Towards Latent Diffusion Suitable For Text

Nesta Midavaine
University of Amsterdam
nesta.midavaine@student.uva.nl

Christian A. Naesseth
University of Amsterdam
c.a.naesseth@uva.nl

Grigory Bartosh
University of Amsterdam
g.bartosh@uva.nl

Abstract

Language diffusion models aim to improve sampling speed and coherence over autoregressive LLMs. We introduce Neural Flow Diffusion Models for language generation, an extension of NFDM that enables the straightforward application of continuous diffusion models to discrete state spaces. NFDM learns a multivariate forward process from the data, ensuring that the forward process and generative trajectory are a good fit for language modeling. Our model substantially reduces the likelihood gap with autoregressive models of the same size, while achieving sample quality comparable to that of previous latent diffusion models. The code is available at https://github.com/Nesta-gitU/discrete_diffusion.

1 Introduction

Language diffusion models go beyond next-token prediction by modeling text jointly, improving sampling speed for long sequences [6] and overcoming the reversal curse [22]. Existing methods achieve this by relying on predefined noise strategies [4, 11] or on noise schedules guided by linguistic priors [5, 29]. These methods require manual adjustments to the schedule to accommodate different data distributions. We propose learning the noising process directly from data, allowing the noise schedule to adapt conditionally to the input.

Language diffusion models fall into two families: discrete models that diffuse directly over tokens [11, 14, 4] and latent models that operate in continuous embedding space [12, 18, 28]. We introduce Neural Flow Diffusion Models (NFDM) for language generation, extending NFDM [3] to language. Instead of a fixed global schedule, NFDM learns a multivariate, data-conditioned forward process, yielding per-token noise schedules and a theoretically lower achievable ELBO compared to other continuous latent space approaches [2, 3, 27]. To demonstrate the effectiveness of our approach, we train several diffusion models with different learnable forward processes. We compare these to an autoregressive and static forward process baseline. We find that our best model significantly reduces the likelihood compared to the static forward process model and is within one standard deviation of the similarly sized autoregressive baseline. While we achieve sample quality comparable to previous diffusion model baselines.

2 Background

Score-based diffusion models define a forward a Stochastic Differential Equation (SDE) that perturbs data $\boldsymbol{x} \sim q(\boldsymbol{x})$ into Gaussian noise through a trajectory of variables $\{\boldsymbol{z}(t)\}_{t\in[0,1]}$ [31].

$$d\boldsymbol{z}_t = f(\boldsymbol{z}_t, t)\, dt + g(t)\, d\boldsymbol{w}, \quad \boldsymbol{z}_0 \approx \boldsymbol{x}, \tag{1}$$

Here $f(z_t, t) : \mathbb{R}^D \times [0,1] \to \mathbb{R}^D$ is a linear in $\boldsymbol{z}_t$ drift term, $g(t) : [0,1] \to \mathbb{R}_+$ a scalar variance and $\boldsymbol{w}$ is a standard Wiener process. To generate samples, the SDE is reversed [31].

$$d\boldsymbol{z}_t = \big(f(\boldsymbol{z}_t, t) - g^2(t)\, \nabla_{\boldsymbol{z}_t} \log q_t(\boldsymbol{z}_t)\big)dt + g(t)\, d\bar{\boldsymbol{w}}. \tag{2}$$

Here, $\bar{\boldsymbol{w}}$ denotes a standard Wiener process where time flows backward. Since the score $\nabla_{\boldsymbol{z}_t} \log q_t(\boldsymbol{z}_t)$ is unknown, a neural network is trained via denoising score matching over noise scales.

EurIPS 2025 Workshop on Principles of Generative Modeling (PriGM).

Neural Flow Diffusion Models NFDM extend Score-based diffusion models by allowing the forward process to be learned [3]. The conditional forward SDE is given implicitly via a learnable transformation $F_\varphi(\varepsilon, t, \boldsymbol{x})$. More specifically, NFDM characterizes the marginal distributions $q_\varphi(\boldsymbol{z}_t|\boldsymbol{x})$ of the forward process through,

$$\boldsymbol{z}_t = F_\varphi(\epsilon, t, \boldsymbol{x}), \quad \epsilon \sim q(\epsilon) = \mathcal{N}(\epsilon; 0, I) \tag{3}$$

During training, this formulation allows $\boldsymbol{z}_t \sim q_\varphi(\boldsymbol{z}_t|\boldsymbol{x})$ to be sampled efficiently. To completely define the distribution of the trajectories $\{\boldsymbol{z}(t)\}_{t\in[0,1]}$, NFDM introduces the forward conditional SDE,

$$\mathrm{d}\boldsymbol{z}_t = \tilde{f}^F(\boldsymbol{z}_t, t, \boldsymbol{x})\, dt + g_\varphi^2(t)\, d\boldsymbol{w}, \quad \text{where} \tag{4}$$

$$\tilde{f}^F(\boldsymbol{z}_t, t, \boldsymbol{x}) = f(\boldsymbol{z}_t, t, \boldsymbol{x}) + \frac{g_\varphi^2(t)}{2} \nabla_{\boldsymbol{z}_t} \log q_\varphi(\boldsymbol{z}_t|\boldsymbol{x}). \tag{5}$$

Equation 5 shows the forward drift term $\tilde{f}^F(z_t, t) : \mathbb{R}^D \times [0, 1] \to \mathbb{R}^D$, where $f(\boldsymbol{z}_t, t, \boldsymbol{x})$ and $\nabla_{\boldsymbol{z}_t} \log q_\varphi(\boldsymbol{z}_t|\boldsymbol{x})$ can be derived through F_φ. NFDM additionally defines a learnable reverse generative process. This framework enables the model to adapt to the data distribution while maintaining a tractable and simulation-free training process.

3 Method

3.1 NFDM in latent space

Neural Flow Diffusion Models define a learnable, continuous forward process dependent on the given data $\boldsymbol{x}$. In our case, $\boldsymbol{x}$ represents a sequence of discrete tokens. We parameterize a continuous forward process by using an embedding matrix $E_\varphi(\boldsymbol{x})$ and defining F_φ as a linear in ϵ transformation that progressively destroys information until it reaches a simple noise-like distribution at $t = 1$. The functions μ and σ in Equation 6 are parameterized as non-linear transformations of the input embeddings. Ensuring that the overall signal decays appropriately during the forward process, the exact parameterization of F_φ used in NFDM is provided in Appendix A.

$$F_\varphi^{\text{Gaussian}}(\epsilon, t, E_\varphi(\boldsymbol{x})) = \mu_\varphi(E_\varphi(\boldsymbol{x}), t) + \sigma_\varphi(E_\varphi(\boldsymbol{x}), t)\epsilon. \tag{6}$$

The distribution of $\boldsymbol{z}_0$ at $t = 0$ is centered on the embeddings with small variance. Since the embeddings are learned, we must include a reconstruction loss during training. Prior work [10, 8] finds that if this loss term is excluded, the embeddings can collapse to the same point. To get a generative model, we can parameterize the reverse flow $\tilde{f}_\varphi^B$ with the predicted embeddings $\hat{E}_\theta(\boldsymbol{z}_t, t)$ as done in Equation 25. We can then learn the parameters of this predictor jointly with the parameters of the forward process using the NFDM diffusion loss in Equation 8.

$$\hat{f}_{\theta,\varphi}^B(\boldsymbol{z}_t, t) = f(\boldsymbol{z}_t, t, \hat{E}_\theta(\boldsymbol{z}_t, t)) - \frac{g_\varphi^2(t)}{2} \nabla_{\boldsymbol{z}_t} \log q_\varphi(\boldsymbol{z}_t|\hat{E}_\theta(\boldsymbol{z}_t, t)) \tag{7}$$

$$\tilde{\mathcal{L}}_{\text{diff}} = \mathbb{E}_{u(t)\, q(\boldsymbol{x})\, q_\varphi(\boldsymbol{z}_t|\boldsymbol{x})} \left[\frac{1}{2g_\varphi^2(t)} \left\| \tilde{f}_\varphi^B(\boldsymbol{z}_t, t, \boldsymbol{x})) - \hat{f}_{\theta,\varphi}^B(\boldsymbol{z}_t, t) \right\|_2^2 \right] \tag{8}$$

In [3], it is shown that several existing approaches that learn the diffusion forward process can be viewed as special cases of NFDM. Among these, we experiment with MuLAN [27], which defines a dimension-wise multiplicative Gaussian forward process. In this formulation, the mean and variance functions (μ, σ) in Equation 6 are linear and parameterized such that each token position in the sequence maintains its own noise parameters. Consequently, the model can learn a soft noising order that adapts to the input. When adopting the linear forward process of MuLAN, the volatility term $g^2(t)$ can be analytically derived, yielding a simplified diffusion loss:

$$\mathcal{L}_{\text{diff}}^{\text{MuLAN}} = \lambda_F(t) \left\| E_\varphi(\boldsymbol{x}) - \hat{E}_\theta(\boldsymbol{z}_t, t) \right\|_2^2 \tag{9}$$

Here $\lambda_F(t)$ is determined by the definition of F_φ, see Appendix B for a detailed derivation. Previous diffusion language modeling works often employ a rescaled loss $\mathcal{L}_{\boldsymbol{x}} = \frac{1}{\lambda_F}\mathcal{L}_{\text{diff}}^{\text{MuLAN}}$ [18, 28]. However, in the general NFDM setting, such rescaling can cause training collapse, as the model will avoid early noise injection when this avoidance is not penalized. In contrast, MuLAN's per-dimension parameterization enables the stabilization of training even under rescaled loss. We fix a global signal-to-noise ratio (SNR) schedule [17] while allowing the model to learn how this global signal is distributed across dimensions. This design constrains the overall noise while enabling dimension-specific information retention. Implementation details for maintaining a fixed-average SNR under the rescaled loss are provided in Appendix B.4.

3.2 Sampling

One way to generate a sequence from NFDM is to sample from the prior distribution and simulate the learned reverse SDE. Alternatively, we can sample from a sequence of marginal distributions that match those of the SDE, as outlined in Algorithm 3. This approach introduces several sources of bias. Each latent state is drawn from its marginal rather than conditioned on the previous state. Moreover, relying on a single model prediction, $\hat{E}_\theta(\boldsymbol{z}_t, t)$, instead of sampling from the full conditional distribution, pulls samples toward the center of the data distribution. These sampling choices reduce variance and bias sample generation toward high-probability regions. This can be advantageous when targeting low perplexity under a teacher model. Finally, once $\boldsymbol{z}_0$ is obtained, the decoder $p_\varphi(\boldsymbol{x}|\boldsymbol{z}_0; E_\varphi)$ is used to map the embeddings back to discrete tokens. In our setting, the decoder inverts the encoder by assigning each latent vector to its closest vocabulary embedding, measured via dot-product similarity.

