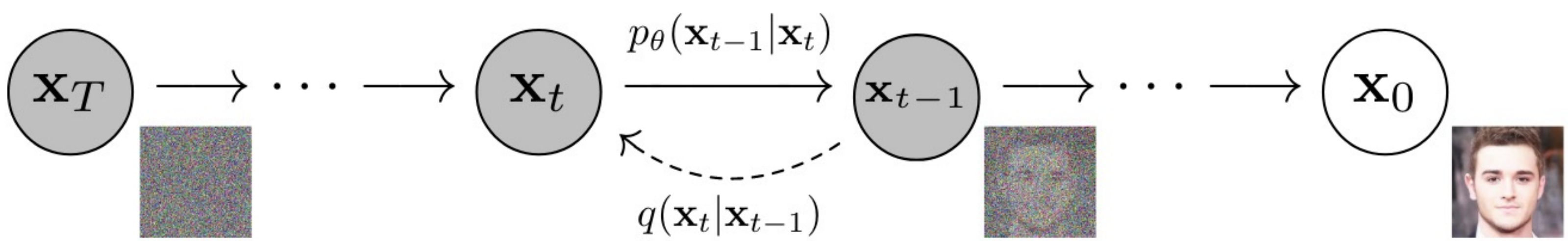
Diffusion Models in Discrete Space

Two technical routes

Vanilla Diffusion Models

- Forward: from clean image x_0, gradually add noise till we get a Gaussian noise x_T
- Reversed: from random Gaussian noise x_T, gradually denoise till we get a clean image x_0
- Trained with variational lower bound



$$\mathcal{L}_{\text{vlb}}(\mathbf{x}_0) = \mathop{\mathbb{E}}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{p_\theta(\mathbf{x}_T)} + \sum_{t=2}^{T} \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0, \mathbf{x}_t)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1) \right].$$

Vanilla Diffusion Models

- Re-parameterize the denoiser to predict the noise or x_{t-1}
- Simplify the training process and loss function

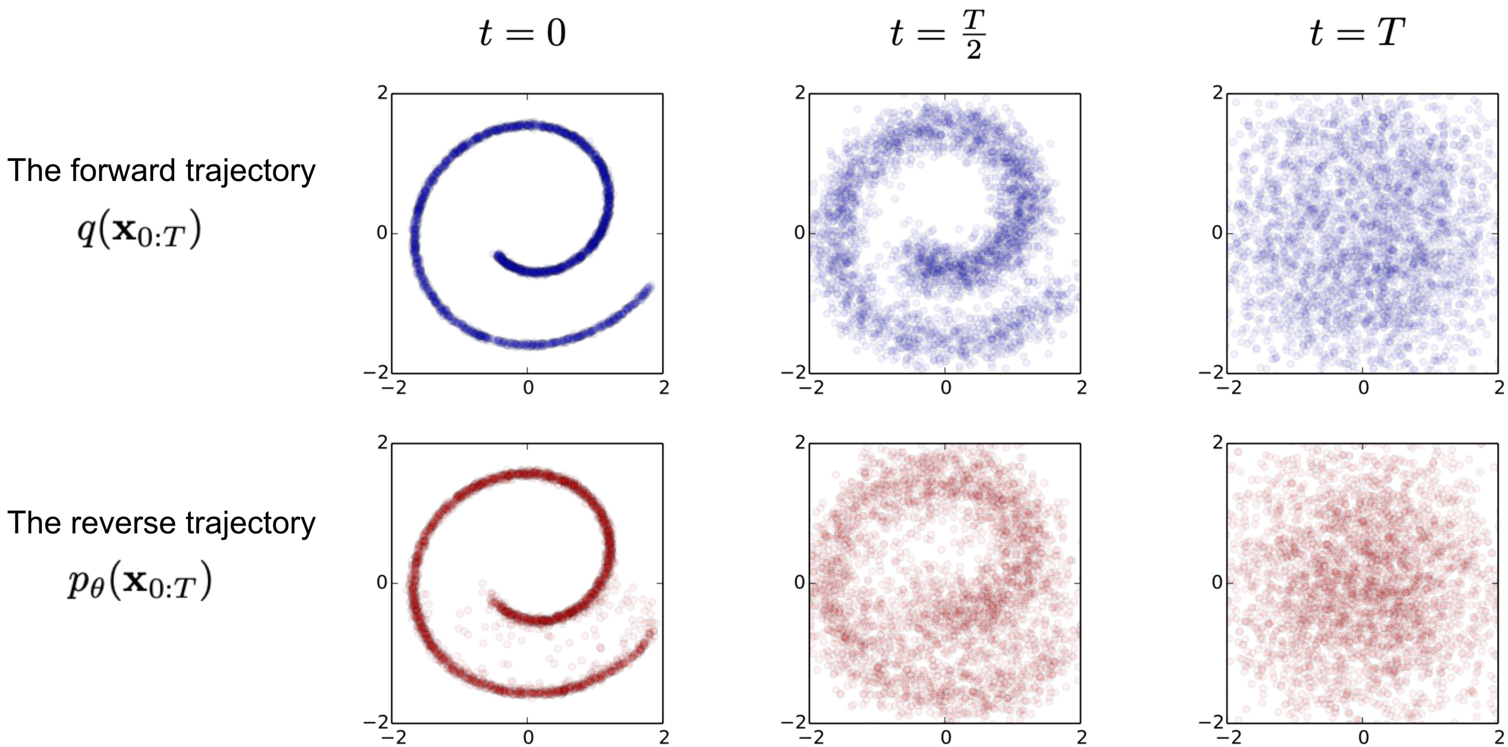
mean of the posterior $q(\mathbf{x}_{t-1}|\mathbf{x}_0, \mathbf{x}_t)$

