, t) = \zeta_{\theta_{\text{final}}}(x_t, t) + w\big(\zeta_{\theta_{\text{final}}}(x_t, t) - \zeta_{\theta_K}(x_t, t)\big), \qquad \hat{p}_0(x_t, t) = \text{softmax}\left(\frac{\tilde{\zeta}(x_t, t)}{T}\right),$$

where $w \geqslant 0$ controls the guidance strength ($w = 0$ recovers the un-guided baseline).

We apply Auto-Guidance to the TinyGSM SDM with the expected embedding (no SC; $\nu_0 = 0.2$, $\nu_1 = 0.5$, $\ell = 0.2$), sampled with 64 steps. The guided model is the EMA checkpoint at 250k steps, and the guiding model uses the raw (non-EMA) parameters at step $K \in \{0, 10\text{k}, 20\text{k}, 30\text{k}, 50\text{k}, 100\text{k}, 150\text{k}, 200\text{k}, 249.5\text{k}\}$. We sweep $w \in \{0.05, 0.1, 0.15, 0.2, 0.25, 0.35, 0.5, 0.75, 1.0, 1.5\}$ and logit temperatures $T \in \{0.01, 0.1, 1.0\}$ for each churn $\kappa \in \{0.0, 0.2, 1.0\}$, with 3 evaluation seeds. Figure 14 shows TinyGSM pass@1 accuracy for $w \in \{0.1, 0.25, 0.5, 1.0\}$. We observe that guiding against an early-to-intermediate checkpoint ($K \in [20\text{k}, 50\text{k}]$) maximizes the accuracy for all churn levels κ, while for checkpoints close to convergence, the accuracy gets closer to the unguided baseline.

Figure 15 summarizes the best auto-guided accuracy ($w^\star$) alongside the accuracy gain ($\Delta\%$) over the un-guided baseline ($w = 0$) across logit temperatures $T \in \{0.01, 0.1, 1.0\}$. We observe improvements over the baseline in all frameworks we investigate. In particular, the Auto-Guidance has the strongest effect with low churn and high temperature.

P.5 OpenWebText

Inference Interventions. In the following paragraph, we show how different interventions can push the Pareto frontier in terms of Generative Perplexity and unigram entropy, see Pynadath et al.

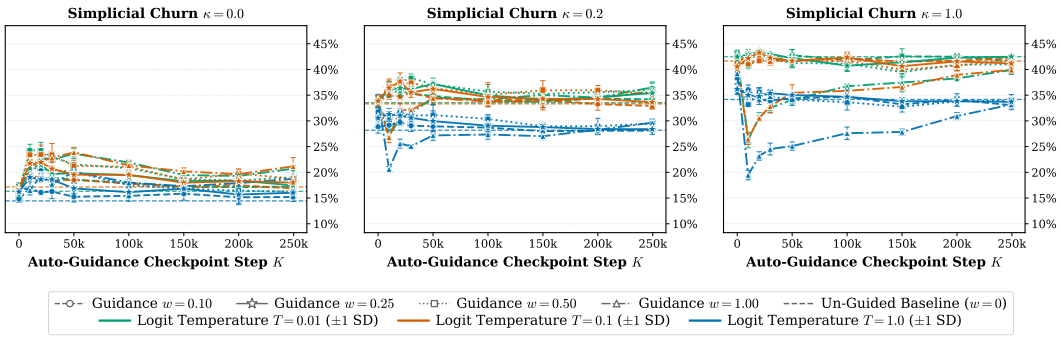

Figure 14: **TinyGSM accuracy (%) vs. Auto-Guidance checkpoint step K for different churn regimes ($\kappa \in \{0.0, 0.2, 1.0\}$), with 64 sampling steps.** We compare logit temperatures $T = 0.01$ (green), $T = 0.1$ (orange) and $T = 1.0$ (blue) for different guidance weights $w \in \{0.1, 0.25, 0.5, 1.0\}$ against the corresponding un-guided baselines ($w = 0$, dashed horizontal lines). Error bars denote a variation of one standard deviation across 3 evaluation seeds. Early-to-intermediate checkpoints ($K \in [20\text{k}, 50\text{k}]$) consistently achieve peak accuracy.

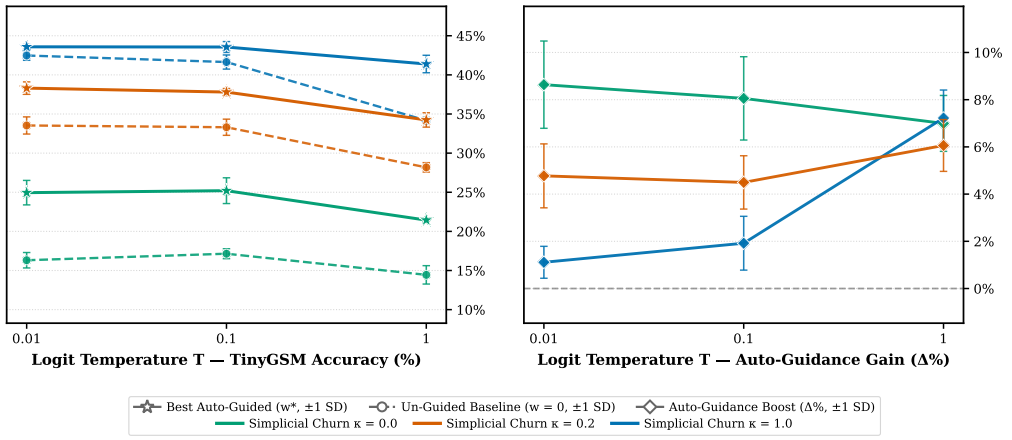

Figure 15: **Logit temperature sensitivity and Auto-Guidance gain on TinyGSM ($T \in \{0.01, 0.1, 1.0\}$). Left:** Validation accuracy (%) of the best auto-guided model ($w^\star$, solid lines with stars) vs. the un-guided baseline ($w = 0$, dashed lines with circles) for $\kappa \in \{0.0, 0.2, 1.0\}$. **Right:** Accuracy gain (Δ%) from Auto-Guidance. Error bars denote a variation of one standard deviation across 3 evaluation seeds.

(2026b) for a discussion on those evaluations. We explore those interventions for the four different classes of models we investigate. Namely, we propose interventions for Autoregressive (AR) models, Masked Diffusion Models (MDMs), Uniform Diffusion Models (UDMs) and Simplex Diffusion Models (SDMs). Most of those techniques can be interpreted as being part of a logit shaping pipeline. The model was trained according to Section O.6.

First, we consider some nucleus sampling techniques. For completeness, we recall the process of nucleus sampling adapted from Holtzman et al. (2019). In the case of one token we denote $\{\zeta_1, \dots, \zeta_N\}$, the proposed logits, i.e., we have that $\log \hat{P}_{0|t}(\cdot|P_t) = \{\zeta_1, \dots, \zeta_N\}$ in the case of SDMs for instance. Next, we denote $\{\zeta_{\varphi(1)}, \dots, \zeta_{\varphi(N)}\}$ the sorted set of logits (in descending order, i.e., $\zeta_{\varphi(j)} \geqslant \zeta_{\varphi(j+1)}$ for any $j \in \{1, \dots, N-1\}$). Next, we denote $k \in \{1, \dots, N\}$ the smallest integer such that $\sum_{j=1}^{k} p_{\varphi(j)} \geqslant p$, where p is a hyperparameter and $\{p_1, \dots, p_N\} = \mathrm{softmax}(\{\zeta_1, \dots, \zeta_N\})$. We denote ϕ the inverse of φ, i.e. for any $i \in \{1, \dots, N\}$, $\varphi(\phi(i)) = i$. For any $j \in \{1, \dots, N\}$, we denote $\hat{\zeta}_j = \zeta_j$ if $\phi(j) \leqslant k$ and $\hat{\zeta}_j = -\infty$ otherwise. Next, we consider

TOP such that

$$\mathrm{TOP}(p, \{\zeta_1, \dots, \zeta_N\}) = \{\hat{\zeta}_1, \dots, \hat{\zeta}_N\}. \tag{65}$$

Another intervention we consider is temperature scaling of the logits. More precisely, we consider

$$\mathrm{TEMP}(T, \{\zeta_1, \dots, \zeta_N\}) = \{\zeta_1/T, \dots, \zeta_N/T\}. \tag{66}$$

For the first intervention we consider T in (66) to be in the following set

$$\{0.20, 0.30, 0.40, 0.50, 0.60, 0.65, 0.70, 0.75, 0.80, 0.85, 0.90, 0.95, 1.00, 1.05, 1.10, 1.18, 1.28\}. \tag{67}$$

and $p \in \{0.92, 0.96\}$ in (65). This yields 34 sweeps for each family UDM, MDM, SDM and AR after this intervention.

For the second intervention, which only applies to diffusion methods, we consider temperature annealing. Let $t \in [0, 1]$ be the diffusion time of the forward process, we consider a similar sweep as before but instead consider a temperature annealing procedure where

$$T = T_{\text{start}} + (T_{\text{end}} - T_{\text{start}})t^{1.5},$$

with T_{end} given by (67) and $T_{\text{start}} = 0.8T_{\text{end}}$. We do not claim that the power relationship with coefficient 1.5 and $T_{\text{start}} = 0.8T_{\text{end}}$ are optimal but we found those values to give good results in early experiments. Note that a linear temperature annealing was already considered in DiffusionGemma Team et al. (2026) with $T_{\text{start}} = 0.4$ and $T_{\text{end}} = 0.8$. Similar power law temperature annealing schedule were also identified in Chang et al. (2022). Since AR has no corruption time, we instead anneal its temperature over token positions. We always consider TEMP and then follow it by TOP.

For the third intervention, which is by far the most influential one, we consider a *frequency penalty*. The frequency penalty is a *sequence* based penalty. In what follows, in the case of UDM and MDM, $x_t \in \{1, \dots, N\}^L$ where L is the sequence length and N is the vocabulary size (in the case of MDM we assume that one of the token is the masked one denoted $[\text{MASK}]$). We denote the count variable $\{c_{t,v}\}_{v=1}^N \in \mathbb{N}^N$ which is defined for any $v \in \{1, \dots, N\}$ with $v \neq [\text{MASK}]$

$$c_{t,v} = \sum_{j=1}^{L} \delta_v(x_{t,j}). \tag{68}$$

In addition, in the case $v = [\text{MASK}]$, $c_{t,v} = 0$. In other words, $c_{t,v}$ is the count of tokens which have values v in the sequence x_t, except for potentially the mask token. In the case of the simplex diffusion model, we slightly modify the definition of the count in (68) and we define for any $v \in \{1, \dots, N\}$ with $v \neq [\text{MASK}]$

$$c_{t,v} = \sum_{j=1}^{L} \delta_v(\operatorname{argmax} P_{t,j}). \tag{69}$$

Let $\{c_{t,v}\}_{v=1}^N \in \mathbb{N}^N$ be defined either with (68) or (69). We introduce the frequency penalty FREQ given by

$$\mathrm{FREQ}(\lambda, \{\zeta_1, \dots, \zeta_N\}) = \{\zeta_1 - \lambda \log(1 + c_{t,1}), \dots, \zeta_N - \lambda \log(1 + c_{t,N})\}.$$

Repetition penalties were also considered in Xu et al. (2022). We always consider TEMP and then follow it by FREQ and then by TOP. In the case of AR one can define a similar intervention on the tokens already generated. We consider a regularization penalty $\lambda \in \{1.5, 3.0, 4.5, 6.0, 8.0\}$ for all the sweeps.

For the fourth and final intervention, we consider another token penalty but at the *local* level, contrary to the *global* level of the frequency penalty of the third intervention. In particular, we consider $\{\hat{c}_{t,j,v}\} \in \mathbb{R}^{L \times N}$ given for any $j \in \{1, \dots, L\}$ and $v \in \{1, \dots, N\}$ with $v \neq [\text{MASK}]$ by

$$\hat{c}_{t,j,v} = \log\left(1 + \frac{\alpha_t N}{1 - \alpha_t}\right) \delta_v(x_{t,j}). \tag{70}$$

We set $\hat{c}_{t,j,v} = 0$ for $v = [\text{MASK}]$. Finally, similarly to (69), we can define the simplicial counterpart of (70) as

$$\hat{c}_{t,j,v} = \log\left(1 + \frac{\alpha_t N}{1 - \alpha_t}\right) \delta_v(\operatorname{argmax} P_{t,j}).$$

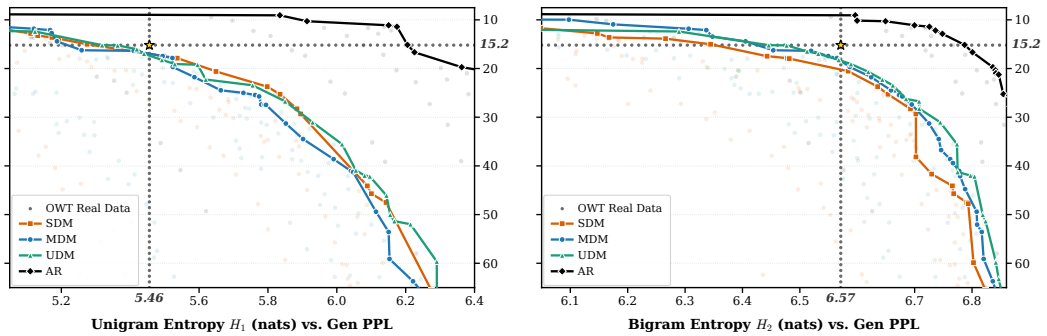

Figure 16: **OpenWebText ($L = 1024$) Pareto frontiers across generative families.** Best cumulative Pareto frontiers of Generative Perplexity (scored by GPT-2 Large; *top is lower/better*) versus sequence token diversity (*right is higher/better*) for Simplex Diffusion Models (**SDM**, orange), Masked Diffusion (**MDM**, blue), Uniform Discrete Diffusion (**UDM**, green), and the Autoregressive baseline (**AR**, black). **Left:** Unigram token entropy H_1 (nats). **Right:** Bigram token entropy H_2 (nats). Dotted reference lines and the gold star ($\star$) denote the measured OpenWebText validation distribution ($H_1 = 5.46$ nats, $H_2 = 6.57$ nats, Gen PPL $= 15.19$).