Algorithm 1 DDIM-style sampling [30]

Require: F_φ, g_φ, $\boldsymbol{x}$, T, E_φ
  $\Delta t = 1/T$, $\mathbf{z}_1 \sim p(\mathbf{z}_1)$
  for $t = 1, \dots, \frac{2}{T}, \frac{1}{T}$ **do**
    $s = t - \Delta t$
    $\epsilon = F_\varphi^{-1}(\boldsymbol{z}_t, t, E_\varphi(\boldsymbol{x}))$
    $\boldsymbol{z}_s = F_\varphi(\epsilon, s, \hat{E}_\theta(\boldsymbol{z}_t, t))$, $\boldsymbol{z}_t = \boldsymbol{z}_s$
  end for
  $\boldsymbol{x} \sim p_\varphi(\boldsymbol{x}|\boldsymbol{z}_0; E_\varphi)$

4 Experiments

We compare the different parameterizations of the forward process discussed above to prior work based on likelihood and sample quality. Comparisons are conducted for the task of unconditional generation on the ROCstories dataset [21], which comprises 90,000 examples of common-sense life stories.

4.1 Experimental setup

Models For the model that predicts the embeddings used to parameterize the reverse dynamics, we use BERT-base [7] with timestep embeddings, following the setup of [18]. To parameterize the transformation $F_\varphi^{\text{Gaussian}}$ in Latent-NFDM, we use a smaller BERT-style model with around 10 million parameters. This model is conditioned on time using Adaptive LayerNorm conditioning [23, 24]. More detailed modeling and hyperparameter choices for training are reported in Appendix E.

Evaluation Following [19], we report GPT-2 Large perplexity [26], MAUVE [25], diversity, memorization, and bits-per-character (ELBO). We follow [18] and use 2000 sampling steps. To compare likelihoods, we compute bits per character (bpc) in terms of ELBO.

Baselines We compare Latent-NFDM and MuLAN against Diffusion-LM [18], a diffusion-based language model that performs Gaussian diffusion directly in the embedding space. Diffusion-LM employs a static forward process and discrete timesteps in its original formulation. For consistency, we reformulate Diffusion-LM as a special case of NFDM and use this version for all training and evaluation, see Appendix C for details. We additionally evaluate samples from the ROCstories test set. For likelihood evaluation, we additionally report results for GPT-J and Diffusion-LM-Likelihood from [18]. GPT-J is an autoregressive transformer trained from scratch on the same dataset, with a comparable parameter count to our models. Diffusion-LM-Likelihood is a variant of Diffusion-LM specifically optimized for density estimation.

4.2 Experimental results

Table 1 shows the likelihood results. The standard MuLAN performs worse than the rescaled version, while MuLAN-rescaled and NFDM outperform both Diffusion-LM baselines. In doing so, NFDM reduces the likelihood gap towards the autoregressive GPT-J to almost zero.

For sample quality, we examine Table 2, which presents the discrete sampling results for the different models. We observe that MuLAN-Rescaled and NFDM both closely match the test set in terms of perplexity, diversity, and memorization. Diffusion-LM performs better in terms of perplexity and MAUVE. However, it should be noted that it achieves lower perplexity than the test set, which indicates diffusion-LM might be overly biased towards the mean of the language distribution. In terms of sample quality, MuLAN-rescaled outperforms the version trained without the rescaled loss by a large margin. When we compare the results of NFDM to MuLAN, we observe that the additional flexibility of NFDM yields a considerable improvement in sample quality. When we examine diversity, we see that none of the models have scores that indicate repetitive samples. In Appendix F we experiment with different sampling methods and hyperparameters.

Table 1: Bits Per Character computed in terms of ELBO.

Model	ELBO (SE)
GPT-J*	3.05 (-)
Diffusion-LM-Likelihood*	3.88 (-)
Diffusion-LM	5.94 (0.41)
MuLAN	5.78 (0.07)
MuLAN-Rescaled	3.47 (0.09)
NFDM	3.12 (0.05)

Table 2: Sample quality results for DDIM-style sampling.

Model	PPL ↓	MAUVE ↑	Diversity ↑	Memorization ↓
Diffusion-LM	21.05 (0.24)	0.76 (0.01)	0.18 (0.00)	0.14 (0.00)
MuLAN	139.74 (1.81)	0.01 (0.00)	0.21 (0.00)	0.02 (0.00)
MuLAN-Rescaled	27.71 (0.31)	0.60 (0.03)	0.20 (0.00)	0.12 (0.00)
NFDM	26.44 (0.22)	0.43 (0.03)	0.11 (0.00)	0.14 (0.00)
Dataset	26.106	0.957	0.200	0.141

5 Related Work

Diffusion models generate data by reversing a noise process [13, 31]. Diffusion models can achieve strong likelihoods by learning the SNR of this process, as done in VDM [17]. Latent diffusion is a straightforward method for extending diffusion models to language. Diffusion-LM embeds tokens and applies Gaussian diffusion with a square-root noise schedule [18], CDCD [8] trains with cross-entropy to encourage commitment to specific tokens. TEncDM diffuses in contextual language-model encodings and reports strong results [28]. Complementary to these latent approaches, recent advances in diffusion modeling, such as NFDM and MuLAN, learn multivariate, input-dependent forward processes with per-dimension/per-token noise allocation, thereby tightening ELBOs [3, 27].

6 Conclusion

We proposed Neural Flow Diffusion Models for language generation, extending NFDMs by learning the multivariate, data-conditioned forward process in latent space. Empirically, NFDM closes much of the likelihood gap with autoregressive models of comparable size while achieving comparable sample quality to prior diffusion-based baselines. The main limitations of this work are the limited dataset scale and its focus on unconditional generation. Future work should explore larger datasets and extend the method to conditional generation tasks.

References

[1] Brian D. O. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

[2] Grigory Bartosh, Dmitry Vetrov, and Christian A. Naesseth. Neural Diffusion Models, 2024. arXiv:2310.08337 [cs].

[3] Grigory Bartosh, Dmitry Vetrov, and Christian A. Naesseth. Neural flow diffusion models: Learnable forward process for improved diffusion modelling. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 73952–73985. Curran Associates, Inc., 2024.

[4] Andrew Campbell, Joe Benton, Valentin De Bortoli, Tom Rainforth, George Deligiannidis, and Arnaud Doucet. A Continuous Time Framework for Discrete Denoising Models, 2022. arXiv:2205.14987 [stat].

[5] Jiaao Chen, Aston Zhang, Mu Li, Alex Smola, and Diyi Yang. A Cheaper and Better Diffusion Language Model with Soft-Masked Noise. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4765–4775, Singapore, 2023. Association for Computational Linguistics.

[6] Google DeepMind. Gemini diffusion. `https://blog.google/technology/google-deepmind/gemini-diffusion/`, 2025. Accessed: 2025-10-11.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2019. arXiv:1810.04805 [cs].

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

[9] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-Weighted Linear Units for Neural Network Function Approximation in Reinforcement Learning, 2017. arXiv:1702.03118 [cs].

[10] Zhujin Gao, Junliang Guo, Xu Tan, Yongxin Zhu, Fang Zhang, Jiang Bian, and Linli Xu. Empowering Diffusion Models on the Embedding Space for Text Generation, 2024. arXiv:2212.09412 [cs].

[11] Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete Flow Matching, 2024. arXiv:2407.15595 [cs].

[12] Ishaan Gulrajani and Tatsunori B Hashimoto. Likelihood-based diffusion language models. *Advances in Neural Information Processing Systems*, 36:16693–16715, 2023.

[13] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.

[14] Emiel Hoogeboom, Alexey A Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. Autoregressive diffusion models. *arXiv preprint arXiv:2110.02037*, 2021.

[15] Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024.

[16] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, 2017. arXiv:1412.6980 [cs].

[17] Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational Diffusion Models, 2023. arXiv:2107.00630.

[18] Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-LM Improves Controllable Text Generation, 2022. arXiv:2205.14217 [cs].

[19] Justin Lovelace, Varsha Kishore, Chao Wan, Eliot Shekhtman, and Kilian Q Weinberger. Latent diffusion for language generation. *Advances in Neural Information Processing Systems*, 36:56998–57025, 2023.

[20] Gisiro Maruyama. Continuous Markov processes and stochastic equations. *Rendiconti del Circolo Matematico di Palermo*, 4(1):48–90, 1955.

[21] Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. A Corpus and Evaluation Framework for Deeper Understanding of Commonsense Stories, 2016. arXiv:1604.01696 [cs].

[22] Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large Language Diffusion Models, 2025. arXiv:2502.09992 [cs].

[23] William Peebles and Saining Xie. Scalable Diffusion Models with Transformers, 2023. arXiv:2212.09748 [cs].

[24] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual Reasoning with a General Conditioning Layer, 2017. arXiv:1709.07871 [cs].

[25] Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. MAUVE: Measuring the Gap Between Neural Text and Human Text using Divergence Frontiers. In *Advances in Neural Information Processing Systems*, volume 34, pages 4816–4828. Curran Associates, Inc., 2021.

[26] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[27] Subham Sekhar Sahoo, Aaron Gokaslan, Chris De Sa, and Volodymyr Kuleshov. Diffusion Models With Learned Adaptive Noise, 2024. arXiv:2312.13236 [cs].

[28] Alexander Shabalin, Viacheslav Meshchaninov, Egor Chimbulatov, Vladislav Lapikov, Roman Kim, Grigory Bartosh, Dmitry Molchanov, Sergey Markov, and Dmitry Vetrov. TEncDM: Understanding the Properties of the Diffusion Model in the Space of Language Model Encodings, 2025. arXiv:2402.19097 [cs].

[29] Neta Shaul, Itai Gat, Marton Havasi, Daniel Severo, Anuroop Sriram, Peter Holderrieth, Brian Karrer, Yaron Lipman, and Ricky T. Q. Chen. Flow Matching with General Discrete Paths: A Kinetic-Optimal Perspective, 2024. arXiv:2412.03487 [cs].

[30] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising Diffusion Implicit Models, 2022. arXiv:2010.02502.

[31] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations, 2021. arXiv:2011.13456.

