Fast Sampling via Discrete Non-Markov Diffusion Models with Predetermined Transition Time

Zixiang Chen Huizhuo Yuan Yongqian Li Yiwen Kou Junkai Zhang Quanquan Gu
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90095
{chenzx19,hzyuan,yongqianl,evankou,jkzhang,qgu}@cs.ucla.edu

Abstract

Discrete diffusion models have emerged as powerful tools for high-quality data generation. Despite their success in discrete spaces, such as text generation tasks, the acceleration of discrete diffusion models remains under-explored. In this paper, we propose discrete non-Markov diffusion models (DNDM), which naturally induce the predetermined transition time set. This enables a training-free sampling algorithm that significantly reduces the number of function evaluations (i.e., calls to the neural network), making the sampling process much faster. Furthermore, we study the transition from finite to infinite step sampling, offering new insights into bridging the gap between discrete and continuous-time processes for discrete diffusion models. Extensive experiments on natural language generation and machine translation tasks demonstrate the superior performance of our method in terms of both generation speed and sample quality compared to existing methods for discrete diffusion models. Codes are available at https://github.com/uclaml/DNDM.

1 Introduction

Diffusion-based generative models, as first introduced by Sohl-Dickstein et al. (2015), have shown remarkable capabilities in generating high-quality samples across various domains, including images (Ho et al., 2020; Song and Ermon, 2020), audio (Chen et al., 2020; Kong et al., 2020), and videos (Ho et al., 2022). The diffusion model utilizes an innovative approach comprising a forward process that gradually transforms training data into pure noise and a reverse process that reconstructs clean data from the noise. Throughout the training phase, the model optimizes a neural network by minimizing an objective derived from maximum likelihood estimation. Once trained, the model can generate samples using various decoding strategies, including implicit dynamics (Song et al., 2020a), analytical processes (Bao et al., 2022), or differential equation solvers (Song et al., 2020b; Liu et al., 2022; Lu et al., 2022). In particular, Song et al. (2020a) introduced the denoising diffusion implicit model (DDIM), providing a non-Markov and de-randomized version of the Denoising Diffusion Probabilistic Model (DDPM) (Sohl-Dickstein et al., 2015; Ho et al., 2020), which enables faster generation of high-quality samples.

Although diffusion models were initially introduced for both discrete and continuous-state spaces (Sohl-Dickstein et al., 2015), these studies have largely focused on Gaussian diffusion processes in continuous-state spaces. Recently, Discrete Denoising Diffusion Probabilistic Models (D3PMs) (Austin et al., 2021) working in discrete-state spaces have gained increasing interest due to their applications in diverse areas such as text generation (Hoogeboom et al., 2021b), medical record generation (Ceritli et al., 2023), and protein design (Gruver et al., 2024). These models, which are distinct from their Gaussian counterparts, employ discrete noises, such as the multinomial distribution, for diffusion processes. Very recently, Zheng et al. (2023) introduced a reparameterized diffusion

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

model (RDM) that can improve sampling speed and sample quality in text generation tasks. However, their proposed algorithm is a training-based approach. Compared with diffusion models using Gaussian noise, discrete diffusion models remain under-studied, especially regarding training-free sampling acceleration.

In this work, we introduce a training-free approach aiming at enhancing the sampling speed of discrete diffusion models. This approach stems from a unique characteristic of discrete diffusion models: unlike continuous diffusion models, which typically employ Gaussian noise for data corruption (Ho et al., 2020; Song and Ermon, 2020; Song et al., 2020b,a), discrete diffusion models often use categorical white noises (Hoogeboom et al., 2021b; Austin et al., 2021; Zheng et al., 2023).

Table 1: Cross Comparison of Diffusion Models.

	Continuous	**Discrete**
Markov	DDPM (Sohl-Dickstein et al., 2015)	D3PM Austin et al. (2021)
Non-Markov	DDIM (Song et al., 2020a)	DNDM (Ours)

By delving into this special property, we develop a discrete non-Markov diffusion model, together with a design of accelerated algorithm. Notably, this new sampling technique does not require any modifications to the training objective of diffusion models and is, therefore, training-free. Our contributions are summarized as follows:

- We propose discrete non-Markov diffusion models (DNDM), which naturally induces a set of latent variables $\mathcal{T}$, termed as the *transition time set*. This key feature enables us to develop a training-free sampling algorithm that can accelerate a large family of discrete diffusion models. Importantly, DNDM preserves the essential properties of the original discrete diffusion model: for any diffusion trajectory $\{\mathbf{x}_t\}$ starting from real data $\mathbf{x}_0$, it provably maintains both the marginal distribution $q(\mathbf{x}_t)$ and the conditional distribution $q(\mathbf{x}_0|\mathbf{x}_t)$. Our method can accelerate the two most widely used discrete diffusion models: multinomial diffusion (Hoogeboom et al., 2021b) and absorbing diffusions (Austin et al., 2021). Similar to how DDIM introduces a de-randomized, faster sampling algorithm compared to DDPM in continuous space, DNDM achieves acceleration through a predetermined transition time set in discrete space (See Table 1).
- Based on the predetermined transition time set $\mathcal{T}$ in DNDM, we design an accelerated sampling algorithm that reduces the required number of neural network function evaluations. In a standard T time-step discrete diffusion process, while D3PM, including Multinomial (Ho et al., 2020) and absorbing state discrete sampling (Austin et al., 2021), requires evaluating the neural network function T times, our approach only requires $|\mathcal{T}|$ function evaluations, where $|\mathcal{T}|$ is the cardinality of the transition set $\mathcal{T}$. Moreover, $|\mathcal{T}|$ is provably less than T and approaches $O(1)$ as T goes to infinity. We provide both theoretical analysis and empirical experiments showing that the improvement in the number of function evaluations (NFE) is significant. Notably, our algorithm is about $3\times$ faster than baselines for $T = 50$ and about $30\times$ faster for $T = 1000$ while preserving the sample quality.
- To further illustrate the effectiveness of DNDM, we explore the limit as $T \to \infty$ and introduce an infinite-step sampling algorithm. With a pretrained neural network, we can generate an initial noise $\mathbf{x}_T$ and a transition time set $\mathcal{T} \subseteq [0, 1]$ with infinitesimal spacing, such that $|\mathcal{T}| = O(1)$. This enables the generation of the real data distribution with only $|\mathcal{T}|$ neural network evaluations. This study offers new insights into bridging the gap between discrete and continuous-time processes for discrete diffusion models.

Notation. We use $|\mathcal{T}|$ to denote the cardinality of the set $\mathcal{T}$ (excluding repeated elements). We use lowercase letters to denote scalars, boldface lowercase letters to denote vectors, and boldface uppercase letters to denote matrices. The notation $1 : N$ indicates the sequence from 1 through N. The symbol $\mathbf{q}$ designates the real distribution in a diffusion process, while $\mathbf{p}$ represents the distribution during sampling. With its success probability inside the parentheses, the Bernoulli distribution is denoted by $\mathrm{Bernoulli}(\cdot)$. We further use $\mathrm{Cat}(\mathbf{x}; \mathbf{p})$ to denote a categorical distribution over a one-hot row vector $\mathbf{x}$ with probabilities given by the row vector $\mathbf{p}$.

2 Background

In this section, we provide the background of discrete diffusion models. We begin by introducing the discrete Markov diffusion model, designed for handling categorical random variables. Specifically,

consider a diffusion model trying to generate distributions over a discrete random variable $\mathbf{x} \in \mathbb{R}^K$ that is one-hot encoded with K categories, i.e., $\mathbf{x}$ can be chosen as one of K categories, and for any $k \in [K]$, $\mathbf{x}$ is categorized as k if $\mathbf{x}$ aligns with the standard basis vector $\mathbf{e}_k$. The sequence $\{\mathbf{x}_t\}_{t=0}^T$ represents how this random variable changes over time $0 \leq t \leq T$, starting from an $\mathbf{x}_0 \in \mathbb{R}^K$ drawn from the real distribution $\mathbf{q}_{\text{data}}$. In this paper, we focus on the two most widely used D3PMs: multinomial diffusion (Hoogeboom et al., 2021b) and absorbing diffusions (Austin et al., 2021).

Forward Process. During the forward process, the real distribution $\mathbf{q}_{\text{data}}$ is gradually transformed into a noise distribution named $\mathbf{q}_{\text{noise}}$. The transformation occurs through T steps, with T intermediate latent variables $\mathbf{x}_1, \dots \mathbf{x}_T$ and update rules given by:

$$\mathbf{x}_t = b_t \mathbf{x}_{t-1} + (1 - b_t)\mathbf{w}_t, \qquad t = 1, \dots, T \tag{1}$$

Here b_t is randomly drawn from a Bernoulli distribution with parameter β_t, denoted by $b_t \sim \text{Bernoulli}(\beta_t)$, and $\mathbf{w}_t$ is randomly drawn from the noise distribution $\mathbf{q}_{\text{noise}}$, while for different t the samples are independent. In this work, we focus on cases where the noise $\mathbf{q}_{\text{noise}}$ can be either a uniform distribution over the vocabulary $\{1, 2, \dots, K\}$ (Hoogeboom et al., 2021b), or a point mass with all of the probability mass lying on an absorbing state (Austin et al., 2021). Following this notation, the process in (1) defines a Markov process characterized by the transition kernel

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \text{Cat}\big(\mathbf{x}_t; \mathbf{p} = \beta_t \mathbf{x}_{t-1} + (1 - \beta_t)\mathbf{q}_{\text{noise}}\big). \tag{2}$$

Moreover, the Markov chain property allows us to get samples $\mathbf{x}_{0:t}$ from $\mathbf{x}_0$ by multiplying the transition probabilities at each step as $p(\mathbf{x}_{1:t}|\mathbf{x}_0) = \prod_{i=1}^t q(\mathbf{x}_t|\mathbf{x}_{t-1})$. It further leads to the following marginal distribution.

$$q(\mathbf{x}_t|\mathbf{x}_0) = \text{Cat}\big(\mathbf{x}_t; \mathbf{p} = \alpha_t \mathbf{x}_0 + (1 - \alpha_t)\mathbf{q}_{\text{noise}}\big), \tag{3}$$

where $\alpha_t := \Pi_{s=1}^t \beta_s$ is determined by the sequence of β_t of our choice and decreases from 1 to 0.

Reverse Process. Given the forward Markov process, the reverse process can be derived by Bayes' rule (Hoogeboom et al., 2021b; Austin et al., 2021; Zheng et al., 2023). The conditional probability $q(\mathbf{x}_{t-1}|\mathbf{x}_0, \mathbf{x}_t)$ can be determined by $q(\mathbf{x}_{t-1}|\mathbf{x}_0, \mathbf{x}_t) = q(\mathbf{x}_t|\mathbf{x}_{t-1})q(\mathbf{x}_{t-1}|\mathbf{x}_0)/q(\mathbf{x}_t|\mathbf{x}_0)$. The reverse process can be used for synthetic data generation by sampling from the noise distribution q_{noise} and repeatedly applying a learned predictor (neural network) $p_{\boldsymbol{\theta}}(\cdot|\mathbf{x}_t)$ parameterized by $\boldsymbol{\theta}$:

$$p_{\boldsymbol{\theta}}(\mathbf{x}_T) = q_{\text{noise}}(\mathbf{x}_T), \qquad q_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t) = \int_{\widehat{\mathbf{x}}_0} q(\mathbf{x}_{t-1}|\mathbf{x}_t, \widehat{\mathbf{x}}_0) p_{\boldsymbol{\theta}}(\widehat{\mathbf{x}}_0|\mathbf{x}_t) d\widehat{\mathbf{x}}_0. \tag{4}$$

We note that the reverse process $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \widehat{\mathbf{x}}_0)$ is stochastic and thus requires function evaluation at every step.

Training the Neural Network. The neural network $p_{\boldsymbol{\theta}}(\cdot|\mathbf{x}_t)$ that predicts $\widehat{\mathbf{x}}_0$ is trained by maximizing the evidence lower bound (ELBO) (Sohl-Dickstein et al., 2015),

$$\begin{aligned}
\log p_\theta(\mathbf{x}_0) &\geq \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\Big[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\Big] d\mathbf{x}_{1:T} \\
&= \mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)}[\log p_{\boldsymbol{\theta}}(\mathbf{x}_0|\mathbf{x}_1)] - \sum_{t=2}^T \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}[\text{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)\|p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t)) \\
&\quad - \mathbb{E}_{q(\mathbf{x}_T|\mathbf{x}_0)}\text{KL}(q(\mathbf{x}_T|\mathbf{x}_0)\|p_{\boldsymbol{\theta}}(\mathbf{x}_T)),
\end{aligned} \tag{5}$$

Here KL denotes Kullback-Liebler divergence and the last term $\mathbb{E}_{q(\mathbf{x}_T|\mathbf{x}_0)}\text{KL}(q(\mathbf{x}_T|\mathbf{x}_0)\|q_{\text{noise}}(\mathbf{x}_T))$ equals zero. Building on this foundation, Austin et al. (2021) introduced an auxiliary denoising objective, which refines the data predictions $\mathbf{x}_0$ at each time step. Since this paper primarily focuses on reverse sampling, we leave detailed discussions of these losses to Appendix B.

3 Discrete Non-Markov Diffusion Models (DNDM)

3.1 Forward and Reverse Process

In this section, we introduce a non-Markov process such that the joint distribution of $(\mathbf{x}_0, \mathbf{x}_t)$ remains the same as the one defined with Markov process in Section 2. The new process aims to gradually

transform input data $\mathbf{q}_{\text{data}}$ to the noise distribution $\mathbf{q}_{\text{noise}}$ through T intermediate latent variables $\mathbf{x}_1, \dots \mathbf{x}_T$ with the following process:

$$\mathbf{x}_t = b_t \mathbf{x}_{t-1} + (1 - b_t)\mathbf{w}, \tag{6}$$

where b_t is independently drawn from the Bernoulli distribution $\text{Bernoulli}(\beta_t)$ and $\mathbf{w}$ is drawn from the noise distribution $\mathbf{q}_{\text{noise}}$. The only difference between (6) and (1) is that we replace $\mathbf{w}_t$ in (1) by $\mathbf{w}$, which is time-invariant during the diffusion. Therefore, the process in (6) becomes non-Markov since $q(\mathbf{x}_t|\mathbf{x}_{t-1}, \dots, \mathbf{x}_0)$ doesn't necessarily equals $q(\mathbf{x}_t|\mathbf{x}_{t-1})$. The following theorem shows that the conditional distribution $q(\mathbf{x}_t|\mathbf{x}_0)$ remains unchanged.

Theorem 3.1. *For the non-Markov process in* (6)*, we have*

$$q(\mathbf{x}_t|\mathbf{x}_0) = \text{Cat}\big(\mathbf{x}_t; \mathbf{p} = \alpha_t \mathbf{x}_0 + (1 - \alpha_t)\mathbf{q}_{\text{noise}}\big),$$

where $\alpha_t := \Pi_{i=1}^{s} \beta_s$ *is specified to decrease from* 1 *to* 0*.*

Using the Bayes' rule, we have $q(\mathbf{x}_0|\mathbf{x}_t) \propto q(\mathbf{x}_t|\mathbf{x}_0)q(\mathbf{x}_0)$. Consequently, the condtional distribution $q(\mathbf{x}_0|\mathbf{x}_t)$ remains consistent with the one induced by the process process in (1). Therefore, neural network $p_{\boldsymbol{\theta}}(\cdot|\mathbf{x}_t)$ trained by the Markov process in (1), remains applicable to our non-Markov process (6) (see Appendix B for detail).

Based on the discrete non-Markov diffusion model, we can give a simple characterization of the reverse process by introducing the transition time.

Definition 3.2. Transition time τ is the time that the token $\mathbf{x}_t$ transition from $\mathbf{x}_0$ to noise, i.e., $\tau := \min_t\{t|b_t = 0\}$.

Remark 3.3. The concept of transition time has also been introduced in Hoogeboom et al. (2021a). However, Hoogeboom et al. (2021a) restricts the transition time to be the first time of entering the absorbing state, which is only applicable to absorbing diffusion. Our definition is more general and applicable to discrete diffusion with various noise including multinomial diffusion.

Given the transition time τ, the forward process reduces to:

$$\mathbf{x}_t = \mathbb{1}(\tau > t)\mathbf{x}_0 + \mathbb{1}(\tau \leq t)\mathbf{w}, \tag{7}$$

which shows that the token will be a real token $\mathbf{x}_0$ before the time τ and will be the noise $\mathbf{w}$ after the transition time. Since token only get changed at the transition time τ, we can derive a reverse process based on (7),

$$\mathbf{x}_{t-1} = \mathbb{1}(\tau = t)\mathbf{x}_0 + \mathbb{1}(\tau \neq t)\mathbf{x}_t. \tag{8}$$

Therefore, the process in (8) is de-randomized given transition time τ. Specifically, after independently sampled transition times τ, $\mathbf{x}_{t-1}$ becomes deterministically known and fixed if we observe $\mathbf{x}_0$ and $\mathbf{x}_t$. It is also worth noting that given $\mathbf{x}_0$ and τ, the exact reverse process (8) is Markovian, since $\mathbf{x}_{t-1}$ solely depends on $\mathbf{x}_0, \tau, \mathbf{x}_t$. Plugging (8) into (4) gives the generation process. We can prove the ELBO of the DNDM is equivalent to the ELBO of the original process (5) up to some constant, which further supports the neural network $p_{\boldsymbol{\theta}}(\cdot|\mathbf{x}_t)$ trained by the Markov process in (1), remains applicable to DNDM. (See Appendix B.3 for details).

