Accelerated sampling with few-step diffusion

Vedant Puri
Mechanical Engineering
Carnegie Mellon University
vedantpuri@cmu.edu

1 Introduction

Diffusion-based generative models have become popular for image generation [8], audio generation [12], and video generation [9] due to their ability to generate high resolution media that feels realistic and adheres to a given prompt. These models generate media samples from randomly sampled Gaussian noise by solving a probability flow differential equation in reverse [22]. This involves iterating through tens to hundreds of denoising steps where a large neural network model is queried to guide the sample towards the final image. The resultant inference latency and high training cost constitute a major impediment to the widespread adoption of this technology. Due to the commercial applicability of diffusion models, the problem of few-step-generation has garnered significant interest.

In this project, we build upon two recent works to produce diffusion models that produce accurate samples with only a few denoising steps: first, Frans et al. [2] proposed *shortcut* diffusion models that allows for taking arbitrary-sized denoising steps in the context of *flow matching* [15] diffusion formulation; second, Lu and Song [17] proposed the *trig flow* formulation of the diffusion process in relation to *consistency models*[23]. Our proposed method redefines shortcut diffusion models in the context of trig flow by integrating a trigonometric noise schedule. Specifically, our key contributions are

- Formulation of shortcut models in context of trig flow
- Development of a training objective and first-order sampler for trig flow
- Development of a training objective for trig flow for shortcut models

Our aim is to achieve high quality sample generation with significantly fewer denoising steps, addressing the inefficiencies of standard approaches. Our modifications will be evaluated along with the baseline model for unconditional class generation. The code for this project is publicly available at `https://github.com/vpuri3/FastDiffusion.py`.

2 Dataset and Task

Our hypothesis in this work is that the combination of trig flow and shortcut modeling would lead to superior sample quality for few-step image generation. As such, we test four different diffusion models

1. Flow matching: vanilla diffusion model in the flow matching framework
2. Trig flow: vanilla diffusion model in the trig flow framework
3. Flow matching + shortcut: Shortcut diffusion model in the flow matching framework
4. Trig flow + shortcut: Shortcut diffusion model in the trig flow framework.

We employ the AFHQ Cats dataset [18], resized to a consistent 64x64 resolution, as our primary experimental domain for unconditional generative modeling. This dataset was meticulously selected for its distinctive characteristics: a manageable size that facilitates rapid experimentation, high-quality

image content that ensures meaningful results, and a focused domain that allows for precise model evaluation. As diffusion formulations are largely uncoupled from the neural network architecture, we employ the sample network architecture (UNet [10] with 9 m) for all cases. A relatively small UNet was chosen over a much larger diffusion transformer [19] (DiT) due to (1) the proven reliability of UNets in diffusion modeling, (2) slow convergence speed of DiT, and (3) limited compute budget for this project. Once trained, the models will be compared in sample quality based upon the *Frechet Inception Distance* [7] (FID), a commonly used metric for assessing generative image models, for a range of generation steps.

3 Related work

The community has made significant gains in accelerating the diffusion model sampling process without sacrificing sample quality Salimans and Ho [20], Liu et al. [15, 16], Frans et al. [2], Song et al. [23]. Most popular approaches employ *distillation*, a process of creating more efficient models that generate high-quality samples in fewer steps, based on a pre-trained diffusion model that take many more steps Salimans and Ho [20]. Although progressive distillation can significantly reduce the inference latency, the training process itself becomes much more expensive as a result of sequential training of multiple models. Another approach, called *rectified flow*, proposed by Liu et al. [15] iteratively learns straight ODE paths that are better suited to few-step generation. However, learning smooth ODE paths often requires a multistage training approach leading to the ballooning of computational costs.

Recent advances have focused on simplifying and improving these approaches. Song et al. [23] proposed consistency models that learn to directly map noise to data, designed to be self-consistent by mapping all latent image tensors along the same diffusion path to the same initial noiseless image Lu and Song [17]. Geng et al. [3] introduced a simplified consistency model training framework that reduces computational complexity while maintaining generation quality. Liu et al. [16] demonstrated with Instaflow that high-quality text-to-image generation is possible in a single step, marking a significant advancement in inference speed. Hang and Gu [4] explored improved noise scheduling techniques, proposing nonlinear noise distribution approaches to optimize signal-to-noise ratios during generation.