A Details of NFDM

NFDM [3] extends the flexibility of the forward process by allowing its parameters to be conditioned directly on the input $\boldsymbol{x}$. In this formulation, the mean and variance of the forward process are parameterized via nonlinear functions of $\boldsymbol{x}$ and t:

$$F_{\varphi}^{\text{NFDM}}(\epsilon, t, E_{\varphi}(\boldsymbol{x})) = \mu_{\varphi}(E_{\varphi}(\boldsymbol{x}), t) + \sigma_{\varphi}(E_{\varphi}(\boldsymbol{x}), t)\epsilon. \tag{10}$$

$$\mu_{\varphi}(E_{\varphi}(\boldsymbol{x}), t) = (1-t)\, E_{\varphi}(\boldsymbol{x}) \;+\; t(1-t)\, \overline{\mu}_{\varphi}(E_{\varphi}(\boldsymbol{x}), t), \tag{11}$$

$$\sigma_{\varphi}(E_{\varphi}(\boldsymbol{x}), t) = \delta^{\,1-t} \left(\overline{\sigma}_{\varphi}(E_{\varphi}(\boldsymbol{x}), t)\right)^{t(1-t)}. \tag{12}$$

At the boundaries, the mean and variance functions satisfy

$$\mu_\varphi(E_\varphi(\boldsymbol{x}), 0) = E_\varphi(\boldsymbol{x}), \qquad \sigma_\varphi(E_\varphi(\boldsymbol{x}), 0) = \delta, \tag{13}$$

$$\mu_\varphi(E_\varphi(\boldsymbol{x}), 1) = 0, \qquad \sigma_\varphi(E_\varphi(\boldsymbol{x}), 1) = 1, \tag{14}$$

where we set $\delta = 0.01$. These boundary conditions ensure that

$$q_\varphi(\boldsymbol{z}_0 \mid \boldsymbol{x}) = \mathcal{N}\big(\boldsymbol{z}_0; E_\varphi(\boldsymbol{x}), \delta^2 I\big), \qquad q_\varphi(\boldsymbol{z}_1 \mid \boldsymbol{x}) = \mathcal{N}\big(\boldsymbol{z}_1; 0, I\big). \tag{15}$$

For intermediate $0 < t < 1$, μ_φ and σ_φ can take arbitrary values, defining a family of conditional Gaussian marginals

NFDM learns a model that can generate samples from the data distribution by parameterizing the reverse dynamics with a neural network that predicts the embeddings $\hat{E}_\theta(\boldsymbol{z}_t, t)$. We can learn the parameters of this predictor θ, jointly with the parameters of the forward process φ. The loss used to achieve this is derived in [3] and shown in Equation 21 for completeness.

As explained in the Background section 2 NFDM gets a generative model by learning to reverse the forward conditional SDE. This is done by first introducing the reverse conditional SDE, and then learning a generative reverse process parameterized by a prediction of the embeddings. The drift terms of the forward and reverse conditional SDEs both include $f(z_t, t, E_\varphi(\boldsymbol{x}))$. This is the ODE drift obtained by differentiating the trajectories from $\boldsymbol{z}_0$ to $\boldsymbol{z}_1$ induced by F_φ, over time. This leads to a velocity field corresponding to the conditional distribution, thereby defining a conditional ODE:

$$d\boldsymbol{z}_t = f(z_t, t, E_\varphi(\boldsymbol{x}))dt, \quad \text{where} \quad f(z_t, t, E_\varphi(\boldsymbol{x})) = \left.\frac{\partial F_\varphi(\epsilon, t, E_\varphi(\boldsymbol{x}))}{\partial t}\right|_{\epsilon = F_\varphi^{-1}(\boldsymbol{z}_t, t, E_\varphi(\boldsymbol{x}))} \tag{16}$$

As explained in section 2, the (conditional) distribution of the latent variable trajectories can then be described by an initial distribution $q(z_0 \mid E_\varphi(\boldsymbol{x}))$ and a Stochastic Differential Equation (SDE) with a drift term $\tilde{f}^F : \mathbb{R}^D \times [0,1] \to \mathbb{R}^D$, scalar variance $g(t) : [0,1] \to \mathbb{R}_+$, and a standard Wiener process $\boldsymbol{w}$:

$$\mathrm{d}\boldsymbol{z}_t = \tilde{f}^F(\boldsymbol{z}_t, t, E_\varphi(\boldsymbol{x}))\, dt + g_\varphi^2(t)\, d\boldsymbol{w}, \quad \text{where} \tag{17}$$

$$\tilde{f}^F(\boldsymbol{z}_t, t, \boldsymbol{x}) = f(\boldsymbol{z}_t, t, E_\varphi(\boldsymbol{x})) + \frac{g_\varphi^2(t)}{2} \nabla_{\boldsymbol{z}_t} \log q_\varphi(\boldsymbol{z}_t | E_\varphi(\boldsymbol{x})). \tag{18}$$

To create a target for our learned generative process, this forward process is then reversed by starting from the prior $z_1 \sim \mathcal{N}(z_1; 0, I)$ and following the reverse SDE [1]:

$$\mathrm{d}\boldsymbol{z}_t = \tilde{f}^B(\boldsymbol{z}_t, t, E_\varphi(\boldsymbol{x}))\, dt + g(t)\, d\bar{\boldsymbol{w}}, \quad \text{where} \tag{19}$$

$$\tilde{f}^B(\boldsymbol{z}_t, t, \boldsymbol{x}) = f(\boldsymbol{z}_t, t, E_\varphi(\boldsymbol{x})) - \frac{g_\varphi^2(t)}{2} \nabla_{\boldsymbol{z}_t} \log q_\varphi(z_t | E_\varphi(\boldsymbol{x})). \tag{20}$$

Here, $\bar{\boldsymbol{w}}$ denotes a standard Wiener process where time flows backward. This reversal results in a sample $\boldsymbol{z}_0 \sim p_\theta(\boldsymbol{z_0})$, such that decoding $\boldsymbol{z}_0$ through dot product similarity search with the vocabulary embeddings leads to an approximate sample from the data distribution $q(\boldsymbol{x})$. The reverse and forward processes can be learned jointly through the ELBO,

$$\mathbb{E}_{q(\boldsymbol{x})}[-\log \tilde{p}_{\theta,\varphi}(\boldsymbol{x})] \le \tilde{\mathcal{L}}_{\text{rec}} + \tilde{\mathcal{L}}_{\text{diff}} + \tilde{\mathcal{L}}_{\text{prior}}, \qquad \text{where} \tag{21}$$

$$\tilde{\mathcal{L}}_{\text{rec}} = \mathbb{E}_{q_\varphi(\boldsymbol{x}, \boldsymbol{z}_0)}[-\log p(\boldsymbol{x} | \boldsymbol{z}_0)], \tag{22}$$

$$\tilde{\mathcal{L}}_{\text{diff}} = \mathbb{E}_{u(t)\, q(\boldsymbol{x})\, q_\varphi(\boldsymbol{z}_t | \boldsymbol{x})} \left[\frac{1}{2 g_\varphi^2(t)} \left\| \tilde{f}_\varphi^B(\boldsymbol{z}_t, t, \boldsymbol{x})) - \hat{f}_{\theta,\varphi}^B(\boldsymbol{z}_t, t) \right\|_2^2 \right] \tag{23}$$

$$\tilde{\mathcal{L}}_{\text{prior}} = \mathbb{E}_{q(\boldsymbol{x})}[\mathcal{D}_{\text{KL}}(q_\varphi(\boldsymbol{z}_1 | \boldsymbol{x}) \parallel \tilde{p}(\boldsymbol{z}_1))]. \tag{24}$$

Here, the reverse drift is parameterized as,

$$\hat{f}_{\theta,\varphi}^B(\boldsymbol{z}_t, t) = f(\boldsymbol{z}_t, t, \hat{E}_\theta(\boldsymbol{z}_t, t)) - \frac{g_\varphi^2(t)}{2} \nabla_{\boldsymbol{z}_t} \log q_\varphi(\boldsymbol{z}_t | \hat{E}_\theta(\boldsymbol{z}_t, t)) \tag{25}$$

B Details of MuLAN

In this section, we derive a simplified version of the NFDM loss that holds when the forward process is parameterized as in MuLAN [27]. We derive this loss for the version of MuLAN that uses the auxiliary latent variable $\boldsymbol{c}$. The same derivations also when we omit the conditioning. Note that in the derivations below, multiplication and exponentiation of $\boldsymbol{\alpha}$, $\boldsymbol{\sigma}$ and $\boldsymbol{\gamma}$ are always elementwise.

B.1 MuLAN forward process

The forward process for MuLAN is defined through $F_{\varphi}^{MuLAN}(\epsilon, t, E_{\varphi}(\boldsymbol{x}), \boldsymbol{c})$:

$$F_{\varphi}^{MuLAN}(\epsilon, t, E_{\varphi}(\boldsymbol{x}), \boldsymbol{c}) = \boldsymbol{\alpha}_{\varphi}(t, \boldsymbol{c})E_{\varphi}(\boldsymbol{x}) + \boldsymbol{\sigma}_{\varphi}(t, \boldsymbol{c})\epsilon \tag{26}$$

Since in MuLAN we learn the SNR, this means that we have:

$$\boldsymbol{z}_t = \boldsymbol{\alpha}_{\varphi}(t, \boldsymbol{c})E_{\varphi}(\boldsymbol{x}) + \boldsymbol{\alpha}_{\varphi}(t, \boldsymbol{c})\epsilon \quad \text{where} \quad \boldsymbol{\alpha}_{\varphi}(t, \boldsymbol{c})^2 + \boldsymbol{\sigma}_{\varphi}(t, \boldsymbol{c})^2 = 1 \tag{27}$$

For simplicity, we define $\boldsymbol{\alpha} \equiv \boldsymbol{\alpha}_{\varphi}(t, \boldsymbol{c})$ and $\boldsymbol{\sigma} = \boldsymbol{\sigma}_{\varphi}(t, \boldsymbol{c})$ from here onwards. Additionally, we define $\boldsymbol{x} \equiv E_{\varphi}(\boldsymbol{x})$, since these derivations hold both in embedding space and when directly working with a continuous modality. Note that we can not have $\boldsymbol{x}$ reflect a discrete sequence like a sequence of tokens. In that case, we have to use the embeddings. Some simple rewriting can then give us the following connections:

$$\boldsymbol{x} = \frac{\boldsymbol{z}_t - \boldsymbol{\alpha}\varepsilon}{\boldsymbol{\alpha}} \quad \text{and} \quad \varepsilon = \frac{\boldsymbol{z}_t - \boldsymbol{\alpha}\boldsymbol{x}}{\boldsymbol{\sigma}} \tag{28}$$