Remark 3.4. (7) and (8) suggest that even though there are T distinct time steps, not every time in the range $1 : T$ is crucial for capturing the process. Therefore, our primary focus should be on the most significant time step, i.e., the transition time τ, enabling faster reverse sampling. We further note that although transition happens only at time τ, the transition time is random, differs across runs, and covers the full range from 1 to T on average.

Remark 3.5. While Song et al. (2020a) proposed a non-Markov multinomial diffusion model in Appendix A, DDIM and DNDM are fundamentally different models when specialized to multinomial diffusion. DDIM's discrete process remains stochastic at every step, even with deterministic noise scheduling. In contrast, DNDM achieves full de-randomization by pre-determined transition time τ (Equation 8 in our paper). By sampling these transition times upfront, DNDM establishes a predetermined transition time set that guides the sampling process, enabling deterministic evolution and faster sampling speed even under the same number of sampling steps, which is not reported under DDIM framework. For detailed technical comparison, see Appendix B.1.

3.2 Accelerated Reverse Sampling

In this section, we demonstrate that sampling from DNDM can lead to accelerated reverse sampling. Although our algorithm is quite general, we focus on text generation in the presentation.

In Section 3.1, we only consider the case of a single token $\mathbf{x} \in \mathbb{R}^K$ being one hot encoding of K categories. In real applications, we are interested in generating a sentence with multiple tokens. So, we extend the terminology in Section 3.1, and we denote the sequence of tokens at t-th time step to be $\mathbf{x}_{t,1:N} = [\mathbf{x}_{t,1}, \dots, \mathbf{x}_{t,N}]$ where $\mathbf{x}_{t,n}$ is the n-th token and N is the sequence length. The noise will be added to each token in a sequence independently. Therefore, each token will have its own transition time defined in Definition 3.2. We denote the transition time for each token $\mathbf{x}_n$ to be τ_n and further denote the transition time set $\mathcal{T} := \{\tau_n\}_{n=1}^N$. Given the transition times $\tau_n \in \mathcal{T}$, our DNDM can now be extended to the sequence with multiple tokens

$$\mathbf{x}_{t-1,n} = \mathbb{1}(\tau_n = t)\mathbf{x}_{0,n} + \mathbb{1}(\tau_n \neq t)\mathbf{x}_{t,n}, \forall n \in [N]. \tag{9}$$

Learning the Reverse Process. We first generate the transition times τ_n for $n \in [N]$, then we follow (9) to generate the learned reverse process. Since $\mathbf{x}_{0,n}$ is unknown in the process, we use the neural network evaluation $p_{\boldsymbol{\theta}}(\cdot|\mathbf{x}_t)$ obtained in Section 3.1 to predict $\mathbf{x}_{0,n}$. In detail, the noisy sequence $\mathbf{x}_{t,1:N}$ is fed into $p_{\boldsymbol{\theta}}(\cdot|\mathbf{x}_{t,1:N})$ and the prediction tokens $\widehat{\mathbf{x}}_{0,1:N} \sim p_{\boldsymbol{\theta}}(\cdot|\mathbf{x}_{t,1:N})$ are collected.

Transition time. Transition time, denoted by τ, is crucial in our reverse process. This is because the reverse sampling becomes deterministic upon using (9). Each instance of transition time τ is a random variable within the set $\{1, 2, \dots, T\}$. Let's assume it follows the distribution $\mathcal{D}_\tau$. Given the schedule $\{\alpha_t\}_{t=0}^T$, we can derive the distribution for $\mathcal{D}_\tau$.

Theorem 3.6. *Each specific transition time τ_n in Definition 3.2 is independent. Furthermore, they collectively adhere to the distribution $\mathcal{D}_\tau$, which obeys the rule $\mathbb{P}(\tau_n = t) = \alpha_{t-1} - \alpha_t$.*

From Theorem 3.6, we discern that the nature of the diffusion model scheduler, α_t, clarifies the distribution of τ. Take the linear schedule as an example, as given by Austin et al. (2021), the relationship is $\alpha_t = 1 - t/T$. This translates to $\mathbb{P}(\tau_n = t) = 1/T$ for every t in the range 1 to T. As a result, transition time distributes uniformly across each moment in the set $\{1, \dots, T\}$. Generally, if we express α_t as $g(t/T)$, then we can simplify to $\mathbb{P}(\tau_n = t) = g((t-1)/T) - g(t/T)$, which further refines to $(1/T)|g'(t/T)| + o(1/T)$. This indicates that transitions are more likely where $|g'|$ is large.

In practice, we observed that the shape of the transition time does not need to exactly match the theoretically predicted schedule $\mathcal{D}\tau$ in Theorem 3.6. Algorithm 1 works even if $\mathcal{D}\tau$ is unknown. In particular, we can approximate the schedule with a Beta distribution by first sampling a time $t \in [0, 1]$ from a Beta distribution, then adjusting these samples to fit by multiplying by T and rounding the result to obtain an integer.

Algorithm 1 Sampling From DNDM

Require: Trained prediction function $p_{\boldsymbol{\theta}}$, $\mathbf{q}_{\text{noise}}$, $\mathcal{D}_\tau$
for $n = 1 \dots N$ **do**
  Initiate each token $\mathbf{x}_{T,n} \sim \mathbf{q}_{\text{noise}}$
  Initiate the transition time $\tau_n \sim \mathcal{D}_\tau$
end for
Collect transition time set $\mathcal{T} = \{\tau_n\}_{n=1}^N$
for $t = T \dots 1$ **do**
  if $t \in \mathcal{T}$ **then**
    Generate $\widetilde{\mathbf{x}}_{0,1:N}$ from $p_{\boldsymbol{\theta}}(\cdot|\mathbf{x}_{t,1:N})$
    for $n = 1 \dots N$ **do**
      Update $\mathbf{x}_{t-1,n}$ based on condition of τ_n
    end for
  else
    Update $\mathbf{x}_{t-1,1:N} = \mathbf{x}_{t,1:N}$
  end if
end for
Return $\mathbf{x}_{0,1:N}$

Accelerated Sampling. According to (9), a token $\mathbf{x}_{t-1,n}$ is updated only if step t is the transition time for the n-th token. If step t is not the transition time for any token, the sentence from the previous step can be directly copied: $\mathbf{x}_{t-1,1:N} = \mathbf{x}_{t,1:N}$. As a result, there is no need to do a function evaluation for the current step. Our attention, therefore, can be solely centered on the transition set $\mathcal{T}$, necessitating function evaluations only for t within $\mathcal{T}$. For our method, when N is fixed while $T \to \infty$, the total NFE $|\mathcal{T}|$ will reach N. On the other hand, when T is fixed and $N \to \infty$, the NFE $\mathcal{T}$ will reach T (See Theorem D.1 for detail). It is worth noting that the auto-regressive diffusion model (ARDM) (Hoogeboom et al., 2021a) can also achieve at most N NFE when $T = \infty$. However, ARDM only focuses on infinite time steps, while our method here is

able to accelerate sampling for finite time steps. More detailed discussion and theoretical analysis can be found in Section D, where additional experiments also demonstrate that our DNDM achieves an NFE that is less than half of the original Markov sampling method for discrete diffusion.

By incorporating the forward process with different noises, we can develop DNDM-Multi and DNDM-Absorb, which accelerate the Multinomial and Absorbing sampling methods respectively. Recent works have demonstrated that the quality of samples can be enhanced by utilizing supplementary information derived from the neural network, (Ghazvininejad et al., 2019; Savinov et al., 2021; Chang et al., 2022; He et al., 2022; Zheng et al., 2023). Our DNDM can also be improved using this idea. We call it a discrete non-Markov Diffusion Model with Top-k Transition Time (DNDM-k). Due to the limit of the pages, we leave the detailed Algorithm and discussion to Appendix E.

3.3 Continous-time (Infinite Step) Reverse Sampling

In the context of continuous state spaces, continuous-time processes have been proposed to accommodate algorithms that offer faster sampling speeds and enhanced sample quality (Jolicoeur-Martineau et al., 2021; Zhang and Chen, 2022; Salimans and Ho, 2022; Chung et al., 2022; Song et al., 2020b; Dockhorn et al., 2021). However, the application of continuous-time schemes to discrete-state spaces remains largely unexplored. Campbell et al. (2022) first developed a continuous framework for discrete-time diffusion for the Markovian process and randomized sampling, but not in our non-Markovian setting. In this section, we investigate the transition from finite to infinite step sampling, providing new insights into bridging the gap between discrete and continuous-time processes for discrete diffusion models.

Continuous-time Forward and Backward process. Recall that the forward process described in (6) can be sampled from $\mathbf{x}_{0,n}$ through the following process:

$$\mathbf{x}_{t,n} = \alpha_t \mathbf{x}_{0,n} + (1-\alpha_t)\mathbf{q}_{\text{noise}}, \quad \alpha_t = \prod_{i=1}^{t} \beta_i. \tag{10}$$

In the previous section, we are constrained to discrete time steps, where we must define a maximum step, denoted by T. The values of $\mathbf{x}_t$ are computed only for $t = 1, \ldots, T$. As a result, during the training process, it is only possible to predict $\mathbf{x}_0$ at these predetermined time steps. This constraint confines the computation of our reverse process exclusively to these fixed time stamps. To derive the continuous limit of (10), for each T we rescale (10) to a diffusion process on $[0,1]$, e.g., $\mathbf{x}_{T,n} = \widehat{\mathbf{x}}_{1,n}, \mathbf{x}_{0,n} = \widehat{\mathbf{x}}_{0,n}$, and $\mathbf{x}_{t,n} = \widehat{\mathbf{x}}_{t/T,n}$. Therefore, when $T \to \infty$, $\widehat{\mathbf{x}}_{t,n}$ represents the continuous process that has values at arbitrary $t \in [0,1]$. If the choice of α_t for each T is scale-invariant, we can define a continuous function $\alpha(t)$ as the continuous α schedule of the discrete counterpart[1]. More specifically, we obtain

Algorithm 2 Sampling from DNDM-C

Require: Trained prediction function $p_{\boldsymbol{\theta}}$, $\mathbf{q}_{\text{noise}}$, $\mathcal{D}_\tau$
1: **for** $n = 1 \ldots N$ **do**
2: Initiate each token $\mathbf{x}_{T,n} \sim \mathbf{q}_{\text{noise}}$
3: Initiate the transition time $\tau_n \sim \mathcal{D}_\tau$ and order them as $\tau_{n_1} < \ldots < \tau_{n_N}$
4: **end for**
5: **for** $k = N \ldots 1$ **do**
6: Generate $\widetilde{\mathbf{x}}_{0,1:N}$ from $p_{\boldsymbol{\theta}}(\cdot|\mathbf{x}_{\tau_{n_k},1:N}, \tau_{n_k})$
7: **for** $n = 1 \ldots N$ **do**
8: Update $\mathbf{x}_{\tau_{n_{k-1}},n}$ based on condition of τ_n
9: **end for**
10: **end for**
11: **Return** $\mathbf{x}_{0,1:N}$

$$\widehat{\mathbf{x}}_{t,n} = \alpha(t)\widehat{\mathbf{x}}_{0,n} + (1-\alpha(t))\mathbf{q}_{\text{noise}}, \quad t \in [0,1]. \tag{11}$$

For the reverse-time process, we define the transition time set $\mathcal{T} := \{\tau_n\}_{n=1}^N$ consistent with Theorem 3.6 and sample it from $\mathbb{P}(\tau_n = t) = -\alpha'(t)$ (we always use decreasing $\alpha(t)$). With $\mathcal{T}$ defined, the updates to $\mathbf{x}_{t,n}$ only occur at $\{\tau_n\}$. Consequently, we arrange τ_n to obtain an ordered sequence τ_{n_k}, where $\tau_{n_1} < \tau_{n_2} < \ldots < \tau_{n_N}$. When omitting the infinitely many time steps between τ_{n_k} and $\tau_{n_{k-1}}$, the resulting reverse process is then given by:

$$\mathbf{x}_{\tau_{n_{k-1}},n} = \mathbb{1}(\tau_n = \tau_{n_{k-1}})\mathbf{x}_{0,n} + \mathbb{1}(\tau_n \neq \tau_{n_{k-1}})\mathbf{x}_{\tau_{n_k},n}, . \tag{12}$$

for all $n \in [N]$. The detailed algorithm named DNDM-C is shown in Algorithm 2.

[1] If we represent α_t with maximum step T as $\alpha_t(T)$, the scale-invariant property states that $\alpha_{ct}(cT) = \alpha_t(T)$. The simplest example of such an α_t schedule is $\alpha_t(T) = 1 - t/T$, under which $\alpha(t) = 1 - t$.

Remark 3.7. Autoregressive Diffusion Model (ARDM) (Hoogeboom et al., 2021a) is a discrete diffusion model built upon the autoregressive nature of data. ARDM is shown to be equivalent to a continuous-time absorbing diffusion model and thus provides a unique perspective for discrete diffusion. For continuous-time ($T = \infty$) reverse sampling, both ARDM and our method achieve N NFEs. Unlike ARDM which is limited to absorbing-state transitions, our method provides a unified framework including both absorbing and multinomial diffusions, applicable to both finite time and continuous time diffusions. For infinite timesteps, Hoogeboom et al. (2021a) also proposed an advanced parallelizing technique that can reduce NFE according to the log-likelihood, which we have not considered in DNDM-C.

4 Experiments

In this section, we evaluate DNDM and demonstrate its superior performance on two types of tasks: conditional sequence-to-sequence text generation (i.e., machine translation) and unconditional text generation. For the fairness of comparison, all the experiments are conducted using a single NVIDIA RTX A6000 GPU with 48 GB memory. Additional experiment details are provided in Appendix F.

4.1 Conditional Text Generation

We evaluate DNDM's effectiveness on conditional text generation through machine translation tasks. Following Zheng et al. (2023), we use Byte Pair Encoding (BPE) (Sennrich et al., 2016) to create a shared vocabulary of words and subwords from both source and target languages. We implement our experiments using FairSeq (Ott et al., 2019), which employs an encoder-decoder architecture. The model uses bi-directional self-attention blocks without causal masking, allowing tokens to attend to both past and future positions during training and inference. The encoder processes the source text, while the decoder generates the target translation.

Datasets. We use the following three datasets to compare with the baselines for machine translation tasks: (1) `IWSLT14 DE-EN` (Cettolo et al., 2014), a dataset with German as the source language and English as the target language. It consists of 174272 examples (sentence pairs), and each of the validation set and the testing set accounts for 7283 and 6750 of the dataset; (2) `WMT14 EN-DE` (Bojar et al., 2014), which is an English-to-German translation dataset consisting of 3967182 examples. Each of the validation set and the testing set accounts for 3000 and 3003 of the dataset; and (3) `WMT16 EN-RO` (Bojar et al., 2016), which is an English-to-Russian translation dataset consisting of 612317 examples. Each of the validation sets and the testing set accounts for 1999 and 1999 of the dataset. The train-validation-test split is fixed across all experiments for all machine translation datasets to ensure fair comparison.

Performance Metrics. We use the BLEU score (Papineni et al., 2002) to evaluate the machine translation quality, where the BLEU score is calculated based on the similarity between the actual target sequence and the predicted target sequence. The sampling speed is measured by wall-clock time (in second).

Baselines. The main baselines we are comparing with are RDM and RDM-k from Zheng et al. (2023). Here, we use RDM-k and RDM to denote the sampling method proposed in their paper with and without the usage of top-k selection for the token generation technique (see Appendix E for more details), respectively. RDM and RDM-k are applied to two previously proposed state-of-the-art discrete diffusion models: Multinomial Diffusion (Hoogeboom et al., 2021b) and Absorbing Diffusion (Austin et al., 2021).

Results and Discussion. Tables 2 and 3 present the performance evaluations of our algorithms in machine translation tasks. Table 2 presents results for multinomial diffusion, while Table 3 displays results for absorbing diffusion. Our reported time and BLEU scores are averaged over 5 repeated experiments, except for the baseline RDM experiment[2].

From Tables 2 and 3, we observe that methods based on DNDM significantly accelerate the sampling process compared to baseline diffusion models. This acceleration allows for greater flexibility in increasing the number of steps (up to infinity) without imposing a significant computational burden.

[2]Due to computational intensity, we did not repeat the 1000-step sampling for the RDM baseline. However, reproducing it was deemed unnecessary as the sampling time is largely stable across repeated experiments, and the precise averaged timing is not critical for demonstrating the speed improvement of DNDM.

