TRAINING CONSISTENCY MODELS WITH VARIATIONAL NOISE COUPLING

Gianluigi Silvestri [*]
OnePlanet Research Center, imec-the Netherlands
Donders Institute for Brain, Cognition and Behaviou
Nijmegen, the Netherlands
gianluigi.silvestri@imec.nl
gianlu.silvestri@gmail.com

Luca Ambrogioni
Donders Institute for Brain, Cognition and Behaviou
Nijmegen, the Netherlands

Chieh-Hsin Lai & Yuhta Takida
Sony AI, Tokyo, Japan
{chieh-hsin.lai, yuta.takida}@sony.com

Yuki Mitsufuji
Sony AI, Tokyo, Japan
Sony Group Corporation

ABSTRACT

Consistency Training (CT) has recently emerged as a promising alternative to diffusion models, achieving competitive performance in image generation tasks. However, non-distillation consistency training often suffers from high variance and instability, and analyzing and improving its training dynamics is an active area of research. In this work, we propose a novel CT training approach based on the Flow Matching framework. Our main contribution is a trained noise-coupling scheme inspired by the architecture of Variational Autoencoders (VAE). By training a data-dependent noise emission model implemented as an encoder architecture, our method can indirectly learn the geometry of the noise-to-data mapping, which is instead fixed by the choice of the forward process in classical CT. Empirical results across diverse image datasets show significant generative improvements, with our model outperforming baselines and achieving the state-of-the-art (SoTA) non-distillation CT FID on CIFAR-10, and attaining FID on par with SoTA on ImageNet at 64×64 resolution in 2-step generation. Our code is available at https://github.com/sony/vct.

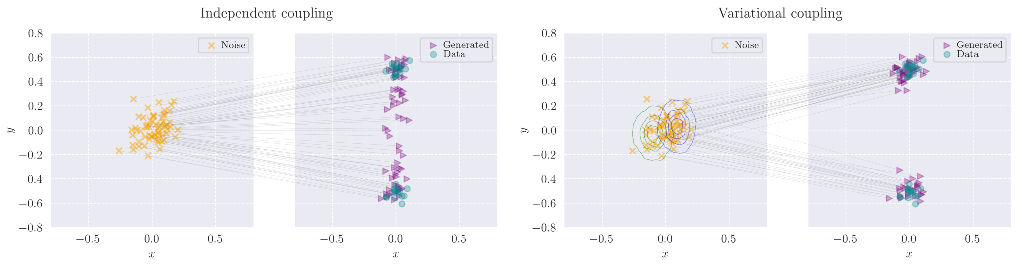


Figure 1: Comparison of 1-step generation on toy data for independent and variational coupling. The data is sampled from a 2-d mixture of Gaussians with means $\boldsymbol{\mu}_1 = (0, 0.5)$ and $\boldsymbol{\mu}_2 = (0, -0.5)$. On the right-hand side plot (our approach), we show the posterior probabilities learned by the encoder (in blue and green) corresponding to $p(\boldsymbol{z} \mid \boldsymbol{\mu}_1)$ and $p(\boldsymbol{z} \mid \boldsymbol{\mu}_2)$, and their cumulative sum approximately recovers the prior distribution. The gray lines connect the samples from the trained models to the corresponding input noise. More details about the toy experiments are given in Appendix G.

[*]Work done during an internship at Sony AI

1 INTRODUCTION

Generative Models are deep learning algorithms designed to learn the underlying probability distribution of a given dataset, in order to then generate samples coming from such a distribution. Some widely used models are Generative Adversarial Networks (Goodfellow et al., 2014), Variational Autoencoders (VAE; Kingma, 2013; Rezende et al., 2014), and Normalizing Flows (Chen et al., 2018; Papamakarios et al., 2021; Kobyzev et al., 2020). More recently, Diffusion Models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021b) have achieved state-of-the-art (SoTA) results in several domains, including images, videos, and audio (Dhariwal & Nichol, 2021; Rombach et al., 2022; Karras et al., 2022; 2024; Ho et al., 2022; Kong et al., 2021). However, a weakness of diffusion models is the need for an iterative sampling procedure, which can require hundreds of network evaluations. Therefore, substantial effort has been made to develop methods that can maintain similar generation quality while requiring fewer sampling iterations (Song et al., 2021a; Jolicoeur-Martineau et al., 2021; Salimans & Ho, 2022; Liu et al., 2022; Lu et al., 2022). Among such methods, a recent and promising direction is given by Consistency Models (CMs) (Song et al., 2023). CMs, while sharing many similarities with DMs, use a different training procedure as they directly learn the probability flow equations rather than the score function. CMs can be either trained by distilling the ODE trajectories of a pre-trained diffusion model (Consistency Distillation, CD), or completely from scratch through a bootstrap loss (Consistency Training, CT), which results in a novel generative modeling framework. However, the CT objective can be subject to high variance, making it difficult to train. A follow-up work (Song & Dhariwal, 2024) analyses the training dynamics of CMs and proposes several improvements which result in a more stable CT procedure, achieving SoTA results in few-step image generation. Since then, several works have proposed additional strategies to further improve CT (Geng et al., 2024; Wang et al., 2024; Lee et al., 2024b; Yang et al., 2024). A possible source of instability of CT training comes from the fact that different noise masks are applied to the same data point, creating ambiguity the target corresponding to the given noisy state, especially early during training with coarse discretization steps. From a more mathematical perspective, training can be destabilized by sharp boundaries in the ODE flow mapping, which can be hard to learn for the model and give raise to high variance in the stochastic gradient estimator. For example, the ODE flow for mixture of delta distributions is defined by a tessellation of the initial noise space, with discontinuities along all the borders. Since the standard CT approach with fixed forward process cannot alter the target ODE flow, the optimum of the standard CT training can potentially be highly singular. The existence of these singularities depend on topological reasons (i.e. non-injectivity of the ODE flow mapping at $t \to 0$ (Cornish et al., 2020)). However, the issue can likely be ameliorated by altering the forward process during training, which can be used to change the location of the singularities. A way to implement this approach is to sample noise using a conditional coupling function between data and noise. The concept of using a coupling function to reduce variance during training was successfully used in Flow Matching, with works such as (Pooladian et al., 2023; Tong et al., 2023; Lee et al., 2023), with the main objective of obtaining straighter ODE trajectories for faster sampling. Forms of coupling in CMs were proposed in works such as (Dou et al., 2024; Issenhuth et al., 2024), but their formulations do not match the performance of standard CMs. In this work, we introduce a variational training of the forward transition kernel with a coupling function between data and noise, which results in a loss function similar to the one used in VAEs. By learning a data-dependent probability distribution over the noise, regularized with an additional Kullback-Leibler divergence loss, we develop an end-to-end training procedure and show how the resulting coupling is effective at improving generation performance and is scalable to high-dimensional data. A simple and intuitive example is shown in Figure 1, where with the same settings, the model trained with the learned coupling generates samples closer to the data distribution. From this figure, we can see how the learned coupling partitions the data differently from how the standard CT does, effectively changing the form of the underlying ODE, likely resulting in an easier training objective as the prior noise is partitioned according to the learned the data-dependent coupling distribution.

The reminder of the paper is organized as follows: we first formulate CT from the Flow Matching perspective, which is a generalization of the diffusion framework and it offers a more natural way to introduce the noise-coupling distribution. Finally, we describe our method, deriving similarities with Variational Autoencoders, and report our experimental results on common image benchmarks. For the related relveant literature, we refer the reader to Appendix A.