As mentioned above, MuLAN learns the forward process through the Signal-To-Noise Ratio (SNR):

$$\textbf{SNR}(t) = \boldsymbol{\alpha}^2/\boldsymbol{\sigma}^2 = e^{-\boldsymbol{\gamma}_{\varphi}(t, \mathbf{c})} \tag{29}$$

For simplicity we use $\boldsymbol{\gamma} \equiv \boldsymbol{\gamma}_{\varphi}(t, \boldsymbol{c})$ from this point forward. $\boldsymbol{\gamma}$ is the quantity that we learn using a monotonically increasing network. Then we can rewrite the $\boldsymbol{\alpha}$ and $\boldsymbol{\sigma}$ coefficients in terms of the gamma function,

$$SNR = \frac{\boldsymbol{\alpha}^2}{1 - \boldsymbol{\alpha}^2} = e^{-\boldsymbol{\gamma}} \quad \Rightarrow \quad \boldsymbol{\alpha}^2 = \frac{e^{-\boldsymbol{\gamma}}}{1 + e^{-\boldsymbol{\gamma}}} = \frac{1}{1 + e^{\boldsymbol{\gamma}}} = \text{sigmoid}(-\boldsymbol{\gamma}) \tag{30}$$

$$\boldsymbol{\sigma}^2 = 1 - \boldsymbol{\alpha}^2 = \frac{1}{1 + e^{-\boldsymbol{\gamma}}} = \text{sigmoid}(\boldsymbol{\gamma}) \tag{31}$$

The NFDM loss is defined in terms of the reverse SDE. This reverse SDE consists of the conditional ODE, the score function and volatility with Brownian motion. The conditional ODE is:

$$f = \dot{\boldsymbol{\alpha}}x + \dot{\boldsymbol{\sigma}}\varepsilon = \dot{\boldsymbol{\alpha}}x + \frac{\dot{\boldsymbol{\sigma}}}{\boldsymbol{\sigma}}(\boldsymbol{z}_t - \boldsymbol{\alpha}\boldsymbol{x}) \tag{32}$$

The conditional score function is:

$$s = -\frac{\varepsilon}{\boldsymbol{\sigma}} = \frac{\boldsymbol{\alpha}\boldsymbol{x} - \boldsymbol{z}_t}{\boldsymbol{\sigma}^2} \tag{33}$$

Combining the drift of the ODE f and the score function s with the volatility $\boldsymbol{g}(t)$ we can write down the conditional forward SDE:

$$dz = f^F dt + \boldsymbol{g}(t) dw, \quad \text{where} \quad f^F = f + \frac{\boldsymbol{g}^2(t)}{2} s \tag{34}$$

Similarly, we can write down the conditional backwards SDE:

$$dz = f^B dt + \boldsymbol{g}(t) d\bar{w}, \quad \text{where} \quad f^B = f - \frac{\boldsymbol{g}^2(t)}{2} s \tag{35}$$

B.2 Markovian volatility

In general, volatility $\boldsymbol{g}(t)$ can be an arbitrary (positive) function of time t. However, there is one useful consideration that can help us parameterise more efficiently.

In diffusion models, our aim is to match the distribution of trajectories of the forward and reversed processes. The reverse process is Markovian by design. Therefore, to be able to match the distributions of trajectories, the forward process should also be Markovian. To guarantee this, we can find such a volatility $\boldsymbol{g}(t)$ that makes the forward process independent of $\boldsymbol{x}$. In the case of MuLAN and also VDM, this can be done analytically.

$$\begin{aligned}
f^F &= f + \frac{\boldsymbol{g}(t)^2}{2} s && (36)\\
&= \dot{\boldsymbol{\alpha}} x + \frac{\dot{\boldsymbol{\sigma}}}{\boldsymbol{\sigma}} (\boldsymbol{z}_t - \boldsymbol{\alpha}\boldsymbol{x}) + \frac{\boldsymbol{g}^2(t)}{2} \frac{\boldsymbol{\alpha}\boldsymbol{x} - \boldsymbol{z}_t}{\boldsymbol{\sigma}^2} && (37)\\
&= \underbrace{\left(\dot{\boldsymbol{\alpha}} - \frac{\dot{\boldsymbol{\sigma}}}{\boldsymbol{\sigma}} \boldsymbol{\alpha} + \frac{\boldsymbol{g}^2(t)}{2} \frac{\boldsymbol{\alpha}}{\boldsymbol{\sigma}^2} \right)}_{=0} \boldsymbol{x} + \left(\frac{\dot{\boldsymbol{\sigma}}}{\boldsymbol{\sigma}} - \frac{\boldsymbol{g}^2(t)}{2} \frac{1}{\boldsymbol{\sigma}^2} \right) \boldsymbol{z}_t && (38)\\
& && (39)
\end{aligned}$$

Setting the final term on the left equal to zero gives us an expression for the volatility $\boldsymbol{g}(t)$.

$$\begin{aligned}
\boldsymbol{g}^2(t) &= 2 \frac{\boldsymbol{\sigma}^2}{\boldsymbol{\alpha}} \left(\frac{\dot{\boldsymbol{\sigma}}}{\boldsymbol{\sigma}} \boldsymbol{\alpha} - \dot{\boldsymbol{\alpha}} \right) && (40)\\
&= 2\boldsymbol{\sigma}\dot{\boldsymbol{\sigma}} - 2\boldsymbol{\sigma}^2 \frac{\dot{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}} && (41)\\
&= (\boldsymbol{\sigma}^2)' - 2(\log \boldsymbol{\alpha})' \boldsymbol{\sigma}^2 && (42)
\end{aligned}$$

We can also rewrite the volatility in terms of the gamma function.

$$\begin{aligned}
\boldsymbol{g}^2(t) &= (\boldsymbol{\sigma}^2)' - 2(\log \boldsymbol{\alpha})' \boldsymbol{\sigma}^2 && (43)\\
&= (\boldsymbol{\sigma}^2)' - \frac{2\boldsymbol{\alpha}\dot{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}^2} \boldsymbol{\sigma}^2 && (44)\\
&= (\boldsymbol{\sigma}(\boldsymbol{\gamma}))' - \frac{(\boldsymbol{\sigma}(-\boldsymbol{\gamma}))'}{\boldsymbol{\sigma}(-\boldsymbol{\gamma})} \boldsymbol{\sigma}(\boldsymbol{\gamma}) && (45)\\
&= \boldsymbol{\sigma}(\boldsymbol{\gamma}) \left(1 - \boldsymbol{\sigma}(\boldsymbol{\gamma})\right) \dot{\boldsymbol{\gamma}} + \frac{\boldsymbol{\sigma}(-\boldsymbol{\gamma}) \left(1 - \boldsymbol{\sigma}(-\boldsymbol{\gamma})\right) \dot{\boldsymbol{\gamma}}}{\boldsymbol{\sigma}(-\boldsymbol{\gamma})} \boldsymbol{\sigma}(\boldsymbol{\gamma}) && (46)\\
&= \boldsymbol{\sigma}(\boldsymbol{\gamma}) \dot{\boldsymbol{\gamma}} \left(1 - \boldsymbol{\sigma}(\boldsymbol{\gamma}) + \underbrace{1 - \boldsymbol{\sigma}(-\boldsymbol{\gamma})}_{=\boldsymbol{\sigma}(\boldsymbol{\gamma})} \right) && (47)\\
&= \boldsymbol{\sigma}(\boldsymbol{\gamma}) \dot{\boldsymbol{\gamma}} && (48)
\end{aligned}$$

To keep the volatility function general, but preserve the connection with the gamma function, we can reparameterize the volatility function as follows:

$$\boldsymbol{g}^2(t) = \boldsymbol{\sigma}(\boldsymbol{\gamma})\dot{\boldsymbol{\gamma}}\eta \tag{49}$$

where η is an arbitrary non-negative function of time t. If we set $\eta = 1$, we will recover the Markovian volatility.

B.3 Simplified MuLAN ELBO

When using the Markovian parameterisation of the volatility and the Gaussian forward process used in Mulan, we can significantly simplify the ELBO. Doing this allows us to reduce the numerical error that can occur when multiplying many small values, and provides us with a more interpretable objective. We define the reverse process through prediction $\hat{\boldsymbol{x}}(\boldsymbol{z}_t, t)$ that we substitute into the conditional backward SDE:

$$dz = \hat{f}^B dt + \boldsymbol{g}(t) d\bar{w}, \quad \text{where} \quad \hat{f}^B(z,t) = f^B(z,t,\hat{\boldsymbol{x}}(\boldsymbol{z}_t,t)) \tag{50}$$

We know that the diffusion loss in NFDM is defined as:

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{u(t)\, q(\boldsymbol{x})\, q_\varphi(\boldsymbol{z}_t|\boldsymbol{x})}\left[\lambda_{f^B}\left\|f^B - \hat{f}^B\right\|_2^2\right], \quad \text{where} \quad \lambda_{f^B} = \frac{1}{2\boldsymbol{g}^2(t)} \tag{51}$$

For MuLAN, we can rewrite the f^B as:

$$\begin{aligned} f^B &= f - \frac{\boldsymbol{g}^2(t)}{2} s && (52)\\ &= \dot{\boldsymbol{\alpha}} x + \frac{\dot{\boldsymbol{\sigma}}}{\boldsymbol{\sigma}}(\boldsymbol{z}_t - \boldsymbol{\alpha}\boldsymbol{x}) - \frac{\boldsymbol{g}^2(t)}{2}\frac{\boldsymbol{\alpha}\boldsymbol{x} - \boldsymbol{z}_t}{\boldsymbol{\sigma}^2} && (53)\\ &= \left(\dot{\boldsymbol{\alpha}} - \frac{\dot{\boldsymbol{\sigma}}}{\boldsymbol{\sigma}}\boldsymbol{\alpha} - \frac{\boldsymbol{g}^2(t)}{2}\frac{\boldsymbol{\alpha}}{\boldsymbol{\sigma}^2}\right) x + \left(\frac{\dot{\boldsymbol{\sigma}}}{\boldsymbol{\sigma}} + \frac{\boldsymbol{g}^2(t)}{2}\frac{1}{\boldsymbol{\sigma}^2}\right)\boldsymbol{z}_t && (54) \end{aligned}$$

Since the second term doesn't depend on $\boldsymbol{x}$ and will cancel out in the ELBO, we can rewrite the ELBO as:

$$\mathcal{L} = \mathbb{E}_{u(t)\, q(\boldsymbol{x})\, q_\varphi(\boldsymbol{z}_t|\boldsymbol{x})}\left[\lambda_x \left\|\boldsymbol{x} - \hat{\boldsymbol{x}}\right\|_2^2\right] \tag{55}$$