The field continues to evolve rapidly with innovative approaches. Wang et al. [24] challenges the conventional wisdom about path straightness in rectified flow while Huang et al. [11] flow generator matching framework presents a novel approach to generative modeling. He et al. [5] consistency diffusion bridge models approach addresses novel modeling techniques, while Xue [25] proposed optimized time-step selection methods. These approaches, along with Lee et al. [13] improvements in rectified flow training and Lu and Song [17] work on scaling continuous-time consistency models, demonstrate various strategies for achieving faster generation. However, standalone implementations of these methods often require complex hyperparameter tuning and significantly more computational resources than traditional diffusion models Liu et al. [16].

4 Methods

4.1 Flow matching formulation of diffusion

Diffusion models generate samples by learning ordinary differential equations (ODE) that transform noise into data. The flow matching formulation considers

$$x_t = (1-t)x_0 + tx_1,\ t \in [0,1], \tag{1}$$

as a linear interpolation between a data point $x_1 \sim \mathcal{D}$ and a noise point $x_0 \sim \mathcal{N}(0, I)$ of the same dimension [2, 15]. Flow matching learn a neural network to estimate the expected velocity $v_t = \mathbb{E}[v_t|x_t, t]$ by minimizing the objective

$$\mathcal{L}^{FM}(\theta) = \mathbb{E}_{x_0,x_1,t}\left[\|v_\theta(x_t,t) - \overline{v}_{FM}(x_0,x_1,t)\|_2^2\right] \tag{2}$$

where $\overline{v}_{FM}(x_0, x_1, t) := x_1 - x_0$. To sample from this model, we first sample $x_1 \sim \mathcal{N}(0, I)$ and then solve the denoising ODE defined by the learned velocity model v_θ, which often requires

hundreds of steps to generate high-quality samples. The first order sampling scheme is as follows We define the first order sampling scheme $s_{\text{FM}}(x_t, v_t, t, \Delta t) := x_{t+\Delta t}$ for flow matching as

$$x_{t+\Delta t} = (1 - (t + \Delta t))x_0 + (t + \Delta t)x_1 \tag{3}$$
$$= (1 - t)x_0 + tx_1 + \Delta t(x_1 - x_0) \tag{4}$$
$$= x_t + \Delta t\, v(t, x_t) \tag{5}$$

where Δt is the intended step size.

4.2 Shortcut diffusion models

Frans et al. [2] proposed *shortcut models*, a class of generative models that can generate samples with any number of denoising steps. The key innovation in this method is to condition the velocity model not only on the current noise level t but also on the size of the denoising step Δt, $v_t = \mathbb{E}[v_t|x_t, t, \Delta t]$. As naively taking large sampling steps in a flow model leads to large discretization errors, conditioning on Δt allows the model to account for curvature along the path and jump to the appropriate point. In order to regress v_θ, Frans et al. [2] leverages the inherent consistency property that one step of size Δt should equal two steps of size $\Delta t/2$. That is,

$$v_\theta(x_t, t, 2\Delta t) = \frac{v_\theta(x_t, t, \Delta t) + v_\theta(x_{t+\Delta t}, t, \Delta t)}{2}. \tag{6}$$

The training objective thus becomes

$$\mathcal{L}^{SM}(\theta) = \mathbb{E}_{x_0, x_1, t, \Delta t}\left[\underbrace{\|v_\theta(x_t, t, 0) - \overline{v}_{FM}(x_0, x_1, t)\|_2^2}_{\text{flow matching loss}} + \underbrace{\|v_\theta(x_t, t, 2\Delta t) - \texttt{stopgrad}(v_{\text{target}})\|_2^2}_{\text{consistency loss}}\right] \tag{7}$$

where $v_{\text{target}} = \dfrac{v_\theta(x_t, t, \Delta t) + v_\theta(x_{t+\Delta t}, t, \Delta t)}{2}$.

