Journal of Machine Learning Research 26 (2025) 1-49 Submitted 1/25; Revised 8/25; Published 10/25

Unified Discrete Diffusion for Categorical Data

Lingxiao Zhao* LINGXIAOZLX@GMAIL.COM
Heinz College
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Xueying Ding* XDING2@CS.CMU.EDU
Heinz College
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Lijun Yu LIJUN@LJ-Y.COM
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Leman Akoglu LAKOGLU@ANDREW.CMU.EDU
Heinz College
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Editor: Alp Kucukelbir

Abstract

Discrete diffusion models have attracted significant attention for their application to naturally discrete data, such as language and graphs. While discrete-time discrete diffusion has been established for some time, it was only recently that Campbell et al. (2022) introduced the first framework for continuous-time discrete diffusion. However, their training and backward sampling processes significantly differ from those of the discrete-time version, requiring nontrivial approximations for tractability. In this paper, we first introduce a series of generalizations and simplifications of the evidence lower bound (ELBO) that facilitate more accurate and easier optimization both discrete- and continuous-time discrete diffusion. We further establish a unification of discrete- and continuous-time discrete diffusion through shared forward process and backward parameterization. Thanks to this unification, the continuous-time diffusion can now utilize the exact and efficient backward process developed for the discrete-time case, avoiding the need for costly and inexact approximations. Similarly, the discrete-time diffusion now also employ the MCMC corrector, which was previously exclusive to the continuous-time case. Extensive experiments and ablations demonstrate the significant improvement, and we open-source our code at: `https://github.com/LingxiaoShawn/USD3`.

Keywords: Diffusion Model, Generative Model

. *Equal contribution.

1. Introduction

Deep generative models have taken the world by storm, capturing complex data distributions and producing realistic data, from human-like text (Brown et al., 2020; Li et al., 2022; OpenAI, 2023) and natural looking images (Dhariwal and Nichol, 2021; Ramesh et al., 2022; Zhang et al., 2023) to novel compounds like molecules and drugs (Kang and Cho, 2018; Li et al., 2021) and video synthesis (Ho et al., 2022). Denoising diffusion models (Ho et al., 2020b), a powerful class of generative models, are trained through a forward diffusion process that gradually adds noise to the training samples, and a backward process that denoises these diffusion trajectories. New data are then generated by sampling from the noise distribution and employing the trained model for recursive denoising.

Discrete diffusion for categorical data has two modeling paradigms: discrete-time and continuous-time. The former discretizes time such that backward denoising is learned only at pre-specified time points. This limits generation, which can only "jump back" through fixed points. In contrast, continuous-time case allows a path through any point in range, and often yields higher sample quality. Current literature on discrete-time discrete diffusion is relatively established, while only recently Campbell et al. (2022) introduced the first continuous-time discrete diffusion framework. While groundbreaking, their their loss requires multiple network evaluations during training. Moreover, the exact sampling through their learned backward process is extremely tedious for multi-dimensional variables. Due to mathematically complicated and computationally demanding formulations, Campbell et al. (2022) propose nontrivial approximations for tractability with unknown potential errors.

In this paper, we introduce a series of improvements to two critical aspects of discrete diffusion: loss computation and backward sampling, for both discrete- and continuous-time scenarios. What is more, for the first time, we show that the discrete-time and continuous-time discrete diffusion can be unified together such that they share exactly the same forward diffusion and backward sampling process. We elaborate on the details with itemized contributions (1-5) and their benefits as follows:

- **Loss simplifcation and generalization**: We derive significantly simplified yet exact Evidence lower bound (ELBO) calculations for both discrete & continuous-time discrete diffusion, taking into account the properties of categorical data. Our simplified formulations allow both forward and backward probabilities to accommodate any noise distribution element-wisely, which is particularly attractive for multi-element objects where each element can exhibit different noise distribution. See Contribution 2 (Section 3.2.5) and Contribution 4 (Section 4.3) for discrete-time and continuous-time case respectively.

- **Efficient backward sampling**: We present a closed-form backward probability $p_\theta(\boldsymbol{x}_s|\boldsymbol{x}_t)$ for all $s < t$ with reconstruction-based model parameterization (see Contribution 1 in Section 3.2.3), paving the way for ELBO simplification and accelerated backward sampling, initially derived in the discrete-time case. Furthermore, we demonstrate that this improved backward process can be applied to continuous-time case, replacing the costly and inexact sampling approximations in Campbell et al. (2022).

- **Unification:** We demonstrate that continuous-time diffusion can share the exact same forward process as discrete-time diffusion for *any* noise distribution (Contribution 3 in

Section 4.2). This unified forward process, along with shared backward parameterization, leads to a unification of both forward and backward processes for discrete- & continuous-time cases. This offers mutual benefits: discrete-time diffusion can utilize continuous-time's MCMC corrector to enhance performance, while continuous-time diffusion can employ fast and exact backward sampling derived in the discrete-time case. See Contribution 5 in Section 4.4.

We present both discrete-time (Section 3) and continuous-time (Section 4) discrete diffusion in a self-contained manner, accompanied by easy-to-use open-source code at `https://github.com/LingxiaoShawn/USD3`. We explicitly mark our novel contributions in (sub)sections with summarization.

2. Related Work

We briefly discuss the recent advances in diffusion models and flow-maching models, while we present a detailed comparison with prior diffusion-based approaches in Table 1, highlighting the advantages of our formulation in terms of generality, simplicity, and performance.

Discrete Diffusion. Recent advances in discrete diffusion models, such as D3PM (Austin et al., 2021) and RDM (Zheng et al., 2023), modify the discrete-time diffusion process by modeling the transition probabilities as categorical distributions. These approaches operate within the discrete domain and reformulate the denoising process accordingly. In the continuous-time regime, Campbell et al. (2022) reinterpret the diffusion process as a Continuous-Time Markov Chain (CTMC), enabling the use of Kolmogorov forward and backward equations to align the transition probabilities of the forward and reverse processes. Building on this CTMC formulation, Lou et al. (2024) and Sun et al. (2023) adopt similar continuous-time perspectives but differ in how they marginalize the stochastic process and compute the training loss. These works emphasize score-matching objectives that, while effective, require specialized model architectures or sampling strategies that are often incompatible with standard discrete-time settings. In contrast, our method retains compatibility with the discrete-time framework while achieving efficient training and improved empirical performance. More recently, Sahoo et al. (2024); Shi et al. (2025) propose simplification specifically designed for absorbing state diffusion models (which also enable model backbone being autoregressive), while our method focuses on generalizing to any type of noise.

Flow Matching Models. Language modeling is one of the most prominent applications of discrete diffusion models. While most popular approaches rely on purely autoregressive models, alternative methods, such as Inverse Autoregressive Flows (IAF) (Ziegler and Rush, 2019), Autoregressive Argmax Flow (Hoogeboom et al., 2022), and Discrete Flows (Tran et al., 2019), have been developed to support both sequential and non-sequential generative tasks. Multinomial Diffusion, introduced in (Hoogeboom et al., 2021b), bridges discrete flows and diffusion models by incorporating a multinomial noise process. This formulation enables training with tractable variational objectives and supports efficient sampling via argmax-based transitions. Our work is also closely related to Gat et al. (2024), while

following CTMC paradigm, considers deterministic mappings between data and reference distributions via optimal transport.

Table 1: Comparison of USD3 and Other Diffusion Models. USD3 provides several unification steps for discrete and continuous time diffusion, yielding simplification that generalize to various noise types.

Method	**Forward Process Posterior**	**Backward Parameterization p_θ**	**Loss Formulation**	**Sampling**	**Type**	**Drawbacks**
D3PM (Austin et al., 2021)	Matrix-form $q(x_{t-1}\|x_0, x_t)$ without simplification	Integral-form $p_\theta(\mathbf{x}_{t-1}\|\mathbf{x}_t)$ without simplification	ELBO with cross-entropy (CE) loss	Direct sampling x_{t-1} from x_t	Discrete Only	Inefficient computation, no close-form parameterization
RDM (Zheng et al., 2023)	Simplified $q(x_{t-1}\|x_0, x_t)$	Reparameterized & Approximated $p_\theta(\mathbf{x}_{t-1}\|\mathbf{x}_t)$ with routing variable	Simplified CE loss	Direct sampling x_{t-1} from x_t, combining with sampling routing variable	Discrete Only	Routing mechanism with approximation error
τ-LDR (Campbell et al., 2022)	CTMC with forward transition rate, no discrete-time posterior	CTMC with reverse transition rate, no discrete-time parameterization	Continuous-time ELBO	Tau-leaping algorithm	Continuous Only	Two forward passes in loss calculation, approximated backward sampling with error
SDDM (Sun et al., 2023)	CTMC with forward transition rate, no discrete-time posterior	Reverse CTMC using marginal ratio, leave-one-out param. $p_\theta(\mathbf{x}_t^d\|\mathbf{x}_t^{\setminus d})$	Generalized ratio matching	Analytical sampling improving Euler's method	Continuous Only	Need Hollow Transformer for leave-one-out parameterization
SEDD (Lou et al., 2024)	CTMC with forward transition rate, no discrete-time posterior	Reverse CTMC using ratio, $s_\theta(x,t) \approx \left[\frac{p_t(y)}{p_t(x)}\right]_{y\neq x}$	Generalized score matching	Discrete Tweedie's Theorem	Continuous Only	Approximated transition probabilities may cause imprecise sampling, exponential terms risk extreme or unstable values
MDLM (Sahoo et al., 2024)	Matrix-form $q(x_s\|x_t, x_0)$ for abosorbing noise	Direct parameterization of $p_\theta(x_s\|x_t)$ for absorbing noise	Simplified loss based on Rao-Blackwellization	Ancestral Sampling	Discrete & Continuous time	Simplification & sampling work with absorbing noise only
MD4 (Shi et al., 2025)	Matrix-form $q(x_s\|x_t, x_0)$ for abosorbing noise	$q(x_s\|x_t, f_\theta(x_t,t))$ with mean parameterization	Simplified ELBO for absorbing noise	Ancestral sampling	Discrete & Continuous time	Simplification on absorbing noise only, sampling is based on discrete-time reverse process only
USD3 (Ours)	**Simplified and unified conditional $q(x_s\|x_t, x_0)$ for any $0 < s < t \leq T$ and any type of noise (Contribution 3)**	**Simplified closed-form parameterization of $p_\theta(x_s\|x_t)$ (Contribution 1)**	**Simplified ELBO with auxiliary cross-entropy loss (Contribution 2 & 4)**	**Unified sampling and unified MCMC corrector steps for both discrete and continuous time (Contribution 5)**	**Discrete & Continuous**	**None**

3. Discrete-time Discrete Diffusion

Notation: Let $\mathbf{x}_0 \sim p_{\text{data}}(\mathbf{x}_0)$ be the random variable of observed data with underlying distribution $p_{\text{data}}(\mathbf{x}_0)$. Let $\mathbf{x}_t \sim q(\mathbf{x}_t)$ be the latent variable at time t of a single-element object, like a pixel of an image or a node/edge of a graph, with maximum time T. Let $\mathbf{x}_{t|s} \sim q(\mathbf{x}_t|\mathbf{x}_s)$ be the conditional random variable. We model the *forward diffusion* process independently for each element of the object, while the *backward denoising* process is modeled jointly for all elements of the object. For simplicity and clarity of presentation, we first assume that the object only has 1 element and extend to multi-element object later. Let $\mathbf{x}_0^{1:D}$ denote the object with D elements, and $\mathbf{x}_t^i$ be the i-th element of latent object at time t. We assume all random variables take categorical values from $\{1, 2, ..., K\}$. Let $\boldsymbol{e}_k \in \{0, 1\}^K$ be the one-hot encoding of category k. For a random variable $\mathbf{x}$, we use $\boldsymbol{x}$ denoting its one-hot encoded sample where $\boldsymbol{x} \in \{\boldsymbol{e}_1, ..., \boldsymbol{e}_K\}$. Also, we interchangeably use $q(\mathbf{x}_t|\mathbf{x}_s)$, $q(\mathbf{x}_t = \boldsymbol{x}_t|\mathbf{x}_s = \boldsymbol{x}_s)$, and $q_{t|s}(\boldsymbol{x}_t|\boldsymbol{x}_s)$ when no ambiguity is introduced. Let $\langle \cdot, \cdot \rangle$ denote inner product. All vectors are column-wise vectors.

3.1 Graphical Model View of Diffusion Models

Diffusion models can be represented by latent variable graphical models (see Appx. Fig. 3). We can write the joint probability as $p_\theta(\mathbf{x}_{0:T}) := p_\theta(\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_T) = p_\theta(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ using the Markov condition. Parameters θ are learned by maximizing the loglikehood of the observed variable $\mathbf{x}_0$: $\log p_\theta(\mathbf{x}_0) = \log \int p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T}$. However the marginalization is intractable, and instead the following ELBO is used:

$$\log p_\theta(\mathbf{x}_0) = \log \int q(\mathbf{x}_{1:T}|\mathbf{x}_0) \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} d\mathbf{x}_{1:T} \geq \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} [\log p_\theta(\mathbf{x}_{0:T}) - \log q(\mathbf{x}_{1:T}|\mathbf{x}_0)]. \quad (1)$$

The above inequality holds for any conditional probability $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$ and finding the best $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$ to tighten the bound is the inference problem in graphical models (i.e. E step in EM algorithm). Exact inference is intractable, thus $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$ in diffusion models is fixed or chosen specifically to simplify the learning objective. To simplify Eq. (1), it is important to assume $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$ is *decomposable*. The typical assumption is $q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1})$ (Ho et al., 2020a), which we also adopt. Others that have been explored include $q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)$ (Song et al., 2021a).

Assuming $q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1})$, Eq. (1) can be simplified as the following

$$\underbrace{\mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)}[\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)]}_{-\mathcal{L}_1(\theta)} - \underbrace{D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)||p_\theta(\mathbf{x}_T))}_{\mathcal{L}_{\text{prior}}} - \sum_{t=2}^T \underbrace{\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}[D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)]}_{\mathcal{L}_t(\theta)} \quad (2)$$

Where $\mathcal{L}_{\text{prior}} \approx 0$, since $p_\theta(\mathbf{x}_T) \approx q(\mathbf{x}_T|\mathbf{x}_0)$ is designed as a fixed noise distribution that is easy to sample from. (See Appendix 9.1 for derivation.) To compute Eq. (2), we need to formalize distributions (i) $q(\mathbf{x}_t|\mathbf{x}_0)$ and (ii) $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$, as well as (iii) the parameterization of $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$. We specify these, respectively, in Section 3.2.1, Section 3.2.2, and Section 3.2.3.

3.2 Components of Discrete-time Discrete Diffusion

3.2.1 The Forward Diffusion Process

We assume each discrete random variable $\mathbf{x}_t$ has a categorical distribution, i.e. $\mathbf{x}_t \sim \text{Cat}(\mathbf{x}_t; \boldsymbol{p})$ with $\boldsymbol{p} \in [0, 1]^K$ and $\mathbf{1}^\top \boldsymbol{p} = 1$. One can verify that $p(\mathbf{x}_t = \boldsymbol{x}_t) = \boldsymbol{x}_t^\top \boldsymbol{p}$, or simply $p(\mathbf{x}_t) = \boldsymbol{x}_t^\top \boldsymbol{p}$.

As shown in (Hoogeboom et al., 2021a; Austin et al., 2021), the forward process with discrete variables $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ at t step can be represented as a transition matrix $Q_t \in [0,1]^{K\times K}$ such that $[Q_t]_{ij} = q(\mathbf{x}_t = \boldsymbol{e}_j|\mathbf{x}_{t-1} = \boldsymbol{e}_i)$. Then, we can write the distribution explicitly as

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathrm{Cat}(\mathbf{x}_t; Q_t^\top \boldsymbol{x}_{t-1}) \ . \tag{3}$$

Given transition matrices $Q_1, ..., Q_T$, the t-step marginal distribution conditioning on $s, \forall t > s$ is

$$q(\mathbf{x}_t|\mathbf{x}_s) = \mathrm{Cat}(\mathbf{x}_t; \overline{Q}_{t|s}^\top \boldsymbol{x}_s), \text{ with } \overline{Q}_{t|s} = Q_{s+1}...Q_t \ . \tag{4}$$

The s-step posterior distribution conditioning on $\mathbf{x}_0$ and t-step can be derived as

$$q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0) = \frac{q(\mathbf{x}_t|\mathbf{x}_s)q(\mathbf{x}_s|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} = \mathrm{Cat}(\mathbf{x}_s; \frac{\overline{Q}_{t|s}\boldsymbol{x}_t \odot \overline{Q}_s^\top \boldsymbol{x}_0}{\boldsymbol{x}_t^\top \overline{Q}_t^\top \boldsymbol{x}_0}), \ \forall t > s \ . \tag{5}$$

The above formulations are valid (see derivations in Appendix 9.3) for *any* transition matrices $Q_1, ..., Q_T$. However, to achieve uninformative noise $\mathbf{x}_T$, $\{Q_i\}_{i=1}^T$ should be chosen such that every row of $\overline{Q}_t = \overline{Q}_{t|0}$ converge to the same known stationary distribution when t becomes large enough (at T). Let the known stationary distribution be $\mathbf{m}_0 \sim \mathrm{Cat}(\mathbf{m}_0; \boldsymbol{m})$. Then, the constraint can be stated as

$$\lim_{t\to T} \overline{Q}_t = \mathbf{1}\boldsymbol{m}^\top \ . \tag{6}$$

In addition, this paper focuses on *nominal data* (see Appendix 9.2) where categories are unordered and only equality comparison is defined. Hence, no ordering prior *except checking equality* should be used to define Q_t[1]. To achieve the desired convergence on nominal data while keeping the flexibility of choosing any categorical stationary distribution $\mathbf{m}_0 \sim \mathrm{Cat}(\mathbf{m}_0; \boldsymbol{m})$, we define Q_t as

$$Q_t = \alpha_t I + (1-\alpha_t)\mathbf{1}\boldsymbol{m}^\top \ , \tag{7}$$

where $\alpha_t \in [0,1]$. This results in the accumulated transition matrix $\overline{Q}_{t|s}$ being equal to

$$\overline{Q}_{t|s} = \overline{\alpha}_{t|s} I + (1-\overline{\alpha}_{t|s})\mathbf{1}\boldsymbol{m}^\top, \ \forall t > s \ , \tag{8}$$

where $\overline{\alpha}_{t|s} = \prod_{i=s+1}^t \alpha_i$. Note that $\overline{\alpha}_t = \overline{\alpha}_{t|0} = \overline{\alpha}_{t|s}\overline{\alpha}_s$. We can achieve uninformative noise while satisfying the constraint Eq. (6) by choosing α_t such that $\lim_{t\to T} \overline{\alpha}_t = 0$.

3.2.2 Analytical Form of $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$

The formulation in Eq. (7) can be used to simplify $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$. We provide a general formulation of $q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0)$ for any s, t with $0 < s < t \leq T$, which will be useful for unifying with continuous-time diffusion. One can recover $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ by setting s as $t-1$.

Proposition 1 (Zheng et al. (2023)) *For both discrete- and continuous-time discrete diffusion, we can write the conditional distribution as*

$$q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0) = \begin{cases} Cat\Big(\mathbf{x}_s; \ (1-\lambda_{t|s})\cdot \boldsymbol{x}_t + \lambda_{t|s}\cdot \boldsymbol{m}\Big) & \text{when } \boldsymbol{x}_t = \boldsymbol{x}_0 \\ Cat\Big(\mathbf{x}_s; \ (1-\mu_{t|s})\cdot \boldsymbol{x}_0 + \mu_{t|s}\overline{\alpha}_{t|s}\cdot \boldsymbol{x}_t + \mu_{t|s}(1-\overline{\alpha}_{t|s})\cdot \boldsymbol{m}\Big) & \text{when } \boldsymbol{x}_t \neq \boldsymbol{x}_0 \end{cases} \tag{9}$$

where $\lambda_{t|s} := \frac{(1-\overline{\alpha}_s)(1-\overline{\alpha}_{t|s})\langle \boldsymbol{m}, \boldsymbol{x}_t\rangle}{\overline{\alpha}_t + (1-\overline{\alpha}_t)\langle \boldsymbol{m}, \boldsymbol{x}_t\rangle}, \ \mu_{t|s} := \frac{1-\overline{\alpha}_s}{1-\overline{\alpha}_t}$.

We remark that the above formulation is originally presented in Zheng et al. (2023) with $s = t-1$. Appendix 9.3 gives the detailed derivation of the probability's formulation.

1. Mathematically, this means $\forall k, j, \ q(\mathbf{x}_t|\mathbf{x}_{t-1} = \boldsymbol{e}_j, \mathbf{x}_t \neq \{\boldsymbol{e}_j, \boldsymbol{e}_k\}) = q(\mathbf{x}_t|\mathbf{x}_{t-1} = \boldsymbol{e}_k, \mathbf{x}_t \neq \{\boldsymbol{e}_j, \boldsymbol{e}_k\})$.

3.2.3 Contribution 1: Analytical Form of $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$

The literature has explored three different parameterizations of $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$: (1) parameterizing $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ directly; (2) parameterizing $p_\theta(\mathbf{x}_0|\mathbf{x}_t)$ with f_t^θ such that $p_\theta(\mathbf{x}_0|\mathbf{x}_t) = \text{Cat}(\mathbf{x}_0; f_t^\theta(\mathbf{x}_t))$ and letting $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = q(\mathbf{x}_{t-1}|\mathbf{x}_t, f_t^\theta(\mathbf{x}_t))$; and (3) parameterizing $p_\theta(\mathbf{x}_0|\mathbf{x}_t)$ with f_t^θ and then marginalizing $q(\mathbf{x}_{t-1}, \mathbf{x}_0|\mathbf{x}_t)$ such that $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \sum_{\mathbf{x}_0} q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)p_\theta(\mathbf{x}_0|\mathbf{x}_t)$.

Method (1) does not reuse any distribution in forward process, and hence is less effective. Method (2) has been widely used for continuous diffusion models as in Ho et al. (2020a) and Song et al. (2021a), and some discrete diffusion models like in Hoogeboom et al. (2021a) and Zheng et al. (2023). It avoids marginalization and works effectively for continuous diffusion. However for discrete diffusion, as shown in Eq. (9), sample $\mathbf{x}_0$ determines which categorical distribution should be used, which cannot be determined without the true $\mathbf{x}_0$. Some heuristics have been proposed in Zheng et al. (2023), however those can have a large gap to the true $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$, leading to an inaccurate sampling process.

Method (3) has been used in Austin et al. (2021) by directly marginalizing out $\mathbf{x}_0$, which introduces additional computational cost in both loss function computation and sampling process as the formulation of $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ has not been simplified to a closed-form distribution. In this paper, we show that method (3) parameterization can be simplified to a clean formulation of categorical distribution.

Proposition 2 *The parameterization of $p_\theta(\mathbf{x}_s|\mathbf{x}_t)$ is simplified for any $0 < s < t \leq T$ as*

$$p_\theta(\mathbf{x}_s|\mathbf{x}_t) = Cat\Big(\mathbf{x}_s; (1-\mu_{t|s})\cdot f_t^\theta(\boldsymbol{x}_t) + (\mu_{t|s}\overline{\alpha}_{t|s} + \gamma_{t|s}^\theta)\cdot \boldsymbol{x}_t + (\mu_{t|s}(1-\overline{\alpha}_{t|s}) - \gamma_{t|s}^\theta)\cdot \boldsymbol{m}\Big) \quad (10)$$

where $\gamma_{t|s}^\theta$ is affected by $f_t^\theta(\mathbf{x}_t)$, $\gamma_{t|s}^\theta := (\mu_{t|s} - \lambda_{t|s} - \mu_{t|s}\overline{\alpha}_{t|s})\langle f_t^\theta(\boldsymbol{x}_t), \boldsymbol{x}_t\rangle$, and from Eq. (9), $\lambda_{t|s} := \frac{(1-\overline{\alpha}_s)(1-\overline{\alpha}_{t|s})\langle \boldsymbol{m}, \boldsymbol{x}_t\rangle}{\overline{\alpha}_t + (1-\overline{\alpha}_t)\langle \boldsymbol{m}, \boldsymbol{x}_t\rangle}$, $\mu_{t|s} := \frac{1-\overline{\alpha}_s}{1-\overline{\alpha}_t}$.

The detailed proof is in Appendix 9.4. Notice that $f_t^\theta(\boldsymbol{x}_t)$ is a parameterized neural network with $\mathbf{1}^T f_t^\theta(\boldsymbol{x}_t) = 1$. As we show next, the above formulation simplifies the negative ELBO loss computation greatly (Section 3.2.4), further motivates an approximated loss that is much easier to optimize (Section 3.2.5), and accelerates the sampling process through reparameterization (Section 9.8).

3.2.4 Loss Function

With $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ in Eq. (9) and $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ in Eq. (10), the $\mathcal{L}_t(\theta)$ in Eq. (2) can be written as

$$\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\Big[\delta_{\mathbf{x}_t,\mathbf{x}_0} D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t = \mathbf{x}_0)\|p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) + (1-\delta_{\mathbf{x}_t,\mathbf{x}_0}) D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t \neq \mathbf{x}_0)\|p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))\Big], \quad (11)$$

where $\delta_{\mathbf{x}_t,\mathbf{x}_0}$ denotes the Kronecker delta of $\mathbf{x}_t$ and $\mathbf{x}_0$. $q(\mathbf{x}_{t-1}|\mathbf{x}_t = \mathbf{x}_0)$ and $q(\mathbf{x}_{t-1}|\mathbf{x}_t \neq \mathbf{x}_0)$ represent the first and second categorical distribution in Eq. (9), respectively.

Apart from negative ELBO, another commonly employed auxiliary loss is the cross-entropy (CE) loss between $q(\mathbf{x}_t|\mathbf{x}_0)$ and $p_\theta(\mathbf{x}_0|\mathbf{x}_t)$, which measures the reconstruction quality.

$$\mathcal{L}_t^{CE}(\theta) := \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}[-\log p_\theta(\mathbf{x}_0|\mathbf{x}_t)] \quad (12)$$

Eq. (12) and Eq. (11) share the same global minima with $p_\theta(\mathbf{x}_0|\mathbf{x}_t)$ being the true posterior $q(\mathbf{x}_0|\mathbf{x}_t)$. However they have different optimization landscape; and thus under limited data and network capacity, which loss is be easier to minimize is unknown (Tewari and Bartlett, 2007; Demirkaya et al., 2020).

3.2.5 Contribution 2: Further Loss Simplification for Easier Optimization

While Eq. (11) is the *exact* negative ELBO loss, in practice we find it harder to minimize than $\mathcal{L}_t^{CE}$. In this section, we first derive a much simpler, approximated loss by observing a relation between $q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0)$ and $p_\theta(\mathbf{x}_t|\mathbf{x}_0)$. We then show that coefficient simplification and $\mathcal{L}_t^{CE}$ are *both* valuable for optimizing a general negative ELBO where only partial time steps are observed. We combine all designs to derive the final approximated loss, denoted as $\tilde{\mathcal{L}}_t$.