2 BACKGROUND

Flow Matching provides a general framework that generalizes diffusion and score-matching models (Lipman et al., 2023; Albergo & Vanden-Eijnden, 2023). In the Flow Matching formalism, a deterministic flow function $\boldsymbol{\psi}_t$ with initial condition $\mathbf{x}_0$ is used to build an interpolating map between two distributions $\boldsymbol{\psi}_t(\boldsymbol{x}_0) = \boldsymbol{x}_t$, such that the data distribution $p_0(\boldsymbol{x}_0)$ is mapped into a distribution $p_1(\boldsymbol{x}_1)$, commonly chosen to be Gaussian noise distribution $p_1(\boldsymbol{x}_1) := \mathcal{N}(\boldsymbol{x}_1; \mathbf{0}, \boldsymbol{I})$, by the pushforward operator. From this quantity, we can define the vector field $\boldsymbol{u}_t(\boldsymbol{x}_t)$ as the infinitesimal generator of $\boldsymbol{\psi}_t$:

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\psi}_t(\boldsymbol{x}_0) = \boldsymbol{u}_t(\boldsymbol{x}_t) = \boldsymbol{u}_t(\boldsymbol{\psi}_t(\boldsymbol{x}_0)) \ ,$$

In a diffusion model, the flow $\boldsymbol{\psi}_t(\boldsymbol{x}_0)$ is the inverse of the probability ODE flow determined by the forward SDE. Instead, in standard Flow Matching, the mapping is specified as a conditional flow $\boldsymbol{\psi}_t(\boldsymbol{x}_0; \boldsymbol{x}_1)$, which is typically taken as a simple linear interpolation between samples from the two densities $p_0(\boldsymbol{x}_0)$ and $p_1(\boldsymbol{x}_1)$. Some common examples are $\boldsymbol{\psi}_t(\boldsymbol{x}_0; \boldsymbol{x}_1) = \boldsymbol{x}_t = (1-t)\boldsymbol{x}_0 + t\boldsymbol{x}_1$ as seen in (Lipman et al., 2023), and $\boldsymbol{\psi}_t(\boldsymbol{x}_0; \boldsymbol{x}_1) = \boldsymbol{x}_t = \boldsymbol{x}_0 + t\boldsymbol{x}_1$ as in (Karras et al., 2022). This conditional flow is analogous to the formal solution kernel of the forward process at time t in the generative diffusion framework. While it is difficult to directly obtain the flow function $\boldsymbol{\psi}_t(\boldsymbol{x}_0)$ from the conditional flow $\boldsymbol{\psi}_t(\boldsymbol{x}_0; \boldsymbol{x}_1)$, it is possible to give a formal expression for the resulting vector field:

$$\boldsymbol{u}_t(\boldsymbol{x}_t) = \mathbb{E}_{\boldsymbol{x}_1|\boldsymbol{x}_t}[\boldsymbol{u}_t(\boldsymbol{x}_t; \boldsymbol{x}_1)] \ , \tag{1}$$

where $\boldsymbol{u}_t(\boldsymbol{x}_t; \boldsymbol{x}_1) = \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\psi}_t(\boldsymbol{x}_0; \boldsymbol{x}_1)$ is the vector field that generates the conditional flow $\boldsymbol{\psi}_t(\boldsymbol{x}_0; \boldsymbol{x}_1)$. In the case of the simple interpolation conditional flows $\boldsymbol{\psi}_t(\boldsymbol{x}_0; \boldsymbol{x}_1) = (1-t)\boldsymbol{x}_0 + t\boldsymbol{x}_1$, the formula specializes as follows:

$$\boldsymbol{u}_t(\boldsymbol{x}_t) = \mathbb{E}_{\boldsymbol{x}_1|\boldsymbol{x}_t}[\boldsymbol{x}_1 - \boldsymbol{x}_0] \ . \tag{2}$$

Readers who are familiar with generative diffusion will immediately recognize that this expression is directly related to the standard expression for the score function. From this connection, it is clear that the conditional vector field can be estimated with a regression objective

$$\mathbb{E}_{t,\boldsymbol{x}_0,\boldsymbol{x}_1}||\boldsymbol{f}_\theta(\boldsymbol{\psi}_t(\boldsymbol{x}_0; \boldsymbol{x}_1), t) - \boldsymbol{u}_t(\boldsymbol{\psi}_t(\boldsymbol{x}_0; \boldsymbol{x}_1); \boldsymbol{x}_1)||_2^2.$$

2.1 NOISE COUPLING

An advantage of the Flow Matching formalism over SDE diffusion is that, as shown in (Pooladian et al., 2023; Tong et al., 2023), it is straightforward to introduce a probabilistic coupling $\pi(\boldsymbol{x}_1 \mid \boldsymbol{x}_0)$ between the data and the noise distribution. In this case, we require that $\int \pi(\boldsymbol{x}_1 \mid \boldsymbol{x}_0)\mathrm{d}\boldsymbol{x}_0$ should follow a standard normal distribution, at least approximately. The use of a non-trivial noise coupling does not alter the form of the conditional velocity fields $\boldsymbol{u}_t(\boldsymbol{x}_t; \boldsymbol{x}_1)$ as far as the coupling is time-independent. However, it does alter the total velocity field $\boldsymbol{u}_t(\boldsymbol{x}_t)$ since it affects the conditional distribution $p(\boldsymbol{x}_1, \boldsymbol{x}_t)$, which determines the expectation in Eq. 2.

3 CONTINUOUS CONSISTENCY MODELS FROM A FLOW MATCHING PERSPECTIVE

As explained above, the flow function $\boldsymbol{\psi}_t(\boldsymbol{x}_0)$ maps a noiseless state $\boldsymbol{x}_0$ to the noisy state $\boldsymbol{x}_t$. Its inverse $\boldsymbol{\psi}_t^{-1}(\boldsymbol{x}_t)$ can then be interpreted as a denoiser, as it maps each noisy state to a uniquely defined noiseless state $\boldsymbol{x}_0$. This function is often referred to as a consistency map, and it follows the identity:

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\psi}_t^{-1}(\boldsymbol{\psi}_t(\boldsymbol{x}_0)) = 0, \quad \text{on} \quad t \in [0, 1], \tag{3}$$

together with the boundary condition $\boldsymbol{\psi}_0^{-1}(\boldsymbol{x}_0) = \boldsymbol{x}_0$. Eq. 3 is a consequence of the fact that all noisy states in an ODE trajectory $\boldsymbol{\psi}_t(\boldsymbol{x}_0)$ share the same initial point $\boldsymbol{x}_0$, which implies that $\boldsymbol{\psi}_t^{-1}(\boldsymbol{\psi}_t(\boldsymbol{x}_0))$

is constant along the trajectory. This property can be used to define a continuous loss for a network $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$, trained to approximate the inverse flow $\boldsymbol{\psi}_0^{-1}(\boldsymbol{x}_t)$:

$$\mathcal{L}_{\text{cont}}^{\text{tot}}(\boldsymbol{\theta}) \equiv \mathbb{E}_{\boldsymbol{x}_0}\left[\int_0^1 \lambda(t) \left\| \frac{\mathrm{d}}{\mathrm{d}t} \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{\psi}_t(\boldsymbol{x}_0), t) \right\|^2 \mathrm{d}t\right] \tag{4}$$