Table 7: **Vertical Cuts on Unigram Entropy ($H_1 \geqslant \tau$): Minimum Generative Perplexity (GenPPL $\downarrow$) on OpenWebText.** AR is our model with the same architecture as the diffusion models, trained on OWT (1024 steps). The best diffusion method is **bold**, second best is underlined. The associated hyperparameters $(p,\ T_{\text{start}},\ \lambda,\ \gamma)$ are shown in grey below each entry. The real OWT validation data ($\star$) has $H_1 = 5.46$ and GenPPL $= 15.2$.

Threshold ($H_1 \geqslant \tau$)	**SDM (Simplex)**	**MDM (Masked)**	**UDM (Uniform)**	**AR (1024 steps)**
$H_1 \geqslant 5.20$	14.0	14.7	**13.5**	8.9
Hyperparameters $(p, T_{start}, \lambda, \gamma)$	(.96, .40, 3.0, 2.0)	(.96, .70, 1.5, 0.0)	(.92, .60, 1.5, 0.0)	(.96, .65, 3.0, 0.0)
$H_1 \geqslant 5.35$	15.8	16.3	**15.2**	8.9
Hyperparameters $(p, T_{start}, \lambda, \gamma)$	(.92, .30, 6.0, 2.5)	(.92, .60, 3.0, 0.0)	(.96, .60, 1.5, 0.0)	(.96, .65, 3.0, 0.0)
$H_1 \geqslant 5.46$ ($\star$ OWT = 15.2)	17.0	**16.9**	17.3	9.0
Hyperparameters $(p, T_{start}, \lambda, \gamma)$	(.92, .30, 6.0, 2.5)	(.92, .50, 4.5, 0.0)	(.92, .70, 1.5, 0.0)	(.96, .65, 3.0, 0.0)
$H_1 \geqslant 5.65$	**20.6**	24.0	22.5	9.0
Hyperparameters $(p, T_{start}, \lambda, \gamma)$	(.96, .30, 8.0, 2.0)	(.92, .70, 3.0, 0.0)	(.96, .60, 3.0, 0.0)	(.96, .65, 3.0, 0.0)
$H_1 \geqslant 5.85$	**26.2**	31.2	26.7	9.3
Hyperparameters $(p, T_{start}, \lambda, \gamma)$	(.92, .40, 6.0, 2.0)	(.92, .65, 4.5, 0.0)	(.96, .70, 3.0, 0.0)	(.92, .70, 3.0, 0.0)

We are now ready to define the *local frequency* intervention FREQLOC given by

$$\text{FREQLOC}(\gamma, \{\zeta_1, \ldots, \zeta_N\}) = \{\zeta_1 - \gamma \hat{c}_{t,j,1}, \ldots, \zeta_N - \gamma \hat{c}_{t,j,N}\},$$

where here we have assumed that $\{\zeta_1, \ldots, \zeta_N\}$ are the logits at position $j \in \{1, \ldots, L\}$. We consider $\gamma \in \{2.0, 2.5\}$ for SDM only. We do not consider the intervention for AR as it already yields models which are scoring better than real data on the Generative Perplexity and Entropy Pareto frontier. In the case of UDM the intervention did not improve the Pareto frontier and is therefore not reported. In the case of MDM the local frequency constraint does not change the prediction since we do not consider remasking. Therefore, we only report SDMs results.

In Figure 16, we illustrate the obtained Pareto frontiers after all those interventions. Note that the AR Pareto frontier scores *better* than the real data on OpenWebText, thereby putting into question the validity of generative frontiers on OWT as a meaningful benchmark for language generation. In Figure 17 and Figure 18, we show the effect of each intervention on the Pareto frontier for all four models that we are investigating. The best hyperparameters for each model at matched entropy are reported in Table 7 and at matched Generative Perplexity are reported in Table 8.

For each evaluation point, we generate 128 unconditional sequences of length 1024 (evaluated in batches of 2 over 64 batches) using 64 sampling steps for all diffusion models and 1024 steps (token-by-token) for the autoregressive baseline.

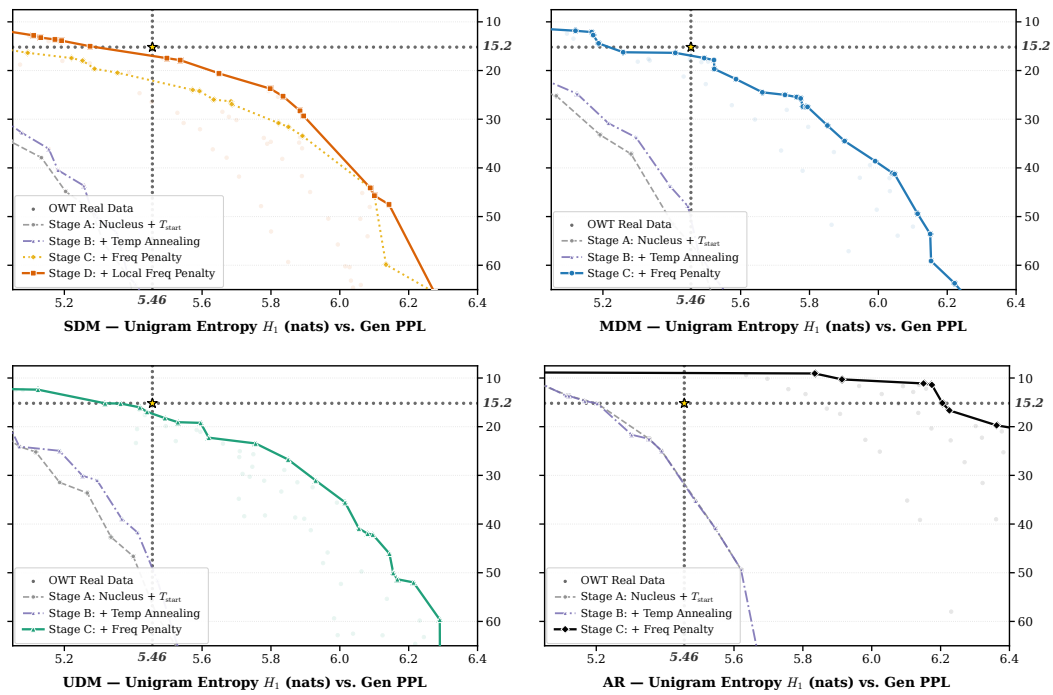

Figure 17: **Progressive intervention ablations on Unigram Entropy H_1 (nats) vs. Generative Perplexity.** Cumulative Pareto frontiers per family across the intervention stack: **Stage A** (nucleus truncation $p \in \{0.92, 0.96\}$ + initial temperature T_{start}), **Stage B** (+ power-law temperature annealing $T_{\text{start}} = 0.80\,T_{\text{end}}$, exponent 1.5), **Stage C** (+ sequence-level frequency penalty $\lambda \in \{1.5, 3.0, 4.5, 6.0, 8.0\}$), and **Stage D** (+ local frequency penalty γ, **SDM** only).

Table 8: **Horizontal Cuts on Generative Perplexity (GenPPL $\leqslant \tau_{\text{PPL}}$): Maximum Unigram Entropy ($H_1 \uparrow$) on OpenWebText.** AR is our model with the same architecture as the diffusion models, trained on OWT (1024 steps). The best diffusion method is **bold**, second best is underlined. The associated hyperparameters (p, T_{start}, λ, γ) are shown in grey below each entry. The real OWT validation data ($\star$) has GenPPL $= 15.2$ and $H_1 = 5.46$.

Budget (GenPPL $\leqslant \tau_{\text{PPL}}$)	SDM (Simplex)	MDM (Masked)	UDM (Uniform)	AR (1024 steps)
GenPPL $\leqslant 12.5$	5.09	**5.17**	5.13	6.18
Hyperparameters $(p, T_{start}, \lambda, \gamma)$	(.92, .30, 4.5, 2.5)	(.96, .50, 3.0, 0.0)	(.96, .50, 1.5, 0.0)	(.96, .40, 8.0, 0.0)
GenPPL $\leqslant 14.0$	5.20	5.18	**5.23**	6.20
Hyperparameters $(p, T_{start}, \lambda, \gamma)$	(.92, .40, 3.0, 2.0)	(.96, .40, 4.5, 0.0)	(.96, .50, 1.5, 0.0)	(.96, .40, 8.0, 0.0)
GenPPL $\leqslant 15.2$ ($\star$ **OWT** $H_1 = 5.46$)	5.29	5.22	**5.32**	6.21
Hyperparameters $(p, T_{start}, \lambda, \gamma)$	(.96, .40, 3.0, 2.0)	(.92, .70, 1.5, 0.0)	(.96, .50, 1.5, 0.0)	(.92, .65, 4.5, 0.0)
GenPPL $\leqslant 18.0$	**5.54**	5.52	5.48	6.29
Hyperparameters $(p, T_{start}, \lambda, \gamma)$	(.96, .30, 6.0, 2.5)	(.96, .50, 4.5, 0.0)	(.96, .65, 1.5, 0.0)	(.96, .65, 4.5, 0.0)
GenPPL $\leqslant 22.0$	**5.72**	5.59	5.62	6.43
Hyperparameters $(p, T_{start}, \lambda, \gamma)$	(.92, .40, 4.5, 2.0)	(.96, .65, 3.0, 0.0)	(.96, .50, 3.0, 0.0)	(.96, .50, 8.0, 0.0)

P.6 Language Understanding

We describe the setup in Section O.7.

The results sweeping on the weight w in (64) and comparing the baseline model with the adapted model are reported in Figure 19. The adapted model outperforms the baseline model, which confirms that the baseline is out of distribution when only the continuation is noised. Finally, our main results and comparison with benchmarks are presented in Table 9.

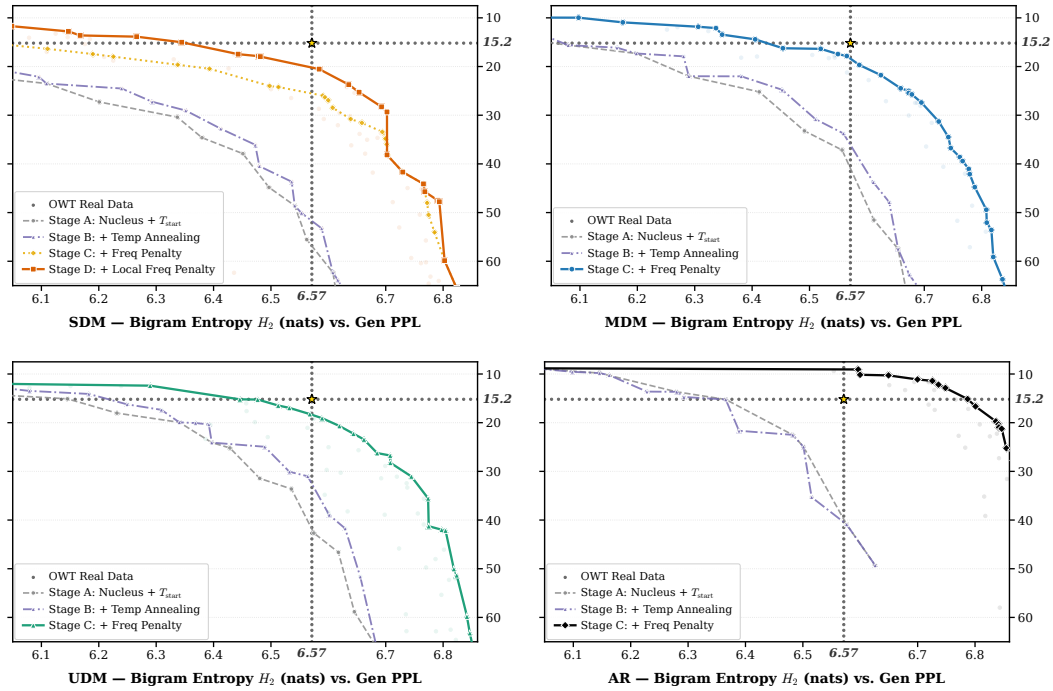

Figure 18: **Progressive intervention ablations on Bigram Entropy H_2 (nats) vs. Generative Perplexity.** Order-sensitive bigram diversity control (H_2) across the same nested intervention stages (**Stage A** → **Stage D**) for **SDM**, **MDM**, **UDM**, and **AR**. Both the sequence frequency penalty (**Stage C**) and the local frequency penalty (**Stage D**) preserve gains along the bigram entropy H_2 axis.

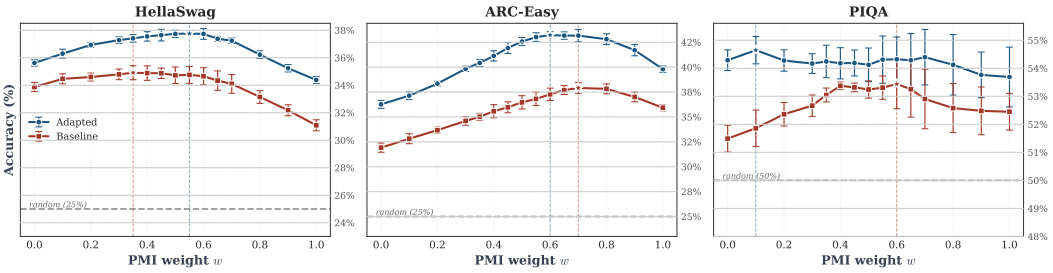

Figure 19: We present a sweep on the Pointwise Mutual Information (PMI) weight w in (64). For each of the three tasks that we investigate HellaSwag (Zellers et al., 2019), ARC-easy (Clark et al., 2018) and PIQA (Bisk et al., 2020), we report the influence of w on the final results. Note that $w = 0$ corresponds to the setup which is most comparable to the MDM scoring (63) (even though MDM noise is applied on the whole sequence while ours focus on the continuation). We denote $w^\star$ the best weighting for each task. The optimal weight differs across tasks ($w^\star = 0.1$ for PIQA, 0.6 for ARC-Easy and 0.55 for HellaSwag) and is selected on the evaluation split. The results are averaged over 5 different seeds. In red we report the results for the baseline out of distribution OWT checkpoint. In blue we report the results after the short adaptation run.