$$\mathcal{L}_{\text{simple}}(\mathbf{x}_0) = \sum_{t=1}^{T} \mathop{\mathbb{E}}_{q(\mathbf{x}_t|\mathbf{x}_0)} ||\boxed{\mu_\theta(\mathbf{x}_t, t)} - \boxed{\hat{\mu}(\mathbf{x}_t, \mathbf{x}_0)}||^2,$$

predicted mean of $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$

Vanilla Diffusion Models

- Operates on continuous space
- How to apply Diffusion Model on discrete space?

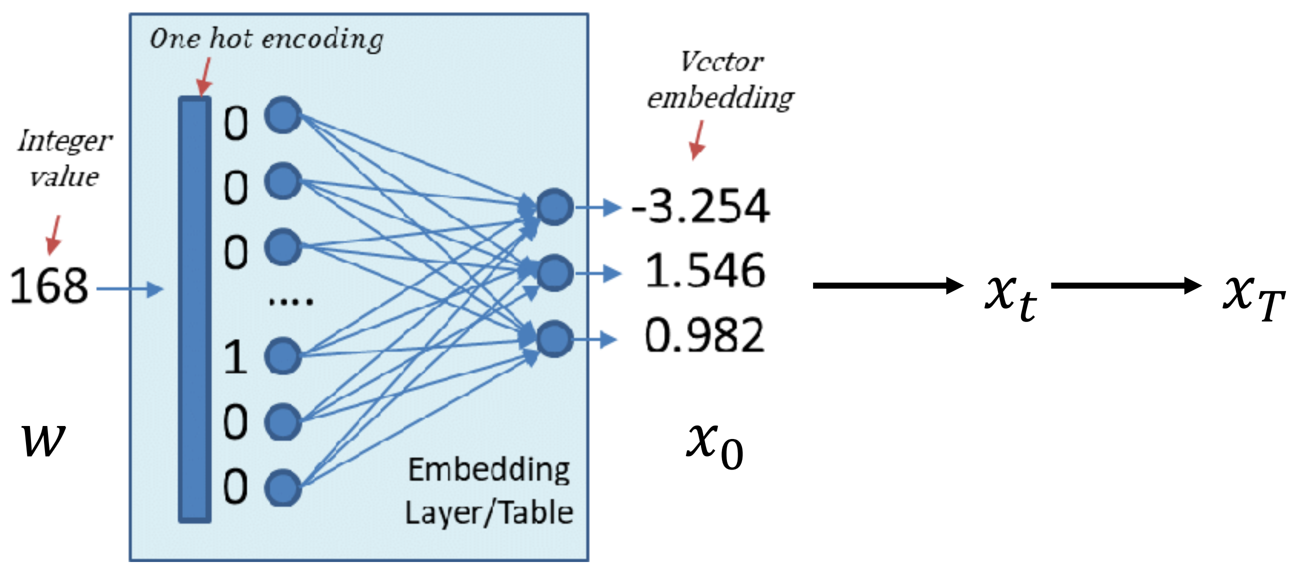


Transform discrete features into/out from continuous space

1. TRANSFORM

Embed Discrete Input