where $\lambda(t)$ is a positive-valued function that weights the loss for different time points. This loss should be used together with the identity boundary condition $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_0, 0) = \boldsymbol{x}_0$, which we will discuss later. In distillation training, the deterministic trajectories $\boldsymbol{x}_t = \boldsymbol{\psi}_t(\boldsymbol{x}_0)$ are obtained by integrating the ODE flow obtained from a pre-trained diffusion or flow matching model. Alternatively, the consistency network can be trained directly by re-writing the total derivative in terms of the conditional flow:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\psi}_t^{-1}(\boldsymbol{\psi}_t(\boldsymbol{x}_0)) &= \nabla \boldsymbol{\psi}_t^{-1}(\boldsymbol{x}_t)\frac{\mathrm{d}x_t}{\mathrm{d}t} + \partial_t \boldsymbol{\psi}_t^{-1}(\boldsymbol{x}_t) = \nabla \boldsymbol{\psi}_t^{-1}(\boldsymbol{x}_t)\mathbb{E}_{\boldsymbol{x}_1|\boldsymbol{x}_t}[\boldsymbol{u}_t(\boldsymbol{x}_t;\boldsymbol{x}_1)] + \partial_t \boldsymbol{\psi}_t^{-1}(\boldsymbol{x}_t) \\ &= \mathbb{E}_{\boldsymbol{x}_1|\boldsymbol{x}_t}\left[\nabla \boldsymbol{\psi}_t^{-1}(\boldsymbol{x}_t)\boldsymbol{u}_t(\boldsymbol{x}_t;\boldsymbol{x}_1) + \partial_t \boldsymbol{\psi}_t^{-1}(\boldsymbol{x}_t)\right] = \mathbb{E}_{\boldsymbol{x}_1|\boldsymbol{x}_t}\left[\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\psi}_t^{-1}(\boldsymbol{\psi}_t(\boldsymbol{x}_0;\boldsymbol{x}_1))\right] .\end{aligned} \tag{5}$$

From this equality, together with the fact that the squared Euclidean norm is a convex function, it follows that

$$\mathcal{L}_{\text{cont}}^{\text{tot}}(\boldsymbol{\theta}) \leq \mathcal{L}_{\text{cont}}^{\text{cond}}(\boldsymbol{\theta}) \ , \quad \text{with} \quad \mathcal{L}_{\text{cont}}^{\text{cond}}(\boldsymbol{\theta}) \equiv \mathbb{E}_{\boldsymbol{x}_1, \boldsymbol{x}_0}\left[\int_0^1 \lambda(t) \left\| \frac{\mathrm{d}}{\mathrm{d}t} \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{\psi}_t(\boldsymbol{x}_0;\boldsymbol{x}_1), t) \right\|^2 \mathrm{d}t\right] \ ,$$

where we moved the expectation outside of the squared norm using Jensen's inequality. Therefore, we can optimize the tractable "conditional loss" $\mathcal{L}_{\text{cont}}^{\text{cond}}(\boldsymbol{\theta})$ instead of $\mathcal{L}_{\text{cont}}^{\text{tot}}(\boldsymbol{\theta})$, which contains the unknown flow function $\boldsymbol{\psi}_t(\boldsymbol{x}_0)$.

4 Discretized consistency training

The continuous loss can be directly minimized in expectation by sampling the time t from a uniform distribution and by computing the total derivative $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{\psi}_t(\boldsymbol{x}_0;\boldsymbol{x}_1), t)$ by automatic differentiation. However, in practice it is often more convenient to instead optimize a time-discretized loss with a finite difference approximation for the total derivative:

$$\begin{aligned}\mathcal{L}_{\text{disc}}^{\text{cond}}(\boldsymbol{\theta}) &= \sum_{i=1}^{N} \lambda(t_i)\mathbb{E}_{\boldsymbol{x}_0\sim p_0(\boldsymbol{x}_0), \boldsymbol{x}_1\sim\pi(\boldsymbol{x}_1|\boldsymbol{x}_0)}\left[||\Delta \boldsymbol{f}_{\boldsymbol{\theta}}||^2\right] \ , \\ \text{with} \quad \Delta \boldsymbol{f}_{\boldsymbol{\theta}} &= \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{\psi}_{t_{i+1}}(\boldsymbol{x}_0;\boldsymbol{x}_1), t_{i+1}) - \boldsymbol{f}_{\boldsymbol{\theta}^-}(\boldsymbol{\psi}_{t_i}(\boldsymbol{x}_0;\boldsymbol{x}_1), t_i) \ .\end{aligned} \tag{6}$$

where $\boldsymbol{x}_1 \mid \boldsymbol{x}_0$ are sampled according to the noise-coupling $\pi(\boldsymbol{x}_1 \mid \boldsymbol{x}_0)$. In this expression, $\boldsymbol{\theta}^-$ denotes a frozen copy of the parameters which does not require gradients. This loss is in fact unbiased for $\Delta t \to 0$, as it was shown in Song et al. (2023). The boundary condition can be enforced through the parametrization introduced in (Karras et al., 2022):

$$\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}, t) = c_{\text{skip}}(t)\boldsymbol{x} + c_{\text{out}}(t)\boldsymbol{F}_{\boldsymbol{\theta}}(\boldsymbol{x}, t),$$

where $\boldsymbol{F}_{\boldsymbol{\theta}}$ is a neural network and c_{skip} and c_{out} are specified such that $c_{\text{skip}}(0) = 1$ and $c_{\text{out}}(0) = 0$.

5 Consistency Models with learned variational coupling

Our method consists in learning a conditional coupling $q_{\boldsymbol{\phi}}(\boldsymbol{x}_1 \mid \boldsymbol{x}_0)$ with a neural network $\boldsymbol{g}_{\boldsymbol{\phi}}(\boldsymbol{x})$ parametrized by $\boldsymbol{\phi}$, which we refer to as the encoder in the following given its analogy with VAEs. During training, we can sample noise conditionally from $\pi(\boldsymbol{x}_1 \mid \boldsymbol{x}_0) = q_{\boldsymbol{\phi}}(\boldsymbol{x}_1 \mid \boldsymbol{x}_0)p_0(\boldsymbol{x}_0)$ instead of the independent noise commonly used in CT, and obtain noisy states for a given time step t as follows:

$$\boldsymbol{x}_t = \boldsymbol{\psi}_t(\boldsymbol{x}_0;\boldsymbol{x}_1), \qquad \boldsymbol{x}_1 \sim q_{\boldsymbol{\phi}}(\boldsymbol{x}_1 \mid \boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_1; \boldsymbol{g}_{\boldsymbol{\phi}}^{\mu}(\boldsymbol{x}_0), \ \boldsymbol{g}_{\boldsymbol{\phi}}^{\sigma}(\boldsymbol{x}_0)^2\mathbf{I}),$$

where we express the corresponding coupled noise $\boldsymbol{x}_1$ using the Gaussian reparameterization formula:

$$\boldsymbol{x}_1 = \boldsymbol{g}^{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}_0) + \ \boldsymbol{g}^{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x}_0) \cdot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \boldsymbol{I})$$

