Bidirectional Consistency Models

Liangchen Li [*1] **Jiajun He** [*2]

Abstract

Diffusion models (DMs) are capable of generating remarkably high-quality samples by iteratively denoising a random vector, a process corresponding to moving along the probability flow ordinary differential equation (PF ODE). Interestingly, DMs can also invert an input image to noise by moving backward along the PF ODE, a key operation for downstream tasks such as interpolation and image editing. However, the iterative nature of this process restricts its speed. Recently, Consistency Models (CMs) have emerged to address this challenge by approximating the integral of the PF ODE, largely reducing the number of iterations in generation. Yet, the absence of an explicit ODE solver complicates the inversion process. To address this limitation, we introduce Bidirectional Consistency Model (BCM), which learns a *single* neural network that enables both *forward and backward* traversal along the PF ODE, unifying generation and inversion tasks within one framework. Our proposed method supports one-step generation and inversion while allowing the use of additional steps to enhance generation quality or reduce reconstruction error. Its bidirectional consistency also broadens its applications, allowing, for instance, interpolation between two real images - a task beyond the reach of previous CMs.

1 Introduction

Two key components in image generation and manipulation are *generation* and its *inversion*. *Generation* aims to learn a mapping from simple noise distributions, such as Gaussian, to complex ones, like the distribution encompassing all real-world images. In contrast, *inversion* seeks to find the reverse mapping, transforming real data back into the corresponding noise[1]. Recent breakthroughs in deep generative models (Goodfellow et al., 2014; Kingma & Welling, 2013; Ho et al., 2020; Song et al., 2021a;b) not only have achieved remarkable success in synthesizing high-fidelity samples across various modalities (Karras et al., 2021; Rombach et al., 2022; Kong et al., 2021; OpenAI, 2024), but have proven effective in downstream applications, such as image editing, by leveraging the inversion (Mokady et al., 2023; Huberman-Spiegelglas et al., 2023; Hertz et al., 2023).

Particularly, score-based diffusion models (DMs) (Song & Ermon, 2019; Ho et al., 2020; Song et al., 2021a;b; Karras et al., 2022) have stood out among their counterparts for generation (Dhariwal & Nichol, 2021). However, DMs typically require hundreds of iterations to produce high-quality generation results. This issue was recently addressed by consistency models (CMs) (Song et al., 2023; Song & Dhariwal, 2024), which directly compute the integral of PF ODE trajectory from any time step to zero. Similar to CMs, Kim et al. (2024) introduced the Consistency Trajectory Model (CTM), which estimates the integral between any two time steps along the trajectory towards the denoising direction. Through these approaches, the consistency model family enables image generation with only a single Number of Function Evaluation (NFE, i.e., number of network forward passes) while offering a trade-off between speed and quality.

Unfortunately, the inversion direction remains challenging. First, the generation process in many DMs (Ho et al., 2020; Karras et al., 2022) is stochastic and hence non-invertible; second, even for methods that employ an ODE-based deterministic sampling (Song et al., 2021a), it necessitates hundreds of iterations for a small reconstruction error; third, although CMs and CTM accelerate the generation process by solving the integral directly, this integration is strongly non-linear, making the inversion process even more difficult.

Therefore, in this work, we aim to bridge this gap through natural yet non-trivial extensions to CMs and CTM. Specifically, they possess a key feature of self-consistency: points along the same PF ODE trajectory map back to the same initial point. Inspired by this, we pose the questions: *is there a stronger consistency where points on the same trajectory*

[*]Equal contribution [1]Independent Researcher [2]University of Cambridge, UK. Correspondence to: Liangchen Li <ll673@cantab.ac.uk>, Jiajun He <jh2383@cam.ac.uk>.

Accepted by the Structured Probabilistic Inference & Generative Modeling workshop of ICML 2024, Vienna, Austria. Copyright 2024 by the author(s).

[1]The inversion problem also refers to restoring a high-quality image from a degraded one. However, in this paper, we define it narrowly as the task to find the corresponding noise for an image.

can map to each other, regardless of their time steps' order?

In this work, we affirmatively answer the question with our proposed *Bidirectional Consistency Model* (BCM). Concretely, we train a *single* neural network that enables both *forward* and *backward* traversal along the PF ODE, unifying the generation and inversion tasks into one framework. BCM can generate or invert an image with a single NFE, and can achieve improved sample quality or lower reconstruction error by multiple time steps. Additionally, we apply BCM for image interpolation, demonstrating its potential applications enabled by its bidirectional consistency.

2 Background and Preliminary

Before launching into the details of the Bidirectional Consistency Model (BCM), we describe some preliminaries, including a brief introduction to Score-based Diffusion Models (DMs), Consistency Models (CMs), and Consistency Trajectory Model (CTM) in the following.

Score-based Diffusion Models. Score-based Diffusion Models (DMs) sample from the target data distribution by progressively removing noise from a random Gaussian vector. Song et al. (2021b) showed that this process can be viewed as solving an ordinary differential equation, dubbed the Probability Flow (PF) ODE, defined as:

$$d\mathbf{x}_t/dt = \boldsymbol{\mu}(\mathbf{x}_t, t) - 1/2\sigma^2(t)\nabla \log p_t(\mathbf{x}_t). \quad (1)$$

where $t \in [0, T]$[2], and p_t is the marginal density of $\mathbf{x}_t$ generated by diffusing the data distribution $p_{\text{data}}(\mathbf{x})$ through the diffusion SDE $d\mathbf{x}_t = \boldsymbol{\mu}(\mathbf{x}_t, t)dt + \sigma(t)d\mathbf{w}_t$. During training, DMs learn to estimate $\nabla \log p_t(\mathbf{x}_t)$ with a score model $\boldsymbol{s}(\mathbf{x}_t, t)$ with score matching (Hyvärinen & Dayan, 2005; Song et al., 2021b). During sampling, DMs solve Equation (1) from time T to 0 numerically. Following Karras et al. (2022), we set $\mu = 0, \sigma = \sqrt{2t}, T = 80$.

Consistency Models, Consistency Training, and improved Consistency Training. The generation of DMs typically requires hundreds of evaluations of the network $\boldsymbol{s}(\mathbf{x}_t, t)$ and hence bottlenecks the speed. To this end, Song et al. (2023) proposed Consistency Models (CMs) that train a network to estimate the integral of the PF ODE, i.e.,

$$\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) \approx \mathbf{x}_0 = \mathbf{x}_t + \int_t^0 \left(d\mathbf{x}_s/ds\right) ds \quad (2)$$

The network can be trained either by distillation or from scratch with consistency training (CT). We describe CT in more detail since it lays the foundation of our proposed method: to begin, consistency training discretizes the time horizon $[0, T]$ into $N-1$ sub-intervals, with boundaries $0 = t_1 < \cdots < t_N = T$. Then, the CT loss is given by

$$\mathcal{L}_{CT}^N(\boldsymbol{\theta}, \boldsymbol{\theta}^-) \quad (3)$$
$$= \mathbb{E}_{\mathbf{z},\mathbf{x},n}[\lambda(t_n) d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x} + t_{n+1}\mathbf{z}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x} + t_n\mathbf{z}, t_n))].$$

$\boldsymbol{\theta}$ and $\boldsymbol{\theta}^-$ represents the parameters of the online network and a target network, respectively. The target network is obtained by $\boldsymbol{\theta}^- \leftarrow \texttt{stopgrad}(\mu\boldsymbol{\theta}^- + (1-\mu)\boldsymbol{\theta})$ at each iteration. $\lambda(\cdot)$ is a reweighting function, $\mathbf{x}$ represents the training data sample, and $\mathbf{z} \sim \mathcal{N}(0, \boldsymbol{I})$ is a random Gaussian noise. During training, N is gradually increased, allowing the model to learn self-consistency incrementally.

In a follow-up work, Song & Dhariwal (2024) suggested to use the Pseudo-Huber loss for d, along with other techniques that include setting $\mu = 0$, proposing a better scheduler function for N, adapting a better reweighting function $\lambda(t_n) = 1/|t_n - t_{n+1}|$. Dubbed improved Consistency Training (iCT), these modifications significantly improve performance, and hence we inherit these improving techniques in our work unless otherwise stated.

Consistency Trajectory Model. Unlike CMs, which learn the integral from an arbitrary starting time to 0, the Consistency Trajectory Model (CTM) (Kim et al., 2024) learns the integral between any two time steps along the PF ODE trajectory towards the denoising direction:

$$\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, u) \approx \mathbf{x}_u = \mathbf{x}_t + \int_t^u \left(d\mathbf{x}_s/ds\right) ds, \quad u \leqslant t. \quad (4)$$

CTM demonstrates that it is possible to learn a stronger consistency: two points $\mathbf{x}_t$ and $\mathbf{x}_u$ along the same trajectory not only can map back to the same initial point $\mathbf{x}_0$, but also can map from $\mathbf{x}_t$ to $\mathbf{x}_u$, provided $u \leqslant t$. This inspires us for a stronger consistency with a bijection between $\mathbf{x}_t$ and $\mathbf{x}_u$.

3 Methods

In this section, we describe the details of BCM. In a nutshell, we train a network $\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, u)$ that traverses along the probability flow (PF) ODE from time t to time u, i.e., $\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, u) \approx \mathbf{x}_u = \mathbf{x}_t + \int_t^u \frac{d\mathbf{x}_s}{ds} ds$. This is similar to Equation (4), but since we aim to learn both generation and inversion, we do not set constraints on t and u, except for $t \neq u$. To this end, we adjust the network parameterization and the training objective of consistency training (Song et al., 2023; Song & Dhariwal, 2024).

3.1 Network Parameterization

Our network takes in three arguments: 1) the sample $\mathbf{x}_t$ at time t, 2) the current time step t, and 3) the target time step u, and outputs the sample at time u, i.e., $\mathbf{x}_u$. To achieve this, we directly expand the models used in Consistency Models (CMs) (Song et al., 2023; Song & Dhariwal, 2024) with an extra argument u. In CMs, the networks first calculate Fourier embeddings (Tancik et al., 2020) or positional embeddings (Vaswani et al., 2017) for the time step t, followed

[2]To avoid numerical issues, we always set t in $[0.002, T]$ in practice. However, to keep the notation simple, we will ignore this small value 0.002 when describing our methods in this paper.

by two dense layers. Here, we simply concatenate the embeddings of t and u, and double the dimensionality of the dense layers correspondingly.

Similar to CMs (Song et al., 2023; Song & Dhariwal, 2024) and EDM (Karras et al., 2022), instead of directly learning $\boldsymbol{f}_{\boldsymbol{\theta}}$, we train $\boldsymbol{F}_{\boldsymbol{\theta}}$ and let $\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, u) = c_{\text{skip}}(t, u)\mathbf{x}_t + c_{\text{out}}(t, u)F_{\boldsymbol{\theta}}(c_{\text{in}}(t, u)\mathbf{x}_t, t, u)$, where we set $c_{\text{in}}(t, u) = \frac{1}{\sqrt{\sigma_{\text{data}}^2 + t^2}}, c_{\text{out}}(t, u) = \frac{\sigma_{\text{data}}(t-u)}{\sqrt{\sigma_{\text{data}}^2 + t^2}}, c_{\text{skip}}(t, u) = \frac{\sigma_{\text{data}}^2 + tu}{\sigma_{\text{data}}^2 + t^2}$. Notice that $c_{\text{skip}}(t, t) = 1$ and $c_{\text{out}}(t, t) = 0$, which explicitly enforce the boundary condition $\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, t) = \mathbf{x}_t$. We detail the derivations for this parameterization in Appendix F.

3.2 Bidirectional Consistency Training

We now discuss the training of BCM, which we dub as Bidirectional Consistency Training (BCT). Following Song & Dhariwal (2024), we discretize the time horizon $[0, T]$ into $N - 1$ intervals, with boundaries $0 = t_1 < t_2 < \cdots < t_N = T$, and increase N gradually during training.

Our training objective has two terms. The first term takes the same form as Equation (3), enforcing the consistency between any points on the trajectory and the starting point. We restate it here for easier reference:

$$\begin{aligned}&\mathcal{L}_{CT}^{N}(\boldsymbol{\theta}) \qquad (5)\\ =&\mathbb{E}_{\mathbf{z},\mathbf{x},t_n}[\lambda(t_n)d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x} + t_{n+1}\mathbf{z}, t_{n+1}, 0), \boldsymbol{f}_{\bar{\boldsymbol{\theta}}}(\mathbf{x} + t_n\mathbf{z}, t_n, 0))],\end{aligned}$$

where $\mathbf{x}$ is one training sample, $\mathbf{z} \sim \mathcal{N}(0, \boldsymbol{I})$, $\bar{\boldsymbol{\theta}}$ represents the same parameter $\boldsymbol{\theta}$ with the stop gradient operation, and $\lambda(t_n) = 1/|t_n - t_{n+1}|$. Note, that we replace $\boldsymbol{\theta}^-$ in Equation (3) with $\bar{\boldsymbol{\theta}}$ according to Song & Dhariwal (2024).