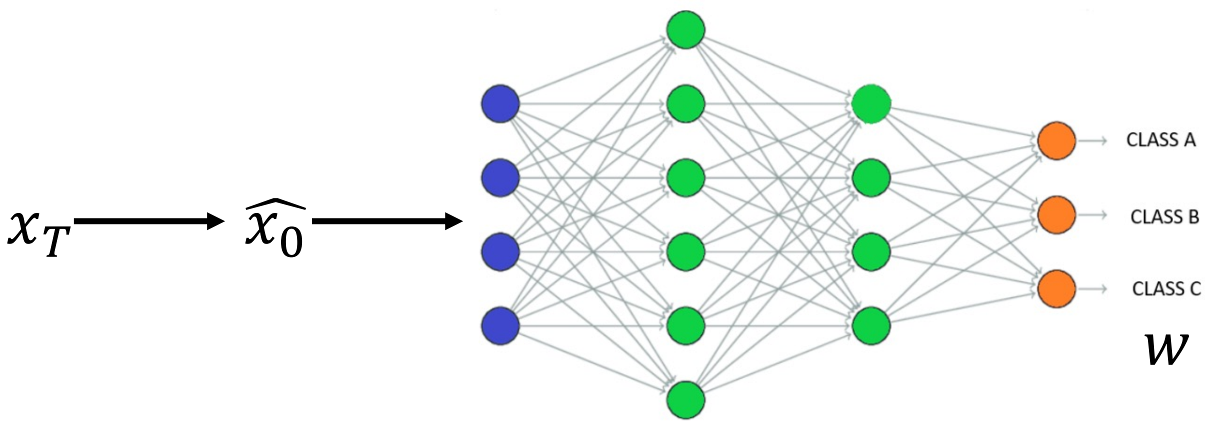
- Implement embedding layers to map discrete tokens w into continuous features x_0
- Adopt vanilla forward diffusion process to add noise



$$q_\phi(\mathbf{x}_0|\mathbf{w}) = \mathcal{N}(\textsc{Emb}(\mathbf{w}), \sigma_0 I)$$

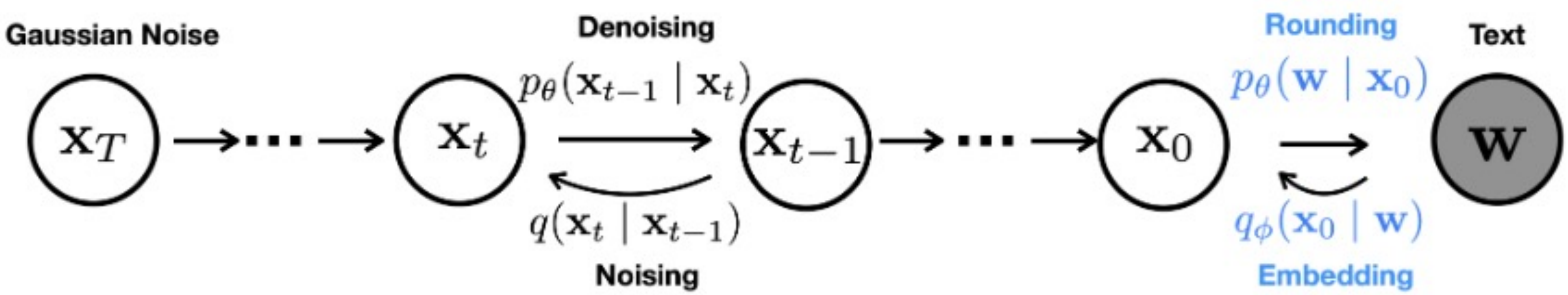
Classify Discrete Output

- Adopt vanilla reversed denoising diffusion process to generate $\widehat{x_0}$
- Implement classification network to obtain $\hat{w}$



$$p_\theta(\mathbf{w} \mid \mathbf{x}_0) = \prod_{i=1}^{n} p_\theta(w_i \mid x_i)$$