4.3 Trig flow formulation of diffusion

Trig flow is similar to flow matching in that both methods regress the velocity field v_t with neural networks. While flow matching considers a linear interpolation between $x_0 \sim \mathcal{N}(0, I)$ and $x_1 \sim \mathcal{D}$ (which induces straight paths), trig flow considers a trigonometric interpolation Lu and Song [17]

$$x_t = \cos\Big(\frac{\pi}{2}t\Big)x_0 + \sin\Big(\frac{\pi}{2}t\Big)x_1,\ t \in [0, 1], \tag{8}$$

that leads to curved paths. The corresponding loss objective for regressing v_t is

$$\mathcal{L}^{TF}(\theta) = \mathbb{E}_{x_0, x_1, t}\left[\|v_\theta(x_t, t) - \overline{v}_{TF}(x_0, x_1, t)\|_2^2\right]. \tag{9}$$

where

$$\overline{v}_{TF}(x_0, x_1, t) = \frac{\pi}{2}(\cos\Big(\frac{\pi}{2}t\Big)x_1 - \sin\Big(\frac{\pi}{2}t\Big)x_0). \tag{10}$$

The corresponding first-order sampling scheme $s_{\text{TM}}(x_t, v_t, t, \Delta t) := x_{t+\Delta t}$ is

$$x_{t+\Delta t} = \cos\Big(\frac{\pi}{2}(t + \Delta t)\Big)x_0 + \sin\Big(\frac{\pi}{2}(t + \Delta t)\Big)x_1 \tag{11}$$
$$= \Big[\cos\Big(\frac{\pi}{2}t\Big)\cos\Big(\frac{\pi}{2}\Delta t\Big) - \sin\Big(\frac{\pi}{2}t\Big)\sin\Big(\frac{\pi}{2}\Delta t\Big)\Big]x_0 + \tag{12}$$
$$\Big[\sin\Big(\frac{\pi}{2}t\Big)\cos\Big(\frac{\pi}{2}\Delta t\Big) + \cos\Big(\frac{\pi}{2}t\Big)\sin\Big(\frac{\pi}{2}\Delta t\Big)\Big]x_1 \tag{13}$$
$$= \cos\Big(\frac{\pi}{2}\Delta t\Big)x_t\left[\cos\Big(\frac{\pi}{2}t\Big)x_0 + \sin\Big(\frac{\pi}{2}t\Big)x_1\right] + \tag{14}$$
$$\sin\Big(\frac{\pi}{2}\Delta t\Big)x_t\left[\cos\Big(\frac{\pi}{2}t\Big)x_1 - \sin\Big(\frac{\pi}{2}t\Big)x_0\right] \tag{15}$$
$$= \cos\Big(\frac{\pi}{2}\Delta t\Big)x_t + \frac{\sin\big(\frac{\pi}{2}\Delta t\big)}{\pi/2}v_t \tag{16}$$

Table 1: Comparison of FID Scores for different number of denoising steps:

Diffusion method \ Steps	**1**	**2**	**4**	**8**	**16**	**32**
Flow matching [15]	331	124	77	83	88	90
Trig flow [17]	322	125	70	66	73	82
Flow matching + shortcut [2]	77	70	69	68	69	72
Trig flow + shortcut (ours)	**69**	**64**	**65**	**64**	**64**	**66**

4.4 Trig flow formulation of shortcut models

Shortcut model has only been developed in context of the flow matching formulation in Frans et al. [2]. In order to formulate shortcut model in the context of the trig flow formulation, we not only have to switch out the flow matching velocity objective $\overline{v}_{FM}$ for the trig flow velocity $\overline{v}_{TF}$, but also redefine the consistency loss term for trig flow. We restate the consistency property as the notion that one step of size $2\Delta t$ should have the same effect as two successive steps of size Δt. That is,

$$s_{TF}(x_t, v_t, t, 2\Delta t) = s_{TF}(x_{t+\Delta t}, v_{t+\Delta t}, t+\Delta t, \Delta t) \tag{17}$$

where $x_{t+\Delta t}$ is obtained by querying s_{TF} at x_t and $v_{t+\Delta t}$ is the corresponding velocity. The loss objective is thus