Proposition 3 *For any $0 < s < t \leq T$, with $\mathbf{x}_0$ known, $\triangle \boldsymbol{p}_\theta(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0)$ is defined as*

$$p_\theta(\mathbf{x}_s|\mathbf{x}_t) - q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0) = (1 - \mu_{t|s})[f_t^\theta(\boldsymbol{x}_t) - \boldsymbol{x}_0 + \phi_{t|s}\langle f_t^\theta(\boldsymbol{x}_t) - \boldsymbol{x}_0, \boldsymbol{x}_t\rangle(\boldsymbol{x}_t - \boldsymbol{m})] \,, \quad (13)$$

where $\phi_{t|s} := \frac{(1-\bar{\alpha}_s)\bar{\alpha}_{t|s}}{\bar{\alpha}_t + (1-\bar{\alpha}_t)\langle \mathbf{x}_t, \boldsymbol{m}\rangle}$.

Proposition 3 shows that the distribution difference between $p_\theta(\mathbf{x}_s|\mathbf{x}_t)$ and $q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0)$ has a closed-form formulation. (Proof in Appendix 9.5) With it, we can apply the Taylor expansion (up to second order) to approximate the KL divergence directly (See Appendix 9.6.)

$$D_{\mathrm{KL}}(q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0) \| p_\theta(\mathbf{x}_s|\mathbf{x}_t)) \approx \sum_{\mathbf{x}_s} \frac{|\triangle \boldsymbol{p}_\theta(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0)|^2}{q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0)} \quad (14)$$

The above formulation along with Proposition 3 are valid for *any* $0 < s < t \leq T$. We next show that minimizing divergence between q and p_θ at any s and t is also valid, as it is inside a general negative ELBO with partial time steps. The initial version of the ELBO is derived under the assumption that observations are made at every time step. Its backward denoising process is designed to advance by a single time step during each generation step for best generation quality. Let us consider a more general case where only partial time steps are observed in the forward process, then, minimizing its negative ELBO can help improve generation quality with fewer steps. Assuming only $\mathbf{x}_s$ and $\mathbf{x}_t$ are observed, where $0 < s < t \leq T$, a derivation analogous to that of Eq. (2) can show that

$$\log p_\theta(\mathbf{x}_0) \geq -D_{\mathrm{KL}}[q(\mathbf{x}_t|\mathbf{x}_0)\|p_\theta(\mathbf{x}_t)] + \mathbb{E}_{q(\mathbf{x}_s|\mathbf{x}_0)}[\log p_\theta(\mathbf{x}_0|\mathbf{x}_s)] - D_{\mathrm{KL}}[q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0)\|p_\theta(\mathbf{x}_s|\mathbf{x}_t)], \quad (15)$$

where the first divergence term between the prior and posterior quantifies the quality of the backward denoising process from T to t. The second term represents the CE loss, $\mathcal{L}_s^{\mathrm{CE}}$, which influences the generation quality from time s to time 0. The final term is given by Eq. (14), and contributes to the generation process from t to s. (See Appendix 9.7 for proof.)

This generalized formulation of the ELBO highlights the significance of the CE loss and the alignment between $q(\mathbf{x}_s|\mathbf{x}_s, \mathbf{x}_0)$ and $p_\theta(\mathbf{x}_s|\mathbf{x}_t)$ at any observed times s and t. While CE loss does not have a coefficient that depends on noise schedules and time, changing s and t or noise schedule (which determines $\bar{\alpha}_t$) during training will greatly impact the scale of the term in Eq. (14). By rendering the loss scaling term independent of time and the noise schedule, the minimization of this adjusted loss concurrently leads to the minimization of the original loss in Eq. (14) for any given s and t. Hence, by removing the sensitive scale $\frac{(1-\mu_{t|s})^2}{q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0)}$ in Eq. (14), we reformulate the loss as

$$\mathcal{L}_t^2 := \|f_t^\theta(\boldsymbol{x}_t) - \boldsymbol{x}_0 + \phi_{t|s}\langle f_t^\theta(\boldsymbol{x}_t) - \boldsymbol{x}_0, \boldsymbol{x}_t\rangle(\boldsymbol{x}_t - \boldsymbol{m})\|_2^2, \quad (16)$$

where we can further clip $\phi_{t|s}$ to $\min(1, \phi_{t|s})$ for minimal scaling influence. It is important to note that while we have modified the coefficient to be invariant to the noise schedule and time s to effectively minimize Eq. (14) at any t and s, a similar approach to coefficient

revision has been previously explored in Ho et al. (2020a); Karras et al. (2022), primarily to facilitate an easier optimization by achieving a balance of terms in the loss function.

Overall, we have the final approximated loss as $\tilde{\mathcal{L}}_t(\theta) = \mathcal{L}_t^2(\theta) + \mathcal{L}_t^{CE}(\theta)$. We find that in practice this loss is much easier to optimize than the original exact negative ELBO for more complicated distributions, leading to faster convergence and improved generation quality: On the VQCIFAR10 image generation task, we observe that the model trained with the approximated loss (denoted as USD3*) converges to a lower training cross-entropy loss more quickly than the model trained with the full, non-simplified discrete-time loss, especially during the early stages of training. We attribute this improvement to the easier optimization and smoother gradient landscape provided by the approximation. As training progresses, the gap between the simplified and non-simplified models narrows, but USD3* continues to perform comparably or even better, as demonstrated in later experiments.

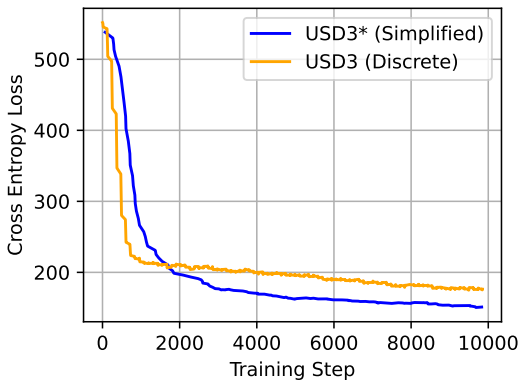


Figure 1: Models trained with simplified loss (USD3*) in Eq. (16) vs. non-simplified discrete-time loss (USD3) on VQCIFAR10 image generation task. Simplified loss shows faster convergence during early training stage (first 10,000 steps).

4. Continuous-time Discrete Diffusion

Despite being simple, discrete-time diffusion limits the generation process as we can only "jump back" through fixed time points. Recent works generalize continuous-state diffusion models to continuous-time (Song et al., 2021b). This generalization enables great flexibility in backward generation as one can "jump back" through any time in $[0, T]$ with improved sample quality.

Nevertheless, generalizing discrete-state diffusion model from discrete-time to continuous-time is nontrivial, as the score-matching based technique (Song et al., 2021b) in continuous-state models requires the score function $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ to be available. This function, however, is evidently non-existent for discrete distributions. Recently, Campbell et al. (2022) presented the first continuous-time diffusion model for discrete data. It formulates the forward process through a Continuous Time Markov Chain (CTMC) and aims at learning a reverse CTMC that matches the marginal distribution with the forward CTMC at any time t. While being theoretically solid, the formulation in Campbell et al. (2022) has two problems: **(1)** the negative ELBO loss of matching the forward and backward CTMCs is analytically complicated and hard to implement; and **(2)** the exact sampling through the learned backward CTMC is unrealistic. Campbell et al. (2022) propose approximate solutions, which however, trade off computational tractability with unknown errors and sample quality. SDDM Sun et al. (2023) takes a different approach with ratio matching (Hyvärinen, 2007; Lyu, 2009), which is the generalization of score matching to discrete data. However, it needs

a specific model architecture. Lou et al. (2024) advances ratio matching, yielding a loss that requires only a single model pass. Interestingly, this loss is similar to ours in formulation, despite originating from a different angle.

In this paper, we build on the CTMC formulation in (Campbell et al., 2022), and show that the loss can be analytically simplified for nominal data. This simplification also inspires an improved MCMC corrector with closed-form formulation. For problem **(2)**, we argue that the difficulty arises from directly using the learned backward CTMC's transition rate. Instead, realizing that the reverse transition rate is computed based on the learned $p_\theta(\mathbf{x}_0|\mathbf{x}_t)$, we propose to compute $p_\theta(\mathbf{x}_s|\mathbf{x}_t)$ through $p_\theta(\mathbf{x}_0|\mathbf{x}_t)$ without using the transition rate matrix. This avoids the approximation error of sampling directly from the backward CTMC and greatly simplifies the generation process.

Remarkably, with the new formulation of generation, we show that the continuous-time and discrete-time diffusion models can be unified, with exactly the same forward diffusion and now also backward generation process. Moreover, we demonstrate that this unification offers mutual benefits: the continuous-time diffusion can leverage the swift and precise sampling formulation derived from the discrete-time case (as detailed in Section 9.8), while the discrete-time diffusion can utilize the MCMC corrector from the continuous-time scenario (see VQCIFAR10 experiment section).

4.1 Background: Continuous-Time Markov Chain

CTMC generalizes Markov chain from discrete- to continuous-time via the Markov property: $\mathbf{x}_{t_1} \perp\!\!\!\perp \mathbf{x}_{t_3} \mid \mathbf{x}_{t_2}$, $\forall t_1 < t_2 < t_3$. Anderson (2012) provides an introduction to time-homogeneous CTMC. It can be derived from discrete-time Markov chain by increasing the number of time stamps N to infinite while keeping the total time T fixed. Specifically, we can define $\triangle t = \frac{T}{N}$, $t_i = i\triangle t$, and a discrete-time Markov chain characterized by transition probability $q(\mathbf{x}_{t_i}|\mathbf{x}_{t_{i-1}})$ and transition matrix Q_{t_i} with $[Q_{t_i}]_{jk} = q(\mathbf{x}_{t_i} = \boldsymbol{e}_k|\mathbf{x}_{t_{i-1}} = \boldsymbol{e}_j)$. By setting N to infinite, the transition probability $q(\mathbf{x}_{t_i}|\mathbf{x}_{t_{i-1}})$ converges to 0, hence is not suitable for describing CTMC. Instead, CTMC is fully characterized by its *transition rate* $r_t(\boldsymbol{y}|\boldsymbol{x})$, s.t.

$$r_t(\boldsymbol{y}|\boldsymbol{x}) = \lim_{N\to\infty} \frac{q(\mathbf{x}_{t_i} = \boldsymbol{y}|\mathbf{x}_{t_{i-1}} = \boldsymbol{x}) - \delta_{\boldsymbol{x},\boldsymbol{y}}}{t_i - t_{i-1}} = \lim_{\triangle t\to 0} \frac{q_{t|t-\triangle t}(\boldsymbol{y}|\boldsymbol{x}) - \delta_{\boldsymbol{x},\boldsymbol{y}}}{\triangle t}\,. \tag{17}$$

As the name suggests, $r_t(\boldsymbol{y}|\boldsymbol{x})$ measures the changing rate of the transition probability of moving from state $\boldsymbol{x}$ to state $\boldsymbol{y}$ at time t in the direction of the process. The corresponding *transition rate matrix* R_t with $[R_t]_{ij} = r_t(\boldsymbol{e}_j|\boldsymbol{e}_i)$ fully determines the underlying stochastic process. A CTMC's transition probabilities satisfy the Kolmogorov equations (Kolmogoroff, 1931) (see Appendix9.10), which have unique solution. As Rindos et al. (1995) stated, when R_{t_1} and R_{t_2} commute (i.e. $R_{t_1}R_{t_2} = R_{t_2}R_{t_1}$) for any t_1, t_2, the transition probability matrix can be written as $\overline{Q}_{t|s} = \exp\left(\int_s^t R_a da\right)$, where $\exp(M) := \sum_{k=0}^{\infty} \frac{M^k}{k!}$. The commutative property of R_t can be achieved by choosing $R_t = \beta(t)R_b$ where $R_b \in \mathbb{R}^{K\times K}$ is a time-independent base rate matrix.

4.2 Contribution 3: Simplified Forward & Backward CTMCs that Connect to Discrete-Time Case

Forward CTMC. Two properties are needed for modeling the forward process: P1) the process can converge to an easy-to-sample stationary distribution at final time T; and P2) the conditional marginal distribution $q(\mathbf{x}_t|\mathbf{x}_0)$ can be obtained analytically for efficient training. As given above, P2) can be achieved by choosing commutative transition rate matrices with $R_t = \beta(t)R_b$.

We next show that property P1), i.e. $\lim_{t\to T} \overline{Q}_{t|0} = \overline{Q}_{T|0} = \mathbf{1}\boldsymbol{m}^\top$ for some stationary distribution $\boldsymbol{m}$, can be achieved by choosing $R_b = \mathbf{1}\boldsymbol{m}^\top - I$, which is a valid transition rate matrix with the property $(-R_b)^2 = (-R_b)$ (see derivation in Appendix 9.11). Then we have

$$\overline{Q}_{t|s} = \exp(\overline{\beta}_{t|s}R_b) = e^{-\overline{\beta}_{t|s}}I + (1 - e^{-\overline{\beta}_{t|s}})\mathbf{1}\boldsymbol{m}^\top. \tag{18}$$

★ Unified forward process. Eq. (18) has the same formulation as the transition matrix of the discrete-time case in Eq. (8), if we set $\overline{\alpha}_{t|s} = \exp(-\overline{\beta}_{t|s}) = \exp(-\int_s^t \beta(a)da)$. Thus, *this formulation unifies the forward processes of adding noise for both discrete- and continuous-time discrete diffusion.* With $\lim_{t\to T} \overline{\alpha}_{t|0} = 0$, or equivalently $\lim_{t\to T} \int_0^T \beta(a)da = \infty$, we achieve the goal $\overline{Q}_{T|0} = \mathbf{1}\boldsymbol{m}^\top$. We use $\overline{\alpha}_{t|s}$ directly in the following sections, i.e. Eq. (8). To summarize, for the forward CTMC

$$R_t = \beta(t)(\mathbf{1}\boldsymbol{m}^\top - I)\ \text{, and} \quad \overline{Q}_{t|s} = \overline{\alpha}_{t|s}I + (1 - \overline{\alpha}_{t|s})\mathbf{1}\boldsymbol{m}^\top\ . \tag{19}$$

We can further get vector-form forward rate

$$r_t(\boldsymbol{x}|\cdot) = R_t\boldsymbol{x} = \beta(t)\big(\langle \boldsymbol{x}, \boldsymbol{m}\rangle \mathbf{1} - \boldsymbol{x}\big), \quad r_t(\cdot|\boldsymbol{x}) = R_t^\top \boldsymbol{x} = \beta(t)(\boldsymbol{m} - \boldsymbol{x})\ . \tag{20}$$

Backward CTMC. To generate samples from the target distribution, we have to reverse the process of the forward CTMC. Let $\widehat{r}_t$ be the transition rate of the backward CTMC with corresponding matrix $\widehat{R}_t$. When the forward and backward CTMCs are matched exactly, theoretically the forward and backward CTMCs have the following relationship (see Campbell et al. (2022)'s Proposition 1):

$$\widehat{r}_t(\boldsymbol{x}|\boldsymbol{y}) = r_t(\boldsymbol{y}|\boldsymbol{x})\frac{q_t(\boldsymbol{x})}{q_t(\boldsymbol{y})}, \quad \forall \boldsymbol{x} \neq \boldsymbol{y}\ . \tag{21}$$

However, $q_t(\boldsymbol{x})$ and $q_t(\boldsymbol{y})$ are intractable analytically, hence we cannot derive the backward CTMC directly from Eq. (21). Instead, Campbell et al. (2022) parameterize the transition rate $\widehat{r}_t^\theta$, by observing that $\frac{q_t(\boldsymbol{x})}{q_t(\boldsymbol{y})} = \sum_{\boldsymbol{x}_0} \frac{q_{t|0}(\boldsymbol{x}|\boldsymbol{x}_0)}{q_{t|0}(\boldsymbol{y}|\boldsymbol{x}_0)} q_{0|t}(\boldsymbol{x}_0|\boldsymbol{y})$[2], as follows

$$\widehat{r}_t^\theta(\boldsymbol{x}|\boldsymbol{y}) = r_t(\boldsymbol{y}|\boldsymbol{x}) \sum_{\boldsymbol{x}_0} \frac{q_{t|0}(\boldsymbol{x}|\boldsymbol{x}_0)}{q_{t|0}(\boldsymbol{y}|\boldsymbol{x}_0)} p_{0|t}^\theta(\boldsymbol{x}_0|\boldsymbol{y}). \tag{22}$$

Then, $\widehat{r}_t^\theta$ is learned with θ to minimize the continuous-time negative ELBO introduced next.

Negative ELBO. Similar to the discrete-time case, the backward CTMC can be learned by maximizing the ELBO for data log-likelihood. Computing ELBO for CTMC is nontrivial, and fortunately Campbell et al. (2022) has derived (see their Proposition 2) that the negative ELBO can be formulated as

$$T\ \mathbb{E}_{\substack{t\sim \mathrm{Uni}(0,T) \\ \boldsymbol{x}\sim q(\mathbf{x}_t|\mathbf{x}_0)}} \Big[\sum_{\boldsymbol{z}\neq\boldsymbol{x}} \widehat{r}_t^\theta(\boldsymbol{z}|\boldsymbol{x}) - \sum_{\boldsymbol{z}\neq\boldsymbol{x}} r_t(\boldsymbol{z}|\boldsymbol{x}) \log \widehat{r}_t^\theta(\boldsymbol{x}|\boldsymbol{z}) \Big]\ . \tag{23}$$

2. As $q_t(\boldsymbol{x}) = \sum_{\boldsymbol{x}_0} q_{t|0}(\boldsymbol{x}|\boldsymbol{x}_0) q_{\mathrm{data}}(\boldsymbol{x}_0)$ and $q_t(\boldsymbol{x}) = \frac{q_{t|0}(\boldsymbol{x}|\tilde{\boldsymbol{x}}_0) q_{\mathrm{data}}(\tilde{\boldsymbol{x}}_0)}{q_{0|t}(\tilde{\boldsymbol{x}}_0|\boldsymbol{x})}$.

However, Campbell et al. (2022)'s original design did not simplify the ELBO with the parameterization of $\widehat{r}_t^\theta$ in Eq. (22), making the implementation nontrivial and inefficient. We show that their formulation can be greatly simplified to a closed-form evaluation.

To simplify Eq. (23), we introduce $g_t^\theta(\boldsymbol{x}|\boldsymbol{y})$ such that $\widehat{r}_t^\theta(\boldsymbol{x}|\boldsymbol{y}) = r_t(\boldsymbol{y}|\boldsymbol{x})g_t^\theta(\boldsymbol{x}|\boldsymbol{y})$, with

$$g_t^\theta(\boldsymbol{x}|\boldsymbol{y}) := \sum_{\boldsymbol{x}_0} \frac{q_{t|0}(\boldsymbol{x}|\boldsymbol{x}_0)}{q_{t|0}(\boldsymbol{y}|\boldsymbol{x}_0)} p_{0|t}^\theta(\boldsymbol{x}_0|\boldsymbol{y}) \approx \frac{q_t(\boldsymbol{x})}{q_t(\boldsymbol{y})} \,, \tag{24}$$

which is the estimator of the marginal probability ratio.

Proposition 4 *(Proof in Appendix 9.12) The vector form parameterization of $g_t^\theta(\boldsymbol{x}|\boldsymbol{y})$ can be simplified analytically as:*

$$g_t^\theta(\cdot|\boldsymbol{y}) = \Big[(1 - \frac{\overline{\alpha}_{t|0}\langle f_t^\theta(\boldsymbol{y}), \boldsymbol{y}\rangle}{\overline{\alpha}_{t|0} + (1-\overline{\alpha}_{t|0})\langle \boldsymbol{y}, \boldsymbol{m}\rangle})\boldsymbol{m} + \frac{\overline{\alpha}_{t|0}}{1-\overline{\alpha}_{t|0}} f_t^\theta(\boldsymbol{y})\Big] \odot \frac{\mathbf{1}-\boldsymbol{y}}{\langle \boldsymbol{y}, \boldsymbol{m}\rangle} + \boldsymbol{y} \tag{25}$$

4.3 Contribution 4: Simplification of Continuous-time Negative ELBO

As the derivation is much harder in multi-element case, we work on multi-element derivation directly and the single-element case can be induced as a special case. Given $\boldsymbol{x}^{1:D}$, let $\boldsymbol{x}^{\setminus d}$ represent $\boldsymbol{x}^{1:D\setminus d}$, i.e. the object without d-th element.

We first need to generalize the definition of transition rate of forward and backward CTMC to multi-element case. We present the extension below, and give the proof in Appendix 9.13.

$$r_t^{1:D}(\boldsymbol{z}^{1:D}|\boldsymbol{x}^{1:D}) = \sum_{d=1}^D r_t^d(\boldsymbol{z}^d|\boldsymbol{x}^d)\delta_{\boldsymbol{x}^{\setminus d},\boldsymbol{z}^{\setminus d}}, \quad \widehat{r}_t^{\theta,1:D}(\boldsymbol{z}^{1:D}|\boldsymbol{x}^{1:D}) = \sum_{d=1}^D r_t^d(\boldsymbol{x}^d|\boldsymbol{z}^d) g_t^{\theta,d}(\boldsymbol{z}^d|\boldsymbol{x}^{1:D}) \cdot \delta_{\boldsymbol{x}^{\setminus d},\boldsymbol{z}^{\setminus d}} \tag{26}$$

where $\delta_{\boldsymbol{x},\boldsymbol{y}}$ is the Kronecker delta, $r_t^{1:D}$ is the forward transition rate and $\widehat{r}_t^{\theta,1:D}$ is the backward transition rate parameterization. As we assume that the forward processes are independent for different elements, r_t^d represents the transition rate of the forward CTMC process at the d-th element.

Similarly, we can extend the parameterization of $g_t^\theta(\boldsymbol{x}|\boldsymbol{y})$ to multi-element formulation $g_t^\theta(\boldsymbol{x}^{1:D}|\boldsymbol{y}^{1:D})$. The proof can be found in Appendix 9.13, Specifically, the target parameterization $g_t^\theta(\boldsymbol{x}^{1:D}|\boldsymbol{y}^{1:D})$ satisfyes $g_t^\theta(\boldsymbol{x}^{1:D}|\boldsymbol{y}^{1:D}) = \prod_{d=1}^D g_t^{\theta,d}(\boldsymbol{x}^d|\boldsymbol{y}^{1:D})$, where

$$g_t^{\theta,d}(\boldsymbol{x}^d|\boldsymbol{y}^{1:D}) := \sum_{\boldsymbol{x}_0^d} \frac{q_{t|0}(\boldsymbol{x}^d|\boldsymbol{x}_0^d)}{q_{t|0}(\boldsymbol{y}^d|\boldsymbol{x}_0^d)} p_{0|t}^\theta(\boldsymbol{x}_0^d|\boldsymbol{y}^{1:D}) \tag{27}$$

With these notations and concepts, we present the simplified negative ELBO as follows

Proposition 5 *The negative ELBO in Eq. (23) in multi-element case can be simplified as*

$$T\,\mathbb{E}_{\substack{t\sim Uni(0,T)\\ \boldsymbol{x}^{1:D}\sim q_{t|0}(\cdot|\boldsymbol{x}_0^{1:D})\\ \boldsymbol{z}^{1:D}\sim S_t(\cdot|\boldsymbol{x}^{1:D})}} \Big[\sum_{d=1}^D r_t^d(\boldsymbol{x}^d|\cdot)^\top g_t^{\theta,d}(\cdot|\boldsymbol{x}^{1:D}) - \frac{1}{\mathcal{M}_{S_t}(\boldsymbol{z}^{1:D}|\boldsymbol{x}_0^{1:D})} \sum_{d=1}^D \frac{\mathbf{1}^\top\left[q_{t|0}(\cdot|\boldsymbol{x}_0^d)\odot r_t^d(\boldsymbol{z}^d|\cdot)\odot \log g_t^{\theta,d}(\cdot|\boldsymbol{z}^{1:D})\right]}{q_{t|0}(\boldsymbol{z}^d|\boldsymbol{x}_0^d)}\Big] \tag{28}$$

where $S_t(\boldsymbol{z}^{1:D}|\boldsymbol{x}^{1:D})$ is any unnormalized distribution (see Eq. (84) in Appendix) of sampling the auxiliary variable $\boldsymbol{z}^{1:D}$ from $\boldsymbol{x}^{1:D}$. The auxiliary variable is introduced to avoid multiple passes of the model for computing the second term of Eq. (23). M_{S_t} is a normalization scalar (see Eq. (88) in the Appendix) that depends only on $S_t, \boldsymbol{z}^{1:D}$, and $\boldsymbol{x}_0^{1:D}$.

The detailed proof is in Appendix 9.14. This formulation simplifies and generalizes the result in Campbell et al. (2022), so that any S_t can be used to introduce the auxiliary variable $\boldsymbol{z}_{1:D}$. Importantly, Eq. (28) shows that changing S_t only affects a scalar weight. Computing the loss requires two passes of the model for $\boldsymbol{z}^{1:D}$ and $\boldsymbol{x}^{1:D}$ separately. Now we show that it can be further simplified to only a single pass of the model.

Proposition 6 *The loss in Eq. (28) can be further simplified as*

$$T\,\mathbb{E}_{\substack{t\sim Uni(0,T)\\ \boldsymbol{x}^{1:D}\sim q_{t|0}(\cdot|\boldsymbol{x}_0^{1:D})}}\Big[\sum_{d=1}^{D} r_t^d(\boldsymbol{x}^d|\cdot)^\top\Big(g_t^{\theta,d}(\cdot|\boldsymbol{x}^{1:D}) - \frac{q_{t|0}(\cdot|\boldsymbol{x}_0^d)\odot \log g_t^{\theta,d}(\cdot|\boldsymbol{x}^{1:D})}{q_{t|0}(\boldsymbol{x}^d|\boldsymbol{x}_0^d)}\Big)\Big] \tag{29}$$

The detailed proof can be found in Appendix 9.15. Note that this loss requires only a single forward pass of the model, making it efficient. Additionally, this loss aligns with the DWDSE loss (while presented in single-element case) in Lou et al. (2024), which is derived from the different concrete score matching direction. This indicates that the ELBO is essentially the same as the ratio matching loss, as also demonstrated in the continuous-state setting, such that the score matching Song et al. (2021b) is shown to be equivalent to ELBO computation in Ho et al. (2020a).

4.4 Contribution 5: Fast Backward Sampling & Unification of Discrete Diffusion

Campbell et al. (2022) proposes to use the learned transition rate $\widehat{r}_t^\theta$ of the backward CTMC to sample reversely for the target distribution $p_{\text{data}}(\mathbf{x}_0)$. However, different from the forward CTMC where all elements are sampled independently, the backward CTMC processes for all elements are coupled together and do not have the closed-form transition probability derived from the learned transition rate. Direct and exact sampling from a CTMC uses the algorithm by Gillespie (1977), which is intractable for multi-element objects. Campbell et al. (2022) proposes using tau-leaping (Gillespie, 2001), which approximately samples all transitions from time t to $t-\tau$, assuming $R_t^{1:D}$ remains fixed during the time period. While tractable, it introduces unknown errors that need a correction process.