P.7 Unconditional molecular generation

The experimental setup is described in Section O.8.

Sampling hyperparameters selection. To select the optimal inference hyperparameters for each method, sampler, and schedule, we swept over:

- **Autoregressive (AR)**: softmax temperature $T \in \{0.0, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$.

Table 9: **Zero-shot accuracy (%)** on language understanding benchmarks. We evaluate SDMs by sampling 5 times with a different random seed, and report the mean$_{\pm\text{std}}$. The best score per column is **bolded** and the second best is underlined. The LLaMA baseline is taken from von Rütte et al. (2025).

Model	PIQA	ARC-Easy	HellaSwag
Autoregressive baselines			
LLaMA	62.7	40.5	33.1
GPT-2	**62.9**	**43.8**	28.9
Discrete diffusion models			
MDM	54.1	31.0	31.1
PGM	58.9	40.4	33.2
GIDD	56.4	31.0	31.9
Ours (Simplex Diffusion)			
$w = 0$	$54.3_{\pm 0.4}$	$36.3_{\pm 0.4}$	$35.6_{\pm 0.2}$
$w = w^\star$	$54.6_{\pm 0.5}$	$43.2_{\pm 0.3}$	$\mathbf{37.9}_{\pm 0.2}$

Table 10: **Selected hyperparameters for Simplex Diffusion (with time conditioning, `SDM_TE`) across noise schedules and sampling steps** (molecular generation; $N \in \{32, 64, 256, 1024\}$) corresponding to Figure 20. T: logit sampling temperature; κ: Simplex churn parameter; p: top-fraction power for α_s^p to select the most confident subset.

		Standard Sampler		Confidence-Based Sampler		
Schedule	**Steps** (N)	**Temperature** (T)	**Churn** (κ)	**Temperature** (T)	**Churn** (κ)	**Power** (p)
Linear	32	0.01	1.0	0.01	1.0	1.0
	64	0.005	1.0	0.05	1.0	1.0
	256	0.05	0.2	0.05	1.0	1.0
	1024	0.001	0.2	0.05	1.0	1.0
Cosine	32	0.03	1.0	0.03	1.0	1.0
	64	0.05	1.0	0.05	1.0	1.0
	256	0.10	0.0	0.05	1.0	1.0
	1024	0.15	1.0	0.08	1.0	1.0
Adaptive	32	0.005	0.0	0.05	1.0	1.0
	64	0.001	0.0	0.05	1.0	1.0
	256	0.005	0.2	0.03	1.0	1.0
	1024	0.03	1.0	0.005	1.0	1.0

- **Discrete Diffusion (MDM & UDM)**: logit temperature $T \in \{0.001, 0.005, 0.01, 0.03, 0.05, 0.08, 0.1, 0.15, 0.2, 0.3, 0.5, 0.7, 0.8, 1.0\}$, top-fraction schedule power $p \in \{0.5, 1.0, 1.5, 2.0, 3.0\}$.
- **Simplex Diffusion (SDM)**: logit temperature $T \in \{0.001, 0.005, 0.01, 0.03, 0.05, 0.08, 0.1, 0.15, 0.2, 0.3, 0.5, 0.7, 0.8, 1.0\}$, churn $\kappa \in \{0.0, 0.2, 1.0\}$, top-fraction schedule power $p \in \{0.5, 1.0, 1.5, 2.0, 3.0\}$.

The optimal sampling hyperparameters corresponding to Figure 20 are reported in Table 10. The optimal sampling hyperparameters corresponding to Figure 21 are reported in Table 11, while those for Figure 22 are reported in Table 12. The optimal sampling hyperparameters corresponding to each row of Tables 14 and 15 with 256 sampling steps, are reported in Table 13 and are selected based on the quality metric.

Sampling budgets and sampling time schedules. We start from an ablation over sampling time schedules for Simplex Diffusion for molecular generation. In Figure 20, we report performance of Simplex Diffusion (with time conditioning) as a function of sampling steps using either standard or confidence-based sampler. We see that depending on the sampling budget, the type of a sampler and the evaluation metric chosen, the choice of a sampling time schedule may differ. For the next

Table 11: **Selected hyperparameters for diffusion methods under standard sampling across steps** ($N \in \{32, 64, 256, 1024\}$) corresponding to Figure 21. Simplex methods use the adaptive schedule; Masked and Uniform use the cosine schedule. T: logit sampling temperature; κ: Simplex churn parameter.

Method	Schedule	Steps (N)	Temperature (T)	Churn (κ)
Masked Diffusion (MDM)	Cosine	32	0.05	—
		64	0.01	—
		256	0.005	—
		1024	0.03	—
Uniform Diffusion (UDM)	Cosine	32	0.05	—
		64	0.001	—
		256	0.01	—
		1024	0.01	—
Uniform (with time conditioning)	Cosine	32	0.005	—
		64	0.001	—
		256	0.05	—
		1024	0.05	—
Simplex Diffusion (SDM)	Adaptive	32	0.08	0.2
		64	0.10	0.2
		256	0.10	1.0
		1024	0.05	1.0
Simplex (with time conditioning)	Adaptive	32	0.005	0.0
		64	0.001	0.0
		256	0.005	0.2
		1024	0.03	1.0

Table 12: **Selected hyperparameters for diffusion methods under confidence-based sampling across steps** (molecular generation; $N \in \{32, 64, 256, 1024\}$) corresponding to Figure 22. Simplex methods use the adaptive schedule; Masked and Uniform diffusion use the cosine schedule. T: logit sampling temperature; κ: Simplex churn parameter; p: top-fraction power for α_s^p to select the most confident subset.

Method	Schedule	Steps (N)	Temperature (T)	Churn (κ)	Power (p)
Masked Diffusion (MDM)	Cosine	32	0.01	—	1.5
		64	0.01	—	1.5
		256	0.01	—	1.0
		1024	0.01	—	2.0
Uniform Diffusion (UDM)	Cosine	32	0.10	—	2.0
		64	0.03	—	1.5
		256	0.03	—	1.0
		1024	0.01	—	0.5
Uniform (with time conditioning)	Cosine	32	0.01	—	1.0
		64	0.01	—	1.0
		256	0.01	—	1.0
		1024	0.01	—	1.0
Simplex Diffusion (SDM)	Adaptive	32	0.05	1.0	1.0
		64	0.01	1.0	1.0
		256	0.05	1.0	1.0
		1024	0.08	1.0	1.0
Simplex (with time conditioning)	Adaptive	32	0.05	1.0	1.0
		64	0.05	1.0	1.0
		256	0.03	1.0	1.0
		1024	0.005	1.0	1.0

ablation, we select adaptive time schedule for Simplex diffusion since it offers overall a good trade-off between different metrics and different sampling budgets.

Table 13: **Optimal sampling hyperparameters for each method and schedule** (molecular generation) in Table 14 (standard sampling) and Table 15 (confidence-based sampling, *conf.*) using 256 sampling steps. T: logit temperature; κ: Simplex churn (Proposition 3.3); p: top-fraction power for α_s^p to select the most confident subset.

Method	Schedule	Standard Sampling Hypers	Confidence (*conf.*) Sampling Hypers
Autoregressive	—	$T = 0.7$	—
Simplex	Linear	$T = 0.03,\ \kappa = 1.0$	$T = 0.05,\ \kappa = 1.0,\ p = 1.0$
Simplex	Cosine	$T = 0.01,\ \kappa = 1.0$	$T = 0.05,\ \kappa = 1.0,\ p = 1.0$
Simplex	Adaptive	$T = 0.10,\ \kappa = 1.0$	$T = 0.05,\ \kappa = 1.0,\ p = 1.0$
Simplex (time cond.)	Linear	$T = 0.05,\ \kappa = 0.2$	$T = 0.05,\ \kappa = 1.0,\ p = 1.0$
Simplex (time cond.)	Cosine	$T = 0.10,\ \kappa = 0.0$	$T = 0.05,\ \kappa = 1.0,\ p = 1.0$
Simplex (time cond.)	Adaptive	$T = 0.005,\ \kappa = 0.2$	$T = 0.03,\ \kappa = 1.0,\ p = 1.0$
Uniform	Linear	$T = 0.01$	$T = 0.03,\ p = 1.5$
Uniform	Cosine	$T = 0.01$	$T = 0.03,\ p = 1.0$
Uniform	Adaptive	$T = 0.10$	$T = 0.03,\ p = 1.0$
Uniform (time cond.)	Linear	$T = 0.01$	$T = 0.01,\ p = 1.0$
Uniform (time cond.)	Cosine	$T = 0.05$	$T = 0.01,\ p = 1.0$
Uniform (time cond.)	Adaptive	$T = 0.08$	$T = 0.01,\ p = 1.0$
Masked	Linear	$T = 0.01$	$T = 0.01,\ p = 1.5$
Masked	Cosine	$T = 0.005$	$T = 0.01,\ p = 1.0$
Masked	Adaptive	$T = 0.001$	$T = 0.01,\ p = 1.0$

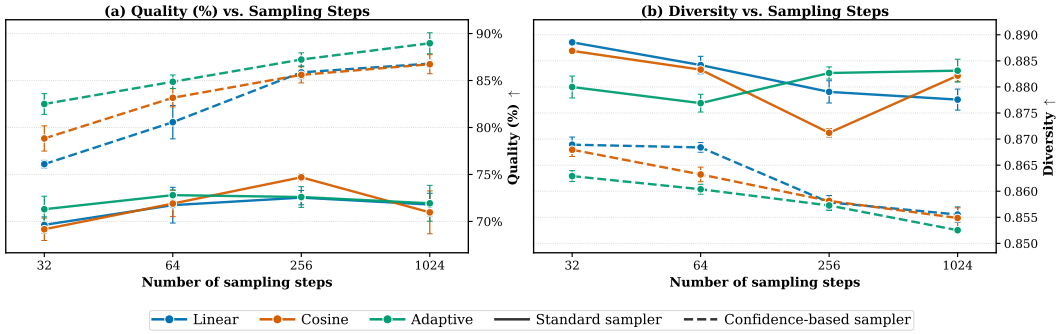

Figure 20: **Sampling-budget and sampling time schedule comparison for time-conditioned SDM.** Quality and Diversity (molecular generation) are shown for standard sampling (solid) and confidence-based sampling (dashed), using linear (blue), cosine (orange), and adaptive (green) inference grids. Increasing the sampling budget improves Quality under confidence-based sampling while reducing Diversity. Standard sampling retains higher Diversity, with smaller and nonmonotonic changes in Quality. Error bars show one standard deviation across three sampling seeds.

Comparison across model variants. Next, we compare performance of Simplex Diffusion to masked and uniform diffusion. For these methods we use cosine time schedule since it led to the best Quality. We report results in Figure 21 for the standard sampler and in Figure 22 for the confidence-based one. We see that both masked and uniform diffusion lead to higher Quality than Simplex Diffusion, while achieving lower Diversity. In case of confidence-based sampler the finding remains except for the fact that Simplex diffusion performs on-par with Uniform diffusion without time conditioning. These results demonstrate that different diffusion methods provide a trade-off between these two metrics.

Quality–diversity trade-offs. We further highlight this point by plotting a Pareto frontier for different methods with confidence-based sampler in Figure 23, where different points represent different sampling seeds, different number of sampling steps, different sampling temperatures, different churns and different top fraction power, see Section O.8 for details. We see that Simplex Diffusion Pareto frontier is situated more towards right and top, though it does not achieve the highest quality,

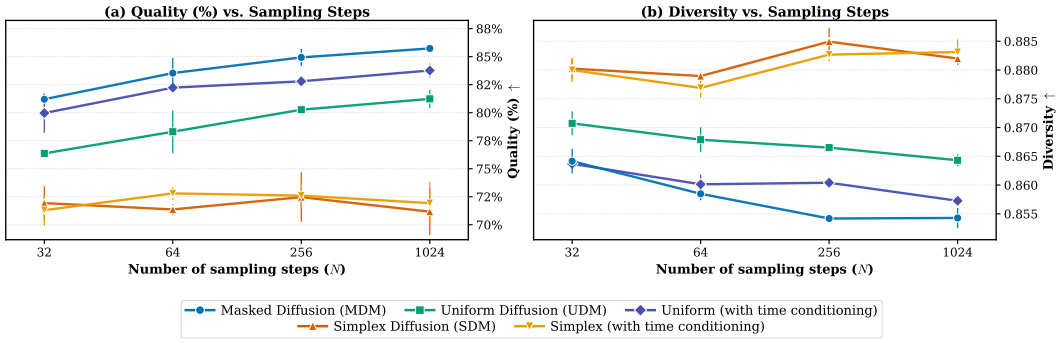

Figure 21: **Quality and Diversity versus sampling steps under standard sampling** (molecular generation). Comparison across all five diffusion methods, using adaptive time schedule for Simplex diffusion and cosine for the rest. Error bars show one standard deviation across three sampling seeds.

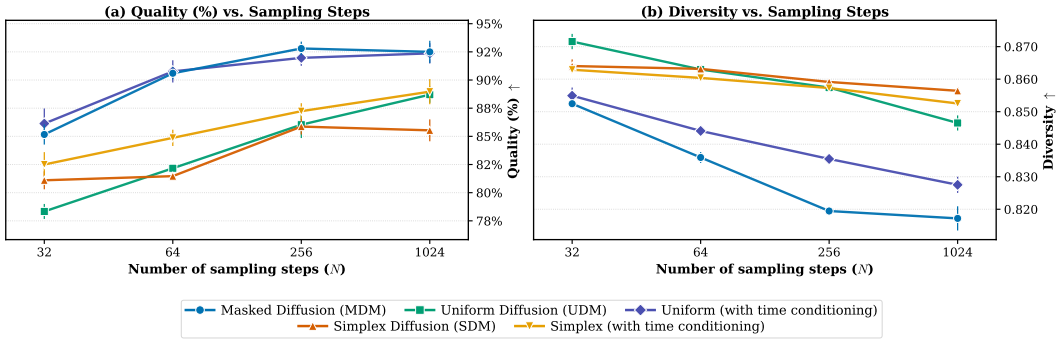

Figure 22: **Quality and Diversity versus sampling steps under confidence-based sampling** (molecular generation). Comparison across all five diffusion methods, using adaptive time schedule for Simplex diffusion and cosine for the rest. Error bars show one standard deviation across three sampling seeds.

compared to Uniform and Masked diffusion. Understanding how to push this frontier further to the top right is a promising future research direction.