Here, we restricted our attention to linear forward models of the form $\boldsymbol{\psi}_t(\boldsymbol{x}_0; \boldsymbol{x}_1) = a_t \boldsymbol{x}_0 + b_t \boldsymbol{x}_1$, which encompasses most models used in the diffusion and flow-matching literature. Moreover $\boldsymbol{g}^{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}_0)$ and $\boldsymbol{g}^{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x}_0)$ denote the mean and scale output of the encoder, which define the signal-noise coupling (see Appendix H for a visual representation). Both $\boldsymbol{g}^{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}_0)$ and $\boldsymbol{g}^{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x}_0)$ preserve the same dimensionality of the input signal. The encoder network that produces the coupling distribution $q_{\boldsymbol{\phi}}(\boldsymbol{x}_1 \mid \boldsymbol{x}_0)$ can be trained end-to-end alongside the consistency model (see Appendix B). This results in a joint optimization where the consistency network adjusts its constancy to minimize its total derivative along the trajectories while the encoder implicitly moves the trajectories towards the space of constancy of the model. In fact, the velocity field $\boldsymbol{u}_t(\boldsymbol{x}_t)$ depends on the coupling, since $\pi(\boldsymbol{x}_1 \mid \boldsymbol{x}_0)$ affects the expectation in Eq. (2). This formulation results in a viable generative model as long as the noise at time 1 remains approximately $p_1(\boldsymbol{x}_1)$, since severe deviation from the prior induced by the coupling would result in improper initialization for the sampling procedure and consequently in reduced sample quality. We therefore add a Kullback-Leibler divergence as a regularizer, $\mathcal{D}_{\mathrm{KL}}(q_{\boldsymbol{\phi}} || p_1)$, resulting in a loss resembling the Evidence Lower Bound loss of Variational Autoencoders Kingma (2013):

$$\tilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathcal{L}^{\text{cond}}_{\text{disc}}(\boldsymbol{\theta}, \boldsymbol{\phi}) + \mathbb{E}_{\boldsymbol{x}_0}[\mathcal{D}_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{x}_1 \mid \boldsymbol{x}_0) || \mathcal{N}(\boldsymbol{x}_1; \mathbf{0}, \mathbf{I}))] \,. \tag{7}$$

While using an encoder to learn the data-noise coupling requires additional computation during training, we empirically find that a relatively small encoder is enough to learn an effective coupling, which results only in a minor increase of training time (see Appendix F.2). At sampling, the speed and computational requirements are identical to vanilla CMs for the one-step procedure, while for multistep sampling we need to account for additional forward passes of the encoder (see Appendix B).

5.1 Connection with variational autoencoders

In this section, we will consider the special case with constant unit time weighting $\lambda_{\text{ct}}(t) = 1, \forall t$.

ELBO perspective. First, we demonstrate the relationship between our model and VAE in terms of their objective functions. Specifically, our loss function in Eq. 7 serves as an upper bound on a standard VAE loss, where the latent vector $\boldsymbol{x}_1$ is regularized to be close to the prior p_1. Using the triangle inequality and the Cauchy-Schwarz inequality, we can establish the following bound for $\mathcal{L}^{\text{cond}}_{\text{disc}}$:

$$\|\boldsymbol{x}_0 - \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_1, 1)\|^2 \leq N \sum_{i=0}^{N} \left\| \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{\psi}_{t_{i+1}}(\boldsymbol{x}_0; \boldsymbol{x}_1), t_{i+1}) - \boldsymbol{f}_{\boldsymbol{\theta}^-}(\boldsymbol{\psi}_{t_i}(\boldsymbol{x}_0; \boldsymbol{x}_1), t_i) \right\|^2 . \tag{8}$$

Given that the KL terms in Eq. 7 and the VAE loss are identical, our loss function serves as an upper bound on the loss of a VAE with an encoder $(\boldsymbol{g}^{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}_0), \boldsymbol{g}^{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x}_0))$ and a prior $\mathcal{N}(\boldsymbol{x}_1; \mathbf{0}, \mathbf{I})$. Since the VAE loss corresponds to an evidence lower bound (ELBO), the consistency loss similarly provides a lower bound on the model evidence. Furthermore, we establish a connection between Eq. 8 and the Continuous-time CM (Lu & Song, 2024); additional details are provided in Appendix D. Compared to traditional VAEs, our method can be viewed as a time-dependent modification where the transition kernel smoothly interpolates between delta distributions centered at datapoints and a Gaussian distribution.

Varying β. In both our model and VAE, the latent vector needs to approximately follow a normal distribution to avoid deviating from the prior. However, previous studies (Hoffman & Johnson, 2016; Rosca et al., 2018; Aneja et al., 2021) have observed that VAE's aggregated posterior fails to match the prior. The same problem could occur in our model without additional tricks (see Sec. E). To mitigate this prior-posterior mismatch, we introduce a scalar hyperparameter β to control the strength of the KL regularization. This was first introduced by Higgins et al. (β-VAE 2022) for a different purpose (inducing disentangled latent representation) in the VAE context.

$$\min_{\boldsymbol{\theta}, \boldsymbol{\phi}} \mathcal{L}^{\text{cond}}_{\text{disc}}(\boldsymbol{\theta}, \boldsymbol{\phi}) + \beta \mathbb{E}_{\boldsymbol{x}_0}[\mathcal{D}_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{x}_1 \mid \boldsymbol{x}_0) || \mathcal{N}(\boldsymbol{x}_1; \mathbf{0}, \mathbf{I}))]. \tag{9}$$

By carefully selecting the value of β, we can achieve a proper balance between flexibility and proximity to the prior. To understand the effect of β in our model, we present an alternative form of our objective function. Formally, we can view the minimization of Eq. 9 as the relaxed Lagrangian problem of the following optimization problem:

$$\min_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathcal{L}^{\text{cond}}_{\text{disc}}(\boldsymbol{\theta},\boldsymbol{\phi}) \quad \text{s.t.} \quad \mathcal{D}_{\text{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{x}_1 \mid \boldsymbol{x}_0)||\mathcal{N}(\boldsymbol{x}_1;\mathbf{0},\mathbf{I})) < \delta,$$

As δ approaches 0, the coupling becomes $\pi(\boldsymbol{x}_1 \mid \boldsymbol{x}_0) = p_0(\boldsymbol{x}_0)p_1(\boldsymbol{x}_1)$, indicating that our model encompasses the standard CT. In VAE, selecting appropriate values of β to achieve reasonable generation performance is generally challenging. Values too close to zero result in strong deviation from the prior, while extremely large values cause over-smoothed decoders (Takida et al., 2022), leading to blurry samples. However, our model does not suffer from this over-regularization issue. While tuning β remains crucial in our model, as demonstrated in Section E, unlike VAE, increasing values of β does not cause the oversmoothing problem but simply reduces our model to the standard CT. Consequently, the CT training objective enables sharp sample generation even when the posterior approximation is nearly a normal distribution.

5.2 Choosing the β parameter

The most common weighting function $\lambda_{\text{ct}}(t)$ for CMs is an adaptive weighting scheme that changes over training based on how fine grained the discretization is, or equivalently the distance between consecutive time steps in ECM. At a given training iteration, for two adjacent time steps t_i and t_{i+1}, we have:

$$\lambda_{\text{ct}}(t_{i+1}) = \frac{1}{\Delta_{t_{i+1}}}, \quad \Delta_{t_{i+1}} = t_{i+1} - t_i$$

For the models trained with such an adaptive weighting function, we found it hard to tune β to a single scalar value. The magnitude of the weights increases during training as the discretization scheme becomes more fine-grained and $\Delta_{t_{i+1}}$ becomes smaller, changing the balance between consistency loss and KL regularization, resulting in a very strong regularization at the early stages of training, or a too weak one at the later stages. A simple yet effective solution is to use an adaptive scaling for the KL regularization that changes according to the discretization scheme. To do so, we take as a reference the weighting of the consistency loss at the last step $t_N = \sigma_{\max}$, and define the adaptive KL weighting as:

$$\lambda_{\text{kl}} = \beta\lambda_{\text{ct}}(t_N) = \frac{\beta}{\Delta_{t_N}}.$$

This way, we only need to specify the scalar hyperparameter β, and it will have a consistent regularization strength over training, as it increases whenever the discretization scheme is changed, which reflects in Δ_{t_N} becoming smaller. For ECM models trained on ImageNet, which use the EDM-style weighting function $\lambda_{\text{ct}}(t) = 1/t^2 + 1/\sigma^2_{\text{data}}$, we simply select a fixed β scalar for the whole training, as the discretization scheme does not affect the magnitude of the weights.