The second term explicitly sets constraints between any two points on the trajectory. Specifically, given a training example $\mathbf{x}$, we randomly sample two time steps t and u, and want to construct a mapping from $\mathbf{x}_t$ to $\mathbf{x}_u$, where $\mathbf{x}_t$ and $\mathbf{x}_u$ represent the results at time t and u along the Probability Flow (PF) ODE trajectory, respectively. Note, that the model learns to denoise (i.e., generate) when $u < t$, and to add noise (i.e., inverse) when $u > t$. Therefore, this single term unifies the generative and inverse tasks within one framework, and with more t and u sampled during training, we achieve consistency over the entire trajectory. To construct such a mapping, we hope to minimize the distance

$$d\left(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, u), \mathbf{x}_u\right). \qquad (6)$$

However, Equation (6) will have different scales for different u values, leading to a high variance during training. Therefore, inspired by Kim et al. (2024), we map both $\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, u)$ and $\mathbf{x}_u$ to time 0, and minimize the distances between these two back-mapped images, i.e.,

$$d\left(\boldsymbol{f}_{\bar{\boldsymbol{\theta}}}(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, u), u, 0)), \boldsymbol{f}_{\bar{\boldsymbol{\theta}}}(\mathbf{x}_u, u, 0)\right), \qquad (7)$$

where $\bar{\boldsymbol{\theta}}$ is the same $\boldsymbol{\theta}$ with stop gradient operation. We denote this as a "soft" trajectory constraint. Unfortunately, directly minimizing Equation (7) without a pre-trained DM is still problematic. This is because, without a pre-trained DM, we can only generate $\mathbf{x}_t$ and $\mathbf{x}_u$ from the diffusion SDE, i.e., by adding Gaussian noise to $\mathbf{x}$. However, $\mathbf{x}_t$ and $\mathbf{x}_u$ generated by the diffusion SDE do not necessarily lie on the same PF ODE trajectory, and hence Equation (7) still fails to build the desired bidirectional consistency. Instead, we notice that when the CT loss defined in Equation (5) converges, we have $\boldsymbol{f}_{\bar{\boldsymbol{\theta}}}(\mathbf{x}_u, u, 0) \approx \boldsymbol{f}_{\bar{\boldsymbol{\theta}}}(\mathbf{x}_t, t, 0) \approx \mathbf{x}$. We therefore optimize:

$$d\left(\boldsymbol{f}_{\bar{\boldsymbol{\theta}}}(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, u), u, 0)), \boldsymbol{f}_{\bar{\boldsymbol{\theta}}}(\mathbf{x}_t, t, 0)\right). \qquad (8)$$

Empirically, we found Equation (8) plays a crucial role in ensuring accurate inversion performance. We provide experimental evidence for this loss choice in Appendix C.1.

Finally, we recognize that the term $\boldsymbol{f}_{\bar{\boldsymbol{\theta}}}(\mathbf{x} + t_n\mathbf{z}, t_n, 0)$ in Equation (5) and the term $\boldsymbol{f}_{\bar{\boldsymbol{\theta}}}(\mathbf{x}_t, t, 0)$ in Equation (8) have exactly the same form. We thus set $t = t_n$ to reduce one forward pass. Putting together, we define our objective as

$$\begin{aligned}&\mathcal{L}_{BCT}^{N}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{z},\mathbf{x},t_n,t_{n'}}[\ell_{\text{CT}} + \ell_{\text{soft}}], \qquad (9)\\ &\ell_{\text{CT}} = \lambda(t_n)d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x} + t_{n+1}\mathbf{z}, t_{n+1}, 0), \boldsymbol{f}_{\bar{\boldsymbol{\theta}}}(\mathbf{x} + t_n\mathbf{z}, t_n, 0))\\ &\ell_{\text{soft}} = \lambda'(t_n, t_{n'})d\left(\boldsymbol{f}_{\bar{\boldsymbol{\theta}}}(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x} + t_n\mathbf{z}, t_n, t_{n'}), t_{n'}, 0)), \boldsymbol{f}_{\bar{\boldsymbol{\theta}}}(\mathbf{x} + t_n\mathbf{z}, t_n, 0)\right),\end{aligned}$$

where we set reweighting as $\lambda'(t_n, t_{n'}) = 1/|t_n - t_{n'}|$.

Regarding other training settings, encompassing the scheduler function for N, the sampling probability for t_n (aka the noise schedule $p(n)$ in (Song et al., 2023; Song & Dhariwal, 2024)), the EMA rate, and more, we follow Song & Dhariwal (2024), and include details in Appendix D.1. We summarize our training algorithm in Algorithm 1 and compare the training of CT, CTM, and BCT in Table 2 in Appendix B.

4 Experiments

4.1 Image Generation

We evaluate BCM's generation on CIFAR-10 dataset (Krizhevsky et al., 2009), with hyperparameters in Appendix D.2. As our BCM supports both forward and backward traversal along the PF ODE, we can employ more complicated sampling procedures compared to CMs to enhance the sample quality. We therefore propose three sampling schemes: ancestral sampling, zigzag sampling, and their combination. From a high-level perspective, *ancestral sampling* removes noise iteratively, while *zigzag sampling* interchanges between denoising steps and noising steps. Different from the zigzag sampling in CMs, where the noising steps are achieved by manually adding Gaussian noise and hence will alter the image content, BCM uses the network to amplify a small noise, which can improve the sample quality while largely preserving the image content. We detail these three approaches in Appendix A.2.

Table 1. NFE and sample quality on CIFAR-10. *Results estimated from Figure 13 in Kim et al. (2024). †For BCM and BCM-deep, we use ancestral sampling when NFE=2, zigzag sampling when NFE=3, and their combination when NFE=4.

METHOD	NFE (↓)	FID (↓)	IS (↑)
DPM-solver-fast (Lu et al., 2022)	10	4.70	-
AMED-plugin (Zhou et al., 2023)	5	6.61	-
Progressive Distillation (Salimans & Ho, 2022)	1	8.34	8.69
CD (LPIPS) (Song et al., 2023)	1	3.55	9.48
	2	2.93	9.75
CTM (LPIPS, GAN loss) (Kim et al., 2024)	1	1.98	-
	2	1.87	-
CTM (LPIPS, w/o GAN loss) (Kim et al., 2024)	1	> 5.00*	-
CT (LPIPS) (Song et al., 2023)	1	8.70	8.49
	2	5.83	8.85
CTM (LPIPS, GAN loss) (Kim et al., 2024)	1	2.39	-
iCT (Song & Dhariwal, 2024)	1	2.83	9.54
	2	2.46	9.80
iCT (deep) (Song & Dhariwal, 2024)	1	2.51	9.76
	2	2.24	9.89
BCM (ours)†	1	3.10	9.45
	2	2.39	9.88
	3	2.50	9.82
	4	2.29	9.92
BCM (ours, deep)†	1	2.64	9.67
	2	2.36	9.86
	3	2.19	9.94
	4	2.07	10.02

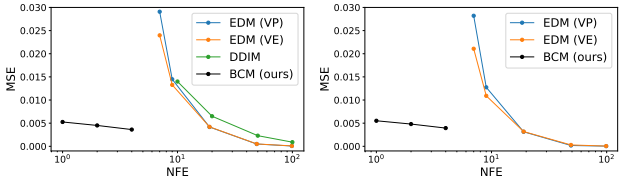


Figure 1. Reconstruction error on CIFAR-10 test set. **Left:** unconditional models; **Right:** conditional models.

We report the NFE/FID/IS in Table 1 and provide visualization of generated samples in Appendix H. We can see BCM yields competitive FID with just 1 NFE and can achieve better generation quality with more steps. Compared with fast solver or distillation on DMs, including DPM-solver-fast (Lu et al., 2022), AMED (Zhou et al., 2023), and Progressive Distillation (Salimans & Ho, 2022), BCM achieves better results with fewer NFEs. Compared within the consistency model family, iCT (Song & Dhariwal, 2024) surpasses BCM for 1-step sampling. However, as the number of function evaluations (NFEs) increases, BCM starts to outperform. On the other hand, our model's performance still falls short of CTM's (Kim et al., 2024). However, we note that CTM relies heavily on adversarial loss, and also uses LPIPS (Zhang et al., 2018) as the distance measure, which may cause feature leakage and performance overestimation (Song & Dhariwal, 2024; Kynkäänniemi et al., 2023).

4.2 Inversion and Reconstruction

As highlighted earlier, a distinctive feature of BCM is its ability to invert an input image back to a noise image and then reconstruct the same image using notably few NFEs. Here, we evaluate BCM's capability for inversion on the CIFAR-10 test set and report the per-dimension mean squared error (scaled to [0,1]) in Figure 1. We include the results by DDIM (Song et al., 2021a) and EDM (Karras et al., 2022) for comparison. We detail the inversion algorithm in Appendix A.3 and present the hyperparameters in Appendix D.2. We can see both unconditional and conditional BCMs achieve a lower reconstruction error than ODE-based diffusion models, with significantly fewer NFEs. We also visualize the noise generated by BCM at Figure 9 in Appendix G, from which we can verify that BCM Gaussianizes the input image as desired.

Figure 2. Interpolation between two *real* images.

4.3 Interpolation

Leveraging the bidirectional consistency, our BCM can interpolate between two given *real* images. This is a more meaningful application compared with CMs, which can only perform interpolations between *generated* images (Song et al., 2023). Specifically, we first invert the given images into noises, smoothly interpolate the noises, and then map the noises back to images. We illustrate some results in Figure 2 and provide more examples in Figure 11. We defer more details and discussions to Appendix D.3.

5 Conclusions and Limitations

In this work, we introduce the Bidirectional Consistency Model (BCM), enhancing upon existing consistency models (Song et al., 2023; Song & Dhariwal, 2024; Kim et al., 2024) by establishing a stronger consistency. This consistency ensures that points along the same trajectory of the probability flow (PF) ODE map to each other, thereby unifying generation and inversion tasks within one framework.

However, BCM still faces several limitations. First, similar to CMs, while employing more steps in generation or inversion can initially enhance results, the performance tends to plateau quickly. Additionally, our method sometimes delivers imperfect inversion, which may alter the image content. Another limitation of BCM is its expensive training cost, particularly for higher-resolution images. Future work can involve employing the parameterization and tricks proposed by Kim et al. (2024), developing accurate inversion methods, like the approach by Wallace et al. (2023), and applying BCM in the latent space to accelerate training, similar to the latent consistency model (Luo et al., 2023).

6 Acknowledgment

We are grateful to Zongyu Guo for the helpful discussion throughout this project, especially for pointing out the significance of the inverse process, suggesting some potential applications, and proofreading the preprint version of this work. We also thank Wenlin Chen and Sergio Calvo-Ordoñez for reviewing and providing feedback on the manuscript. JH acknowledges support from Prof. José Miguel Hernández-Lobato's Turing AI Fellowship under grant EP/V023756/1.

References

Dhariwal, P. and Nichol, A. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Hertz, A., Mokady, R., Tenenbaum, J., Aberman, K., Pritch, Y., and Cohen-or, D. Prompt-to-prompt image editing with cross-attention control. In *The Eleventh International Conference on Learning Representations*, 2023.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Huberman-Spiegelglas, I., Kulikov, V., and Michaeli, T. An edit friendly ddpm noise space: Inversion and manipulations. *arXiv preprint arXiv:2304.06140*, 2023.

Hyvärinen, A. and Dayan, P. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

Karras, T., Aittala, M., Laine, S., Härkönen, E., Hellsten, J., Lehtinen, J., and Aila, T. Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems*, 34:852–863, 2021.

Karras, T., Aittala, M., Aila, T., and Laine, S. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35: 26565–26577, 2022.

Kim, D., Lai, C.-H., Liao, W.-H., Murata, N., Takida, Y., Uesaka, T., He, Y., Mitsufuji, Y., and Ermon, S. Consistency trajectory models: Learning probability flow ode trajectory of diffusion. In *International Conference on Learning Representations*, 2024.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kong, Z., Ping, W., Huang, J., Zhao, K., and Catanzaro, B. Diffwave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations*, 2021.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Kynkäänniemi, T., Karras, T., Aittala, M., Aila, T., and Lehtinen, J. The role of imagenet classes in fréchet inception distance. In *The Eleventh International Conference on Learning Representations*, 2023.

Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., and Han, J. On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations*, 2020.

Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022.

Luo, S., Tan, Y., Huang, L., Li, J., and Zhao, H. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*, 2023.

Lyu, J., Chen, Z., and Feng, S. Convergence guarantee for consistency models. *arXiv preprint arXiv:2308.11449*, 2023.

Mokady, R., Hertz, A., Aberman, K., Pritch, Y., and Cohen-Or, D. Null-text inversion for editing real images using guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6038–6047, 2023.