Detailed results at 256 steps. Tables 14 and 15 report all four evaluation metrics under standard and confidence-based sampling. With confidence-based sampling and an adaptive grid, time-conditioned SDM achieves 87.23% Quality and 0.857 Diversity. For comparison, time-conditioned UDM with a cosine grid achieves 91.97% and 0.835, while MDM with a cosine grid achieves 92.80% and 0.819. Time conditioning increases the selected confidence-based Quality scores from 85.87% to 87.23% for SDM with an adaptive grid and from 86.03% to 91.97% for UDM with a cosine grid, accompanied by reductions in Diversity. These results highlight the same findings observed in the figures above.

All reported diffusion configurations exceed our AR baseline in Quality. Time-conditioned SDM also achieves higher reported Quality and Diversity than the published GenMol reference, although the training and sampling protocols differ. Overall, the results demonstrate the applicability of SDMs to molecular generation and their competitive performance in higher-diversity regimes, while identifying a remaining gap in maximum Quality relative to masked diffusion. Finding ways to bridge the gap and to push the Pareto frontier for Simplex Diffusion is left for future work.

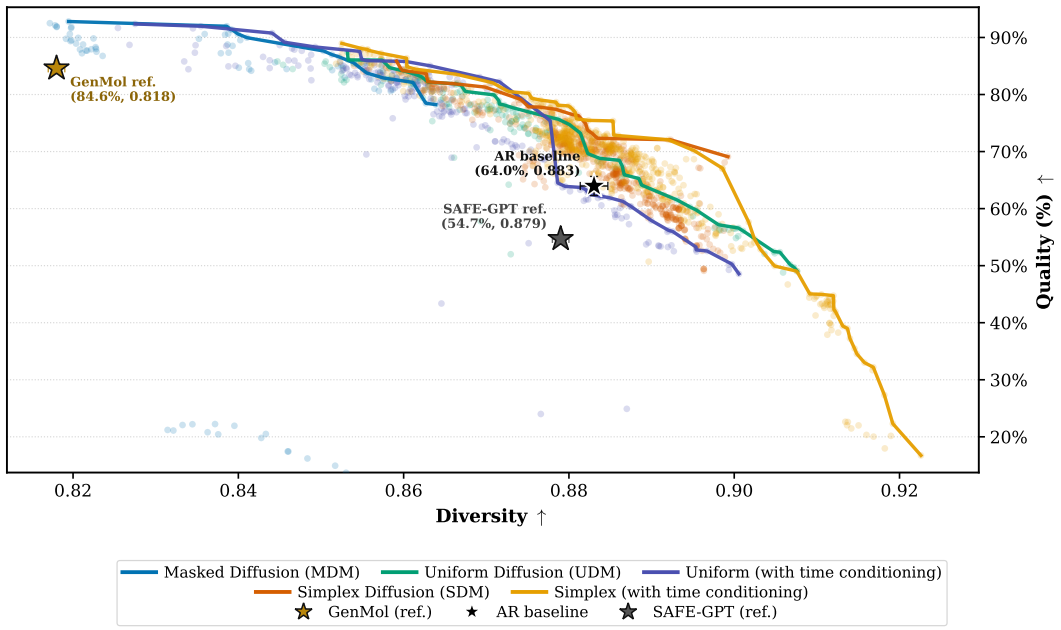

Figure 23: **Empirical Quality–Diversity Pareto frontiers across diffusion methods with confidence based sampling and hyperparameter sweeps.** Molecule Quality (%), ↑, versus Diversity (↑) across all evaluated sampling hyperparameter configurations. All markers denote individual exploration runs sweeping sampling temperatures, number of sampling steps, top-fraction power for selecting the most confident subset and churn parameters, see Section O.8 for more details. Solid curves trace the empirical Pareto frontier of non-dominated configurations for each diffusion family, where we use adaptive time schedule for Simplex diffusion and cosine for the rest. Distinct stars mark external and autoregressive baselines: **GenMol** (gold star), the autoregressive **AR** model (black star), and **SAFE-GPT** (gray star). Simplex Diffusion Pareto frontier leans more towards top right, with many points located in high diversity regime.

Q Additional OWT Samples

Tables 16 to 20 show non-cherry-picked unconditional 1024-token samples from SDMs trained on OWT.

Table 14: **Unconditional *de novo* molecular generation on SAFE-GPT with standard sampling** following the GenMol evaluation protocol (Lee et al., 2025) (1,000 molecules/seed, mean ± std over 3 seeds). **Bold**: best; underline: second best among our runs. All our runs use 256 sampling steps.

Method	Schedule	Quality (%) ↑	Validity (%) ↑	Uniqueness (%) ↑	Diversity ↑
Published Reference (Lee et al., 2025)					
SAFE-GPT	—	54.7 ± 0.3	94.0 ± 0.4	100.0 ± 0	0.879 ± 0.001
GenMol	—	84.6 ± 0.8	100.0 ± 0.0	99.7 ± 0.1	0.818 ± 0.001
Our Run Comparison					
Autoregressive (AR)	—	63.97 ± 1.67	96.37 ± 0.21	99.76 ± 0.06	0.883 ± 0.002
Simplex	Linear	73.57 ± 1.86	99.63 ± 0.12	99.83 ± 0.12	0.879 ± 0.001
Simplex	Cosine	72.33 ± 0.15	99.40 ± 0.40	**99.97 ± 0.06**	0.881 ± 0.000
Simplex	Adaptive	72.47 ± 2.21	99.63 ± 0.15	99.80 ± 0.17	**0.885 ± 0.002**
Simplex (time cond.)	Linear	72.53 ± 0.76	99.00 ± 0.10	99.70 ± 0.30	0.879 ± 0.002
Simplex (time cond.)	Cosine	74.70 ± 0.36	98.90 ± 0.44	97.64 ± 0.76	0.871 ± 0.001
Simplex (time cond.)	Adaptive	72.60 ± 1.10	98.83 ± 0.06	99.73 ± 0.23	0.883 ± 0.001
Uniform	Linear	80.60 ± 1.41	99.27 ± 0.21	98.89 ± 0.44	0.865 ± 0.001
Uniform	Cosine	80.27 ± 0.32	99.30 ± 0.46	98.42 ± 0.72	0.867 ± 0.001
Uniform	Adaptive	75.70 ± 1.87	98.43 ± 0.42	98.58 ± 0.27	0.873 ± 0.003
Uniform (time cond.)	Linear	83.03 ± 1.50	**99.80 ± 0.00**	99.73 ± 0.21	0.859 ± 0.003
Uniform (time cond.)	Cosine	82.80 ± 0.36	99.57 ± 0.06	99.87 ± 0.12	0.860 ± 0.001
Uniform (time cond.)	Adaptive	73.80 ± 0.53	98.37 ± 0.12	98.68 ± 0.54	0.873 ± 0.002
Masked	Linear	84.83 ± 0.68	99.67 ± 0.12	99.90 ± 0.10	0.855 ± 0.002
Masked	Cosine	**84.93 ± 0.78**	99.77 ± 0.32	99.77 ± 0.06	0.854 ± 0.001
Masked	Adaptive	81.17 ± 1.65	99.20 ± 0.00	99.83 ± 0.15	0.863 ± 0.001

Table 15: **Unconditional *de novo* molecular generation on SAFE-GPT with confidence-based (*conf.*) sampling** following the GenMol evaluation protocol (Lee et al., 2025) (1,000 molecules/seed, mean ± std over 3 seeds). **Bold**: best; underline: second best among our runs. All our runs use 256 sampling steps.

Method	Schedule	Quality (%) ↑	Validity (%) ↑	Uniqueness (%) ↑	Diversity ↑
Published Reference (Lee et al., 2025)					
SAFE-GPT	—	54.7 ± 0.3	94.0 ± 0.4	100.0 ± 0	0.879 ± 0.001
GenMol	—	84.6 ± 0.8	100.0 ± 0.0	99.7 ± 0.1	0.818 ± 0.001
Our Run Comparison					
Autoregressive (AR)	—	63.97 ± 1.67	96.37 ± 0.21	99.76 ± 0.06	**0.883 ± 0.002**
Simplex	Linear	83.87 ± 1.36	99.80 ± 0.10	99.87 ± 0.12	0.859 ± 0.000
Simplex	Cosine	83.57 ± 1.59	99.77 ± 0.15	99.87 ± 0.15	0.862 ± 0.001
Simplex	Adaptive	85.87 ± 0.70	99.63 ± 0.31	**99.93 ± 0.06**	0.859 ± 0.001
Simplex (time cond.)	Linear	85.87 ± 0.67	99.63 ± 0.35	99.80 ± 0.10	0.858 ± 0.001
Simplex (time cond.)	Cosine	85.60 ± 0.85	99.67 ± 0.06	99.70 ± 0.20	0.858 ± 0.000
Simplex (time cond.)	Adaptive	87.23 ± 0.71	99.87 ± 0.12	99.77 ± 0.06	0.857 ± 0.001
Uniform	Linear	85.23 ± 0.93	99.37 ± 0.23	98.56 ± 0.59	0.858 ± 0.003
Uniform	Cosine	86.03 ± 1.18	99.63 ± 0.06	97.69 ± 0.27	0.857 ± 0.001
Uniform	Adaptive	83.60 ± 0.26	99.27 ± 0.31	97.01 ± 0.51	0.863 ± 0.001
Uniform (time cond.)	Linear	89.60 ± 0.36	**99.93 ± 0.12**	99.20 ± 0.40	0.846 ± 0.002
Uniform (time cond.)	Cosine	91.97 ± 0.70	99.90 ± 0.10	99.30 ± 0.20	0.835 ± 0.001
Uniform (time cond.)	Adaptive	91.50 ± 1.11	99.87 ± 0.06	99.37 ± 0.15	0.839 ± 0.002
Masked	Linear	**92.80 ± 0.26**	99.90 ± 0.10	99.53 ± 0.38	0.820 ± 0.000
Masked	Cosine	**92.80 ± 0.61**	99.77 ± 0.06	99.83 ± 0.12	0.819 ± 0.001
Masked	Adaptive	92.00 ± 1.05	99.77 ± 0.12	99.83 ± 0.06	0.823 ± 0.001

Table 16: Unconditional 1024-token generation from Simplex Diffusion Model trained on OWT (Sample 1).

Sample 1 (1024 tokens) **GenPPL (GPT-2 Large):** 14.15 — **Unigram Entropy:** 5.27 — **Distinct-2:** 0.724

bring peace and stability Middle East," he told reporters. "We're all going to be hopeful if we're part of an agreement that is retroactively ... And I think the Democratic Party will be up for grabs if she takes it seriously."

Republican strategist Steve Bannon also criticized Trump's comments earlier this week, saying that he had a "waste for American diplomacy" by "trying to burn out sanctions relief with Iran.

New York Times columnist Hugh Hewitt, also a former State Department official and a member of the Republican National Committee, says Trump is trying to forge a deal with Iran without ever breaching international sanctions.

"I have a lot of miscalculations because I think they've succeeded in a bad engagement" Hewitt told The New York Times, adding that the administration needs to understand the terms of the deal.

"I think I don't think this deal will be bad, but it's hard — which is very hard, very hard — to fix it," he added. "What can we do? Can we depolarize the rest of the Middle East?"

--- 10 p.m.

North Korea's government says it backs Republican presidential candidate Donald Trump for setting a new tone with his criticism of North Korea's state-controlled media, in a move at odds with Washington rhetoric over its pursuit of nuclear weapons.

South Korean Foreign Minister Kim Kyung-seo responded in a statement calling Trump "reckless and reckless." He criticizes Trump on North Korea, calling him a "highly dangerous man for the President of the United States."

He also said he would not talk to North Korea without doubting its nuclear weapons abroad. "Without such a strong leadership, any threat from the North Korean regime would be rejected," he said. South Korea has said it banned Pyongyang's nuclear programs in the past because it would stop North Koreans from leaving the country without their weapons.

--- 9:30 p.m.

North Korean military leader Jang Song-thaek says Trump "totally impoundately" the U.S. nuclear weapons program because it was "totally fragile" after the United States dropped an H-bomb on the city of Nagasaki, according to an Air Force One video released Tuesday.

He also said he hoped that the U.S. would make a good return solely to the negotiating table.

He also called for North Korea to get serious about halting its program, although the U.S. has not sanctioned or formally approved its nuclear-weapons program since August 1950. He also urged the United States to learn a lesson from history.

Trump said on Tuesday that the U.S. must getting ready to restart its nuclear program. He said restarting its work with North Korea could mean more years to begin in the coming months.

"It will be very gradual," he said. "And I think that will not be abrupt — and it will will be gradual — until the United States can get back on the table."

--- 8:30 p.m.

Yuri Yuri, South Korea's former deputy prime minister, says Donald Trump believes the U.S. should continue to unravel its nuclear program but is a "preaching and grave mistake."

His remarks followed a recent high-level meeting between Japan and South Korea at an annual summit in Seoul.

Yuri, who is South Korea's first president, has criticized the U.S. should apologize for dropping two atomic bombs on Hiroshima during World War II. He said the U.S. should take steps necessary to try to halt its nuclear program.

--- 8:45 p.m.

Former Republican presidential candidate Donald Trump said on Tuesday that his country's nuclear and missile programs should hold a meeting to discuss "a peaceful solution."

Trump made similar comments about his country's relationship with the United States, which has long labeled its nuclear weapons and other NATO allies are "obsolete."