6 Experiments

In the following, we show that learning the data-noise coupling with our method is a simple yet effective improvement for CT. We pair our Variational Coupling with two established methods as baselines, namely improved Consistency Training (iCT) from (Song & Dhariwal, 2024) and Easy Consistency Tuning (ECM) from (Geng et al., 2024). Note that for the latter, the model is initialized with the weights of a pretrained score model from (Karras et al., 2022), while in the former the weights are initialized at random. More details about the baselines are provided in Appendix F.1. In this work, we consider only the framework of CT, where the unbiased vector field estimator $\boldsymbol{u}_t(\boldsymbol{x}_t)$ from equation 2 is used to approximate the noisy states $\boldsymbol{x}_t$ during training, as opposed to Consistency Distillation that uses a pretrained score model as a teacher. We evaluate the models on the image datasets Fashion-MNIST (Xiao et al., 2017), CIFAR-10 (Krizhevsky et al., 2009), FFHQ 64×64 (Karras, 2019) and (class-conditional) ImageNet 64×64 (Deng et al., 2009). To learn the coupling, we add a smaller version of the neural network used for CT, without time conditioning and with weights always initialized at random. For all the models, we use the variance exploding transition kernel (iCT-VE and ECM-VE) used in (Karras et al., 2022) and (Song & Dhariwal, 2024), with $a_t = 1$, $b_t = t$, and the linear interpolation kernel (iCT-LI and ECM-LI) commonly used in Flow Matching (Lipman et al., 2023), with $a_t = 1 - t/\sigma_{\max}$ and $b_t = t/\sigma_{\max}$ (details in Appendix C). For both kernels, we set $\sigma_{\min} = 0.002$ and $\sigma_{\max} = 80$. More experimental details can be found in Appendix F.2, while samples obtained with our best models are shown in I.

6.1 Baselines and models

As baselines, we re-implement the iCT and ECM models, corresponding to our iCT-VE and ECM-VE. As an additional model, we add CT with the minibatch Optimal Transport Coupling (-OT) proposed in (Pooladian et al., 2023; Tong et al., 2023) and used in (Issenhuth et al., 2024), to compare the effectiveness and scalability of the OT coupling with the learned one. Finally, we combine the baselines with our proposed Variational Coupling (-VC). For the models with learned coupling, we use gradient clipping with a large value (200 in all the experiments) to avoid instabilities at the early stages of training.

Model	Fashion-MNIST	CIFAR10
iCT-VE*	-	**2.83** / 2.46
iCT-VE	4.79 / 3.54	3.61 / 2.79
iCT-LI	4.75 / 3.46	3.81 / 2.87
iCT-VE-OT	4.42 / 2.82	3.28 / 2.66
iCT-LI-OT	4.41 / 2.91	3.42 / 2.77
iCT-VE-VC (ours)	3.88 / 2.37	2.86 / **2.32**
iCT-LI-VC (ours)	**3.62 / 2.22**	2.94 / **2.32**

Table 1: FID (1-step / 2-step) for iCT-based Models. * Baseline from (Song & Dhariwal, 2024). Ours are re-implementations. Bold indicates best.

Model	CIFAR10	FFHQ (64×64)	ImageNet (64×64)
ECM-VE*	3.60 / 2.11	-	5.51† / 3.18†
ECM-VE	3.68 / 2.14	5.99 / 4.39	5.26 / 3.22
ECM-LI	3.65 / 2.14	6.42 / 4.73	5.13 / 3.20
ECM-VE-OT	3.46 / 2.13	6.11 / 4.68	6.02 / 4.27
ECM-LI-OT	3.49 / 2.13	6.19 / 4.73	5.63 / 4.09
ECM-VE-VC (ours)	**3.26 / 2.02**	**5.47 / 4.16**	5.08 / 3.15
ECM-LI-VC (ours)	3.39 / 2.09	5.57 / 4.29	**4.93 / 3.07**

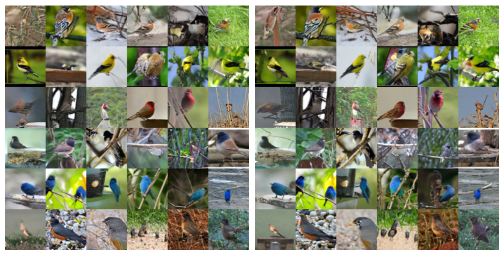
Table 2: Comparison of FID (lower is better, reported as 1-step / 2-step performance) for different models based on ECM. The model marked with a $*$ is the baseline as reported in (Geng et al., 2024). All the other models are from our re-implementation. The best entries are highlighted in bold. For ImageNet, the results marked with † are obtained with models trained for $100k$ iterations, while the others use $200k$ iterations.

6.2 Results

In tables 1 and 2 we report the 1 and 2 step sample quality evaluated with Frechet Inception Distance (FID) (Heusel et al., 2017), for both the results reported in the original papers and our re-implementations. For high-dimensional data, we only use models based on ECM, as they require lower computational budget, while for FashionMNIST we only use models based on iCT as there is no available pretrained EDM model.

FashionMNIST: We choose FashionMNIST as a first benchmark to test the performance of iCT. On this dataset, we use a small version of DDPM++, with 64 model channels instead of 128 and no attention, and batch size 128. Our variant with Variational Coupling outperforms both iCT and iCT-OT, with best performance obtained with the LI transition kernel, showing the benefit of the learned coupling.

CIFAR-10: For all the CIFAR-10 experiments, we use the DDPM++ architecture from (Song et al., 2021b) as implemented in (Karras et al., 2022), with EMA rate 0.9999 as in (Geng et al., 2024).


(a) 1-step (FID=5.13, left) and 2-step (FID=3.20, right) samples from ECM-LI.

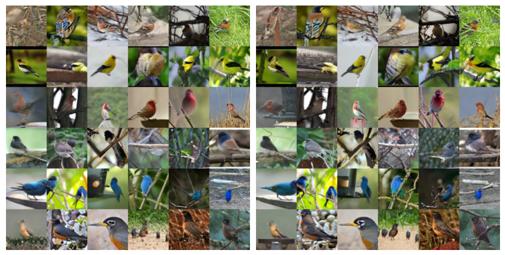

(b) 1-step (FID=4.93, left) and 2-step (FID=3.07, right) samples from ECM-LI-VC.

Figure 2: Visual comparison of generated class-conditional samples on ImageNet 64×64.

While this differs from the settings in (Song & Dhariwal, 2024), we found it to work better in our re-implementation. The remaining hyperparameters are the same as used in the respective baselines. From the results, we can see how using the learned coupling results in improved performance for both one and two steps generation, outperforming all the re-implemented baselines. The 1-step result from the original iCT is superior to our model. However, to the best of our knowledge, there is no open-source implementation that can reproduce the results reported in the paper. The learned coupling outperforms the minibatch OT coupling in all cases, as it is less affected by the effective (per device) batch size and the data dimensionality. Finally, our 2-step sampling performance for ECM-VE-VC is on par with the current SoTA achieved by other methods with similar settings (Wang et al., 2024; Lee et al., 2024b; Lu & Song, 2024).

FFHQ 64×64: We use FFHQ 64×64 as an additional dataset to assess our method on higher-dimensional data. We reuse the same training settings used for CIFAR10, without additional tuning, and with the same network architecture used in EDM. While the results are worse than current SoTA generative models (e.g. 2.39 FID from EDM), they confirm the benefit of using the learned coupling over the baselines. Moreover, the results highlight the limits of using the minibatch OT coupling, which scales poorly with increased data dimensionality and in some cases performs worse than the independent coupling.