We propose to avoid the costly sampling from using the transition rate of backward CTMC. We first observe that the estimated transition rate $\widehat{r}_t^\theta$ is essentially derived from the estimated $p_{0|t}^\theta(\mathbf{x}_0|\mathbf{x}_t)$. Hence using $\widehat{r}_t^\theta$ is equivalent to using $p_{0|t}^\theta(\mathbf{x}_0|\mathbf{x}_t)$ in an indirect way. We realize that the transition probability $q_{s|t}(\mathbf{x}_s|\mathbf{x}_t)$ can be computed easily from $p_{0|t}^\theta(\mathbf{x}_0|\mathbf{x}_t)$ directly, as shown in Eq. (10). With the help of Eq. (10), we can sample $\mathbf{x}_0$ or equivalently $\mathbf{x}_{0|T}$ through sampling $\mathbf{x}_{t_{n-1}|t_n}, ..., \mathbf{x}_{t_1|t_2}, \mathbf{x}_{t_0|t_1}$ sequentially, with any $\{t_i\}_{i=0}^n$ that satisfies $0{=}t_0 < t_1 < ... < t_n{=}T$. Although Eq. (10) is derived for discrete-time case, it applies directly in continuous-time case, thanks to the unification of the forward process for discrete- and continuous-time diffusion (§4.2). Hence, discrete-time and continuous-time diffusion also have the same **unified backward generation process**. Note that since our method also estimates transition rate probability, it can adapt to tau-leaping sampling method in

Algorithm 1 USD3 Unified Training: Red: discrete-time step, and blue: continuous-time step.

Input: A stationary distribution $\boldsymbol{m}$, samples $\sim p_{\text{data}}$, weight factor λ, max time T
repeat
  Draw $\mathbf{x}_0 \sim p_{\text{data}}(\mathbf{x}_0)$
  Draw $t \sim \textbf{Uniform}(0, ..., T)$ or $t \sim \textbf{Uniform}(0, 1)$
  Compute $\overline{\alpha}_t$ from Table 9
  Draw $\mathbf{m}_0^{1:D} \sim \text{Cat}(\mathbf{m}_0^{1:D}; \boldsymbol{m})$
  $\mathbf{x}_{t|0}^{1:D} = \delta_{1,\mathbf{b}_t^{1:D}}^{1:D} \odot \mathbf{x}_0^{1:D} + (1 - \delta_{1,\mathbf{b}_t^{1:D}}^{1:D}) \odot \mathbf{m}_0^{1:D}$ where $\mathbf{b}_t \sim \text{Bernoulli}(\overline{\alpha}_t)$, from Eq. (51)
  Compute $f_t^\theta(\boldsymbol{x}_t^{1:D})$ as parameterization of $p_\theta(\mathbf{x}_0^{1:D}|\mathbf{x}_t^{1:D})$
  Take gradient descent step on $\nabla_\theta\{\mathcal{L}_t(\theta) + \lambda\mathcal{L}_t^{CE}(\theta)\}$ (from Eq. (11) + Eq. (12))
    or $\nabla_\theta\{\mathcal{L}_t^{CTMC}(\theta) + \lambda\mathcal{L}_t^{CE}(\theta)\}$ (from Eq. (23) + Eq. (12))
until convergence

Algorithm 2 USD3 Unified Sampling

Input: A stationary distribution $\boldsymbol{m}$, $\{t_i\}_{i=0}^n$ s.t. $0 = t_0 < t_1 < ... < t_n = T$, learned $f_{t_i}^\theta$.

MCMC Input: use_MCMC, step size $\triangle n$, total steps N.
Set $\mathbf{x}_n^{1:D} \sim \text{Cat}(\mathbf{x}_n^{1:D}; \boldsymbol{m})$
for $i \in \{n, ..., 0\}$ **do**
  Compute $f_{t_i}^\theta(\boldsymbol{x}_i^{1:D})$ and $p_\theta(\mathbf{x}_{i-1}^{1:D}|\mathbf{x}_i^{1:D})$ from Eq. (10)
  Draw $\mathbf{x}_{i-1}^{1:D} \sim \text{Cat}(\mathbf{x}_{i-1}^{1:D}; p_\theta(\mathbf{x}_{i-1}^{1:D}|\mathbf{x}_i^{1:D}))$
  If use_MCMC:
    $\mathbf{x}_{i-1}^{1:D} \leftarrow \text{MCMC\_Corrector}(t_i, \mathbf{x}_{i-1}^{1:D}, \triangle n, f_{t_i}^\theta, N)$ (Algo. 3)
return $\mathbf{x}_0$

Campbell et al. (2022); Lou et al. (2024), while sampling with fewer time steps is not the focus of our paper, which we will leave for future work.

★ Unified Discrete Diffusion. All in all, through a series of mathematical simplifications, we have shown that both discrete&continuous-time discrete diffusion share **(1)** the same forward diffusion process; **(2)** the same parameterization for learning $p_{0|t}^\theta$; and **(3)** the same backward denoising process. In light of our reformulations, we propose USD3, a novel Unified and Simplified Discrete Denoising Diffusion model. USD3 utilizes the same source code for both discrete- and continuous-time, up to a single alteration in the loss function during training (Algorithm 1), and a shared sample generation process (Algorithm 2).

5. Experiments

Discrete diffusion has limited and no established guidelines on how to do model training. Many prior work optimize a combined ELBO and cross-entropy (CE) loss with a fixed weight, without deeper investigation. In this work we not only improve ELBO loss mathematically, but also empirically explore an extensive testbed of training regimes for discrete diffusion—including various loss combinations, both discrete- and continuous-time training, as well as

sampling schedules—toward a deeper understanding of which yield tractable optimization and higher generation quality. USD3 is a hybrid of our simplified exact ELBO and CE, and USD3* refers to its approximation with further simplifications (Section 3.2.5 and Section 4.3). USD3-CE and USD3-ELBO are variants, with respect with CE or ELBO loss *only*.

Configurations. In USD3, we combine ELBO and CE with weight 0.001 for the latter, following prior literature. For USD3*, we combine ELBO with CE weight 1.0, as it is suggested in Section 3.2.5 and Section 4.3. For architecture, we parameterize $f_t^\theta(\mathbf{x}_t)$ with a Diffusion Transformer model (Peebles and Xie, 2023).

5.1 Music Generation

Lakh Pianoroll Datasets. We evaluate monophonic music generation on Piano, the cleaned Lakh pianoroll dataset (Raffel, 2016; Dong et al., 2017), containing $6,000$ training and 973 evaluation (or test) sequences of 256 notes each. Here we perform conditional generation; given the first 32 notes, the models are required to generate the rest 224 notes.

Metrics. In evaluation of the quality of music generation, we apply the following metrics (the detailed description is given in Appendix 9.17.3):

- $\{1, 2, 3\}$-gram Hellinger Distance (↓) and $\{1, 2, 3\}$-gram Proportion of Outliers (↓): these metrics measure the similarities of generated music notes by n-gram sequences.
- Diverse Edit Distance (↑): measuring the creativity/novelty of generated samples across multiple generation runs.
- Train-to-Test Ratios (↑): these ratios quantify the extent of "parroting"; a smaller ratio indicates a higher (lower) degree of memorization (generalization).

Training Details. For training USD3 on the Piano dataset, we use a similar Diffusion Transformer architecture described in SEDD (12 layers, each with 12 heads, input dimension of 768 and max MLP dimension of 3072) . We apply a batch size of 64, a learning rate of $2e^{-4}$, with a warmup of the first 5000 steps. We adopt a consine learning rate scheduler with EMA $= 0.999$. The result is over $800,000$ steps. We run our results with 1 A6000 GPU. In both continuous- and discrete- time diffusion, we use a cosine scheduler with $\alpha = 0.008$. More details about baselines can be found in Appendix 9.17.5.

Table 2: Conditional music gen. quality (3 samples avg.) on Piano w.r.t. n-gram Hellinger Distance, n-gram Proportion of Outliers, and Diverse Edit Distance. Following previous practices (Campbell et al., 2022; Sun et al., 2023), numbers are rounded to four decimal places. Also, we report Train-to-Test Ratio for 2-gram Proportion of Outliers and Hellinger Distance on Piano-P to quantify "parroting". **First** & **Second** shown in color. Full mean and standard deviation across three experimental runs are in Table 13. Full Test-to-Ratio results across n-gram Hellinger Distances and Proportion of Outliers are in Table 14.

	Method	n-gram Hellinger(↓) 1gram	2gram	3gram	n-gram Prop. of Out.(↓) 1gram	2gram	3gram	Div. Edit Dist. (↑)	2g-Prop. Ratio(↑)	2g.Hellinger Ratio(↑)
Discrete-time	D3PM	0.3982	0.5303	0.5918	0.1209	0.2532	0.3790	**0.2950**	5.7052	1.3623
	RDM	0.4123	0.5964	0.6198	0.1401	0.2459	0.3864	**0.2891**	5.6050	1.4023
	USD3-CE	0.3677	0.4493	0.5236	0.1068	0.1669	0.2455	0.0482	7.3224	1.4722
	USD3-ELBO	0.3676	0.4483	0.5229	0.1142	0.1670	0.2528	0.0485	**7.9642**	1.5159
	USD3	**0.3672**	0.4487	0.5234	0.1126	0.1757	0.2562	0.0484	7.5278	1.5601
	USD3*	0.3680	0.4485	0.5251	0.1110	0.1749	0.2538	0.0502	7.6239	**1.5985**
Continuous-time	SDDM	0.3759	0.4856	0.5773	0.1101	0.2059	0.3009	0.0612	7.0654	1.4516
	τ-LDR-0	0.3796	0.4681	0.5710	0.1149	0.2078	0.3202	0.0504	7.0864	1.4726
	SEDD	0.3715	0.4472	0.5250	**0.1052**	0.1723	0.2509	0.0501	7.0603	1.4898
	USD3-CE	**0.3658**	**0.4465**	**0.5201**	**0.1043**	**0.1630**	**0.2414**	0.0483	6.5462	1.5033
	USD3-ELBO	0.3673	0.4474	0.5204	0.1125	0.1685	0.2455	0.0504	7.3620	**1.5945**
	USD3	0.3676	**0.4467**	**0.5198**	0.1097	**0.1668**	**0.2423**	0.0498	**8.2954**	1.5912

Results. Table 2 shows that USD3 and its variants perform better than the baseline methods across almost all metrics. An exception is RDM's and D3PM's Diverse Edit Distance, because they trade-off high generation novelty with low generation quality.

Within USD3, the continuous-time variants achieve the best results across the board with respect to the Hellinger n-gram and the Outlier proportion n-gram, due to their ability to denoise and generate through any timesteps. Although both discrete-time and continuous-time USD3-CE perform well within n-gram metrics, they do not compete with USD3-ELBO and USD3 in terms of Train-To-Test Ratios. This indicates that *CE only* loss can lead to more "parroting" of the training data, potentially due to the ease of overfitting with a single cross-entropy loss. The combined USD3 achieves a higher Train-to-Test ratio than pure USD3-CE or USD3-ELBO, suggesting that the combination of two losses can alleviate the overfitting phenomenon and difficulty of optimization for pure ELBO loss.

We show the full mean and standard deviation across three experimental runs in Table 13. In addition, we show the full Test-to-Ratio results across n-gram Hellinger Distances and Proportion of Outliers in Table 14.

5.2 Image Generation

VQCIFAR10 Dataset. For the image generation task, we train all the models on CIFAR10 images. While the original CIFAR10 contains $50,000$ training images in continuous values, which we instead convert to vectors from 64×512-dimensional quantization hash-code

Table 3: Image gen. quality w.r.t. Inception Score (IS) and the Frechet Inception Dist. (FID) over 50,000 samples generated and decoded by VQGAN, as compared against original CIFAR10 training images. **First** & **Second** shown in color.

	Method	IS (↑)	FID (↓)
Discrete-time	D3PM	8.13	18.08
	RDM	7.16	24.23
	USD3-CE	9.10	13.62
	USD3*	9.08	**11.89**
	USD3	**9.17**	**11.92**
Cont.-time	SDDM	8.72	14.17
	τ-LDR	8.37	17.61
	SEDD	9.12	13.01
	USD3-CE	9.13	13.05
	USD3	**9.20**	12.67
VQGAN Recons.		10.42	7.68

Table 4: MCMC gen. quality with USD3.

MCMC Config.	IS (↑)	FID (↓)
Cont.-time, N=2, $\Delta n = 0.005$	9.15	12.87
Cont.-time, N=5, $\Delta n = 0.001$	9.24	12.08
Discrete-time, N=2, $\Delta n = 0.005$	**9.31**	13.53
Discrete-time, N=5, $\Delta n = 0.001$	9.15	**11.88**

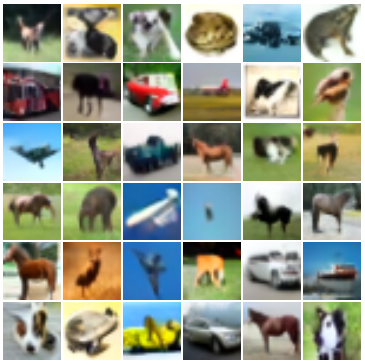


Figure 2: Overview of decoded samples unconditionally generated by USD3.

space with a pre-trained VQGAN (Esser et al., 2020) in MaskGIT (Chang et al., 2022). The reconstruction gives a FID of 7.68 and IS of 10.42, using the Inception V3 Model [3]. This conversion allows us to (1) evaluate our methods on *nominal* data by breaking the neighboring orders in the image space, and (2) select a closed-form stationary distribution $\boldsymbol{m}$ (details in Appendix 9.17.4).

Metrics. For evaluating image quality, we first feed the generated samples through the VQGAN decoder to obtain representations from the discretized space. Then, we measure the Inception Score (IS) (↑) and Frechet Inception Distance (FID) (↓) against the original CIFAR10 dataset. Note that our training set, which are discretized images from VQGAN, achieves IS=10.42 and FID=7.68, which are optimistic limits for generation.

Training Details. We apply a Diffusion Transformer, with 12 layers and train 1 million steps. In discrete-time diffusion, all the USD3 and its variants use the same cosine noise scheduler with $\alpha = 0.008$. For continuous-time diffusion, and for continuous-time , USD3 utilizes a constant noise scheduler with 0.007, to match the scheduler in τ-LDR-0 and SDDM. More training configurations about baselines can be found in Appendix 9.17.6.

Results. Table 3 shows results with USD3 and its variants along with the baselines. Discrete-time D3PM and RDM fall short of the harder image generation task, whereas our simplified discrete-time loss increases the FID of discrete-time USD3. In continuous-time methods, τ-LDR with a similar loss to USD3 falls short due to its complicated generation

3. `https://github.com/openai/consistency_models/tree/main/evaluations`

process, while SDDM and SEDD are the most competitive among the baselines. SDDM requires substantial compute resources while being limited to a specialized model architecture. Continuous-time USD3 achieves the best overall IS. While USD3 and SEDD share similar loss formulations in training, USD3's sampling procedure provides an additional boost in the quality of generated samples. Discrete-time USD3 and its simplified USD3* achieve the best FID score. Figure 2 provides a qualitative overview of samples generated by USD3 and decoded with VQGAN.

Additional results. We conduct an additional experiment on the trained USD3 to show that the derived MCMC further improves the image generation quality. We apply MCMC sampling with various hyper-parameters for $50,000$ sampled images at the last 10% of the sampling phase (last 100 timesteps), for both discrete- and continuous-time USD3. The result is shown in Table 4. Thanks to the unification scheme for discrete-time and continuous-time diffusions, our proposed MCMC is able to boost up the performances of both methods, which provide better generation quality than all the baselines.

5.3 Unconditional Text Generation

Text8. For unconditional text generation, we measure USD3's performance against a list of diffusion and autoregressive models, on text8 dataset. The text8 dataset is a character-level text corpus with a compact vocabulary of 27 tokens, comprising the letters 'a' -'z' and the '_' whitespace token. We adhere to the previous practices in Austin et al. (2021), training and evaluating text8 in chunks of length 256, without additional preprocessing.

Training Details. We use a Diffusion Transformer with 12 layers and train 1 million steps with batch size 512. Since absorbing noise has been shown to perform better than uniform noise in Austin et al. (2021); Lou et al. (2024), our USD3 is trained with absorbing noise as well. The noise schedule type we have used is log-linear with $\epsilon = 1e-3$, similar to SEDD.

Results. We evaluate the unconditional generation results with Bits Per Character (BPC) on all test sets. Table 5 shows USD3 performances against existing benchmarks (The baselines are taken from Graves et al. (2024); Austin et al. (2021); Lou et al. (2024). In addition, the IAF/SCF results are shown in Ziegler and Rush (2019), the Autoregressive Argmax Flow in Hoogeboom et al. (2022), the Discrete Flow in Tran et al. (2019), Multinomial Diffusion in Hoogeboom et al. (2021b), and MAC in Shih et al. (2022)). The discrete-time diffusion USD3 improves over Multinomial Diffusion and D3PM on their reported benchmarks. For continuous-time diffusion, USD3 performs comparably to SEDD, since USD3's loss function aligns with the DWDSE in Lou et al. (2024).

Table 5: Comparison of different methods in terms of BPC (bits per character) (↓). USD3 is compared against both autoregressive and non-autoregressive benchmarks.

Type	**Method**	**BPC (↓)**
Autoregressive & Flows	IAF/SCF	1.88
	AR Argmax Flow	1.39
	Discrete Flow	1.23
	Autoregressive	1.23
	Bayesian Flow Network	$\leq$ 1.41
Discrete-Time Diffusion	Multinomial Diffusion	$\leq$ 1.72
	D3PM Absorb	$\leq$ 1.45
	USD3 Absorb	$\leq$ **1.43**
Continuous-Time Diffusion	SEDD Absorb	$\leq$ 1.39
	USD3 Absorb	$\leq$ **1.40**

5.4 Ablation Study

While many previous works optimize a combined ELBO and cross-entropy (CE) loss and fix the cross-entropy coefficients at 0.001, we conduct an additional ablation experiment on different combinations of weight coefficients. We train the several diffusion models on Lakpiano dataset with the same backbone architecture and as in training schemes as the previous experiment. We alter the weights on cross-entropy only. We set CE to be in the range $\{0, 0.001, 0.1, 1.0\}$. When $CE = 0.0$, it is our USD3-ELBO model that purely optimizes the ELBO loss. USD3-CE refers to a model that optimizes with cross-entropy loss equals to 1.0 and not repliant on ELBO loss ($ELBO = 0.0$).

Table 6 shows the evaluation metrics of different loss coefficients. This ablation study shows that the combination of CE loss and ELBO effectively balances generalization and pattern recognition in the data. While CE loss alone in the continuous-time diffusion yields strong results across n-gram Hellinger distances and Proportion of Outliers measurements, it produces the least diverse samples (with least Diverse Edit Distance) and Train-to-Test Ratios for Proportion of Outliers and Hellinger Distances. Adding larger CE terms in the loss function also affects the Diverse Edit Distance and Train-to-Test Ratios for generated samples, while $CE = 0.001$ and USD3-ELBO are often the first or second place in providing music notes that do not "parrot" training data. The result shows that CE term in the diffusion optimization stage is more likely to cause overfitting, often “parroting” training data and limiting generalization to unseen examples. These findings highlight the need to carefully balance loss coefficients for optimal robustness and adaptability.

Table 6: Metrics comparing different loss combinations for Lakh Pianoroll. For each of n-gram Hellinger Distance (ng.-Hellinger) and Proportion of Outliers (ng.-Prop. Outlier) metrics, we show mean with respect to 3 generated samples. The top two are highlighted by **First**, **Second**. We report Train-to-Test Ratio for 2-gram Proportion of Outliers and Hellinger Distance on Piano-P to quantify "parroting".

	Method	n-gram Hellinger(↓) 1gram	2gram	3gram	n-gram Prop. of Out.(↓) 1gram	2gram	3gram	Div. Edit Dist. (↑)	2g-Prop. Ratio(↑)	2g.Hellinger Ratio(↑)
Discrete-time	USD3-ELBO	0.3676	0.4483	0.5229	0.1142	0.1670	0.2528	0.0485	**7.9642**	1.5159
	CE=0.001	0.3672	0.4487	0.5234	0.1126	0.1757	0.2562	0.0484	7.5278	1.5601
	CE=0.1	0.3664	0.4472	0.5221	0.1132	0.1747	0.2544	0.0485	7.5923	1.5532
	CE=1.0	**0.3654**	0.4472	0.5228	0.1118	0.1752	0.2576	0.0482	7.3524	1.4826
	USD3-CE	0.3677	0.4493	0.5236	**0.1068**	0.1669	0.2455	0.0482	7.3224	1.4722
Continuous-time	USD3-ELBO	0.3673	0.4474	0.5204	0.1125	0.1685	0.2455	**0.0504**	7.3620	**1.5945**
	CE=0.001	0.3676	**0.4467**	**0.5198**	0.1097	**0.1668**	0.2423	**0.0498**	**8.2954**	**1.5912**
	CE=0.1	0.3670	**0.4465**	0.5197	0.1106	0.1695	0.2449	0.0479	7.3821	1.5026
	CE=1.0	0.3673	0.4486	0.5204	0.1070	0.1679	**0.2417**	0.0480	7.0124	1.5253
	USD3-CE	**0.3658**	**0.4465**	**0.5201**	**0.1043**	**0.1630**	**0.2414**	0.0483	6.5462	1.5033

5.5 Memory and Running-time Comparison

Table 7: The GPU-memory, running time and number of network parameters in all methods.USD3 is easier to train and incurs the least GPU memory in both discrete- and continuous- time diffusions.

	Method	Num. Parameters	Memory	Runtime		Method	Num. Parameters	Memory	Runtime
Discrete-time	D3PM	∼ 102,700,000	15669MiB	93 hrs	Continuous-time	SDDM	∼ 120,350,000	85528MiB	96 hrs
	RDM	∼ 102,700,000	9570MiB	86 hrs		SEDD	∼ 102,700,000	4620MiB	85 hrs
	USD3-CE	∼ 102,700,000	9735MiB	83 hrs		τ-LDR-0	∼ 102,700,000	17669 MiB	129 hrs
	USD3*	∼ 102,700,000	9735MiB	82 hrs		USD3-CE	∼ 102,700,000	15703MiB	93 hrs
	USD3	∼ 102,700,000	9735MiB	90 hrs		USD3*	∼ 102,700,000	17620MiB	91 hrs
						USD3	∼ 102,700,000	17620MiB	101 hrs

Table 7 provides the running time, number of parameters (in the transformer architecture) and GPU-memory for VQCIFAR10 generation task. While D3PM, and τ-LDR-0 use the same backbone transformer structure as USD3 and contain the same number of parameters in the transformer, they require a passing of transition matrices, which incur additional memory and slow down the training process. RDM is similar to USD3-CE and USD3* in memory calculation and runtime. SDDM requires a special architecture and considerably larger memory during training. SEDD requires significantly less memory than USD3 (continuous version), and training time is almost equivalent to the discrete-version USD3-CE and USD3*.

5.6 Sampling Time Discrete-time and Continuous-time Discrete Diffusion

In Table 8, we have calculated the time required for baselines and our models to sample 50,000 VQ-encoded images with 1000 timesteps on a single A6000 GPU. In addition, we calculate the time required when USD3 and τ-LDR need MCMC corrector steps (=5) in

sampling. We compare our results in the following tables. In terms of sampling time, RDM yields the most sampling time due to its re-routing sampling scheme. SEDD and τ-LDR are similar in sampling time, due to the tau-leaping scheme. USD3 falls short of SDDM, which requires a special NN architecture and different code base (jaxlib) than other methods. But when considering MCMC sampling, USD3 brings faster sampling time than tau-leaping in τ-LDR, while allowing sampling for both continuous-time and discrete-time models.

Table 8: Sample times and MCMC sampling steps for different models.

Model	**Sample Time**	**MCMC Sampling Step=5**
RDM	∼23 hours	-
D3PM	∼8 hours	-
τ-LDR	∼9 hours	∼2 days, 20 hours
SDDM	∼7 hours	-
SEDD	∼8 hours	-
USD3 (continuous version)	∼8 hours	∼2 days, 12 hours
USD3 (discrete version)	∼8 hours	∼2 days, 12 hours

6. Practitioners' Guidelines

Noise Sampling: Due to its masking nature and similarity to autoregressive models, absorbing noise has become a mainstream approach in language modeling. Many studies have focused on simplifying the use of absorbing noise, as it allows for a more straightforward derivation of forward conditional probabilities and leads to simpler parameterizations (Sahoo et al., 2024; Lou et al., 2024; Zhang et al., 2025). In the context of image generation in the VQ space, we observe that absorbing states do not necessarily outperform uniform noise. However, Shi et al. (2025) show that autoregressive models incorporating absorbing noise can produce high-quality texts. We hypothesize that absorbing noise works better for naturally discrete space given its sparsity in the manifold, while for other domains like images whose manifold is much denser, uniform noise could work better. What's more, for domains like graphs, Zhao et al. (2024) theoretically shows that absorbing noise cannot have enough expressivity for graph transformation, while uniform noise helps on breaking structural symmetries that solves the expressivity limitation of generation. Moreover, if the marginal distributions of nodes and edges can be effectively learned, the noise can be tailored to reflect underlying graph properties, resulting in generated graphs that better align with the original distributions.

Noise Schedules Noise scheduling is another crucial factor in diffusion training. For instance, Chen (2023b) show that, at the same noise level, diffusion tasks become easier as image size increases. In our experiments, we adopt the same noise schedulers as recommended in previous literature, for a fair comparison. However, for modalities such as text, which reside in high-dimensional and sparse vector spaces, a more adaptive or complex noise scheduler should be considered to prevent degeneration (Gao et al., 2024). In general, we also found that the noise schedules have relative smaller impact on performance when doing ablation, that simple linear or cosine schedule works well. This probably aligns with flow matching (Gat et al., 2024) .

Training Configurations Important configurations include loss, architectures, and training hyperparameters. For combining CE loss and negative ELBO, we find that setting coefficients of CE to 0.001 yields better generation performance, as illustrated in our previous ablation study. During training, we find the Diffusion Transformer (Peebles and Xie, 2023) consistently outperforms the standard Transformer architecture across a range of tasks. This performance improvement can be attributed to the integration of diffusion-based mechanisms, such as timestep embeddings, which enhance the model's ability to capture complex dependencies and enhance diffusion generation. Additionally, training stability and convergence are significantly improved by employing an exponential learning rate decay alongside cosine learning rate scheduler and linear warmup. While EMA parameters can be difficult to tune and have a significant impact on image synthesis quality, we heuristically set $\beta = 0.999$ without further tuning. Karras et al. (2024) provides further instructions on a post-hoc approach to tune EMA parameters during diffusion training.

7. Limitations and Future Work

There are several limitations in our current work that warrant further refinement:

Scalability Limitations. When the number of classes (in the tokenizer space or corpus space) increases, our current simplifications become insufficient and require further adjustments. Several prior studies have employed simplifications in absorbing state diffusions (Sahoo et al., 2024; Shi et al., 2025; Lou et al., 2024), achieving improved results, particularly in the domain of text generation. These studies underscore the potential of diffusion models for both text and image domains. A promising direction for future work would be to focus on simplifying the process to accommodate uniform noise distributions or noise distributions specific to marginal distributions. For instance, Ou et al. (2025) demonstrates that absorbing noise is mathematically equivalent to the marginal distribution of clean data, scaled by time-dependent factors. This insight could be instrumental in enhancing the efficiency and applicability of diffusion models in various domains.