OpenAI. Video generation models as world simulators. https://openai.com/research/video-generation-models-as-world-simulators, 2024. Accessed: 2024-02-26.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Salimans, T. and Ho, J. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a.

Song, Y. and Dhariwal, P. Improved techniques for training consistency models. In *International Conference on Learning Representations*, 2024.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021b.

Song, Y., Dhariwal, P., Chen, M., and Sutskever, I. Consistency models. In *Proceedings of the 40th International Conference on Machine Learning*. JMLR.org, 2023.

Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., and Ng, R. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33: 7537–7547, 2020.

Van der Maaten, L. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Vincent, P. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

Wallace, B., Gokul, A., and Naik, N. Edict: Exact diffusion inversion via coupled transformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22532–22541, 2023.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018.

Zhou, Z., Chen, D., Wang, C., and Chen, C. Fast ode-based sampling for diffusion models in around 5 steps. *arXiv preprint arXiv:2312.00094*, 2023.

A Algorithms

A.1 Bidirectional Consistency Training

We provide the training algorithms of BCM in this section.

Algorithm 1 Bidirectional Consistency Training (Orange indicates differences from CT/iCT (Song et al., 2023; Song & Dhariwal, 2024)

Input: Training set $\mathcal{D}$, initial model parameter $\boldsymbol{\theta}$, learning rate η, step schedule $N(\cdot)$, noise schedule $p(\cdot)$, EMA rate μ_{EMA}, distance metric $d(\cdot,\cdot)$, reweighting function $\lambda(\cdot)$ and $\lambda'(\cdot,\cdot)$.
Output: Model parameter $\boldsymbol{\theta}_{\text{EMA}}$.

Initialize: $\boldsymbol{\theta}_{\text{EMA}} \leftarrow \boldsymbol{\theta}, k \leftarrow 0$.

repeat until convergence

Sample training example, time steps, and random noise:
Sample $\mathbf{x} \in \mathcal{D}$, $n \sim p(n|N(k))$.
Sample $n' \sim \tilde{p}(n'|N(k))$, where $\tilde{p}(n'|N(k)) \propto \begin{cases} 0, & \text{if } n' = n, \\ p(n'|N(k)), & \text{otherwise.} \end{cases}$
Sample $\mathbf{z} \sim \mathcal{N}(0, \boldsymbol{I})$.
Calculate and optimize BCT loss:
$\mathcal{L}_{\text{CT}}(\boldsymbol{\theta}) \leftarrow \lambda(t_n) d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x} + t_{n+1}\mathbf{z}, t_{n+1}, 0), \boldsymbol{f}_{\bar{\boldsymbol{\theta}}}(\mathbf{x} + t_n\mathbf{z}, t_n, 0))$;
$\mathcal{L}_{\text{ST}}(\boldsymbol{\theta}) \leftarrow \lambda'(t_n, t_{n'}) d\left(\boldsymbol{f}_{\bar{\boldsymbol{\theta}}}(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x} + t_n\mathbf{z}, t_n, t_{n'}), t_{n'}, 0)), \boldsymbol{f}_{\bar{\boldsymbol{\theta}}}(\mathbf{x} + t_n\mathbf{z}, t_n, 0)\right)$;
$\mathcal{L}(\boldsymbol{\theta}) \leftarrow \mathcal{L}_{\text{CT}}(\boldsymbol{\theta}) + \mathcal{L}_{\text{ST}}(\boldsymbol{\theta})$;
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$;
Update the EMA parameter and the iteration number:
$\boldsymbol{\theta}_{\text{EMA}} \leftarrow \mu_{\text{EMA}} \boldsymbol{\theta}_{\text{EMA}} + (1 - \mu_{\text{EMA}}) \boldsymbol{\theta}, k \leftarrow k + 1$;

end repeat

A.2 BCM's Sampling Algorithms

In this section, we detail BCM's sampling algorithms.

First, BCM supports 1-step sampling, similar to CMs and CTM. More interestingly, BCM's capability to navigate both forward and backward along the PF ODE trajectory allows us to design multi-step sampling strategies to improve the sample quality. We present two schemes and a combined approach that has empirically demonstrated superior performance in the following.

Ancestral Sampling. The most straightforward way for multi-step sampling is to remove noise sequentially. Specifically, we first divide the time horizon $[0, T]$ into N sub-intervals with boundaries $0 = t_0 < t_1 < \cdots < t_N = T$. Then, we sample a noise image $\mathbf{x}_T \sim \mathcal{N}(0, T^2\boldsymbol{I})$, and sequentially remove noise with the network: $\mathbf{x}_{t_{n-1}} \leftarrow \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_n}, t_n, t_{n-1}), \quad n = N, N-1, ..., 1$. We note that the discretization strategy may differ from the one used during the training of BCM. Since this sampling procedure can be viewed as drawing samples from the conditional density $p_{\mathbf{x}_{t_{n-1}}|\mathbf{x}_{t_n}}(\mathbf{x}_{t_{n-1}}|\mathbf{x}_{t_n})$, we dub it as ancestral sampling, and summarize it in Algorithm 2. We can also view 1-step sampling as ancestral sampling, where we only divide the time horizon into a single interval.

Zigzag Sampling. Another effective sampling method (Algorithm 1 in (Song et al., 2023)) is to iteratively re-add noise after denoising. Similar to ancestral sampling, we also define a sequence of time steps $t_1 < \cdots < t_N = T$. However, different from ancestral sampling where we gradually remove noise, we directly map $\mathbf{x}_T$ to $\mathbf{x}_0$ by $\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_T, T, 0)$. We then add a fresh Gaussian noise to $\mathbf{x}_0$, mapping it from time 0 to time t_{n-1}, i.e., $\mathbf{x}_{t_{n-1}} = \mathbf{x}_0 + t_{n-1}\boldsymbol{\sigma}$, where $\boldsymbol{\sigma} \sim \mathcal{N}(0, \boldsymbol{I})$. This process repeats in this zigzag manner until all the designated time steps are covered. The two-step zigzag sampler effectively reduces FID in CMs (Song et al., 2023) and is theoretically supported (Lyu et al., 2023). However, the injected fresh noise can alter the content of the image after each iteration, which is undesirable, especially considering that our tasks will both include generation and inversion. One may immediately think that setting the injected noise to be the same as the initial random noise can fix this issue. However, we reveal that this will significantly damage the quality of the generated images.

Fortunately, our proposed BCM provides a direct solution, leveraging its capability to traverse both forward and backward along the PF ODE. Specifically, rather than manually reintroducing a large amount of fresh noise, we initially apply a small

Algorithm 2 BCM's ancestral sampling

Input: Network $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot,\cdot,\cdot)$, time steps $0 = t_0 < t_1 < \cdots < t_N = T$, initial noise $\mathbf{x}_T$.
Output: Generated image $\mathbf{x}_{t_0}$.

$\mathbf{x}_{t_N} \leftarrow \mathbf{x}_T$.
for $n = N, \cdots, 1$ **do**
  $\mathbf{x}_{t_{n-1}} \leftarrow \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_n}, t_n, t_{n-1})$. ▷ Denoise image from time step t_n to t_{n-1}.
end for
Return: $\mathbf{x}_{t_0}$.

Algorithm 3 BCM's zigzag sampling

Input: Network $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot,\cdot,\cdot)$, time steps $t_1 < \cdots < t_N = T$, manually-added noise scale at each time step $\varepsilon_1, \ldots, \varepsilon_{N-1}$, initial noise $\mathbf{x}_T$.
Output: Generated image $\mathbf{x}$.

$\mathbf{x}_{t_N} \leftarrow \mathbf{x}_T$.
for $n = N, \cdots, 2$ **do**
  $\mathbf{x} \leftarrow \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_n}, t_n, 0)$. ▷ Denoise image from time step t_n to 0.
  $\boldsymbol{\sigma} \sim \mathcal{N}(0, \boldsymbol{I})$, and $\mathbf{x}_{\varepsilon_{n-1}} \leftarrow \mathbf{x} + \varepsilon_{n-1}\boldsymbol{\sigma}$. ▷ Add small fresh noise.
  $\mathbf{x}_{t_{n-1}} \leftarrow \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{\varepsilon_{n-1}}, \varepsilon_{n-1}, t_{n-1})$. ▷ Amplify noise by network.
end for
$\mathbf{x} \leftarrow \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_1}, t_1, 0)$.
Return: $\mathbf{x}$.

Algorithm 4 Combination of ancestral and zigzag sampling

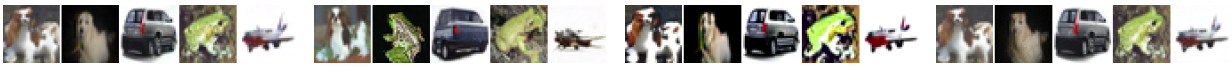
Input: Network $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot,\cdot,\cdot)$, ancestral time steps $t_1 < \cdots < t_N = T$, zigzag time steps $\tau_1 < \cdots < \tau_M = t_1$, manually-added noise scale at each time step $\varepsilon_1, \ldots, \varepsilon_{M-1}$, initial noise $\mathbf{x}_T$.
Output: Generated image $\mathbf{x}$.

$\mathbf{x}_{t_N} \leftarrow \mathbf{x}_T$.
# Ancestral sampling steps
for $n = N, \cdots, 2$ **do**
  $\mathbf{x}_{t_{n-1}} \leftarrow \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_n}, t_n, t_{n-1})$. ▷ Denoise image from time step t_n to t_{n-1}.
end for
# Zigzag sampling steps
$\mathbf{x}_{\tau_M} \leftarrow \mathbf{x}_{t_1}$.
for $m = M, \cdots, 2$ **do**
  $\mathbf{x} \leftarrow \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{\tau_m}, \tau_m, 0)$. ▷ Denoise image from time step τ_m to 0.
  $\boldsymbol{\sigma} \sim \mathcal{N}(0, \boldsymbol{I})$, and $\mathbf{x}_{\varepsilon_{m-1}} \leftarrow \mathbf{x} + \varepsilon_{m-1}\boldsymbol{\sigma}$. ▷ Add small fresh noise.
  $\mathbf{x}_{\tau_{m-1}} \leftarrow \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{\varepsilon_{m-1}}, \varepsilon_{m-1}, \tau_{m-1})$. ▷ Amplify noise by network.
end for
$\mathbf{x} \leftarrow \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{\tau_1}, \tau_1, 0)$.
Return: $\mathbf{x}$.

Algorithm 5 BCM's inversion

Input: Network $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot,\cdot,\cdot)$, time steps $\varepsilon = t_1 < \cdots < t_N \leqslant T$, initial image $\mathbf{x}_0$.
Output: Noise $\mathbf{x}_{t_N}$.

$\boldsymbol{\sigma} \sim \mathcal{N}(0, \boldsymbol{I})$, $\quad \mathbf{x}_{t_1} \leftarrow \mathbf{x} + \varepsilon\boldsymbol{\sigma}$.
for $n = 2, \cdots, N$ **do**
  $\mathbf{x}_{t_n} \leftarrow \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n-1}}, t_{n-1}, t_n)$. ▷ Add noise to image from time step t_{n-1} to t_n.
end for
Return: $\mathbf{x}_{t_N}$.



(a) (b) (c) (d)

Figure 3. Comparison of different strategies of adding fresh noise in zigzag sampling. (a) 1-step generation. (b) Zigzag sampling with manually added fresh noise, where the new noises drastically alter the content. (c) Zigzag sampling with manually added, fixed noise, i.e., we fix the injected fresh noise in each iteration to be the same as the initial one. We can see that the quality significantly deteriorates. (d) Zigzag sampling with BCM. At each iteration, we apply a small amount of noise and let the network amplify it. We can see that the image content is mostly maintained.

amount and let the network amplify it. In a nutshell, for iteration n ($n = N, N-1, \cdots, 2$), we have

$$\mathbf{x}_0 \leftarrow \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_n}, t_n, 0), \quad \mathbf{x}_{\varepsilon_{n-1}} \leftarrow \mathbf{x}_0 + \varepsilon_{n-1}\boldsymbol{\sigma}, \quad \mathbf{x}_{t_{n-1}} \leftarrow \boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{\varepsilon_{n-1}}, \varepsilon_{n-1}, t_{n-1}). \tag{10}$$

where ε_{n-1} is the scale of the small noises we add in n-th iteration, and $\boldsymbol{\sigma} \sim \mathcal{N}(0, \boldsymbol{I})$ is a fresh Gaussian noise. We detail this scheme in Algorithm 3. To verify its effectiveness, in Figure 3, we illustrate some examples to compare the generated images by 1) manually adding fresh noise, 2) manually adding fixed noise, and 3) our proposed sampling process, i.e., adding a small noise and amplifying it with the network. We can clearly see our method maintains the generated content.