$$\mathcal{L}^{TS}(\theta) = \mathbb{E}_{x_0, x_1, t, \Delta t}[\underbrace{\|v_\theta(x_t, t, 0) - \overline{v}_{TF}(x_0, x_1, t)\|_2^2}_{\text{trig flow loss}} + \tag{18}$$

$$\underbrace{\|s_{TF}(x_t, v_\theta(x_t, t, 2\Delta t), t, 2\Delta t) - \texttt{stopgrad}(\overline{x}_{t+2\Delta t})\|_2^2}_{\text{consistency loss}}] \tag{19}$$

where

$$\overline{x}_{t+\Delta t} = s_{TF}(x_t, v_\theta(x_t, t, \Delta t), t, \Delta t) \tag{20}$$

$$\overline{x}_{t+2\Delta t} = s_{TF}(\overline{x}_{t+\Delta t}, v_\theta(\overline{x}_{t+\Delta t}, t+\Delta t, \Delta t), t, \Delta t) \tag{21}$$

Note that this reformulation of shortcut model is not specific to trig flow, and can be adapted to any flow-based diffusion method.

5 Experiments

Our hypothesis in this work is that the combination of trig flow and shortcut modeling would lead to superior sample quality for few-step image generation. As outlined in Section 2, we test four different diffusion models, flow matching, trig flow, flow matching + shortcut, trig flow + shortcut for few-step unconditional sample generation. The models, each with an underlying 9 m parameter UNet, are trained on the AFHQ Cats dataset for unconditional generation. This includes the necessary architecture modifications, diffusion formulation, and sampling scheme detailed in Section 4. We evaluate FID scores for each model for differing number of sampling steps in Table 1. A corresponding set of generated samples are presented in Figure 1.

In this study, we define 'few-step generation' as $N = 1, 2, 4$ denoising steps. Both flow matching and trig flow perform poorly for $N = 1, 2$ steps. Flow matching and trig flow achieve optimal performance for $N = 4$ and $N = 8$ steps respectively. While trig flow performs better than flow matching for $N \geq 4$ steps, it does not show significant improvement for $N = 1, 2$ steps. Both shortcut diffusion methods perform significantly better than vanilla diffusion methods for few-step generation. Furthermore, their performance does not deteriorate for larger N. Our proposed method, trig flow + shortcut, improves performance over flow matching + shortcut for all step sizes.

6 Code Overview

The core functionality of the implementation is encapsulated in the `Diffusion` class found in `./fastdiff/diffusion.py`, which supports both vanilla diffusion models and shortcut models. `Diffusion` implements `self.noise`, `self.noise_like` to generate samples from $\mathcal{N}(0, I)$. A key

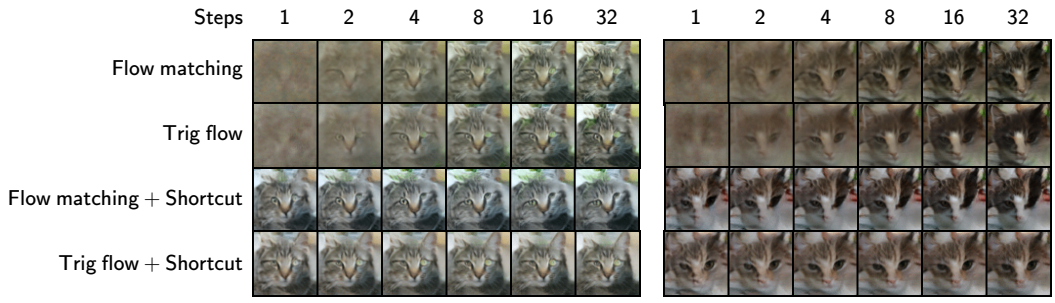

Figure 1: Comparison of FID scores for different number of denoising steps

Table 2: Time spent on various tasks for the project

Activity	**Time Spent (hours)**
Reading papers on diffusion models	10
Defining project deliverables, hypothesis and goals	4
Understanding code from existing implementation	10
Implementing flow matching and shortcut models	20
Implementing noise schedule modifications	9
Running experiments and collecting results	20
Implementing trig flow and shortcut + trig flow	12
Report writing	15
Miscellaneous (meetings, coordination)	7
Total	107