In particular, more sampling steps lead to better generation quality (BLEU) at the expense of longer sampling time, as indicated in each column of Tables 2 and 3. For RDM-based methods, generation time increases linearly with the number of sampling steps. On the contrary, for our DNDM-based method, generation time only increases marginally (See Figure 4 in Section G). As a result of the difference in the growing speed of sampling time with respect to sampling steps, the more sampling steps, the more speedup DNDM can obtain.

Continuous-time results, as the ultimate limit of increasing sampling steps, are presented in the last row of each dataset with the tag ∞. Given that the results with 1000 steps consistently outperform those with 50 steps, we compare ∞ with 1000 steps in Table 2 and 3. For `IWSLT14` and `WMT16`, where the generation BLEU score is relatively high, we observe a consistent performance improvement of up to 0.3 in BLEU score when utilizing the DNDM-C algorithm, with the exception of a single case in the absorbing diffusion setting for `WMT16` without the use of top-k selection. The performance gain of the continuous-time method on `WMT14` is less significant, with both drops and gains. However, `WMT14` itself has not reached a high level of performance, with a BLEU score significantly lower than other datasets. In general, training `WMT14` poses challenges across all diffusion models, including multinomial diffusion (Hoogeboom et al., 2021b), absorbing diffusion (Austin et al., 2021), and RDM diffusion (Zheng et al., 2023), etc. We defer a more detailed discussion on WMT14 to Appendix F.1. Finally, when compared with the results obtained with 50 steps, the performance of DNDM-C demonstrates improvement consistently. Furthermore, we note that regardless of the dataset or the method (i.e., RDM or DNDM) employed, top-k token generation consistently outperforms vanilla methods. This approach enhances the BLEU score by approximately 1-2 points without introducing significant increases in sampling time.

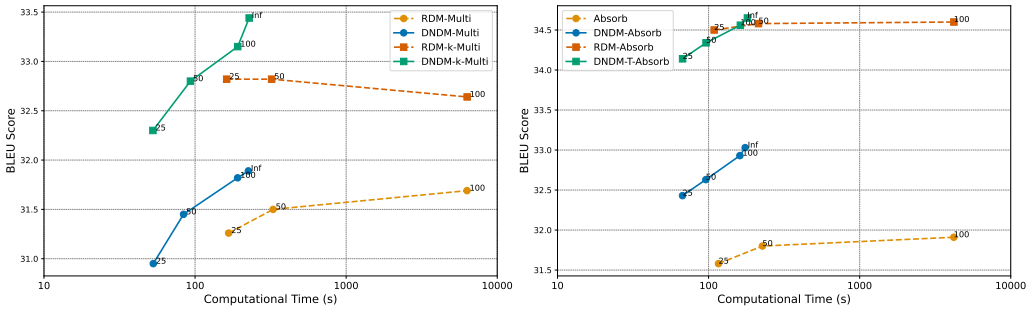
Table 2: BLEU score comparison of multinomial diffusion on machine translation benchmarks `IWSLT14 DE-EN`, `WMT14 EN-DE`, and `WMT16 EN-RO`. Below the dataset, we present the amount of data used to run the evaluation (sentences). The blue background highlights our algorithms, and the bold number indicates the best performance within each row and each setting (i.e., with or without top-k).

Dataset	Steps	RDM-Multi		DNDM-Multi		RDM-k-Multi		DNDM-k-Multi	
		BLEU	**Time (s)**	**BLEU**	**Time (s)**	**BLEU**	**Time(s)**	**BLEU**	**Time (s)**
`IWSLT14` (6.75k)	25	**31.26**	166.9	30.95	**52.9**	**32.82**	161.9	32.30	**52.6**
	50	**31.50**	328.6	31.45	**83.9**	**32.82**	321.2	32.80	**93.2**
	1000	31.69	6308.9	**31.82**	**191.3**	32.64	6321.3	**33.15**	**191.5**
	∞	-	-	**31.89**	**225.2**	-	-	**33.44**	**228.1**
`WMT14` (3k)	25	**25.25**	237.3	25.01	**90.7**	**26.03**	230.9	25.98	**90.5**
	50	**25.75**	466.1	25.33	**138.4**	26.14	500.2	**26.37**	**138.3**
	1000	25.66	8996.7	**25.71**	**265.4**	25.82	8991.7	**26.88**	**265.5**
	∞	-	-	**24.79**	**307.5**	-	-	**26.39**	**307.3**
`WMT16` (2k)	25	**32.29**	145.2	31.97	**36.4**	**33.12**	143.5	32.94	**36.4**
	50	**32.53**	286.1	32.50	**63.2**	**33.41**	312.4	33.26	**62.7**
	1000	32.63	5588.9	**32.86**	**171.4**	33.67	5601.0	**33.79**	**171.2**
	∞	-	-	**32.91**	**196.4**	-	-	**33.86**	**196.3**

Scaling Law in Sampling Speed. For illustrative purposes, we use the example of `IWSLT14` to visualize how the sample quality scales regarding sampling speed for different methods. In Figure 1, we observe the trend of the BLEU score in relation to computational time. Each line in the legend represents a different sampling algorithm, and a steeper slope indicates a larger marginal gain when sampling for longer periods. Figure 1 demonstrates that our algorithm displays nearly linear growth in BLEU score over the log of time, which is remarkable in contrast with the flat curve of the baseline. Particularly, for multinomial diffusion, the BLEU score increases by 1 in less than 60 seconds of additional sampling time. For absorbing diffusion, DNDM outperforms RDM before RDM samples 50 steps. In Tables 7 and 8 in Appendix D, we further use the average number of function evaluations (NFE) to measure the improved speed within the specified number of sampling steps. Additionally, in Figure 2, we visualize how the BLEU score and the generated text change throughout the sampling process.

Table 3: BLEU score comparison of absorbing diffusion on machine translation benchmarks `IWSLT14 DE-EN`, `WMT14 EN-DE`, and `WMT16 EN-RO`. Below the dataset, we present the amount of data used to run the evaluation (sentences). The blue background highlights our algorithms, and the bold number indicates the best performance within each row and each setting (i.e., with or without top-k).

Dataset	Steps	RDM-Absorb		DNDM-Absorb		RDM-k-Absorb		DNDM-k-Absorb	
		BLEU	Time (s)	BLEU	Time (s)	BLEU	Time(s)	BLEU	Time (s)
IWSLT14 (6.75k)	25	31.58	116.3	**32.43**	**67.2**	**34.50**	108.9	34.14	**67.3**
	50	31.80	227.2	**32.63**	**95.9**	**34.58**	213.9	34.34	**96.2**
	1000	31.91	4197.4	**32.93**	**161.1**	**34.60**	4205.9	34.56	**162.3**
	∞	-	-	**33.03**	**174.6**	-	-	**34.65**	**180.7**
WMT14 (3k)	25	24.97	116.4	**25.79**	**68.1**	**27.50**	107.5	27.18	**68.0**
	50	24.95	231.1	**26.10**	**102.0**	**27.73**	255.2	27.66	**102.5**
	1000	25.22	4169.4	**26.43**	**178.3**	27.75	4167.4	**27.82**	**179.1**
	∞	-	-	**26.50**	**180.1**	-	-	**27.50**	**181.2**
WMT16 (2k)	25	32.86	75.5	**33.20**	**41.2**	33.92	69.9	**33.96**	**41.4**
	50	32.93	148.4	**33.30**	**62.5**	34.10	166.1	**34.20**	**62.7**
	1000	33.25	2951.7	**33.60**	**121.3**	**34.44**	2718.7	34.38	**122.7**
	∞	-	-	**33.42**	**121.8**	-	-	**34.41**	**121.9**



(a) Multinomial Diffusion (b) Absorbing Diffusion

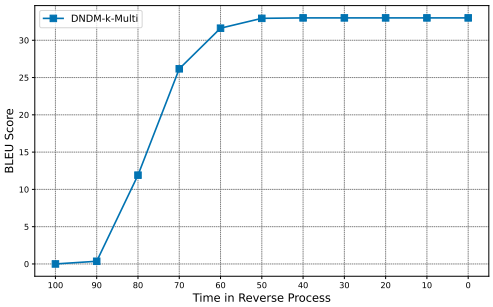
Figure 1: Generation quality to generation time comparison on `IWSLT14`. x-axis: computational time in seconds; y-axis: BLEU score.

4.2 Unconditional Text Generation

For unconditional text generation, we evaluate our approach on language modeling tasks, where the model learns to generate text that matches the statistical patterns of the training data. Unlike conditional generation, this task involves directly learning $q(\mathbf{x}_0|\mathbf{x}_t)$ without conditioning on any input text. We conduct experiments on the text8 and enwik8 datasets using a decoder-only architecture similar to GPT models. Since unconditional generation does not require encoding input sequences, we employ a 12-layer Transformer decoder without an encoder component.

Datasets. The natural language generation task is evaluated on two language datasets following Hoogeboom et al. (2021b): `text8` and `enwik8`. Both datasets are from Wikipedia, but their contents are highly distinct. In `text8`, the plain text consists of English words (all the letters are in lower case) and spaces, and it is tokenized into 26 characters and one blank space, resulting in 27 categories. In contrast to the cleanness of `text8`, `enwik8` preserves the original XML dump contents, and there exist various special symbols in its raw text, so its text is tokenized into 1 Byte, resulting in 256 categories. We utilize `text8` dataset with sequence length 256 and `enwik8` dataset with sequence length 320. The train/val/test splits are 9e7/5e6/5e5 for both `text8` and `enwik8`.

Performance Metrics. Our evaluation of text generation quality relies on the perplexity score. When generating `text8` data, we calculate perplexity scores using the GPT2 model, while for enwik8 data generation, we employ the GPT2-large model. The sampling speed is measured in seconds.



(a) The BLEU Score in the Generation Process

t = 100 **[noise] [noise] [noise] [noise]** $\cdots$

t = 75 **[noise]** $\cdots$ **[noise]** `and we` **[noise]** $\cdots$ **[noise]** `govern` **[noise] [noise]** `year` **[noise]**

t = 67 `we` **[noise] [noise]** `fello` **[noise] [noise] [noise]** and we let them **[noise] [noise]** `city govern` **[noise]** `every year.`

t = 39 `we choose some fellows every year and we let them work with city governance every year.`

t = 0 `we choose some fellows every year and we let them work with city governance every year.`

(b) Text in the Generation Process

Figure 2: We demonstrate the 100-step generation process of DNDM-k-Multi as an example, where the left is the change of the BLEU score along the generation process, and the right is the text at different time steps. As the time goes from 100 to 0, noise is gradually removed until the corresponding English text emerges. Since the transition time follows a Beta distribution as described in Section 3.2, the majority of transitions occur near the starting time.

Baselines. We compare our proposed DNDM on unconditional text generation task with the vanilla Multinomial Diffusion (Hoogeboom et al., 2021b).

Results and Discussion. Table 4 displays the performance of our algorithms in text generation tasks. We run the multinomial diffusion model on the `text8` dataset for 1000 diffusion steps and on the `enwik8` dataset for 4000 diffusion steps. Our DNDM-based algorithms outperform the vanilla sampling algorithm used in Hoogeboom et al. (2021b) in terms of both sampling time and perplexity score. Specifically, for the `text8` dataset, DNDM-based algorithms are 5 times faster than the vanilla algorithm. For the `enwik8` dataset, DNDM-based algorithms are 14 times faster than the vanilla algorithm.

Table 4: Comparison of different sampling methods for unconditional text generation (multinomial diffusion) on `text8` and `enwik8` benchmarks. Sampling time is computed by generating a single text sample of length 256 for `text8` and length 320 for `enwik8`, averaged over 10 runs. The blue background represents our algorithms, and the bold number indicates the optimal value.

		Vanilla	DNDM
`text8`	Perplexity	1,465.75	**600.02**
	Time (s)	135.9	**31.1**
`enwik8`	Perplexity	801.78	**556.78**
	Time (s)	602.8	**47.4**

5 Conclusion and Future Work

This paper presents a novel discrete non-Markov diffusion model (DNDM) accompanied by an accelerated sampling algorithm designed to boost sampling speed in a discrete-state space. Our discrete diffusion model incorporates "transition time set" latent variables, establishing itself as an efficacious diffusion and data generation method. Thanks to our acceleration technique, we significantly decrease the number of neural network function evaluations without sacrificing sample quality. We also introduce an infinite-step sampling algorithm, DNDM-C, which provides new insights into bridging the gap between discrete and continuous-time processes for discrete diffusion models. While this study focuses on text generation using non-autoregressive models, a promising direction for future exploration is applying our method to other tasks, such as audio and image generation.

Acknowledgement

We thank the anonymous reviewers and area chair for their helpful comments. ZC, HY, YL, YK, JZ, and QG are supported in part by the National Science Foundation CAREER Award 1906169, IIS-2008981, and the Sloan Research Fellowship. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

References

ALAIN, G., BENGIO, Y., YAO, L., YOSINSKI, J., THIBODEAU-LAUFER, E., ZHANG, S. and VINCENT, P. (2016). Gsns: generative stochastic networks. *Information and Inference: A Journal of the IMA* **5** 210–249.

ALIAS PARTH GOYAL, A. G., KE, N. R., GANGULI, S. and BENGIO, Y. (2017). Variational walkback: Learning a transition operator as a stochastic recurrent net. *Advances in Neural Information Processing Systems* **30**.

AUSTIN, J., JOHNSON, D. D., HO, J., TARLOW, D. and VAN DEN BERG, R. (2021). Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems* **34** 17981–17993.

BAO, F., LI, C., ZHU, J. and ZHANG, B. (2022). Analytic-dpm: an analytic estimate of the optimal reverse variance in diffusion probabilistic models. *arXiv preprint arXiv:2201.06503* .

BENGIO, Y., LAUFER, E., ALAIN, G. and YOSINSKI, J. (2014). Deep generative stochastic networks trainable by backprop. In *International Conference on Machine Learning*. PMLR.

BOJAR, O., BUCK, C., FEDERMANN, C., HADDOW, B., KOEHN, P., LEVELING, J., MONZ, C., PECINA, P., POST, M., SAINT-AMAND, H., SORICUT, R., SPECIA, L. and TAMCHYNA, A. (2014). Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Baltimore, Maryland, USA.

BOJAR, O., CHATTERJEE, R., FEDERMANN, C., GRAHAM, Y., HADDOW, B., HUCK, M., JIMENO YEPES, A., KOEHN, P., LOGACHEVA, V., MONZ, C., NEGRI, M., NÉVÉOL, A., NEVES, M., POPEL, M., POST, M., RUBINO, R., SCARTON, C., SPECIA, L., TURCHI, M., VERSPOOR, K. and ZAMPIERI, M. (2016). Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*. Association for Computational Linguistics, Berlin, Germany.

BORDES, F., HONARI, S. and VINCENT, P. (2017). Learning to generate samples from noise through infusion training. *arXiv preprint arXiv:1703.06975* .

CAMPBELL, A., BENTON, J., DE BORTOLI, V., RAINFORTH, T., DELIGIANNIDIS, G. and DOUCET, A. (2022). A continuous time framework for discrete denoising models. *Advances in Neural Information Processing Systems* **35** 28266–28279.

CERITLI, T., GHOSHEH, G. O., CHAUHAN, V. K., ZHU, T., CREAGH, A. P. and CLIFTON, D. A. (2023). Synthesizing mixed-type electronic health records using diffusion models. *arXiv preprint arXiv:2302.14679* .

CETTOLO, M., NIEHUES, J., STÜKER, S., BENTIVOGLI, L. and FEDERICO, M. (2014). Report on the 11th IWSLT evaluation campaign. In *Proceedings of the 11th International Workshop on Spoken Language Translation: Evaluation Campaign*. Lake Tahoe, California.

CHANG, H., ZHANG, H., JIANG, L., LIU, C. and FREEMAN, W. T. (2022). Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

CHEN, N., ZHANG, Y., ZEN, H., WEISS, R. J., NOROUZI, M. and CHAN, W. (2020). Wavegrad: Estimating gradients for waveform generation. *arXiv preprint arXiv:2009.00713* .

CHUNG, H., SIM, B. and YE, J. C. (2022). Come-closer-diffuse-faster: Accelerating conditional diffusion models for inverse problems through stochastic contraction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

DOCKHORN, T., VAHDAT, A. and KREIS, K. (2021). Score-based generative modeling with critically-damped langevin diffusion. *arXiv preprint arXiv:2112.07068* .

DOCKHORN, T., VAHDAT, A. and KREIS, K. (2022). Genie: Higher-order denoising diffusion solvers. *Advances in Neural Information Processing Systems* **35** 30150–30166.

GHAZVININEJAD, M., LEVY, O., LIU, Y. and ZETTLEMOYER, L. (2019). Mask-predict: Parallel decoding of conditional masked language models. *arXiv preprint arXiv:1904.09324* .

GRUVER, N., STANTON, S., FREY, N., RUDNER, T. G., HOTZEL, I., LAFRANCE-VANASSE, J., RAJPAL, A., CHO, K. and WILSON, A. G. (2024). Protein design with guided discrete diffusion. *Advances in Neural Information Processing Systems* **36**.