Combination of Both. Kim et al. (2024) note that long jumps along the PF ODE in zigzag sampling can lead to accumulative errors, especially at high noise levels, which potentially hampers further improvements in sample quality. Therefore, we propose a combination of ancestral sampling and zigzag sampling. Specifically, we first perform ancestral sampling to rapidly reduce the large initial noise to a more manageable noise scale and then apply zigzag sampling within this reduced noise level. We describe this combined process in Algorithm 4, and empirically find it results in superior sample quality compared to employing either ancestral sampling or zigzag sampling in isolation.

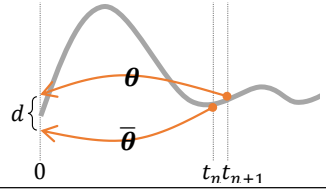
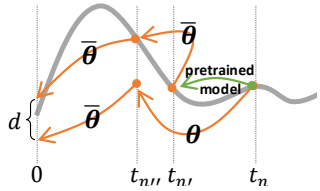
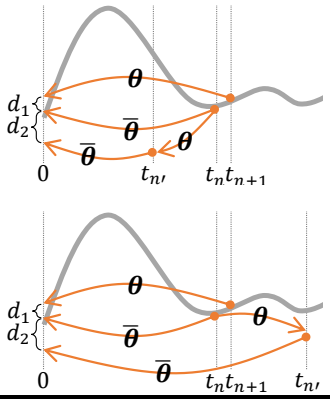
A.3 BCM's Inversion Algorithms

BCM inverts an image following the same principle of sampling. Specifically, we also set an increasing sequence of noise scales $\varepsilon = t_1 < t_2 < \cdots < t_N \leqslant T$. Note that, in contrast to the generation process, it is not always necessary for t_N to equal T. Instead, we can adjust it as a hyperparameter based on the specific tasks for which we employ inversion. Then, given an image $\mathbf{x}_0$, we first inject a small Gaussian noise by $\mathbf{x}_{t_1} = \mathbf{x}_0 + \varepsilon\boldsymbol{\sigma}$, and then sequentially add noise with the network, i.e., $\mathbf{x}_{t_{n+1}} = f(\mathbf{x}_{t_n}, t_n, t_{n+1}), \quad n = 1, 2, \cdots, N-1$. The adoption of small initial noise is due to the observation that the endpoint of the time horizon is less effectively covered and learned during training, as discussed in Appendix C.3. Empirically, we find this minor noise does not change the image's content and leads to lower reconstruction errors when $\varepsilon \approx 0.07$. One may also include denoising steps interleaved with noise magnifying steps, like zigzag sampling, but we find it helps little in improving inversion quality. We summarize the inversion procedure in Algorithm 5.

B Comparing Training of CM, CTM, and BCM

We compare the training objective of CT, CTM and our proposed BCT in Table 2, where we can see how our method naturally extends CT and differs from CTM.

Table 2. Comparison of CT, CTM training, and BCT training methodology. The figures illustrate the main objective of each method, where $\bar{\boldsymbol{\theta}}$ stands for stop gradient operation. Note that for BCM, there are two possible scenarios corresponding to the denoising and diffusion direction, respectively.

Model	Illustration of Training Objective	Detailed Form of Loss
CT		$\mathcal{L}_{CT} = \mathbb{E}_{t_n,\mathbf{x}}[\lambda(t_n)d]$, $\mathbf{x}$ is the training sample, $\lambda(\cdot)$ is the reweighting function, t_n, d are illustrated in the left plot.
CTM		$\mathcal{L}_{CTM} = \mathbb{E}_{t_n,t_{n'},t_{n''},\mathbf{x}}[d] + \lambda_{GAN}\mathcal{L}_{GAN} + \lambda_{DSM}\mathcal{L}_{DSM}$ $\mathcal{L}_{DSM}$ is the adversarial loss, $\mathcal{L}_{DSM}$ is Denoising Score Matching loss (Song et al., 2021b; Vincent, 2011), $\lambda_{GAN}, \lambda_{DSM}$ are the reweighting functions, $t_n, t_{n'}, t_{n''}, d$ are illustrated in the left plot.
BCT		$\mathcal{L}_{BCT} = \mathbb{E}_{t_n,t_{n'},\mathbf{x}}[\lambda(t_n)d_1 + \lambda'(t_n,t_{n'})d_2]$, $\lambda(\cdot), \lambda'(\cdot,\cdot)$ are the reweighting functions, $t_n, t_{n'}, d_1, d_2$ are illustrated in the left plot.

C Ablation Studies and Additional Conclusions

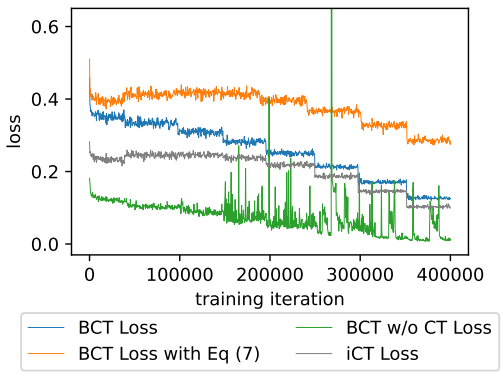


Figure 4. Tracking the loss with different objection functions. We include the loss curve of iCT (Song & Dhariwal, 2024) for reference. We can see that the model with BCT loss defined in Equation (9) converges well. Conversely, the model applying Equation (7) instead of Equation (8) for the soft constraint has a much higher loss at the end of the optimization. While the one without CT loss totally diverges.

C.1 Comparison between Loss defined with Equation (7) and Equation (8)

(a) original images (b) Inversion w. NFE=1 (c) Inversion w. NFE=4 (d) Inversion w. NFE=12

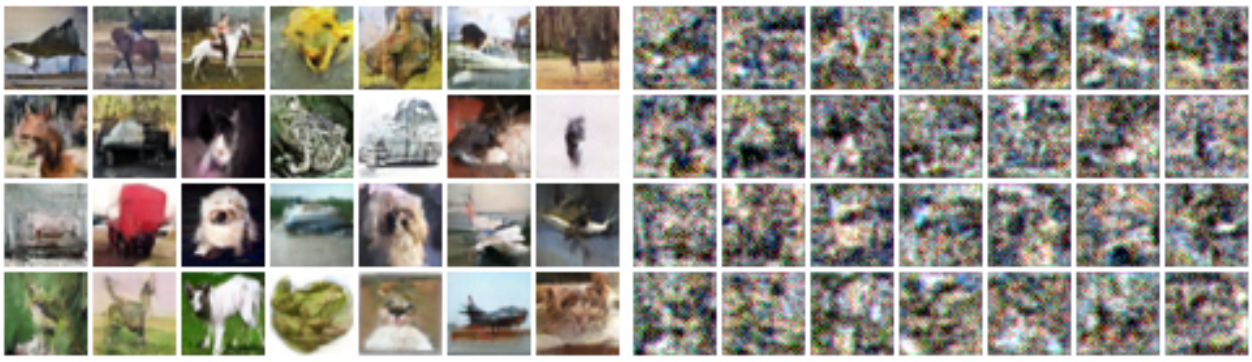
Figure 5. Inversion and reconstruction by BCM trained with Equation (7). We can see the model trained fails to provide an accurate inversion. Even though the images start to look plausible with NFE=12, the content compared with the original images has been significantly changed.

In Section 3.2, we discussed that we optimize Equation (8) instead of Equation (7). Here we provide experimental evidence for our design choice.

We track the loss by models trained with both choices in Figure 4, where we can see that the model trained with Equation (7) features a much higher loss in the end. This echoes its failure in the inversion process: as shown in Figure 5, the model trained with Equation (7) fails to provide an accurate inversion. This is because Equation (7) contains two trajectories, starting from $\mathbf{x}_u$ and $\mathbf{x}_t$. While both of them are along the SDE trajectories starting from the same $\mathbf{x}_0$, they do not necessarily reside on the same PF ODE trajectory; in fact, the probability that they are on the same PF ODE trajectory is 0. On the contrary, Equation (8) bypasses this issue since it only involves trajectories starting from the same $\mathbf{x}_t$.

C.2 Ablation of CT loss

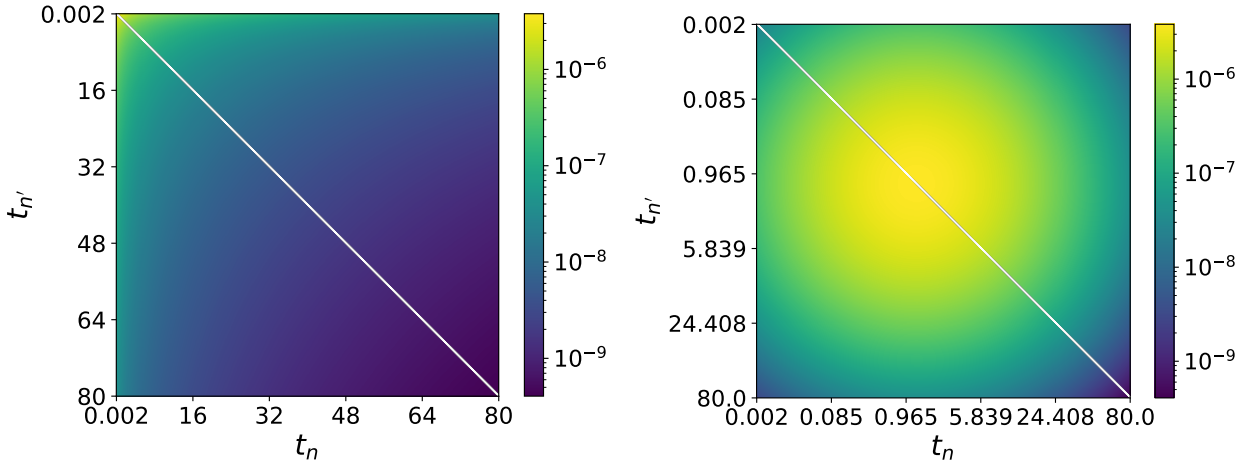
Recall our final loss function has two terms, the soft trajectory constraint term and the CT loss term. We note that the soft constraint defined in Equation (8) can, in principle, cover the entire trajectory, so it should also be able to learn the mapping from any time step t to 0, which is the aim of CT loss. However, we find it crucial to include CT loss in our objective. We provide the loss curve trained without CT loss term in Figure 4, where we can see the training fails to converge. For further verification, we also visualize the images generated by the model trained with full BCM loss and the model trained without CT loss term after 200k iterations in Figure 6. We can clearly see that the model without CT loss cannot deliver meaningful outcomes.



(a) (b)

Figure 6. Images generated by (a) the model trained with full BCM loss for 200k iterations, and (b) the model trained without CT loss term for 200k iterations.

C.3 Coverage of Trajectory during Training



(a) (b)

Figure 7. Probability mass of $(t_n, t_{n'})$ pair being selected during BCT. We transfer the time axes to log space in (b) for a clearer visualization.

In Figure 7, we visualize the probability of selecting a $(t_n, t_{n'})$ pair during the BCT process, where $(t_n, t_{n'})$ is defined in Equation (9). This offers insights into how well the entire trajectory is covered and trained. We can see that most of the probability mass is concentrated in small-time regions. This reveals some of our observations in the experiments:

- first, in the generation process, we find the combination of a zigzag and ancestral sampling yields optimal performance because the ancestral sampler can rapidly jump over the large-time regions, which we cover less in training;
- second, it also explains why adding a small initial noise in inversion helps: simply adding a small $\varepsilon \approx 0.085$ noise increases the probability of being selected during BCT by more than a thousandfold;
- third, it offers insights into the necessity of incorporating CT loss into our final objective, as defined in Equation (9): while theoretically, the soft constraint is expected to cover the entire trajectory, including boundary conditions, it is highly inefficient in practice. Therefore, explicitly including CT loss to learn the mapping from any noise scale t to 0 is crucial.

This also points out some future directions to improve BCM. For example, we can design a better sampling strategy during training to ensure a better coverage of the entire trajectory. Or using different sampling strategy for CT loss term and the soft trajectory constraint term.

D Experiment Details

In this section, we provide the experiment details omitted from the main paper.

