Flow Matching for Sequence to Sequence to Text Generation

Sheel F. Shah
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213
sheels@cs.cmu.edu

Abstract

Diffusion Models are a generative modeling paradigm that have led to state-of-the-art performance in image generation. Inspired by their success, Diffusion Language Models (DLMs) try to extend diffusion models to discrete modalities such as text and code. This project adapts flow matching to DLMs, in particular, for sequence to sequence text generation in the embedding space. We achieve comparable performance to baselines with as few as just *one* function evaluation, compared to the baseline's 2000 function evaluations. Our code is available at `https://github.com/sheelfshah/FlowMatchingForSeq2SeqTextGeneration`.

1 Introduction

The Diffusion Model paradigm has become the de-facto method for unconditional and conditional generation of images [Rombach et al., 2021], audio [Evans et al., 2024], and video. Diffusion models are trained by gradually adding noise to samples from a target distribution and learning to denoise these noised samples. During inference, the models follow the time-reversed stochastic differential equation (SDE) corresponding to the noising process. Initially introduced in [Sohl-Dickstein et al., 2015] and [Ho et al., 2020], noise is added over $T = 2000$ steps and, consequently, inference of these algorithms requires 2000 function evaluations (NFEs). A large body of recent work focuses on improving the inference efficiency of diffusion models. Flow Matching, introduced in [Lipman et al., 2022], models the reverse process as a continuous time ordinary differential equation (ODE). This enables inference with far fewer NFEs and also benefits from highly optimized ODE solving libraries.

A parallel line of work has focused on extending diffusion to discrete modalities such as text. There are two approaches in the literature — diffusion over discrete Markov chains of the probability mass function [Austin et al., 2021] and diffusion over a continuous representation of the discrete space [Li et al., 2022]. Since diffusion over continuous spaces is amenable to direct transfer of the aforementioned approaches, we focus on continuous diffusion for text.

Existing approaches that perform text diffusion in continuous spaces, such as DiffusionLM [Li et al., 2022] and DiffuSeq [Gong et al., 2023], follow the method of DDPM [Ho et al., 2020]. We propose to extend Flow Matching to the setting of DiffuSeq, and observe gains of **2000x** in inference speed and NFE (from 2000 to just 1), while having performance comparable to that of DiffuSeq. Following DiffuSeq, we work on the Quora Question Pairs [DataCanary et al., 2017] dataset which is a series of duplicate/paraphrased question pairs on Quora evaluated by human labelers. We also use the same set of metrics as DiffuSeq — BLEU score [Papineni et al., 2002] and ROUGE-L score [Lin, 2004]. Because of the limited scope of this project, we do not evaluate our method on other larger datasets considered in DiffuSeq, and also do not calculate the BERTScore metric.

Preprint. Under review.

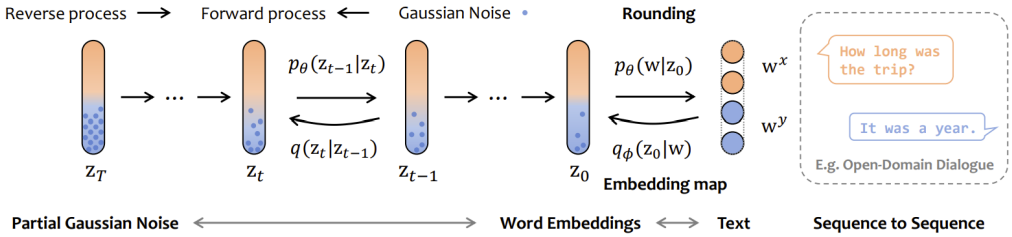

Figure 1: A visualization of the input/output concatenation and the forward and reverse noising process. x_0 corresponds to w^x, and x_1 to w^y. (Figure sourced from Gong et al. [2023])