HE, Z., SUN, T., WANG, K., HUANG, X. and QIU, X. (2022). Diffusionbert: Improving generative masked language models with diffusion models. *arXiv preprint arXiv:2211.15029* .

HO, J., CHAN, W., SAHARIA, C., WHANG, J., GAO, R., GRITSENKO, A., KINGMA, D. P., POOLE, B., NOROUZI, M., FLEET, D. J. ET AL. (2022). Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303* .

HO, J., JAIN, A. and ABBEEL, P. (2020). Denoising diffusion probabilistic models. *Advances in neural information processing systems* **33** 6840–6851.

HOOGEBOOM, E., GRITSENKO, A. A., BASTINGS, J., POOLE, B., BERG, R. V. D. and SALIMANS, T. (2021a). Autoregressive diffusion models. *arXiv preprint arXiv:2110.02037* .

HOOGEBOOM, E., NIELSEN, D., JAINI, P., FORRÉ, P. and WELLING, M. (2021b). Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in Neural Information Processing Systems* **34** 12454–12465.

JOLICOEUR-MARTINEAU, A., LI, K., PICHÉ-TAILLEFER, R., KACHMAN, T. and MITLIAGKAS, I. (2021). Gotta go fast when generating data with score-based models. *arXiv preprint arXiv:2105.14080* .

KARRAS, T., AITTALA, M., AILA, T. and LAINE, S. (2022). Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems* **35** 26565–26577.

KONG, Z. and PING, W. (2021). On fast sampling of diffusion probabilistic models. *arXiv preprint arXiv:2106.00132* .

KONG, Z., PING, W., HUANG, J., ZHAO, K. and CATANZARO, B. (2020). Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761* .

LIU, L., REN, Y., LIN, Z. and ZHAO, Z. (2022). Pseudo numerical methods for diffusion models on manifolds. *arXiv preprint arXiv:2202.09778* .

LU, C., ZHOU, Y., BAO, F., CHEN, J., LI, C. and ZHU, J. (2022). Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems* **35** 5775–5787.

LYU, S. (2012). Interpretation and generalization of score matching. *arXiv preprint arXiv:1205.2629* .

MOVELLAN, J. R. (2008). Contrastive divergence in gaussian diffusions. *Neural Computation* **20** 2238–2252.

NACHMANI, E., ROMAN, R. S. and WOLF, L. (2021). Non gaussian denoising diffusion models. *arXiv preprint arXiv:2106.07582* .

NICHOL, A. Q. and DHARIWAL, P. (2021). Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*. PMLR.

OTT, M., EDUNOV, S., BAEVSKI, A., FAN, A., GROSS, S., NG, N., GRANGIER, D. and AULI, M. (2019). fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038* .

PAPINENI, K., ROUKOS, S., WARD, T. and ZHU, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*.

REID, M., HELLENDOORN, V. J. and NEUBIG, G. (2022). Diffuser: Discrete diffusion via edit-based reconstruction. *arXiv preprint arXiv:2210.16886* .

SALIMANS, T. and HO, J. (2022). Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512* .

SAN-ROMAN, R., NACHMANI, E. and WOLF, L. (2021). Noise estimation for generative diffusion models. *arXiv preprint arXiv:2104.02600* .

SAVINOV, N., CHUNG, J., BINKOWSKI, M., ELSEN, E. and OORD, A. V. D. (2021). Step-unrolled denoising autoencoders for text generation. *arXiv preprint arXiv:2112.06749* .

SENNRICH, R., HADDOW, B. and BIRCH, A. (2016). Neural machine translation of rare words with subword units.

SOHL-DICKSTEIN, J., BATTAGLINO, P. and DEWEESE, M. R. (2009). Minimum probability flow learning. *arXiv preprint arXiv:0906.4779* .

SOHL-DICKSTEIN, J., WEISS, E., MAHESWARANATHAN, N. and GANGULI, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*. PMLR.

SONG, J., MENG, C. and ERMON, S. (2020a). Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502* .

SONG, Y., DHARIWAL, P., CHEN, M. and SUTSKEVER, I. (2023). Consistency models. *arXiv preprint arXiv:2303.01469* .

SONG, Y. and ERMON, S. (2019). Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems* **32**.

SONG, Y. and ERMON, S. (2020). Improved techniques for training score-based generative models. *Advances in neural information processing systems* **33** 12438–12448.

SONG, Y., SOHL-DICKSTEIN, J., KINGMA, D. P., KUMAR, A., ERMON, S. and POOLE, B. (2020b). Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456* .

SUN, H., YU, L., DAI, B., SCHUURMANS, D. and DAI, H. (2022). Score-based continuous-time discrete diffusion models. *arXiv preprint arXiv:2211.16750* .

VAHDAT, A., KREIS, K. and KAUTZ, J. (2021). Score-based generative modeling in latent space. *Advances in Neural Information Processing Systems* **34** 11287–11302.

WATSON, D., HO, J., NOROUZI, M. and CHAN, W. (2021). Learning to efficiently sample from diffusion probabilistic models. *arXiv preprint arXiv:2106.03802* .

YE, J., ZHENG, Z., BAO, Y., QIAN, L. and GU, Q. (2023). Diffusion language models can perform many tasks with scaling and instruction-finetuning. *arXiv preprint arXiv:2308.12219* .

ZHANG, Q. and CHEN, Y. (2022). Fast sampling of diffusion models with exponential integrator. *arXiv preprint arXiv:2204.13902* .

ZHENG, L., YUAN, J., YU, L. and KONG, L. (2023). A reparameterized discrete diffusion model for text generation. *arXiv preprint arXiv:2302.05737* .

Broader Impact

This paper presents work that aims to advance the field of diffusion models. We believe this work may enable future applications of synthetic data generation, which may lead to positive impacts. Our experiments demonstrate that the proposed method achieves state-of-the-art performance in the acceleration of the generative model. However, proper controls may be needed whenever applying our method to tasks that involve sensitive data data. There may be other potential societal consequences of our work, none of which we feel must be specifically highlighted here.

Limitations

- The scope of the empirical claims is limited to the text domain with non-auto regressive setting. The applicability and performance of DNDM for other tasks like audio and image generation, as well as with other architectures like auto-regressive GPT models, are not explored and left as future work.
- While DNDM-C, the infinite-step sampling algorithm, offers new insights into bridging the gap between discrete and continuous-time processes for discrete diffusion models, the sample quality is not guaranteed to be superior to the accelerated algorithm with 1000 steps. Some intuitions here: the assumption that the neural network can be optimally trained is an ideal case and is often not realized in practice. There is an inherent estimation error associated with the training process. As the number of steps increases, these estimation errors can accumulate, potentially leading to a degradation in performance. This cumulative estimation error might explain why using an infinite number of steps does not necessarily yield better results than a finite number of steps like 1000 in the conditional generation experiments. How to further improve sample quality of infinite steps is interesting but beyond the scope of this paper.
- This paper focuses on the comparison with discrete Markov diffusion models since it aims to propose an accelerated algorithm for discrete diffusion with DNDM. Other text generation models, such as continuous diffusion models or auto-regressive models, are not considered in this paper.
- This paper focuses on acceleration while maintaining good sample quality. The hyper parameter regions with poor sample qualities are not explored in this paper.

By highlighting these limitations, this paper aims to clearly scope its contributions and spark future work on addressing these important challenges with discrete diffusion models for generative modeling.

A Related Work

Continous Diffusion Models. Generative modeling via continuous-time stochastic process has been investigated thoroughly in a series of work (Movellan, 2008; Lyu, 2012; Sohl-Dickstein et al., 2009; Bengio et al., 2014; Alain et al., 2016; ALIAS PARTH GOYAL et al., 2017; Bordes et al., 2017). The two lines of probabilistic modeling, *denoising diffusion probabilistic model* (Sohl-Dickstein et al., 2015; Ho et al., 2020) and *score matching with Langevin dynamics* (Song and Ermon, 2019) are unified by Song et al. (2020b) through introducing the SDE framework for SGM. Based on it, subsequent works (Dockhorn et al., 2021; Nachmani et al., 2021; Vahdat et al., 2021) introduced a more complex diffusion process to improve the generation speed and quality. On the other hand, the score-based sampling process is time-consuming and has attracted much attention for improvements in speed (San-Roman et al., 2021; Watson et al., 2021; Kong and Ping, 2021; Karras et al., 2022; Song et al., 2023). "Gotta go fast" (GGF), an SDE solver with adaptive step size tailored to SGM, is proposed in Jolicoeur-Martineau et al. (2021). Song et al. (2020a) introduced a non-Markov diffusion process that corresponds to a deterministic sampling process, enabling the generation of high-quality samples more rapidly. Dockhorn et al. (2022); Liu et al. (2022) proposed a high-order SDE/ODE solver to achieve lower discretization error. Lu et al. (2022); Zhang and Chen (2022) leveraged the semi-linear structure of reverse ODE to reduce the discretization error and achieve state-of-the-art sampling speed.

Discrete Diffusion Models. Research on discrete diffusion models was initiated by Sohl-Dickstein et al. (2015), who investigated diffusion processes over binary random variables. The methodology was expanded upon by Ho et al. (2020), integrating categorical random variables through transition matrices with uniform probabilities. Though Song et al. (2020a) suggested a similar extension in

their supplementary content, they abstained from experimenting with this model type. Later on, Austin et al. (2021) unveiled a more intricate framework for diffusion concerning categorical random variables, enhancing the discrete diffusion models by merging them with Masked language models (MLMs). Contemporary research has furthered this domain by introducing features like editing-based operations (Jolicoeur-Martineau et al., 2021; Reid et al., 2022), auto-regressive diffusion models (Hoogeboom et al., 2021a; Ye et al., 2023), the evolution of a continuous-time structure (Campbell et al., 2022), and the exploration of neural network analogs for learning (Sun et al., 2022). Additionally, Zheng et al. (2023) introduced a re-parameterized loss and an associated sampling technique, attaining commendable outcomes in fewer iterations. Our contributions run parallel to these aforementioned studies.

B Additional details of Discrete Diffusion

In our paper, we treat all the $\mathbf{x}, \mathbf{q}_{\text{noise}}$ as a row vector and treat $\mathbb{1}$ as a column vector with all elements equal 1.

B.1 Comparison between D3PM and DNDM

In Section 3.1, we introduced two different diffusion processes, the Markov process in (1) and the non-Markov process in (6). In this section, we explain why they are different but result in the same joint distribution of $(\mathbf{x}_0, \mathbf{x}_t)$ for every time step t. Since $\mathbf{q}(\mathbf{x}_0)$ keeps the same, we only need to prove that the conditional distribution $\mathbf{q}(\mathbf{x}_t|\mathbf{x}_0)$ is the same for the two processes.

Markov Process. 1 is a Markov process since $\mathbf{w}_n$ is independent with $\mathbf{x}_{t-1}, \ldots, \mathbf{x}_0$, so $\mathbf{x}_t$ is independent of all the past states given the present state. This can also be inferred from the following distribution, which does not depend on $\mathbf{x}_0, \ldots, \mathbf{x}_{t-2}$,

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \text{Cat}\big(\mathbf{x}_t; \mathbf{p} = \beta_t \mathbf{x}_{t-1} + (1-\beta_t)\mathbf{q}_{\text{noise}}\big). \tag{13}$$

Denote $\mathbf{Q}_t := \beta_t \mathbf{I} + (1-\beta_t)\,\mathbb{1}\,\mathbf{q}_{\text{noise}}$, then we have that

$$\mathbf{x}_{t-1}\mathbf{Q}_t = \beta_t \mathbf{x}_{t-1} + (1-\beta_t)\mathbf{x}_{t-1}\,\mathbb{1}\,\mathbf{q}_{\text{noise}} = \beta_t \mathbf{x}_{t-1} + (1-\beta_t)\mathbf{q}_{\text{noise}},$$

where the last equality holds due to the fact that $\mathbf{x}_{t-1}$ is a one hot vector and thus $\mathbf{x}_{t-1}\,\mathbb{1} = 1$. Therefore, we can rewrite (13) as $q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \text{Cat}\big(\mathbf{x}_t; \mathbf{p} = \mathbf{x}_{t-1}\mathbf{Q}_t\big)$. Then, it is a Markov process with transition kernel $\mathbf{Q}_t$. So $q(\mathbf{x}_t|\mathbf{x}_0) = \text{Cat}\big(\mathbf{x}_t; \mathbf{p} = \mathbf{x}_0\mathbf{Q}_0 \ldots \mathbf{Q}_t\big)$ (Austin et al., 2021). We can then have that

$$\begin{aligned}\mathbf{Q}_0 \ldots \mathbf{Q}_t &= [\beta_0 \mathbf{I} + (1-\beta_0)\,\mathbb{1}\,\mathbf{q}_{\text{noise}}] \ldots [\beta_t \mathbf{I} + (1-\beta_t)\,\mathbb{1}\,\mathbf{q}_{\text{noise}}] \\ &= \Pi_{s=0}^t \beta_s \mathbf{I} + (1 - \Pi_{s=0}^t \beta_s)\,\mathbb{1}\,\mathbf{q}_{\text{noise}},\end{aligned}$$

where the last equality holds since identity matrix $\mathbf{I}$ multiplying any vector equals the vector itself and $\mathbb{1}\,\mathbf{q}_{\text{noise}}\,\mathbb{1}\,\mathbf{q}_{\text{noise}} = \mathbb{1}(\mathbf{q}_{\text{noise}}\,\mathbb{1})\mathbf{q}_{\text{noise}} = \mathbb{1}\,\mathbf{q}_{\text{noise}}$. Therefore, we have that

$$q(\mathbf{x}_t|\mathbf{x}_0) = \text{Cat}\big(\mathbf{x}_t; \mathbf{p} = \Pi_{s=0}^t \beta_s \mathbf{x}_0 + (1 - \Pi_{s=0}^t \beta_s)\mathbf{q}_{\text{noise}}\big) = \text{Cat}\big(\mathbf{x}_t; \mathbf{p} = \alpha_t \mathbf{x}_0 + (1-\alpha_t)\mathbf{q}_{\text{noise}}\big),$$

where the last equality holds due to the definition $\alpha_t = \Pi_{s=0}^t \beta_s$. This gives rise to why the Markov process (1) results in conditional distribution $q(\mathbf{x}_t|\mathbf{x}_0) = \text{Cat}\big(\mathbf{x}_t; \mathbf{p} = \alpha_t \mathbf{x}_0 + (1-\alpha_t)\mathbf{q}_{\text{noise}}\big)$.

Non-Markov Process. Recall that our DNDM is defined by

$$\mathbf{x}_t = b_t \mathbf{x}_{t-1} + (1-b_t)\mathbf{w},$$

where $\mathbf{w}$ is fixed for any time t. Therefore, $\mathbf{w}$ is no longer independent with $\mathbf{x}_0, \ldots, \mathbf{x}_{t-1}$. Therefore, we can't define the transition kernel and compute $\mathbf{q}(\mathbf{x}_t|\mathbf{x}_0)$ by using the property of Markov. Therefore, we need to advance the technique to calculate the conditional distribution.

Proof of Theorem 3.1. By (6), we can derive the following explicit expression for a recursive sequence,

$$\begin{aligned}\mathbf{x}_t &= b_1 \ldots b_t \mathbf{x}_{0,n} + \sum_{s=1}^{t} (1-b_s) b_{s+1} \ldots b_t \mathbf{w} \\ &= b_1 \ldots b_t \mathbf{x}_0 + (1 - b_1 \ldots b_t)\mathbf{w}\end{aligned}$$

$$= a_t \mathbf{x}_0 + (1 - a_t)\mathbf{w},$$

where second equality is by cancellation of terms, the last inequality holds by defining $a_t = b_1 \dots b_t$. Since a_t either equals to 1 or 0. Besides, a_t equals 1 if and only if $b_1 = b_2 = \ldots = b_t = 1$, so we have that a_t follows Bernoulli distribution $\text{Bernoulli}(\beta_1 \dots \beta_t) = \text{Bernoulli}(\alpha_t)$ where $\alpha_t = \Pi_{i=1}^{t} \beta_s$. Therefore, we can conclude that $\mathbf{q}(\mathbf{x}_t|\mathbf{x}_0) = \text{Cat}\big(\mathbf{x}_t; \mathbf{p} = \alpha_t \mathbf{x}_0 + (1 - \alpha_t)\mathbf{q}_{\text{noise}}\big)$, which completes the proof. □

Comparison between D3PM-Absorb and DNDM. Recall the forward processes of D3PM and DNDM as follows:

$$\begin{aligned} \text{D3PM}: &\quad \mathbf{x}_t = b_t \mathbf{x}_{t-1} + (1 - b_t)\mathbf{w}_t, \quad \forall t = 1 \dots T, \\ \text{DNDM}: &\quad \mathbf{x}_t = b_t \mathbf{x}_{t-1} + (1 - b_t)\mathbf{w}, \quad \forall t = 1 \dots T. \end{aligned}$$

For absorbing diffusion where $\mathbf{w} = [\text{Mask}]$, DNDM's forward process becomes equivalent to D3PM since $\mathbf{w}_t = \mathbf{w} = [\text{Mask}]$ in this special case. However, for multinomial diffusion or other diffusion processes where $\mathbf{w}_t \neq \mathbf{w}$, these two processes exhibit different behaviors. In addition, even for absorbing diffusion, our proposed reverse sampling algorithm for DNDM is still different from that for D3PM.