We can now derive the λ_x coefficient,

$$\lambda_x = \frac{1}{2\boldsymbol{g}^2(t)}\left(\dot{\boldsymbol{\alpha}} - \frac{\dot{\boldsymbol{\sigma}}}{\boldsymbol{\sigma}}\boldsymbol{\alpha} - \frac{\boldsymbol{g}^2(t)}{2}\frac{\boldsymbol{\alpha}}{\boldsymbol{\sigma}^2}\right)^2 \tag{56}$$

$$= \frac{1}{2\boldsymbol{g}^2(t)}\left(\frac{\boldsymbol{\alpha}}{2}\frac{2\boldsymbol{\alpha}\dot{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}^2} - \frac{\boldsymbol{\alpha}}{2}\frac{2\boldsymbol{\sigma}\dot{\boldsymbol{\sigma}}}{\boldsymbol{\sigma}^2} - \frac{\boldsymbol{\alpha}}{2}\frac{g^2}{\boldsymbol{\sigma}^2}\right)^2 \tag{57}$$

$$= \frac{1}{2\boldsymbol{g}^2(t)}\frac{\boldsymbol{\alpha}^2}{2^2}\left(\frac{(\boldsymbol{\alpha}^2)'}{\boldsymbol{\alpha}^2} - \frac{(\boldsymbol{\sigma}^2)'}{\boldsymbol{\sigma}^2} - \frac{g^2}{\boldsymbol{\sigma}^2}\right)^2 \tag{58}$$

$$= \frac{1}{2}\frac{\boldsymbol{\alpha}^2}{\boldsymbol{\sigma}^2}\frac{1}{2^2\dot{\boldsymbol{\gamma}}\eta}\left(\frac{(\boldsymbol{\alpha}^2)'}{\boldsymbol{\alpha}^2} - \frac{(\boldsymbol{\sigma}^2)'}{\boldsymbol{\sigma}^2} - \frac{\boldsymbol{\sigma}^2\dot{\boldsymbol{\gamma}}\eta}{\boldsymbol{\sigma}^2}\right)^2 \tag{59}$$

$$= \frac{1}{2}e^{-\boldsymbol{\gamma}}\frac{1}{2^2\dot{\boldsymbol{\gamma}}\eta}\left(\frac{\boldsymbol{\sigma}(-\boldsymbol{\gamma})\left(1-\boldsymbol{\sigma}(-\boldsymbol{\gamma})\right)(-1)\dot{\boldsymbol{\gamma}}}{\boldsymbol{\sigma}(-\boldsymbol{\gamma})} - \frac{\boldsymbol{\sigma}(\boldsymbol{\gamma})\left(1-\boldsymbol{\sigma}(\boldsymbol{\gamma})\right)\dot{\boldsymbol{\gamma}}}{\boldsymbol{\sigma}(\boldsymbol{\gamma})} - \dot{\boldsymbol{\gamma}}\eta\right)^2 \tag{60}$$

$$= \frac{1}{2}e^{-\boldsymbol{\gamma}}\frac{1}{2^2\dot{\boldsymbol{\gamma}}\eta}\left(\big[-\underbrace{\left(1-\boldsymbol{\sigma}(-\boldsymbol{\gamma})\right)}_{=\boldsymbol{\sigma}(\boldsymbol{\gamma})} - 1 + \boldsymbol{\sigma}(\boldsymbol{\gamma})\big]\dot{\boldsymbol{\gamma}} - \dot{\boldsymbol{\gamma}}\eta\right)^2 \tag{61}$$

$$= \frac{1}{2}e^{-\boldsymbol{\gamma}}\frac{1}{2^2\dot{\boldsymbol{\gamma}}\eta}\left(-\dot{\boldsymbol{\gamma}} - \dot{\boldsymbol{\gamma}}\eta\right)^2 \tag{62}$$

$$= \frac{1}{2}e^{-\boldsymbol{\gamma}}\frac{\dot{\boldsymbol{\gamma}}^2\left(1+\eta\right)^2}{2^2\dot{\boldsymbol{\gamma}}\eta} \tag{63}$$

$$= \frac{1}{2}e^{-\boldsymbol{\gamma}}\dot{\boldsymbol{\gamma}}\frac{\left(1+\eta\right)^2}{2^2\eta} \tag{64}$$

Since nothing except the last coefficient depends on function η. By computing the first and second derivatives of this term we can find the optimal η. It is $\eta = 1$. Therefore, the optimal volatility function is a Markovian volatility. We can also find a connection with the SNR function, when $\eta = 1$.

$$\text{SNR}' = (e^{-\gamma})' = -e^{-\gamma}\dot{\gamma}, \quad \lambda_x = \frac{1}{2}e^{-\gamma}\dot{\gamma} = -\frac{1}{2}\text{SNR}' \tag{65}$$

B.4 Fixed average SNR

In subsection 3.1, we discussed restricting the flexibility of the learned forward process. Specifically, we use a restricted formulation of $\boldsymbol{\gamma}_\varphi(t, \boldsymbol{c})$ to prevent collapse when training with the rescaled ELBO.

$$\boldsymbol{\gamma}_\varphi(t, \boldsymbol{c}))_j = \gamma(t)^{\text{global}} + \boldsymbol{\gamma}'_\varphi(t, \boldsymbol{c})_j - \tilde{\gamma}_\varphi(t, c), \quad \text{where,} \tag{66}$$

$$\tilde{\gamma}_\varphi(t, c)' = \log D - \log \sum_i^D \exp(-\boldsymbol{\gamma}'_\varphi(t, \boldsymbol{c})_i) \tag{67}$$

This parameterization fixes the global average of the SNR, over the spatial dimensions of gamma, to the SNR given by $\gamma(t)^{\text{global}}$. In our case, $\gamma(t)^{\text{global}}$ is fixed to the SNR of the baseline model Diffusion-LM. this SNR is derived in Appendix C. In the following, we show that this is the case by computing the average SNR over the dimensions D of $\boldsymbol{\gamma}_\varphi(\boldsymbol{c}, t)$. Where D corresponds to the total dimensionality of $\boldsymbol{\gamma}_\varphi(\boldsymbol{c}, t)$, multiplying the spatial dimensions.

$$\frac{1}{D}\sum_{j=1}^{D} SNR(t)_j = \frac{1}{D}\sum_{j=1}^{D} e^{-\gamma_\varphi(t,\boldsymbol{c})_j} \tag{68}$$

$$= \frac{1}{D}\sum_{j=1}^{D} e^{-\gamma(t)^{\text{global}} - \gamma'_\varphi(t,\boldsymbol{c})_j + \tilde{\gamma}_\varphi(t,\boldsymbol{c})} \tag{69}$$

$$= \frac{1}{D}\sum_{j=1}^{D} e^{-\gamma(t)^{\text{global}} - \gamma'_\varphi(t,\boldsymbol{c})_j + \log D - \log \sum_i e^{(-\gamma'_\varphi(t,\boldsymbol{c})_i)}} \tag{70}$$

$$= \frac{1}{D}\sum_{j=1}^{D} e^{-\gamma(t)^{\text{global}}} e^{-\gamma'_\varphi(t,\boldsymbol{c})_j} D \frac{1}{\sum_i e^{(-\gamma'_\varphi(t,\boldsymbol{c})_i)}} \tag{71}$$

$$= e^{-\gamma(t)^{\text{global}}} \left(\sum_{j=1}^{D} e^{-\gamma'_\varphi(t,\boldsymbol{c})_j}\right) \frac{1}{\sum_i^D e^{(-\gamma'_\varphi(t,\boldsymbol{c})_i)}} \tag{72}$$

$$= e^{-\gamma(t)^{\text{global}}} \tag{73}$$

$$\tag{74}$$

C Continuous time Diffusion-LM

In this work, we use Diffusion-LM [18] as the baseline model with a static, non-learnable, forward process. We focus on using the continuous-time NFDM framework in this work. Since Diffusion-LM as presented in [18] we rewrite it to continuous time form using the NFDM framework in the section below. Note that as T approaches infinity, the discrete-time model matches the continuous-time model precisely. Since [18] uses $T = 2000$ that means we saw very similar training losses in practice between the two models.

The Diffusion-LM forward process is described by $q(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1}) = \mathcal{N}(\boldsymbol{z}_t; \sqrt{1-\beta_t}\boldsymbol{z}_{t-1}, \beta_t \boldsymbol{I})$, where $\boldsymbol{z}_t$ are the continuous latent variables. The forward variance schedule is governed by $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$, with $\bar{\alpha}_t = 1 - \sqrt{t/T + s}$. This corresponds to the following parameterization of F in the NFDM framework,

$$F(x, t, \varepsilon) = \alpha(t) E_\varphi(\boldsymbol{x}) + \sigma(t)\varepsilon \quad \text{where,} \tag{75}$$

$$\alpha(t) = \sqrt{1 - \sqrt{t+s}} \tag{76}$$

$$\sigma(t) = \sqrt{\sqrt{t+s}} \tag{77}$$

In diffusion-LM, s is a small constant set to 0.0001. To ensure differentiability with respect to t, we set it to the following in NFDM,

$$s = (0.99 - t) \cdot 0.0001 \tag{78}$$

We can also rewrite this in terms of the $\gamma(t)$ parameterization used in VDM [17] and MuLAN [27]. This is necessary since we want to use the Diffusion-LM forward process as the reference process when training MuLAN with a rescaled loss. Using the connection of $\gamma(t)$ and $\alpha(t)$ in Equation 30 we get,

$$\alpha^2 = \text{sigmoid}(-\gamma(t)) = 1 - \sqrt{t+s} \tag{79}$$
$$\Leftrightarrow \tag{80}$$
$$\frac{1}{1+e^{\gamma(t)}} = 1 - \sqrt{t+s} \tag{81}$$
$$\Leftrightarrow \tag{82}$$
$$e^{\gamma(t)} = \frac{1}{1-\sqrt{t+s}} - 1 \tag{83}$$
$$\Leftrightarrow \tag{84}$$
$$\gamma(t) = \ln\left(\frac{1}{1-\sqrt{t+s}} - 1\right) \tag{85}$$
$$= \ln\left(\frac{\sqrt{t+s}}{1-\sqrt{t+s}}\right) \tag{86}$$
$$= \ln\left(\sqrt{t+s}\right) - \ln\left(1-\sqrt{t+s}\right) \tag{87}$$

Note that we can arrive at the same conclusion by starting from the connection between $\gamma(t)$ and $\sigma(t)$ in this case. Additionally, in the used version of gamma, we clamp $\sqrt{t+s}$ between $1e^{-6}$ and $1 - 1e^{-6}$ for numerical stability in the fraction and natural logarithm.