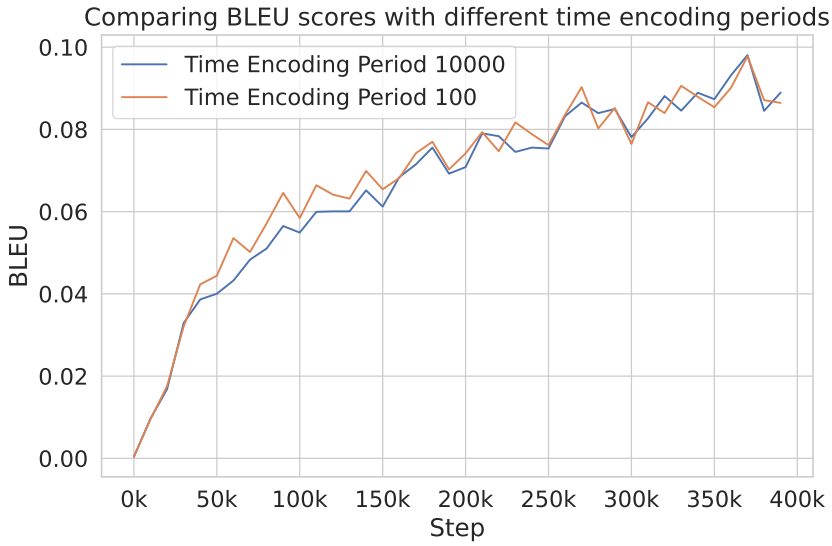

Figure 2: Effect of a different time encoding period in the DiT model

2 Literature Review

Diffusion Models: Diffusion models [Song et al., 2020] are a generative modeling paradigm in which the model learns to generate data by gradually adding noise to a target and learning the denoising function. The target is usually a continuous representation of an image, and the noising method adds Gaussian noise to the target. Due to their ability to circumvent issues such as training instability in GANs [Salimans et al., 2016], these models have been widely adapted in several modalities such as images, audio and video.

Flow matching: Flow matching [Lipman et al., 2022, Albergo et al., 2023] uses a vector field flow that defines a smooth invertible mapping between the target (eg, image) and the prior (eg, pure noise). The model learns this flow and solves an ordinary differential equation (ODE) during inference to transport samples from the prior distribution to the target distribution. Compared to existing score-based stochastic differential equations (SDE), flow matching enables more stable training and faster inference because it can benefit from well-optimized ODE solving libraries.

Discrete Diffusion: DLMs are a new paradigm for generating targets with a discrete modality, such as text. There are two main streams of work — Discrete and Continuous. Discrete DLMs [Sahoo et al., 2025] represent text in the tokens space and learn to generate text by often learning to “unmask" sequences of masked tokens. Continuous DLMs, on the contrary, embed tokens to a continuous space and then perform diffusion in this continuous space similar to classical diffusion models. This enables them to be amenable to direct transfer of several advances in diffusion modeling, and we hence focus this project on Continuous DLMs.

DiffuSeq: Consider samples from a source distribution $x_0 \sim \rho_0$ and a target distribution $x_1 \sim \rho_1$. DiffuSeq considers text sequences, where $x = (x^1, x^2, \cdots x^L); x^i \in [K]$ is a sequence of L tokens from a vocabulary size of K. We have access to a dataset $D = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}) \cdots (x^{(N)}, y^{(N)})\}$ of N source-target pairs. Our goal is to learn a function that takes as input $x_0 \sim \rho_0$ and produces $x_1 \sim \rho_1$. To this end, DiffuSeq casts this as a generation problem and learn a generative model that can generate sample pairs (x, y) from input (x, z), where z is Gaussian noise. They do so by slowly adding noise to the sample pair and then learning a denoising function parametrized by a neural network. As visualized in Fig. 1, the noise corruption only acts on the latter half of the pair (i.e. y), and this allows the model to condition its generation on the input x. By jointly learning an embedding function and a DDPM based denoiser, DiffuSeq is able to achieve state-of-the-art results in several sequence to sequence tasks. However, their method requires 2000 inference steps and is much slower than their autoregressive counterparts.

3 Methods

Data processing. Data are tokenized through Byte-Pair Encoding, and sequences are truncated/padded to a fixed length of 128. Only the 147K positive pairs from QQP are chosen for training. While DiffuSeq concatenates source and target encodings before padding the rest of the sequence, we pad both the source and the target sentences to length 64 each before concatenation. This affixes the noised portion of the string, and avoids the complicated masking procedure employed by DiffuSeq.