D.1 Training Settings

For all the experiments on CIFAR-10, following the optimal settings in Song & Dhariwal (2024), we use a batch size of 1,024 with the student EMA decay rate of 0.99993, scale parameter in Fourier embedding layers of 0.02, and dropout rate of 0.3 for 400,000 iterations with RAdam optimizer (Liu et al., 2020) using learning rate 0.0001. We use the NCSN++ network architecture proposed by Song et al. (2021b), with the modification described in Section 3.1. In our network parameterization, we set $\sigma_{\text{data}} = 0.5$ following Karras et al. (2022) and Song et al. (2023). Regarding other training settings, including the scheduler function for $N(\cdot)$, the sampling probability for t_n (aka the noise schedule $p(n)$ in (Song et al., 2023; Song & Dhariwal, 2024)), the distance measure $d(\cdot, \cdot)$, we follow exactly Song & Dhariwal (2024), and restate below for

completeness's sake:

$$d(\mathbf{x},\mathbf{y}) = \sqrt{||x-y||^2 + c^2} - c, \quad c = 0.00054\sqrt{d}, \quad d \text{ is data dimensionality}, \tag{11}$$

$$p(n) \propto \text{erf}\left(\frac{\log(t_{n+1} - P_{\text{mean}})}{\sqrt{2}P_{\text{std}}}\right) - \text{erf}\left(\frac{\log(t_n - P_{\text{mean}})}{\sqrt{2}P_{\text{std}}}\right), \quad P_{\text{mean}} = -1.1, \quad P_{\text{std}} = 2.0, \tag{12}$$

$$N(k) = \min(s_0 2^{\lfloor k/K' \rfloor}, s_1) + 1, \quad s_0 = 10, \quad s_1 = 1280, \quad K' = \left\lfloor \frac{K}{\log_2[s_1/s_0] + 1} \right\rfloor, \tag{13}$$

K is the total training iterations

$$t_n = \left(t_{\text{min}}^{1/\rho} + \frac{n-1}{N(k)-1}\left(t_{\text{max}}^{1/\rho} - t_{\text{min}}^{1/\rho}\right)\right)^{\rho}, \quad t_{\text{min}} = 0.002, \quad t_{\text{max}} = 80, \quad \rho = 7. \tag{14}$$

We implement our model and training algorithm based on the codes released by Song et al. (2023) at `https://github.com/openai/consistency_models_cifar10` (Apache-2.0 License).

D.2 Sampling and Inversion Configurations

Here we provide hyperparameters for all the experiments on CIFAR-10 to reproduce the results in the main paper.

Sampling. For unconditional BCM, we use ancestral sampling with $t_1 = 1.2$ for NFE$= 2$, zigzag sampling with $\varepsilon_1 = 0.2, t_1 = 0.8$ for NFE$= 3$ and the combination of ancestral sampling and zigzag sampling with $t_1 = 1.2, \varepsilon_1 = 0.1, \tau_1 = 0.3$ for NFE$= 4$. For BCM-deep, we use ancestral sampling with $t_1 = 0.7$ for NFE$= 2$, zigzag sampling with $\varepsilon_1 = 0.4, t_1 = 0.8$ for NFE$= 3$ and the combination with $t_1 = 0.6, \varepsilon_1 = 0.14, \tau_1 = 0.3$ for NFE$= 4$.

Inversion. For both unconditional and conditional BCM, we set $\varepsilon = t_1 = 0.07$, $t_2 = 6.0$ and $t_3 = T = 80.0$ for NFE$= 2$ and $t_2 = 1.5, t_3 = 4.0, t_4 = 10.0, t_5 = T = 80.0$ for NFE$= 4$. These hyperparameters are tuned on 2,000 training samples, and we find them generalize well to all test images. We use 1-step generation to map the inverted noise to reconstructed images and evaluate the per-dimension MSE between the original images and their reconstructed counterparts.

We highlight that the hyperparameters are relatively robust for 2-step sampling/inversion, and the trend that the combination of ancestral and zigzag sampling is superior is also general. However, to achieve optimal performance with more steps, the tuning of each specific time step may require great effort. We should note that similar effort is also required in CMs, where Song et al. (2023) use ternary search to optimize the time steps.

Inversion Baselines. For DDIM, we use the reported MSE by Song et al. (2021a); for EDM, we load the checkpoint provided in the official implementation at `https://github.com/NVlabs/edm?tab=readme-ov-file` (CC BY-NC-SA 4.0 License), and re-implement a deterministic ODE solver following Algorithm 1 in (Karras et al., 2022).

D.3 Interpolation

We first invert the two given images $\mathbf{x}_1$ and $\mathbf{x}_2$ to noise at $T = 80.0$ using Algorithm 5. To avoid subscript overloading, in this section, we denote their noise as $\mathbf{z}_1$ and $\mathbf{z}_2$, respectively. Specifically, we find that adopting a 3-step inversion with $\varepsilon = t_1 = 0.07, t_2 = 1.5, t_3 = 6.0$, and $t_4 = T = 80.0$ is sufficient for good reconstruction results.

Since BCMs learn to amplify the two initial Gaussian i.i.d. noises, it is reasonable to hypothesize that the amplified noises (i.e., the embeddings) $\mathbf{z}_1$ and $\mathbf{z}_2$ reside on the same hyperspherical surface as if $\mathbf{z}_1$ and $\mathbf{z}_2$ are directly sampled from $\mathcal{N}(0, T^2\boldsymbol{I})$. Therefore, following Song et al. (2023), we use spherical linear interpolation as

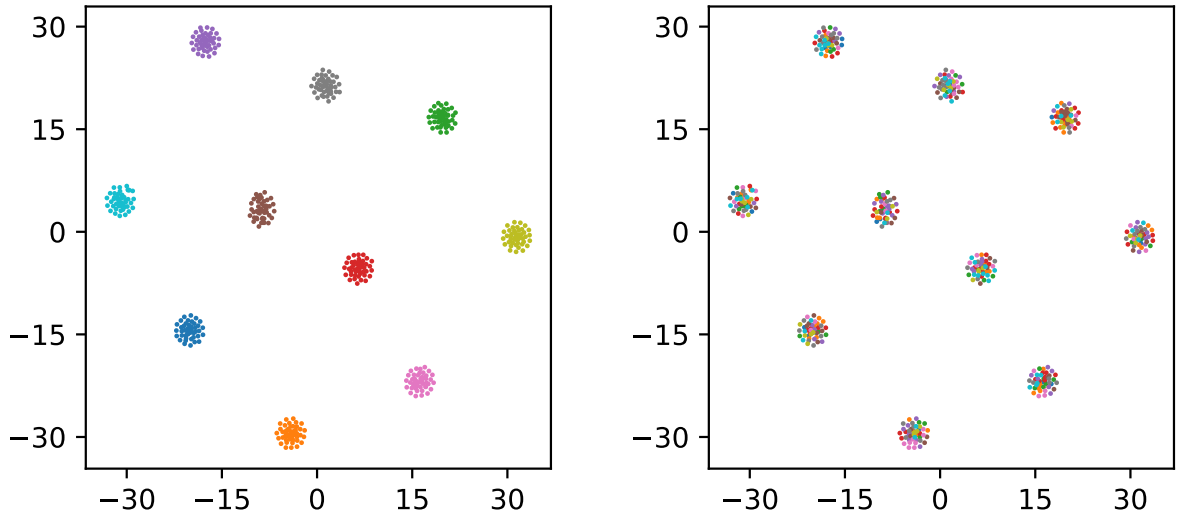
$$\mathbf{z} = \frac{\sin[(1-\alpha)\psi]}{\sin(\psi)}\mathbf{z}_1 + \frac{\sin[\alpha\psi]}{\sin(\psi)}\mathbf{z}_2, \tag{15}$$

in which $\alpha \in [0,1]$ and $\psi = \arccos\left(\frac{\mathbf{z}_1^T\mathbf{z}_2}{\|\mathbf{z}_1\|_2\|\mathbf{z}_2\|_2}\right)$.

Here, we note a caveat in the implementation of BCM's interpolation: recall that when inverting an image, we first inject a small initial noise, as described in Appendix A.3. In the context of interpolation, we find it crucial to inject different *initial noises for each of the two given images*, i.e., we sample $\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2 \sim \mathcal{N}(0, \boldsymbol{I}), \boldsymbol{\sigma}_1 \neq \boldsymbol{\sigma}_2$ when inverting $\mathbf{x}_1$ and $\mathbf{x}_2$ respectively. A possible reason is that if using $\boldsymbol{\sigma}_1 = \boldsymbol{\sigma}_2$ for inversion, the inverted noises $\mathbf{z}_1$ and $\mathbf{z}_2$ may reside on an unknown submanifold instead of the hyperspherical surface of Gaussian, and hence Equation (15) cannot yield ideal interpolation results. In Appendix E, we present visualization and discussions on the geometric properties of the noise space.

We include results by using the same initial noise in Figure 12. As we can see, the interpolated images are blurry compared to the results obtained using different initial noises (Figure 11).

E Understanding the Learned Noise ("Embedding") Space



(a) images inverted with the same initial noise are shown in the same color

(b) images with the same class label are shown in the same color

Figure 8. t-SNE of the inverted noise generated from 500 randomly selected CIFAR images.

This section provides some insights into the learned "embedding" space. Recall that during inversion (Algorithm 5), we first inject a small Gaussian noise to the image. Here we investigate the influence of this noise and the original image content on the noise generated by inversion.

We randomly select 500 CIFAR-10 images, and randomly split them into 10 groups. We then invert the images to their corresponding noise by Algorithm 5. We inject the *same* initial noise during inversion for images in the same group. Figure 8 visualize the t-SNE results (Van der Maaten & Hinton, 2008) of the inversion outcomes. In Figure 8a, images injected with the same initial noise are shown in the same color; while in Figure 8b, we color the points according to their class label (i.e., airplane, bird, cat, ...).

Interestingly, we can see that images inverted with the same initial noise are clustered together. We, therefore, conjecture that each initial noise corresponds to a submanifold in the final "embedding" space. The union of all these submanifolds constitutes the final "embedding" space, which is the typical set of $\mathcal{N}(0, T^2\boldsymbol{I})$, closed to a hypersphere. This explains why applying the same initial noise is suboptimal in interpolation, as discussed in Section 4.3 and Appendix D.3.

F Network Parameterization

In this section, we provide more details about our network parameterization design. To start with, recall that in Song et al. (2023), they parameterize the consistency model using skip connections as

$$\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) = c_{\text{skip}}(t)\mathbf{x}_t + c_{\text{out}}(t)F_{\boldsymbol{\theta}}(c_{\text{in}}(t)\mathbf{x}_t, t), \tag{16}$$

in which

$$c_{\text{in}}(t) = \frac{1}{\sqrt{\sigma_{\text{data}}^2 + t^2}}, \quad c_{\text{out}}(t) = \frac{\sigma_{\text{data}}(t-\varepsilon)}{\sqrt{\sigma_{\text{data}}^2 + t^2}}, \quad c_{\text{skip}}(t) = \frac{\sigma_{\text{data}}^2}{\sigma_{\text{data}}^2 + (t-\varepsilon)^2}, \tag{17}$$

that ensures

$$c_{\text{out}}(\varepsilon) = 0, \quad c_{\text{skip}}(\varepsilon) = 1 \tag{18}$$

to hold at some very small noise scale $\varepsilon \approx 0$ so $\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_0, \varepsilon) = \mathbf{x}_0$[3]. Since we expect the output is a noise image of target noise scale u, we expand the parameterization of $c_{\text{skip}}, c_{\text{out}}$ and c_{in} to make them related to u, as

$$\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, u) = c_{\text{skip}}(t, u)\mathbf{x}_t + c_{\text{out}}(t, u)F_{\boldsymbol{\theta}}(c_{\text{in}}(t, u)\mathbf{x}_t, t, u). \tag{19}$$

Our derivation of network parameterization shares the same group of principles in EDM (Karras et al., 2022). Specifically, we first require the input to the network $F_{\boldsymbol{\theta}}$ to have unit variance. Following Eq. (114) $\sim$ (117) in EDM paper (Karras et al., 2022), we have

$$c_{\text{in}}(t, u) = \frac{1}{\sqrt{\sigma_{\text{data}}^2 + t^2}}. \tag{20}$$

Then, as we discussed in the main text, we expect the model to achieve consistency in Equation (6) along the entire trajectory. According to Lemma 1 in Song et al. (2023), we have

$$\begin{aligned}
\nabla \log p_t(\mathbf{x}_t) &= \frac{1}{t^2}\left(\mathbb{E}[\mathbf{x}|\mathbf{x}_t] - \mathbf{x}_t\right) && (21)\\
&\overset{(i)}{\approx} \frac{1}{t^2}\left(\mathbf{x} - \mathbf{x}_t\right) && (22)\\
&= -\frac{\boldsymbol{\sigma}}{t}, && (23)
\end{aligned}$$

in which we follow Song et al. (2023) to estimate the expectation with $\mathbf{x}$ in (i). When u and t are close, we can use the Euler solver to estimate $\mathbf{x}_u$, i.e,

$$\begin{aligned}
\mathbf{x}_u &\approx \mathbf{x}_t - t(u - t)\nabla \log p_t(\mathbf{x}_t) && (24)\\
&= \mathbf{x}_t + (u - t)\boldsymbol{\sigma} && (25)\\
&= \mathbf{x} + u\boldsymbol{\sigma}. && (26)
\end{aligned}$$