To match the continuous-time formulation of diffusion in NFDM to the discrete-time formulation in Diffusion-LM, we need to set the volatility $g^2(t)$ according to the following formula presented in [17], which allows us to convert from discrete to continuous time diffusion models.

$$g^2(t) = \frac{d\sigma(t)^2}{dt} - 2\frac{d\log\alpha(t)}{dt}\sigma_t^2 \tag{88}$$
$$= \frac{0.9999}{(2\sigma^2(1-\sigma^2))} \tag{89}$$

Since the forward process is quite sharp, it has high second derivatives at the endpoints, as can be seen in Figure 3. Training with the standard ELBO is unstable in this formulation. Thus, we train with the rescaled $\boldsymbol{x}$-prediction version of the ELBO, similar to what is done in the original Diffusion-LM paper.

$$\mathcal{L}_x = \mathbb{E}_{u(t)\,q(\boldsymbol{x})\,q_\varphi(\boldsymbol{z}_t|\boldsymbol{x})}\left[\left\|E_\varphi(\boldsymbol{x}) - \hat{E}_\varphi(\boldsymbol{z}_t, t)\right\|_2^2\right] \tag{90}$$

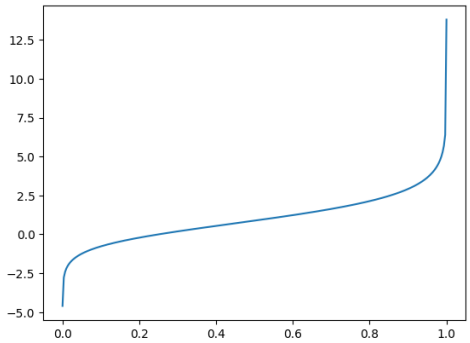

Figure 1: Diffusion-LM $\gamma(t)$ for $t \in [0, 1]$

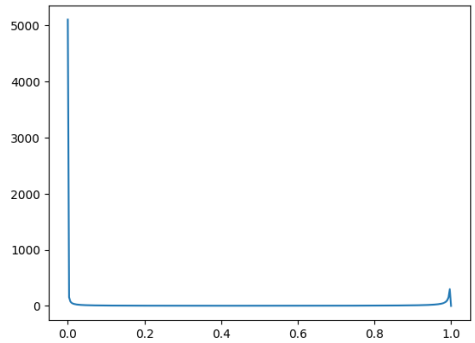

Figure 2: Diffusion-LM $\dot{\gamma}(t)$ for $t \in [0, 1]$

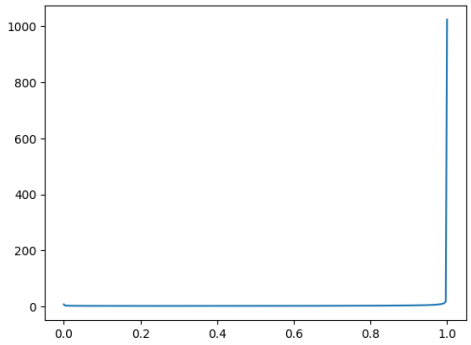

Figure 3: Diffusion-LM $g(t)^2$ for $t \in [0, 1]$

D Sampling

Algorithm 2 Sampling with an SDE

Require: F_φ, g_φ, $\boldsymbol{x}$, T, E_φ
  $\Delta t = 1/T$, $\mathbf{z}_1 \sim p(\mathbf{z}_1)$
  for $t = 1, \dots, \frac{2}{T}, \frac{1}{T}$ **do**
    $\bar{\boldsymbol{w}} \sim \mathcal{N}(0, 1)$
    $\boldsymbol{z}_{t-\Delta t} = \boldsymbol{z}_t - f_\varphi\left(z_t, t, \hat{E}_\theta(\boldsymbol{z}_t, t)\right)\Delta t + g_\varphi(t)\bar{\boldsymbol{w}}\sqrt{\Delta t}$
  end for
  $\boldsymbol{x} \sim p_\varphi(\boldsymbol{x}|\boldsymbol{z}_0; E_\varphi)$

Algorithm 3 Sampling with a conditional Markov chain

Require: F_φ, g_φ, $\boldsymbol{x}$, T, E_φ, $\tilde{\sigma}_{s|t}$
  $\Delta t = 1/T$, $\mathbf{z}_1 \sim p(\mathbf{z}_1)$
  for $t = 1, \dots, \frac{2}{T}, \frac{1}{T}$ **do**
    $s = t - \Delta t$
    $\epsilon_{new} \sim \mathcal{N}(0, 1)$
    $\epsilon_{old} = F_\varphi^{-1}(\boldsymbol{z}_t, t, E_\varphi(\boldsymbol{x})))$
    $\tilde{\epsilon}_{s|t} = \sqrt{1 - \tilde{\sigma}_{s|t}}\epsilon_{old} + \sqrt{\tilde{\sigma}_{s|t}}\epsilon_{new}$
    $\boldsymbol{z}_s = F_\varphi(\tilde{\epsilon}_{s|t}, s, \hat{E}_\theta(\boldsymbol{z}_t, t))$
  end for
  $\boldsymbol{x} \sim p_\varphi(\boldsymbol{x}|\boldsymbol{z}_0; E_\varphi)$

In the main text, we introduced two sampling approaches. However, the discrete sampling method discussed there represents only one possible way to sample from a discrete process in NFDM. Moreover, the described SDE can itself be converted into an ODE. This section explores additional sampling possibilities. First, we discuss sampling from the reverse SDE, explaining how it can be reformulated as an ODE at sampling time. Second, we note that sampling through the generative Markov chain is connected to the generative SDE via its marginals and asymptotic behavior as the number of steps approaches infinity. We then use these connections as a heuristic for sampling from the discrete process.

D.1 Generative SDE

To sample a sequence from NFDM, we can follow the sampling setup described in [3]. Here we sample from the prior distribution $\boldsymbol{z}_1 \sim p(\boldsymbol{z}_1)$, and simulate the reverse SDE in Equation 19.

$$\mathrm{d}z_t = \left[f_\varphi\left(z_t, t, \hat{E}_\theta(\boldsymbol{z}_t, t)\right) - \frac{g_\varphi^2(t)}{2}\nabla_{z_t} \log q_\varphi\left(z_t \mid \hat{E}_\theta(\boldsymbol{z}_t, t)\right)\right] \mathrm{d}t + g_\varphi(t)\,\mathrm{d}\bar{w}. \tag{91}$$

We can vary $g_\varphi(t)$ during inference to control the level of stochasticity in the process. In the extreme case where $g_\varphi(t) \equiv 0$ Equation 92 becomes a deterministic ODE with the same marginals $q(\boldsymbol{z}_t)$ as the SDE. Deterministic sampling may increase the sampling speed compared to the stochastic method.

$$\mathrm{d}\boldsymbol{z}_t = f_\varphi(z_t, t, \hat{E}_\theta(\boldsymbol{z}_t, t)) \tag{92}$$

We can simulate the generative SDE using Euler-Maruyama discretization [20], and the generative ODE using the standard Euler method. The complete sampling procedure corresponding to these methods is shown in Algorithm 2. Alternatively, adaptive ODE or SDE solvers can be employed to trade off between the number of function evaluations and solution accuracy.

D.2 Generative conditional Markov chain

In Appendix C, we discuss how to convert from a discrete-time model to a continuous-time model with the same forward process. Similarly, we can convert from continuous to discrete. Here, we discuss this conversion and what might happen when we use a model trained with the continuous time loss to sample from the discrete-time process for a finite number of steps. The discrete-time process is specified using the forward conditional Markov chain,

$$\bar{q}_\varphi(\boldsymbol{x}, \boldsymbol{z}_{0:1}) = q(\boldsymbol{x}) q_\varphi(\boldsymbol{z}_0|\boldsymbol{x}) \left[\prod_t \bar{q}_\varphi(\boldsymbol{z}_{t+\Delta t}|\boldsymbol{z}_t, \boldsymbol{x}) \right]. \tag{93}$$

which we approximate using the reverse conditional Markov chain,

$$\bar{p}(\boldsymbol{x}, \boldsymbol{z}_{0:1}) = p(\boldsymbol{z}_1) \left[\prod_t \bar{p}_\theta(\boldsymbol{z}_{t-\Delta t}|\boldsymbol{z}_t, \hat{E}_\theta(\boldsymbol{z}_t, t)) \right] p_\theta(\boldsymbol{x}|\boldsymbol{z}_0). \tag{94}$$

We define the posterior distribution $\bar{q}(\boldsymbol{z}_{t+\Delta t}|\boldsymbol{z}_t, \boldsymbol{x})$ by describing how to sample $\boldsymbol{z}_s$ given $\boldsymbol{z}_t$ and $\boldsymbol{x}$ where $s < t$. To do this while preserving the correct marginal distribution $q_\varphi(\boldsymbol{z}_s|\boldsymbol{x})$. We can combine the Gaussian noise terms ϵ_s and ϵ_t into a new noise term $\tilde{\epsilon}_{s|t}$. We fill this into $F_\varphi^{\text{Gaussian}}(\tilde{\epsilon}_{s|t}, t, \boldsymbol{x})$ to sample $\boldsymbol{z}_s$. Specifically, we have:

$$\tilde{\epsilon}_{s|t} = \sqrt{1 - \tilde{\sigma}_{s|t}} \epsilon_{old} + \sqrt{\tilde{\sigma}_{s|t}} \epsilon_{new}, \quad \text{gives} \tag{95}$$

$$\tag{96}$$

$$\begin{aligned} z_s &= F_\varphi^{\text{Gaussian}}(\tilde{\epsilon}_{s|t}, t, E_\varphi(\boldsymbol{x})) && (97)\\ &= \mu_\varphi + \sqrt{1 - \tilde{\sigma}_{s|t}^2} \epsilon_t \sigma_{\varphi,s} + \tilde{\sigma}_{s|t} \epsilon_s \sigma_{\varphi,s} && (98)\\ &= \mu_\varphi + \sqrt{\sigma_{\varphi,s}^2 - \sigma_{\varphi,s}^2 \tilde{\sigma}_{s|t}^2} \epsilon_t + \tilde{\sigma}_{s|t} \sigma_{\varphi,s} \epsilon_s && (99) \end{aligned}$$

This result matches [2] in the fact that by marginalizing ϵ_t and $\tilde{\epsilon}_{s|t}$, we obtain a normal distribution with mean μ_ε and variance $\sigma_s^2 - \sigma_s^2 \tilde{\sigma}_{s|t}^2 + \sigma_s^2 \tilde{\sigma}_{s|t}^2 = \sigma_s^2$. Therefore, this sampling procedure satisfies $q_\varphi(\boldsymbol{z}_s \mid \boldsymbol{x}) = \int q_\varphi(\boldsymbol{z}_t \mid \boldsymbol{x})\, \bar{q}_\varphi(\boldsymbol{z}_s \mid \boldsymbol{z}_t, \boldsymbol{x})\, d\boldsymbol{z}_t$. Thus, the discrete forward process has the same marginals as the continuous process, regardless of the value we choose for $\tilde{\sigma}_{s|t}$. If we want the discrete forward process to match the continuous forward as $T \to \infty$, we can set $\tilde{\sigma}_{s|t}$ equal to an SNR-based quantity [17]:

$$\tilde{\sigma}_{s|t} = \tilde{\sigma}_{s|t}^* = 1 - \frac{SNR(t)}{SNR(s)} \tag{100}$$

Note that this only works if we have easy access to the SNR, which is the case for Diffusion-LM, MuLAN. But for the more general forward process in NFDM, it is not. We can sample from the corresponding reverse process in Equation 94 by predicting $\hat{E}_\theta(\boldsymbol{z}_t, t)$ and filling this into Equation 99, the whole procedure for sampling is shown Algorithm 3.