To elucidate the key differences between the sampling algorithm in DNDM and that in D3PM for absorbing diffusion, let's directly compare the algorithms:

- For the D3PM-Absorb algorithm: We begin with an all $[\text{Mask}]$ sequence. At each time step t, we sample $\mathbf{x}_0 \sim p_\theta(\mathbf{x}_0|\mathbf{x}_t)$. If $\mathbf{x}_t = [\text{Mask}]$, $\mathbf{x}_{t-1}$ transitions to $[\text{Mask}]$ with probability $(1 - \alpha_{t-1})/(1 - \alpha_t)$ and to $\mathbf{x}_0$ with probability $(\alpha_{t-1} - \alpha_t)/(1 - \alpha_t)$. If $x_t \neq [\text{Mask}]$, it remains unchanged.
- For the DNDM-Absorb algorithm: We also start with an all $[\text{Mask}]$ sequence, but crucially, we first determine the transition time set. During sampling, if $\mathbf{x}_t = [\text{Mask}]$, the transition probabilities for $\mathbf{x}_{t-1}$ are identical to D3PM. However, we only sample $\mathbf{x}_0 \sim p_\theta(\mathbf{x}_0|\mathbf{x}_t)$ when at least one token needs to change, as determined by our pre-computed transition set. This selective sampling is the key to our algorithm's efficiency.

Therefore, you can see that DNDM will skip many steps during the sampling process to avoid function evaluation and save computational cost. Even though the forward process of DNDM is the same as that of D3PM for absorbing diffusion, our DNDM approach introduces an algorithm design in the sampling process by pre-computing the transition time set and selectively applying function evaluations. This distinguishes DNDM from D3PM algorithm, offering a more computationally efficient approach to inference in discrete diffusion.

Comparison between DDIM and DNDM for Multinomial Diffusion. While there are similarities between DNDM and DDIM (Appendix A), they are fundamentally different models, and DNDM is not a special case of DDIM. DNDM introduces a novel framework specifically designed for discrete spaces, while DDIM was originally developed for continuous diffusion models. The key differences for multinomial diffusion are as follows.

- DDIM: Following Song et al. (2020a) (eq. 19 in Appendix A), $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \text{Cat}(\sigma_t \mathbf{x}_t + (\alpha_{t-1} - \sigma_t \alpha_t)\mathbf{x}_0 + ((1 - \alpha_{t-1}) - (1 - \alpha_t)\sigma_t)\mathbf{1}_K)$. Even with $\sigma_t = \frac{1-\alpha_{t-1}}{1-\alpha_t}$, the process remains stochastic: $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \text{Cat}(\sigma_t \mathbf{x}_t + (1-\sigma_t)\mathbf{x}_0)$. This means at every step, there's a probability of choosing $\mathbf{x}_0$, regardless of whether it has transitioned to $\mathbf{x}_0$ or not. Unlike Absorbing discrete diffusion, no $[\text{Mask}]$ exists in multinomial diffusion. Therefore, DDIM cannot distinguish whether $\mathbf{x}_t$ already equals $\mathbf{x}_0$ or not. In particular, although the sampling process becomes less stochastic in the DDIM setting, it will still be predicted $\mathbf{x}_0$ with high probability $1 - \sigma_t = \frac{\alpha_{t-1}-\alpha_t}{1-\alpha_t}$.
- DNDM: Achieves full de-randomization using transition time τ, where:

$$\mathbf{x}_{t-1} = \mathbb{1}(\tau = t)\mathbf{x}_0 + \mathbb{1}(\tau \neq t)\mathbf{x}_t, \quad \text{with } P(\tau = t) = \alpha_{t-1} - \alpha_t. \tag{14}$$

  This crucial difference allows DNDM to achieve full de-randomization once τ is sampled, leading to a deterministic evolution that DDIM cannot achieve.

While DNDM and DDIM are both non-Markov models for multinomial diffusion, their fundamental approaches to and achievements in de-randomization differ significantly in discrete spaces.

B.2 Training Objective

Hoogeboom et al. (2021b) utilized L_t derived from the negative variational bound. In detail,

$$L_t = \mathrm{KL}\big(\mathrm{Cat}(\mathbf{x}; \mathbf{p} = \boldsymbol{\theta}_{\mathrm{post}}(\mathbf{x}_t, \mathbf{x}_0)\big|\mathrm{Cat}(\mathbf{x}; \mathbf{p} = \boldsymbol{\theta}_{\mathrm{post}}(\mathbf{x}_t, \widehat{\mathbf{x}}_0)\big), \tag{15}$$

where $\widehat{\mathbf{x}}_0 \sim p_{\boldsymbol{\theta}}(\cdot|\mathbf{x}_t)$, $\boldsymbol{\theta}_{\mathrm{post}} = (\beta_t \mathbf{x}_t + (1-\beta_t)/K\,\mathbb{1}^\top) \odot (\alpha_{t-1}\mathbf{x}_0 + (1-\alpha_{t-1})/K\,\mathbb{1}^\top)$ and $\boldsymbol{\theta}_{\mathrm{post}} = (\beta_t \mathbf{x}_t + (1-\beta_t)/K\,\mathbb{1}^\top) \odot (\alpha_{t-1}\widehat{\mathbf{x}}_0 + (1-\alpha_{t-1})/K\,\mathbb{1}^\top)$. This loss evolves KL divergence between two categorical distributions.

Building on this foundation, Austin et al. (2021) introduced an auxiliary denoising objective to strengthen the data predictions $\mathbf{x}_0$ at each time step. In detail, the auxiliary objective is as follows,

$$\mathbb{E}_{q(\mathbf{x}_t, \mathbf{x}_0)}\Big[-\log p_{\boldsymbol{\theta}}(\mathbf{x}_0|\mathbf{x}_t)\Big],$$

where the auxiliary loss term is minimized exactly when $p_\theta(\cdot|\mathbf{x}_t)$ has all its mass on the data point $\mathbf{x}_0$.

Furthering the advancements, Zheng et al. (2023) put forth a reparametrized loss L_t that incorporates a re-weighted parameter λ_t. The detailed loss is

$$\overline{L}_t = \lambda_{t-1}\mathbb{E}_{\mathbf{x}_{t-1}, \mathbf{x}_t \sim q(\cdot|\mathbf{x}_0)} \mathrm{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)|p_\theta^{(t)}(\mathbf{x}_{t-1}|\mathbf{x}_t)).$$

This loss can be related to the standard multi-class cross-entropy loss function, which is also simple and powerful. That's why we consider Zheng et al. (2023) as the baseline model.

In Section 3.3, we consider the continuous-time forward and backward process. Based on that, we were motivated to analyze the infinite limit of the average loss $\lim_{t\to\infty} \frac{1}{T}\sum_{t=1}^{T} L_t$. We find that the new loss can provide a better checkpoint than the loss averaged on the finite step on some tasks.

B.3 Calculation of the Evidence Lower Bound

B.3.1 Finite Time DNDM

In this section, we derive the evidence lower bound (ELBO) for our model. The derivatives are inspired by the reasoning in DDIM (Song et al., 2020a). Specifically, We denote the generative process as $p_\theta(\mathbf{x}_{0:T}|\tau) = p_\theta^{(T)}(\mathbf{x}_T|\tau)\prod_{t=1}^{T} p_\theta^{(t)}(\mathbf{x}_{t-1}|\mathbf{x}_t, \tau)$. Here, $p_\theta^{(T)}$ is the pure noise and $p_\theta^{(t)}(\mathbf{x}_{t-1}|\mathbf{x}_t, \tau) = q(\mathbf{x}_{t-1}|\mathbf{x}_t, \widehat{\mathbf{x}}_0, \tau)$, where $\widehat{\mathbf{x}}_0$ is given by a neural network $p_{\boldsymbol{\theta}}$, i.e., $\widehat{\mathbf{x}}_0 = p_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$. Notice that by Jensen's inequality,

$$\log p_\theta(\mathbf{x}_0) = \log \mathbb{E}_{\tau\sim\mathcal{D}_\tau}[p_\theta(\mathbf{x}_0|\tau)] \geq \mathbb{E}_{\tau\sim\mathcal{D}_\tau}[\log p_\theta(\mathbf{x}_0|\tau)]. \tag{16}$$

The evidence lower bound inequality gives

$$\log p_\theta(\mathbf{x}_0|\tau) \geq \mathbb{E}_{\mathbf{x}_{1:T}\sim q(\mathbf{x}_{1:T}|\mathbf{x}_0,\tau)} \log \frac{p_\theta(\mathbf{x}_{0:T}|\tau)}{q(\mathbf{x}_{1:T}|\mathbf{x}_0, \tau)}. \tag{17}$$

Plugging (17) into (16) gives the following ELBO,

$$\log p_\theta(\mathbf{x}_0) \geq \mathbb{E}_{\tau\sim\mathcal{D}_\tau}\mathbb{E}_{\mathbf{x}_{1:T}\sim q(\mathbf{x}_{1:T}|\mathbf{x}_0,\tau)} \log \frac{p_\theta(\mathbf{x}_{0:T}|\tau)}{q(\mathbf{x}_{1:T}|\mathbf{x}_0, \tau)} := \text{ELBO}.$$

We factorize the p_θ and q by

$$p_\theta(\mathbf{x}_{0:T}|\tau) = p_\theta^{(T)}(\mathbf{x}_T|\tau)\prod_{t=1}^{T} p_\theta^{(t)}(\mathbf{x}_{t-1}|\mathbf{x}_t, \tau),$$

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0, \tau) = q(\mathbf{x}_T|\mathbf{x}_0, \tau)\prod_{t=2}^{T} q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0, \tau).$$

Here q admits such a decomposition due to our definition of the diffusion process in (6), which introduce the following reverse process:

$$\mathbf{x}_{t-1} = \mathbb{1}(\tau = t)\mathbf{x}_0 + \mathbb{1}(\tau \neq t)\mathbf{x}_t.$$

Therefore, $\mathbf{x}_{1:T}$ is Markovian when conditioned on $\mathbf{x}_0$ and τ. Based on the factorization, we have

$$\begin{aligned}
\text{ELBO} &= \mathbb{E}_{\tau\sim\mathcal{D}_\tau}\mathbb{E}_{\mathbf{x}_{1:T}\sim q(\mathbf{x}_{1:T}|\mathbf{x}_0,\tau)}\Big[\log p_\theta^{(T)}(\mathbf{x}_T|\tau) + \sum_{t=1}^{T}\log p_\theta^{(t)}(\mathbf{x}_{t-1}|\mathbf{x}_t,\tau) \\
&\qquad - \log q(\mathbf{x}_T|\mathbf{x}_0,\tau) - \sum_{t=2}^{T}\log q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0,\tau)\Big] \\
&= \mathbb{E}_{\tau\sim\mathcal{D}_\tau}\mathbb{E}_{\mathbf{x}_{1:T}\sim q(\mathbf{x}_{1:T}|\mathbf{x}_0,\tau)}\Big[\log p_\theta^{(1)}(\mathbf{x}_0|\mathbf{x}_1,\tau) + \sum_{t=2}^{T}\log\frac{p_\theta^{(t)}(\mathbf{x}_{t-1}|\mathbf{x}_t,\tau)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0,\tau)} \\
&\qquad + \log\frac{p_\theta^{(T)}(\mathbf{x}_T|\tau)}{q(\mathbf{x}_T|\mathbf{x}_0,\tau)}\Big] \\
&= \mathbb{E}_{\tau\sim\mathcal{D}_\tau}\mathbb{E}_{\mathbf{x}_1\sim q(\cdot|\mathbf{x}_0,\tau)}\log p_\theta^{(1)}(\mathbf{x}_0|\mathbf{x}_1,\tau) \\
&\qquad + \sum_{t=2}^{T}\mathbb{E}_{\mathbf{x}_{t-1},\mathbf{x}_t\sim q(\cdot|\mathbf{x}_0,\tau)}\log\frac{p_\theta^{(t)}(\mathbf{x}_{t-1}|\mathbf{x}_t,\tau)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0,\tau)} + \text{const} \\
&= \mathbb{E}_{\tau\sim\mathcal{D}_\tau}\underbrace{\mathbb{E}_{\mathbf{x}_1\sim q(\cdot|\mathbf{x}_0,\tau)}\log p_\theta^{(1)}(\mathbf{x}_0|\mathbf{x}_1,\tau)}_{\overline{\mathcal{L}}_1} \\
&\qquad - \sum_{t=2}^{T}\mathbb{E}_{\tau\sim\mathcal{D}_\tau}\underbrace{\mathbb{E}_{\mathbf{x}_{t-1},\mathbf{x}_t\sim q(\cdot|\mathbf{x}_0,\tau)}\mathrm{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0,\tau)|p_\theta^{(t)}(\mathbf{x}_{t-1}|\mathbf{x}_t,\tau))}_{\overline{\mathcal{L}}_t} + \text{const.}
\end{aligned}$$

By a slight abuse of notations we use $q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0), p_\theta^{(t)}(\mathbf{x}_0|\mathbf{x}_1)$ to indicate the distribution of the diffusion process defined in Zheng et al. (2023), that is, the standard Markov discrete diffusion process. In particular, we have

$$\begin{aligned}
\overline{\mathcal{L}}_1 &= \begin{cases} \mathbb{E}_{\mathbf{x}_1\sim q(\cdot|\mathbf{x}_0)}\log p_\theta^{(1)}(\mathbf{x}_0|\mathbf{x}_1), & \tau = 1, \\ \text{const}, & \tau \neq 1. \end{cases} \\
\overline{\mathcal{L}}_t &= \begin{cases} \mathbb{E}_{\mathbf{x}_{t-1},\mathbf{x}_t\sim q(\cdot|\mathbf{x}_0)}\mathrm{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)|p_\theta^{(t)}(\mathbf{x}_{t-1}|\mathbf{x}_t)), & \tau = t, \\ 0, & \tau \neq t. \end{cases}
\end{aligned}$$

Thus, we can obtain that

$$\begin{aligned}
\text{ELBO} =& \mathbb{P}(\tau=1)\cdot\underbrace{\mathbb{E}_{\mathbf{x}_1\sim q(\cdot|\mathbf{x}_0)}\log p_\theta^{(1)}(\mathbf{x}_0|\mathbf{x}_1)}_{\mathcal{L}_1} \\
&- \sum_{t=2}^{T}\mathbb{P}(\tau=t)\cdot\underbrace{\mathbb{E}_{\mathbf{x}_{t-1},\mathbf{x}_t\sim q(\cdot|\mathbf{x}_0)}\mathrm{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)|p_\theta^{(t)}(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{\mathcal{L}_t} + \text{const.}
\end{aligned}$$

Here $\mathcal{L}_t$ matches the loss terms in Zheng et al. (2023). In the practical training process, Zheng et al. (2023) samples t from Unif$\{1,\cdots,T\}$ in each iteration and optimizes $\lambda_t\cdot\mathcal{L}_t$, where λ_t's are weights. Thus, when we sample τ and optimize $\mathcal{L}_\tau$, our ELBO indeed leads to the same training objective as Zheng et al. (2023) up to reweighting. Since Zheng et al. (2023) is a parametrization of existing works (Austin et al., 2021; Hoogeboom et al., 2021b), our training objective indeed aligns with previous discrete diffusion models.

B.3.2 Continous Time DNDM

In Section B.3, we derived an ELBO for DNDM and its accelerated algorithm defined in Section 3.1 and 3.2. While for finite sampling steps, we can decompose the diffusion process via the sampling steps $1,\ldots,T$ in (17), it becomes intractable for continuous Time DNDM (Infinite steps $T\to\infty$). Therefore, we can formulate the ELBO of continuous time DNDM by decomposing the transition times. The idea of decomposition of transition times follows Hoogeboom et al. (2021a), but their

proof is only applicable to absorbing discrete diffusion, while ours can deal with discrete diffusion with various noise q_{noise} including multinomial diffusion.