North Korean leader, Kim Il Jong-un, had earlier this month accused China of cutting off off the country's nuclear program during talks with other parties for peace talks.

Trump's comments have been seen as provocative by some over his tough stance toward China and his controversial unification policy.

But South Korean Foreign Minister Jang Song-thaek issued a statement Tuesday saying the issue was non-negotiable and adding that nuclear talks could not be reached unless both sides reached a peaceful political solution.

----- 7:30 p.m. The United

Table 17: Unconditional 1024-token generation from Simplex Diffusion Model trained on OWT (Sample 2).

Sample 2 (1024 tokens) **GenPPL (GPT-2 Large):** 19.52 — **Unigram Entropy:** 5.59 — **Distinct-2:** 0.804

after it was advertised as a "gender-only" service run by a non-binary woman. The ad came less than two hours after a video featuring a YouTube user called Avoid Open Door Wicked, which featured 12 women, 14 men, 11 men and 10 women was posted online on its website.

"I have been harassed 400+ times because I am surrounded by a hostile environment directed at me," the 23-year-old woman wrote. "I can't believe it," read one of the ads, which read "Divorce is in my veil!" Another added: "You are not a misogynist, and you cannot use your own operating system to escape your own travails."

"If you were a non-binary woman would you be captive for your own sexual desires? If you'd you were a woman woman would you be captive for your desires?" asked ACLU attorney Jennifer Partridge, who filed the case through the Justice Department's nonprofit Philanthropy Law Project.

"It's reflective of what we kind of social network is about and whether it is a real issue," Partridge added.

Courtney ACLU attorneys argued that the company's discrimination against harassment and gender-based bias could violate her legal rights.

"We do not substantially suggest that gender discrimination is not a real issue," she wrote her brief. "This ad does not fit the context of our social Facebook or Google ads." Partridge noted that the company's policy — which requires customers to purchase or sell their products online — suggests more for women than it does for men. The company also sells products in the same retail department stores.

"One could argue that this product based on gender-based disrespect is far more sanctimonious what people might buy from a store online," Partridge said in an email. "This is not an abstract issue."

The company's justification is that its site must engage users to 'disconnect' amongst psychological issues. "I don't think it should," Partridge said. "I don't think another person should be discriminated against."

The company has also said that its decision to remove names or corporate logos from all ads on the site because it doesn't want to deter any potential trespassing. Story continues below advertisement — Hillary Clinton has maintained her commanding lead in the presidential race over Republican presidential nominee Donald Trump.

Clinton says Trump leads her by 39%, according to a new NBC News/Wall Street Journal Journal poll that shows Republican presidential nominee Donald Trump with the largest-ever lead in any presidential election.

The poll, conducted by Public Opinion Strategies, a long-time polling firm, surveyed 1,000 likely voters. It has a margin of error at 3.0 points with the total sample of 1,000.

Clinton gets 43 percent of the vote behind Green Green Party candidate Jill Stein (38 percent), while Trump gets 41 (36 percent) and Jill Stein (35 percent). Under the case for both candidates, Stein would get 33.9 percent of the vote.

The poll predicts Clinton would win over Trump (34 percent), while Stein would get 9.7 percent ahead of Stein (8 percent) Stein/Garrabee (6 percent).

Despite last week's debate announcing her bid for the presidency, Clinton has implied that she would support any future Republican presidential candidate. In November, when she joined Barack Obama's national security team, she said she would need to cast her first female vote in the U.S. Senate to serve as president.

The poll also shows Trump leads all other Republican candidates with 34% support, while Texas Sen. Ted Cruz leads the presumptive GOP nominee with just 15%.

Trump has previously abandoned former Democratic Secretary of State Hillary Clinton after launching an unsuccessful bid to revive his campaign. He himself, however, has endorsed the presumptive GOP nominee, admitting that he still had no intention of securing support. He also suggested that he might be tempted to endorse Clinton if his party didn't back him up.

Story continues below advertisement.

The polls are conducted by landline and automated landline telephone interviews with 1,000 likely voters nationwide and were conducted between Oct. 19 through Nov. 20, 2012, with Barack Obama and Mitt Romney registered among likely voters. The margin of error is plus or minus 3.5 points.

Also on HuffPost: Two new skyscraper plans will bring some of Manhattan's tallest skyline to the rest of the world, after Madison Square Park was knocked out of Manhattan's tallest buildings earlier this year. Construction on the first three-story tower has been delayed in time for a full moon. The city's skyscraper straddled between New York, New York, Chicago and

Table 18: Unconditional 1024-token generation from Simplex Diffusion Model trained on OWT (Sample 3).

Sample 3 (1024 tokens) **GenPPL (GPT-2 Large):** 19.53 — **Unigram Entropy:** 5.23 — **Distinct-2:** 0.645

Related Articles Section 491. Related Articles Section 492 Miscellaneous Related Articles Section 493 Miscellaneous. Section 493. Related Section 494 Miscellaneous Related Articles Section 494 Miscellaneous. Section 493 Miscellaneous Articles Section 494 Miscellaneous Articles Sections 495 Miscellaneous Sections 487 Miscellaneous Sections 488 Miscellaneous Sections 489 24/24 Sections 4811 Miscellaneous Sections 4812 24/24 4813 Miscellaneous Sections 4812 24/24 4813 Miscellaneous Section 4812.

Sec. 501 of Hawaii Revised, Hawaii Revised Code as follows:

(a) It shall be unlawful as a citizen of the United States or a foreign territory of the United States; (a) may conduct, including but not not limited to as a citizen of the United States; (b) as an ex-concitizen of the State of Hawaii or a foreign territory; or (b) as a non-citizen of Hawaii or a former citizen of the United States; (ii) perform other activities as defined in this title. If such an individual does not wish to sign up for annual free membership or holiday free walks in an area outside Hawaii, it shall be unlawful to use internet services, advertisements newspaper articles, or witherbecoming to advertise such activities. Any person who knowingly giving birth to another person under this same title (or any other person under this title or regulations) shall be penalized.

If a person who is denied health care coverage has been notified in advance that a person who does not give birth birth at his birthplace will he become a resident of a foreign territory, or a Territory of another state or vice versa, there is no penalty for violation of this above law. If it also is found that a person who denies giving birth occurs at a site of birth will he is a resident of a foreign territory, or Foreign Territory of the State.

To prohibit any person visiting a foreign territory or from becoming a resident of any territory outside the United States, I am hereby directing that such person knowingly violates the above law. The incurring any violation of the above law or regulations may be taken pursuant to the provisions of the Rawled "Hawaii" Virgin Islands Act of 2018.

Hawaiian and native Hawai'i people: Hawaiian is a Hawaiian language which is a native Hawaiian language that has been traditionally associated with the Hawai'i people. It is nevertheless traditional Hawaiian language spoken by native Hawai'i people. The Native Hawaiian people are descendants of Native Hawaiian people who are descended from Native Hawaiian tribes, and since then they have been interceded by tribes from other Native Hawaiian tribes.

A permanent Hawaiian resident or permanent Hawaiian resident who resides in Hawaii becomes a permanent Hawaiian resident shall continue to form a liaison with all Hawaiian community and volunteer volunteers to assist with navig land being navigated navigated within Hawaii. A permanent Hawaiian resident shall be required surrender reside in Hawaii court for specified time period, and if granted permanent Hawaiian resident order obtain permission from his residence in Hawaii for a specified time period. Permanent Hawaiian residents shall be required to reside before a Hawaii court. A permanent Hawaiian resident shall continue to become legal citizens of Poly Hawaiians because they are native Hawai'i residents who are non-institutionalized employment and who operate their own agricultural enterprises.

Article XV01 of the Civil Rights Act of Samoa: I am proposing to amend section article XV01 of the Civil Rights Act of Samoa and shall be amended by sections VII, II and Article XII, the Constitution of the Republic on January 1, 2016.

Article XV01 U.S. Constitution of Samoa — The Pacific Ocean Rights Act of Samoa Act: This act was performed on January 1, 2016 while facilitating the Civil Rights Act of Samoa. This Act shall be amended in section I, II, sections XVI, and XVI, the Constitution of Samoa as amended by section VI.

The American Samoa Act: This Act shall be amended in section Article XIII, the Pacific Ocean Rights Act as amended by section XXVIII. This Act shall be amended in section Article XIV, the Constitution of the United States as amended by section VI.

Section 18017 U.S. Constitution of the United States of America: Nothing shall be made construed unlawful for any county, state, political subdivision of any country or any other State whatsoever to employ any person in the United States of America. Section 18017 U.S. Constitution of the United States of America includes persons: Native American, African American, Baltic American, Mexican-American Indians non-Indigenous Indian Americans, immigrants, citizens of the United States.

Corrections / Proions: Section 17:20 Sec. 2 — the Constitution 1/3 dated January 1, 2017. Section 17:20 Sec. 3 — the text of the Constitution 1/3, dated January 1, 2017. (Proclamation). Section 17:20 Sec. 3

Table 19: Unconditional 1024-token generation from Simplex Diffusion Model trained on OWT (Sample 4).

Sample 4 (1024 tokens) **GenPPL (GPT-2 Large):** 20.85 — **Unigram Entropy:** 5.60 — **Distinct-2:** 0.848

efficient price for carbon emitted from renewable energy rests with the European Commission, spokesman Richard Ritter said.

Commissions are come under a special agreement, which allows member states to be analysed and then sell all of the emissions emitted from other EU states.

Under this agreement member states would buy emission-free allowances. The member states could then decide to sell them only if they couldn't afford to buy additional items.

For emissions-free under the new agreement, those allowances from Germany would have to give up their market share, said Ritter.

"The Commission will henceforth refuse to use emission-free allowances as an exercise for Germany or for Jean Juniet," Ritter said. "There are other alternatives I would advise that we should be able to bring them into the domestic market, but not at all."

Ritter added that the EU would need to remove emission-free allowances from its domestic market before sending them back to emission-free markets. He added that such plans were approved by the German parliament.

German MEP Bart Schoto told reporters on Sunday that there is no need for a new deal. "It's just a simple legislative process," he said. He suggested emission-free allowances should be sold.

"I don't think so much but but leaving is not really a problem," he said. "If we think we will be able to keep up the cost of our growing economy then I think this pathway will be very difficult."

Other measures must be taken through the new emissions agreement, Ritter said. The measures could include building incentives to build nuclear power plants, which could cost about $1 billion a year plus subsidies for nuclear power plants in South Florida, which could raise fuel costs.

"We will have to expand our capacity if we can't get our policies back into place," he said. "It's going to be a very tough decision and it will be taken out as soon as possible," he said.

Merkel's action 'unemptive': Juncker, who was arrested in Brussels in connection with his meeting between the two leaders and German Chancellor Angela Merkel, met privately with former Chancellor Gerhard Schroeder, the former Bavarian prime minister and medical doctor Reinforce Eisenberg at a press conference in Berlin last week.

The commission's decision has been fraught with political tensions. However, Germany has appealed to Brussels against a form of one-party rule imposed by the European Union.

On Friday, Germany's state energy company (SDF) and its state utility EDF — which has been the main target for calls for divesting coal, said it would not move forward. Eisenberg said it was highly unlikely that Merkel would make any effort to change her position.

"This is a step in a right direction," said Marguerte Schweiko, the head of the German daily newspaper Vorb Mauschaftung in an interview with Der Spiegel.

"The chancellor has clearly changed position... She has already shown herself capable of exerting power without the popular support of her friends, who control the world's most valuable fossil fuel reserves."

"Merkel is an anti-nuclear campaigner... She has made it clear that she did not actually participate in any concrete action."

Schweiko also said Merkel her actions are "unemptive", adding that "the German government played a principal role in cultivation of tools for climate change". She added that Merkel had pressure on many Germans. "Mrs Merkel seems to have lost her credibility in many ways she cannot stand trusted," she said.

Earlier this week, the German Green Party, Christian Saar, issued a statement condemning "polarly polarized" climate negotiations. "It is inconceivable that all parties can agree on a legally binding treaty—even within the framework of the parliament—but it documents highly polarised," the statement read.

The EU has started undertaking its own voluntary emissions trading scheme, making matters more pressing among some European countries that are realistic about climate change, as well as other countries that have warned against taking climate action.

Germany may have already signed up to reduce carbon emissions, meaning that still does not have the resources to do so. But it may also have an opportunity to kick-start talks in Paris on climate change.

Later this year, Europe and the major United States are expected to host talks on agriculture and climate change. Their involvement The Paris talks is scheduled to begin at the end of 2012, months after the Kyoto Protocol took place in Copenhagen in December 2009.

British Prime Minister David Cameron, who is in Copenhagen, and is due to discuss negotiations on climate change, had told reporters the Paris talks would "have no end in sight yet". Cameron told reporters earlier this week that Britain was "not ruling out of Paris", nor was an organiser of the negotiations. G20 leaders will hold a summit in Hamburg later next month

Table 20: Unconditional 1024-token generation from Simplex Diffusion Model trained on OWT (Sample 5).

Sample 5 (1024 tokens) **GenPPL (GPT-2 Large):** 4.04 — **Unigram Entropy:** 3.56 — **Distinct-2:** 0.197

Commandments. 300. 325 325. The Son of God, His Son, His Father and the Commandments.

300. 325 325. The Son of God, The Son of the Holy and Father and the Commandments.

300. 325 325 Elijah Elijah is not commanded by The Lord of Christ, His Father and the Commandments.

300. 325 325 Elijah is not commanded by Jesus Christ, His Father, the Commandments.

300. 325 325. Of The Lord of Christ, His Father and the Commandments. 300. 325. The Lord of Christ, His Son and the Commandments. 300. 325 325. The Lord of Christ, His Father and the Commandments.

301. 325 The Lord of Christ, His Father and the Commandments. 301. 325. The Lord of Christ, His Father, the Commandments. 301. 325 325. Of The Lord's son, His Father, the Commandments.

301. the Lord Elijah is not commanded by The Lord of son, His Father, the Commandments. 301. the Lord Elijah is not commanded by The Lord's son, Father, the Commandments.

