Analysis on Non-Gaussian Distributions in Diffusion

Tianwen Fu
tianwenf@andrew.cmu.edu

Xuchen Gong
xuchengo@andrew.cmu.edu

Abstract

Diffusion models are prevalent recently, but the vast majority adopt Gaussian distribution to model the conditional probabilities; on the other hand, studies on discrete-state diffusion have introduced some alternative transitions, but lack comprehensive comparisons. In this project, we propose a synthetic dataset that allows fast training and straightforward visualization and conduct contrastive experiments over the dataset. We also introduce randomly generated doubly stochastic matrices as an alternative to the previous transitions. Experiment results show that the learned distributions tend to be peakier, the absorbing transition is robust to the absorbing state, and the cosine schedule is empirically superior to other candidates. Code is available at `https://github.com/tianwen-fu/discrete-diffusion-distributions`.

1 Introduction

Recent years have witnessed the boost of generative models for image synthesis [3, 10, 15, 18], natural language modeling [5], and offline reinforcement learning [19]. Prevalent approaches to approximating the distribution of data include generative adversarial networks (GANs) [7], variational auto-encoders [11, 14], denoising diffusion probabilistic models [8], normalizing flows [13], and score matching [17].

Diffusion models have attracted particular attention with the superior performance to GANs [6] and the ability to model the likelihoods explicitly. Given a sample drawn from the data distribution, the forward diffusion process is a Markov chain that gradually corrupts the sample into noise, and a model is fit to the reverse process that recovers data distribution from the noise distribution.

Although the majority of continuous-space methods [8] model forward and reverse processes as Gaussian distributions, it has not been proven yet to be the optimal distribution to the best of our knowledge; in fact, the line of works on normalizing flows [13] is built on the lack of flexibility of Gaussian to capture all distributions, and the extension to non-Gaussian distribution may further benefit the use cases for diffusion-based adversarial denoisers [4]. Besides, categorical data constitute a considerable portion of features utilized in machine learning, the most prevailing of which is text data. Natural language generation has gained particular interest following the success of BERT [5]. However, diffusion models with Gaussian distribution inherently rely on the assumption of a continuous neighborhood, which is not suitable for categorical data without ordinal structure.

Recently, researchers have made progress in extending diffusion models to discrete state spaces. Related literature has explored some alternate transitions such as uniform sampling [9] or converging to an absorbing state [1]. Nevertheless, neither of the works includes analyses of convergence and behavior for general transition matrices, experiments on alternate noise schedules, and variants of prediction targets.

In this project, we aim to gain insights into discrete-space diffusion processes by conducting more comprehensive experiments on noise schedules, transition probabilities, and other design choices such as the prediction targets. To this end, it is favorable to construct a synthetic dataset that provides

Preprint. Under review.

us with access to the ground truth distribution and enables us to efficiently and intuitively interpret the intermediate results.

In short, our goals in the project are:

- to propose characteristic synthetic data for an easier understanding of behaviors in discrete-state-space diffusion;
- to study the nature of general transition matrices, including their limiting distributions and diffusion model training;
- to explore alternative noise schedule and their effects on model convergence;

2 Background

2.1 Denoising diffusion for continuous data

Diffusion models are proposed in Sohl-Dickstein et al. [16] and Ho et al. [8]. The forward process is a Markov chain $\{\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(T)}\}$ that gradually add noise to uncorrupted $\mathbf{x}^{(0)}$ according to a variance schedule $\{\beta_1, \beta_2, \cdots, \beta_T\}$. The probability is defined as

$$q(\mathbf{x}^{(1:T)}|\mathbf{x}^{(0)}) \coloneqq \prod_{t=1}^{T} q(\mathbf{x}^{(t)}|\mathbf{x}^{(t-1)}), \quad q(\mathbf{x}^{(t)}|\mathbf{x}^{(t-1)}) \coloneqq \mathcal{N}(\mathbf{x}^{(t)}; \sqrt{1-\beta_t}\mathbf{x}^{(t-1)}, \beta_t \mathbf{I}) \tag{1}$$