Model. We adopt the Diffusion Transformer (DiT) [Peebles and Xie, 2022] architecture for our training. Compared to the BERT Encoder [Devlin et al., 2019] architecture used by DiffuSeq, we observed that DiT enabled a more natural way of encoding both time and position into the model. Since our time range is continuous in $[0, 1]$ instead of discrete in $[2000]$, we use a time encoding period of 100 instead of 10000. The effect of a different time encoding period is shown in Fig. 2. For ablations, we use a DiT architecture with $4M$ parameters — 4 attention heads in 4 layers, with a hidden dimension of 256. For our final model, we use a DiT architecture with $60M$ parameters — 8 attention heads in 16 layers, with a hidden dimension of 512. Note that this is roughly half the size of the BERT Encoder used by DiffuSeq ($110M$ parameters).

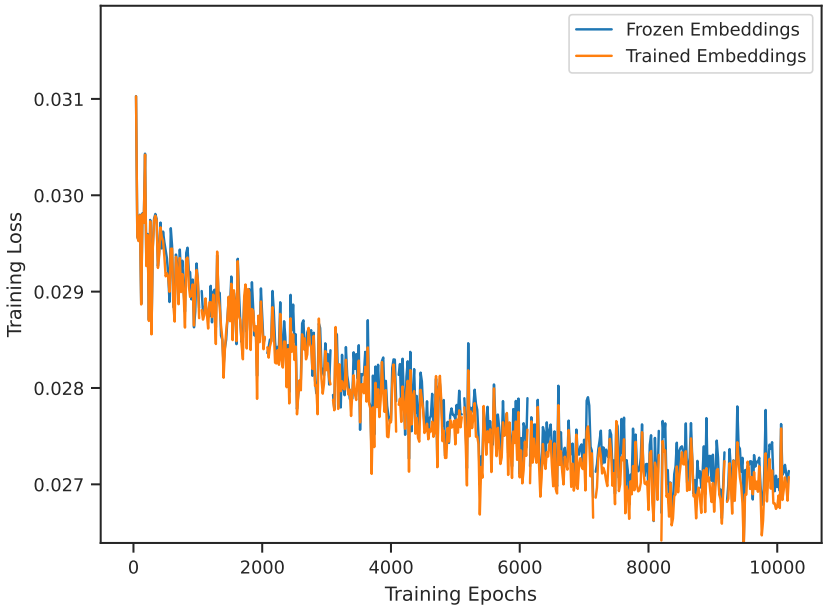

Figure 3: Effect of freezing the embedding layer on continued training from a pre-trained checkpoint is negligible.

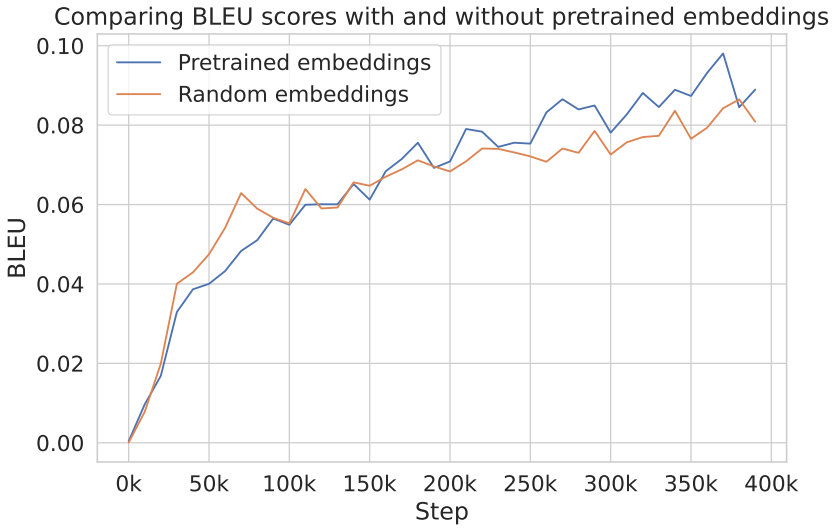

Figure 4: Effect of using random embeddings instead of pre-trained embeddings.