Theoretical Bound on Simplification of Loss Function. Proposition 3 presents a useful simplification of the discrete-time loss function. However, the theoretical error bounds associated with the simplified loss, particularly after removing the sensitive scale term in Eq. Eq. (14), as well as the guidance method for clipping coefficients, need more rigorous study. A deeper analysis is required to understand the impact of these modifications on the overall model performance and convergence properties, as well as to assess the trade-offs in terms of accuracy and computational efficiency.

Sampling Efficiency. While our work focuses on improving the accuracy of the sampling process, it should be noted that incorporating additional MCMC steps leads to slower sampling due to the increased number of steps required. An alternative approach that has gained attention in the literature involves efficient sampling by reducing the number of sampling steps, as demonstrated by Geng et al. (2025). We view this as a promising avenue for future exploration, by refining the noise generation process or designing novel, noise-

guided generation strategies to improve sampling efficiency without sacrificing performance.

8. Conclusion

We introduce two fundamental contributions for both discrete-time and continuous-time diffusion for categorical data. First, we presented extensive *mathematical simplifications for the loss functions*, including exact closed-form derivations as well as novel easy-to-optimize approximations. Second, we established a *mathematical unification* of discrete-time and continuous-time discrete diffusion, enabling mutual benefits. Our proposed approach USD3 for discrete diffusion achieved significant improvements on established image, text and music notes datasets, across a suite of generation quality metrics. Experimentally, we have conducted extensive experiments on diverse benchmarks to ablate design choices, analyze training mechanisms, investigate various loss terms and their effects on generalization performance and data quality.

References

W. J. Anderson. *Continuous-time Markov chains: An applications-oriented approach*. Springer Science & Business Media, 2012.

J. Austin, D. D. Johnson, J. Ho, D. Tarlow, and R. van den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021.

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

A. Campbell, J. Benton, V. De Bortoli, T. Rainforth, G. Deligiannidis, and A. Doucet. A continuous time framework for discrete denoising models. *Advances in Neural Information Processing Systems*, 35:28266–28279, 2022.

H. Chang, H. Zhang, L. Jiang, C. Liu, and W. T. Freeman. Maskgit: Masked generative image transformer, 2022.

T. Chen. On the importance of noise scheduling for diffusion models. *arXiv preprint arXiv:2301.10972*, 2023a.

T. Chen. On the importance of noise scheduling for diffusion models, 2023b. URL `https://arxiv.org/abs/2301.10972`.

A. Demirkaya, J. Chen, and S. Oymak. Exploring the role of loss functions in multiclass classification. In *2020 54th annual conference on information sciences and systems (ciss)*, pages 1–5. IEEE, 2020.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment, 2017.

P. Esser, R. Rombach, and B. Ommer. Taming transformers for high-resolution image synthesis, 2020.

Z. Gao, J. Guo, X. Tan, Y. Zhu, F. Zhang, J. Bian, and L. Xu. Empowering diffusion models on the embedding space for text generation, 2024. URL https://arxiv.org/abs/2212.09412.

I. Gat, T. Remez, N. Shaul, F. Kreuk, R. T. Q. Chen, G. Synnaeve, Y. Adi, and Y. Lipman. Discrete flow matching, 2024. URL https://arxiv.org/abs/2407.15595.

Z. Geng, M. Deng, X. Bai, J. Z. Kolter, and K. He. Mean flows for one-step generative modeling, 2025. URL https://arxiv.org/abs/2505.13447.

D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The journal of physical chemistry*, 81(25):2340–2361, 1977.

D. T. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *The Journal of chemical physics*, 115(4):1716–1733, 2001.

A. Graves, R. K. Srivastava, T. Atkinson, and F. Gomez. Bayesian flow networks, 2024. URL https://arxiv.org/abs/2308.07037.

J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020a.

J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020b.

J. Ho, W. Chan, C. Saharia, J. Whang, R. Gao, A. Gritsenko, D. P. Kingma, B. Poole, M. Norouzi, D. J. Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.

E. Hoogeboom, D. Nielsen, P. Jaini, P. Forré, and M. Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in Neural Information Processing Systems*, 34:12454–12465, 2021a.

E. Hoogeboom, D. Nielsen, P. Jaini, P. Forré, and M. Welling. Argmax flows and multinomial diffusion: Learning categorical distributions, 2021b. URL https://arxiv.org/abs/2102.05379.

E. Hoogeboom, A. A. Gritsenko, J. Bastings, B. Poole, R. van den Berg, and T. Salimans. Autoregressive diffusion models, 2022. URL https://arxiv.org/abs/2110.02037.

A. Hyvärinen. Some extensions of score matching. *Computational statistics & data analysis*, 51(5):2499–2512, 2007.

S. Kang and K. Cho. Conditional molecular design with deep generative models. *Journal of chemical information and modeling*, 59(1):43–52, 2018.

T. Karras, M. Aittala, T. Aila, and S. Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35:26565–26577, 2022.

T. Karras, M. Aittala, J. Lehtinen, J. Hellsten, T. Aila, and S. Laine. Analyzing and improving the training dynamics of diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24174–24184, June 2024.

A. Kolmogoroff. Über die analytischen methoden in der wahrscheinlichkeitsrechnung. *Mathematische Annalen*, 104:415–458, 1931.

X. Li, J. Thickstun, I. Gulrajani, P. S. Liang, and T. B. Hashimoto. Diffusion-lm improves controllable text generation. *Advances in Neural Information Processing Systems*, 35: 4328–4343, 2022.

Y. Li, J. Pei, and L. Lai. Structure-based de novo drug design using 3d deep generative models. *Chemical science*, 12(41):13664–13675, 2021.

A. Lou, C. Meng, and S. Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.16834*, 2024.

S. Lyu. Interpretation and generalization of score matching. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 359–366, 2009.

OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023. URL `https://arxiv.org/abs/2303.08774`.

J. Ou, S. Nie, K. Xue, F. Zhu, J. Sun, Z. Li, and C. Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data, 2025. URL `https://arxiv.org/abs/2406.03736`.

W. Peebles and S. Xie. Scalable diffusion models with transformers, 2023. URL `https://arxiv.org/abs/2212.09748`.

C. Raffel. *Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching*. PhD thesis, Columbia University, 2016.

A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.

A. Rindos, S. Woolet, I. Viniotis, and K. Trivedi. Exact methods for the transient analysis of nonhomogeneous continuous time markov chains. In *Computations with Markov Chains: Proceedings of the 2nd International Workshop on the Numerical Solution of Markov Chains*, pages 121–133. Springer, 1995.

S. S. Sahoo, M. Arriola, Y. Schiff, A. Gokaslan, E. Marroquin, J. T. Chiu, A. Rush, and V. Kuleshov. Simple and effective masked diffusion language models, 2024. URL https://arxiv.org/abs/2406.07524.

J. Shi, K. Han, Z. Wang, A. Doucet, and M. K. Titsias. Simplified and generalized masked diffusion for discrete data, 2025. URL https://arxiv.org/abs/2406.04329.

A. Shih, D. Sadigh, and S. Ermon. Training and inference on any-order autoregressive models the right way, 2022. URL https://arxiv.org/abs/2205.13554.

J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=St1giarCHLP.

Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=PxTIG12RRHS.

H. Sun, L. Yu, B. Dai, D. Schuurmans, and H. Dai. Score-based continuous-time discrete diffusion models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=BYWWwSY2G5s.

A. Tewari and P. L. Bartlett. On the consistency of multiclass classification methods. *Journal of Machine Learning Research*, 8(5), 2007.

D. Tran, K. Vafa, K. K. Agrawal, L. Dinh, and B. Poole. Discrete flows: Invertible generative models of discrete data, 2019. URL https://arxiv.org/abs/1905.10347.

C. Zhang, C. Zhang, M. Zhang, and I. S. Kweon. Text-to-image diffusion model in generative ai: A survey. *arXiv preprint arXiv:2303.07909*, 2023.

Y. Zhang, S. He, D. Levine, L. Zhao, D. Zhang, S. A. Rizvi, E. Zappala, R. Ying, and D. van Dijk. Non-markovian discrete diffusion with causal language models, 2025. URL https://arxiv.org/abs/2502.09767.

L. Zhao, X. Ding, and L. Akoglu. Pard: Permutation-invariant autoregressive diffusion for graph generation, 2024. URL https://arxiv.org/abs/2402.03687.

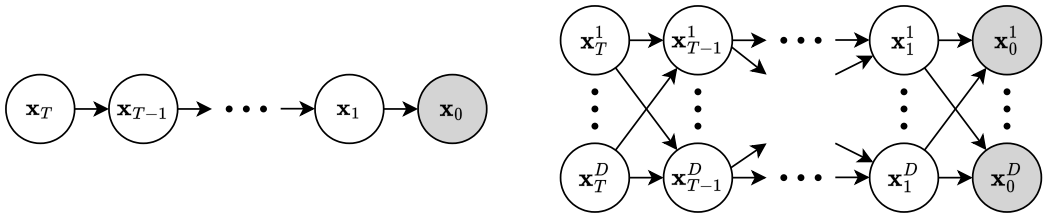
L. Zheng, J. Yuan, L. Yu, and L. Kong. A reparameterized discrete diffusion model for text generation. *arXiv preprint arXiv:2302.05737*, 2023.

Z. Ziegler and A. Rush. Latent normalizing flows for discrete sequences. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7673–7682. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/ziegler19a.html.

9. Appendix

9.1 Evidence Lower Bound Derivation

$$
\begin{aligned}
\log \int p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T} &\geq \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\big[\log p_\theta(\mathbf{x}_{0:T}) - \log q(\mathbf{x}_{1:T}|\mathbf{x}_0)\big] \\
&= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\big[\log p_\theta(\mathbf{x}_T) + \sum_{t=1}^{T} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}\big] \\
&= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\big[\log p_\theta(\mathbf{x}_T) + \log \frac{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)} + \sum_{t=2}^{T} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)}\big] \\
&= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\big[\log p_\theta(\mathbf{x}_T) + \log \frac{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)} + \sum_{t=2}^{T} \log \left(\frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} \cdot \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)}\right)\big] \\
&= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\big[\log p_\theta(\mathbf{x}_T) + \log \frac{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{q(\mathbf{x}_T|\mathbf{x}_0)} + \sum_{t=2}^{T} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}\big] \\
&= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\big[\log p_\theta(\mathbf{x}_0|\mathbf{x}_1) + \log \frac{p_\theta(\mathbf{x}_T)}{q(\mathbf{x}_T|\mathbf{x}_0)} - \sum_{t=2}^{T} \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}\big] \\
&= \underbrace{\mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)}\big[\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\big]}_{-\mathcal{L}_1(\theta)} - \sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\big[D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)\big]}_{\mathcal{L}_t(\theta)} - \text{const.} \quad (30)
\end{aligned}
$$



(a) Latent graphical model for diffusion (b) Generalize (a) to multi-element object

Figure 3: The graphical model diagram for both element-wise diffusion and multi-element diffusion.

9.2 Definition of nominal data

Nominal data, or categorical data, refers to data that can be categorized but not ranked or ordered. In mathematics, nominal data can be represented as a set of discrete categories or distinct labels without inherent ordering. As there is no ordering inside, operations of comparing categories is meaningless except checking equality. In the real world, most categories of data are nominal in nature, such as nationality, blood type, colors, among others.

9.3 Derivation of $q(x_{t-1}|x_t, x_0)$

First, let us define $\overline{Q}_{t|s} = Q_{s+1}...Q_t$. Note that $\overline{Q}_{t|0} = \overline{Q}_t$ and $\overline{Q}_{t|t-1} = Q_t$. Accordingly, we can derive the following two equalities.

$$q(\mathbf{x}_t|\mathbf{x}_s) = \text{Cat}(\mathbf{x}_t; \overline{Q}_{t|s}^\top \mathbf{x}_s) \tag{31}$$

$$\begin{aligned} q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0) &= \frac{q(\mathbf{x}_t|\mathbf{x}_s)q(\mathbf{x}_s|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} = \frac{\text{Cat}(\mathbf{x}_t; \overline{Q}_{t|s}^\top \mathbf{x}_s)\text{Cat}(\mathbf{x}_s; \overline{Q}_s^\top \mathbf{x}_0)}{\text{Cat}(\mathbf{x}_t; \overline{Q}_t^\top \mathbf{x}_0)} \\ &= \frac{\mathbf{x}_s^\top \overline{Q}_{t|s}\mathbf{x}_t \cdot \mathbf{x}_s^\top \overline{Q}_s^\top \mathbf{x}_0}{\mathbf{x}_t^\top \overline{Q}_t^\top \mathbf{x}_0} = \mathbf{x}_s^\top \frac{\overline{Q}_{t|s}\mathbf{x}_t \odot \overline{Q}_s^\top \mathbf{x}_0}{\mathbf{x}_t^\top \overline{Q}_t^\top \mathbf{x}_0} = \text{Cat}(\mathbf{x}_s; \frac{\overline{Q}_{t|s}\mathbf{x}_t \odot \overline{Q}_s^\top \mathbf{x}_0}{\mathbf{x}_t^\top \overline{Q}_t^\top \mathbf{x}_0}) \end{aligned} \tag{32}$$

Using the formulation of Q_t in Eq. (7), $\overline{Q}_{t|s}$ can be written as

$$\overline{Q}_{t|s} = \overline{\alpha}_{t|s} I + (1 - \overline{\alpha}_{t|s})\mathbf{1}\boldsymbol{m}^\top \tag{33}$$

where $\overline{\alpha}_{t|s} = \prod_{i=s+1}^{t} \alpha_i$. Note that $\overline{\alpha}_t = \overline{\alpha}_{t|s}\overline{\alpha}_s$.

Next, we can simplify Eq. (5) using the above formulations as

$$\begin{aligned} &\frac{\overline{Q}_{t|s}\mathbf{x}_t \odot \overline{Q}_s^\top \mathbf{x}_0}{\mathbf{x}_t^\top \overline{Q}_t^\top \mathbf{x}_0} = \frac{(\overline{\alpha}_{t|s}\mathbf{x}_t + (1 - \overline{\alpha}_{t|s})\langle \boldsymbol{m}, \mathbf{x}_t\rangle \mathbf{1}) \odot (\overline{\alpha}_s \mathbf{x}_0 + (1 - \overline{\alpha}_s)\boldsymbol{m})}{\overline{\alpha}_t \langle \mathbf{x}_t, \mathbf{x}_0\rangle + (1 - \overline{\alpha}_t)\langle \boldsymbol{m}, \mathbf{x}_t\rangle} \\ &= \frac{\overline{\alpha}_t \mathbf{x}_t \odot \mathbf{x}_0 + (\overline{\alpha}_s - \overline{\alpha}_t)\langle \boldsymbol{m}, \mathbf{x}_t\rangle \mathbf{x}_0 + (\overline{\alpha}_{t|s} - \overline{\alpha}_t)\mathbf{x}_t \odot \boldsymbol{m} + (1 - \overline{\alpha}_s)(1 - \overline{\alpha}_{t|s})\langle \boldsymbol{m}, \mathbf{x}_t\rangle \boldsymbol{m}}{\overline{\alpha}_t \langle \mathbf{x}_t, \mathbf{x}_0\rangle + (1 - \overline{\alpha}_t)\langle \boldsymbol{m}, \mathbf{x}_t\rangle} \end{aligned} \tag{34}$$

We can simplify the above equation further by considering two cases: (1) $\mathbf{x}_t = \mathbf{x}_0$ and (2) $\mathbf{x}_t \neq \mathbf{x}_0$.

(Case 1) When $\mathbf{x}_t = \mathbf{x}_0$, using the fact that both $\mathbf{x}_t$ and $\mathbf{x}_0$ are one-hot encoded, we observe

$$\begin{aligned} \text{Eq. (34)} &= \frac{\overline{\alpha}_t \mathbf{x}_t + (\overline{\alpha}_s - \overline{\alpha}_t)\langle \boldsymbol{m}, \mathbf{x}_t\rangle \mathbf{x}_t + (\overline{\alpha}_{t|s} - \overline{\alpha}_t)\langle \boldsymbol{m}, \mathbf{x}_t\rangle \mathbf{x}_t + (1 - \overline{\alpha}_s)(1 - \overline{\alpha}_{t|s})\langle \boldsymbol{m}, \mathbf{x}_t\rangle \boldsymbol{m}}{\overline{\alpha}_t + (1 - \overline{\alpha}_t)\langle \boldsymbol{m}, \mathbf{x}_t\rangle} \\ &= \frac{\overline{\alpha}_t + (\overline{\alpha}_{t|s} + \overline{\alpha}_s - 2\overline{\alpha}_t)\langle \boldsymbol{m}, \mathbf{x}_t\rangle}{\overline{\alpha}_t + (1 - \overline{\alpha}_t)\langle \boldsymbol{m}, \mathbf{x}_t\rangle}\mathbf{x}_t + \frac{(1 - \overline{\alpha}_s)(1 - \overline{\alpha}_{t|s})\langle \boldsymbol{m}, \mathbf{x}_t\rangle}{\overline{\alpha}_t + (1 - \overline{\alpha}_t)\langle \boldsymbol{m}, \mathbf{x}_t\rangle}\boldsymbol{m} \\ &= (1 - \lambda_{t|s}) \cdot \mathbf{x}_t + \lambda_{t|s} \cdot \boldsymbol{m} \ , \end{aligned} \tag{35}$$

where $\lambda_{t|s}$ is defined as

$$\lambda_{t|s} = \frac{(1 - \overline{\alpha}_s)(1 - \overline{\alpha}_{t|s})\langle \boldsymbol{m}, \mathbf{x}_t\rangle}{\overline{\alpha}_t + (1 - \overline{\alpha}_t)\langle \boldsymbol{m}, \mathbf{x}_t\rangle} \ . \tag{36}$$

(Case 2) When $\mathbf{x}_t \neq \mathbf{x}_0$, we can similarly derive

$$\begin{aligned} \text{Eq. (34)} &= \frac{(\overline{\alpha}_s - \overline{\alpha}_t)\langle \boldsymbol{m}, \mathbf{x}_t\rangle \mathbf{x}_0 + (\overline{\alpha}_{t|s} - \overline{\alpha}_t)\langle \boldsymbol{m}, \mathbf{x}_t\rangle \mathbf{x}_t + (1 - \overline{\alpha}_s)(1 - \overline{\alpha}_{t|s})\langle \boldsymbol{m}, \mathbf{x}_t\rangle \boldsymbol{m}}{(1 - \overline{\alpha}_t)\langle \boldsymbol{m}, \mathbf{x}_t\rangle} \\ &= \frac{\overline{\alpha}_s - \overline{\alpha}_t}{1 - \overline{\alpha}_t}\mathbf{x}_0 + \frac{1 - \overline{\alpha}_s}{1 - \overline{\alpha}_t}\overline{\alpha}_{t|s}\mathbf{x}_t + \frac{1 - \overline{\alpha}_s}{1 - \overline{\alpha}_t}(1 - \overline{\alpha}_{t|s})\boldsymbol{m} \\ &= (1 - \mu_{t|s}) \cdot \mathbf{x}_0 + \mu_{t|s}\overline{\alpha}_{t|s} \cdot \mathbf{x}_t + \mu_{t|s}(1 - \overline{\alpha}_{t|s}) \cdot \boldsymbol{m} \ , \end{aligned} \tag{37}$$

where $\mu_{t|s}$ is defined as

$$\mu_{t|s} = \frac{1 - \overline{\alpha}_s}{1 - \overline{\alpha}_t} \tag{38}$$

Combining the results from both cases together, we can write $q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0)$ in the following form.

$$q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0) = \begin{cases} \text{Cat}\Big(\mathbf{x}_s;\ (1-\lambda_{t|s})\cdot \mathbf{x}_t + \lambda_{t|s}\cdot \boldsymbol{m}\Big) & \text{when } \mathbf{x}_t = \mathbf{x}_0 \\ \text{Cat}\Big(\mathbf{x}_s;\ (1-\mu_{t|s})\cdot \mathbf{x}_0 + \mu_{t|s}\overline{\alpha}_{t|s}\cdot \mathbf{x}_t + \mu_{t|s}(1-\overline{\alpha}_{t|s})\cdot \boldsymbol{m}\Big) & \text{when } \mathbf{x}_t \neq \mathbf{x}_0 \end{cases} \tag{39}$$

9.4 Parameterization of $p_\theta(x_s|x_t)$

We first provide the formulation as follows.

$$p_\theta(\mathbf{x}_s|\mathbf{x}_t) = \sum_{\mathbf{x}_0} q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0) p_\theta(\mathbf{x}_0|\mathbf{x}_t) \tag{40}$$

Using $q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0)$ in Eq. (9) and $p_\theta(\mathbf{x}_0|\mathbf{x}_t) = \text{Cat}(\mathbf{x}_0; f_t^\theta(\mathbf{x}_t))$, we can expand it as

$$\begin{aligned}
p_\theta(\mathbf{x}_s|\mathbf{x}_t) =& q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_t) p_\theta(\mathbf{x}_t|\mathbf{x}_t) + \sum_{\mathbf{x}\neq\mathbf{x}_t} q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}) p_\theta(\mathbf{x}|\mathbf{x}_t) \\
=& \mathbf{x}_s^\top \Big((1-\lambda_{t|s})\mathbf{x}_t + \lambda_{t|s}\boldsymbol{m}\Big)\mathbf{x}_t^\top f_t^\theta(\mathbf{x}_t) \\
&+ \sum_{\mathbf{x}\neq\mathbf{x}_t} \mathbf{x}_s^T \Big((1-\mu_{t|s})\mathbf{x} + \mu_{t|s}\overline{\alpha}_{t|s}\mathbf{x}_t + \mu_{t|s}(1-\overline{\alpha}_{t|s})\boldsymbol{m}\Big)\mathbf{x}^\top f_t^\theta(\mathbf{x}_t) \\
=& \mathbf{x}_s^\top \Big[((1-\lambda_{t|s})\mathbf{x}_t + \lambda_{t|s}\boldsymbol{m})\mathbf{x}_t^\top f_t^\theta(\mathbf{x}_t) + (1-\mu_{t|s})(\sum_{\mathbf{x}\neq\mathbf{x}_t} \mathbf{x}\mathbf{x}^\top) f_t^\theta(\mathbf{x}_t) \\
&+ (\mu_{t|s}\overline{\alpha}_{t|s}\mathbf{x}_t + \mu_{t|s}(1-\overline{\alpha}_{t|s})\boldsymbol{m})(\sum_{\mathbf{x}\neq\mathbf{x}_t} \mathbf{x})^\top f_t^\theta(\mathbf{x}_t)\Big] \\
=& \mathbf{x}_s^\top \Big[((1-\lambda_{t|s})\mathbf{x}_t + \lambda_{t|s}\boldsymbol{m})\mathbf{x}_t^\top f_t^\theta(\mathbf{x}_t) + (1-\mu_{t|s})(I - \mathbf{x}_t\mathbf{x}_t^\top) f_t^\theta(\mathbf{x}_t) \\
&+ (\mu_{t|s}\overline{\alpha}_{t|s}\mathbf{x}_t + \mu_{t|s}(1-\overline{\alpha}_{t|s})\boldsymbol{m})(\mathbf{1} - \mathbf{x}_t)^\top f_t^\theta(\mathbf{x}_t)\Big] \\
=& \mathbf{x}_s^\top \Big[(1-\lambda_{t|s})\mathbf{x}_t^\top f_t^\theta(\mathbf{x}_t)\mathbf{x}_t + \lambda_{t|s}\mathbf{x}_t^\top f_t^\theta(\mathbf{x}_t)\boldsymbol{m} + (1-\mu_{t|s})(f_t^\theta(\mathbf{x}_t) - \mathbf{x}_t^\top f_t^\theta(\mathbf{x}_t)\mathbf{x}_t) \\
&+ (\mu_{t|s}\overline{\alpha}_{t|s}\mathbf{x}_t + \mu_{t|s}(1-\overline{\alpha}_{t|s})\boldsymbol{m})(1 - \mathbf{x}_t^T f_t^\theta(\mathbf{x}_t))\Big] \\
=& \mathbf{x}_s^\top \Big[(1-\mu_{t|s})\cdot f_t^\theta(\mathbf{x}_t) + \Big((\mu_{t|s} - \lambda_{t|s} - \mu_{t|s}\overline{\alpha}_{t|s})\mathbf{x}_t^\top f_t^\theta(\mathbf{x}_t) + \mu_{t|s}\overline{\alpha}_{t|s}\Big)\cdot \mathbf{x}_t \\
&+ \Big(-(\mu_{t|s} - \lambda_{t|s} - \mu_{t|s}\overline{\alpha}_{t|s})\mathbf{x}_t^\top f_t^\theta(\mathbf{x}_t) + \mu_{t|s}(1-\overline{\alpha}_{t|s})\Big)\cdot \boldsymbol{m}\Big] \\
=& \quad \text{Cat}\Big(\mathbf{x}_s;\ (1-\mu_{t|s})\cdot f_t^\theta(\mathbf{x}_t) + (\mu_{t|s}\overline{\alpha}_{t|s} + \gamma_{t|s}^\theta)\cdot \mathbf{x}_t + (\mu_{t|s}(1-\overline{\alpha}_{t|s}) - \gamma_{t|s}^\theta)\cdot \boldsymbol{m}\Big)
\end{aligned} \tag{41}$$

9.5 Proof of Proposition 3

As $q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0)$ has two different categorical distributions based on whether $\mathbf{x}_t$ is equal to $\mathbf{x}_0$, we prove the Proposition 3 based on two cases.