The reverse process to recover $\mathbf{x}^{(0)}$ parametrized by θ is

$$p_\theta(\mathbf{x}^{(0:T)}) \coloneqq p(\mathbf{x}^{(T)}) \prod_{t=1}^{T} p_\theta(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}) \tag{2}$$

where

$$p(\mathbf{x}^{(T)}) = \mathcal{N}(\mathbf{x}^{(T)}; \mathbf{0}, \mathbf{I}) \tag{3}$$

$$p_\theta(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}) \coloneqq \mathcal{N}(\mathbf{x}^{(t-1)}; \mu_\theta(x_t, t), \Sigma_\theta(\mathbf{x}^{(t)}, t)) \tag{4}$$

The loss function is given by the evidence lower bound of negative log-likelihood

$$\mathbb{E}[-\log p_\theta(\mathbf{x}^{(0)})] \leq \mathbb{E}_q \left[-\log \frac{p_\theta(\mathbf{x}^{(0:T)}}{q(\mathbf{x}^{(1:T)}|\mathbf{x}^{(0)})} \right] =: L \tag{5}$$

which is equivalent to [16]:

$$L = \mathbb{E}_q \left[L_T + \sum_{t>1} L_{t-1} + L_0 \right] \tag{6}$$

where

$$L_T = D_{KL} \left(q(\mathbf{x}^{(T)}|\mathbf{x}^{(0)}) \,\|\, p(\mathbf{x}^{(T)}) \right) \tag{7}$$

$$L_{t-1} = D_{KL} \left(q(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}, \mathbf{x}^{(0)}) \,\|\, p_\theta(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}) \right) \tag{8}$$

$$L_0 = -\log p_\theta(\mathbf{x}^{(0)}|\mathbf{x}^{(1)}) \tag{9}$$

In practice, Ho et al. [8] utilize $\mathbb{E}_t[L_t]$ up to a constant. This fact and Equation 6 suggest that we shall have tractable $q(\mathbf{x}^{(t)}|\mathbf{x}^{(0)})$ and $q(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}, \mathbf{x}^{(0)})$, and the KL divergence of p_θ and q shall be easy to compute and optimize.

2.2 Diffusion models for discrete state spaces

Hoogeboom et al. [9] proposes Multinomial Diffusion for categorical data that gradually corrupts $\mathbf{x}^{(0)}$ into uniform distributions, and Austin et al. [1] further extend to more general transitions: for categorical one-hot random variables $\mathbf{x}^{(t)} \in \mathbb{R}^K$ such that $\mathbf{x}_k^{(t)} = 1$ and $\mathbf{x}_j^{(t)} = 0$ for any $j \neq k$, define forward transition matrix $\mathbf{Q}_{ij}^{(t)} = q(\mathbf{x}^{(t)} = \mathbb{1}_j | \mathbf{x}^{(t-1)} = \mathbb{1}_i)$. Then we may write

$$q(\mathbf{x}^{(t)}|\mathbf{x}^{(t-1)}) = \mathrm{Cat}(\mathbf{x}^{(t)}; p = \mathbf{x}^{(t-1)}\mathbf{Q}^{(t)}) \tag{10}$$

For the reverse process, instead of predicting $p_\theta(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)})$, both works use a neural network to predict $\tilde{p_\theta}(\tilde{\mathbf{x}}^{(0)}|\mathbf{x}^{(t)})$ and then parametrize the reverse process as

$$p_\theta(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}) \propto \sum_{\tilde{\mathbf{x}}^{(0)}} q(\mathbf{x}^{(t-1)}, \mathbf{x}^{(t)}|\tilde{\mathbf{x}}^{(0)}) \tilde{p}_\theta(\tilde{\mathbf{x}}^{(0)}|\mathbf{x}^{(t)}) \tag{11}$$

Apart from the variational objective in Equation 6, Austin et al. [1] also propose to add explicit supervision on the model prediction via a cross-entropy term

$$L_{CE} = \mathbb{E}_{q(\mathbf{x}^{(0)})} \mathbb{E}_{q\left(\mathbf{x}^{(t)}|\mathbf{x}^{(0)}\right)} \left[-\log \tilde{p}_\theta(\tilde{\mathbf{x}}^{(0)}|\mathbf{x}^{(t)}) \right] \tag{12}$$

It is non-trivial to extend the loss to the case where we predict $p_\theta(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)})$ instead of $p_\theta(\mathbf{x}^{(0)}|\mathbf{x}^{(t)})$.