component of `Diffusion` is `diffusion.schedule` which is either `LinearSchedule`, corresponding to flow matching, or `TrigSchedule`, corresponding to trig flow (both are defined in `./fast-diff/diffusion.py`). When called like `schedule(t)`, the schedule returns $(\alpha(t), \beta(t))$ where $x_t = \alpha(t)x_0 + \beta(t)x_1$ for its respective diffusion formulation. For example, $(\alpha(t), \beta(t)) = (1-t, t)$ for `LinearSchedule`. Similarly, `schedule.derv(t)` returns $(\alpha'(t), \beta'(t))$, which is useful for computing the velocity $\overline{v}(x_0, x_1, t) = \alpha'(t)x_0 + \beta'(t)x_1$.

The training objectives for vanilla diffusion models is implemented in `Diffusion` in `self.loss_flow`. The function relies on `self.schedule` to switch between the flow matching objective or trig flow objective. For the case of shortcut models (shortcut + flow matching, shortcut + trig flow), the loss computation is implemented in `self.loss_SM`. The function calls splits the input in two and evaluates vanilla diffusion loss (flow matching loss / trig flow loss) on one chunk with `self.loss_flow`, and consistency loss on the other with `self.loss_consistency`. `self.loss_consistency` takes advantage of the fact that the consistency loss in Section 4.4 can be implemented independent in a uniform way for all flow based methods. As such, `self.loss_consistency` relies on the schedule to query $\overline{x}_{t+\Delta t}$ and $\overline{x}_{t+2\Delta t}$ with `self.schedule.next`.

Once the `Diffusion` model is trained, it can be sampled with `self.sample` which relies on the schedule for calculating $s(x_t, v_t, t, \Delta t)$ with `schedule.next`.

7 Timeline

The timeline for our project work is given in Table 2

8 Research Log

We began by reviewing recent work on diffusion modeling targeted at few step generation such as flow matching [15], shortcut models [2], and consistency models [23]. As flow matching is a well established technique, we considered it as a baseline diffusion model, and then attempted modifications on the others to improve sample quality for few-step generation. Following Frans

et al. [2], our initial plan was to validate our flow matching and shortcut model implementation on a diffusion transformer architecture with 100 m parameters. However, due to limited computational resources, switched to a smaller UNet model with 9 m parameters. As a consequence of differing neural architectures, we engaged in arduous hyperparameter tuning for finding the best training strategy.

Our initial strategy was to apply different noise schedules to flow matching and shortcut models at inference time. We believed that a linear schedule may not be the optimal target for flow matching as it may not allocate noise optimally for different data distributions [4]. We experimented with three noise schedules - cosine, exponential and quadratic, with the goal of improving the performance of shortcut models. However, our experiments with switching out noise schedules at inference time did not improve performance over the baseline.

The drastic failure of inference-time approaches led us to consider deeper integration of the noise schedule in the diffusion process. Following Lu and Song [17], we implemented trig flow for a baseline diffusion model. This involed deriving the appropriate loss objective and sampling scheme for the method. As the results were positive, we then formulated shortcut models in context of trig flow. The critical piece was to redefine the consistency loss to work for flow-based methods with arbitrary schedules.

9 Conclusion

This project explored combining trigonometric flow and shortcut modeling to improve few-step generation in diffusion models. By integrating a trigonometric noise schedule and developing tailored training objectives, we achieved high quality image generation with fewer denoising steps. We have derived the training objective and sampling scheme for trig flow and formulated shortcut diffusion models in context of trig flow. Future work may consider extending the shortcut modeling to *continuous* consistency models Lu and Song [17].

10 Acknowledgments

This was a course project for CMU 10-623 Generative AI course with teammates Nihali Shetty and Nikhitha Beedala during the Fall 2024 semester.