Case 1: $\mathbf{x}_t = \mathbf{x}_0$. In this case, let $h_t^\theta := f_t^\theta(\boldsymbol{x}_t) - \boldsymbol{x}_0$, now replace f_t^θ with g_t^θ in Eq. (2) and Eq. (10), we get

$$\gamma_{t|s}^\theta = (\mu_{t|s} - \lambda_{t|s} - \mu_{t|s}\overline{\alpha}_{t|s})\langle h_t^\theta + \boldsymbol{x}_0, \boldsymbol{x}_t\rangle = (\mu_{t|s} - \lambda_{t|s} - \mu_{t|s}\overline{\alpha}_{t|s})\langle h_t^\theta, \boldsymbol{x}_t\rangle + (\mu_{t|s} - \lambda_{t|s} - \mu_{t|s}\overline{\alpha}_{t|s}) \tag{42}$$

Taking the formulation into Eq. (10) and cancel out terms, we can simplify $p_\theta(\mathbf{x}_s|\mathbf{x}_t)$ as

$$\begin{aligned} p_\theta(\mathbf{x}_s|\mathbf{x}_t) &= (1-\mu_{t|s})h_t^\theta + (1-\lambda_{t|s})\boldsymbol{x}_t + \lambda_{t|s}\boldsymbol{m} + (\mu_{t|s} - \lambda_{t|s} - \mu_{t|s}\overline{\alpha}_{t|s})\langle h_t^\theta, \boldsymbol{x}_t\rangle(\boldsymbol{x}_t - \boldsymbol{m}) \\ &= q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0) + (1-\mu_{t|s})\Big[h_t^\theta + \frac{\mu_{t|s} - \lambda_{t|s} - \mu_{t|s}\overline{\alpha}_{t|s}}{1-\mu_{t|s}}\langle h_t^\theta, \boldsymbol{x}_t\rangle(\boldsymbol{x}_t - \boldsymbol{m})\Big] \\ &= q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0) + (1-\mu_{t|s})\Big[h_t^\theta + \phi_{t|s}\langle h_t^\theta, \boldsymbol{x}_t\rangle(\boldsymbol{x}_t - \boldsymbol{m})\Big] \end{aligned} \tag{43}$$

Case 2: $\mathbf{x}_t \neq \mathbf{x}_0$**.** Based on Eq. (9) and Eq. (10), it's easy to find that

$$\begin{aligned} p_\theta(\mathbf{x}_s|\mathbf{x}_t) &= (1-\mu_{t|s})(f_t^\theta(\boldsymbol{x}_t) - \boldsymbol{x}_0) + \gamma_{t|s}^\theta(\boldsymbol{x}_t - \boldsymbol{m}) + (1-\mu_{t|s})\boldsymbol{x}_0 + \mu_{t|s}\bar{\alpha}_{t|s}\boldsymbol{x}_t + \mu_{t|s}(1-\bar{\alpha}_{t|s})\boldsymbol{m} \\ &= (1-\mu_{t|s})(f_t^\theta(\boldsymbol{x}_t) - \boldsymbol{x}_0) + \gamma_{t|s}^\theta(\boldsymbol{x}_t - \boldsymbol{m}) + q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0) \\ &= q(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0) + (1-\mu_{t|s})\Big[f_t^\theta(\boldsymbol{x}_t) - \boldsymbol{x}_0 + \frac{\gamma_{t|s}^\theta}{(1-\mu_{t|s})}(\boldsymbol{x}_t - \boldsymbol{m})\Big] \end{aligned} \tag{44}$$

Taking the definition of $\gamma_{t|s}^\theta$ in Eq. (2) and using the fact $< \boldsymbol{x}_0, \boldsymbol{x}_t >= 0$ we get the formulation in proposition 3.

For both cases, we proved that the formulation in proposition 3 is correct.

9.6 Proof of KL divergence approximation

Assume that $p(\boldsymbol{x}) = q(\boldsymbol{x}) + \triangle p(\boldsymbol{x})$, and both $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$ are valid categorical distributions. Then

$$D_{\mathrm{KL}}(q(\boldsymbol{x})\|p(\boldsymbol{x})) = -\sum_{\boldsymbol{x}} q(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} = -\sum_{\boldsymbol{x}} q(\boldsymbol{x}) \log(1 + \frac{\triangle p(\boldsymbol{x})}{q(\boldsymbol{x})}) \tag{45}$$

By Taylor expansion, $\log(1+x) \approx x - \frac{x^2}{2}$, we ignore larger than 2 order terms. Then apply this we get

$$\begin{aligned} D_{\mathrm{KL}}(q(\boldsymbol{x})\|p(\boldsymbol{x})) &= -\sum_{\boldsymbol{x}} q(\boldsymbol{x})[\frac{\triangle p(\boldsymbol{x})}{q(\boldsymbol{x})} - (\frac{\triangle p(\boldsymbol{x})}{q(\boldsymbol{x})})^2] \\ &= -\sum_{\boldsymbol{x}} \triangle p(\boldsymbol{x}) + \sum_{\boldsymbol{x}} \frac{\triangle p(\boldsymbol{x})^2}{q(\boldsymbol{x})} \\ &= \sum_{\boldsymbol{x}} \frac{\triangle p(\boldsymbol{x})^2}{q(\boldsymbol{x})} \end{aligned} \tag{46}$$

Change probabilities q and p to $q_(\mathbf{x}_s|\mathbf{x}_t, \mathbf{x}_0)$ and $p_\theta(\mathbf{x}_s|\mathbf{x}_t)$ we get the targeted formulation.

9.7 ELBO with partial time steps

Assume that we only observe $\mathbf{x}_t$, $\mathbf{x}_s$, and $\mathbf{x}_0$, and assume that we have learned the prior $p_\theta(\mathbf{x}_t)$.

$$\begin{aligned}
\log p_\theta(\mathbf{x}_0) &= \log \mathbb{E}_{q(\mathbf{x}_t,\mathbf{x}_s|\mathbf{x}_0)} \frac{p_\theta(\mathbf{x}_t,\mathbf{x}_s,\mathbf{x}_0)}{q(\mathbf{x}_t,\mathbf{x}_s|\mathbf{x}_0)} \geq \mathbb{E}_{q(\mathbf{x}_t,\mathbf{x}_s|\mathbf{x}_0)} \log \frac{p_\theta(\mathbf{x}_t,\mathbf{x}_s,\mathbf{x}_0)}{q(\mathbf{x}_t,\mathbf{x}_s|\mathbf{x}_0)} \\
&= \mathbb{E}_{q(\mathbf{x}_t,\mathbf{x}_s|\mathbf{x}_0)} [\log \frac{p_\theta(\mathbf{x}_t) p_\theta(\mathbf{x}_s|\mathbf{x}_t) p_\theta(\mathbf{x}_0|\mathbf{x}_s)}{q(\mathbf{x}_t|\mathbf{x}_0) q(\mathbf{x}_s|\mathbf{x}_t,\mathbf{x}_0)}] \\
&= \mathbb{E}_{q(\mathbf{x}_t,\mathbf{x}_s|\mathbf{x}_0)} \Big[\log \frac{p_\theta(\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_0)} + \log p_\theta(\mathbf{x}_t) + \log \frac{p_\theta(\mathbf{x}_s|\mathbf{x}_t)}{q(\mathbf{x}_s|\mathbf{x}_t,\mathbf{x}_0)} \Big] \\
&= -D_{\mathrm{KL}}[q(\mathbf{x}_t|\mathbf{x}_0) \| p_\theta(\mathbf{x}_t)] + \mathbb{E}_{q(\mathbf{x}_s|\mathbf{x}_0)} \log p_\theta(\mathbf{x}_s|\mathbf{x}_0) - D_{\mathrm{KL}}[q(\mathbf{x}_s|\mathbf{x}_t,\mathbf{x}_0) \| p_\theta(\mathbf{x}_s|\mathbf{x}_t)]
\end{aligned} \tag{47}$$

9.8 Reparameterization Form for Sampling

In practice, we need to sample $\mathbf{x}_{t|0} \sim q(\mathbf{x}_t|\mathbf{x}_0)$ for training and $\mathbf{x}_{s|t} \sim p_\theta(\mathbf{x}_s|\mathbf{x}_t)$ for generation (backward denoising). In what follows, we provide the reparameterization form for these to facilitate fast sampling. Given $q(\mathbf{x}_t|\mathbf{x}_0) = \mathrm{Cat}(\mathbf{x}_t; \overline{\alpha}_t \mathbf{x}_0 + (1-\overline{\alpha}_t)\boldsymbol{m})$ and $p_\theta(\mathbf{x}_s|\mathbf{x}_t)$ as in Eq. (10), we can rewrite the corresponding variables as

$$\mathbf{x}_{t|0} = \delta_{1,\mathbf{b}_t} \mathbf{x}_0 + (1-\delta_{1,\mathbf{b}_t}) \mathbf{m}_0, \text{ where } \mathbf{b}_t \sim \mathrm{Bernoulli}(\overline{\alpha}_t) \tag{48}$$

$$\mathbf{x}_{s|t} = \delta_{1,\mathbf{b}_{s|t}} \tilde{\mathbf{x}}_{0|t} \ + \ \delta_{2,\mathbf{b}_{s|t}} \mathbf{x}_t \ + \ \delta_{3,\mathbf{b}_{s|t}} \mathbf{m}_0, \tag{49}$$

where $\tilde{\mathbf{x}}_{0|t} \sim \mathrm{Cat}(f_t^\theta(\mathbf{x}_t))$ and $\mathbf{b}_{s|t} \sim \mathrm{Cat}(\cdot; [1-\mu_{t|s}, \mu_{t|s}\overline{\alpha}_{t|s} + \gamma_{t|s}^\theta, \ \mu_{t|s}(1-\overline{\alpha}_{t|s}) - \gamma_{t|s}^\theta])$.

Eq. (48) and Eq. (49) essentially show that the sampling process can be divided into two steps: first, sample the branch indicator $\mathbf{b}_t$ (or $\mathbf{b}_{s|t}$), and then sample from the categorical distribution of that branch, i.e. $\mathbf{x}_t$, $\mathbf{m}_0$, or $\tilde{\mathbf{x}}_{0|t}$. Moreover, the three terms in Eq. (49) highlight that the denoising step of generating $\mathbf{x}_s$ from $\mathbf{x}_t$ essentially draws samples via three levers: (1) use the predicted sample $\mathbf{x}_0$ from the trained network $f_t^\theta(\mathbf{x}_t)$ directly, (2) keep it unchanged as $\mathbf{x}_t$, or (3) roll it back to noise $\mathbf{m}_0$, offering an intuitive understanding.

9.9 Discrete-Time Multi-element Object Extension

We first extend $q(\mathbf{x}_t|\mathbf{x}_0)$, $q(\mathbf{x}_s|\mathbf{x}_t,\mathbf{x}_0)$, and $p_\theta(\mathbf{x}_s|\mathbf{x}_t)$ to multi-element object, and then present the negative ELBO loss extension. First, for the forward process conditional on $\mathbf{x}_0$, each element $\mathbf{x}_t^d$ of the multi-element object $\mathbf{x}_t^{1:D}$ has its own diffusion process without interactions with others. Formally,

$$q(\mathbf{x}_t^{1:D}|\mathbf{x}_0^{1:D}) = \prod_{d=1}^{D} q(\mathbf{x}_t^d|\mathbf{x}_0^d) \quad \text{, and} \quad q(\mathbf{x}_s^{1:D}|\mathbf{x}_t^{1:D},\mathbf{x}_0^{1:D}) = \prod_{d=1}^{D} q(\mathbf{x}_s^d|\mathbf{x}_t^d,\mathbf{x}_0^d) \ . \tag{50}$$

The corresponding forward reparameterization form for generating $\mathbf{x}_{t|0}^{1:D}$ can be updated as

$$\mathbf{x}_{t|0}^{1:D} = \delta_{1,\mathbf{b}_t^{1:D}}^{1:D} \odot \mathbf{x}_0^{1:D} + (1 - \delta_{1,\mathbf{b}_t^{1:D}}^{1:D}) \odot \mathbf{m}_0^{1:D} \quad , \tag{51}$$

where $\delta_{1,\mathbf{b}_t^{1:D}}^{1:D}$ is a D-dimensional vector with d-th element being $\delta_{1,\mathbf{b}_t^d}$ and $\mathbf{b}_t^d \sim \mathrm{Bernoulli}(\overline{\alpha}_t)$.

For the backward process, all elements' transition processes are coupled together, as shown in the graphical model in Appendix Fig. 3(b). We start with the parameterization

$p_\theta(\mathbf{x}_0^{1:D}|\mathbf{x}_t^{1:D})$, with the form

$$p_\theta(\mathbf{x}_0^{1:D}|\mathbf{x}_t^{1:D}) = \prod_{d=1}^{D} p_\theta(\mathbf{x}_0^d|\mathbf{x}_t^{1:D}) \quad \text{with} \quad p_\theta(\mathbf{x}_0^d|\mathbf{x}_t^{1:D}) = \mathrm{Cat}(\mathbf{x}_0^d|f_t^{\theta,d}(\mathbf{x}_t^{1:D})) \tag{52}$$

Then, $p_\theta(\mathbf{x}_s^{1:D}|\mathbf{x}_t^{1:D}) = \prod_{d=1}^{D} p_\theta(\mathbf{x}_s^d|\mathbf{x}_t^{1:D})$, with the component $p_\theta(\mathbf{x}_s^d|\mathbf{x}_t^{1:D})$ has the form

$$\mathrm{Cat}\Big(\mathbf{x}_s^d;\ (1-\mu_{t|s})f_t^{\theta,d}(\mathbf{x}_t^{1:D}) + (\mu_{t|s}\overline{\alpha}_{t|s} + \gamma_{t|s}^{\theta,d})\mathbf{x}_t^d + (\mu_{t|s}(1-\overline{\alpha}_{t|s}) - \gamma_{t|s}^{\theta,d})\boldsymbol{m}^d\Big)\ , \tag{53}$$

$$\text{where } \gamma_{t|s}^{\theta,d} = (\mu_{t|s} - \lambda_{t|s}^d - \mu_{t|s}\overline{\alpha}_{t|s})\langle f_t^{\theta,d}(\mathbf{x}_t^{1:D}), \mathbf{x}_t^d\rangle,\ \lambda_{t|s}^d = \frac{(1-\overline{\alpha}_s)(1-\overline{\alpha}_{t|s})\langle \boldsymbol{m}^d, \mathbf{x}_t^d\rangle}{\overline{\alpha}_t + (1-\overline{\alpha}_t)\langle \boldsymbol{m}^d, \mathbf{x}_t^d\rangle}\ .$$

Similarly, the reparameterization form can be expressed as

$$\mathbf{x}_{s|t}^{1:D} = \delta_{1,\mathbf{b}_{s|t}^{1:D}}^{1:D} \odot \tilde{\mathbf{x}}_{0|t}^{1:D} \ +\ \delta_{2,\mathbf{b}_{s|t}^{1:D}}^{1:D} \odot \mathbf{x}_t^{1:D} \ +\ \delta_{3,\mathbf{b}_{s|t}^{1:D}}^{1:D} \odot \mathbf{m}_0^{1:D}\ , \tag{54}$$

where $\tilde{\mathbf{x}}_{0|t}^d \sim \mathrm{Cat}(f_t^{\theta,d}(\mathbf{x}_t^{1:D}))$ and $\mathbf{b}_{s|t}^d \sim \mathrm{Cat}(1-\mu_{t|s},\ \mu_{t|s}\overline{\alpha}_{t|s} + \gamma_{t|s}^{\theta,d},\ \mu_{t|s}(1-\overline{\alpha}_{t|s}) - \gamma_{t|s}^{\theta,d})$.

Finally, the following reformulate the loss functions $\mathcal{L}_t(\theta)$ for multi-element objects, where we drop the constant term for simplicity.

$$\begin{aligned}\mathcal{L}_t^{1:D}(\theta) =& \mathbb{E}_{q(\mathbf{x}_t^{1:D}|\mathbf{x}_0^{1:D})} \sum_{d=1}^{D} \Big[\delta_{\mathbf{x}_t^d,\mathbf{x}_0^d} \mathbb{E}_{q(\mathbf{x}_{t-1}^d|\mathbf{x}_t^d=\mathbf{x}_0^d)}[-\log p_\theta(\mathbf{x}_{t-1}^d|\mathbf{x}_t^{1:D})] + \\ & (1-\delta_{\mathbf{x}_t^d,\mathbf{x}_0^d}) \mathbb{E}_{q(\mathbf{x}_{t-1}^d|\mathbf{x}_t^d\neq\mathbf{x}_0^d)}[-\log p_\theta(\mathbf{x}_{t-1}^d|\mathbf{x}_t^{1:D})]\Big]\end{aligned} \tag{55}$$

9.10 CTMC introduction

The CTMC process can go with either increasing t or decreasing t, and we use increasing t as the default direction to introduce CTMC. For any CTMC, its *transition rate matrix* R_t with $[R_t]_{ij} = r_t(\boldsymbol{e}_j|\boldsymbol{e}_i)$ fully determines the underlying stochastic process. CTMC is categorized into time-homogeneous and time-*in*homogeneous based on whether R_t is static with respect to t. In this paper we work with time-inhomogeneous CTMC.

Based on the definition in Eq. (17), we can derive some properties as follows

$$\forall \boldsymbol{x}, r_t(\boldsymbol{x}|\boldsymbol{x}) \leq 0; \qquad \forall \boldsymbol{y} \neq \boldsymbol{x}, r_t(\boldsymbol{y}|\boldsymbol{x}) \geq 0; \qquad \forall \boldsymbol{x}, \sum_{\boldsymbol{y}} r_t(\boldsymbol{y}|\boldsymbol{x}) = 0 \tag{56}$$

$$q_{t|t-\triangle t}(\boldsymbol{y}|\boldsymbol{x}) = \delta_{\boldsymbol{x},\boldsymbol{y}} + r_t(\boldsymbol{y}|\boldsymbol{x})\triangle t + o(\triangle t) \tag{57}$$

Eq. (56) shows the properties of transition rate, which imply $r_t(\boldsymbol{x}|\boldsymbol{x}) = -\sum_{\boldsymbol{y}\neq\boldsymbol{x}} r_t(\boldsymbol{y}|\boldsymbol{x})$. Eq. (57) characterizes the infinitesimal change of transition probability, and can be used to derive the relationship between transition matrix $\overline{Q}_{t|s}$ and transition rate matrix R_t. Formally, a CTMC's transition probabilities satisfies the Kolmogorov forward and backward equations (Kolmogoroff, 1931) :

$$\frac{d}{dt}q_{t|s}(\boldsymbol{y}|\boldsymbol{x}) = \sum_{\mathbf{z}} q_{t|s}(\boldsymbol{z}|\boldsymbol{x}) r_t(\boldsymbol{y}|\boldsymbol{z}) \quad \text{or} \quad \frac{d}{dt}\overline{Q}_{t|s} = \overline{Q}_{t|s} R_t \qquad \text{Kolmogorov Forward} \tag{58}$$

$$\frac{d}{ds}q_{t|s}(\boldsymbol{y}|\boldsymbol{x}) = -\sum_{\boldsymbol{z}} r_s(\boldsymbol{z}|\boldsymbol{x}) q_{t|s}(\boldsymbol{y}|\boldsymbol{z}) \quad \text{or} \quad \frac{d}{ds}\overline{Q}_{t|s} = -R_s \overline{Q}_{t|s} \quad \text{Kolmogorov Backward} \tag{59}$$

The above equations are Ordinary Different Equations (ODEs) and have unique solutions. Rindos et al. (1995) mentioned that when R_{t_1} and R_{t_2} commute (i.e. $R_{t_1}R_{t_2} = R_{t_2}R_{t_1}$) for

any t_1, t_2, the transition probability matrix can be written as

$$\overline{Q}_{t|s} = \exp\Big(\int_s^t R_a da\Big) \tag{60}$$

where $\exp(M) := \sum_{k=0}^{\infty} \frac{M^k}{k!}$ is the matrix exponential operation. The commutative property of R_t can be achieved by choosing $R_t = \beta(t)R_b$ where $R_b \in \mathbb{R}^{K\times K}$ is a time-independent base transition rate matrix satisfying properties in Eq. (56) and $\beta(t)$ is a positive scalar dependent on time. Now assume R_b is diagonalizable and $R_b = U\Sigma U^{-1}$ by eigen-decomposition, then we can simplify Eq. (60) as

$$\overline{Q}_{t|s} = \exp\Big(\int_s^t \beta(t)R_b da\Big) = \exp(\overline{\beta}_{t|s}R_b) = \exp(\overline{\beta}_{t|s}U\Sigma U^{-1}) = U\exp(\overline{\beta}_{t|s}\Sigma)U^{-1} \tag{61}$$

where $\overline{\beta}_{t|s} := \int_s^t \beta(a)da$.

9.11 Derivation of Eq. (18)

$$\begin{aligned}\overline{Q}_{t|s} = \exp(\overline{\beta}_{t|s}R_b) &= I + \sum_{k=1}^{\infty}\frac{(-\overline{\beta}_{t|s})^k(-R_b)^k}{k!} = I - (\sum_{k=1}^{\infty}\frac{(-\overline{\beta}_{t|s})^k}{k!})R_b \\ &= I - (e^{-\overline{\beta}_{t|s}} - 1)R_b = e^{-\overline{\beta}_{t|s}}I + (1 - e^{-\overline{\beta}_{t|s}})\mathbf{1}\boldsymbol{m}^\top\end{aligned} \tag{62}$$

9.12 Derivation of Eq. (25)

With the help of Eq. (19), we first show that g_t^θ can be simplified as follows.

$$\begin{aligned}g_t^\theta(\boldsymbol{z}|\boldsymbol{x}) &= \frac{1}{\langle \boldsymbol{x}, \boldsymbol{m}\rangle}\Big[\langle \boldsymbol{z}, \boldsymbol{m}\rangle + \frac{\overline{\alpha}_{t|0}}{1-\overline{\alpha}_{t|0}}p_{0|t}^\theta(\boldsymbol{z}|\boldsymbol{x}) - \frac{\overline{\alpha}_{t|0}\langle \boldsymbol{z}, \boldsymbol{m}\rangle}{\overline{\alpha}_{t|0} + (1-\overline{\alpha}_{t|0})\langle \boldsymbol{x}, \boldsymbol{m}\rangle}p_{0|t}^\theta(\boldsymbol{x}|\boldsymbol{x})\Big] \\ &= \frac{1}{\langle \boldsymbol{x}, \boldsymbol{m}\rangle}\Big[(1 - \frac{\overline{\alpha}_{t|0}p_{0|t}^\theta(\boldsymbol{x}|\boldsymbol{x})}{\overline{\alpha}_{t|0} + (1-\overline{\alpha}_{t|0})\langle \boldsymbol{x}, \boldsymbol{m}\rangle})\langle \boldsymbol{z}, \boldsymbol{m}\rangle + \frac{\overline{\alpha}_{t|0}}{1-\overline{\alpha}_{t|0}}p_{0|t}^\theta(\boldsymbol{z}|\boldsymbol{x})\Big] \quad \forall \boldsymbol{z} \neq \boldsymbol{x}\end{aligned} \tag{63}$$

Proof

$$\frac{q_{t|0}(\boldsymbol{z}|\boldsymbol{x}_0)}{q_{t|0}(\boldsymbol{x}|\boldsymbol{x}_0)} = \frac{\boldsymbol{z}^\top \overline{Q}_{t|0}^\top \boldsymbol{x}_0}{\boldsymbol{x}^\top \overline{Q}_{t|0}^\top \boldsymbol{x}_0} = \frac{\overline{\alpha}_{t|0}\boldsymbol{z}^\top\boldsymbol{x}_0 + (1-\overline{\alpha}_{t|0})\langle \boldsymbol{z}, \boldsymbol{m}\rangle}{\overline{\alpha}_{t|0}\boldsymbol{x}^\top\boldsymbol{x}_0 + (1-\overline{\alpha}_{t|0})\langle \boldsymbol{x}, \boldsymbol{m}\rangle} \tag{64}$$

Then when $\boldsymbol{z} \neq \boldsymbol{x}$, we can write it as

$$\frac{q_{t|0}(\boldsymbol{z}|\boldsymbol{x}_0)}{q_{t|0}(\boldsymbol{x}|\boldsymbol{x}_0)} = \begin{cases} \frac{\langle \boldsymbol{z}, \boldsymbol{m}\rangle}{\langle \boldsymbol{x}, \boldsymbol{m}\rangle} & \text{if } \boldsymbol{x}_0 \neq \boldsymbol{x} \text{ and } \boldsymbol{x}_0 \neq \boldsymbol{z} \\ \frac{\frac{\overline{\alpha}_{t|0}}{1-\overline{\alpha}_{t|0}} + \langle \boldsymbol{z}, \boldsymbol{m}\rangle}{\langle \boldsymbol{x}, \boldsymbol{m}\rangle} & \text{if } \boldsymbol{x}_0 = \boldsymbol{z} \\ \frac{\langle \boldsymbol{z}, \boldsymbol{m}\rangle}{\frac{\overline{\alpha}_{t|0}}{1-\overline{\alpha}_{t|0}} + \langle \boldsymbol{x}, \boldsymbol{m}\rangle} & \text{if } \boldsymbol{x}_0 = \boldsymbol{x} \end{cases} \tag{65}$$

Hence,

$$
\begin{aligned}
g_t^\theta(\boldsymbol{z}|\boldsymbol{x}) &= \sum_{\boldsymbol{x}_0} \frac{q_{t|0}(\boldsymbol{z}|\boldsymbol{x}_0)}{q_{t|0}(\boldsymbol{x}|\boldsymbol{x}_0)} p_{0|t}^\theta(\boldsymbol{x}_0|\boldsymbol{x}) \\
&= \frac{q_{t|0}(\boldsymbol{z}|\boldsymbol{x})}{q_{t|0}(\boldsymbol{x}|\boldsymbol{x})} p_{0|t}^\theta(\boldsymbol{x}|\boldsymbol{x}) + \frac{q_{t|0}(\boldsymbol{z}|\boldsymbol{z})}{q_{t|0}(\boldsymbol{x}|\boldsymbol{z})} p_{0|t}^\theta(\boldsymbol{z}|\boldsymbol{x}) + \sum_{\boldsymbol{x}_0 \neq \boldsymbol{z}, \boldsymbol{x}} \frac{q_{t|0}(\boldsymbol{z}|\boldsymbol{x}_0)}{q_{t|0}(\boldsymbol{x}|\boldsymbol{x}_0)} p_{0|t}^\theta(\boldsymbol{x}_0|\boldsymbol{x}) \\
&= \frac{\langle \boldsymbol{z}, \boldsymbol{m} \rangle}{\frac{\overline{\alpha}_{t|0}}{1-\overline{\alpha}_{t|0}} + \langle \boldsymbol{x}, \boldsymbol{m} \rangle} p_{0|t}^\theta(\boldsymbol{x}|\boldsymbol{x}) + \frac{\frac{\overline{\alpha}_{t|0}}{1-\overline{\alpha}_{t|0}} + \langle \boldsymbol{z}, \boldsymbol{m} \rangle}{\langle \boldsymbol{x}, \boldsymbol{m} \rangle} p_{0|t}^\theta(\boldsymbol{z}|\boldsymbol{x}) + \frac{\langle \boldsymbol{z}, \boldsymbol{m} \rangle}{\langle \boldsymbol{x}, \boldsymbol{m} \rangle} (1 - p_{0|t}^\theta(\boldsymbol{z}|\boldsymbol{x}) - p_{0|t}^\theta(\boldsymbol{x}|\boldsymbol{x})) \\
&= \frac{1}{\langle \boldsymbol{x}, \boldsymbol{m} \rangle} \Big[(1 - \frac{\overline{\alpha}_{t|0} p_{0|t}^\theta(\boldsymbol{x}|\boldsymbol{x})}{\overline{\alpha}_{t|0} + (1-\overline{\alpha}_{t|0}) \langle \boldsymbol{x}, \boldsymbol{m} \rangle}) \langle \boldsymbol{z}, \boldsymbol{m} \rangle + \frac{\overline{\alpha}_{t|0}}{1-\overline{\alpha}_{t|0}} p_{0|t}^\theta(\boldsymbol{z}|\boldsymbol{x}) \Big] \quad \forall \boldsymbol{z} \neq \boldsymbol{x}
\end{aligned} \tag{66}
$$

Additionally, when $\boldsymbol{z} = \boldsymbol{x}$, $g_t^\theta(\boldsymbol{z}|\boldsymbol{x}) = \sum_{\boldsymbol{x}_0} p_{0|t}^\theta(\boldsymbol{x}_0|\boldsymbol{x}) = 1$. ■

We can also compute the vectorization $g_t^\theta(\cdot|\boldsymbol{x})$ directly as

$$
g_t^\theta(\cdot|\boldsymbol{x}) = \frac{1}{\langle \boldsymbol{x}, \boldsymbol{m} \rangle} \Big[(1 - \frac{\overline{\alpha}_{t|0} \langle f_t^\theta(\boldsymbol{x}), \boldsymbol{x} \rangle}{\overline{\alpha}_{t|0} + (1-\overline{\alpha}_{t|0}) \langle \boldsymbol{x}, \boldsymbol{m} \rangle}) \boldsymbol{m} + \frac{\overline{\alpha}_{t|0}}{1-\overline{\alpha}_{t|0}} f_t^\theta(\boldsymbol{x}) \Big] \odot (\mathbf{1} - \boldsymbol{x}) + \boldsymbol{x} \tag{67}
$$

where $f_t^\theta(\boldsymbol{x})$ is the parameterized neural network for $p_{0|t}^\theta(\cdot|\boldsymbol{x})$ that outputs the distribution of $\mathbf{x}_{0|t}$. Eq. (25) combines Eq. (63) and $g_t^\theta(\boldsymbol{x}|\boldsymbol{x}) = 1, \forall \boldsymbol{x}$.