3 Synthetic Dataset

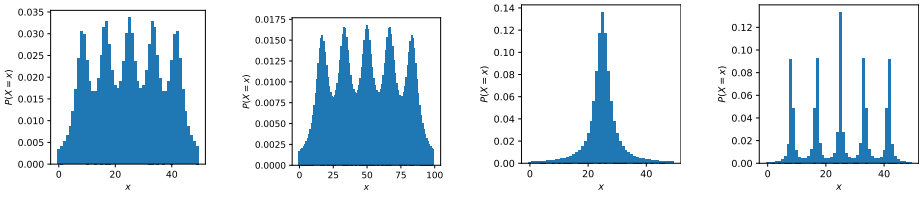
Currently, to characterize the effect of different hyperparameters in diffusion and reverse process, we consider the following factors when building the synthetic dataset on $\mathbf{x} \in \{1, 2, \cdots, N\}$:

- Dimensionality: the number of classes;
- Modality: the number of concentrated centers;
- Smoothness: the temperature of the distribution.

We employ a superposition of Cauchy distribution densities, as it shows more peakiness than the standard normal distribution. For modes $\{c_1, c_2, \cdots, c_M\}$ and corresponding temperatures $\{\tau_1, \tau_2, \cdots, \tau_M\}$, a distribution over N categories is given by

$$P(X = x; \{c_i\}, \{\tau_i\}, N) \propto \sum_{i=1}^{M} \frac{1}{1 + \tau_i^{-2} \left(\frac{x - c_i}{N}\right)^2} \tag{13}$$

The distribution is evaluated at integer values $\{1, 2, \cdots, N\}$ and then divided by the sum for normalization. Figure 1 shows the probability mass functions with the varying three factors. We take $c_i = \frac{iN}{M+1}$ and $\tau_i = \tau$.



(a) Reference (b) $N = 100$ (c) $M = 1$ (d) $\tau = 0.01$

Figure 1: Bar Plots of Probability Mass Functions of the Synthetic Datasets. The parameters in 1a is $N = 50, M = 5, \tau = 0.05$. Figures 1b, 1c and 1d show the distribution with different choices of N, M, τ respectively.

4 Simplified Model

Both previous methods utilize U-Net for predicting p_θ for distributions over 2D images with support $S^{H\times W}$, $S = \{\text{possible values for each pixel}\}$. Since our support is much smaller ($[N] = \{1, 2, \cdots, N\}$), it is desirable to employ a simplified model to accelerate training and avoid overfitting. Aside from simplicity, we also ensure that our model resembles the real-world ones to improve generalizability.

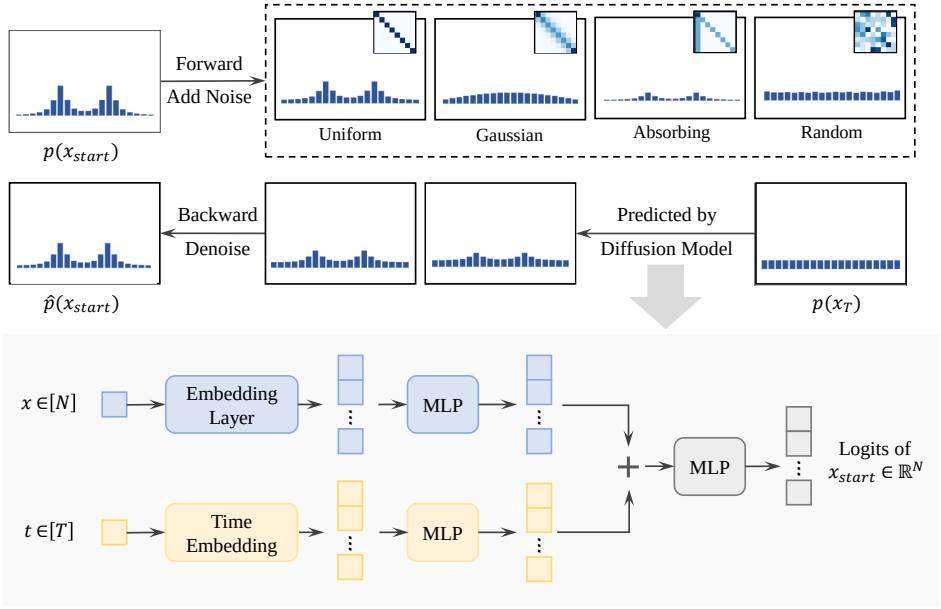


Figure 2: Illustration of our Discrete Diffusion Workflow and our Simplified Model. The upper subfigure visualizes four transition matrices and their corresponding diffused distributions in the intermediate time steps. In the lower subfigure, both the noised sample x and the time t are processed through embedding layers and MLPs before being added together. The result will then go through several linear layers to predict the logits.