In Section B.3, we only consider the case of a single token $\mathbf{x} \in \mathbb{R}^K$ for simplicity as we decompose with the sampling steps T. In this section, we decompose over the transition time τ. Therefore, we need to consider a sentence with multiple tokens $\mathbf{x}_{t,1:N} = [\mathbf{x}_{t,1}, \dots, \mathbf{x}_{t,N}]$ where $\mathbf{x}_{t,n}$ is the n-th token and N is the sequence length. Recall that we defined the transition time set $\mathcal{T} = \{\tau_n\}_{n=1}^N$ in Section 3.2. We arrange τ_n to obtain an ordered sequence τ_{n_k}, where $0 = \tau_{n_0} < \tau_{n_1} < \tau_{n_2} < \ldots < \tau_{n_N} = T$. Then conditioning on the transition time set $\mathcal{T} = \{\tau_1, \dots, \tau_N\}$, we have that

$$p_\theta(\mathbf{x}_{0:T,1:N}|\mathcal{T}) = p_\theta(\mathbf{x}_{\tau_{n_N},1:N}|\mathcal{T}) \prod_{s=N,\dots,1} p_\theta(\mathbf{x}_{\tau_{n_{s-1}},1:N}|\mathbf{x}_{\tau_{n_s},1:N}, \mathcal{T}),$$

where we omit the time superscript of p for simplicity. Then, the evidence lower bound inequality gives

$$\log p_\theta(\mathbf{x}_{0,1:N}|\mathcal{T}) \geq \mathbb{E}_{\mathbf{x}_{\tau_{n_1}:T,1:N} \sim q(\mathbf{x}_{\tau_{n_1}:T,1:N}|\mathbf{x}_{0,1:N},\mathcal{T})} \log \frac{p_\theta(\mathbf{x}_{0:T,1:N}|\mathcal{T})}{q(\mathbf{x}_{\tau_{n_1}:T,1:N}|\mathbf{x}_{0,1:N}, \mathcal{T})}. \tag{18}$$

By Jensen's inequality, we have

$$\log p_\theta(\mathbf{x}_{0,1:N}) = \log \mathbb{E}_{\tau_1,\dots,\tau_n \sim \mathcal{D}_\tau}[p_\theta(\mathbf{x}_{0,1:N}|\mathcal{T})] \geq \mathbb{E}_{\tau_1,\dots,\tau_n \sim \mathcal{D}_\tau}[\log p_\theta(\mathbf{x}_0|\mathcal{T})]. \tag{19}$$

Plugging (18) into (19) gives the following ELBO,

$$\log p_\theta(\mathbf{x}_{0,1:N}) \geq \mathbb{E}_{\tau_1,\dots,\tau_n \sim \mathcal{D}_\tau} \mathbb{E}_{\mathbf{x}_{\tau_{n_1}:T} \sim q(\mathbf{x}_{\tau_{n_1}:T}|\mathbf{x}_0,\mathcal{T})} \log \frac{p_\theta(\mathbf{x}_{0:T}|\mathcal{T})}{q(\mathbf{x}_{\tau_{n_1}:T}|\mathbf{x}_0, \mathcal{T})} := \text{ELBO}.$$

We factorize the p_θ and q by

$$p_\theta(\mathbf{x}_{0:T,1:N}|\mathcal{T}) = p_\theta(\mathbf{x}_{T,1:N}|\mathcal{T}) \prod_{s=N,\dots,1} p_\theta(\mathbf{x}_{\tau_{n_{s-1}},1:N}|\mathbf{x}_{\tau_{n_s},1:N}, \mathcal{T}),$$

$$q(\mathbf{x}_{\tau_{n_1}:T,1:N}|\mathbf{x}_{0,1:N}, \mathcal{T}) = q(\mathbf{x}_{T,1:N}|\mathbf{x}_0, \mathcal{T}) \prod_{s=N,\dots,2} q(\mathbf{x}_{\tau_{n_{s-1}},1:N}|\mathbf{x}_{\tau_{n_s},1:N}, \mathbf{x}_{0,1:N}, \mathcal{T}).$$

Therefore, we have

$$\begin{aligned}
\text{ELBO} &= \mathbb{E}_{\tau_1,\dots,\tau_n \sim \mathcal{D}_\tau} \mathbb{E}_{\mathbf{x}_{\tau_{n_1}:T,1:N} \sim q(\mathbf{x}_{\tau_{n_1}:T,1:N}|\mathbf{x}_{0,1;N},\mathcal{T})} \Big[\log p_\theta(\mathbf{x}_{T,1:N}|\mathcal{T}) \\
&\quad + \sum_{s=1}^N \log p_\theta(\mathbf{x}_{\tau_{n_{s-1}},1:N}|\mathbf{x}_{\tau_{n_s},1:N}, \mathcal{T}) - \log q(\mathbf{x}_{T,1:N}|\mathbf{x}_{0,1:N}, \mathcal{T}) \\
&\quad - \sum_{s=2}^N \log q(\mathbf{x}_{\tau_{n_{s-1}},1:N}|\mathbf{x}_{\tau_{n_s},1:N}, \mathbf{x}_{0,1:N}, \mathcal{T}) \Big] \\
&= \mathbb{E}_{\tau_1,\dots,\tau_n \sim \mathcal{D}_\tau} \mathbb{E}_{\mathbf{x}_{\tau_{n_1}:T,1:N} \sim q(\mathbf{x}_{\tau_{n_1}:T,1:N}|\mathbf{x}_{0,1:N},\mathcal{T})} \Big[\log p_\theta(\mathbf{x}_{0,1:N}|\mathbf{x}_{1,1:N}, \mathcal{T}) \\
&\quad + \sum_{s=2}^N \log \frac{p_\theta(\mathbf{x}_{\tau_{n_{s-1}},1:N}|\mathbf{x}_{\tau_{n_s},1:N}, \mathcal{T})}{q(\mathbf{x}_{\tau_{n_{s-1}},1:N}|\mathbf{x}_{\tau_{n_s},1:N}, \mathbf{x}_{0,1:N}, \mathcal{T})} + \log \frac{p_\theta(\mathbf{x}_{T,1:N}|\mathcal{T})}{q(\mathbf{x}_{T,1:N}|\mathbf{x}_{0,1:N}, \mathcal{T})} \Big] \\
&= \mathbb{E}_{\tau_1,\dots,\tau_n \sim \mathcal{D}_\tau} \mathbb{E}_{\mathbf{x}_{1,1:N} \sim q(\cdot|\mathbf{x}_{0,1:N},\mathcal{T})} \log p_\theta(\mathbf{x}_{0,1:N}|\mathbf{x}_{1,1:N}, \mathcal{T}) \\
&\quad + \sum_{s=2}^N \mathbb{E}_{\mathbf{x}_{\tau_{n_{s-1}},1:N}, \mathbf{x}_{\tau_{n_s},1:N} \sim q(\cdot|\mathbf{x}_{0,1:N},\mathcal{T})} \log \frac{p_\theta(\mathbf{x}_{\tau_{n_{s-1}},1:N}|\mathbf{x}_{\tau_{n_s},1:N}, \mathcal{T})}{q(\mathbf{x}_{\tau_{n_{s-1}},1:N}|\mathbf{x}_{\tau_{n_s},1:N}, \mathbf{x}_{0,1:N}, \mathcal{T})} + \text{const} \\
&= \mathbb{E}_{\tau_1,\dots,\tau_n \sim \mathcal{D}_\tau} \mathbb{E}_{\mathbf{x}_{1,1:N} \sim q(\cdot|\mathbf{x}_{0,1:N},\mathcal{T})} \log p_\theta(\mathbf{x}_{0,1:N}|\mathbf{x}_{1,1:N}, \mathcal{T}) \\
&\quad - \sum_{s=2}^N \mathbb{E}_{\tau_1,\dots,\tau_n \sim \mathcal{D}_\tau} \mathbb{E}_{\mathbf{x}_{\tau_{n_{s-1}},1:N}, \mathbf{x}_{\tau_{n_s},1:N} \sim q(\cdot|\mathbf{x}_{0,1:N},\mathcal{T})} \\
&\qquad \text{KL}(q(\mathbf{x}_{\tau_{n_{s-1}},1:N}|\mathbf{x}_{\tau_{n_s},1:N}, \mathbf{x}_{0,1:N}, \mathcal{T})|p_\theta(\mathbf{x}_{\tau_{n_{s-1}},1:N}|\mathbf{x}_{\tau_{n_s},1:N}, \mathcal{T})) + \text{const}.
\end{aligned} \tag{20}$$

Remark B.1. (20) represents the ELBO utilized by the DNDM-C architecture. As our transition times τ_n are independently and identically drawn from the distribution $\mathcal{D}_\tau$, we are unable to further decompose (20) into a loss function related to the position information $1:N$, as was accomplished by Hoogeboom et al. (2021a).

C Choice of the Transition Time

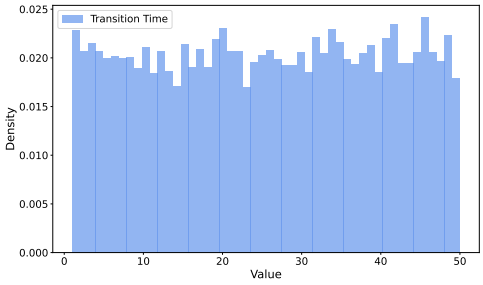
Transition time τ in Definition 3.2 plays an important role in DNDM. In this section, we provide a deeper discussion of the transition time. We first give a proof of the Theorem 3.6.

Proof of Theorem 3.6. By the definition of τ, we know that $\tau_n = t$ is equivalent to $b_{0,n} = 1, \dots, b_{t-1,n} = 1$ and $b_{t,n} = 0$. Since $\{b_{t,n}\}_{t=0}^T$ is independent for different n by definition, each τ_n is also independent. Therefore, we drop the subscript n for simplicity. On the other hand if $b_0 = 1, \dots, b_{t-1} = 1$ and $b_t = 0$ we can also conclude that $\tau = t$. Therefore, we have that

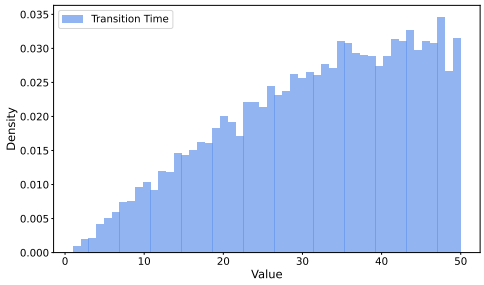
$$\begin{aligned}\mathbb{P}(\tau = t) &= \mathbb{P}(b_0 = 1, \dots, b_{t-1} = 1, b_t = 0) \\ &= \left[\Pi_{s=1}^{t-1}\beta_s\right] \cdot (1 - \beta_t) \\ &= \Pi_{s=1}^{t-1}\beta_s - \Pi_{s=1}^{t}\beta_s \\ &= \alpha_{t-1} - \alpha_t,\end{aligned}$$

where the second equality is due to $b_s, s = 1, 2, \dots, t$ are independent random variable following Bernoulli(β_s) distribution and the last equality is by the definition of $\alpha_t = \Pi_{s=1}^{t}\beta_s$. □

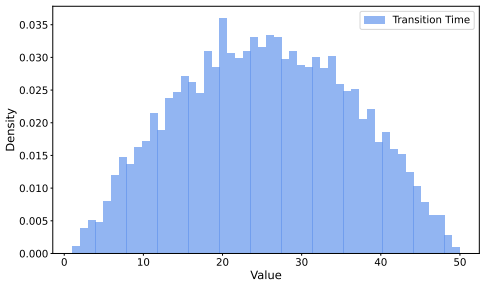
Notice that α_t is a decreasing sequence in the 0 to 1 range. Therefore, $\mathbb{P}(\tau = t) \in [0, 1]$ for any $t \in \{1, \dots, T\}$. Besides $\sum \mathbb{P}(\tau = t) = \sum_{t=1}^{T} \left(\alpha_{t-1} - \alpha_t\right) = \alpha_0 - \alpha_T = 1$. Therefore, the derived distribution is valid as long as the α_t is decreasing from 1 to 0.



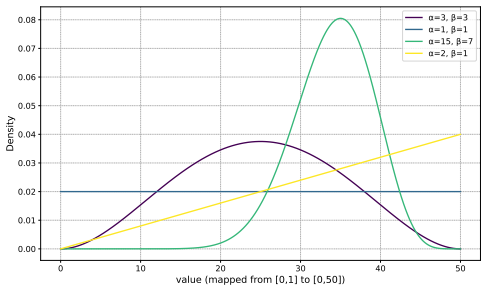
(a) $\alpha_t = 1 - t/T$



(b) $\alpha_t = \cos(\pi * t/2T)$



(c) $\alpha_t = \cos^2(\pi * t/2T)$



(d) Beta Distribution with Different Parameter

Figure 3: Different distribution of transition time for $T = 50$. $a), b), c)$ The transition time sampled 1K times under the different α_t schedule. d) The approximated transition time for $t = 1, \dots, T$ using different hypter-parameters.

From Theorem 3.6, we discern that the nature of the diffusion model scheduler, α_t, clarifies the distribution of τ.

Linear α schedule. This is a schedule studied in Austin et al. (2021), where $\alpha_t = 1 - t/T$. This will result in $\mathbb{P}(\tau_n = t) = 1/T$ for every t in the range 1 to T. As a result, transition time distributes uniformly across each moment in the set $\{1, \dots, T\}$. This can be verified in a) of Figure 3.

Cosine α schedule. This is a schedule studied in Hoogeboom et al. (2021b), where $\alpha_t = \cos(\pi * t/2T)$. For numerical consideration of the noise, a small offset s is added, i.e., $\alpha_t = f(t)/f(0)$

where $f(t) = \cos((s + t/T)/(1 + s) * \pi/2)$. As shown in b) of Figure 3, the transition time will concentrate more on the large T.

Cosine square α schedule. This is a schedule studied in Zheng et al. (2023), where $\alpha_t = \cos^2(\pi * t/2T)$, which motivated by Nichol and Dhariwal (2021). Again, for numerical consideration of the noise, a small offset s is added, i.e., $\alpha_t = f(t)/f(0)$ where $f(t) = \cos^(s + t/T)/(1 + s) * \pi/2)$. As shown in c) of Figure 3, the transition time will concentrate more on the middle of the range.

Generally, if we express α_t as $g(t/T)$, then we can simplify to $\mathbb{P}(\tau = t) = g((t-1)/T) - g(t/T)$, which further refines to $(1/T)|g'(t/T)| + o(1/T)$. This indicates that transitions are more likely where $|g'|$ is large. Such a mathematical finding can match our observation in Figure 3.

In practice, we find that the shape of the transition time doesn't need to match the theoretical prediction schedule exactly. As we can see from d) in Figure 3. A reshaped Beta distribution can approximate all the transition time distributions in a fixed range. We first extract a time $t \in [0, 1]$ from a Beta distribution, then adjust these samples to fit by multiplying T and round them to acquire the integer. Our experiment finds that a properly chosen Beta distribution (tuned on the validation set) makes DNDM perform better on the translation tasks. Specifically, the chosen Beta distributions and the searching method are reported in Appendix F. The performance of the four transition time schedules mentioned above, including the reported Beta distributions for comparison, are listed in Table 5, where we find the other three schedules affect the performance, and most of their scores are lower than the scores of Beta distribution, but their scores are at least still close to the reported Beta distributions, especially for DNDM-k-absorb and DNDM-absorb. The efficiencies (measured by NFE) are also similar to one another.

Additionally, the ablation study on a reasonable range of different Beta distributions with 50 and 1000 sampling steps are shown in Tables 10 and 9, where the BLEU scores and NFE values on the test set of one of the three machine translation datasets, `WMT16`, are shown for demonstration. The range of Beta distributions covers our chosen Beta schedules based on validation sets and a variety of basic Beta distribution shapes. These results show that the different Beta distributions influence the performance, but most of these choices of parameters still achieve results close to the optimal. Since the Beta distributions of the reported results in Tables 2 and 3 are selected using the validation set, they do not always have the highest scores on the test set, but their scores still at least belong to the top tiers according to these tables.

Another view of the transition time. In Algorithm 1, we only need to call the neural network when $t \in \mathcal{T}$, which can significantly speed up the sampling since we reduce the function call. Notice that after we get the $\mathbf{x}_0$ prediction, we only update the $\mathbf{x}_t$ for those tokens at the transition time. However, (7) implies that $\mathbf{x}_t = \mathbf{x}_0$ as long as $\tau > t$. Therefore, instead of only updating the $\mathbf{x}_t$ for those tokens at the transition time, i.e., $\tau = t$, we can also update those tokens with transition time $\tau >= t$. This motivates us to consider a variation presented as Algorithm 3, which keeps almost the same sampling time but will update the tokens several times rather than just once. Since the tokens now get the chance to be corrected over time. The new Algorithm 3 will be more robust than Algorithm 1.

Table 5: The BLEU scores and average number of function evaluations (NFE) values of different distributions of transition time for 1000 sampling steps with batch size 100. The parameters of the Beta distributions in this table are the same as in Tables 2 and 3 and are reported in Appendix F.

Datasets	Schedules	DNDM-multi		DNDM-absorb		DNDM-k-multi		DNDM-k-absorb	
		BLEU	Avg NFE	BLEU	Avg NFE	BLEU	Avg NFE	BLEU	Avg NFE
`IWSLT14`	Cosine	31.72	31.71	32.71	31.21	32.91	31.71	34.50	31.21
	Cosine^2	31.78	31.74	**32.93**	31.21	32.78	31.74	34.53	31.21
	Linear α	31.77	31.82	32.65	31.33	32.83	31.82	34.53	31.33
	Beta (reported)	**31.82**	**30.33**	**32.93**	**31.08**	**33.15**	**30.33**	**34.56**	**31.08**
`WMT14`	Cosine	**25.80**	39.61	**26.54**	39.18	26.63	39.61	27.81	39.18
	Cosine^2	25.52	39.48	26.53	39.18	25.01	39.48	**27.95**	39.18
	Linear α	25.58	39.97	26.33	39.82	25.47	39.97	27.63	39.82
	Beta (reported)	25.71	**38.94**	26.43	**38.76**	**26.88**	**38.94**	27.82	**38.76**
`WMT16`	Cosine	32.71	40.50	33.56	40.45	33.46	40.50	34.37	40.45
	Cosine^2	32.73	40.50	33.51	40.45	33.44	40.50	34.24	40.45
	Linear α	32.85	40.36	33.46	40.36	33.47	40.36	33.88	40.36
	Beta (reported)	**32.86**	**38.46**	**33.60**	**38.27**	**33.79**	**38.45**	**34.38**	**38.27**

Table 6: Comparison of left-to-right and right-to-left transition approaches across different datasets and step counts.