Loss Function



$$\mathcal{L}^{\text{e2e}}_{\text{simple}}(\mathbf{w}) = \mathop{\mathbb{E}}_{q_\phi(\mathbf{x}_{0:T}|\mathbf{w})}\Big[\mathcal{L}_{\text{simple}}(\mathbf{x}_0) + \overbrace{\boxed{||\textsc{Emb}(\mathbf{w}) - \mu_\theta(\mathbf{x}_1, 1)||^2}}^{\text{MSE Loss on embedding layer}} - \overbrace{\boxed{\log p_\theta(\mathbf{w}|\mathbf{x}_0)}}^{\text{Log-likelihood on classifier}}\Big] .$$

Co-train the embedding layer and classifier

Improve Generation Accuracy

- Train denoiser to explicitly model x_0 in every step

$$\mathcal{L}^{\text{e2e}}_{\mathbf{x}_0\text{-simple}}(\mathbf{x}_0) = \sum_{t=1}^{T} \mathbb{E}_{\mathbf{x}_t} ||f_\theta(\mathbf{x}_t, t) - \mathbf{x}_0||^2$$

$$\mathbf{x}_{t-1} = \sqrt{\bar{\alpha}} f_\theta(\mathbf{x}_t, t) + \sqrt{1-\bar{\alpha}}\epsilon$$

- Clamp predicted $\widehat{x_0}$ to the nearest word embeddings

$$\mathbf{x}_{t-1} = \sqrt{\bar{\alpha}} \cdot \text{Clamp}(f_\theta(\mathbf{x}_t, t)) + \sqrt{1-\bar{\alpha}}\epsilon.$$

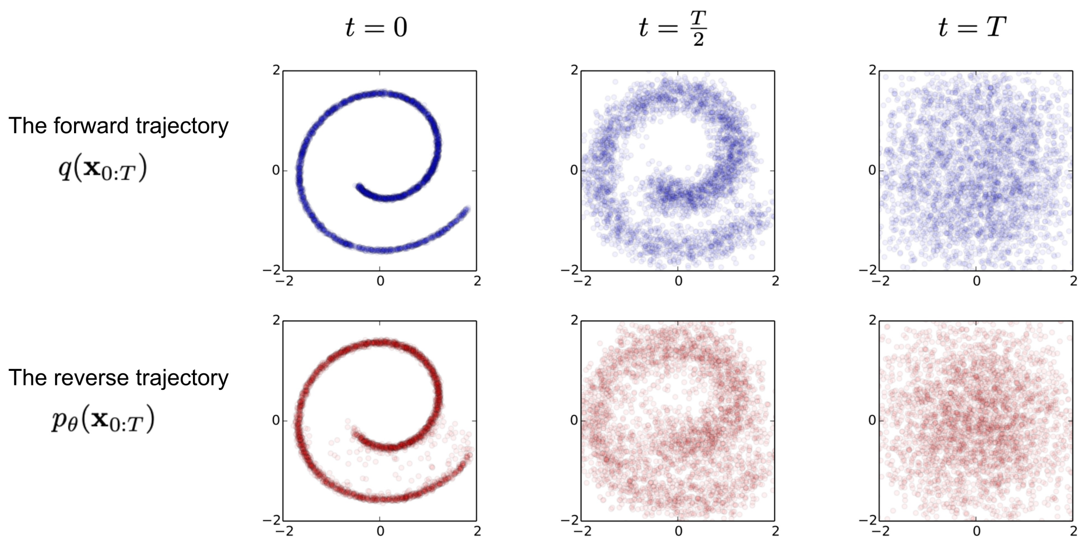
- Theoretically will guide the generated $\widehat{x_0}$ closer to the ground truth word embeddings