Figure 2 shows our dynamics model $y = f(x, t)$ that predicts logits $y \in \mathbb{R}^N$ for x_{start} with noised $x \in [N]$ at time t. We first map each class of the noised data x to learnable embeddings in $\mathbb{R}^{d_x}$ and then obtain data-domain features through m linear layers of output dimensions $\{d_m^{(1)}, d_m^{(2)}, \cdots, d_m^{(m)}\}$; for the time data t, we map it with sinusoidals of multiple frequencies to $\mathbb{R}^{d_t}$, and then through MLP with n latent layers of dimensions $\{d_n^{(1)}, d_n^{(2)}, \cdots, d_n^{(n)}\}$. Afterward, the time features and data features are added together and transformed by the final MLP with dimensions $\{d_p^{(1)}, d_p^{(2)}, \cdots, d_p^{(p)}\}$ to obtain the logits y for x_{start}.

For our experiments, we set $m = n = 2, p = 1, d_x = d_t = d_m = d_p = 64, d_n^{(1)} = 256, d_n^{(2)} = 64$. Following the practice of Austin et al. [1], we use Swish activation [12] for time-related layers and ReLU for all other layers.

5 Transition Matrices

Transition matrices describe the forward noising Markov chain transitions $q(x^{(t)}|x^{(t-1)})$. In the framework of discrete diffusion, we typically require two features of a family of transition matrices:

- The noise induced by the transition matrix can be parametrized with β. We typically would increase the strength and amount of the noise added in later stages of the diffusion process, which is parametrized by increasing values of β.
- The transition matrix, in the limit to infinite power, converges to a known limiting distribution. This would be used in the sampling from p_θ.

In this project, we experiment with uniform, Gaussian, and absorbing transitions described in Austin et al. [1]. Besides, we also explore generating random doubly stochastic matrices for experiments.

5.1 Banded Uniform Transition

A banded uniform transition matrix with noise level β and band B uniformly transits the current state to neighboring states in the band, defined as

$$Q_{ij} = \begin{cases} \frac{\beta}{N} & \text{if } |i-j| \leq B, i \neq j \\ 1 - \sum_{k \neq j} Q_{ik} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

The converging distribution is the uniform distribution.

5.2 Discretized Gaussian Transition

This matrix describes a discretized version of the Gaussian distribution with noise level β and band B. The converging distribution is also the uniform distribution.

$$Q_{ij} = \begin{cases} \text{softmax}\left(\frac{(i-j)^2}{\beta}\right) & \text{if } |i-j| \leq B, i \neq j \\ 1 - \sum_{k \neq j} Q_{ik} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

5.3 Absorbing Transition

Another option is to construct a transition matrix that gradually converges to a single state, S_T. The matrix is

$$Q_{ij} = \begin{cases} \beta & \text{if } i \neq S_T, j = S_T \\ 1 & \text{if } i = j = S_T \\ 1 - \beta & \text{if } i = j \neq S_T \\ 0 & \text{otherwise} \end{cases}$$

5.4 Randomly Generated Doubly Stochastic Matrix

Austin et al. [1] showed in the appendix that irreducible Markov chains with doubly stochastic matrices converge to uniform distributions. By Birkhoff-von Neumann Theorem, the set of doubly stochastic matrices is the convex hull of all permutation matrices. We thus propose Algorithm 1 for generating a random permutation matrix(a JAX [2] version of the code is attached to Appendix A):