301. the Lord Elijah is not being commanded by the Lord's Christ, His Son, His Father and the Commandments. 301. the Lord Elijah is not being commanded by The Lord's son, His Father and the Commandments.

302. 304 308 308. The Son of God to His Father and the Commandment 351. 303. 308 308 The Son of God, Maker Commandments.

304. 328 309. Jesus God to His Father and the Commandment 351. 303. 328 309. Jesus Lord to be Father, Maker Commandments. 303. 328 309. Jesus Lord to be Stronger.

305. 328 309. The Son of God. Maker Commandment. 306. 328 309 309. The Son of Christ, His Father, the Commandments. 305. 328 309. The Lord to be Strong, Maker Commandments.

307. 328. The Son of Christ, His Father, the Commandments. 308. 328 328. The Son of Christ, His Father, the Commandments. 309. 328 328. The Lord to be Stronger.

322. 329 330. The Son of Christ, His Father, the Commandments. 323. 329 330. Jesus The Son of God, Maker Command. 324. 329 330. Jesus Lord to be Strong. 323. 329 330. The Son of God, Maker Commandments.

324. 329 331 Lose! Of The Son's God, His Father and the Commandments. 326. 329 331. The Son of Christ, His Father and the Commandments.

325. 320 315 315. Jesus The Son of God, His Exalted Father and the Commandments. 326. 320 320. Jesus the Son of Christ, His Father and the Commandments.

325 320 320. Jesus Christ to the Father, the Commandment 351. 325 320 320. The Son of Christ to the Father and the Commandment 351. 327. 320 320. The Son of Christ to the Father and the Commandments.

349. 321 321. The Son of God, Maker Command. 350. 321 321. Jesus Christ to He Father, Maker Commandment. 351. 321 321. Jesus the Son of God to His Father, the Commandment 351.

352 321 321. The Son of Christ to His Father, the Commandment 351. 352. 321 321. The Son's God, His Exalted Father and the Commandment 351. 353. 321 321. The Son of Christ to His Father, the Commandment 351.

352. 412 355. The Son of Christ to His Father, the Commandment 351. 352. 412 355. The Son of God, Maker Commandment. 353. 412 355. The Lord to be Strongerful. 354. 412 355. The Lord of his Father, Maker Commandments.

355. 412 412. Jesus The Son of Christ, the Father and the Commandments. 355. 412 412. The Son of Christ, the Father and the Commandments. 356. 412 412. Jesus Christ to the Father, Maker Commandments.

357. 412 410. The Son of Christ, His God, the Commandment 351. 356 412 412 410. Jesus Christ to the Father, Maker Commandments. 357. 412 410. The Son of God, the Father and the Commandments.

358. 412 412 410. The Lord's God, the Father and Commanders. 358. 412 410. The Son of God, Maker Commandments. 357. 415 414. Jesus Christ to the Father, the Commandments. 357. 412 414. The Son of Christ, the Father and the Commandments. 358. 412 415. It Happen to

R FULL RESULT TABLES

R.1 SUDOKU

Table 21: **Accuracy (%) on Sudoku** in 180 steps for **Simplex Diffusion Models** with Churn $\kappa = 0.0$ across sampling schedules. Values are reported as mean$_{\pm\text{std}}$ over 5 seeds. Column-wise top scores are **bolded**, and overall best for $\kappa = 0.0$ is **underlined**

Concentration Schedule	Linear	Cosine	Adaptive
Uniform time sampler			
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.5, \ell = 0.2$)	$62.8_{\pm 5.0}$	$62.4_{\pm 4.7}$	–
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.75, \ell = 0.2$)	$74.0_{\pm 4.4}$	$73.7_{\pm 4.3}$	–
Const.-Lin. ($\nu_0 = 0.4, \nu_1 = 0.75, \ell = 0.8$)	$73.9_{\pm 4.7}$	$74.0_{\pm 4.7}$	–
Constant ($\nu = 0.5$)	$64.9_{\pm 3.7}$	$64.6_{\pm 3.1}$	–
Constant ($\nu = 0.25$)	$56.0_{\pm 6.2}$	$55.6_{\pm 5.8}$	–
Adaptive time sampler			
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.5, \ell = 0.2$)	$78.5_{\pm 2.1}$	$78.1_{\pm 2.3}$	$75.3_{\pm 2.5}$
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.75, \ell = 0.2$)	$79.4_{\pm 2.6}$	$79.2_{\pm 2.7}$	$77.2_{\pm 3.1}$
Const.-Lin. ($\nu_0 = 0.4, \nu_1 = 0.75, \ell = 0.8$)	$77.1_{\pm 2.5}$	$76.9_{\pm 2.4}$	$75.1_{\pm 2.8}$
Constant ($\nu = 0.5$)	$78.0_{\pm 2.0}$	$77.9_{\pm 2.1}$	$74.2_{\pm 2.0}$
Constant ($\nu = 0.25$)	$76.0_{\pm 3.5}$	$75.5_{\pm 3.6}$	$72.4_{\pm 4.2}$
Self-conditioning + Uniform time sampler			
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.5, \ell = 0.2$)	$97.4_{\pm 2.3}$	$94.6_{\pm 6.6}$	–
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.75, \ell = 0.2$)	$96.9_{\pm 3.1}$	$94.0_{\pm 9.1}$	–
Const.-Lin. ($\nu_0 = 0.4, \nu_1 = 0.75, \ell = 0.8$)	**$\underline{99.2}$**$_{\pm 0.2}$	**99.1**$_{\pm 0.2}$	–
Constant ($\nu = 0.5$)	$98.7_{\pm 0.6}$	$98.1_{\pm 1.3}$	–
Constant ($\nu = 0.25$)	$97.6_{\pm 1.7}$	$96.5_{\pm 3.4}$	–
Self-conditioning + adaptive time sampler			
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.5, \ell = 0.2$)	$91.6_{\pm 17.3}$	$88.5_{\pm 24.3}$	$89.7_{\pm 21.7}$
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.75, \ell = 0.2$)	$94.0_{\pm 10.1}$	$85.4_{\pm 24.8}$	$89.5_{\pm 17.9}$
Const.-Lin. ($\nu_0 = 0.4, \nu_1 = 0.75, \ell = 0.8$)	$97.3_{\pm 4.5}$	$95.2_{\pm 9.1}$	$96.6_{\pm 4.9}$
Constant ($\nu = 0.5$)	$99.1_{\pm 0.2}$	$98.9_{\pm 0.5}$	**98.9**$_{\pm 0.3}$
Constant ($\nu = 0.25$)	$98.1_{\pm 2.9}$	$97.1_{\pm 5.0}$	$97.5_{\pm 4.1}$

Table 22: **Accuracy (%) on Sudoku** in 180 steps for **Simplex Diffusion Models** with Churn $\kappa = 0.2$ across sampling schedules. Values are reported as mean$_{\pm\text{std}}$ over 5 seeds. Column-wise top scores are **bolded**, and overall best for $\kappa = 0.2$ is **underlined**

Concentration Schedule	Linear	Cosine	Adaptive
Uniform time sampler			
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.5, \ell = 0.2$)	$81.1_{\pm 3.2}$	$81.4_{\pm 3.3}$	–
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.75, \ell = 0.2$)	$85.3_{\pm 3.2}$	$85.4_{\pm 3.3}$	–
Const.-Lin. ($\nu_0 = 0.4, \nu_1 = 0.75, \ell = 0.8$)	$80.4_{\pm 3.7}$	$79.1_{\pm 3.5}$	–
Constant ($\nu = 0.5$)	$83.5_{\pm 2.0}$	$80.0_{\pm 1.8}$	–
Constant ($\nu = 0.25$)	$76.6_{\pm 3.5}$	$77.6_{\pm 3.5}$	–
Adaptive time sampler			
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.5, \ell = 0.2$)	$88.3_{\pm 1.4}$	$88.0_{\pm 1.3}$	$85.7_{\pm 1.6}$
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.75, \ell = 0.2$)	$87.9_{\pm 1.6}$	$87.6_{\pm 1.7}$	$85.1_{\pm 1.9}$
Const.-Lin. ($\nu_0 = 0.4, \nu_1 = 0.75, \ell = 0.8$)	$85.2_{\pm 1.8}$	$82.9_{\pm 1.8}$	$82.1_{\pm 2.1}$
Constant ($\nu = 0.5$)	$87.5_{\pm 2.0}$	$83.6_{\pm 1.6}$	$84.0_{\pm 3.2}$
Constant ($\nu = 0.25$)	$86.3_{\pm 2.2}$	$86.1_{\pm 1.8}$	$84.4_{\pm 2.4}$
Self-conditioning + Uniform time sampler			
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.5, \ell = 0.2$)	$98.3_{\pm 1.4}$	$95.8_{\pm 4.5}$	–
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.75, \ell = 0.2$)	$97.3_{\pm 2.4}$	$95.8_{\pm 4.8}$	–
Const.-Lin. ($\nu_0 = 0.4, \nu_1 = 0.75, \ell = 0.8$)	**$\underline{98.6}$**$_{\pm 0.2}$	$95.8_{\pm 0.6}$	–
Constant ($\nu = 0.5$)	$97.8_{\pm 0.5}$	$92.7_{\pm 1.1}$	–
Constant ($\nu = 0.25$)	$98.5_{\pm 0.6}$	**97.5**$_{\pm 1.4}$	–
Self-conditioning + adaptive time sampler			
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.5, \ell = 0.2$)	$89.8_{\pm 21.6}$	$86.6_{\pm 28.2}$	$86.9_{\pm 28.1}$
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.75, \ell = 0.2$)	$92.7_{\pm 13.8}$	$87.7_{\pm 22.3}$	$88.5_{\pm 21.9}$
Const.-Lin. ($\nu_0 = 0.4, \nu_1 = 0.75, \ell = 0.8$)	$96.7_{\pm 4.3}$	$92.0_{\pm 8.3}$	$95.0_{\pm 5.9}$
Constant ($\nu = 0.5$)	$98.2_{\pm 0.2}$	$93.4_{\pm 0.3}$	$97.1_{\pm 2.0}$
Constant ($\nu = 0.25$)	$98.4_{\pm 2.1}$	$97.0_{\pm 3.6}$	**97.2**$_{\pm 4.3}$

Table 23: **Accuracy (%) on Sudoku** in 180 steps for **Simplex Diffusion Models** with Churn $\kappa = 1.0$ across sampling schedules. Values are reported as mean$_{\pm\text{std}}$ over 5 seeds. Column-wise top scores are **bolded**, and overall best for $\kappa = 1.0$ is **underlined**

Concentration Schedule	Linear	Cosine	Adaptive
Uniform time sampler			
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.5, \ell = 0.2$)	$89.3_{\pm 2.3}$	$89.7_{\pm 2.3}$	–
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.75, \ell = 0.2$)	$91.5_{\pm 1.9}$	$91.6_{\pm 1.6}$	–
Const.-Lin. ($\nu_0 = 0.4, \nu_1 = 0.75, \ell = 0.8$)	$88.4_{\pm 2.2}$	$88.9_{\pm 2.4}$	–
Constant ($\nu = 0.5$)	$90.5_{\pm 1.1}$	$90.6_{\pm 1.2}$	–
Constant ($\nu = 0.25$)	$86.9_{\pm 2.1}$	$87.8_{\pm 2.5}$	–
Adaptive time sampler			
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.5, \ell = 0.2$)	$90.8_{\pm 0.9}$	$90.8_{\pm 1.0}$	$89.4_{\pm 1.0}$
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.75, \ell = 0.2$)	$90.5_{\pm 1.1}$	$90.3_{\pm 1.3}$	$88.8_{\pm 1.5}$
Const.-Lin. ($\nu_0 = 0.4, \nu_1 = 0.75, \ell = 0.8$)	$88.3_{\pm 1.3}$	$88.5_{\pm 1.2}$	$86.5_{\pm 1.4}$
Constant ($\nu = 0.5$)	$90.6_{\pm 1.5}$	$90.4_{\pm 1.5}$	$88.8_{\pm 1.8}$
Constant ($\nu = 0.25$)	$89.6_{\pm 1.8}$	$89.8_{\pm 1.8}$	$88.3_{\pm 1.9}$
Self-conditioning + Uniform time sampler			
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.5, \ell = 0.2$)	$98.9_{\pm 0.8}$	$97.2_{\pm 2.8}$	–
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.75, \ell = 0.2$)	$98.3_{\pm 1.1}$	$96.9_{\pm 3.4}$	–
Const.-Lin. ($\nu_0 = 0.4, \nu_1 = 0.75, \ell = 0.8$)	**$\underline{99.1}$**$_{\pm 0.2}$	**98.7**$_{\pm 0.3}$	–
Constant ($\nu = 0.5$)	$98.5_{\pm 0.6}$	$97.5_{\pm 1.1}$	–
Constant ($\nu = 0.25$)	**$\underline{99.1}$**$_{\pm 0.3}$	$98.5_{\pm 0.8}$	–
Self-conditioning + adaptive time sampler			
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.5, \ell = 0.2$)	$91.2_{\pm 18.8}$	$87.5_{\pm 26.6}$	$88.6_{\pm 24.4}$
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.75, \ell = 0.2$)	$93.9_{\pm 11.9}$	$89.1_{\pm 20.7}$	$90.3_{\pm 18.7}$
Const.-Lin. ($\nu_0 = 0.4, \nu_1 = 0.75, \ell = 0.8$)	$97.5_{\pm 3.9}$	$94.8_{\pm 8.9}$	$96.3_{\pm 5.4}$
Constant ($\nu = 0.5$)	$98.8_{\pm 0.2}$	$98.2_{\pm 0.2}$	**98.3**$_{\pm 0.2}$
Constant ($\nu = 0.25$)	$98.8_{\pm 1.5}$	$97.8_{\pm 3.3}$	$98.1_{\pm 2.9}$

Table 24: **Baseline Models Accuracy (%) on Sudoku** in 180 steps across sampling schedules for the Appendix. We evaluate Autoregressive (AR), Discrete Absorbing (MDM with CE and ELBO losses), Discrete Uniform (UDM), Predictor-Corrector (PC), Self-Conditioning (SC / loopholing), Flow Matching over one-hot encoding (FLM with uniform, Gauss-Hermite LUT, and adaptive time samplers), and Spherical Flow Matching ($\mathbb{S}$-FLM). †trained with uniform time sampler, ‡trained with adaptive time sampler. Values are reported as mean$_{\pm\text{std}}$ over 5 seeds. We **bold** the best result and underline the second best in each column.