ImageNet 64×64 (class conditional): As a baseline, we reuse the settings from ECM with the EDM2-S architecture and batch size 128. While the baseline is trained for $100k$ iteration, we found that our models with Variational Coupling needed more time to converge properly, as the encoder weights are not pretrained and initialized at random. We therefore train our re-implemented baselines and models for $200k$ iterations instead. In this case, the models with Variational Coupling outperform the other models, with the LI kernel obtaining the best overall FID, while the OT coupling performs poorly due to the small batch size and high data dimensionality. In Figure 2 we compare samples from ECM-LI and ECM-LI-VC, where we can see how the images generated with VC are more clear and detailed.

7 Conclusions

In this work, we introduced a novel approach to Consistency Training (CT) by incorporating a variational noise coupling mechanism. Our method leverages an encoder-based coupling function to learn a data-dependent noise distribution, which results in improved generative performance. By framing CT within the Flow Matching perspective, we provided a principled way to introduce adaptive noise coupling while maintaining the efficiency of standard CT. Empirical results on multiple image benchmarks, demonstrate that our approach consistently outperforms baselines in one and two-step generation settings. Our findings highlight the potential of learned coupling in CT and suggest several promising directions for future work. These include exploring more expressive posterior distributions, extending our method to the continuous-time CT formulation, and integrating variational consistency training with other recent CT improvements. We hope this work contributes to the broader understanding of the effect of coupling in CT and inspires further advancements in efficient generative sampling techniques.

REFERENCES

Michael Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *ICLR 2023 Conference*, 2023.

Michael Samuel Albergo, Mark Goldstein, Nicholas Matthew Boffi, Rajesh Ranganath, and Eric Vanden-Eijnden. Stochastic interpolants with data-dependent couplings. In *Forty-first International Conference on Machine Learning*, 2024.

Jyoti Aneja, Alex Schwing, Jan Kautz, and Arash Vahdat. A contrastive learning approach for training variational autoencoder priors. In *Advances in neural information processing systems*, volume 34, pp. 480–493, 2021.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Rob Cornish, Anthony Caterini, George Deligiannidis, and Arnaud Doucet. Relaxing bijectivity constraints with continuously indexed normalising flows. In *International Conference on Machine Learning*, pp. 2133–2143, 2020.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

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

Hongkun Dou, Junzhe Lu, Jinyang Du, Chengwei Fu, Wen Yao, Xiao qian Chen, and Yue Deng. A unified framework for consistency generative modeling. 2024.

Zhengyang Geng, Ashwini Pokle, William Luo, Justin Lin, and J Zico Kolter. Consistency models made easy. *arXiv preprint arXiv:2406.14548*, 2024.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Jonathan Heek, Emiel Hoogeboom, and Tim Salimans. Multistep consistency models. *arXiv preprint arXiv:2403.06807*, 2024.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022.

Matthew D Hoffman and Matthew J Johnson. Elbo surgery: yet another way to carve up the variational evidence lower bound. In *Workshop in Advances in Approximate Bayesian Inference, NIPS*, volume 1, 2016.

Thibaut Issenhuth, Ludovic Dos Santos, Jean-Yves Franceschi, and Alain Rakotomamonjy. Improving consistency models with generator-induced coupling. *arXiv preprint arXiv:2406.09570*, 2024.

Alexia Jolicoeur-Martineau, Ke Li, Rémi Piché-Taillefer, Tal Kachman, and Ioannis Mitliagkas. Gotta go fast when generating data with score-based models. *arXiv preprint arXiv:2105.14080*, 2021.

Tero Karras. A style-based generator architecture for generative adversarial networks. *arXiv preprint arXiv:1812.04948*, 2019.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and improving the training dynamics of diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24174–24184, 2024.

Beomsu Kim, Yu-Guan Hsieh, Michal Klein, Marco Cuturi, Jong Chul Ye, Bahjat Kawar, and James Thornton. Simple reflow: Improved techniques for fast flow models. *arXiv preprint arXiv:2410.07815*, 2024a.

Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency trajectory models: Learning probability flow ode trajectory of diffusion. In *The Twelfth International Conference on Learning Representations*, 2024b.

Diederik P Kingma. Auto-encoding variational bayes. *International Conference on Learning Representations*, 2013.

Ivan Kobyzev, Simon JD Prince, and Marcus A Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43 (11):3964–3979, 2020.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations*, 2021.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Jeongjun Lee, Jonggeon Park, Jongmin Yoon, and Juho Lee. Stabilizing the training of consistency models with score guidance. In *ICML 2024 Workshop on Structured Probabilistic Inference $\{\backslash\&\}$ Generative Modeling*, 2024a.

Sangyun Lee, Beomsu Kim, and Jong Chul Ye. Minimizing trajectory curvature of ode-based generative models. In *International Conference on Machine Learning*, pp. 18957–18973. PMLR, 2023.

Sangyun Lee, Yilun Xu, Tomas Geffner, Giulia Fanti, Karsten Kreis, Arash Vahdat, and Weili Nie. Truncated consistency models. *arXiv preprint arXiv:2410.14895*, 2024b.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. In *International Conference on Learning Representations*, 2022.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

Cheng Lu and Yang Song. Simplifying, stabilizing and scaling continuous-time consistency models. *arXiv preprint arXiv:2410.11081*, 2024.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. 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.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

Aram-Alexandre Pooladian, Heli Ben-Hamu, Carles Domingo-Enrich, Brandon Amos, Yaron Lipman, and Ricky TQ Chen. Multisample flow matching: straightening flows with minibatch couplings. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 28100–28127, 2023.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pp. 1278–1286. PMLR, 2014.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 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.

Mihaela Rosca, Balaji Lakshminarayanan, and Shakir Mohamed. Distribution matching in variational inference. *arXiv preprint arXiv:1802.06847*, 2018.

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

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. PMLR, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a.

Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. In *The Twelfth International Conference on Learning Representations*, 2024.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021b.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, pp. 32211–32252. PMLR, 2023.

Richard S Sutton. Reinforcement learning: An introduction. *A Bradford Book*, 2018.

Yuhta Takida, Wei-Hsiang Liao, Chieh-Hsin Lai, Toshimitsu Uesaka, Shusuke Takahashi, and Yuki Mitsufuji. Preventing oversmoothing in vae via generalized variance parameterization. *Neurocomputing*, 509:137–156, 2022.

Alexander Tong, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Kilian Fatras, Guy Wolf, and Yoshua Bengio. Conditional flow matching: Simulation-free dynamic optimal transport. *arXiv preprint arXiv:2302.00482*, 2(3), 2023.

Fu-Yun Wang, Zhengyang Geng, and Hongsheng Li. Stable consistency tuning: Understanding and improving consistency models. *arXiv preprint arXiv:2410.18958*, 2024.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

Ling Yang, Zixiang Zhang, Zhilong Zhang, Xingchao Liu, Minkai Xu, Wentao Zhang, Chenlin Meng, Stefano Ermon, and Bin Cui. Consistency flow matching: Defining straight flows with velocity consistency. *arXiv preprint arXiv:2407.02398*, 2024.

A Related Work

Since the introduction of Consistency Models in (Song et al., 2023; Song & Dhariwal, 2024), several strategies have been proposed to improve training stability. The work from (Geng et al., 2024) proposes Easy Consistency Models (ECM) a novel training strategy where time steps are sampled in a continuous fashion and the discretization step is adjusted during training, as opposed to the discrete time grid used in iCT. It further shows the benefits of initializing the network weights with the ones from a pretrained score model, achieving superior performance with smaller training budget. (Wang et al., 2024) builds on top of ECM, introducing additional improvements and framing consistency training as value estimation in Temporal Difference learning (Sutton, 2018). Truncated consistency models, introduced in (Lee et al., 2024b), proposes to add a second training stage on top of ECM, to allow the model to focus its capacity on the later time steps, resulting in improved few-steps generation performance. Other recent contributions to the consistency model literature are works such as (Kim et al., 2024b; Heek et al., 2024) where the focus is on improving multistep sample quality, (Lee et al., 2024a) which trains a model with both consistency and score loss to reduce variance, and (Lu & Song, 2024), which proposes several improvements to the continuous-time training of consistency models. Our work can be seen as a parallel contribution to the aforementioned methods, as we focus on learning the data-noise coupling, which can be used as drop-in replacement to the standard independent coupling.