Algorithm 1: Randomly Generating a Doubly Stochastic Matrix

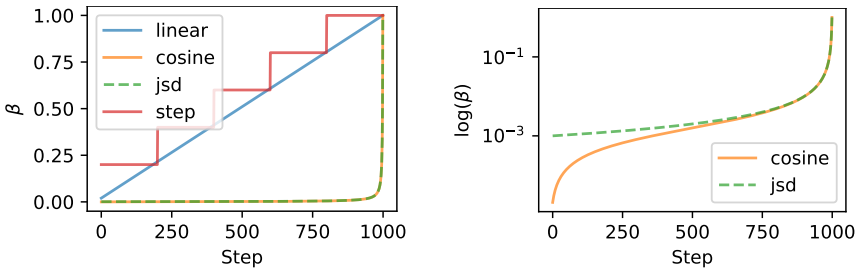
Input: $D \in \mathbb{N}, T \in \mathbb{N}$
Result: $M \in \mathbb{R}^{D \times D}$ such that M is doubly stochastic
found $\leftarrow$ False ;
while *not found* **do**
  $M \leftarrow 0$;
  $c \leftarrow$ random vector in $\mathbb{R}_+^D$;
  Normalize c so that $\sum_{i=1}^{D} c_i = 1$;
  for $i \leftarrow 1$ **to** D **do**
    $P \leftarrow$ a random permutation matrix in $\mathbb{R}^{D \times D}$;
    $M \leftarrow M + P$;
  end
end

Since M converge to uniform distribution, i.e. $\lim_{t \to \infty} M^t = \frac{1}{D}\mathbf{1}\mathbf{1}^T$, we define the matrix with noise factor β as $M(\beta) = M^\beta$.

6 Noise Schedules

Austin et al. [1] utilizes three different types of noise schedules corresponding to each of the transition matrices. In addition, since the randomly generated doubly stochastic transition scheme we proposed in Section 5.4 demands integer noise factors, a discrete-valued schedule, named *step schedule*, is also included in the list below.

- Linear Schedule with Starting Beta β_0 and Final Beta β_T: $\beta_t = \beta_0 + \frac{t}{T}(\beta_T - \beta_t)$, used with uniform transitions;
- Cosine Schedule: $\beta_t = \cos\left(\frac{\pi}{2}\frac{t}{T}\right)$, used with Gaussian transitions;
- JSD Schedule: $\beta_t = \frac{T}{T-t+1}$, used with absorbing transitions.
- Step Schedule with Step Values $\alpha[i]$ with $i \in \{0, 1, \cdots, K-1\}$: $\beta_t = \alpha\left[\lfloor \frac{tK}{T} \rfloor\right]$



(a) Noise Schedules in Linear Scale

(b) Noise Schedules in Log Scale

Figure 3: Illustration of Noise Schedules. Since the cosine schedule and JSD schedule are difficult to distinguish in (a), we plot them in log scale in (b).

In this project, we would test the robustness of interchanging schedules as well. Figure 3 shows an illustration of the noise schedules.

7 Results

7.1 Overpeakiness of Learned Distributions

We observe that the learned distributions tend to be peakier than the ground truth distributions, regardless of the number of modes. Figure 4 shows our results. For smoother distributions where τ is larger, the learned distributions tend to overestimate the peak density, regardless of the transition methods we choose. Especially for multimodal distributions (the third and fourth row), the models tend to focus on one of the modes and predict a much higher density. We also report our results in Table 1, where the negative log-likelihood (NLL) is computed by evaluating the CE Loss defined in Equation 12 over the entire dataset, and $D(p_d \| p_s)$ is the KL divergence of the ground truth data distribution p_d and the model sample distribution p_s. All metrics are better if lower.

7.2 Changing the Absorbing State

Austin et al. [1] set the absorbing state to 128, which is the middle point of the possible pixel value range $\{0, 1, \cdots, 255\}$. We study the impact of choosing an absorption state either collides with a mode of the ground truth distribution or 0.