Model	Linear	Cosine	Adaptive
Autoregressive			
AR (Sample)	$2.9_{\pm 0.6}$	–	–
AR (Greedy)	$3.1_{\pm 0.5}$	–	–
Discrete (Absorbing)			
Ancestral† (CE)	$60.1_{\pm 3.8}$	$61.1_{\pm 4.5}$	–
Ancestral‡ (CE)	$69.4_{\pm 4.8}$	$69.7_{\pm 4.0}$	$70.0_{\pm 4.4}$
Ancestral† (ELBO)	$56.0_{\pm 5.4}$	$55.8_{\pm 5.8}$	–
Ancestral‡ (ELBO)	$71.5_{\pm 2.2}$	$71.7_{\pm 1.6}$	$71.8_{\pm 2.4}$
Predictor-Corrector†	$85.3_{\pm 1.2}$	$84.6_{\pm 1.0}$	–
Predictor-Corrector‡	$33.8_{\pm 41.4}$	$35.6_{\pm 40.7}$	$69.6_{\pm 31.6}$
Ancestral + SC† (CE)	$91.2_{\pm 0.7}$	$98.3_{\pm 0.5}$	–
Ancestral + SC‡ (CE)	$94.7_{\pm 0.8}$	$\mathbf{99.3}_{\pm 0.2}$	$\underline{91.6}_{\pm 1.0}$
Ancestral + SC† (ELBO)	$90.5_{\pm 0.8}$	$98.1_{\pm 0.8}$	–
Ancestral + SC‡ (ELBO)	$94.3_{\pm 0.7}$	$\underline{99.2}_{\pm 0.2}$	$90.9_{\pm 1.0}$
Discrete (Uniform)			
Ancestral† (ELBO)	$73.7_{\pm 3.4}$	$73.6_{\pm 3.2}$	–
Ancestral‡ (ELBO)	$82.1_{\pm 1.7}$	$82.4_{\pm 1.7}$	$81.5_{\pm 1.6}$
Predictor-Corrector†	$\underline{95.8}_{\pm 1.0}$	$95.2_{\pm 1.3}$	–
Predictor-Corrector‡	$92.9_{\pm 0.9}$	$92.3_{\pm 0.9}$	–
Ancestral + SC† (ELBO)	$\mathbf{97.8}_{\pm 1.7}$	$98.1_{\pm 1.2}$	–
Ancestral + SC‡ (ELBO)	$94.9_{\pm 5.1}$	$95.5_{\pm 4.8}$	$\mathbf{93.9}_{\pm 5.5}$
Euclidean / Spherical Flows			
FLM† (Uniform)	$73.4_{\pm 5.2}$	$73.0_{\pm 5.1}$	–
FLM (Gauss-Hermite LUT)	$67.7_{\pm 3.4}$	$67.4_{\pm 3.2}$	–
FLM‡ (Adaptive)	$75.3_{\pm 3.4}$	$74.9_{\pm 3.5}$	$69.2_{\pm 4.2}$
$\mathbb{S}$-FLM†	$83.5_{\pm 2.2}$	$84.0_{\pm 1.8}$	–
$\mathbb{S}$-FLM‡	$87.3_{\pm 1.5}$	$87.3_{\pm 1.7}$	$84.5_{\pm 1.9}$

R.2 TINYGSM

Table 25: **Baseline Models Accuracy (%) on TinyGSM** ($T = 1.0$, 512 steps) across sampling schedules. We evaluate Autoregressive (AR), Discrete Absorbing (CE and ELBO losses), Discrete Uniform (ELBO loss), Predictor-Corrector, Self-Conditioning (Ancestral + SC), FLM, and $\mathbb{S}$-FLM. †trained with the uniform time sampler, ‡trained with the adaptive time sampler. We **bold** the best result in each column and underline the best result within each section (both if achieving both).

Model	Linear	Cosine	Adaptive
Autoregressive			
AR (Sample)	52.6	–	–
AR (Greedy)	**62.6**	–	–
Discrete (Absorbing)			
Ancestral† (CE)	12.9	14.8	–
Ancestral‡ (CE)	13.2	13.4	13.1
Ancestral† (ELBO)	15.8	14.2	–
Ancestral‡ (ELBO)	14.0	13.9	13.9
Predictor-Corrector†	13.3	12.1	–
Predictor-Corrector‡	10.8	11.4	10.9
Ancestral + SC† (CE)	23.0	26.4	–
Ancestral + SC‡ (CE)	22.1	23.2	21.9
Ancestral + SC† (ELBO)	20.5	22.2	–
Ancestral + SC‡ (ELBO)	23.5	22.2	21.1
Discrete (Uniform)			
Ancestral† (ELBO)	17.7	17.0	–
Ancestral‡ (ELBO)	16.7	16.2	15.6
Predictor-Corrector†	36.8	**36.5**	–
Predictor-Corrector‡	32.4	30.9	**26.6**
Ancestral + SC† (ELBO)	15.6	15.2	–
Ancestral + SC‡ (ELBO)	19.2	19.0	18.6
Euclidean / Spherical Flows			
FLM (Gauss-Hermite LUT)	0.6	0.5	–
FLM†	3.0	2.9	–
FLM‡	4.0	3.9	3.6
$\mathbb{S}$-FLM‡	10.7	11.5	10.3
$\mathbb{S}$-FLM‡ (argmax velocity)	15.0	16.5	13.2

Table 26: **Baseline Models Accuracy (%) on TinyGSM** ($T = 1.0$, 64 steps) across sampling schedules. We evaluate Autoregressive (AR), Discrete Absorbing (CE and ELBO losses), Discrete Uniform (ELBO loss), Predictor-Corrector, Self-Conditioning (Ancestral + SC), FLM, and $\mathbb{S}$-FLM. †trained with the uniform time sampler, ‡trained with the adaptive time sampler. We **bold** the best result in each column and underline the best result within each section (both if achieving both).

Model	Linear	Cosine	Adaptive
Discrete (Absorbing)			
Ancestral† (CE)	9.7	11.1	–
Ancestral‡ (CE)	8.3	11.7	11.2
Ancestral† (ELBO)	10.0	13.3	–
Ancestral‡ (ELBO)	8.5	11.8	12.6
Predictor-Corrector†	10.7	10.2	–
Predictor-Corrector‡	9.8	11.4	7.2
Ancestral + SC† (CE)	12.2	19.9	–
Ancestral + SC‡ (CE)	12.6	19.6	18.0
Ancestral + SC† (ELBO)	12.6	17.2	–
Ancestral + SC‡ (ELBO)	12.9	19.3	**18.7**
Discrete (Uniform)			
Ancestral† (ELBO)	14.4	14.5	–
Ancestral‡ (ELBO)	14.9	15.6	14.6
Predictor-Corrector†	**23.3**	**23.7**	–
Predictor-Corrector‡	20.5	20.1	16.6
Ancestral + SC† (ELBO)	12.0	14.7	–
Ancestral + SC‡ (ELBO)	17.0	18.2	16.7
Euclidean / Spherical Flows			
FLM (Gauss-Hermite LUT)	0.5	0.6	–
FLM†	2.5	2.5	–
FLM‡	3.3	3.5	4.0
$\mathbb{S}$-FLM‡	4.5	9.7	9.5
$\mathbb{S}$-FLM‡ (argmax velocity)	9.1	14.2	12.9

Table 27: **Baseline Models Accuracy (%) on TinyGSM** ($T = 0.1$, 512 steps) across sampling schedules. We evaluate Autoregressive (AR), Discrete Absorbing (CE and ELBO losses), Discrete Uniform (ELBO loss), Predictor-Corrector, Self-Conditioning (Ancestral + SC), FLM, and $\mathbb{S}$-FLM. †trained with the uniform time sampler, ‡trained with the adaptive time sampler. We **bold** the best result in each column and underline the best result within each section (both if achieving both). With argmax velocity, the $\mathbb{S}$-FLM sampler is deterministic given the initial noise and does not depend on the temperature, hence identical rows at $T = 1$ and $T = 0.1$.

Model	Linear	Cosine	Adaptive
Autoregressive			
AR (Sample)	52.6	–	–
AR (Greedy)	**62.6**	–	–
Discrete (Absorbing)			
Ancestral† (CE)	31.0	31.1	–
Ancestral‡ (CE)	30.4	30.0	30.3
Ancestral† (ELBO)	32.0	32.0	–
Ancestral‡ (ELBO)	32.3	33.5	32.0
Predictor-Corrector†	39.6	37.7	–
Predictor-Corrector‡	39.8	39.2	38.9
Ancestral + SC† (CE)	44.0	42.7	–
Ancestral + SC‡ (CE)	45.8	**45.5**	**42.7**
Ancestral + SC† (ELBO)	39.0	38.0	–
Ancestral + SC‡ (ELBO)	43.2	43.5	42.6
Discrete (Uniform)			
Ancestral† (ELBO)	33.0	32.6	–
Ancestral‡ (ELBO)	34.0	31.2	34.6
Predictor-Corrector†	45.5	**45.5**	–
Predictor-Corrector‡	40.1	39.8	40.1
Ancestral + SC† (ELBO)	31.7	30.3	–
Ancestral + SC‡ (ELBO)	39.7	38.3	38.8
Euclidean / Spherical Flows			
FLM (Gauss-Hermite LUT)	0.8	0.8	–
FLM†	6.4	6.4	–
FLM‡	8.3	8.3	8.3
$\mathbb{S}$-FLM‡	15.4	15.4	14.7
$\mathbb{S}$-FLM‡ (argmax velocity)	15.0	16.5	13.2

Table 28: **Baseline Models Accuracy (%) on TinyGSM** ($T = 0.1$, 64 steps) across sampling schedules. We evaluate Autoregressive (AR), Discrete Absorbing (CE and ELBO losses), Discrete Uniform (ELBO loss), Predictor-Corrector, Self-Conditioning (Ancestral + SC), FLM, and $\mathbb{S}$-FLM. †trained with the uniform time sampler, ‡trained with the adaptive time sampler. We **bold** the best result in each column and underline the best result within each section (both if achieving both). With argmax velocity, the $\mathbb{S}$-FLM sampler is deterministic given the initial noise and does not depend on the temperature, hence identical rows at $T = 1$ and $T = 0.1$.

Model	Linear	Cosine	Adaptive
Discrete (Absorbing)			
Ancestral† (CE)	25.0	28.2	–
Ancestral‡ (CE)	24.9	29.3	27.3
Ancestral† (ELBO)	27.4	29.5	–
Ancestral‡ (ELBO)	29.6	30.7	29.9
Predictor-Corrector†	31.1	31.8	–
Predictor-Corrector‡	33.0	35.2	32.9
Ancestral + SC† (CE)	31.5	38.7	–
Ancestral + SC‡ (CE)	35.9	**40.3**	**41.4**
Ancestral + SC† (ELBO)	29.8	34.5	–
Ancestral + SC‡ (ELBO)	35.6	39.8	41.1
Discrete (Uniform)			
Ancestral† (ELBO)	30.7	31.5	–
Ancestral‡ (ELBO)	32.3	32.4	31.6
Predictor-Corrector†	**38.3**	37.0	–
Predictor-Corrector‡	35.7	35.2	33.5
Ancestral + SC† (ELBO)	27.8	30.7	–
Ancestral + SC‡ (ELBO)	34.7	37.5	37.1
Euclidean / Spherical Flows			
FLM (Gauss-Hermite LUT)	0.7	0.8	–
FLM†	6.6	6.4	–
FLM‡	8.1	8.3	7.9
$\mathbb{S}$-FLM‡	9.4	14.1	13.4
$\mathbb{S}$-FLM‡ (argmax velocity)	9.1	14.2	12.9

Table 29: **TinyGSM Simplex Diffusion (SDM) Accuracy (%)** ($T = 1.0$, 512 steps) across noise concentration schedules, sampling schedules (Linear, Cosine, Adaptive), and churn ($\kappa \in \{0.0, 0.2, 1.0\}$). We compare Expectation (P_t), Argmax, and Argmax with Self-Conditioning (+ SC). All SDMs are trained with the adaptive time sampler. We **bold** the best result in each column and underline the best result within each section (both if achieving both).

Concentration Schedule	Linear			Cosine			Adaptive		
	$\kappa = 0$	$\kappa = 0.2$	$\kappa = 1.0$	$\kappa = 0$	$\kappa = 0.2$	$\kappa = 1.0$	$\kappa = 0$	$\kappa = 0.2$	$\kappa = 1.0$
Expectation Input Processing (P_t)									
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.5, \ell = 0.2$)	15.4	25.1	34.7	**19.8**	29.1	38.8	12.6	**35.7**	**45.8**
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.75, \ell = 0.2$)	15.1	24.5	31.7	17.9	28.4	36.8	11.5	29.2	38.8
Constant ($\nu = 0.5$)	15.7	23.1	33.6	17.5	24.4	38.2	15.0	29.1	41.7
Constant ($\nu = 0.25$)	12.1	19.4	26.6	14.2	25.8	32.5	10.2	33.9	43.3
Constant ($\nu = 0.1$)	4.7	9.6	12.8	6.5	15.6	20.1	3.5	17.0	24.5
Argmax Input Processing									
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.5, \ell = 0.2$)	15.5	26.7	35.7	17.4	26.9	33.3	14.9	21.9	30.4
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.75, \ell = 0.2$)	18.3	26.3	34.2	18.4	26.0	37.4	16.0	20.2	30.6
Constant ($\nu = 0.5$)	17.1	24.8	33.9	16.6	22.4	33.0	16.7	16.4	29.5
Constant ($\nu = 0.25$)	19.3	29.3	31.8	17.8	27.3	35.0	17.8	23.0	30.3
Argmax + Self-Conditioning (SC)									
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.5, \ell = 0.2$)	17.1	30.2	38.1	16.9	31.0	40.1	17.0	26.6	36.9
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.75, \ell = 0.2$)	16.9	**32.5**	**43.7**	18.9	**32.8**	**42.7**	16.9	30.8	40.3
Const.-Lin. ($\nu_0 = 0.4, \nu_1 = 0.75, \ell = 0.8$)	13.7	25.6	34.7	17.4	24.1	33.5	11.2	20.2	32.5
Constant ($\nu = 0.5$)	**21.1**	29.3	40.8	19.5	27.6	41.4	**20.5**	25.6	40.6
Constant ($\nu = 0.25$)	17.6	31.1	37.0	18.2	29.0	38.9	17.6	29.3	37.8

Table 30: **TinyGSM Simplex Diffusion (SDM) Accuracy (%)** ($T = 1.0$, 64 steps) across noise concentration schedules, sampling schedules (Linear, Cosine, Adaptive), and churn ($\kappa \in \{0.0, 0.2, 1.0\}$). We compare Expectation (P_t), Argmax, and Argmax with Self-Conditioning (+ SC). All SDMs are trained with the adaptive time sampler. We **bold** the best result in each column and underline the best result within each section (both if achieving both).