There are two important observations about this sampling procedure. First, the equality between discrete and continuous only exists when we have an infinite number of timesteps. When we do not, bias will be injected into the sampling procedure. Second, the sampling procedure used in Algorithm 3 approximates the posterior over embeddings given $\boldsymbol{z}_t$ by selecting a single prediction rather than sampling from the full conditional distribution. This deterministic approach reduces the

variance of the sampling process and introduces a bias. The model is more likely to generate samples corresponding to the modes of the data distribution $q(\boldsymbol{x})$.

Since in some instances analytically determining $\tilde{\sigma}_{s|t}$ is not possible, we propose employing a further heuristic. We can treat $\tilde{\sigma}_{s|t} \in [0, 1]$ as a tunable hyperparameter of the sampling process, and at test time select the samples that look best. Since tuning $\tilde{\sigma}_{s|t}$ may again inject bias, we should pay attention to metrics of distributional coverage as well as purely qualitative measures. In the extreme case, when we choose to set $\sigma_{s|t} = 1$, we remove any dependence of the posterior of $\boldsymbol{z}_s$ on the previous sample $\boldsymbol{z}_t$. Thus, we sample from a series of marginal distributions as is written here:

$$\bar{p}(\boldsymbol{x}, \boldsymbol{z}_{0:1}) = p(\boldsymbol{z}_1) \left[\prod_t \bar{p}_\theta(\boldsymbol{z}_{t-\Delta t} | \hat{E}_\theta(\boldsymbol{z}_t, t)) \right] p_\theta(\boldsymbol{x}|\boldsymbol{z}_0). \tag{101}$$

This process is called a star diffusion because each $\boldsymbol{z}$ depends on the predicted datapoint, forming a star pattern. This method corresponds to the DDIM-style sampling method we presented in the main text. Star sampling may make the problem of sampling from only the modes of $q(\boldsymbol{x})$ described above more pronounced, since the intermediate latent variables now depend only on the collapsed posterior over the embeddings. Additionally, not taking into account any of the direction of the previous sample should make sampling more difficult and potentially introduce bias.

In the discrete-time setting, adaptive timestep solvers are not available as in continuous-time SDEs. Instead, we can approximate adaptivity by using the learned importance sampling weights over t, drawing more samples from regions where the model experienced higher loss during training. This focuses sampling on more challenging regions, potentially improving efficiency and accuracy.

E Implementation Details

We train three model variants: Diffusion-LM, MuLAN, and NFDM. This section describes the used architectures and hyperparameter.

E.1 Architecture

For the predictor $\hat{E}_\theta(\boldsymbol{z}_t, t)$, we follow the parameterization of [18], using BERT-base [7] as the backbone with a dropout rate of 0.1 (approximately 90M parameters). The timestep t is encoded using Fourier embeddings [23] passed through a two-layer MLP with SiLU activations [9]. The resulting time embeddings are added to the token embeddings before being passed to the BERT encoder. We employ a word-level tokenizer, yielding a vocabulary of 10,767 entries.

MuLAN extends this setup by introducing an auxiliary latent variable $\boldsymbol{c}$ that conditions both the predictor and the forward process. The context $\boldsymbol{c}$ is encoded using a smaller BERT model (approximately 10M parameters) with hidden dimension equal to the embedding size H. This encoder is not time-conditioned. The predictor incorporates $\boldsymbol{c}$ by appending it as an additional token, while the forward process employs $\boldsymbol{\gamma}_\varphi(t, \boldsymbol{c})$ as in [27], where polynomial coefficients are predicted by an MLP applied to $\boldsymbol{c}$.

NFDM uses two transformer backbones. The predictor reuses the same BERT-base model as above, while the forward process, which outputs $\overline{\mu}_\varphi(E_\varphi(\boldsymbol{x}), t)$ and $\overline{\sigma}_\varphi(E_\varphi(\boldsymbol{x}), t)$, uses a smaller BERT encoder (approximately 10M parameters). Because this model does not directly observe $\boldsymbol{z}_t$, it may struggle to condition on t. To address this, NFDM employs Adaptive LayerNorm conditioning [23, 24] within the transformer blocks and before the final MLP.

E.2 Training Setup

All models use a batch size of 512, a hidden dimension of $H = 128$, and a sequence length of $S = 64$. Training runs for 800,000 iterations on NVIDIA A100 or H100 GPUs. Diffusion-LM and MuLAN require approximately 70–90 hours on an A100, while NFDM takes roughly 120 hours on an H100 due to the lack of flash attention support in the reverse process, the Jacobian Vector Product (JVP) computation for $\dot{F}_\varphi$ is incompatible with standard flash attention implementations. Overall, the total compute budget amounts to about 1,000 H100 hours and 2,000 A100 hours.

Diffusion-LM and MuLAN are trained using the Adam optimizer [16] with a learning rate linearly decayed from 1×10^{-4} and no weight decay. Gradients are clipped to a norm of 1.5 after 10K steps for the rescaled variants. MuLAN uses $\gamma_{\min} = -10$ and $\gamma_{\max} = 10$, while its rescaled version uses $\gamma_{\min} = -4.6$ and $\gamma_{\max} = 13.8$ to match the Diffusion-LM reference process.

NFDM is optimized jointly with Adam and Muon [15]. Muon is applied to all multi-dimensional weight matrices within the transformer backbones, using a learning rate of 0.002 and momentum of 0.95, with gradient clipping to 0.3 after 10K steps. Adam is applied to all remaining parameters using the same configuration as for the other models.

F Additional results

In this section we provide further results complimenting those in subsection 4.2. We specifically look at modeling ablations for NFDM and MuLAN, and different choices for the sampling procedure.

F.1 Modeling ablations

We investigate whether the specific implementation of time conditioning in the forward process impacts textual quality. We compare NFDM with Adaptive Layernorm conditioning as presented in the main text to NFDM with time conditioning through additive timestep embeddings (NFDM-Additive). Figure 4 shows the cosine similarity for subsequent transformation of the input in the forward process for $t = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$. For NFDM-Additive, we observe that the model learns an input-dependent transformation of the data at $t = 0.1$. However, subsequent transformations do not change with t. NFDM, by contrast, does learn a time-dependent transformation of the input, as shown on the right in Figure 4. Table 3 shows that this results in a large improvement in perplexity for NFDM, although the model performs worse in terms of MAUVE, Diversity, and Memorization. Since NFDM achieves a perplexity similar to that of the data set and does not ignore t, we use this variant of the model for further experiments.

Table 3: Quality of diffusion models with different time conditioning architectures.

Conditioning	PPL ↓	MAUVE ↑	Diversity ↑	Memorization ↓
Timestep Embeddings	44.687 (0.731)	0.688 (0.039)	0.234 (0.002)	0.100 (0.001)
Adaptive LayerNorm	26.435 (0.221)	0.426 (0.032)	0.109 (0.001)	0.143 (0.003)
Dataset	26.106	0.957	0.200	0.141

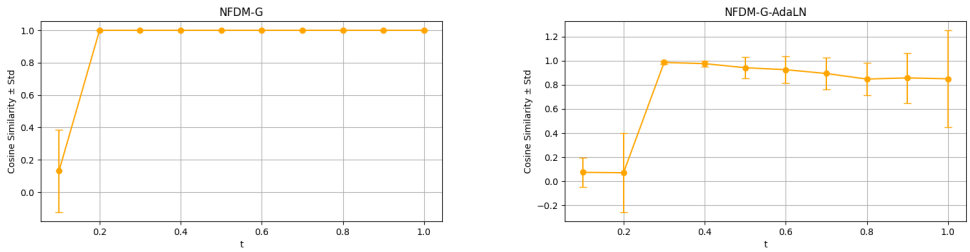

Figure 4: Cosine-similarity between $\overline{\mu}_\varphi(E_\varphi(\boldsymbol{x}), t)$ and $\overline{\mu}_\varphi(E_\varphi(\boldsymbol{x}), t + \Delta t)$, with steps of $t = 0.1$ comparing NFDM-Additive (left) and NFDM (right). Standard deviation computed within batch.

F.2 Sampling ablations

F.2.1 Noise injection: generative conditional Markov chain

We begin with $\tilde{\sigma}_{s|t}$. As noted in subsection D.2, setting $\tilde{\sigma}_{s|t} = \tilde{\sigma}^*_{s|t}$ ensures asymptotic equivalence with the continuous-time formulation. Alternatively, setting $\tilde{\sigma}_{s|t} \in [0, 1]$ can be used as a heuristic that retains the same marginals $q_\varphi(\boldsymbol{z}_t|\boldsymbol{x})$. We experiment with $\tilde{\sigma}_{s|t} \in \{1, 0.8, 0.5, \tilde{\sigma}^*_{s|t}\}$. Here, setting $\tilde{\sigma}_{s|t} = 1$ corresponds to the DDIM-style sampling procedure used in the main text. For

MuLAN-Rescaled, $\tilde{\sigma}^*_{s|t}$ is set according to the global SNR, and thus identical to the one used in Diffusion-LM.