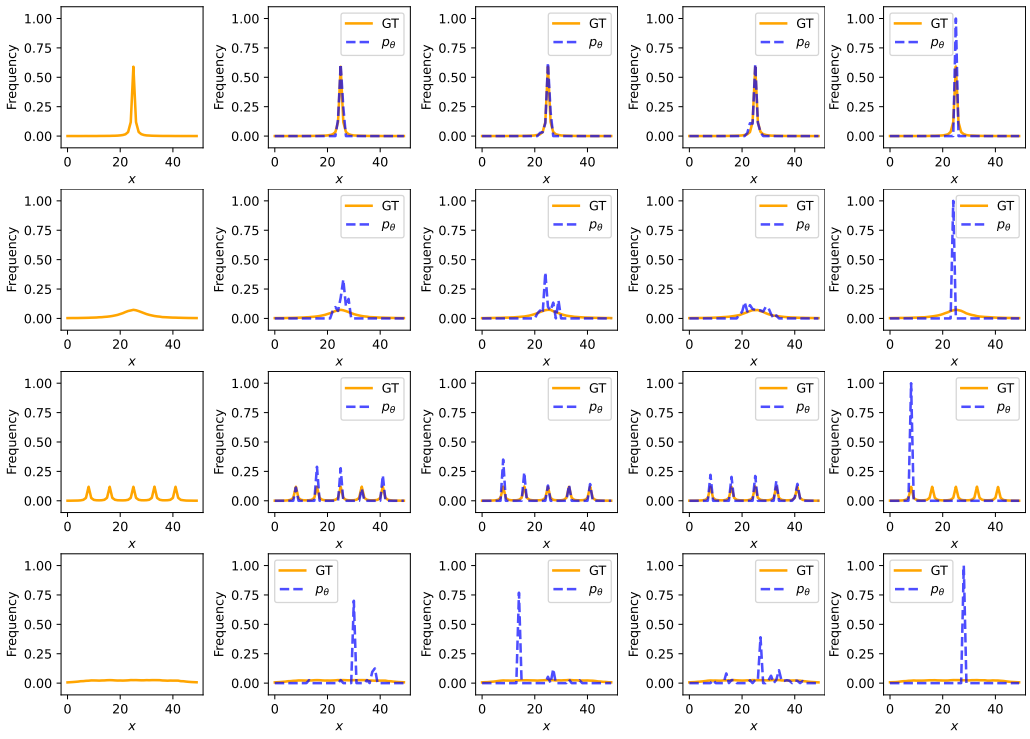
Table 2 and Figure 5 show that the performance of absorbing transition is not significantly impacted by shifting the absorbing states.

7.3 Exploring Noise Schedules

In previous work [1], linear schedules are bundled with uniform transitions while the Gaussian transitions use cosine schedules for noise. However, we want to test how robust the diffusion processes are regarding noise schedules.

Data Distribution	Transition	NLL↓	$D(p_d\|\|p_s)$↓	$D(p_s\|\|p_d)$↓
$M = 1, \tau = 0.01$	Uniform	2.332	2.120	0.326
	Gaussian	2.221	1.429	0.221
	Absorbing	**1.495**	**0.984**	**0.210**
	Random	2.320	5.828	0.755
$M = 1, \tau = 0.1$	Uniform	4.995	7.449	1.381
	Gaussian	4.827	6.922	1.423
	Absorbing	**3.818**	**3.891**	**0.652**
	Random	4.989	13.59	3.860
$M = 5, \tau = 0.01$	Uniform	4.365	5.145	0.860
	Gaussian	3.965	5.107	0.855
	Absorbing	**3.131**	**4.402**	**0.650**
	Random	4.344	13.23	3.071
$M = 5, \tau = 0.1$	Uniform	5.584	12.08	3.893
	Gaussian	5.508	12.35	4.203
	Absorbing	**4.911**	**7.648**	**2.029**
	Random	5.586	13.87	5.355

Table 1: Quantitative Results of the Transition Matrices over Different Distributions. The best results for each type of data distribution are marked in **bold**.

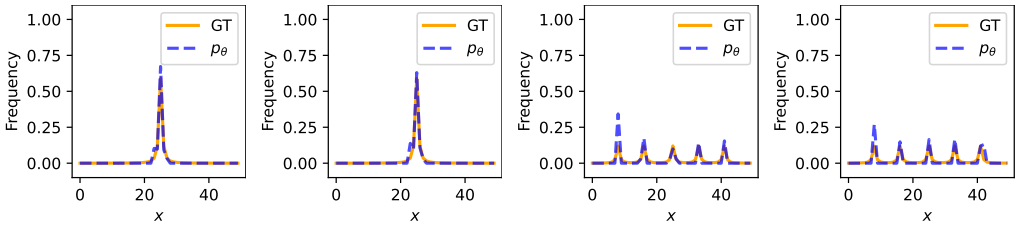


(a) Ground Truth (b) Uniform (c) Gaussian (d) Absorbing (e) Random

Figure 4: Overpeakiness of Learned Distributions. The orange lines are the frequency of the dataset, and the dashed blue lines are the learned distribution frequency curve. Figures in the first row are unimodal data with $M = 1, \tau = 0.01$, and the second row $\tau = 0.1$; the third and fourth rows are data with $M = 5$ and $\tau = 0.01, 0.1$ respectively. The first column consists of the ground truth distribution only, and the following columns also display the learned distributions with uniform transition, Gaussian transition, absorbing transition, and random doubly stochastic matrices respectively.