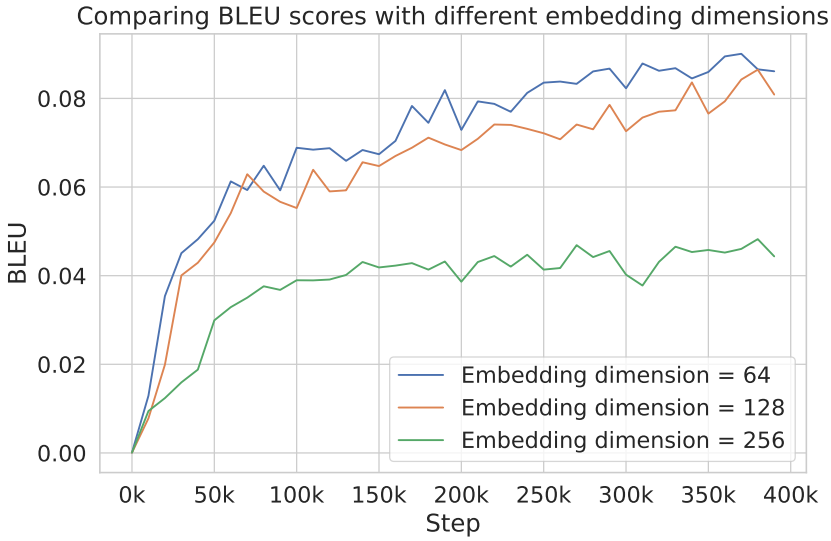

Figure 5: Effect of embedding dimension on performance.

Embeddings. To be able to represent text in continuous space, DiffuSeq proposes to learn an embedding $\text{EMB}(x^i) : [K] \to \mathrm{R}^d$ ($d = 128$) that is applied to each token of the source and target samples. They use a nearest neighbor decoder to round embeddings back to tokens. We use the pretrained embeddings learnt by DiffuSeq, but treat them as a fixed matrix in our algorithm. The effect of continued training of the embedding layer in DiffuSeq is shown in Fig. 3. We analyze the effect of using these pretrained embeddings versus random embeddings in Fig. 4. We also inspect the effect of the dimension of the embeddings in Fig. 5 using random embeddings.

Training. Following [Lipman et al., 2022], we use the standard l_2 loss to train our model to predict the velocity of the linearly interpolated sample. Unlike DiffuSeq, this is a simpler loss that does not lead to any training instability and also does not need any importance sampling [Nichol and Dhariwal, 2021] or regularization. Our training algorithm is described in Algorithm 1. We also maintain an exponential moving average (EMA) of the model parameters. The EMA parameters significantly outperform the actual parameters. As demonstrated in Fig. 6, using an EMA with $\lambda = 0.9999$ produces the best results, and hence we use the EMA parameters for our final model.

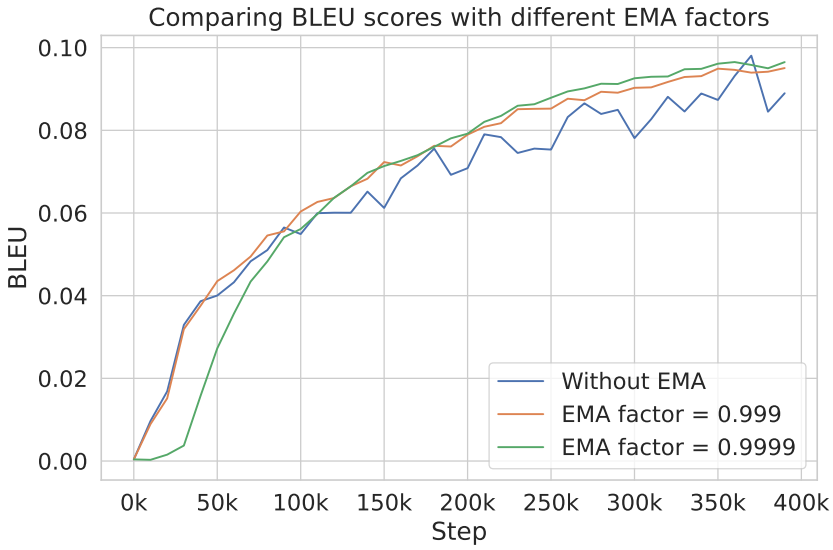

Figure 6: Using an exponential moving average of the training parameters greatly boosts performance.