Diffusion models that operates on discrete space

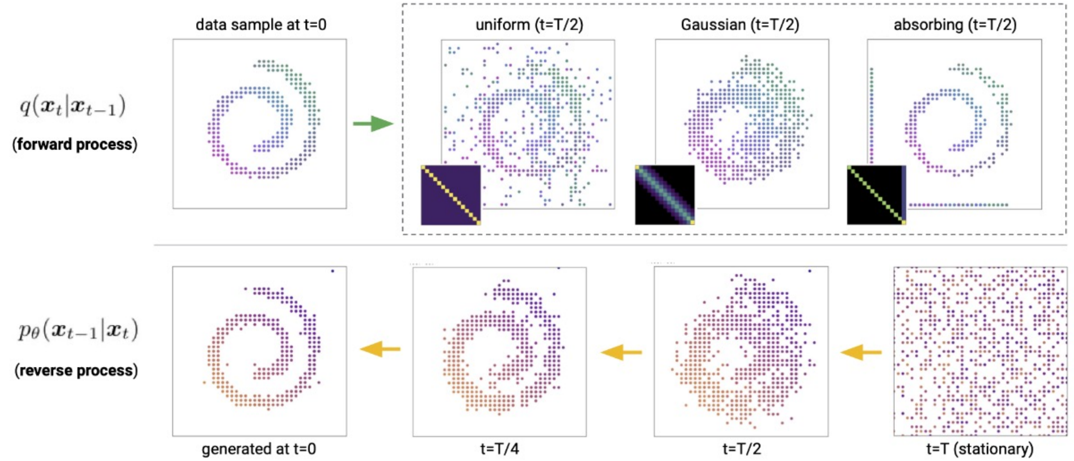
2. DISCRETE DIFFUSION

Generalize to Discrete Space

Diffusion models on
continuous space



Diffusion models on
discrete space



Discrete Forward Diffusion

- Using transition probability matrix to manipulate discrete features

$$[\boldsymbol{Q}_t]_{ij} = q(x_t = j | x_{t-1} = i).$$

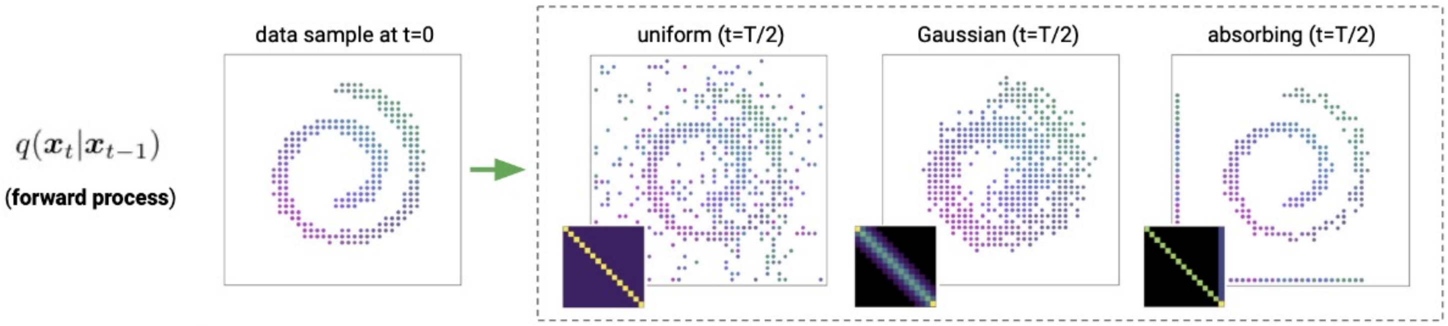
$$\boldsymbol{Q}_t = \begin{bmatrix} 0.4 & 0.6 & 0 \\ 0.3 & 0.2 & 0.5 \\ 0.5 & 0 & 0.5 \end{bmatrix}$$

x_{t-1}	x_t
0	1
2	0
1	2
1	2

$$x_{t-1} \xrightarrow[q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathrm{Cat}(\boldsymbol{x}_t; \boldsymbol{p} = \boldsymbol{x}_{t-1}\boldsymbol{Q}_t),]{} x_t$$

Discrete Forward Diffusion

- Various implementations of transition probability matrix



Uniform $\boldsymbol{Q}_t = (1-\beta_t)\boldsymbol{I} + \beta_t/K\ \mathbb{1}\mathbb{1}^T$ with $\beta_t \in [0,1]$