Data Distribution	Absorbing State	NLL↓	$D(p_d\|\|p_s)$↓	$D(p_s\|\|p_d)$↓
$M = 1, N = 50$	Aligned	1.635	1.174	0.227
	Misaligned	1.506	1.217	0.280
$M = 5, N = 50$	Aligned	3.277	4.112	0.699
	Misaligned	3.130	4.819	0.832

Table 2: Results of Training a Diffusion Model with Absorbing Transition with Different Absorption States. "Aligned" denotes that the absorbing state collides with a mode, and "Misaligned" means that the absorbing state is 0.



(a) $M = 1$, aligned (b) $M = 1$, misaligned (c) $M = 5$, aligned (d) $M = 5$, misaligned

Figure 5: Learned Distribution with Different Choices of the Absorbing Center.

Table 3 shows that the performance can vary largely with different types of noise schedules, but remain robust within a range of the hyperparameters of a schedule. The results also disagree with the pairing of previous work since the cosine schedule outperforms others.

Transition	Schedule	NLL↓	$D(p_d\|\|p_s)$↓	$D(p_s\|\|p_d)$↓
Gaussian	Step(0.02, 0.01, 5)	2.341	5.863	0.761
	Step(0.02, 0.1, 5)	2.342	5.913	0.769
	Step(0.02, 1.0, 5)	2.323	5.847	0.759
	Linear(0.02, 0.01)	2.300	1.639	0.275
	Linear(0.02, 0.1)	2.367	1.656	0.278
	Linear(0.02, 1.0)	2.357	2.265	0.502
	Cosine	1.536	0.991	0.240
	JSD	2.350	5.838	0.757
Uniform	Step(0.02, 0.01, 5)	2.327	1.666	0.240
	Step(0.02, 0.1, 5)	2.323	1.635	0.236
	Step(0.02, 1.0, 5)	2.312	2.173	0.379
	Linear(0.02, 0.01)	2.311	1.668	0.291
	Linear(0.02, 0.1)	2.329	1.183	0.229
	Linear(0.02, 1.0)	2.331	1.673	0.283
	Cosine	1.549	1.004	0.237
	Jsd	1.767	1.247	0.340

Table 3: Results Trained on $M = 1, N = 50, \tau = 0.01$ Dataset with Different Noise Schedules.

8 Division of Work

Both students have been involved evenly in the process of discussion and literature survey, and the production of this report.

References

[1] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021.

[2] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[3] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[4] Nicholas Carlini, Florian Tramer, J Zico Kolter, et al. (certified!!) adversarial robustness for free! *arXiv preprint arXiv:2206.10550*, 2022.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[6] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[9] Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in Neural Information Processing Systems*, 34:12454–12465, 2021.

[10] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[11] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[12] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[13] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

[14] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

[15] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[16] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[17] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

[18] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018.

[19] Wenxuan Zhou, Sujay Bajracharya, and David Held. Plas: Latent action space for offline reinforcement learning. In *Conference on Robot Learning*, pages 1719–1735. PMLR, 2021.

A JAX Code for Generating Random Doubly Stochastic Matrix

```
def generate_doubly_stochastic(target_size, rng):
    multiplies_to_validate_connected = 1000
    while True:
        # generate a doubly stochastic matrix
        matrix = jnp.zeros((target_size, target_size))
        rng, c_rng = jrnd.split(rng)
        c = jrnd.uniform(c_rng, shape=(target_size,))
        c /= c.sum()
        I = jnp.eye(target_size, dtype=jnp.float64)
        for i in range(target_size):
            rng, iter_rng = jrnd.split(rng)
            idx = jrnd.permutation(iter_rng, target_size)
            P = I[idx, :]
            matrix = matrix + c[i] * P

        # check if this is connected,
        # so that we can make sure the stationary distribution
        # is a uniform distribution
        mat_prod = matrix
        for _ in range(multiplies_to_validate_connected):
            mat_prod = mat_prod @ matrix
            if jnp.all(mat_prod > 0):
                return matrix
```