Table 4 shows the results. Across all models, perplexity increases as $\tilde{\sigma}_{s|t}$ rises from 0.5 to 1. For models with a learned forward process, this increase in perplexity is accompanied by reduced diversity and memorization. This suggests that higher $\tilde{\sigma}_{s|t}$ values, which bring the process closer to star sampling, increase bias in the sampling trajectory, causing the models to sample more from high-probability modes. Supporting this, at $\tilde{\sigma}_{s|t} = 1$, all models achieve perplexities below the dataset baseline, indicating overly predictable sequences.

We do not observe a consistent trend in MAUVE. The reason for this could be that MAUVE is based on GPT-2 Large sequence encodings, which it uses to compute distributional coverage. Thus, it benefits from both improved predictability under GPT-2 (PPL) and improved diversity. This leads to a trade-off when the two move in opposite directions.

Comparing fixed $\tilde{\sigma}_{s|t}$ values to the asymptotically correct $\tilde{\sigma}^*_{s|t}$, we find that the latter results in higher diversity and lower memorization, but also higher perplexity and lower MAUVE. This indicates that while the sampling trajectory may be less biased under $\sigma^*_{s|t}$, this leads to a considerable loss in sample quality.

Table 4: Quality of diffusion model samples when varying schedule of $\tilde{\sigma}_{s|t}$

Method	$\sigma_{s\|t}$	PPL ↓	MAUVE ↑	Diversity ↑	Memorization ↓
Diffusion-LM	1.0	21.05 (0.42)	0.72 (0.04)	0.18 (0.00)	0.14 (0.00)
	0.8	21.05 (0.24)	0.76 (0.01)	0.18 (0.00)	0.14 (0.00)
	0.5	21.68 (0.22)	0.74 (0.02)	0.18 (0.00)	0.14 (0.00)
	$\tilde{\sigma}^*_{s\|t}$	64.47 (0.90)	0.68 (0.06)	0.28 (0.00)	0.09 (0.00)
MuLaN-Rescaled	1.0	23.10 (0.20)	0.60 (0.04)	0.18 (0.00)	0.12 (0.00)
	0.8	27.71 (0.31)	0.60 (0.03)	0.20 (0.00)	0.12 (0.00)
	0.5	34.02 (0.26)	0.62 (0.06)	0.22 (0.00)	0.11 (0.00)
	$\tilde{\sigma}^*_{s\|t}$	124.40 (1.60)	0.18 (0.01)	0.36 (0.00)	0.06 (0.00)
NFDM	1.0	23.52 (0.36)	0.42 (0.02)	0.10 (0.00)	0.15 (0.00)
	0.8	26.44 (0.22)	0.43 (0.03)	0.11 (0.00)	0.14 (0.00)
	0.5	30.23 (0.22)	0.50 (0.06)	0.13 (0.00)	0.13 (0.00)
	$\tilde{\sigma}^*_{s\|t}$	-	-	-	-

F.2.2 Noise injection: generative SDE

We now turn to continuous sampling, examining both the stochastic (SDE) and deterministic (ODE) variants, and comparing their performance to discrete-time sampling. These settings are governed by the volatility schedule $g^2(t)$, as discussed in subsection D.1.

Table 5 presents the results. Broadly, sampling from the continuous-time formulation results in worse perplexity compared to discrete sampling. However, NFDM breaks this trend: when sampled via the SDE, it performs on par with or better than Diffusion-LM under $\tilde{\sigma}^*_{s|t}$, achieving similar PPL, higher MAUVE, increased diversity, and lower memorization. Compared to using discrete sampling for NFDM, we see that the generated samples have higher diversity, higher mauve, and lower memorization. This suggests that using the generative SDE can reduce sampling bias, leading to broader distributional coverage and less mode-seeking behavior.

Comparing the SDE to the ODE directly, we find that SDE sampling consistently outperforms its deterministic counterpart for almost all models. The exception is Diffusion-LM. In this case, SDE sampling leads to significantly degraded performance. We attribute this to the design of its volatility schedule $g^2(t)$, which exhibits large spikes near the endpoints ($t = 0$ and $t = 1$), injecting excessive noise late in the trajectory, which leads to poor generation quality.

Table 5: Quality of diffusion model samples when sampled with generative SDE versus generative ODE

Method	$g_\varphi(t)$	PPL ↓	MAUVE ↑	Diversity ↑	Memorization ↓
Diffusion-LM	SDE	2658.17 (21.67)	0.01 (0.00)	0.78 (0.00)	0.00 (0.00)
	ODE	74.07 (1.92)	0.42 (0.02)	0.29 (0.01)	0.09 (0.00)
MuLaN-Rescaled	SDE	116.61 (3.37)	0.20 (0.03)	0.35 (0.01)	0.06 (0.00)
	ODE	145.54 (1.49)	0.09 (0.02)	0.36 (0.00)	0.06 (0.00)
NFDM	SDE	68.33 (1.45)	0.76 (0.05)	0.32 (0.00)	0.07 (0.00)
	ODE	289.14 (6.14)	0.08 (0.02)	0.47 (0.00)	0.05 (0.00)

F.2.3 Sampling steps

In this section, we evaluate the impact of reducing the number of sampling steps to values equal to or below the sequence length. This setting gives the model at most one step per token. We hypothesize that models which learn the forward process, conditionally on time t and/or data $\boldsymbol{x}$, will produce smoother sampling trajectories. Consequently, we expect them to perform better than models with fixed forward processes when operating on a sparse timestep grid.

Table 7 presents the results for 64 and 32 sampling steps with discrete sampling and $\tilde{\sigma}_{s|t} = 1$. We find that Diffusion-LM, which uses a static forward process, handles low-step generation relatively well. In contrast, NFDM struggles significantly, especially at 32 steps, where MAUVE drops to zero and perplexity increases sharply, indicating that many generated sequences become unintelligible. [3] describes a way to straighten the learned trajectories with an additional loss term, which could significantly improve the few-step performance of the models with a learned forward process. We defer this to future work. Table 6 shows that the results for the sampling with generative SDE are much worse than for the discrete procedure, across all models.

These results do not align with the findings of [3], which report that NFDM models outperform fixed-forward process baselines under few-step sampling.

Table 6: Quality of diffusion model samples when decreasing the number of sampling steps (NFE). Samples are obtained using the generative Markov chain

Method	NFE	PPL ↓	MAUVE ↑	Diversity ↑	Memorization ↓
Diffusion-LM	64	29.55 (0.29)	0.81 (0.01)	0.21 (0.00)	0.13 (0.00)
	32	38.23 (0.81)	0.77 (0.03)	0.22 (0.00)	0.12 (0.00)
MuLaN-Rescaled	64	47.68 (0.63)	0.57 (0.04)	0.26 (0.00)	0.09 (0.00)
	32	72.97 (0.82)	0.23 (0.03)	0.28 (0.00)	0.08 (0.00)
NFDM	64	141.87 (2.76)	0.19 (0.01)	0.39 (0.00)	0.05 (0.01)
	32	571.78 (10.90)	0.02 (0.00)	0.41 (0.01)	0.02 (0.00)

Table 7: Quality of diffusion model samples when decreasing the number of sampling steps (NFE). Samples are obtained using the generative SDE

Method	NFE	PPL ↓	MAUVE ↑	Diversity ↑	Memorization ↓
Diffusion-LM	64	3950.27 (24.05)	0.00 (0.00)	0.58 (0.00)	0.00 (0.00)
	32	3707.34 (55.64)	0.00 (0.00)	0.55 (0.01)	0.00 (0.00)
MuLaN-Rescaled	64	156.06 (1.15)	0.10 (0.01)	0.38 (0.00)	0.05 (0.00)
	32	205.21 (2.77)	0.04 (0.00)	0.41 (0.00)	0.04 (0.00)
NFDM	64	7181.29 (167.44)	0.00 (0.00)	0.90 (0.00)	0.00 (0.00)
	32	363.41 (14.76)	0.00 (0.00)	0.38 (0.01)	0.00 (0.00)

F.3 Generated samples

Below, we show two generated samples for each of the three best-performing diffusion models. We use discrete sampling with $\tilde{\sigma}_{s|t} = 1$, for which we achieved the lowest perplexity. The samples displayed here have a whitespace between each of the generated tokens. To evaluate the samples correctly, we post-process them with a simple function that removes trailing whitespace and whitespace between tokens that should be one word.

(a) **Diffusion-LM**

START My friends and I went for a hike in the woods . The weather was very crowded when we got there . When we got there it started to rain . We sat for so long until it started to pour . We had to go back to our car to go out . END PAD PAD PAD PAD PAD PAD PAD PAD"

START Kelly really wanted a new haircut . Luckily she could n't find one she wanted . Thankfully she ordered one . When she got it on it looked great . Kelly was thrilled . END PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD

(b) **MuLAN-Rescaled**

START Kate bought a pair of pretty boots . They were longer than she expected . Because of them were too small for them . Kate had to wear them out the next day . Kate wore them to school again . END PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD

START Last night I made pasta . I used a new recipe . It tasted amazing . It still tasted amazing . I wanted to try it . END PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD

(c) **NFDM**

START Tom needed to get a pair for school . He realized he had to get a new pair of shoes . He realized that the store was too high . He went and saw his shoes were in stock . He had to get a good deal for his shoes . END PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD

START Billy and his mother decided to go to the beach . They got in the car and drove to the beach . The UNK told them they would take this trip . They got to the beach and got a lot of sand . They finally got in the car and drove to the beach . END PAD PAD PAD PAD PAD PAD

Figure 5: Samples generated using discrete sampling with $\tilde{\sigma}_{s|t} = 1$ and 2000 steps.

(a) **Diffusion-LM**

START Jim was on a road trip . He had been driving for a while when the bell rang . He had never been to a hotel . He got into the hotel and a hotel when he got to . He had a great time . END PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD

START John did n't have anything to do with his family . He decided that he should stay home to work alone . He called his boss and offered to help . John accepted the offer . He now has more money at home with his family . END PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD

(b) **MuLAN-Rescaled**

START UNK had been craving pizza lately . She asked her husband if they could eat it . He gave her a box of stale pizza . He told her they were sweet . UNK now eats at fifteen times . END PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD

START I was getting a divorce . My wife wanted me out of her closet every single . It was cold and UNK . I tried to dig on them . My wife . me on a hand and got rid of ! END PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD

(c) **NFDM**

START Rob was wondering Rob Rob could tell Rob was then Rob proposed to race but was determined to rob Rob Rob He decided to give up Rob UNK Rob Rob Rob Finally Rob and afterwards END PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD

START The kids were traveling through China . They got bored until it opened and it was beautiful ! They spent hours on making their own favorite meal . They made so much food ! END PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD PAD

Figure 6: Samples generated using discrete sampling with $\tilde{\sigma}_{s|t} = 1$ and 64 steps.