Steps	Direction	IWSLT14	WMT14	WMT16
25	Left-to-right	31.08	24.41	31.67
	Right-to-left	30.54	23.33	31.33
50	Left-to-right	32.87	26.46	33.37
	Right-to-left	32.47	25.18	32.78
1000	Left-to-right	34.45	27.93	34.43
	Right-to-left	34.04	27.02	34.15

Impact of Transition Order. We further evaluate the impact of transition order. Building upon the results in Table 3, we investigate how the model performance will change if the transition time is influenced by the position of the tokens: from left to right and from right to left. In the left-to-right approach, tokens positioned on the left are transitioned to $\mathbf{x}_0$ earlier, and vice versa for the right-to-left approach. Our experiments show that the left-to-right approach consistently outperforms the right-to-left approach across all datasets and step counts, as demonstrated in Table 6.

This result suggests that the order of token transitions significantly influences the model's performance, with earlier transitions of left-side tokens leading to better generation quality.

D Discussion on the Number of Function Evaluations (NFE).

In this section, we discuss the number of function evaluations (NFE) in DNDM. According to (9), the update of a token $\mathbf{x}_{t-1,n}$ occurs solely at its designated transition time. Meanwhile, if step t does not coincide with a transition time for any token, we maintain the sentence from the preceding step unchanged: $\mathbf{x}_{t,1:N} = \mathbf{x}_{t-1,1:N}$. Therefore, our algorithm removes the need of function evaluation for steps outside the set of transition times. Given this structure, our analytical emphasis is on the transition set $\mathcal{T}$ since function evaluations are required only at times t that are members of $\mathcal{T}$. Consequently, the NFE is precisely the cardinality of the transition set, denoted by $|\mathcal{T}|$. In our main paper, we propose a naive upper bound for $|\mathcal{T}|$ as $\min\{N, T\}$, which effectively demonstrates the speed of our method when $T > N$. Next, we demonstrate that DNDM also reduces the NFE when $T < N$, by providing a precise estimation of $|\mathcal{T}|$.

Theorem D.1. *Suppose transition time follows distribution $\mathcal{D}_\tau$, and consider a sequence of length N. Then, the cardinality of the transition set $\mathcal{T} := \{\tau_1, \ldots, \tau_N\}$ satisfies:*

- $1 \leq |\mathcal{T}| \leq \min\{N, T\}$,
- $\mathbb{E}[|\mathcal{T}|] = [1 - C_{T,N,\mathcal{D}_\tau}] \cdot T$, *where $C_{T,N,\mathcal{D}_\tau}$ is a constant in the range $(0, 1)$. Furthermore,*

$$C_{T,N,\mathcal{D}_\tau} = \Big(\sum_{i=1}^{T} (1 - p_i)^N \Big) / T \geq (1 - 1/T)^N,$$

where $p_i = \mathbb{P}(\tau = i)$ for $\tau \sim \mathcal{D}_\tau$, and the equality holds if and only if $\mathcal{D}_\tau$ is a uniform distribution.

Proof. The first statement is straightforward. For completeness, the proof is provided. Since there are only N transition times (possibly repeated): $\tau_1, \ldots, \tau_N$, the distinct transition times must satisfy $|\mathcal{T}| \leq N$. Additionally, since $\mathcal{T} \subseteq \{1, \ldots, T\}$, we also have $|\mathcal{T}| \leq T$.

To prove the second statement, we decompose $\mathcal{T}$ and use the property of expectation. Note that $|\mathcal{T}| = \sum_{i=1}^{T} \mathbb{1}\{i \in \mathcal{T}\}$. Thus,

$$\mathbb{E}[|\mathcal{T}|] = \mathbb{E}\Big[\sum_{i=1}^{T} \mathbb{1}\{i \in \mathcal{T}\} \Big] = \sum_{i=1}^{T} \mathbb{P}(i \in \mathcal{T}). \tag{21}$$

Assuming $\mathbb{P}_{\mathcal{D}_\tau}(\tau = i) = p_i$, and that τ_n are i.i.d. draws from $\mathcal{D}_\tau$, we have

$$\mathbb{P}(i \in \mathcal{T}) = 1 - \mathbb{P}(i \notin \mathcal{T}) = 1 - (1 - p_i)^N. \tag{22}$$

Substituting (22) into (21) yields

$$\mathbb{E}[|\mathcal{T}|] = \sum_{i=1}^{T} \left[1 - (1-p_i)^N\right] = \left[1 - \frac{\sum_{i=1}^{T}(1-p_i)^N}{T}\right] \cdot T = [1 - C_{T,N,\mathcal{D}_\tau}] \cdot T,$$

where $C_{T,N,\mathcal{D}_\tau} = \left(\sum_{i=1}^{T}(1-p_i)^N\right)/T$. An upper bound for $C_{T,N,\mathcal{D}_\tau}$ is given as

$$C_{T,N,\mathcal{D}_\tau} = \left[1 - \frac{\sum_{i=1}^{T}(1-p_i)^N}{T}\right] \cdot T \leq \left[1 - \left(1 - \frac{1}{T}\right)^N\right] \cdot T,$$

where the inequality holds if and only if $p_i = 1/T$ for all $i \in [T]$, i.e., $\mathcal{D}_\tau$ is a uniform distribution. □

Remark D.2. Theorem D.1 suggests that even when $T \leq N$, our method still provides a significant improvement. Specifically, for $T = N \geq 4$, we have $C_{T,N,\mathcal{D}_\tau} = (1 - 1/N)^N \geq 0.3$. This implies that our model requires at most $0.7T$ even in the worst case. Moreover, if we consider a special scenario where the number of p_i satisfying $p_i < \epsilon$ is more than M, then we have $C_{T,N,\mathcal{D}_\tau} > M(1-\epsilon)^N/T$, indicating that with M sufficiently large and ϵ sufficiently small, $C_{T,N,\mathcal{D}_\tau}$ can be pretty close to 1.

Remark D.3. In practical applications of our model, we employ a beta distribution for $\mathcal{D}_\tau$, which typically exhibits a right-heavy tail. Therefore $C_{T,N,\mathcal{D}_\tau}$ tends to be larger than that in the worst-case scenario. In Tables 7 and 8, we list the average NFE for each experiment we run in §4. These results demonstrate a significant reduction in NFE compared to the original counts: for $T = 25$, the NFE is only about half of the original count; for $T = 50$, it is approximately one-third; and for $T = 1000$, it reduces to less than one-twentieth of the original count.

Remark D.4. By Bernoulli's inequality, $(1-p)^N > 1 - N \cdot p$ for $1 > p > 0$. Therefore, $C_{T,N,\mathcal{D}_\tau} > 1 - N/T$, implying that $\mathbb{E}[|\mathcal{T}|] < N$. As $T \to \infty$, assuming the transition time does not concentrate at a single point, the probability that two transitions occur simultaneously is zero. Consequently, the generation process will sequentially go through each token. Thus, the expected number of function evaluations (NFE), $\mathbb{E}[|\mathcal{T}|]$, will be N. In contrast, when T is finite, there is a non-zero probability that multiple transitions happen at the same time. Hence, in this case, the NFE, $|\mathcal{T}|$, is strictly less than N

Table 7: BLEU score and the average number of function evaluations (NFE) comparison of multinomial diffusion on machine translation benchmarks `IWSLT14 DE-EN`, `WMT14 EN-DE`, and `WMT16 EN-RO`. The blue background highlights our algorithms. The average NFE values are calculated by dividing the number of times calling the denoising function (neural network) during generation by the number of batches, where the batch sizes of all experiments are 100.

Dataset	Steps	RDM-Multi		DNDM-Multi		RDM-k-Multi		DNDM-k-Multi	
		BLEU	Avg NFE	BLEU	Avg NFE	BLEU	Avg NFE	BLEU	Avg NFE
IWSLT14	25	**31.26**	25	30.95	**9.03**	**32.82**	25	32.30	**9.03**
	50	**31.50**	50	31.45	**14.07**	**32.82**	50	32.80	**14.07**
	1000	31.69	1000	**31.82**	**30.33**	32.64	1000	**33.15**	**30.33**
	∞	-	-	**31.89**	**32.73**	-	-	**33.44**	**32.73**
WMT14	25	**25.25**	25	25.01	**13.52**	**26.03**	25	25.98	**13.52**
	50	**25.75**	50	25.33	**20.58**	26.14	50	**26.37**	**20.58**
	1000	25.66	1000	**25.71**	**38.94**	25.82	1000	**26.88**	**38.94**
	∞	-	-	**24.79**	**40.67**	-	-	**26.39**	**40.67**
WMT16	25	**32.29**	25	31.97	**8.5**	**33.12**	25	32.94	**8.5**
	50	**32.53**	50	32.50	**14.73**	**33.41**	50	33.26	**14.73**
	1000	32.63	1000	**32.86**	**38.45**	33.67	1000	**33.79**	**38.45**
	∞	-	-	**32.91**	**41.64**	-	-	**33.86**	**41.64**

E Discrete Non-Markov Diffusion Model with Top-k Transition Time (DNDM-K).

Table 8: BLEU score and the average number of function evaluations (NFE) comparison of absorbing diffusion on machine translation benchmarks `IWSLT14 DE-EN`, `WMT14 EN-DE`, and `WMT16 EN-RO`. The blue background highlights our algorithms. The average NFE values are calculated by dividing the number of times calling the denoising function (neural network) during generation by the number of batches, where the batch sizes of all experiments are 100.

Dataset	Steps	RDM-Absorb		DNDM-Absorb		RDM-k-Absorb		DNDM-k-Absorb	
		BLEU	Avg NFE	BLEU	Avg NFE	BLEU	Avg NFE	BLEU	Avg NFE
IWSLT14	25	31.58	25	**32.43**	**13.81**	**34.50**	25	34.14	**13.81**
	50	31.80	50	**32.63**	**19.24**	**34.58**	50	34.34	**19.24**
	1000	31.91	1000	**32.93**	**31.08**	**34.60**	1000	34.56	**31.08**
	∞	-	-	**33.03**	**32.07**	-	-	**34.65**	**32.07**
WMT14	25	24.97	25	**25.79**	**15.09**	**27.50**	25	27.18	**15.09**
	50	24.95	50	**26.10**	**22.45**	**27.73**	50	27.66	**22.45**
	1000	25.22	1000	**26.43**	**38.76**	27.75	1000	**27.82**	**38.76**
	∞	-	-	**26.50**	**40.39**	-	-	**27.50**	**40.39**
WMT16	25	32.86	25	**33.20**	**13.91**	33.92	25	**33.96**	**13.91**
	50	32.93	50	**33.30**	**20.95**	34.10	50	**34.20**	**20.95**
	1000	33.25	1000	**33.60**	**38.27**	**34.44**	1000	34.38	**38.27**
	∞	-	-	**33.42**	**41.59**	-	-	**34.41**	**41.59**

Algorithm 3 Sampling From DNDM (Version 2)

Require: Trained prediction function $p_{\boldsymbol{\theta}}$, $\mathbf{q}_{\text{noise}}$, $\mathcal{D}_\tau$
for $n = 1 \dots N$ **do**
  Initiate each token $\mathbf{x}_{T,n} \sim \mathbf{q}_{\text{noise}}$
  Initiate the transition time $\tau_n \sim \mathcal{D}_\tau$
end for
Collect transition time set $\mathcal{T} = \{\tau_n\}_{n=1}^N$
for $t = T \dots 1$ **do**
  if $t \in \mathcal{T}$ **then**
    Generate $\widetilde{\mathbf{x}}_{0,1:N}$ from $p_{\boldsymbol{\theta}}(\cdot|\mathbf{x}_{t,1:N})$
    for $n = 1 \dots N$ **do**
      Update $\mathbf{x}_{t-1,n}$ if $\tau_n \geq t$
    end for
  else
    Update $\mathbf{x}_{t-1,1:N} = \mathbf{x}_{t,1:N}$
  end if
end for
Return $\mathbf{x}_{0,1:N}$

Algorithm 4 Sampling From DNDM-K

Input: Trained prediction function $p_{\boldsymbol{\theta}}$, $\mathbf{q}_{\text{noise}}$ and $\mathcal{D}_\tau$
for $n = 1 \dots N$ **do**
  Initiate each token $\mathbf{x}_{T,n} \sim \mathbf{q}_{\text{noise}}$
  Initiate the top K number $\{K_t\}$
  Initiate an empty set $U = \{\}$, which includes the index of the tokens that have been updated.
end for
for $t = T \dots 1$ **do**
  if $K_{t-1} > K_t$ **then**
    Calculate the $\mathcal{P} = \text{argtop}_{K_t}\{s_{t,n}\}_{n=1}^N$;
    Generate $\widetilde{\mathbf{x}}_{0,1:N}$ from $p_{\boldsymbol{\theta}}(\cdot|\mathbf{x}_{t,1:N})$
    Update $\mathbf{x}_{t-1,n} = \widetilde{\mathbf{x}}_{0,n}$ for all n in the set $\mathcal{P}$ but not in the set U (top score but not updated yet)
    Update the set U by appending the index of the updated tokens
  else
    Update $\mathbf{x}_{t-1,1:N} = \mathbf{x}_{t,1:N}$;
  end if
end for
Return $\mathbf{x}_{0,1:N}$.

Recent works have demonstrated that the quality of samples can be enhanced by utilizing supplementary information derived from the neural network (Ghazvininejad et al., 2019; Savinov et al., 2021; Chang et al., 2022; He et al., 2022). Very recently, Zheng et al. (2023) applied this idea in their RDM framework and can achieve significant performance improvement. Specifically, after decoding $\widehat{\mathbf{x}}_{0,1:N}$ from transformer $p_\theta(\cdot|\mathbf{x}_{t,1:N})$, the score corresponding to this decoded token from the transformer's last layer, is also recorded and denote as $s_{t,n}$. Tokens with high scores are more likely to be selected for updates.

Inspired by Zheng et al. (2023), we introduce the discrete non-Markov discrete diffusion Model with top-K transition time (DNDM-K). Instead of directly determining which token gets updated at step t by first drawing transition time $\tau \sim \mathcal{D}_\tau$, we employ a two-step process.

1. We first compute $K_t = \sum_{n=1}^{N} \mathbb{1}(\tau_n \geq t)$. k_t represents how many tokens should be decoded at the current step.
2. Compare K_{t-1} and K_t, if $K_{t-1} = K_t$. There is no transition time at time t, we just update $\mathbf{x}_{t-1,1:N} = \mathbf{x}_{t,1:N}$. If $K_{t-1} > K_t$, Then there exist transition time at time t, we calculate and select the indexes with top-K_{t-1} scores. Then we update those tokens if it hasn't been updated yet.

Subsequently, we will only update those tokens with the highest K_t score that hasn't been changed yet. Since the function evaluation occurs only when K_t changes, DNDM-K can give an accelerated sampling algorithm. The details are presented in Algorithm 4.

F Experiment details

F.1 Conditional Text Generation

Parameter choices. In all experiments, the batch size is chosen to be 100. For RDM and RDM-k, our hyperparameter settings follow the original paper (Zheng et al., 2023) except for the batch size. Before the sampling, we used the saved checkpoint of trained models provided by the authors for discrete sampling experiments, and we trained the corresponding models for continuous sampling experiments.

For finite-step DNDM, the transition times are determined by the schedule, and we approximate the schedule with a Beta distribution Beta(α, β) (please refer to Section 3.2 for detailed explanation). The α and β values are selected by applying grid search on the validation sets. Based on the BLEU scores on the validation sets, we have selected Beta$(15, 7)$ for Multinormial Diffusion on `IWSLT14`, Beta$(3, 3)$ for Absorbing Diffusion on both `IWSLT14` and `WMT14`, Beta$(5, 3)$ for Multinormial Diffusion on `WMT14` and Absorbing Diffusion on `WMT16`, and Beta$(20, 7)$ for Multinormial Diffusion on `WMT16`.

For infinite-steps (continuous-step) diffusion (DNDM-C), the transition timestamps are sampled from Beta(α, β), where the choice of (α, β) are chosen from $(100.0, 4.0)$ or $(17.0, 4.0)$, based on the performance comparison on the validation set. In the end we choose Beta$(17, 4)$ for `IWSLT14` and Beta$(100, 4)$ for `WMT14` and `WMT16`.