Implementation. We write all our training code in JAX for faster training. The data processing and evaluation pipelines are adapted from DiffuSeq with major revamps for speed and clarity. Our code, training scripts, and checkpoints for the final model and all ablations are shared at `https://github.com/sheelfshah/FlowMatchingForSeq2SeqTextGeneration`. We train our final model for $500k$ steps on a single NVIDIA A100 80GB GPU for 24 hours.

Algorithm 1 Flow Matching For Sequence to Sequence Generation

Input: Dataset $D = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}) \cdots (x^{(N)}, y^{(N)})\}$, pre-trained embedding $\text{EMB}(x^i) : [K] \to \mathrm{R}^d$, model f_θ, number of training steps T, learning rate η
Output: Trained parameters θ
for $i = 1$ to T **do**
  Sample $x, y \sim D$
  Sample $z \sim \mathcal{N}(0, \mathbf{I})$
  Sample $t \sim \mathcal{U}(0, 1)$
  Create pairings $x_0 = \text{Concat}(x, z)$, $x_1 = \text{Concat}(x, y)$
  Interpolate: $x_t = (1 - t)x_0 + tx_1$
  Target velocity: $v^\star(x_t, t) = x_1 - x_0$
  Loss:

$$\mathcal{L} = \|v_\theta(x_t, t) - v^\star(x_t, t)\|^2$$

  Update: $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$
end for

4 Results

Fig. 7 shows the evaluation of our method against DiffuSeq on unseen data. Flow matching achieves performance comparable to that of DiffuSeq with far fewer inference steps. Interestingly, Flow Matching has high BLEU and ROUGE-L scores with just **one inference step**. Table 1 shows some sample outputs for DiffuSeq and our method.

5 Discussion and Analysis

Flow Matching performed surprisingly well for sequence to sequence generation. The performance of the model increased steadily with more training time and larger sizes. Observing such high scores

Table 1: Sample Responses

Question Prompt	Inference Steps	Method	Duplicate Question Response
how can i become more fluent in chinese?	2000	DiffuSeq	how can i become more speaking in chinese?
		Flow Matching	how do i get fluent of chinese?
	100	DiffuSeq	russo fleetingptonzuki eileen [unused827] vilec sinister on ...
		Flow Matching	how do i get fluent of chinese?
	1	**Flow Matching**	how do i become fluent in chinese?
how do you get deleted instagram chats?	2000	DiffuSeq	how do i recover deleted completely instagram account?
		Flow Matching	how can i get back my instagram deleted d messages?
	100	DiffuSeq	programming sea side and zulu ...
		Flow Matching	how can i get back my instagram deleted d messages?
	1	**Flow Matching**	how can i get deleted instagramtagram deleted?

with only 1 NFE was unexpected — we expected performance to drop sharply with fewer NFEs, just as it did with DiffuSeq. But the drop was not as steep as anticipated and one-step predictions did not follow the trend.

In order to inspect this behavior, we plot the loss of the model against t in Fig. 8. This plot reveals that all of the loss is concentrated near $t = 0$. We believe that this is due to the model "memorizing" tokens and being able to implicitly identify both x_0 and x_1 given the interpolated x_t for larger t. At smaller t, the noise term dominates the actual token and makes it difficult for the model to predict it directly. This means that the learned velocity at $t = 0$ will be highly inaccurate, whereas the learned velocity at, say, $t = 0.5$ will be overfit to the actual velocity $x_1 - x_0$. When taking just one step, the model jumps directly to $\hat{x}_1$. However, for two steps, the model first jumps to $\hat{x}_{0.5}$, which is quite different from $x_{0.5}$. Because the model has learned to overfit for $t = 0.5$, it performs poorly on an out-of-distribution sample $\hat{x}_{0.5}$. For higher inference steps, each step is small enough for the model to stay close to the distribution it was trained on, i.e. $\hat{x}_t \approx x_t$.

We also note that using an EMA of the parameters has a significant boost on performance. Since each step minimizes the loss of a uniform sample $\|f_\theta(x_t, t) - (x_1 - x_0)\|_2^2$ that represents the integral $\int_{t=0}^{1} \|f_\theta(x_t, t) - (x_1 - x_0)\|_2^2$, we hypothesize that the EMA is able to reduce the variance due to this sampling.