9.13 Derivation of Forward and Backward Transition Rate in Multi-element Case

In this section, we show how to extend transition rates, and and the ratio g_t^θ, into multi-element case. We let $\boldsymbol{x}^{\setminus d}$ represent $\boldsymbol{x}^{1:D \setminus d}$, i.e. the object without d-th element, for simplicity.

Forward Transition Rates: First, the transition rates for forward sampling has a specific decomposition formulation in multi-element case as proven by Campbell et al. (2022), thus, we summarize the result as follows. The key assumption for CTMC is that at a single time, only one dimension can change.

$$
r_t^{1:D}(\boldsymbol{y}^{1:D}|\boldsymbol{x}^{1:D}) = \sum_{d=1}^{D} r_t^d(\boldsymbol{y}^d|\boldsymbol{x}^d) \delta_{\boldsymbol{x}^{\setminus d}, \boldsymbol{y}^{\setminus d}} \tag{68}
$$

where $\delta_{\boldsymbol{x}^{\setminus d}, \boldsymbol{y}^{\setminus d}}$ is the Kronecker delta and it is 1 if and only if $\boldsymbol{x}^{\setminus d} = \boldsymbol{y}^{\setminus d}$. As we also assume that all dimension processes are indepedent, r_t^d denotes the transition rate of the CTMC process at d-th element/dimension.

Backward Transition Rates: Now let us work on $\hat{r}_t^{1:D}(\boldsymbol{y}^{1:D}|\boldsymbol{x}^{1:D})$. Notice that as the backward process is also a CTMC, it also satisfies that only one dimension can change at a time. We summarize two equivalent formulations as follows.

$$
\hat{r}_t^{1:D}(\boldsymbol{y}^{1:D}|\boldsymbol{x}^{1:D}) = \sum_{d=1}^{D} r_t^d(\boldsymbol{x}^d|\boldsymbol{y}^d) \delta_{\boldsymbol{x}^{\setminus d}, \boldsymbol{y}^{\setminus d}} \sum_{\boldsymbol{x}_0^d} \frac{q_{t|0}(\boldsymbol{y}^d|\boldsymbol{x}_0^d)}{q_{t|0}(\boldsymbol{x}^d|\boldsymbol{x}_0^d)} q_{0|t}(\boldsymbol{x}_0^d|\boldsymbol{x}^{1:D}) \tag{69}
$$

$$\hat{r}_t^{1:D}(\boldsymbol{y}^{1:D}|\boldsymbol{x}^{1:D}) = \sum_{d=1}^{D} \frac{r_t^d(\boldsymbol{x}^d|\boldsymbol{y}^d)\delta_{\boldsymbol{x}^{\backslash d},\boldsymbol{y}^{\backslash d}}}{\sum_{\boldsymbol{x}_0^d} \frac{q_{t|0}(\boldsymbol{x}^d|\boldsymbol{x}_0^d)}{q_{t|0}(\boldsymbol{y}^d|\boldsymbol{x}_0^d)} q_{0|t}(\boldsymbol{x}_0^d|\boldsymbol{y}^{1:D})} \tag{70}$$

Notice that these two formulations should be equivalent. In practice, we use the first formulation to parameterize the reverse transition rate in learning.

Proof

$$\frac{q_t(\boldsymbol{y}^{1:D})}{q_t(\boldsymbol{x}^{1:D})} = \sum_{\boldsymbol{x}_0^{1:D}} \frac{q_{t|0}(\boldsymbol{y}^{1:D}|\boldsymbol{x}_0^{1:D})}{q_{t|0}(\boldsymbol{x}^{1:D}|\boldsymbol{x}_0^{1:D})} q_{0|t}(\boldsymbol{x}_0^{1:D}|\boldsymbol{x}^{1:D}) = \sum_{\boldsymbol{x}_0^{1:D}} q_{0|t}(\boldsymbol{x}_0^{1:D}|\boldsymbol{x}^{1:D}) \prod_{d=1}^{D} \frac{q_{t|0}(\boldsymbol{y}^d|\boldsymbol{x}_0^d)}{q_{t|0}(\boldsymbol{x}^d|\boldsymbol{x}_0^d)} \tag{71}$$

$$\sum_{\boldsymbol{x}_0^{\backslash d}} q_{0|t}(\boldsymbol{x}_0^{1:D}|\boldsymbol{x}^{1:D}) = \sum_{\boldsymbol{x}_0^{\backslash d}} q_{0|t}(\boldsymbol{x}_0^d|\boldsymbol{x}^{1:D}) q_{0|t}(\boldsymbol{x}_0^{\backslash d}|\boldsymbol{x}^{1:D}, \boldsymbol{x}_0^d) = q_{0|t}(\boldsymbol{x}_0^d|\boldsymbol{x}^{1:D}) \tag{72}$$

(Case 1)

$$\begin{aligned}
\hat{r}_t^{1:D}(\boldsymbol{y}^{1:D}|\boldsymbol{x}^{1:D}) &= r_t^{1:D}(\boldsymbol{x}^{1:D}|\boldsymbol{y}^{1:D}) \frac{q_t(\boldsymbol{y}^{1:D})}{q_t(\boldsymbol{x}^{1:D})} \\
&= \Big(\sum_{d=1}^{D} r_t^d(\boldsymbol{x}^d|\boldsymbol{y}^d)\delta_{\boldsymbol{x}^{\backslash d},\boldsymbol{y}^{\backslash d}} \Big) \Big(\sum_{\boldsymbol{x}_0^{1:D}} q_{0|t}(\boldsymbol{x}_0^{1:D}|\boldsymbol{x}^{1:D}) \prod_{d=1}^{D} \frac{q_{t|0}(\boldsymbol{y}^d|\boldsymbol{x}_0^d)}{q_{t|0}(\boldsymbol{x}^d|\boldsymbol{x}_0^d)} \Big) \\
&= \sum_{d=1}^{D} \sum_{\boldsymbol{x}_0^{1:D}} r_t^d(\boldsymbol{x}^d|\boldsymbol{y}^d)\delta_{\boldsymbol{x}^{\backslash d},\boldsymbol{y}^{\backslash d}} q_{0|t}(\boldsymbol{x}_0^{1:D}|\boldsymbol{x}^{1:D}) \frac{q_{t|0}(\boldsymbol{y}^d|\boldsymbol{x}_0^d)}{q_{t|0}(\boldsymbol{x}^d|\boldsymbol{x}_0^d)} \\
&= \sum_{d=1}^{D} r_t^d(\boldsymbol{x}^d|\boldsymbol{y}^d)\delta_{\boldsymbol{x}^{\backslash d},\boldsymbol{y}^{\backslash d}} \sum_{\boldsymbol{x}_0^d} \frac{q_{t|0}(\boldsymbol{y}^d|\boldsymbol{x}_0^d)}{q_{t|0}(\boldsymbol{x}^d|\boldsymbol{x}_0^d)} \sum_{\boldsymbol{x}_0^{\backslash d}} q_{0|t}(\boldsymbol{x}_0^{1:D}|\boldsymbol{x}^{1:D}) \\
&= \sum_{d=1}^{D} r_t^d(\boldsymbol{x}^d|\boldsymbol{y}^d)\delta_{\boldsymbol{x}^{\backslash d},\boldsymbol{y}^{\backslash d}} \sum_{\boldsymbol{x}_0^d} \frac{q_{t|0}(\boldsymbol{y}^d|\boldsymbol{x}_0^d)}{q_{t|0}(\boldsymbol{x}^d|\boldsymbol{x}_0^d)} q_{0|t}(\boldsymbol{x}_0^d|\boldsymbol{x}^{1:D})
\end{aligned} \tag{73}$$

(Case 2)

$$\begin{aligned}
\hat{r}_t^{1:D}(\boldsymbol{y}^{1:D}|\boldsymbol{x}^{1:D}) &= r_t^{1:D}(\boldsymbol{x}^{1:D}|\boldsymbol{y}^{1:D}) / \frac{q_t(\boldsymbol{x}^{1:D})}{q_t(\boldsymbol{y}^{1:D})} \\
&= \sum_{d=1}^{D} \frac{r_t^d(\boldsymbol{x}^d|\boldsymbol{y}^d)\delta_{\boldsymbol{x}^{\backslash d},\boldsymbol{y}^{\backslash d}}}{\sum_{\boldsymbol{x}_0^{1:D}} q_{0|t}(\boldsymbol{x}_0^{1:D}|\boldsymbol{y}^{1:D}) \prod_{i=1}^{D} \frac{q_{t|0}(\boldsymbol{x}^i|\boldsymbol{x}_0^i)}{q_{t|0}(\boldsymbol{y}^i|\boldsymbol{x}_0^i)}} \\
&= \sum_{d=1}^{D} \frac{r_t^d(\boldsymbol{x}^d|\boldsymbol{y}^d)\delta_{\boldsymbol{x}^{\backslash d},\boldsymbol{y}^{\backslash d}}}{\sum_{\boldsymbol{x}_0^{1:D}} q_{0|t}(\boldsymbol{x}_0^{1:D}|\boldsymbol{y}^{1:D}) \frac{q_{t|0}(\boldsymbol{x}^d|\boldsymbol{x}_0^d)}{q_{t|0}(\boldsymbol{y}^d|\boldsymbol{x}_0^d)}} \\
&= \sum_{d=1}^{D} \frac{r_t^d(\boldsymbol{x}^d|\boldsymbol{y}^d)\delta_{\boldsymbol{x}^{\backslash d},\boldsymbol{y}^{\backslash d}}}{\sum_{\boldsymbol{x}_0^d} \frac{q_{t|0}(\boldsymbol{x}^d|\boldsymbol{x}_0^d)}{q_{t|0}(\boldsymbol{y}^d|\boldsymbol{x}_0^d)} q_{0|t}(\boldsymbol{x}_0^d|\boldsymbol{y}^{1:D})}
\end{aligned} \tag{74}$$

■

Ratio: We now define an estimator $g_t^{\theta,d}(\boldsymbol{x}^d|\boldsymbol{y}^{1:D})$ as follows.

$$g_t^{\theta,d}(\boldsymbol{x}^d|\boldsymbol{y}^{1:D}) := \sum_{\boldsymbol{x}_0^d} \frac{q_{t|0}(\boldsymbol{x}^d|\boldsymbol{x}_0^d)}{q_{t|0}(\boldsymbol{y}^d|\boldsymbol{x}_0^d)} p_{0|t}^{\theta}(\boldsymbol{x}_0^d|\boldsymbol{y}^{1:D}) \approx \sum_{\boldsymbol{x}_0^d} \frac{q_{t|0}(\boldsymbol{x}^d|\boldsymbol{x}_0^d)}{q_{t|0}(\boldsymbol{y}^d|\boldsymbol{x}_0^d)} q_{0|t}(\boldsymbol{x}_0^d|\boldsymbol{y}^{1:D}) = \frac{q_t(\boldsymbol{x}^d|\boldsymbol{y}^{\setminus d})}{q_t(\boldsymbol{y}^d|\boldsymbol{y}^{\setminus d})} \tag{75}$$

$$g_t^{\theta}(\boldsymbol{x}^{1:D}|\boldsymbol{y}^{1:D}) := \prod_{d=1}^{D} g_t^{\theta,d}(\boldsymbol{x}^d|\boldsymbol{y}^{1:D}) = \sum_{\boldsymbol{x}_0^{1:D}} \frac{q_{t|0}(\boldsymbol{x}^{1:D}|\boldsymbol{x}_0^{1:D})}{q_{t|0}(\boldsymbol{y}^{1:D}|\boldsymbol{x}_0^{1:D})} p_{0|t}^{\theta}(\boldsymbol{x}_0^{1:D}|\boldsymbol{y}^{1:D}) \approx \frac{q_t(\boldsymbol{x}^{1:D})}{q_t(\boldsymbol{y}^{1:D})} \tag{76}$$

We can extend the vector formulation Eq. (25) in Proposition 4 to $g_t^{\theta,d}(\boldsymbol{x}^d|\boldsymbol{y}^{1:D})$:

$$g_t^{\theta,d}(\cdot|\boldsymbol{x}^{1:D}) = \frac{1}{\langle \boldsymbol{x}^d, \boldsymbol{m}^d \rangle} \Big[(1 - \frac{\overline{\alpha}_{t|0} \langle f_t^{\theta,d}(\boldsymbol{x}^{1:D}), \boldsymbol{x}^d \rangle}{\overline{\alpha}_{t|0} + (1 - \overline{\alpha}_{t|0}) \langle \boldsymbol{x}^d, \boldsymbol{m}^d \rangle}) \boldsymbol{m}^d + \frac{\overline{\alpha}_{t|0}}{1 - \overline{\alpha}_{t|0}} f_t^{\theta,d}(\boldsymbol{x}^{1:D}) \Big] \odot (\mathbf{1} - \boldsymbol{x}^d) + \boldsymbol{x}^d \tag{77}$$

Then, we can derive two approximators for transition rate $\hat{r}_t^{1:D}(\boldsymbol{y}^{1:D}|\boldsymbol{x}^{1:D})$ as follows.

$$[\hat{r}_t^{\theta,1:D}]^1(\boldsymbol{y}^{1:D}|\boldsymbol{x}^{1:D}) := \sum_{d=1}^{D} r_t^d(\boldsymbol{x}^d|\boldsymbol{y}^d) g_t^{\theta,d}(\boldsymbol{y}^d|\boldsymbol{x}^{1:D}) \cdot \delta_{\boldsymbol{x}^{\setminus d}, \boldsymbol{y}^{\setminus d}} \approx \text{Eq. (69)} \tag{78}$$

$$[\hat{r}_t^{\theta,1:D}]^2(\boldsymbol{y}^{1:D}|\boldsymbol{x}^{1:D}) := \sum_{d=1}^{D} \frac{r_t^d(\boldsymbol{x}^d|\boldsymbol{y}^d)}{g_t^{\theta,d}(\boldsymbol{x}^d|\boldsymbol{y}^{1:D})} \cdot \delta_{\boldsymbol{x}^{\setminus d}, \boldsymbol{y}^{\setminus d}} \approx \text{Eq. (70)} \tag{79}$$

9.14 Derivation of Negative ELBO Loss in Multi-element Case

As forward process is defined in Eq. (19), in multi-element case we can easily get

$$r_t^d(\boldsymbol{x}^d|\cdot) = R_t^d \boldsymbol{x}^d = \beta(t)(\langle \boldsymbol{x}^d, \boldsymbol{m}^d \rangle \mathbf{1} - \boldsymbol{x}^d) \tag{80}$$

$$r_t^d(\cdot|\boldsymbol{x}^d) = (R_t^d)^\top \boldsymbol{x}^d = \beta(t)(\boldsymbol{m}^d - \boldsymbol{x}^d) \tag{81}$$

The $r_t^d(\boldsymbol{x}|\cdot)$ and $r_t^d(\cdot|\boldsymbol{x})$ are essentially the $\boldsymbol{x}$-th column and row of the transition rate matrix R_t^d.

In Mult-element case, the negative ELBO loss in Eq. (23) can be written as

$$T \, \mathbb{E}_{\substack{t \sim \text{Uni}(0,T) \\ \boldsymbol{x}^{1:D} \sim q_{t|0}}} \Big[\sum_{\boldsymbol{z}^{1:D} \neq \boldsymbol{x}^{1:D}} \hat{r}_t^{\theta,1:D}(\boldsymbol{z}^{1:D}|\boldsymbol{x}^{1:D}) - \sum_{\boldsymbol{z}^{1:D} \neq \boldsymbol{x}^{1:D}} r_t^{1:D}(\boldsymbol{z}^{1:D}|\boldsymbol{x}^{1:D}) \log \hat{r}_t^{\theta,1:D}(\boldsymbol{x}^{1:D}|\boldsymbol{z}^{1:D}) \Big] \tag{82}$$

As there are two terms, let's work on each term separately.

9.14.1 Term 1

Based on the formulation of $r_t^{1:D}$ and $\hat{r}_t^{\theta,1:D}$, we can rewrite the first term as

$$\begin{aligned}
\text{Term1} &= T\ \mathbb{E}_{t,\boldsymbol{x}^{1:D}} \sum_{\boldsymbol{z}^{1:D}\neq\boldsymbol{x}^{1:D}} \hat{r}_t^{\theta,1:D}(\boldsymbol{z}^{1:D}|\boldsymbol{x}^{1:D}) \\
&= T\ \mathbb{E}_{t,\boldsymbol{x}^{1:D}} \sum_{\boldsymbol{z}^{1:D}\neq\boldsymbol{x}^{1:D}} \sum_{d=1}^{D} r_t^d(\boldsymbol{x}^d|\boldsymbol{z}^d) g_t^{\theta,d}(\boldsymbol{z}^d|\boldsymbol{x}^{1:D}) \cdot \delta_{\boldsymbol{x}^{\backslash d},\boldsymbol{z}^{\backslash d}} \\
&= T\ \mathbb{E}_{t,\boldsymbol{x}^{1:D}} \Big[\sum_{d=1}^{D} \sum_{\boldsymbol{z}^d\neq\boldsymbol{x}^d} r_t^d(\boldsymbol{x}^d|\boldsymbol{z}^d) g_t^{\theta,d}(\boldsymbol{z}^d|\boldsymbol{x}^{1:D}) \Big] \\
&= T\ \mathbb{E}_{t,\boldsymbol{x}^{1:D}} \Big[\sum_{d=1}^{D} r_t^d(\boldsymbol{x}^d|\cdot)^\top g_t^{\theta,d}(\cdot|\boldsymbol{x}^{1:D}) \Big] + \text{const.} \\
&= T\ \mathbb{E}_{\substack{t\sim\text{Uni.}(0,T) \\ \boldsymbol{x}^{1:D}\sim q_{t|0}}} \beta(t) \sum_{d=1}^{D} \langle \boldsymbol{x}^d, \boldsymbol{m}^d \rangle \Big[\mathbf{1}^\top g_t^{\theta,d}(\cdot|\boldsymbol{x}^{1:D}) \Big] + \text{const.}
\end{aligned} \tag{83}$$

9.14.2 Term 2

As the evaluation of $\hat{r}_t^\theta(\cdot|\boldsymbol{x})$ for any $\boldsymbol{x}$ requires a single forward pass of the parameterized network $p_{0|t}^\theta(\cdot|\boldsymbol{x})$, the second term within the expectation of Eq. (23) requires multiple passes of the model. This complexity is even greatly amplified in cases with multi-element objects. Campbell et al. (2022) avoids the multiple passes by changing the expectation variable through importance sampling. We take a similar approach to simplify the second term. Differently, Campbell et al. (2022) uses a specific sampling distribution (same as the forward transition rate) to introduce the auxiliary variable for changing the expectation variable, we generalize it to use a general sampling process S_t defined below.

Let $\boldsymbol{y}^{1:D}$ be the new variable upon which the exchanged expectation is based, and assume that $\boldsymbol{y}^{1:D}$ is sampled from an unnormalized joint distribution $S_t(\boldsymbol{y}^{1:D}|\boldsymbol{x}^{1:D})$. We restrict $S_t(\boldsymbol{y}^{1:D}|\boldsymbol{x}^{1:D})$ to be a unnormalized probability that is nonzero if and only if $\boldsymbol{y}^{1:D}$ and $\boldsymbol{x}^{1:D}$ are different at a single element. Formally, we can write the unnormalized distribution as

$$\begin{aligned}
S_t(\boldsymbol{y}^{1:D}|\boldsymbol{x}^{1:D}) &= (1 - \delta_{\boldsymbol{y}^{1:D},\boldsymbol{x}^{1:D}}) \sum_{d=1}^{D} S_t^d(\boldsymbol{y}^d|\boldsymbol{x}^d) \delta_{\boldsymbol{y}^{\backslash d},\boldsymbol{x}^{\backslash d}} \\
\text{with normalizer } \mathcal{S}_t(\boldsymbol{x}^{1:D}) &= \sum_{\boldsymbol{y}^{1:D}} S_t(\boldsymbol{y}^{1:D}|\boldsymbol{x}^{1:D}) = \sum_{d=1}^{D} \sum_{\boldsymbol{y}^d\neq\boldsymbol{x}^d} S_t^d(\boldsymbol{y}^d|\boldsymbol{x}^d)
\end{aligned} \tag{84}$$

where $S_t^d(\boldsymbol{y}^d|\boldsymbol{x}^d)$ is any unnormalized probability at dimension d.

Now for the second term, we have

$$
\begin{aligned}
\text{Term2} &= T\ \mathbb{E}_{t,\boldsymbol{x}^{1:D}} \sum_{\boldsymbol{z}^{1:D}\neq\boldsymbol{x}^{1:D}} r_t(\boldsymbol{z}^{1:D}|\boldsymbol{x}^{1:D}) \log \hat{r}_t^\theta(\boldsymbol{x}^{1:D}|\boldsymbol{z}^{1:D}) \\
&= T\ \mathbb{E}_{t,\boldsymbol{x}^{1:D}} \sum_{\boldsymbol{z}^{1:D}\neq\boldsymbol{x}^{1:D}} \frac{S_t(\boldsymbol{z}^{1:D}|\boldsymbol{x}^{1:D})}{\mathcal{S}(\boldsymbol{x}^{1:D})} \cdot \frac{\mathcal{S}_t(\boldsymbol{x}^{1:D})}{S_t(\boldsymbol{z}^{1:D}|\boldsymbol{x}^{1:D})} \cdot r_t(\boldsymbol{z}^{1:D}|\boldsymbol{x}^{1:D}) \log \hat{r}_t^\theta(\boldsymbol{x}^{1:D}|\boldsymbol{z}^{1:D}) \\
&= T\ \mathbb{E}_{\substack{t,\boldsymbol{x}^{1:D} \\ \boldsymbol{z}^{1:D}\sim S_t}} \Big[\frac{\mathcal{S}_t(\boldsymbol{x}^{1:D})}{S_t(\boldsymbol{z}^{1:D}|\boldsymbol{x}^{1:D})} \cdot r_t(\boldsymbol{z}^{1:D}|\boldsymbol{x}^{1:D}) \log \hat{r}_t^\theta(\boldsymbol{x}^{1:D}|\boldsymbol{z}^{1:D})\Big] \qquad (\text{As } S_t \text{ samples } \boldsymbol{z}^{1:D}\neq\boldsymbol{x}^{1:D}\ ) \\
&= T\ \mathbb{E}_{t,\boldsymbol{z}^{1:D}\sim S_t} \sum_{\boldsymbol{x}^{1:D}} \Big[q_{S_t}(\boldsymbol{x}^{1:D}|\boldsymbol{x}_0^{1:D},\boldsymbol{z}^{1:D}) \cdot \frac{\mathcal{S}_t(\boldsymbol{x}^{1:D})}{S_t(\boldsymbol{z}^{1:D}|\boldsymbol{x}^{1:D})} \cdot r_t(\boldsymbol{z}^{1:D}|\boldsymbol{x}^{1:D}) \log \hat{r}_t^\theta(\boldsymbol{x}^{1:D}|\boldsymbol{z}^{1:D})\Big]
\end{aligned}
\tag{85}
$$

where $q_{S_t}(\boldsymbol{x}^{1:D}|\boldsymbol{x}_0^{1:D},\boldsymbol{z}^{1:D})$ is the conditional posterior distribution such that (for clearity we replace $\boldsymbol{x}^{1:D},\boldsymbol{z}^{1:D}$ with $\boldsymbol{x}_t^{1:D},\boldsymbol{z}_t^{1:D}$ respectively, as they are variables at time t)

$$
\begin{aligned}
&q_{S_t}(\boldsymbol{x}_t^{1:D}|\boldsymbol{x}_0^{1:D},\boldsymbol{z}_t^{1:D}) \\
&= \frac{q_{S_t}(\boldsymbol{x}_t^{1:D},\boldsymbol{z}_t^{1:D}|\boldsymbol{x}_0^{1:D})}{\sum_{\boldsymbol{y}_t^{1:D}} q_{S_t}(\boldsymbol{y}_t^{1:D},\boldsymbol{z}_t^{1:D}|\boldsymbol{x}_0^{1:D})} = \frac{q_{t|0}(\boldsymbol{x}_t^{1:D}|\boldsymbol{x}_0^{1:D}) \cdot S_t(\boldsymbol{z}_t^{1:D}|\boldsymbol{x}_t^{1:D})/\mathcal{S}_t(\boldsymbol{x}_t^{1:D})}{\sum_{\boldsymbol{y}_y^{1:D}} q_{t|0}(\boldsymbol{x}^{1:D}|\boldsymbol{x}_0^{1:D}) \cdot S_t(\boldsymbol{z}_t^{1:D}|\boldsymbol{y}_t^{1:D})/\mathcal{S}_t(\boldsymbol{y}_t^{1:D})} \\
&= \frac{(1-\delta_{\boldsymbol{z}_t,\boldsymbol{x}_t})\sum_{d=1}^D \delta_{\boldsymbol{z}_t^{\setminus d},\boldsymbol{x}_t^{\setminus d}} S_t^d(\boldsymbol{z}_t^d|\boldsymbol{x}_t^d)/\mathcal{S}_t(\boldsymbol{x}_t^{1:D}) \cdot q_{t|0}(\boldsymbol{x}_t^d \circ \boldsymbol{z}_t^{\setminus d}|\boldsymbol{x}_0^{1:D})}{\sum_{\boldsymbol{y}_t^{1:D}} \left[(1-\delta_{\boldsymbol{z}_t,\boldsymbol{y}_t})\sum_{d=1}^D \delta_{\boldsymbol{z}_t^{\setminus d},\boldsymbol{y}_t^{\setminus d}} S_t^d(\boldsymbol{z}_t^d|\boldsymbol{y}_t^d)/\mathcal{S}_t(\boldsymbol{y}_t^{1:D}) \cdot q_{t|0}(\boldsymbol{y}_t^d \circ \boldsymbol{z}_t^{\setminus d}|\boldsymbol{x}_0^{1:D})\right]} \\
&= \frac{(1-\delta_{\boldsymbol{z}_t,\boldsymbol{x}_t})\sum_{d=1}^D \delta_{\boldsymbol{z}_t^{\setminus d},\boldsymbol{x}_t^{\setminus d}} \frac{S_t^d(\boldsymbol{z}_t^d|\boldsymbol{x}_t^d)}{\mathcal{S}_t(\boldsymbol{x}_t^{1:D})} \cdot \frac{q_{t|0}(\boldsymbol{x}_t^d|\boldsymbol{x}_0^d)}{q_{t|0}(\boldsymbol{z}_t^d|\boldsymbol{x}_0^d)}}{\sum_{\boldsymbol{y}_t^{1:D}} \left[(1-\delta_{\boldsymbol{z}_t,\boldsymbol{y}_t})\sum_{d=1}^D \delta_{\boldsymbol{z}_t^{\setminus d},\boldsymbol{y}_t^{\setminus d}} \frac{S_t^d(\boldsymbol{z}_t^d|\boldsymbol{y}_t^d)}{\mathcal{S}_t(\boldsymbol{y}_t^{1:D})} \cdot \frac{q_{t|0}(\boldsymbol{y}_t^d|\boldsymbol{x}_0^d)}{q_{t|0}(\boldsymbol{z}_t^d|\boldsymbol{x}_0^d)}\right]} \\
&= \frac{(1-\delta_{\boldsymbol{z}_t,\boldsymbol{x}_t})\sum_{d=1}^D \delta_{\boldsymbol{z}_t^{\setminus d},\boldsymbol{x}_t^{\setminus d}} \frac{S_t^d(\boldsymbol{z}_t^d|\boldsymbol{x}_t^d)}{\mathcal{S}_t(\boldsymbol{x}_t^{1:D})} \cdot \frac{q_{t|0}(\boldsymbol{x}_t^d|\boldsymbol{x}_0^d)}{q_{t|0}(\boldsymbol{z}_t^d|\boldsymbol{x}_0^d)}}{\sum_{d=1}^D \sum_{\boldsymbol{y}_t^d\neq\boldsymbol{z}^d} \frac{S_t^d(\boldsymbol{z}_t^d|\boldsymbol{y}_t^d)}{\mathcal{S}_t(\boldsymbol{y}_t^d \circ \boldsymbol{z}_t^{\setminus d})} \cdot \frac{q_{t|0}(\boldsymbol{y}_t^d|\boldsymbol{x}_0^d)}{q_{t|0}(\boldsymbol{z}_t^d|\boldsymbol{x}_0^d)}}
\end{aligned}
\tag{86}
$$