References

[1] A. Blattmann, R. Rombach, K. Oktay, and B. Ommer, "Retrieval-augmented diffusion models," 2022. [Online]. Available: https://arxiv.org/abs/2204.11824

[2] K. Frans, D. Hafner, S. Levine, and P. Abbeel, "One step diffusion via shortcut models," *arXiv preprint arXiv:2410.12557*, 2024.

[3] Z. Geng, A. Pokle, W. Luo, J. Lin, and J. Z. Kolter, "Consistency models made easy," *arXiv preprint arXiv:2406.14548*, 2024.

[4] T. Hang and S. Gu, "Improved noise schedule for diffusion training," *arXiv preprint arXiv:2407.03297*, 2024.

[5] G. He, K. Zheng, J. Chen, F. Bao, and J. Zhu, "Consistency diffusion bridge models," *arXiv preprint arXiv:2410.22637*, 2024. [Online]. Available: https://arxiv.org/abs/2410.22637

[6] E. Heitz, L. Belcour, and T. Chambon, "Iterative α-(de) blending: A minimalist deterministic diffusion model," in *ACM SIGGRAPH 2023 Conference Proceedings*, 2023, pp. 1–8.

[7] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in neural information processing systems*, vol. 30, 2017.

[8] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *arXiv preprint arXiv:2006.11239*, 2020.

[9] J. Ho, C. Saharia, W. Chan, D. J. Fleet, M. Norouzi, and T. Salimans, "Imagen video: High definition video generation with diffusion models," *arXiv preprint arXiv:2210.02303*, 2022.

[10] H. Huang, L. Lin, R. Tong, H. Hu, Q. Zhang, Y. Iwamoto, X. Han, Y.-W. Chen, and J. Wu, "Unet 3+: A full-scale connected unet for medical image segmentation," in *ICASSP 2020-2020 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2020, pp. 1055–1059.

[11] Z. Huang, Z. Geng, W. Luo, and G.-j. Qi, "Flow generator matching," *arXiv preprint arXiv:2410.19310*, 2024. [Online]. Available: https://arxiv.org/abs/2410.19310

[12] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, "Diffwave: A versatile diffusion model for audio synthesis," *arXiv preprint arXiv:2009.09761*, 2020.

[13] S. Lee, Z. Lin, and G. Fanti, "Improving the training of rectified flows," *arXiv preprint arXiv:2405.20320*, 2024.

[14] Y. Lipman, J. Tenenbaum, and D. J. Rezende, "Flow matching for generative modeling," *arXiv preprint arXiv:2210.05675*, 2022.

[15] X. Liu, C. Gong *et al.*, "Flow straight and fast: Learning to generate and transfer data with rectified flow," in *The Eleventh International Conference on Learning Representations*.

[16] X. Liu, X. Zhang, J. Ma, J. Peng *et al.*, "Instaflow: One step is enough for high-quality diffusion-based text-to-image generation," in *The Twelfth International Conference on Learning Representations*.

[17] C. Lu and Y. Song, "Simplifying, stabilizing and scaling continuous-time consistency models," *arXiv preprint arXiv:2410.11081*, 2024.

[18] D. Nvidia, "Afhq dataset (512 × 512)," https://www.kaggle.com/datasets/dimensi0n/afhq-512, 2023, accessed: 2024-11-25.

[19] W. Peebles and S. Xie, "Scalable diffusion models with transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023, pp. 4195–4205.

[20] T. Salimans and J. Ho, "Progressive distillation for fast sampling of diffusion models," *arXiv preprint arXiv:2202.00512*, 2022.

[21] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *arXiv preprint arXiv:2010.02502*, 2020.

[22] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," in *International Conference on Learning Representations*.

[23] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever, "Consistency models," in *International Conference on Machine Learning*. PMLR, 2023, pp. 32 211–32 252.

[24] F.-Y. Wang, L. Yang, Z. Huang, M. Wang, and H. Li, "Rectified diffusion: Straightness is not your need in rectified flow," *arXiv preprint arXiv:2410.07303*, 2024. [Online]. Available: https://arxiv.org/abs/2410.07303

[25] F. C. S. Z. T. H. E. X. Z. L. Xue, Zhaoqiang, "Accelerating diffusion sampling with optimized time steps," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.