Therefore, when u and t are close, and base on Song et al. (2023)'s approximation in (i), we can rewrite the consistency defined in Equation (6) as

$$\begin{aligned}
& d\left(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x} + t\boldsymbol{\sigma}, t, u), \mathbf{x} + u\boldsymbol{\sigma}\right) && (27)\\
=& \left|\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x} + t\boldsymbol{\sigma}, t, u) - (\mathbf{x} + u\boldsymbol{\sigma})\right| && (28)\\
=& \left|c_{\text{skip}}(t, u)(\mathbf{x} + t\boldsymbol{\sigma}) + c_{\text{out}}(t, u)F_{\boldsymbol{\theta}}(c_{\text{in}}(t, u)(\mathbf{x} + t\boldsymbol{\sigma}), t, u) - (\mathbf{x} + u\boldsymbol{\sigma})\right| && (29)\\
=& \left|c_{\text{out}}(t, u)F_{\boldsymbol{\theta}}(c_{\text{in}}(t, u)(\mathbf{x} + t\boldsymbol{\sigma}), t, u) - (\mathbf{x} + u\boldsymbol{\sigma} - c_{\text{skip}}(t, u)(\mathbf{x} + t\boldsymbol{\sigma}))\right| && (30)\\
=& \left|c_{\text{out}}(t, u)\right| \cdot \Bigg| F_{\boldsymbol{\theta}}(c_{\text{in}}(t, u)(\mathbf{x} + t\boldsymbol{\sigma}), t, u) \\
& \qquad - \frac{1}{c_{\text{out}}(t, u)}\left((1 - c_{\text{skip}}(t, u))\mathbf{x} + (u - c_{\text{skip}}(t, u)t)\boldsymbol{\sigma}\right)\Bigg|. && (31)
\end{aligned}$$

For simplicity, we set $d(\cdot, \cdot)$ to L_1 norm in Equation (28). Note that, in practice, for D-dimensional data, we follow Song & Dhariwal (2024) to use Pseudo-Huber loss $d(\mathbf{a}, \mathbf{b}) = \sqrt{||\mathbf{a} - \mathbf{b}||^2 + 0.00054^2 D} - 0.00054\sqrt{D}$, which can be well approximated by L_1 norm.

We should note that Equation (27) is based on the assumption that u and t are reasonably close. *This derivation is only for the pursuit of reasonable parameterization and should not directly serve as an objective function.* Instead, one should use the soft constraint we proposed in Equation (8) as the objective function.

[3]While the parameterization written in the original paper of Song et al. (2023) did not explicitly include $c_{\text{in}}(t)$, we find it is actually included in its official implementation at https://github.com/openai/consistency_models_cifar10/blob/main/jcm/models/utils.py#L189 in the form of Equations (16) and (17).

The approximate effective training target of network $F_{\boldsymbol{\theta}}$ is therefore

$$\frac{1}{c_{\text{out}}(t,u)}\left((1-c_{\text{skip}}(t,u))\mathbf{x}+(u-c_{\text{skip}}(t,u)t)\boldsymbol{\sigma}\right). \tag{32}$$

Following Karras et al. (2022), we require the effective training target to have unit variance, i.e.,

$$\text{Var}\left[\frac{1}{c_{\text{out}}(t,u)}\left((1-c_{\text{skip}}(t,u))\mathbf{x}+(u-c_{\text{skip}}(t,u)t)\boldsymbol{\sigma}\right)\right]=1, \tag{33}$$

so we have

$$\begin{aligned}
c_{\text{out}}^2(t,u) &= \text{Var}\left[(1-c_{\text{skip}}(t,u))\mathbf{x}+(u-c_{\text{skip}}(t,u)t)\boldsymbol{\sigma}\right] && (34)\\
&= (1-c_{\text{skip}}(t,u))^2\sigma_{\text{data}}^2+(u-c_{\text{skip}}(t,u)t)^2 && (35)\\
&= (\sigma_{\text{data}}^2+t^2)c_{\text{skip}}^2(t,u)-2(\sigma_{\text{data}}^2+tu)c_{\text{skip}}(t,u)+(\sigma_{\text{data}}^2+u^2), && (36)
\end{aligned}$$

which is a hyperbolic function of $c_{\text{skip}}(t,u)$. Following Karras et al. (2022), we select $c_{\text{skip}}(t,u)$ to minimize $|c_{\text{out}}(t,u)|$ so that the errors of $F_{\boldsymbol{\theta}}$ are amplified as little as possible, as

$$c_{\text{skip}}(t,u)=\arg\min_{c_{\text{skip}}(t,u)}|c_{\text{out}}(t,u)|=\arg\min_{c_{\text{skip}}(t,u)}c_{\text{out}}^2(t,u). \tag{37}$$

So we have

$$\begin{aligned}
(\sigma_{\text{data}}^2+t^2)c_{\text{skip}}(t,u) &= \sigma_{\text{data}}^2+tu && (38)\\
c_{\text{skip}}(t,u) &= \frac{\sigma_{\text{data}}^2+tu}{\sigma_{\text{data}}^2+t^2}. && (39)
\end{aligned}$$

Substituting Equation (39) into Equation (35), we have

$$\begin{aligned}
c_{\text{out}}^2(t,u) &= \frac{\sigma_{\text{data}}^2t^2(t-u)^2}{(\sigma_{\text{data}}^2+t^2)^2}+\left(\frac{\sigma_{\text{data}}^2t+t^2u}{\sigma_{\text{data}}^2+t^2}-u\right)^2 && (40)\\
&= \frac{\sigma_{\text{data}}^2t^2(t-u)^2+\sigma_{\text{data}}^4(t-u)^2}{(\sigma_{\text{data}}^2+t^2)^2} && (41)\\
&= \frac{\sigma_{\text{data}}^2(t-u)^2}{\sigma_{\text{data}}^2+t^2}, && (42)
\end{aligned}$$

and finally

$$c_{\text{out}}(t,u)=\frac{\sigma_{\text{data}}(t-u)}{\sqrt{\sigma_{\text{data}}^2+t^2}}. \tag{43}$$

One can immediately verify that when $u=t$, $c_{\text{skip}}(t,u)=1$ and $c_{\text{out}}(t,u)=0$ so that the boundary condition

$$\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_t,t,t)=\mathbf{x}_t \tag{44}$$

holds.

On the side of CMs, setting $u=\varepsilon$ will arrive at exactly the same form of $c_{\text{in}}(t,u)$ and $c_{\text{out}}(t,u)$ in Equation (17). While $c_{\text{skip}}(t,u)$ does not degenerate exactly to the form in Equation (17) when taking $u=\varepsilon$ and $t>\varepsilon$, this inconsistency is

negligible when $\varepsilon \approx 0$, as

$$\left|c_{\text{skip}}^{\text{BCM}}(t,\varepsilon) - c_{\text{skip}}^{\text{CM}}(t)\right| = \left|\frac{\sigma_{\text{data}}^2 + t\varepsilon}{\sigma_{\text{data}}^2 + t^2} - \frac{\sigma_{\text{data}}^2}{\sigma_{\text{data}}^2 + (t-\varepsilon)^2}\right| \tag{45}$$

$$= \frac{\left|\left(\sigma_{\text{data}}^2 + (t-\varepsilon)^2\right)\left(\sigma_{\text{data}}^2 + t\varepsilon\right) - \sigma_{\text{data}}^2\left(\sigma_{\text{data}}^2 + t^2\right)\right|}{\left(\sigma_{\text{data}}^2 + t^2\right)\left(\sigma_{\text{data}}^2 + (t-\varepsilon)^2\right)} \tag{46}$$

$$= \frac{\left|\varepsilon^2\sigma_{\text{data}}^2 - \varepsilon t\sigma_{\text{data}}^2 + \varepsilon t(t-\varepsilon)^2\right|}{\left(\sigma_{\text{data}}^2 + t^2\right)\left(\sigma_{\text{data}}^2 + (t-\varepsilon)^2\right)} \tag{47}$$

$$= \frac{\varepsilon(t-\varepsilon)\left|(t-\varepsilon)t - \sigma_{\text{data}}^2\right|}{\left(\sigma_{\text{data}}^2 + t^2\right)\left(\sigma_{\text{data}}^2 + (t-\varepsilon)^2\right)} \tag{48}$$

$$< \frac{\varepsilon(t-\varepsilon)\max\left\{(t-\varepsilon)t, \sigma_{\text{data}}^2\right\}}{\left(\sigma_{\text{data}}^2 + t^2\right)\left(\sigma_{\text{data}}^2 + (t-\varepsilon)^2\right)} \tag{49}$$

$$\leqslant \frac{\varepsilon(t-\varepsilon)\max\left\{t^2, \sigma_{\text{data}}^2\right\}}{\left(\sigma_{\text{data}}^2 + t^2\right)\left(\sigma_{\text{data}}^2 + (t-\varepsilon)^2\right)} \tag{50}$$

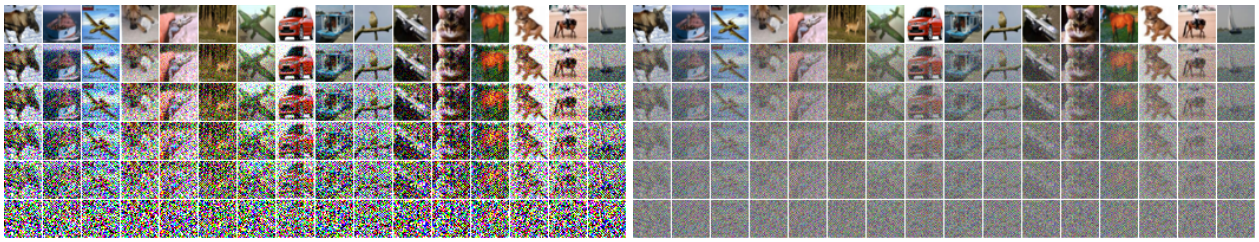
$$< \frac{\varepsilon(t-\varepsilon)}{\sigma_{\text{data}}^2 + (t-\varepsilon)^2} \tag{51}$$

$$= \frac{\varepsilon}{\frac{\sigma_{\text{data}}^2}{t-\varepsilon} + (t-\varepsilon)} \tag{52}$$

$$\leqslant \frac{\varepsilon}{2\sigma_{\text{data}}}. \tag{53}$$

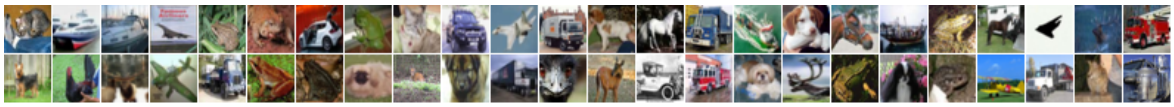
Therefore, we conclude that our parameterization is compatible with CM's parameterization, so with the same CT target of Equation (5), any CT techniques (Song et al., 2023; Song & Dhariwal, 2024) should directly apply to our model and it should inherit all properties from CMs just by setting $u = \varepsilon$, which is a clear advantage compared with models that adopt completely different parameterizations (e.g., CTM (Kim et al., 2024)).

G Additional Results

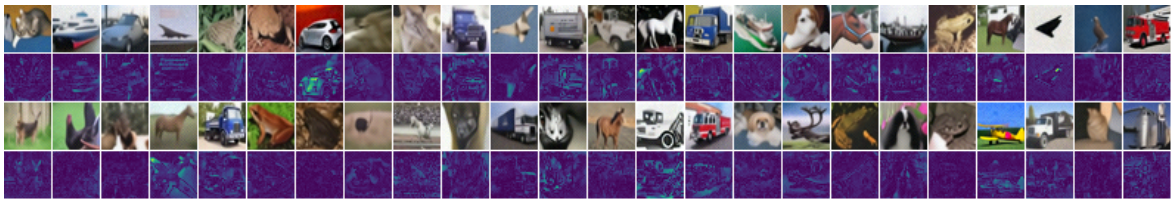


(a) (b)

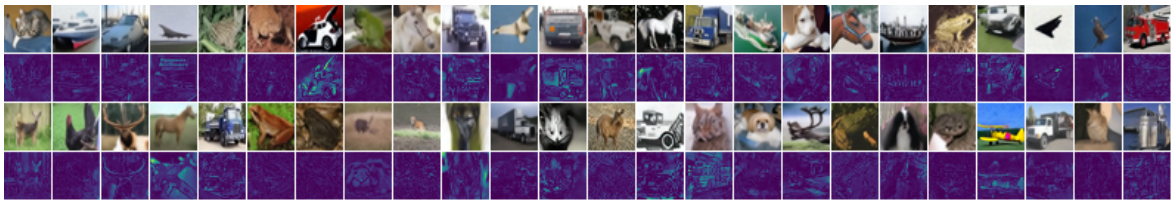
Figure 9. Visualization of the noise images generated by inversion with BCM. Each line corresponds to a noise scale of $0, 0.2, 0.5, 1.0, 2.0, 80.0$, respectively. In (a), we truncate the image to $[-1, 1]$, while in (b) we normalize the image to $[-1, 1]$.