We conduct a performance comparison based on varying configurations of the Beta and Alpha distributions. The results of these comparisons are presented in Tables 10 and 9. Furthermore, to evaluate the efficacy of discrete versus continuous step schemes, we also conduct an ablation study under the same set of parameters $(100, 4)$ in Table 11.

Table 9: BLEU scores on dataset `WMT16` from the ablation study of other different Beta(α, β) distributions of the transition time with 1000 sampling steps.

Model	Alpha	Beta									
		3	5	7	9	11	13	15	17	19	21
DNDM-k-Multi	3	33.47	33.67	33.62	33.77	**33.87**	33.64	33.73	33.60	33.68	33.56
	5	33.18	33.47	33.68	33.53	33.71	33.69	33.73	33.72	33.74	33.82
	7	32.99	33.20	33.49	33.56	33.58	33.61	33.67	33.72	33.78	33.83
DNDM-Multi	3	32.73	32.66	32.74	32.82	32.77	**32.92**	32.80	32.81	32.76	32.86
	5	32.32	32.62	32.70	32.80	32.83	32.83	32.90	32.95	32.91	32.87
	7	32.35	32.35	32.53	32.67	32.75	32.78	32.86	32.80	32.86	32.88
DNDM-k-Absorb	3	34.19	34.38	34.34	34.22	34.21	34.24	34.07	34.31	**34.42**	34.36
	5	32.15	33.99	34.29	34.30	34.29	34.40	34.40	34.24	34.30	34.22
	7	27.67	32.87	33.94	34.28	34.27	34.38	34.31	34.29	34.38	34.40
DNDM-Absorb	3	33.53	33.60	33.67	33.71	33.71	33.70	33.58	33.63	33.53	33.54
	5	32.70	33.33	33.52	33.60	33.66	33.73	33.70	**33.74**	33.72	**33.74**
	7	30.56	32.65	33.28	33.37	33.51	33.52	33.61	33.67	33.63	33.67

Table 10: BLEU scores on dataset WMT16 from the ablation study of other different Beta(α, β) distributions of the transition time with 50 sampling steps.

Model	Alpha	Beta									
		3	5	7	9	11	13	15	17	19	21
DNDM-k-Multi	3	33.31	33.47	33.39	33.48	33.29	33.23	33.25	33.27	33.11	33.17
	5	32.93	33.28	33.29	**33.58**	33.45	33.21	33.40	33.49	33.16	33.19
	7	32.61	32.98	33.31	33.20	33.27	33.41	33.39	33.53	33.35	33.08
DNDM-Multi	3	32.63	32.46	32.44	32.56	32.59	32.55	32.37	32.33	32.22	32.23
	5	32.31	32.43	32.66	32.64	**32.68**	32.55	32.55	32.44	32.35	32.30
	7	31.95	32.11	32.22	32.26	32.54	32.52	32.50	32.58	32.48	32.41
DNDM-k-Absorb	3	34.05	34.2	34.31	34.37	34.15	34.05	34.06	33.77	33.81	33.84
	5	32.30	34.08	34.30	**34.38**	34.26	34.23	34.09	34.06	34.02	34.13
	7	27.39	32.64	33.71	34.18	34.02	34.33	34.31	34.17	34.12	34.19
DNDM-Absorb	3	33.26	33.30	33.29	33.24	33.23	32.97	33.06	32.85	32.89	32.63
	5	32.47	33.08	33.31	33.22	**33.41**	33.25	33.15	33.27	33.04	32.98
	7	30.34	32.27	33.27	33.03	33.16	33.14	33.27	33.11	33.11	33.07

Table 11: The BLEU scores on dataset WMT16 with Beta(100,4) as the transition time schedule for discrete sampling or the distribution to sample transition timestamps for continuous sampling.

Steps	DNDM-k-multi	DNDM-k-absorb	DNDM-multi	DNDM-absorb
50	31.60	31.74	30.39	29.69
1000	33.59	34.37	32.87	33.52
∞	33.86	34.41	32.91	33.42

Continuous time vs discrete time diffusions. To test our hypothesis that the continuous-time sampler will produce more accurate results in reverse sampling if our $\mathbf{x}_0$ estimator consistently approximates the true $\mathbf{x}_0$ over time, we conduct various sampling experiments using a shared pre-trained neural network. For discrete-time sampling, we consider three cases: $T = 25, 50, 1000$. In each case, we rescale the interval $[0, T]$ to $[0, 50]$ and divide it into T fractions. In contrast, for continuous-time sampling, we directly sample from a continuous distribution over the interval $[0, 50]$ without any partitioning.

Training approach. In machine translation tasks, the neural network is designed to learn $q(\mathbf{x}_0|\mathbf{x}_t, \mathbf{z})$, where $\mathbf{z}$ represents the embedding of the source text obtained using transformer encoder layers. For a fair comparison, we employ the same neural network structure as our baseline, with detailed architecture specifications available in Section E.2 of Zheng et al. (2023). Furthermore, given that the primary focus of this paper is the speed and effectiveness of our sampling algorithm, we omit the training procedure and instead use a state-of-the-art diffusion-based pretrained checkpoint from Zheng et al. (2023). In the Appendix, we present additional results of continuous sampling based on a continuously trained checkpoint. In this setting, we rescale our network input to the interval $[0, 1]$ and uniformly sample from this interval. The rest of the architecture follows that of Zheng et al. (2023).

Performance on WMT14. Our work primarily focuses on the sampling process, and for the training, we utilized a pretrained checkpoint trained on 50 steps. In our sampling experiments we noticed that our method does not work ideally on WMT14, this could be possibly attributed to the fact that the training performance on WMT14 was not ideal. Specifically, when we performed sampling using 1000 steps, the network was trained with exposure to only 50 time steps, specifically at intervals of 20 (0, 20, 40, ..., 980, 1000). As a result, when we apply our model to generation using 1000 steps, the checkpoint NN has only been explicitly trained on these intervals. While we generally assume that the network can still provide a good estimate for the untrained steps, this might not hold under some hard scenarios. Considering the longer training time and poorer performance of WMT14, it is likely that the training performance is insufficient for us to rely on those unseen steps. In a word, the model's trained checkpoint may not be robust enough to effectively handle unseen steps, especially for timesteps 1000 or infinite timesteps.

F.2 Unconditional Text Generation

Parameter choices. We recover the checkpoints of the multinomial diffusion model employing the provided code by Hoogeboom et al. (2021b). We train 12-layer Transformers for both `text8` and enwik8 datasets for 500 epochs with the cosine schedule. For the `text8` dataset, we utilize a training batch size of 256, while for the `enwik8` dataset, we use a batch size of 128. During training, we employ a learning rate of 0.0001, a weight decay parameter of 0.99, and the Adam optimizer.

G Additional Experiments

In this section, we present additional experimental results. We begin by plotting the relationship between computational time and the number of sampling steps, using the absorbing diffusion in `IWSLT14` as an example. Figure 4 displays the growth of computational time for absorbing diffusion (yellow and orange lines), RDM-absorbing diffusion, and our model DNDM-Absorb and DNDM-T-Absorb (green and blue lines). We see from Figure 4 that previous algorithms, including absorbing

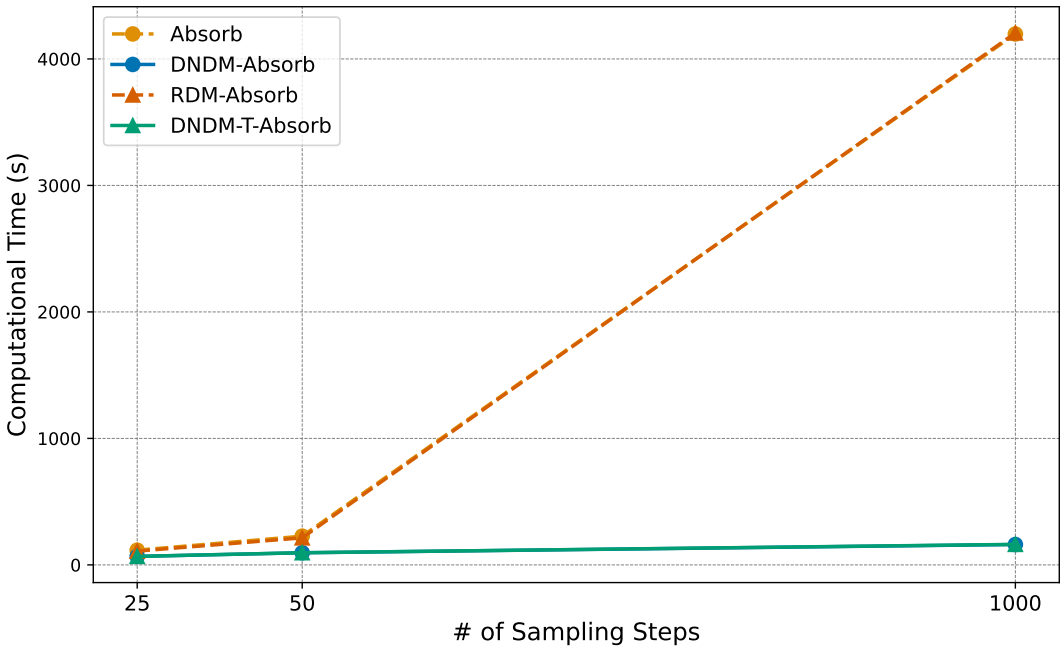


Figure 4: The growth of computational time with the increase of the sampling steps

diffusion and RDM-absorbing diffusion all suffer from linear growth of computational time.

G.1 Continuous Training

In Section 4.1, we introduce the DNDM-C algorithm, designed for continuous-time, over discrete-time algorithms. However, this algorithm assumes that we have learned a sufficiently accurate neural network at any timestamp $t \in [0, 1]$. Using the checkpoint trained with 50 discrete time partitions might not suffice for the purpose of continuous sampling. In this section, we investigate the performance of continuous sampling when training is also done continuously.

Table 12: Continuous Training + Continuous Sampling

Dataset	Step scheme	C-DNDM-Multi		C-DNDM-Absorb	
		Default	Top-k	Default	Top-k
`IWSLT14`	Continuous	**32.07**	**33.57**	32.80	34.52
`WMT16`	Continuous	**33.48**	33.71	**33.50**	34.36

In Table 12, we summarize the performance of DNDM-C based on a neural network estimated continuously during training time. This involves sampling time uniformly from $[0, 1]$ during training, and the forward process follows (11) in Section 3.3. The training objective remains the same as in discrete-time training. In Table 12 we list the result of IWSLT14 and WMT16 with continuous training followed by continuous sampling. In addition, we compare the value with the corresponding value during discrete training and continuous sampling in Section 4.1 and mark every item that improves in bold. As demonstrated in Table 12, there is room for enhancement in the overall sampling scores by training the neural network in a complete space of timestamps.

G.2 Comparison with more generative models

In our study, a key aspect of evaluating our fast discrete generative model involves comparisons with prior work known for speed in sampling with minimal steps. Specifically, we draw a direct comparison with the Mask-Predict (Ghazvininejad et al., 2019), which is notable for its ability to generate high-quality results within just 10 iterations. The results are shown in Table 13. All experiments were conducted on the same GPU and within the same machine setup.

Table 13: The performance comparison on `WMT16` of DNDM with Mask-Predict (Ghazvininejad et al., 2019). We align the number of sampling steps used in Mask-Predict with a similar number of function evaluations (NFE) in our DNDM algorithm. We see that our Algorithm runs faster, with better BLEU score.

Mask-Predict			DNDM-Absorb				DNDM-k-Absorb			
Steps	**BLEU**	**Time**	**Steps**	**BLEU**	**Time**	**NFE**	**Steps**	**BLEU**	**Time**	**NFE**
10	33.08	49.25	25	33.20	41.2	13.91	25	33.96	41.4	13.91
15	33.06	67.94	50	33.30	62.5	20.95	50	34.20	62.7	20.95
25	33.16	111.89	1000	33.60	121.3	38.27	1000	34.38	122.7	38.27
40	33.10	169.95	∞	33.42	121.8	41.59	∞	34.41	121.9	41.59

G.3 Samples from the multinomial text models

Conditional Generation. For DNDM-Multi trained on `IWSLT14`, we provide a full generation process with 100 steps in Figure 5. A token ending with `@@` indicates it is an incomplete word; it will be concatenated with the following token to form a complete word. For example, "`fel@@ lo@@ ws`" means "`fellows`". We can see that after $t = 39$, the generate sentence converges.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The contributions are summarized as three points at the end of the introduction. The scope is fast sampling via discrete non-Markov diffusion models, provided in the abstract.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We add a limitation section in front of the Appendix.

   Guidelines:

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

3. **Theory Assumptions and Proofs**

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

Answer: [Yes]

Justification: Theorems 3.1, 3.5, and D.1 are clearly stated, well-organized with consistent numbering, and supported by rigorous proofs that establish their validity.

Guidelines:

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

4. **Experimental Result Reproducibility**

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

Answer: [Yes]

Justification: The paper provides detailed information on the experimental setup, model architecture, and training procedures. The authors have submitted their training code along with the main paper, which enables reproducibility of the main results. The code and detailed instructions allow other researchers to replicate the key findings of the paper.

Guidelines: In addition to experiment and implementation details on appendix, we submit our training and evaluation codes when submtting our main paper.

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

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: All the datasets are public and can be open accessed. Our codebase will be available in public upon acceptance.

   Guidelines:

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

6. **Experimental Setting/Details**

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

   Answer: [Yes]

   Justification: We provide these details on Appendix (D, E, F).

   Guidelines:

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

7. **Experiment Statistical Significance**

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

   Answer: [Yes]

   Justification: Confidence intervals are provided in the experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

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

8. **Experiments Compute Resources**

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

   Answer: [Yes]

   Justification: We provided detailed information about the computation resources in Section 4: a single NVIDIA258 RTX A6000 GPU with 48 GB memory.

   Guidelines:

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

9. **Code Of Ethics**

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

   Answer: [Yes]

   Justification: We have checked NeurIPS Code of Ethics. Our submission satisfies all the requirement.

   Guidelines:

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

10. **Broader Impacts**

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

    Answer: [Yes]

    Justification: We provide Broader Impacts Section in the beginning of Appendix.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

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

11. **Safeguards**

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

    Answer: [NA]

    Justification: The paper poses no related risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All used code, data and models in this project are properly cited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

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

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

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

14. **Crowdsourcing and Research with Human Subjects**

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

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

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

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

t = 100
[noise] [noise] [noise] [noise] [noise] [noise] [noise] [noise] [noise] [noise] [noise] [noise] [noise] [noise] [noise] [noise] [noise] [noise] [noise] [noise]

t = 79
[noise] [noise] [noise] [noise] [noise] [noise] [noise] [noise] [noise] [noise] [noise] [noise] [noise] [noise] [noise] [noise] [noise] [noise] year **[noise]**

t = 78
[noise] [noise] [noise] [noise] [noise] [noise] [noise] [noise] [noise] we **[noise] [noise] [noise] [noise] [noise] [noise] [noise] [noise] [noise]** year **[noise]**

t = 77
[noise] [noise] [noise] [noise] [noise] [noise] [noise] [noise] and we **[noise] [noise] [noise] [noise] [noise] [noise] [noise] [noise] [noise]** year **[noise]**

t = 75
[noise] [noise] [noise] [noise] [noise] [noise] [noise] [noise] and we **[noise] [noise] [noise] [noise]** govern@@ **[noise] [noise]** year **[noise]**

t = 74
we **[noise] [noise] [noise]** lo@@ **[noise] [noise] [noise]** and we **[noise] [noise] [noise] [noise]** govern@@ **[noise] [noise]** year **[noise]**

t = 73
we **[noise] [noise]** fel@@ lo@@ **[noise] [noise] [noise]** and we let **[noise] [noise] [noise] [noise]** govern@@ **[noise] [noise]** year **[noise]**

t = 71
we **[noise] [noise]** fel@@ lo@@ **[noise] [noise] [noise]** and we let **[noise] [noise] [noise] [noise]** govern@@ **[noise]** every year **[noise]**

t = 67
we **[noise] [noise]** fel@@ lo@@ **[noise] [noise] [noise]** and we let them **[noise] [noise]** city govern@@ **[noise]** every year .

t = 66
we **[noise] [noise]** fel@@ lo@@ ws **[noise] [noise]** and we let them work **[noise]** city govern@@ **[noise]** every year .

t = 64
we **[noise] [noise]** fel@@ lo@@ ws **[noise] [noise]** and we let them work **[noise]** city govern@@ ance every year .

t = 61
we **[noise] [noise]** fel@@ lo@@ ws **[noise] [noise]** and we let them work with city govern@@ ance every year .

t = 60
we **[noise] [noise]** fel@@ lo@@ ws **[noise]** year and we let them work with city govern@@ ance every year .

t = 58
we **[noise] [noise]** fel@@ lo@@ ws every year and we let them work with city govern@@ ance every year .

t = 52
we **[noise]** some fel@@ lo@@ ws every year and we let them work with city govern@@ ance every year .

t = 39
we choose some fel@@ lo@@ ws every year and we let them work with city governance every year.

t = 0
we choose some fel@@ lo@@ ws every year and we let them work with city governance every year.

Figure 5: Text in the Generation Process