The strengths of our approach are the following.

1. Training is straightforward. We are able to circumvent the need for complicated loss functions and training procedure hacks such as importance sampling.
2. Inference is significantly faster. The learned flow can be integrated with 200 NFEs, beyond which there are no significant gains in performance. This is **10x** faster than DiffuSeq. Moreover, NFEs can be reduced to as few as just **one** with minimal loss in performance.

There are, however, several limitations to our approach as well.

1. Inability to jointly learn the embeddings. Learning embeddings in conjuction with the flow is a direction that we have not explored in this project. As seen in Fig. 4, the use of the right embeddings can have a significant impact on performance.
2. Problem formulation. The proposed diffusion process is only applicable to sequence to sequence tasks. A more general problem would be the unconditional generation of text. Methods for guiding the diffusion process would also be helpful.

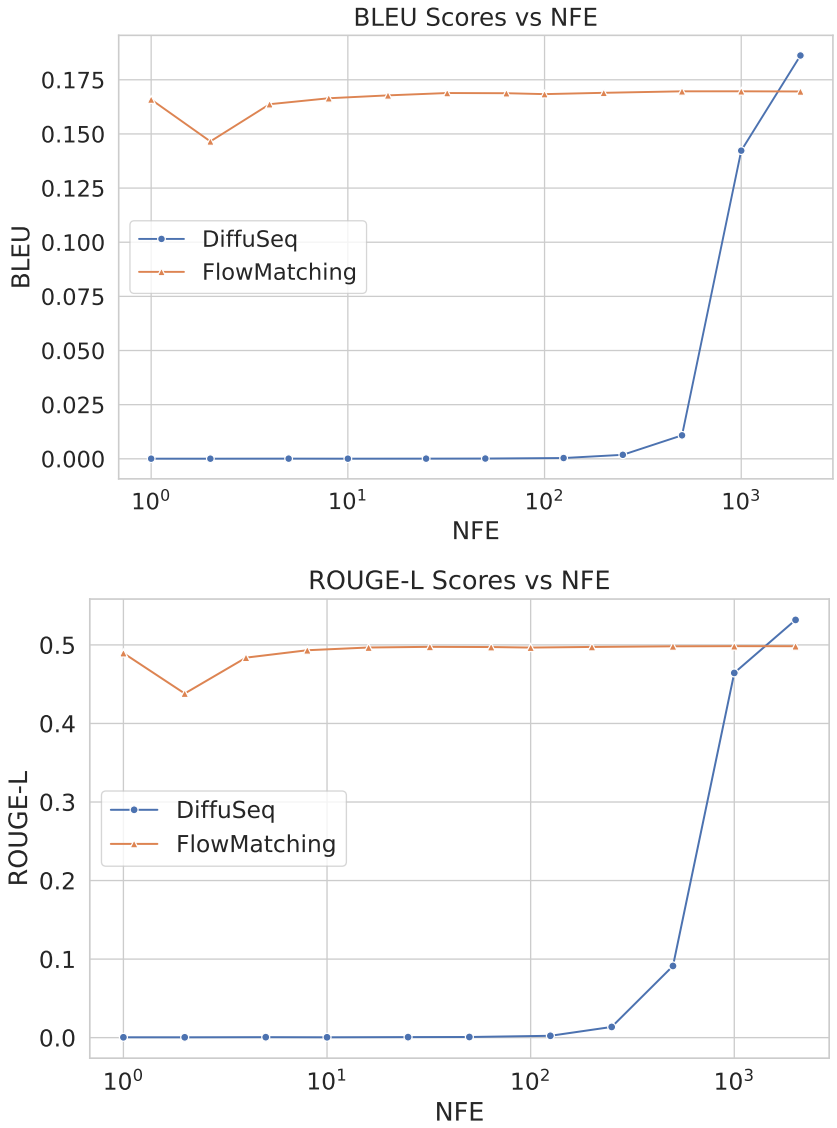

Figure 7: Comparison of BLEU and ROUGE-L scores of DiffuSeq to that of our method

3. Fixed length generation. Extending this method to generate sequences of arbitrary length is not trivial.

4. Evaluation on larger and more challenging datasets. While it is straightforward to extend our method to other datasets, we have not explored this direction.