Now taking the formulation of q_{S_t} back into Term 2, we further simplify Term 2 as

Term2

$$
\begin{aligned}
&= T\ \mathbb{E}_{t,\boldsymbol{z}^{1:D}\sim S_t} \frac{\sum_{\boldsymbol{x}^{1:D}} \Big[(1-\delta_{\boldsymbol{z},\boldsymbol{x}}) \sum_{d=1}^D \delta_{\boldsymbol{z}^{\backslash d},\boldsymbol{x}^{\backslash d}} \frac{S_t^d(\boldsymbol{z}^d|\boldsymbol{x}^d)}{S_t(\boldsymbol{z}^{1:D}|\boldsymbol{x}^{1:D})} \cdot \frac{q_{t|0}(\boldsymbol{x}^d|\boldsymbol{x}_0^d)}{q_{t|0}(\boldsymbol{z}^d|\boldsymbol{x}_0^d)} \cdot r_t(\boldsymbol{z}^{1:D}|\boldsymbol{x}^{1:D}) \log \hat{r}_t^\theta(\boldsymbol{x}^{1:D}|\boldsymbol{z}^{1:D})\Big]}{\sum_{d=1}^D \sum_{\boldsymbol{y}^d \neq \boldsymbol{z}^d} \frac{S_t^d(\boldsymbol{z}^d|\boldsymbol{y}^d)}{\mathcal{S}_t(\boldsymbol{y}^d \circ \boldsymbol{z}^{\backslash d})} \cdot \frac{q_{t|0}(\boldsymbol{y}^d|\boldsymbol{x}_0^d)}{q_{t|0}(\boldsymbol{z}^d|\boldsymbol{x}_0^d)}} \\
&= \mathbb{E}_{t,\boldsymbol{z}^{1:D}\sim S_t} \frac{\sum_{d=1}^D \sum_{\boldsymbol{x}^d \neq \boldsymbol{z}^d} \frac{S_t^d(\boldsymbol{z}^d|\boldsymbol{x}^d)}{S_t(\boldsymbol{z}^{1:D}|\boldsymbol{x}^d \circ \boldsymbol{z}^{\backslash d})} \cdot \frac{q_{t|0}(\boldsymbol{x}^d|\boldsymbol{x}_0^d)}{q_{t|0}(\boldsymbol{z}^d|\boldsymbol{x}_0^d)} \cdot r_t(\boldsymbol{z}^{1:D}|\boldsymbol{x}^d \circ \boldsymbol{z}^{\backslash d}) \log \hat{r}_t^\theta(\boldsymbol{x}^d \circ \boldsymbol{z}^{\backslash d}|\boldsymbol{z}^{1:D})}{\sum_{d=1}^D \sum_{\boldsymbol{y}^d \neq \boldsymbol{z}^d} \frac{S_t^d(\boldsymbol{z}^d|\boldsymbol{y}^d)}{\mathcal{S}_t(\boldsymbol{y}^d \circ \boldsymbol{z}^{\backslash d})} \cdot \frac{q_{t|0}(\boldsymbol{y}^d|\boldsymbol{x}_0^d)}{q_{t|0}(\boldsymbol{z}^d|\boldsymbol{x}_0^d)}} \\
&= \mathbb{E}_{t,\boldsymbol{z}^{1:D}\sim S_t} \frac{\sum_{d=1}^D \sum_{\boldsymbol{x}^d \neq \boldsymbol{z}^d} \frac{q_{t|0}(\boldsymbol{x}^d|\boldsymbol{x}_0^d)}{q_{t|0}(\boldsymbol{z}^d|\boldsymbol{x}_0^d)} \cdot r_t^d(\boldsymbol{z}^d|\boldsymbol{x}^d) \cdot \log r_t^d(\boldsymbol{z}^d|\boldsymbol{x}^d) g_t^{\theta,d}(\boldsymbol{x}^d|\boldsymbol{z}^{1:D})}{\sum_{d=1}^D \sum_{\boldsymbol{y}^d \neq \boldsymbol{z}^d} \frac{q_{t|0}(\boldsymbol{y}^d|\boldsymbol{x}_0^d)}{q_{t|0}(\boldsymbol{z}^d|\boldsymbol{x}_0^d)} \cdot \frac{S_t^d(\boldsymbol{z}^d|\boldsymbol{y}^d)}{\mathcal{S}_t(\boldsymbol{y}^d \circ \boldsymbol{z}^{\backslash d})}} \\
&= \mathbb{E}_{t,\boldsymbol{z}^{1:D}\sim S_t} \frac{\sum_{d=1}^D \sum_{\boldsymbol{x}^d \neq \boldsymbol{z}^d} \frac{q_{t|0}(\boldsymbol{x}^d|\boldsymbol{x}_0^d)}{q_{t|0}(\boldsymbol{z}^d|\boldsymbol{x}_0^d)} \cdot r_t^d(\boldsymbol{z}^d|\boldsymbol{x}^d) \cdot \log g_t^{\theta,d}(\boldsymbol{x}^d|\boldsymbol{z}^{1:D})}{\sum_{d=1}^D \sum_{\boldsymbol{y}^d \neq \boldsymbol{z}^d} \frac{q_{t|0}(\boldsymbol{y}^d|\boldsymbol{x}_0^d)}{q_{t|0}(\boldsymbol{z}^d|\boldsymbol{x}_0^d)} \cdot \frac{S_t^d(\boldsymbol{z}^d|\boldsymbol{y}^d)}{\mathcal{S}_t(\boldsymbol{y}^d \circ \boldsymbol{z}^{\backslash d})}} + \text{const.}
\end{aligned} \tag{87}
$$

The above equation further shows that the sampling distribution S_t for adding exchanging variable $\boldsymbol{z}^{1:D}$ only affect a weighting term of the loss computation. Let us define the scalar weighting term as

$$
\mathcal{M}_{S_t}(\boldsymbol{z}^{1:D}|\boldsymbol{x}_0^{1:D}) := \sum_{d=1}^D \sum_{\boldsymbol{y}^d \neq \boldsymbol{z}^d} \frac{q_{t|0}(\boldsymbol{y}^d|\boldsymbol{x}_0^d)}{q_{t|0}(\boldsymbol{z}^d|\boldsymbol{x}_0^d)} \cdot \frac{S_t^d(\boldsymbol{z}^d|\boldsymbol{y}^d)}{\mathcal{S}_t(\boldsymbol{y}^d \circ \boldsymbol{z}^{\backslash d})} \tag{88}
$$

With this definition, the Term 2 can be further rewrited as

$$
\begin{aligned}
\text{Term2} &= \mathbb{E}_{t,\boldsymbol{z}^{1:D}\sim S_t}\Big[\frac{1}{\mathcal{M}_{S_t}(\boldsymbol{z}^{1:D}|\boldsymbol{x}_0^{1:D})} \sum_{d=1}^D \frac{1}{q_{t|0}(\boldsymbol{z}^d|\boldsymbol{x}_0^d)} \sum_{\boldsymbol{x}^d \neq \boldsymbol{z}^d} q_{t|0}(\boldsymbol{x}^d|\boldsymbol{x}_0^d) \cdot r_t^d(\boldsymbol{z}^d|\boldsymbol{x}^d) \cdot \log g_t^{\theta,d}(\boldsymbol{x}^d|\boldsymbol{z}^{1:D})\Big] \\
&= \mathbb{E}_{t,\boldsymbol{z}^{1:D}\sim S_t}\Big[\frac{1}{\mathcal{M}_{S_t}(\boldsymbol{z}^{1:D}|\boldsymbol{x}_0^{1:D})} \sum_{d=1}^D \frac{\mathbf{1}^\top[q_{t|0}(\cdot|\boldsymbol{x}_0^d) \odot r_t^d(\boldsymbol{z}^d|\cdot) \odot \log g_t^{\theta,d}(\cdot|\boldsymbol{z}^{1:D})]}{q_{t|0}(\boldsymbol{z}^d|\boldsymbol{x}_0^d)}\Big] + \text{const.} \\
&= \mathbb{E}_{t,\boldsymbol{z}^{1:D}\sim S_t}\Big[\frac{\beta(t)}{\mathcal{M}_{S_t}(\boldsymbol{z}^{1:D}|\boldsymbol{x}_0^{1:D})} \sum_{d=1}^D \frac{\langle \boldsymbol{z}^d, \boldsymbol{m}^d\rangle}{q_{t|0}(\boldsymbol{z}^d|\boldsymbol{x}_0^d)} \mathbf{1}^\top[q_{t|0}(\cdot|\boldsymbol{x}_0^d) \odot \log g_t^{\theta,d}(\cdot|\boldsymbol{z}^{1:D})] + \text{const.}\Big]
\end{aligned} \tag{89}
$$

9.14.3 All Terms

Combine term 1 and term 2 together, we can write the negative ELBO loss as

$$
\begin{aligned}
&T\ \mathbb{E}_{\substack{t\sim\text{Uni}(0,T) \\ \boldsymbol{x}^{1:D}\sim q_{t|0}}}\Big[\sum_{\boldsymbol{z}^{1:D}\neq\boldsymbol{x}^{1:D}} \hat{r}_t^{\theta,1:D}(\boldsymbol{z}^{1:D}|\boldsymbol{x}^{1:D}) - \sum_{\boldsymbol{z}^{1:D}\neq\boldsymbol{x}^{1:D}} r_t^{1:D}(\boldsymbol{z}^{1:D}|\boldsymbol{x}^{1:D}) \log \hat{r}_t^{\theta,1:D}(\boldsymbol{x}^{1:D}|\boldsymbol{z}^{1:D})\Big] \\
&= T\ \mathbb{E}_{\substack{t\sim\text{Uni}(0,T) \\ \boldsymbol{x}^{1:D}\sim q_{t|0}(\boldsymbol{x}_0^{1:D}) \\ \boldsymbol{z}^{1:D}\sim S_t(\boldsymbol{x}^{1:D})}}\Big[\sum_{d=1}^D r_t^d(\boldsymbol{x}^d|\cdot)^\top g_t^{\theta,d}(\cdot|\boldsymbol{x}^{1:D}) \\
&- \frac{1}{\mathcal{M}_{S_t}(\boldsymbol{z}^{1:D}|\boldsymbol{x}_0^{1:D})} \sum_{d=1}^D \frac{\mathbf{1}^\top[q_{t|0}(\cdot|\boldsymbol{x}_0^d) \odot r_t^d(\boldsymbol{z}^d|\cdot) \odot \log g_t^{\theta,d}(\cdot|\boldsymbol{z}^{1:D})]}{q_{t|0}(\boldsymbol{z}^d|\boldsymbol{x}_0^d)}\Big] + \text{const.}
\end{aligned} \tag{90}
$$

9.15 Further Simplification of Continuous ELBO

Proposition 7 *The loss in Eq. (28) can be further simplified as*

$$T\ \mathbb{E}_{\substack{t\sim Uni(0,T)\\ \boldsymbol{x}^{1:D}\sim q_{t|0}(\cdot|\boldsymbol{x}_0^{1:D})}}\Big[\sum_{d=1}^{D} r_t^d(\boldsymbol{x}^d|\cdot)^\top\Big(g_t^{\theta,d}(\cdot|\boldsymbol{x}^{1:D}) - \frac{q_{t|0}(\cdot|\boldsymbol{x}_0^d)\odot \log g_t^{\theta,d}(\cdot|\boldsymbol{x}^{1:D})}{q_{t|0}(\boldsymbol{x}^d|\boldsymbol{x}_0^d)}\Big)\Big] \tag{91}$$

where the loss only requires a single forward pass of the model and no auxiliary variable is needed.

Proof We use $q_{t|0}(\boldsymbol{x}^{1:D}|\boldsymbol{x}_0^{1:D})$ to denote $P(\mathbf{x}_t^{1:D}=\boldsymbol{x}^{1:D}|\boldsymbol{x}_0^{1:D})$. Similarly, let $\tilde{q}_{t|0}(\boldsymbol{x}^{1:D}|\boldsymbol{x}_0^{1:D})$ denotes $P(\mathbf{z}_t^{1:D}=\boldsymbol{x}^{1:D}|\boldsymbol{x}_0^{1:D})$. Based on the definition of S_t in Eq. (84) and the procedure of sampling $\mathbf{z}_t^{1:D}$ conditional on $\mathbf{x}_t^{1:D}$ described in Proposition 5, we can compute the conditional probability $\tilde{q}_{t|0}(\boldsymbol{x}^{1:D}|\boldsymbol{x}_0^{1:D})$ as follows:

$$\begin{aligned}
\tilde{q}_{t|0}(\boldsymbol{x}^{1:D}|\boldsymbol{x}_0^{1:D}) &= \sum_{\boldsymbol{x}_t^{1:D}} P(\mathbf{z}_t^{1:D}=\boldsymbol{x}^{1:D},\boldsymbol{x}_t^{1:D}|\boldsymbol{x}_0^{1:D}) = \sum_{\boldsymbol{x}_t^{1:D}} \frac{S_t(\boldsymbol{x}^{1:D}|\boldsymbol{x}_t^{1:D})}{S_t(\boldsymbol{x}_t^{1:D})} q_{t|0}(\boldsymbol{x}_t^{1:D}|\boldsymbol{x}_0^{1:D})\\
&= \sum_{\boldsymbol{x}_t^{1:D}} (1-\delta_{\boldsymbol{x}^{1:D},\boldsymbol{x}_t^{1:D}})\Big(\sum_{d=1}^{D}\delta_{\boldsymbol{x}^{\setminus d},\boldsymbol{x}_t^{\setminus d}}\cdot\frac{S_t^d(\boldsymbol{x}^d|\boldsymbol{x}_t^d)}{S_t(\boldsymbol{x}^{\setminus d}\circ\boldsymbol{x}_t^d)} q_{t|0}(\boldsymbol{x}^{\setminus d}\circ\boldsymbol{x}_t^d|\boldsymbol{x}_0^{1:D})\Big)\\
&= q_{t|0}(\boldsymbol{x}^{1:D}|\boldsymbol{x}_0^{1:D})\sum_{\boldsymbol{x}_t^{1:D}} (1-\delta_{\boldsymbol{x}^{1:D},\boldsymbol{x}_t^{1:D}})\Big(\sum_{d=1}^{D}\delta_{\boldsymbol{x}^{\setminus d},\boldsymbol{x}_t^{\setminus d}}\cdot\frac{S_t^d(\boldsymbol{x}^d|\boldsymbol{x}_t^d)}{S_t(\boldsymbol{x}^{\setminus d}\circ\boldsymbol{x}_t^d)}\frac{q_{t|0}(\boldsymbol{x}_t^d|\boldsymbol{x}_0^d)}{q_{t|0}(\boldsymbol{x}^d|\boldsymbol{x}_0^d)}\Big)\\
&= q_{t|0}(\boldsymbol{x}^{1:D}|\boldsymbol{x}_0^{1:D})\ \mathcal{M}_{S_t}(\boldsymbol{x}^{1:D}|\boldsymbol{x}_0^{1:D}) \text{ (Based on } \mathcal{M}_{S_t} \text{ in Eq. (88))}.
\end{aligned} \tag{92}$$

The original loss in Eq. (28) can be written as

$$\begin{aligned}
Eq.\ (28) = T\ \mathbb{E}_{\substack{t\sim \mathrm{Uni}(0,T)\\ \boldsymbol{x}^{1:D}\sim q_{t|0}(\cdot|\boldsymbol{x}_0^{1:D})}}&\Big[\sum_{d=1}^{D} r_t^d(\boldsymbol{x}^d|\cdot)^\top g_t^{\theta,d}(\cdot|\boldsymbol{x}^{1:D})\Big] -\\
T\ \mathbb{E}_{\substack{t\sim \mathrm{Uni}(0,T)\\ \boldsymbol{x}^{1:D}\sim q_{t|0}(\cdot|\boldsymbol{x}_0^{1:D})\\ \boldsymbol{z}^{1:D}\sim S_t(\cdot|\boldsymbol{x}^{1:D})}}&\Big[\frac{1}{\mathcal{M}_{S_t}(\boldsymbol{z}^{1:D}|\boldsymbol{x}_0^{1:D})}\sum_{d=1}^{D}\frac{\mathbf{1}^\top[q_{t|0}(\cdot|\boldsymbol{x}_0^d)\odot r_t^d(\boldsymbol{z}^d|\cdot)\odot\log g_t^{\theta,d}(\cdot|\boldsymbol{z}^{1:D})]}{q_{t|0}(\boldsymbol{z}^d|\boldsymbol{x}_0^d)}\Big]
\end{aligned}$$

Notice that the second part of the loss can rewritten to

$$\begin{aligned}
&\mathbb{E}_{\substack{t\sim \mathrm{Uni}(0,T)\\ \boldsymbol{z}^{1:D}\sim \tilde{q}_{t|0}(\cdot|\boldsymbol{x}_0^{1:D})}}\Big[\frac{1}{\mathcal{M}_{S_t}(\boldsymbol{z}^{1:D}|\boldsymbol{x}_0^{1:D})}\sum_{d=1}^{D}\frac{\mathbf{1}^\top[q_{t|0}(\cdot|\boldsymbol{x}_0^d)\odot r_t^d(\boldsymbol{z}^d|\cdot)\odot\log g_t^{\theta,d}(\cdot|\boldsymbol{z}^{1:D})]}{q_{t|0}(\boldsymbol{z}^d|\boldsymbol{x}_0^d)}\Big]\\
=&\mathbb{E}_t\sum_{\boldsymbol{z}^{1:D}}\Big[\frac{\tilde{q}_{t|0}(\boldsymbol{z}^{1:D}|\boldsymbol{x}_0^{1:D})}{\mathcal{M}_{S_t}(\boldsymbol{z}^{1:D}|\boldsymbol{x}_0^{1:D})}\sum_{d=1}^{D}\frac{\mathbf{1}^\top[q_{t|0}(\cdot|\boldsymbol{x}_0^d)\odot r_t^d(\boldsymbol{z}^d|\cdot)\odot\log g_t^{\theta,d}(\cdot|\boldsymbol{z}^{1:D})]}{q_{t|0}(\boldsymbol{z}^d|\boldsymbol{x}_0^d)}\Big](\text{now take Eq. (92) into})\\
=&\mathbb{E}_t\sum_{\boldsymbol{z}^{1:D}}\Big[q_{t|0}(\boldsymbol{z}^{1:D}|\boldsymbol{x}_0^{1:D})\sum_{d=1}^{D}\frac{\mathbf{1}^\top[q_{t|0}(\cdot|\boldsymbol{x}_0^d)\odot r_t^d(\boldsymbol{z}^d|\cdot)\odot\log g_t^{\theta,d}(\cdot|\boldsymbol{z}^{1:D})]}{q_{t|0}(\boldsymbol{z}^d|\boldsymbol{x}_0^d)}\Big]\\
=&\mathbb{E}_{\substack{t\sim \mathrm{Uni}(0,T)\\ \boldsymbol{z}^{1:D}\sim q_{t|0}(\cdot|\boldsymbol{x}_0^{1:D})}}\Big[\sum_{d=1}^{D}\frac{\mathbf{1}^\top[q_{t|0}(\cdot|\boldsymbol{x}_0^d)\odot r_t^d(\boldsymbol{z}^d|\cdot)\odot\log g_t^{\theta,d}(\cdot|\boldsymbol{z}^{1:D})]}{q_{t|0}(\boldsymbol{z}^d|\boldsymbol{x}_0^d)}\Big]
\end{aligned} \tag{93}$$

Taking Eq. (93) back into Eq. (28) we can get the following further simplified loss that only have a single forward pass of the model:

$$T\,\mathbb{E}_{\substack{t\sim\text{Uni}(0,T)\\ \boldsymbol{x}^{1:D}\sim q_{t|0}(\cdot|\boldsymbol{x}_0^{1:D})}}\Big[\sum_{d=1}^{D} r_t^d(\boldsymbol{x}^d|\cdot)^\top g_t^{\theta,d}(\cdot|\boldsymbol{x}^{1:D}) - \sum_{d=1}^{D}\frac{\mathbf{1}^\top[q_{t|0}(\cdot|\boldsymbol{x}_0^d)\odot r_t^d(\boldsymbol{x}^d|\cdot)\odot\log g_t^{\theta,d}(\cdot|\boldsymbol{x}^{1:D})]}{q_{t|0}(\boldsymbol{x}^d|\boldsymbol{x}_0^d)}\Big]$$

$$= T\,\mathbb{E}_{\substack{t\sim\text{Uni}(0,T)\\ \boldsymbol{x}^{1:D}\sim q_{t|0}(\cdot|\boldsymbol{x}_0^{1:D})}}\Big[\sum_{d=1}^{D} r_t^d(\boldsymbol{x}^d|\cdot)^\top\Big(g_t^{\theta,d}(\cdot|\boldsymbol{x}^{1:D}) - \frac{q_{t|0}(\cdot|\boldsymbol{x}_0^d)\odot\log g_t^{\theta,d}(\cdot|\boldsymbol{x}^{1:D})}{q_{t|0}(\boldsymbol{x}^d|\boldsymbol{x}_0^d)}\Big)\Big] \tag{94}$$

■

9.16 Continuous-Time MCMC Sampling Corrector & Noise Scheduling

Campbell et al. (2022) show that the MCMC for discrete data can be done by a predictor step to simulate $\widehat{r}_t^\theta$ and a corrector step using $r_t + \widehat{r}_t^\theta$. We extend this result with improved derivation and show that, thanks to the shared parameterized form, *both* discrete- and continuous-time discrete diffusion can leverage the same transition probability calculation (see Eq. (100)), leading to a shared MCMC scheme, with detailed derivation provided in Appendix 9.16.

9.16.1 The MCMC Sampling Corrector

Song et al. (2021b) introduced a predictor-corrector step to further improve the quality of generated data based on score-based Markov Chain Monte Carlo (MCMC) for continuous-time diffusion over continuous distribution. Campbell et al. (2022) showed that there is a similar MCMC based corrector that can be used for CTMC to improve reverse sampling at any time t. Although we use different reverse sampling than Campbell et al. (2022), the similar corrector step can also be developed to improve the quality of reverse sampling introdued in Section 4.4. In this section, we derive the corrector formally and simplify it based the multi-element formulations summarized in Eq. (100).

Formally, at any time t, Campbell et al. (2022) proved that a *time-homogeneous* CTMC with transition rate being $c_t := r_t + \hat{r}_t$ has its stationary distribution being $q_t(\mathbf{x}_t^{1:D})$. To avoid ambiguity, we use $n \in [0, +\infty)$ as the time variable for that CTMC with stationary distribution $q_t(\mathbf{x}_t^{1:D})$. Then for any sample $\boldsymbol{z}_t^{1:D}$ generated from reverse sampling process at time t, we can push it closer to the target marginal distribution $q_t(\mathbf{x}_t^{1:D})$ by sampling from the corrector CTMC with initial value being $\boldsymbol{z}_t^{1:D}$, named as $\boldsymbol{z}_{t,n=0}^{1:D}$. Let N be the maximum time allocated in the corrector CTMC, then after the corrector step $\boldsymbol{z}_{t,n=N}^{1:D}$ is used to replace the original $\boldsymbol{z}_t^{1:D}$.