There are several works showing the benefit of using coupling in Flow Matching (Pooladian et al., 2023; Tong et al., 2023; Liu et al., 2023; Lee et al., 2023; Albergo et al., 2024; Kim et al., 2024a). Among these, (Lee et al., 2023) shares the most similarities with our method, as they also use an encoder to learn a probability distribution over the noise conditioned on the data. Their method results in improved performance compared to equivalent Flow Matching models, while requiring less function evaluations. Our method consists of a similar procedure but applied to CT, resulting in improved few-steps generation performance and confirming the effectiveness of learning the data-noise coupling. A different coupling strategy for CT is proposed in (Issenhuth et al., 2024), where the data-noise coupling is extracted directly from the prediction of the consistency model during training. Compared to our method, they do not need the additional encoder to learn the coupling, but their generator-induced coupling needs to be alternated with the standard independent coupling to avoid instabilities. The Flow Matching formulation in Consistency Models with linear interpolation kernel was previously used in (Dou et al., 2024; Yang et al., 2024), where the former also explores the use of minibatch OT coupling, while the latter trains the model to learn the velocity field and adds a regularization term to enforce constant velocity. In our work, we use the Flow Matching formulation, but keep most of the CT building blocks, resulting in a simpler formulation with superior performance.

B PSEUDOCODE

In this section we report the pseudocode for training and sampling with our learned coupling, in Algorithms 1 and 2 respectively.

Algorithm 1 Variational Consistency Training

```
  Input: data distribution $p_{\text{data}}$, initial model parameter $\boldsymbol{\theta}$, initial encoder parameter $\boldsymbol{\phi}$, learning rate $\eta$, EMA rate $\mu$, distance function $d(\cdot,\cdot)$, consistency weighting $\lambda_{\text{ct}}(\cdot)$, KL weighting $\lambda_{\text{kl}}$
  $\boldsymbol{\theta}_{\text{EMA}} \leftarrow \boldsymbol{\theta}$, $\boldsymbol{\phi}_{\text{EMA}} \leftarrow \boldsymbol{\phi}$ and $k \leftarrow 0$
  repeat
    Sample $\boldsymbol{x}_0 \sim p_{\text{data}}$, $t \sim p(t)$, $r = t - \Delta t$
    Sample $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{I})$
    $\boldsymbol{x}_1 \leftarrow \boldsymbol{g}^{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}_0) + \boldsymbol{g}^{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x}_0)\boldsymbol{\epsilon}$
    $\boldsymbol{x}_t \leftarrow a_t\boldsymbol{x}_0 + b_t\boldsymbol{x}_1$
    $\boldsymbol{x}_r \leftarrow a_r\boldsymbol{x}_0 + b_r\boldsymbol{x}_1$
    $\mathcal{L}^{\text{cond}}_{\text{disc}}(\boldsymbol{\theta}, \boldsymbol{\phi}) \leftarrow \lambda_{\text{ct}}(t)d(\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t), \boldsymbol{f}_{\boldsymbol{\theta}^-}(\boldsymbol{x}_r, r))$
    $\mathcal{L}_{\text{kl}}(\boldsymbol{\phi}) \leftarrow \mathcal{D}_{\text{KL}}(\mathcal{N}(\boldsymbol{g}^{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}_0),\ \boldsymbol{g}^{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x}_0)^2\mathbf{I})||\mathcal{N}(\mathbf{0}, \mathbf{I}))$
    $\tilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}) \leftarrow \mathcal{L}^{\text{cond}}_{\text{disc}}(\boldsymbol{\theta}, \boldsymbol{\phi}) + \lambda_{\text{kl}}\mathcal{L}_{\text{kl}}(\boldsymbol{\phi})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\nabla_{\boldsymbol{\theta}}\tilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi})$
    $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \eta\nabla_{\boldsymbol{\phi}}\tilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi})$
    $\boldsymbol{\theta}_{\text{EMA}} \leftarrow \texttt{stopgrad}(\mu\boldsymbol{\theta}_{\text{EMA}} + (1-\mu)\boldsymbol{\theta})$
    $\boldsymbol{\phi}_{\text{EMA}} \leftarrow \texttt{stopgrad}(\mu\boldsymbol{\phi}_{\text{EMA}} + (1-\mu)\boldsymbol{\phi})$
    $k \leftarrow k + 1$
  until convergence
```

Algorithm 2 Multistep Variational Consistency Sampling

```
  Input: Consistency model $\boldsymbol{f}_{\boldsymbol{\theta}}$, encoder $\boldsymbol{g}_{\boldsymbol{\phi}}$, sequence of time points $\tau_1 > \tau_2 > \cdots > \tau_{N-1}$, initial noise $\hat{\boldsymbol{x}}_T$
  $\boldsymbol{x} \leftarrow \boldsymbol{f}_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_T, T)$
  for $n = 1$ to $N - 1$ do
    Sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    $\boldsymbol{x}_1 \leftarrow \boldsymbol{g}^{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}) + \boldsymbol{g}^{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x})\boldsymbol{\epsilon}$
    $\hat{\boldsymbol{x}}_{\tau_n} \leftarrow a_{\tau_n}\boldsymbol{x} + b_{\tau_n}\boldsymbol{x}_1$
    $\boldsymbol{x} \leftarrow \boldsymbol{f}_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{\tau_n}, \tau_n)$
  end for
  Output: $\boldsymbol{x}$
```

C Consistency Models with Flow Matching Kernel

In addition to the variance exploding forward process commonly used in CT, here we propose to use the linear interpolation kernel commonly used in Flow Matching:

$$\boldsymbol{x}_t = (1-t)\boldsymbol{x}_0 + t\boldsymbol{x}_1. \tag{10}$$

We reuse all of the building blocks from iCT and ECM and make only the necessary adjustments. Accounting for the boundary conditions, the transition kernel becomes:

$$\boldsymbol{x}_t = \left(1 - \frac{t}{\sigma_{\max}}\right)\boldsymbol{x}_0 + \left(\frac{t}{\sigma_{\max}}\right)\boldsymbol{x}_1\sigma_{\max}. \tag{11}$$

Other crucial components for stability during training are the scaling factors c_{in}, c_{skip} and c_{out}, and we derive them for the linear interpolation kernel following the same procedure used in (Karras et al., 2022), also accounting for the boundary conditions when $\sigma_{\min} \neq 0$:

$$c_{\text{in}}(\sigma) = \frac{1}{\sqrt{\sigma^2_{\text{data}}(1 - \frac{\sigma}{\sigma_{\max}})^2 + \sigma^2}} \tag{12}$$

$$c_{\text{skip}}(\sigma) = \frac{\sigma^2_{\text{data}}(1 - \frac{\sigma - \sigma_{\min}}{\sigma_{\max} - \sigma_{\min}})}{(\sigma - \sigma_{\min})^2 + \sigma^2_{\text{data}}(1 - \frac{\sigma - \sigma_{\min}}{\sigma_{\max} - \sigma_{\min}})^2} \tag{13}$$

$$c_{\text{out}}(\sigma) = (\sigma - \sigma_{\min})\sigma_{\text{data}}c_{\text{in}}(\sigma) \tag{14}$$