(a) Ground Truth

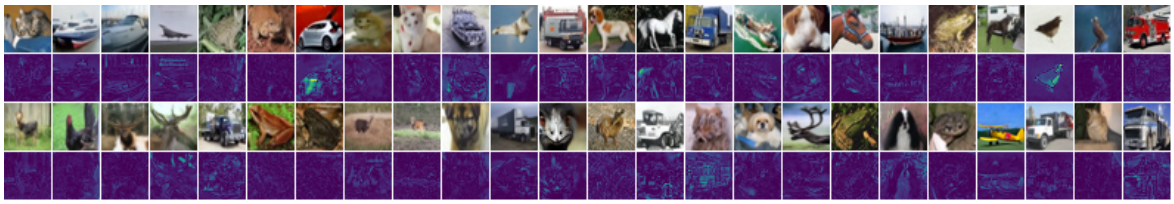


(b) BCM inversion and reconstruction with 1 NFE (MSE=0.00526)

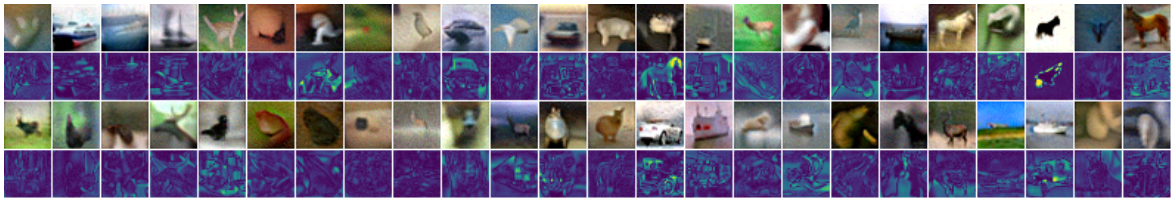


(c) BCM inversion and reconstruction with 2 NFE (MSE=0.00451)

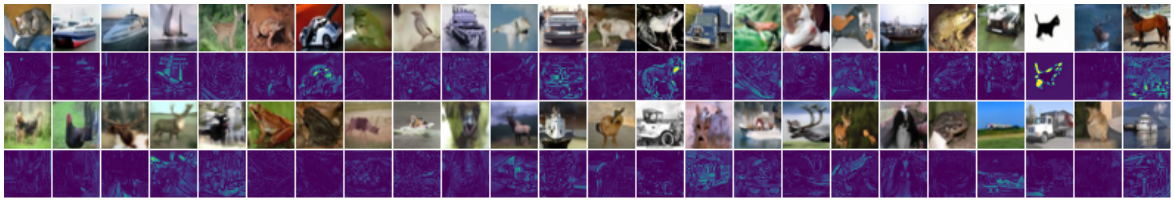
(d) BCM inversion and reconstruction with 3 NFE (MSE=0.00377)



(e) BCM inversion and reconstruction with 4 NFE (MSE=0.00362)



(f) EDM inversion and reconstruction with 9 NFE (MSE=0.01326)



(g) EDM inversion and reconstruction with 19 NFE (MSE=0.00421)

Figure 10. Reconstructed images and their residual with unconditional BCM on CIFAR-10. We include EDM's results in (e) and (f) for comparison.

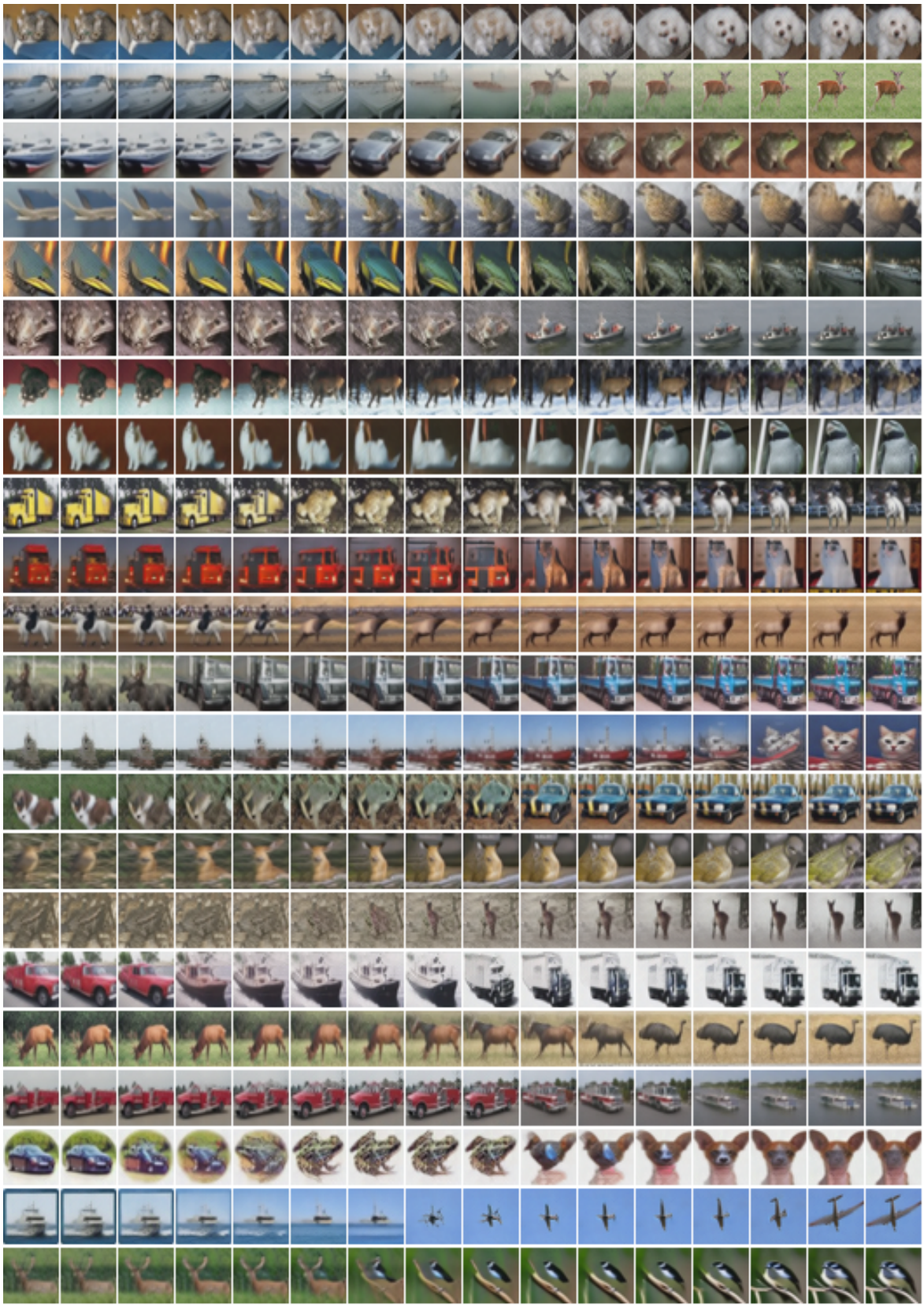


Figure 11. Interpolation between two real CIFAR-10 images (injecting *different* initial noise in inversion).

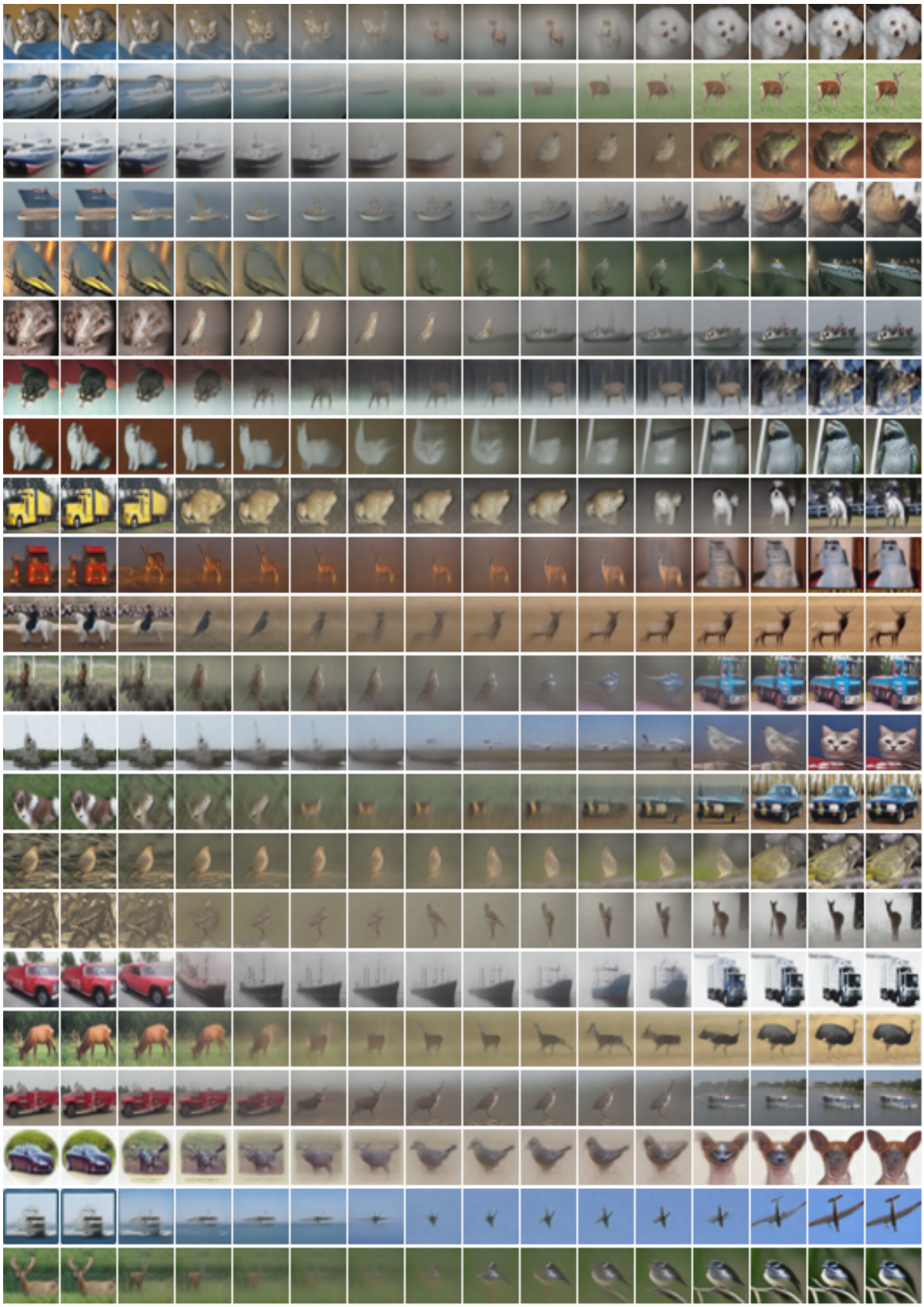
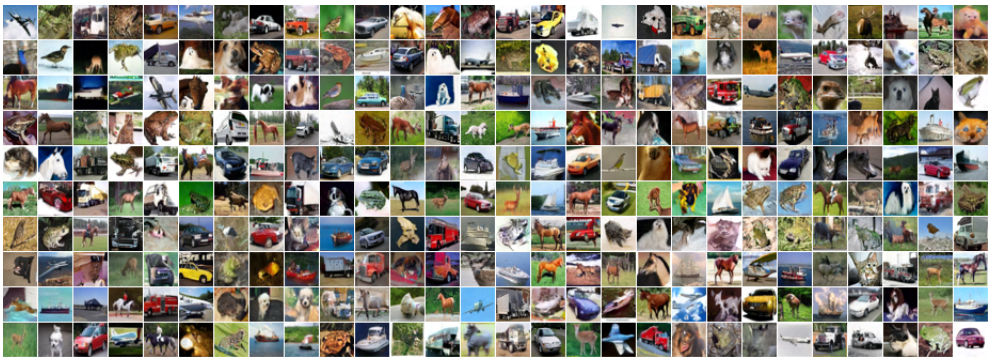
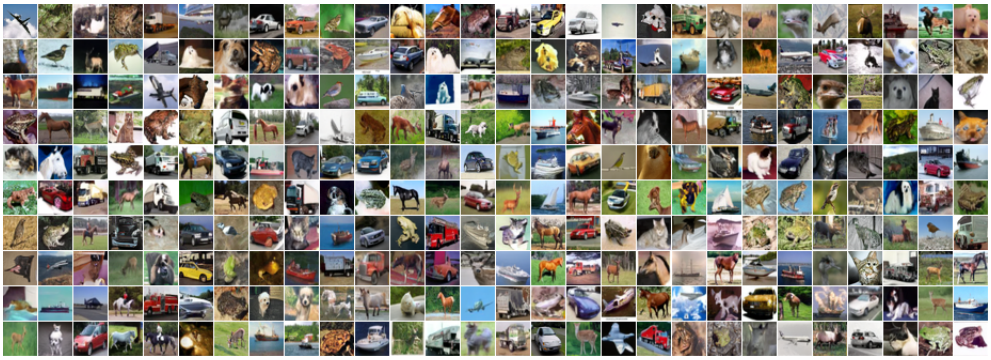


Figure 12. Interpolation between two real CIFAR-10 images (injecting the *same* initial noise in inversion).