We now introduce how to sampling from the CTMC. Let $\triangle n$ be the time incremental for each sampling step of the corrector CTMC. Solving the Kolmogorov forward equation of this time-homogeneous CTMC can derive the transition probability at any time n as

$$\forall n \text{ and } \triangle n,\ p_{n+\triangle n|n}(\boldsymbol{y}^{1:D}|\boldsymbol{x}^{1:D}) = \exp(\triangle n \cdot C_t^{1:D})[\boldsymbol{x}^{1:D}, \boldsymbol{y}^{1:D}] \tag{95}$$

Where C_t is the transition rate matrix of the corrector CTMC, and $\exp(\triangle n \cdot C_t^{1:D})$ is the transition probability matrix at time n. Notice that this matrix exponential does not have

analytical formulation. Instead, we propose to control $\triangle n$ to be small enough such that

$$\exp(\triangle n \cdot C_t^{1:D}) \approx I + \triangle n \cdot C_t^{1:D} \tag{96}$$

Then taking it back we can obtain

$$\begin{aligned} p_{n+\triangle n|n}(\boldsymbol{y}^{1:D}|\boldsymbol{x}^{1:D}) &\approx \delta_{\boldsymbol{x}^{1:D},\boldsymbol{y}^{1:D}} + \triangle n \cdot c_t^{1:D}(\boldsymbol{y}^{1:D}|\boldsymbol{x}^{1:D}) \\ &= \delta_{\boldsymbol{x}^{1:D},\boldsymbol{y}^{1:D}} + \triangle n \sum_{d=1}^{D} \Big[r_t^d(\boldsymbol{y}^d|\boldsymbol{x}^d) + r_t^d(\boldsymbol{x}^d|\boldsymbol{y}^d) \cdot g_t^d(\boldsymbol{y}^d|\boldsymbol{x}^{1:D}) \Big] \delta_{\boldsymbol{x}^{\setminus d},\boldsymbol{y}^{\setminus d}} \end{aligned} \tag{97}$$

Instead of sampling all elements jointly, we propose to sample each element of the object independently from their individual marginal distribution, which can be analytically formulated as

$$\begin{aligned} p_{n+\triangle n|n}(\boldsymbol{y}^d|\boldsymbol{x}^{1:D}) &= \sum_{\boldsymbol{y}^{\setminus d}} p_{n+\triangle n|n}(\boldsymbol{y}^{1:D}|\boldsymbol{x}^{1:D}) \\ &= \triangle n \Big[r_t^d(\boldsymbol{y}^d|\boldsymbol{x}^d) + r_t^d(\boldsymbol{x}^d|\boldsymbol{y}^d) \cdot g_t^d(\boldsymbol{y}^d|\boldsymbol{x}^{1:D}) \Big] \text{ if } \boldsymbol{y}^d \neq \boldsymbol{x}^d \\ &= \triangle n (\boldsymbol{y}^d)^\top \Big[r_t^d(\cdot|\boldsymbol{x}^d) + r_t^d(\boldsymbol{x}^d|\cdot) \odot g_t^d(\cdot|\boldsymbol{x}^{1:D}) \Big] \text{ if } \boldsymbol{y}^d \neq \boldsymbol{x}^d \\ &= \triangle n \beta(t) (\boldsymbol{y}^d)^\top \Big[\boldsymbol{m}^d - \boldsymbol{x}^d + (\langle \boldsymbol{x}^d, \boldsymbol{m}^d \rangle \mathbf{1} - \boldsymbol{x}^d) \odot g_t^d(\cdot|\boldsymbol{x}^{1:D}) \Big] \text{ if } \boldsymbol{y}^d \neq \boldsymbol{x}^d \\ &= \triangle n \beta(t) (\boldsymbol{y}^d)^\top \Big[\boldsymbol{m}^d + \langle \boldsymbol{x}^d, \boldsymbol{m}^d \rangle g_t^d(\cdot|\boldsymbol{x}^{1:D}) \Big] \text{ if } \boldsymbol{y}^d \neq \boldsymbol{x}^d \\ &\approx \triangle n \beta(t) (\boldsymbol{y}^d)^\top \Big[\boldsymbol{m}^d + \langle \boldsymbol{x}^d, \boldsymbol{m}^d \rangle g_t^{\theta,d}(\cdot|\boldsymbol{x}^{1:D}) \Big] \text{ if } \boldsymbol{y}^d \neq \boldsymbol{x}^d \end{aligned} \tag{98}$$

Now we define the notation $\overline{p_{\triangle n}(\cdot|\boldsymbol{x}^{1:D})}$ to derive the distributional form of $p_{n+\triangle n|n}(\boldsymbol{y}^d|\boldsymbol{x}^{1:D})$.

$$\overline{p_{\triangle n}^{\theta,d}(\cdot|\boldsymbol{x}^{1:D})} := \triangle n \beta(t) \Big[(2 - \frac{\overline{\alpha}_{t|0} \langle f_t^{\theta,d}(\boldsymbol{x}^{1:D}), \boldsymbol{x}^d \rangle}{\overline{\alpha}_{t|0} + (1 - \overline{\alpha}_{t|0}) \langle \boldsymbol{x}^d, \boldsymbol{m}^d \rangle}) \boldsymbol{m}^d + \frac{\overline{\alpha}_{t|0}}{1 - \overline{\alpha}_{t|0}} f_t^{\theta,d}(\boldsymbol{x}^{1:D}) \Big] \odot (\mathbf{1} - \boldsymbol{x}^d) \tag{99}$$

With the above notation, the sampling probability can be further simplified as

$$p_{n+\triangle n|n}(\boldsymbol{y}^d|\boldsymbol{x}^{1:D}) = \mathrm{Cat}\Big(\boldsymbol{y}^d; \overline{p_{\triangle n}^{\theta,d}(\cdot|\boldsymbol{x}^{1:D})} + (1 - \mathbf{1}^\top \overline{p_{\triangle n}^{\theta,d}(\cdot|\boldsymbol{x}^{1:D})}) \boldsymbol{x}^d \Big) \tag{100}$$

Notice that $\triangle n$ should be set small enough such that $\mathbf{1}^\top \overline{p_{\triangle n}^{\theta,d}(\cdot|\boldsymbol{x}^{1:D})} \leq 1$. This condition can be used to derive $\triangle n$ dynamically. In practice, we can also easily clip the scale of $\overline{p_{\triangle n}^{\theta,d}(\cdot|\boldsymbol{x}^{1:D})}$ to 1 when $\mathbf{1}^\top \overline{p_{\triangle n}^{\theta,d}(\cdot|\boldsymbol{x}^{1:D})} > 1$ to prevent illness condition. Intuitively, $1 - \mathbf{1}^\top \overline{p_{\triangle n}^{\theta,d}(\cdot|\boldsymbol{x}^{1:D})}$ defines the keeping rate of the d-th element during correction step, and it should be larger with increasing t and n during the reverse sampling and correction period.

9.16.2 Noise Scheduling in Training

Notice that $\beta(t) = -\frac{1}{\overline{\alpha}_t} \cdot \frac{d\overline{\alpha}_t}{dt}$ as $\overline{\alpha}_{t|s} = \exp(-\int_s^t \beta(a) da)$. We now first present a general way to design the scheduling of $\overline{\alpha}_t$ based on Chen (2023a). For any continuous function $h(t)$, we define $\overline{\alpha}_t$ as the follows that satisfies $\overline{\alpha}_0 = 1$ and $\overline{\alpha}_T = 0$:

$$\overline{\alpha}_t = \frac{h(T) - h(t)}{h(T) - h(0)} \tag{101}$$

Algorithm 3 The MCMC correcting algorithm at time t

Input: The sample at time t from reverse sampling, $\boldsymbol{z}_t^{1:D}$; step size $\triangle n$; learned f_t^θ; total steps N.
Initialize $\boldsymbol{x}_{t,0}^{1:D} \leftarrow \boldsymbol{z}_t^{1:D}$
for i from 1 to N **do**
  Compute $p_{\triangle n}^{\theta,d}(\cdot|\boldsymbol{x}_{t,i-1}^{1:D})$ from Eq. (99);
  $\forall d$, Draw $\boldsymbol{x}_{t,i}^d \sim \text{Cat}\Big(\cdot\,; \overline{p_{\triangle n}^{\theta,d}(\cdot|\boldsymbol{x}_{t,i-1}^{1:D})} + (1 - \mathbf{1}^\top \overline{p_{\triangle n}^{\theta,d}(\cdot|\boldsymbol{x}_{t,i-1}^{1:D})})\boldsymbol{x}_{t,i-1}^d \Big)$
end for
Output: the improved (corrected) sample $\boldsymbol{x}_{t,N}^{1:D}$

We can easily derive that

$$\beta(t) = -\frac{1}{\overline{\alpha}_t} \cdot \frac{h'(t)}{h(0) - h(T)} = \frac{h'(t)}{h(T) - h(t)} \tag{102}$$

$$\overline{\alpha}_{t|s} = \exp(-\int_s^t \frac{dh(t)}{h(T) - h(t)}) = \exp(\int_s^t \frac{d(h(T) - h(t))}{h(T) - h(t)}) = \frac{h(T) - h(t)}{h(T) - h(s)} \tag{103}$$

Based on the above general formulation, we now present some widely used noise schedules

Table 9: Widely used noise scheduling with maximum time T

Type	Param.	Reference	$h(t)$	$\overline{\alpha}_t$	$\beta(t)$	$\overline{\alpha}_{t\|s}$
Cosine	$a = 0.008$	(Hoogeboom et al., 2021a)	$\cos(\frac{t/T+a}{1+a}\frac{\pi}{2})$	$\frac{h(t)}{h(0)}$	$\frac{\pi \tan(\frac{t/T+a}{1+a}\frac{\pi}{2})}{2T(1+a)}$	$\frac{h(t)}{h(s)}$
Linear	-	(Ho et al., 2020a)	t	$1 - \frac{t}{T}$	$\frac{1}{T-t}$	$\frac{T-t}{T-s}$
Exp.	a, b	(Campbell et al., 2022)	-	$\exp(Ta(1 - b^{\frac{t}{T}}))$	$ab^{\frac{t}{T}} \log b$	$\exp(Ta(b^{\frac{s}{T}} - b^{\frac{t}{T}}))$

9.17 Experiment Details

9.17.1 Baselines

For music and image generation tasks, we compare USD3 and variants to latest SOTA discrete diffusion models in the literature: D3PM in Austin et al. (2021), RDM in Zhang et al. (2023) (discrete-time), τ-LDR-0 in Campbell et al. (2022) , SDDM in Sun et al. (2023) and SEDD in Lou et al. (2024) (continuous time). For text generation task, we compare our model to both diffusion, autoregressive and flow models. The baselines are taken from Graves et al. (2024); Austin et al. (2021); Lou et al. (2024). In addition, IAF/SCF results and configurations are shown in Ziegler and Rush (2019), the Autoregressive Argmax Flow in Hoogeboom et al. (2022), the Discrete Flow in Tran et al. (2019), Multinomial Diffusion in Hoogeboom et al. (2021b), and MAC in Shih et al. (2022).

9.17.2 List of Hyperparameters

Refer to Table 11, Table 10, and Table 12 for a list of detailed hyperparameters used in experiment.

Table 10: Summary of Architectures, Training Parameters, Diffusion Noise Schedulers, and Noise Types for VQCIFAR10 Image Generation

Category	**Details**
Architecture (Diffusion Transformer)	
Number of Layers	12
Attention Heads per Layer	12
Input Dimension	768
Fully-connected Layer Dimension	3072
Activation Function	GeLU
Dropout	0.1
Training Parameters	
Learning Rate	2×10^{-4}
Warmup Steps	5000
Learning Rate Scheduler	Cosine decay
Total Training Steps	1 million
EMA Decay Factor	0.999
Gradient Clipping Norm	1.0
Diffusion Noise Schedulers	
Discrete-Time Noise Scheduler	Cosine scheduler with $\alpha = 0.008$
Continuous-Time Noise Scheduler	Constant scheduler with 0.007
Noise Type	
USD3 & All baselines	Uniform Noise

9.17.3 Evaluation Metrics

Among the test music sequences, we find that 124 of the evaluation sequences share the *same* first 32 notes as one or more training sequences. Such repetition gives us a unique opportunity to investigate what-we-call "parroting"—the phenomenon that models are simply re-playing the exact training data during generation, i.e. the notion of memorizing/overfitting, leading to lack of diversity in generated samples. To that end, we create Piano-P, a subset consisting only of these 124 evaluation sequences paired with their first-32-note-matching training sequences. Among Piano-P, 20 samples can be found to contain the same first 32 notes with 2 samples from the training sequences, three of them contain the same first 32 notes with 4 training samples each, one of them shares the same with 6 training samples, and one shares the same with 8 training samples. If the same 32 notes appear both in the training samples and as quests for the model to provide the inferences, the model is likely to directly memorizing the remaining 224 notes from the training set, and "parroting" music sequences according to the training samples.

We provide a detailed description of metrics we used in music generation task:

- 1-gram Hellinger Distance (↓) and 1-gram Proportion of Outliers (↓): for these two metrics, we following the same evaluation as described in Campbell et al. (2022) and Chen (2023a).

Table 11: Training Details for USD3 on the Piano Dataset

Category	Details
Architecture (Diffusion Transformer)	
Number of Layers	12
Attention Heads per Layer	12
Input Dimension	768
Fully-connected Layer Dimension	3072
Activation Function	GeLU
Dropout	0.1
Training Parameters	
Learning Rate	2×10^{-4}
Warmup Steps	5000
Learning Rate Scheduler	Cosine decay
EMA decay factor	0.999
Total Training Steps	800,000
Gradient Clipping Norm	1.0
Diffusion Noise Scheduler	
Discrete-Time Noise Scheduler	Cosine scheduler with $\alpha = 0.008$
Continuous-Time Noise Scheduler	Cosine scheduler with $\alpha = 0.008$
Noise Type	
USD3 & All baselines	Uniform Noise

Table 12: Training Details for USD3 on Text8 Generation

Category	Details
Architecture (Diffusion Transformer)	
Number of Layers	12
Attention Heads per Layer	12
Attention Dimension	768
Fully-connected Layer Dimension	3072
Training Parameters	
Learning Rate	5×10^{-4}
Total Training Steps	1 million
Batch Size	512
Diffusion Noise Scheduler	
Scheduler Type	Log-linear
Scheduler Parameter	$\epsilon = 1 \times 10^{-3}$
Noise Type	
USD3	Absorbing noise

- $\{2, 3\}$-gram Hellinger Distance (↓) and $\{2, 3\}$-gram Proportion of Outliers (↓): similar to n-gram models, we first convert the music sequences into tuples of neighboring nodes. Then for Hellinger Distance, we compute the distance of the empirical probabilities of conditional generated samples to the ground truth samples. The empirical probabilities

are constructed based on the histograms of the neighboring tuples, with bins being all possible $\{2, 3\}$-gram nodes. Similarly, for the Proportion of Outliers, we count the fractions of newly appeared tuples that are not seen in the training samples. With these metrics, we are able to capture the sequence information instead of just measuring the single node distributions.

- Diverse Edit Distance (↑): which accounts for the creativity/novelty of generated samples across multiple generation runs. For the conditionally generated music samples given the same first 32 notes, we calculate the edit distance, which is the minimum number of single-character edits (insertions, deletions, or substitutions) required to change one music sequence into the other, between each two of the generation samples. The mean and standard deviation are obtained for all edit distances between pairs. The higher the diverse edit distance, the further apart the music sequences are and more creativity are enforced in the generation process. However, there is also a trade-off between diverse edit distance and other accuracy measurements when the model processes large uncertainty about the underlying distribution.

- Train-to-Test Ratios (↑) for $\{1, 2, 3\}$-gram Hellinger as well as Proportion of Outliers, which compare the weighted distance of a generated sample to its evaluation (test) vs. training sequence that share the same first 32 notes. We evaluate the ratios only for Piano-P. Denote the evaluation ground truth set in Piano-P as tr , the corresponding set of training samples as ts, and conditionally generated samples as gs. For the selected distance metrics *dist()* (from n-gram Hellinger or n-gram Proportion of Outliers), we calculate the ratio as: $\frac{1}{dist(tr,gs)+dist(ts,gs)} * \frac{dist(tr,gs)}{dist(ts,gs)}$. Such ratio measures quantify the extent of "parroting" in Piano-P. The larger the ratio, the more equally distant the generated examples is from its training and evaluating set, and the less "parroting" occurs by simply memorizing all training sequences. We apply an additional coefficient to the ratio such that it will also penalize the models that provide unrealistic examples that do not conform both training and evaluation distributions. Such ratios quantify the extent of "parroting"; a smaller ratio indicates a higher (lower) degree of memorization (generalization).

9.17.4 VQCIFAR10 Generation

In our setup for CIFAR10, a VQGAN encodes an image of shape $H \times W \times 3$ to $(\frac{H}{4} \times \frac{W}{4})$ tokens with vocabulary size of 512. For the encoder, we use three convolutional blocks, with filter sizes of 64,128 and 256, and an average pooling between each blocks. After an image is encoded, the output is mapped to a token index with a codebook of 512×256. For VQGAN loss objective, the additional GAN loss and perceptual loss are added with weight 0.1. To train a general VQGAN model that allows us to embed CIFAR10 images without overfitting, we apply random flipping and cropping to the 64×64 ImageNet dataset (Deng et al., 2009). The VQGAN is trained with Adam optimizer ($\beta_1 = 0$, $\beta_2 = 0.99$), and the learning rate is linearly warmed up to the peak of $1e^{-4}$ and then drops with cosine decay.

After VQGAN is trained, we freeze the VQGAN and apply encoder to the CIFAR10 images to create 8×8 latent codes. The latent codes then flattened to a vector of size 64 as the input of the diffusion model. Before we evaluate our diffusion methods, we will feed the

generated latent codes back to the VQGAN decoder to reconstruct a sample in the image spaces. We test the effectivenss of VQGAN by trying to reconstruct the CIFAR10 dataset.

9.17.5 Baseline Training Configurations on Music Generation

D3PM and **τ-LDR-0**: we follow a similar transformer model as utilized by in τ-LDR-0 (Campbell et al., 2022). The sequence transformer is composed of several encoder blocks, where for each internal block, time is fed into the block through a FiLM layer and added to the input. The transformer contains 6 encoder blocks, each containing 16 attention heads, input dimension of 1024 and MLP dimension of 4096. For each attention block, we apply RELU for activation. At the output of self-attention layer and fully-connected layer, a dropout of 0.1 is applied. After obtaining the final embedding of each token, we feed that into a 1-block ResNet to obtain the predicted logits. For calculating the loss of both methods, we follow the previous literature and give 0.001 to CE loss, and 1 to the ELBO loss. In diffusion process, the noise is a cosine noise scheduling with $\alpha = 0.008$.

RDM: we use the official implementation of loss function in reparam multinomial diffusion[4].We keep the same neural network architecture, and apply reweighting = 1.0, with no label smoothing, and let sampling strategy be cmlm-dep. We remove the padding functions and sampling temperatures when adapting RDM from language tasks to music and image generation.

SEDD: we follow the official code implementation with uniform noise and loss functions[5]. SEDD's base architecture is a Diffusion Transformer (Peebles and Xie, 2023). To keep approximately the same parameter counts, the transformer has 12 layers, each layer is 12 heads, with input dimension of 768 and max MLP dimension of 3072. The GeLU activation is used, and dropout is set to 0.1. We add an additional masking to the output (the same masking as in SEDD's diffusion transformer), since SEDD models the ratio of two probabilities instead of direct log probabilities. We use log-linear noise schedules with $1e^{-3}$.

SDDM: SDDM designs a specific transformer structure that enables its diffusion loss calculation without information leakage. We use the experiment configurations as given in the paper: the backbone structure is a hollow transformer (Sun et al., 2023). Each transformer block has 6 layers with embedding size of 256, 8 attention heads and hidden dimension of 2048. The batch size is 64 and the number of training steps is 2 million. The weight decay is set to $1e^{-6}$. The learning rate is constant $1e^{-3}$. For diffusion, SDDM adopts the constant noise scheduler with a uniform rate constant of 0.04.

9.17.6 Baseline Training Configurations on Image Generation

τ-LDR-0 and **D3PM**: For image generation task, we use the same transformer as described in Campbell et al. (2022). The model is a 12 layer transformer, where each layer has 16 attention heads, input embedding of 768 and a hidden layer size of 3072 for the MLPs. We use ReLU for activation. At the output of each internal block, a dropout of 0.1 is applied. The time is input into the network through FiLM layers before the self-attention block. We apply a hybrid loss with cross entropy as a directly supervision, added with 0.001 CE loss. The τ-LDR-0 uses a constant rate noise scheduler of 0.007, while D3PM uses cosine

4. https://github.com/HKUNLP/reparam-discrete-diffusion
5. https://github.com/louaaron/Score-Entropy-Discrete-Diffusion

scheduler with $\alpha = 0.008$. We train all models in parallel on 2 A6000 GPUs. We follow the previous literature and give 0.001 to CE loss, and 1 to the ELBO loss.

RDM: We use the same architecture as described by the abvoe two methods. We apply reweighting = 1.0, with no label smoothing, and let sampling strategy be cmlm-dep. We apply cosine scheduler with uniform noise.

SEDD: We apply the Diffusion Transformer to the SEDD calculation. The Diffusion Transformer contains 12 layers, and each attention block is constituted with 12 attention heads, input dimension of 768 and fullly-connected layer's dimension of 3072. The GeLU activation is used and the dropout is set to 0.1. We use log-linear noise schedules with $1e^{-3}$.

SDDM: the model uses a masked modeling, where the backbone neural network is BERT-based, with 12 layers of transformers. Each layer has 12 attention heads, embedding size of 768 and hidden layer size of 3072 for MLPs. After obtaining the final embedding of each token, the output is fed into a 2-block ResNet to acquire the predicted logits. For diffusion, SDDM uses a constant uniform rate of 0.007 as the noise scheduler in the forward process. In Sun et al. (2023), the number of training steps is set to $700,000$, where the learning rate is warmed up to $1e^{-4}$ during the first 3% steps, and then decays to 0 in a linear schedule. We extended the training to 1 million steps, but did not observe improved performance.

9.17.7 Full Experiment Results on Music Generation

We present the full evaluation metrics and results, including mean and standard deviation, for Lakh Pianoroll music generation tasks, as shown in Table 13.

Table 13: Metrics comparing generated conditional samples and evaluating ground truths for Lakh Pianoroll. For each of n-gram Hellinger Distance (ng.-Hellinger) and Proportion of Outliers (ng.-Prop. Outlier) and Diverse Edit Distance metrics, we show mean ± std with respect to 3 generated samples. The top two are highlighted by **First**, **Second**.

	Method	1g.-Hellinger(↓)	2g.-Hellinger(↓)	3g.-Hellinger(↓)	1g.-Prop.Out.(↓)	2g.-Prop.Out.(↓)	3g.-Prop.Out.(↓)	Edit Distance(↑)
Discrete-time	D3PM	0.3982±0.0004	0.5303±0.0038	0.5918±0.0046	0.1209±0.0008	0.2532±0.0053	0.3790±0.0059	**0.2950±0.0040**
	RDM	0.4123±0.0012	0.5964±0.0021	0.6198±0.0039	0.1401±0.0011	0.2459±0.0023	0.3864±0.0030	**0.2891±0.0052**
	USD3-CE	0.3677±0.0003	0.4493±0.0018	0.5236±0.0018	0.1068±0.0012	0.1669±0.0012	0.2455±0.0017	0.0482±0.0012
	USD3-ELBO	0.3676±0.0007	0.4483±0.0006	0.5229±0.0006	0.1142±0.0007	0.1670±0.0003	0.2577±0.0005	0.0485±0.0010
	USD3	**0.3672±0.0005**	0.4487±0.0006	0.5234±0.0004	0.1126±0.0005	0.1757±0.0004	0.2562±0.0004	0.0484±0.0023
	USD3*	0.3680±0.0007	0.4485±0.0006	0.5251±0.0006	0.1110±0.0005	0.1749±0.0009	0.2538±0.0008	0.0502±0.0011
Continuous-time	SDDM	0.3759±0.0020	0.4856±0.0015	0.5773±0.0009	0.1101±0.0015	0.2059±0.0005	0.3009±0.0017	0.0612±0.0024
	τ-LDR-0	0.3796±0.0009	0.4681±0.0008	0.5710±0.0007	0.1149±0.0008	0.2078±0.0014	0.3202±0.0014	0.0504±0.0063
	SEDD	0.3715±0.0012	0.4472±0.0008	0.5250±0.0006	**0.1052±0.0005**	0.1723±0.0004	0.2509±0.0008	0.0501±0.0012
	USD3-CE	**0.3658±0.0004**	**0.4465±0.0003**	**0.5201±0.0004**	**0.1043±0.0005**	**0.1630±0.0004**	**0.2414±0.0007**	0.0483±0.0009
	USD3-ELBO	0.3673±0.0006	0.4474±0.0003	0.5204±0.0006	0.1125±0.0001	0.1685±0.0012	0.2455±0.0001	0.0504±0.0011
	USD3	0.3676±0.0002	**0.4467±0.0013**	**0.5198±0.0012**	0.1097±0.0014	**0.1668±0.0015**	**0.2423±0.0018**	0.0498±0.0020

9.17.8 Study of "Parroting" with Ratio Metrics

As shown in Table 14, we find that models with combined losses, USD3 and USD3*, achieve higher Train-to-Test ratio than pure USD3-CE or USD3-ELBO, suggesting that the combination of two losses can alleviate overfitting, i.e. "parroting" the training data. Overall, τ-LDR-0 is a strong competitor in n-gram Hellinger Train-to-Test ratios, suggesting

that its ELBO focused loss optimization can also alleviate parroting and generate diverse samples.

Table 14: Train-to-test Ratio Metrics (↑) for Lakh Pianoroll, using evaluation sequences fron Piano-P and the training sequences. The higher ratios indicate less "parroting" phenomenon and better generalization ability. Mean and std are calculated across 3 generated samples.The top two are highlighted by **First**, **Second**.

	Method	1g.-Hellinger Ratio	2g.-Hellinger Ratio	3g.-Hellinger Ratio	1g.-Prop.Out. Ratio	2g.-Prop.Out. Ratio	3g.-Prop.Out. Ratio
Discrete-time	D3PM	1.7812±0.0500	1.3624±0.0411	1.1307±0.0304	5.6237±0.1463	5.7052±0.1620	2.2216±0.6160
	RDM	1.7534±0.0928	1.4023±0.0071	1.2023±0.0392	5.9981±0.0497	5.6050±0.5302	2.5211±0.5672
	USD3-CE	1.9234±0.0492	1.4722±0.0200	1.1545±0.0054	6.9231±0.6659	7.3224±1.4502	2.7308±0.5980
	USD3-ELBO	1.9946±0.0202	1.5159±0.0072	1.2002±0.0109	7.0462±0.4290	**7.9642±1.2883**	2.8634±0.4701
	USD3	**2.0662±0.0413**	1.5601±0.0380	1.2019±0.0500	**8.6303±0.6402**	7.5278±0.5692	**3.2584±0.3701**
	USD3*	1.9701±0.0233	**1.5985±0.0201**	1.9161±0.0122	7.6454±0.7302	7.6239±1.1294	2.9542±0.5303
Continuous-time	SDDM	1.7872±0.0131	1.4516±0.0097	1.1812±0.0103	6.3019±0.2552	7.0654±0.5093	2.9014±0.0782
	τ-LDR-0	1.8648±0.0306	1.4726±0.0233	**1.2497±0.0185**	6.4586±0.3802	7.0864±0.2012	2.9651±0.1089
	SEDD	2.0112±0.0303	1.4898±0.0092	**1.2540±0.0152**	6.4632±0.5100	7.0603±0.4602	3.0200±0.3001
	USD3-CE	1.9832±0.0224	1.5033±0.0352	1.1540±0.0202	6.7542±0.2470	6.5462±0.7500	2.7324±0.6828
	USD3-ELBO	2.0650±0.0312	**1.5945±0.0122**	1.1804±0.0230	7.6452±0.4442	7.3620±1.1708	2.9041±0.5809
	USD3	**2.1010±0.0802**	1.5912±0.0650	1.1714±0.0235	**8.4320±0.7420**	**8.2954±1.8711**	**3.8010±0.2520**