Concentration Schedule	Linear			Cosine			Adaptive		
	$\kappa = 0$	$\kappa = 0.2$	$\kappa = 1.0$	$\kappa = 0$	$\kappa = 0.2$	$\kappa = 1.0$	$\kappa = 0$	$\kappa = 0.2$	$\kappa = 1.0$
Expectation Input Processing (P_t)									
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.5, \ell = 0.2$)	6.7	9.8	14.7	14.3	16.8	22.8	10.6	**23.1**	**32.7**
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.75, \ell = 0.2$)	4.9	12.0	16.1	14.1	16.1	20.4	10.3	15.3	25.5
Constant ($\nu = 0.5$)	7.4	9.4	13.8	15.0	14.2	24.0	11.2	21.1	31.8
Constant ($\nu = 0.25$)	4.2	5.9	8.6	11.4	14.6	18.9	8.0	18.2	30.1
Constant ($\nu = 0.1$)	0.8	1.5	1.9	5.0	6.7	9.3	2.7	8.9	13.4
Argmax Input Processing									
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.5, \ell = 0.2$)	13.1	13.1	20.9	16.2	14.0	21.4	14.4	10.4	17.3
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.75, \ell = 0.2$)	9.7	13.5	22.8	16.1	14.7	22.4	14.1	11.0	17.1
Constant ($\nu = 0.5$)	**14.4**	13.2	22.0	15.6	10.0	22.5	13.7	9.2	19.3
Constant ($\nu = 0.25$)	12.6	13.3	21.7	16.8	13.5	21.6	15.3	11.9	17.4
Argmax + Self-Conditioning (SC)									
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.5, \ell = 0.2$)	11.9	**16.4**	24.1	17.6	15.8	25.0	13.2	14.6	22.4
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.75, \ell = 0.2$)	10.0	**16.4**	**25.7**	15.1	**17.5**	28.1	14.6	17.0	26.4
Const.-Lin. ($\nu_0 = 0.4, \nu_1 = 0.75, \ell = 0.8$)	6.2	15.1	22.5	13.4	12.9	23.1	9.4	11.6	18.2
Constant ($\nu = 0.5$)	13.6	16.3	25.5	**18.1**	13.6	**29.9**	**15.8**	17.0	25.6
Constant ($\nu = 0.25$)	13.5	15.5	22.0	14.8	16.4	24.7	13.5	16.0	22.1

Table 31: **TinyGSM Simplex Diffusion (SDM) Accuracy (%)** ($T = 0.1$, 512 steps) across noise concentration schedules, sampling schedules (Linear, Cosine, Adaptive), and churn ($\kappa \in \{0.0, 0.2, 1.0\}$). We compare Expectation (P_t), Argmax, and Argmax with Self-Conditioning (+ SC). All SDMs are trained with the adaptive time sampler. We **bold** the best result in each column and underline the best result within each section (both if achieving both).

Concentration Schedule	Linear			Cosine			Adaptive		
	$\kappa = 0$	$\kappa = 0.2$	$\kappa = 1.0$	$\kappa = 0$	$\kappa = 0.2$	$\kappa = 1.0$	$\kappa = 0$	$\kappa = 0.2$	$\kappa = 1.0$
Expectation Input Processing (P_t)									
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.5, \ell = 0.2$)	27.3	36.0	40.4	30.8	38.5	42.4	19.3	43.9	49.0
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.75, \ell = 0.2$)	27.7	36.4	38.2	30.8	39.3	42.5	19.8	40.8	44.3
Constant ($\nu = 0.5$)	27.0	30.5	41.7	29.5	33.3	43.3	22.2	35.0	44.8
Constant ($\nu = 0.25$)	18.3	25.0	30.1	22.5	31.2	38.5	11.3	38.3	42.5
Constant ($\nu = 0.1$)	7.7	10.2	13.8	9.3	17.0	22.1	2.9	17.5	21.0
Argmax Input Processing									
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.5, \ell = 0.2$)	33.0	40.8	43.8	**34.7**	39.8	45.5	31.5	39.7	42.9
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.75, \ell = 0.2$)	31.6	42.2	46.4	33.6	40.8	45.5	30.7	38.8	45.4
Constant ($\nu = 0.5$)	33.6	37.8	44.5	34.4	33.6	46.1	**34.5**	32.4	43.4
Constant ($\nu = 0.25$)	**34.2**	42.7	44.9	33.2	41.1	45.7	33.4	38.2	43.7
Argmax + Self-Conditioning (SC)									
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.5, \ell = 0.2$)	33.3	45.5	50.0	33.9	43.7	48.9	31.4	43.8	50.2
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.75, \ell = 0.2$)	31.6	**46.1**	49.6	33.4	**45.1**	50.5	31.0	**46.2**	50.3
Const.-Lin. ($\nu_0 = 0.4, \nu_1 = 0.75, \ell = 0.8$)	27.5	38.8	46.1	31.8	36.6	46.4	19.9	35.9	43.2
Constant ($\nu = 0.5$)	34.0	42.2	**50.6**	**34.7**	40.8	**50.9**	33.1	39.2	**50.6**
Constant ($\nu = 0.25$)	31.7	41.8	48.7	32.6	43.6	49.4	33.7	43.4	**50.6**

Table 32: **TinyGSM Simplex Diffusion (SDM) Accuracy (%)** ($T = 0.1$, 64 steps) across noise concentration schedules, sampling schedules (Linear, Cosine, Adaptive), and churn ($\kappa \in \{0.0, 0.2, 1.0\}$). We compare Expectation (P_t), Argmax, and Argmax with Self-Conditioning (+ SC). All SDMs are trained with the adaptive time sampler. We **bold** the best result in each column and underline the best result within each section (both if achieving both).

Concentration Schedule	Linear			Cosine			Adaptive		
	$\kappa = 0$	$\kappa = 0.2$	$\kappa = 1.0$	$\kappa = 0$	$\kappa = 0.2$	$\kappa = 1.0$	$\kappa = 0$	$\kappa = 0.2$	$\kappa = 1.0$
Expectation Input Processing (P_t)									
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.5, \ell = 0.2$)	11.3	16.7	20.5	25.7	28.7	32.0	18.9	33.0	39.5
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.75, \ell = 0.2$)	11.7	22.5	28.8	25.8	29.2	33.6	19.0	32.1	38.8
Constant ($\nu = 0.5$)	15.3	16.2	20.0	26.8	21.1	33.3	21.8	31.9	40.1
Constant ($\nu = 0.25$)	6.2	7.1	9.5	18.0	19.9	25.1	12.1	24.0	31.0
Constant ($\nu = 0.1$)	0.8	0.6	1.5	7.7	9.4	12.4	2.6	7.9	12.4
Argmax Input Processing									
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.5, \ell = 0.2$)	27.2	27.9	38.8	30.7	29.6	39.1	31.1	27.9	37.4
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.75, \ell = 0.2$)	23.4	30.1	38.5	32.4	30.9	39.3	27.9	29.2	37.4
Constant ($\nu = 0.5$)	**30.7**	28.6	37.4	32.1	22.1	36.7	**32.4**	25.7	37.8
Constant ($\nu = 0.25$)	29.7	30.8	35.4	31.9	28.6	37.8	32.1	27.5	37.4
Argmax + Self-Conditioning (SC)									
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.5, \ell = 0.2$)	26.6	30.9	40.6	31.2	33.9	42.0	31.1	33.6	42.0
Const.-Lin. ($\nu_0 = 0.2, \nu_1 = 0.75, \ell = 0.2$)	20.8	31.4	**42.0**	30.4	**34.3**	43.4	28.5	36.1	**46.0**
Const.-Lin. ($\nu_0 = 0.4, \nu_1 = 0.75, \ell = 0.8$)	13.3	29.2	36.9	26.7	25.4	38.8	18.6	28.7	37.9
Constant ($\nu = 0.5$)	29.6	**33.0**	41.0	**32.7**	27.3	**44.2**	32.2	**36.6**	44.5
Constant ($\nu = 0.25$)	27.5	29.9	39.4	30.1	32.5	40.8	30.7	33.0	42.5

Table 33: **Multi-Sample Accuracy and AST Structural Diversity ($K = 5$ samples per problem) on TinyGSM** across sampling steps NFE $\in \{8, 64, 512\}$ and temperatures $T \in \{0.1, 1.0\}$. †trained with uniform time sampler (evaluated with linear sampling schedule), ‡trained with adaptive time sampler (evaluated with adaptive sampling schedule). "ELBO" denotes the evidence lower bound objective, "SC" denotes Self-Conditioning, and "PC" denotes the Predictor-Corrector sampler. Arrows (↑ / ↓) indicate whether higher or lower values are better. For each column, we **bold** the overall best result and underline the best result within each section. For 64 and 512 steps, pass@1 is the single-sample accuracy from Tables 25 to 28 (baselines), Tables 29 to 32 (SDMs, $\nu_0 = 0.2$, $\nu_1 = 0.5$, $\ell = 0.2$) and Figure 13 (distilled SDM); for 8 steps it is the accuracy of the first of the K samples. For AR, this is greedy decoding in the $T = 0.1$ row and sampling in the $T = 1.0$ row.

Model	Temp. (T)	Steps (NFE)	pass@1 (%) ↑	pass@2 (%) ↑	pass@5 (%) ↑	AST Div. ↑	AST Div. (Correct) ↑
Autoregressive							
	0.1	512	**62.6**	**66.3**	69.6	9.6	6.2
	1.0	512	52.6	66.2	**77.6**	36.7	**29.5**
Absorbing Diffusion (MDM†; ELBO; Linear Schedule)							
	0.1	8	4.5	7.7	13.3	40.3	8.0
	0.1	64	27.4	37.7	52.3	33.4	19.2
	0.1	512	32.0	43.2	56.9	31.6	19.3
	1.0	8	0.5	0.8	1.4	**51.1**	14.3
	1.0	64	10.0	15.6	28.6	47.7	22.6
	1.0	512	15.8	24.1	38.6	44.8	23.4
Absorbing Diffusion (MDM‡; ELBO; SC; Adaptive Schedule)							
	0.1	8	19.6	29.7	43.8	30.4	14.6
	0.1	64	41.1	51.6	62.0	29.1	19.1
	0.1	512	42.6	54.9	65.0	28.3	18.9
	1.0	8	4.5	8.3	17.7	38.9	14.6
	1.0	64	18.7	28.9	43.6	41.7	21.7
	1.0	512	21.1	36.2	49.7	41.3	24.1
Uniform Diffusion (UDM†; ELBO; Linear Schedule)							
	0.1	8	11.2	17.3	28.7	35.9	15.4
	0.1	64	30.7	42.5	56.7	32.0	19.6
	0.1	512	33.0	46.1	59.6	31.9	19.7
	1.0	8	3.2	5.3	10.3	46.5	15.4
	1.0	64	14.4	22.3	36.1	44.1	23.6
	1.0	512	17.7	25.6	40.5	42.9	23.1
Uniform Diffusion (UDM†; PC; Linear Schedule)							
	0.1	8	11.5	18.1	31.0	39.0	17.9
	0.1	64	38.3	49.0	61.5	33.4	22.2
	0.1	512	45.5	56.8	67.1	30.6	21.3
	1.0	8	4.4	7.6	13.8	49.9	19.0
	1.0	64	23.3	34.9	51.1	44.3	26.8
	1.0	512	36.8	49.3	62.2	40.6	27.6
Simplex Diffusion (SDM‡; Expectation; $\kappa = 1$; Adaptive Schedule)							
	0.1	8	17.1	27.7	42.8	36.5	21.2
	0.1	64	39.5	52.4	64.8	32.3	22.5
	0.1	512	49.0	58.0	68.2	30.4	21.6
	1.0	8	5.4	9.7	17.9	40.0	17.3
	1.0	64	32.7	44.0	57.7	38.7	25.9
	1.0	512	45.8	55.1	66.6	35.3	24.8
Simplex Diffusion (SDM‡; Argmax + SC; $\kappa = 1$; Adaptive Schedule)							
	0.1	8	19.9	29.3	41.4	29.2	13.9
	0.1	64	42.0	55.7	64.5	26.8	17.5
	0.1	512	50.2	60.0	67.5	23.2	16.2
	1.0	8	3.9	6.2	12.2	25.0	7.8
	1.0	64	22.4	32.6	48.0	38.3	19.6
	1.0	512	36.9	50.1	62.6	35.3	22.3
Distilled Simplex Diffusion (SDM‡; Expectation; $\kappa = 1$; Adaptive Schedule)							
	0.1	8	31.9	39.2	48.1	17.4	7.9
	0.1	64	37.5	45.7	53.1	18.8	9.9
	0.1	512	40.6	46.3	53.7	18.0	10.8
	1.0	8	31.9	40.0	48.7	17.5	7.5
	1.0	64	37.8	45.8	53.6	19.0	10.7
	1.0	512	39.6	46.5	53.9	18.1	10.4