Absorbing Transit to [MASK] with probability β_t

$$[\boldsymbol{Q}_t]_{ij} = \begin{cases} 1 & \text{if} \quad i = j = m \\ 1-\beta_t & \text{if} \quad i = j \neq m \\ \beta_t & \text{if} \quad j = m, i \neq m \end{cases}$$

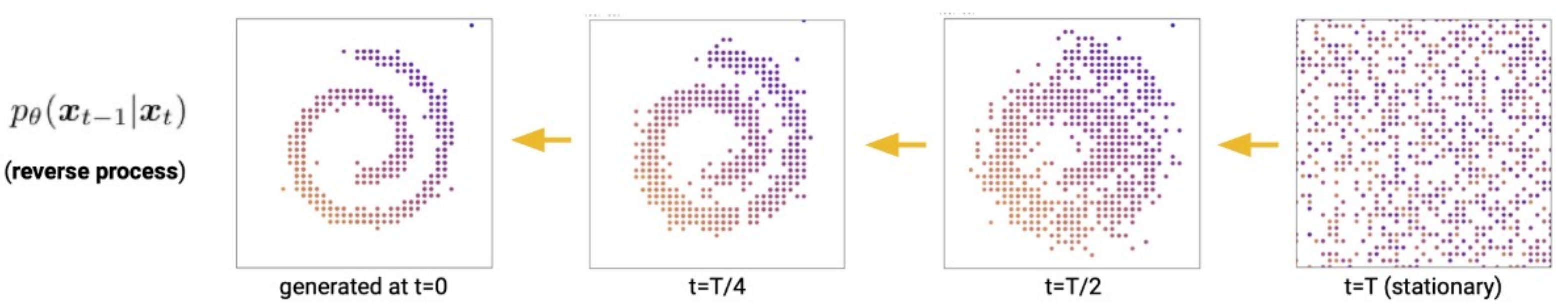
Gaussian Discretized, truncated Gaussian distribution

$$[\boldsymbol{Q}_t]_{ij} = \begin{cases} \dfrac{\exp\left(-\frac{4|i-j|^2}{(K-1)^2\beta_t}\right)}{\sum_{n=-(K-1)}^{K-1} \exp\left(-\frac{4n^2}{(K-1)^2\beta_t}\right)} & \text{if} \quad i \neq j \\ 1 - \sum_{l=0, l\neq i}^{K-1} [\boldsymbol{Q}_t]_{il} & \text{if} \quad i = j \end{cases}$$

Discrete Reversed Diffusion

- Parameterized logits $p_\theta(x_{t-1}|x_t)$
- Re-parameterize to predict logits $p_\theta(x_0|x_t)$

$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) \propto \sum_{\widetilde{\boldsymbol{x}}_0} q(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t|\widetilde{\boldsymbol{x}}_0)\widetilde{p}_\theta(\widetilde{\boldsymbol{x}}_0|\boldsymbol{x}_t).$$



Improve Generation Accuracy

- Supervise the prediction of x_0 on every step

Standard VLB loss

$$L_\lambda = \boxed{L_{\mathrm{vb}}} + \lambda\, \mathbb{E}_{q(\boldsymbol{x}_0)} \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} \boxed{\left[-\log \widetilde{p}_\theta(\boldsymbol{x}_0|\boldsymbol{x}_t)\right]}.$$

Log-likelihood of x_0 prediction

Reference

- X. L. Li, J. Thickstun, I. Gulrajani, P. Liang, and T. B. Hashimoto, "Diffusion-LM Improves Controllable Text Generation." NeuralPS 2022.
- J. Austin, D. D. Johnson, J. Ho, D. Tarlow, and R. van den Berg, "Structured Denoising Diffusion Models in Discrete State-Spaces." NeuralPS 2021.
- Z. Lin et al., "GENIE: Large Scale Pre-training for Text Generation with Diffusion Model." http://arxiv.org/abs/2212.11685
- K. K. Haefeli, K. Martinkus, N. Perraudin, and R. Wattenhofer, "Diffusion Models for Graphs Benefit From Discrete State Spaces." http://arxiv.org/abs/2210.01549

Thanks!