We note that during the final stages of our research, we became aware of the work by [Hu et al., 2024] which also addresses Flow Matching for sequence to sequence generation. Despite the existence of concurrent work, the development of our distinct data processing pipeline, model architecture, frozen embedding training, and code implementation offers a unique and independently validated perspective on this problem. We also note that the method in [Hu et al., 2024] supports generation with just one step, which is an artifact of the anchor loss in their training, whereas our procedure allows for any-step inference, which is in line with the Flow Matching paradigm. We also include novel analysis towards understanding the model's performance at different NFEs, use EMAs for better performance, and inspect the effect of embeddings in depth.

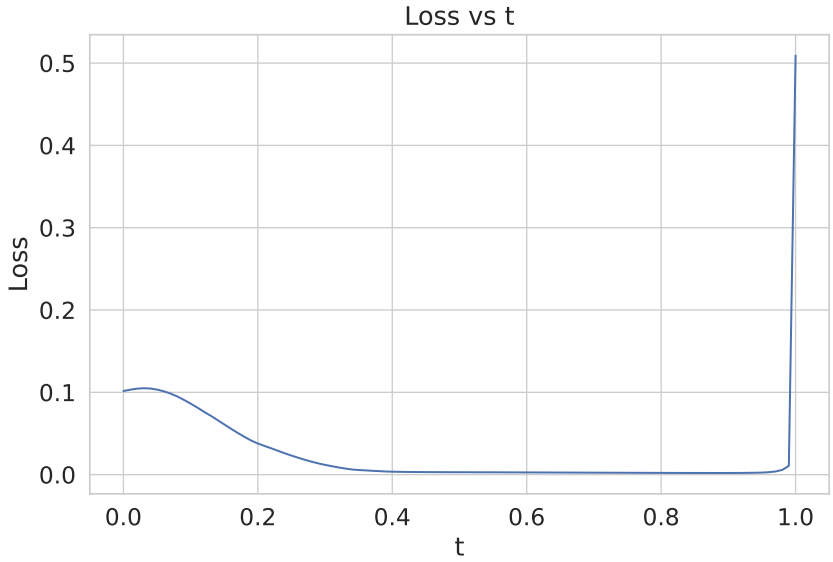

Figure 8: l_2 loss of the model on validation data plotted across t in $[0, 1]$

References

Michael S. Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *ArXiv*, abs/2303.08797, 2023. URL `https://api.semanticscholar.org/CorpusID:257532329`.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, 2021.

DataCanary, hilfialkaff, Lili Jiang, Meg Risdal, Nikhil Dandekar, and tomtung. Quora question pairs. `https://kaggle.com/competitions/quora-question-pairs`, 2017. Kaggle.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019. URL `https://api.semanticscholar.org/CorpusID:52967399`.

Zach Evans, CJ Carr, Josiah Taylor, Scott H. Hawley, and Jordi Pons. Fast timing-conditioned latent audio diffusion. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. DiffuSeq: Sequence to sequence text generation with diffusion models. In *International Conference on Learning Representations, ICLR*, 2023.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arxiv:2006.11239*, 2020.

Vincent Tao Hu, Di Wu, Yuki M Asano, Pascal Mettes, Basura Fernando, Björn Ommer, and Cees G M Snoek. Flow matching for conditional text generation in a few sampling steps. In *EACL*, 2024.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-lm improves controllable text generation. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL `https://aclanthology.org/W04-1013/`.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *ArXiv*, abs/2210.02747, 2022. URL `https://api.semanticscholar.org/CorpusID:252734897`.

Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. *ArXiv*, abs/2102.09672, 2021. URL https://api.semanticscholar.org/CorpusID:231979499.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Pierre Isabelle, Eugene Charniak, and Dekang Lin, editors, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040/.

William Peebles and Saining Xie. Scalable diffusion models with transformers. *arXiv preprint arXiv:2212.09748*, 2022.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2025. Curran Associates Inc. ISBN 9798331314385.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 2234–2242, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 2256–2265. JMLR.org, 2015.

Yang Song, Jascha Narain Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *ArXiv*, abs/2011.13456, 2020. URL https://api.semanticscholar.org/CorpusID:227209335.