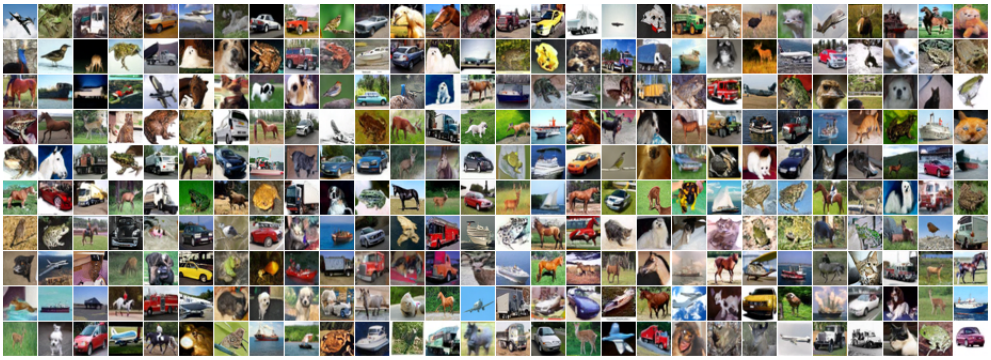
H Generation Results



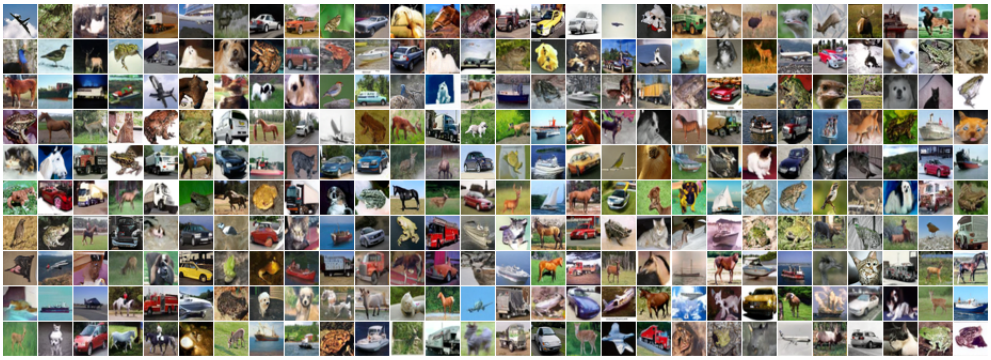
(a) 1-step generation (FID=3.10)



(b) 2-step generation (FID=2.39)

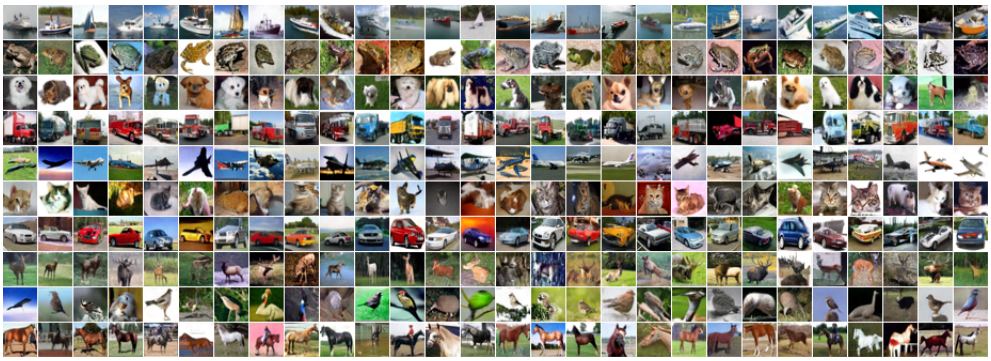


(c) 3-step generation (FID=2.50)

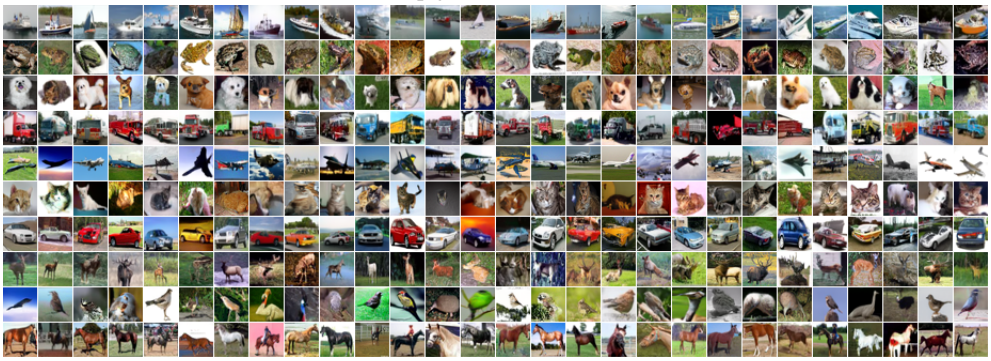


(d) 4-step generation (FID=2.29)

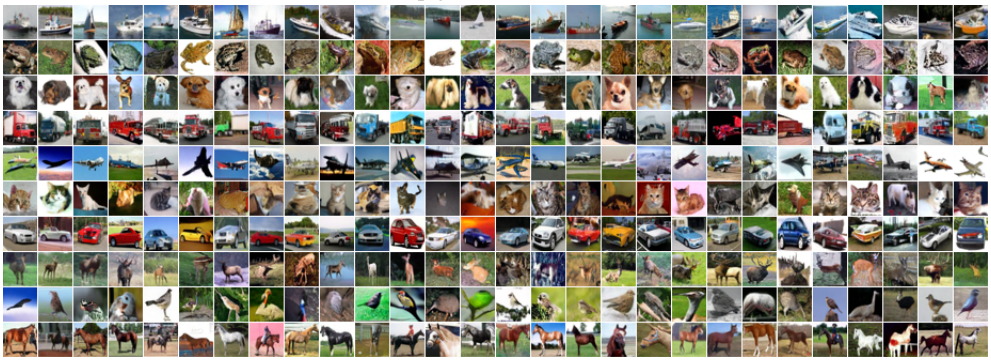
Figure 13. Uncurated CIFAR-10 samples generated by BCM. We can observe that our sampling methods improve the FID while largely preserving the content.



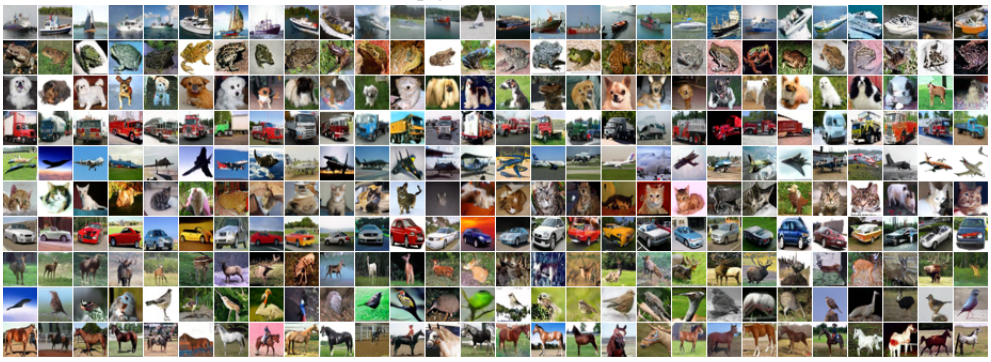
(a) 1-step generation (FID=2.68)



(b) 2-step generation (FID=2.44)

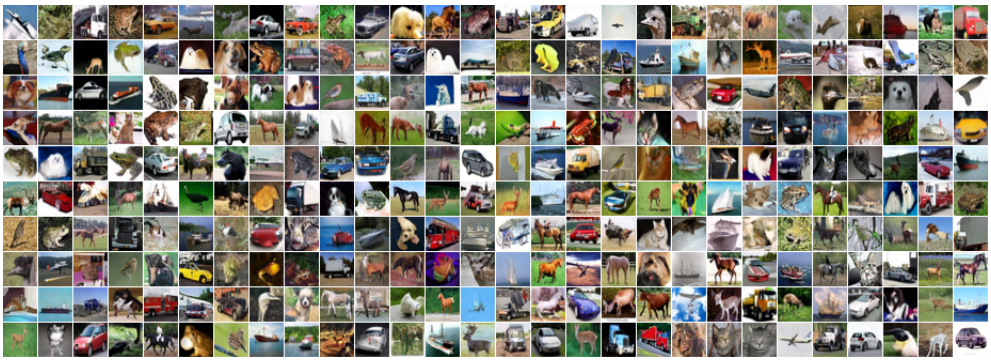


(c) 3-step generation (FID=2.28)

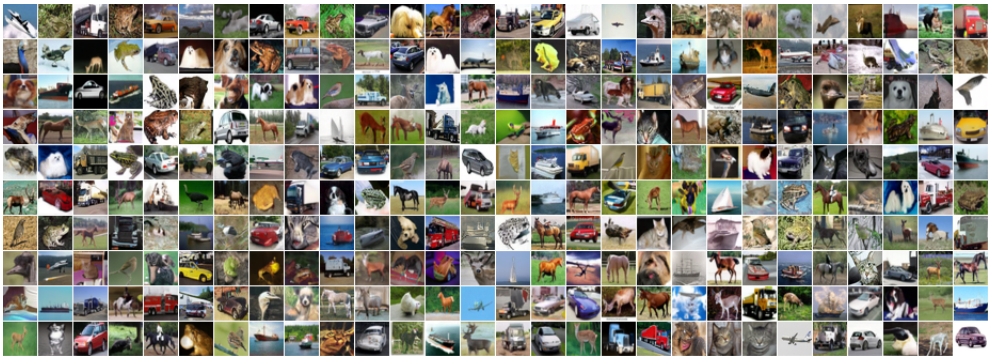


(d) 4-step generation (FID=2.20)

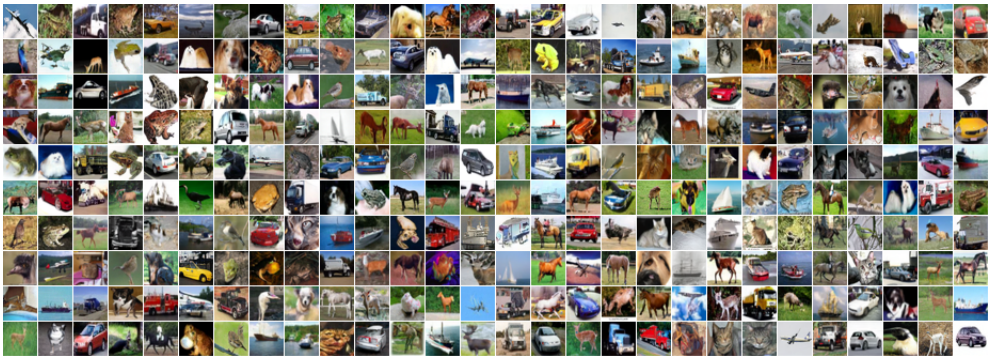
Figure 14. Uncurated CIFAR-10 samples generated by BCM-conditional. Each line corresponds to one class.



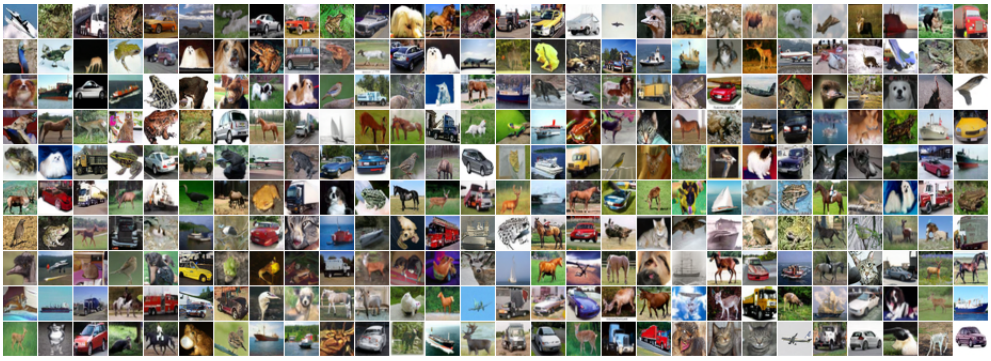
(a) 1-step generation (FID=2.64)



(b) 2-step generation (FID=2.36)



(c) 3-step generation (FID=2.19)



(d) 4-step generation (FID=2.07)

Figure 15. Uncurated CIFAR-10 samples generated by BCM-deep. We can observe that our sampling methods improve the FID while largely preserving the content.