C.1 Derivations

We report the derivations for the scaling factors used for the linear interpolation transition kernel. We follow the same derivations from (Karras et al., 2022) (Appendix B.6), where the score matching objective is written as:

$$E||D_{\boldsymbol{\theta}}(\boldsymbol{y} + \boldsymbol{n}; \sigma) - \boldsymbol{y}||_2^2 \tag{15}$$

Where $\boldsymbol{y}$ is data sampled from the data distribution with standard deviation σ_{data} and $\boldsymbol{n}$ is a sample from noise distribution with standard deviation σ. Given this objective, they propose to derive the scaling factors $c_{\text{in}}(\sigma)$, $c_{\text{skip}}(\sigma)$, $c_{\text{out}}(\sigma)$ as follows:

$$c_{\text{in}}(\sigma) = \frac{1}{\sqrt{\text{Var}_{\boldsymbol{y},\boldsymbol{n}}[\boldsymbol{y} + \boldsymbol{n}]}} \tag{16}$$

$$c_{\text{out}}(\sigma)^2 = \text{Var}_{\boldsymbol{y},\boldsymbol{n}}[\boldsymbol{y} - c_{\text{skip}}(\sigma)(\boldsymbol{y} + \boldsymbol{n})] \tag{17}$$

$$c_{\text{skip}}(\sigma) = \text{argmin}_{c_{\text{skip}}(\sigma)} c_{\text{out}}(\sigma)^2. \tag{18}$$

In our formulation, we only need to rescale $\boldsymbol{y}$ by $1 - \frac{\sigma}{\sigma_{\max}}$ and perform the same derivations. For simplicity, we define $\alpha = 1 - \frac{\sigma}{\sigma_{\max}}$ (omitting the dependence on σ), and proceed as follows:

$$c_{\text{in}}(\sigma) = \frac{1}{\sqrt{\text{Var}_{\boldsymbol{y},\boldsymbol{n}}[\alpha\boldsymbol{y} + \boldsymbol{n}]}} \tag{19}$$

$$c_{\text{out}}(\sigma)^2 = \text{Var}_{\boldsymbol{y},\boldsymbol{n}}[\boldsymbol{y} - c_{\text{skip}}(\sigma)(\alpha\boldsymbol{y} + \boldsymbol{n})] \tag{20}$$

$$c_{\text{skip}}(\sigma) = \text{argmin}_{c_{\text{skip}}(\sigma)} c_{\text{out}}(\sigma)^2. \tag{21}$$

The factor $c_{\text{in}}(\sigma)$ simply becomes:

$$c_{\text{in}}(\sigma) = \frac{1}{\sqrt{\sigma^2_{\text{data}} * \alpha^2 + \sigma^2}}. \tag{22}$$

To derive $c_{\text{out}}(\sigma)$ we can proceed as:

$$c_{\text{out}}(\sigma)^2 = \text{Var}_{\boldsymbol{y},\boldsymbol{n}}[\boldsymbol{y} - c_{\text{skip}}(\sigma)(\alpha\boldsymbol{y} + \boldsymbol{n})] \tag{23}$$
$$c_{\text{out}}(\sigma)^2 = \text{Var}_{\boldsymbol{y},\boldsymbol{n}}[(1 - \alpha c_{\text{skip}}(\sigma))\boldsymbol{y} + c_{\text{skip}}(\sigma)\boldsymbol{n}] \tag{24}$$
$$c_{\text{out}}(\sigma)^2 = (1 - \alpha c_{\text{skip}}(\sigma))^2\sigma_{\text{data}}^2 + c_{\text{skip}}(\sigma)^2\sigma^2. \tag{25}$$

We can use this result to solve for $c_{\text{skip}}(\sigma)$:

$$0 = d[c_{\text{out}}(\sigma)^2]/dc_{\text{skip}}(\sigma) \tag{26}$$
$$0 = d[(1 - \alpha c_{\text{skip}}(\sigma))^2\sigma_{\text{data}}^2 + c_{\text{skip}}(\sigma)^2\sigma^2]/dc_{\text{skip}}(\sigma) \tag{27}$$
$$0 = \sigma_{\text{data}}^2 d[(1 - \alpha c_{\text{skip}}(\sigma))^2]/dc_{\text{skip}}(\sigma) + \sigma^2 d[c_{\text{skip}}(\sigma)^2]/dc_{\text{skip}}(\sigma) \tag{28}$$
$$0 = \sigma_{\text{data}}^2[2\alpha^2 c_{\text{skip}}(\sigma) - 2\alpha] + \sigma^2[2c_{\text{skip}}(\sigma)] \tag{29}$$
$$0 = (\sigma^2 + \alpha^2\sigma_{\text{data}}^2)c_{\text{skip}}(\sigma) - \alpha\sigma_{\text{data}}^2 \tag{30}$$
$$c_{\text{skip}}(\sigma) = \alpha\sigma_{\text{data}}^2/(\sigma^2 + \alpha^2\sigma_{\text{data}}^2). \tag{31}$$

Finally, we can compute $c_{\text{out}}(\sigma)$:

$$c_{\text{out}}(\sigma)^2 = (1 - \alpha c_{\text{skip}}(\sigma))^2\sigma_{\text{data}}^2 + c_{\text{skip}}(\sigma)^2\sigma^2 \tag{32}$$
$$c_{\text{out}}(\sigma)^2 = \left(1 - \left[\frac{\alpha^2\sigma_{\text{data}}^2}{(\sigma^2 + \alpha^2\sigma_{\text{data}}^2)}\right]\right)^2\sigma_{\text{data}}^2 + \left[\frac{\alpha\sigma_{\text{data}}^2}{(\sigma^2 + \alpha^2\sigma_{\text{data}}^2)}\right]^2\sigma^2 \tag{33}$$
$$c_{\text{out}}(\sigma)^2 = \left[\frac{\sigma^2\sigma_{\text{data}}}{(\sigma^2 + \alpha^2\sigma_{\text{data}}^2)}\right]^2 + \left[\frac{\alpha\sigma_{\text{data}}^2 + \sigma}{(\sigma^2 + \alpha^2\sigma_{\text{data}}^2)}\right]^2 \tag{34}$$
$$c_{\text{out}}(\sigma)^2 = \frac{(\sigma^2\sigma_{\text{data}})^2 + (\sigma\alpha\sigma_{\text{data}}^2)^2}{(\sigma^2 + \alpha^2\sigma_{\text{data}}^2)^2} \tag{35}$$
$$c_{\text{out}}(\sigma)^2 = \frac{(\sigma\sigma_{\text{data}})^2 + (\alpha^2\sigma_{\text{data}}^2 + \sigma^2)}{(\sigma^2 + \alpha^2\sigma_{\text{data}}^2)^2} \tag{36}$$
$$c_{\text{out}}(\sigma)^2 = \frac{(\sigma\sigma_{\text{data}})^2}{(\sigma^2 + \alpha^2\sigma_{\text{data}}^2)} \tag{37}$$
$$c_{\text{out}}(\sigma) = \frac{\sigma\sigma_{\text{data}}}{\sqrt{\sigma^2 + \alpha^2\sigma_{\text{data}}^2}}. \tag{38}$$

If we want to use the boundary conditions for $\sigma_{\min} \neq 0$, then we can modify $c_{\text{skip}}(\sigma)$ and $c_{\text{out}}(\sigma)$ as:

$$c_{\text{skip}}(\sigma) = \frac{\sigma_{\text{data}}^2(1 - \frac{\sigma - \sigma_{\min}}{\sigma_{\max} - \sigma_{\min}})}{(\sigma - \sigma_{\min})^2 + \sigma_{\text{data}}^2(1 - \frac{\sigma - \sigma_{\min}}{\sigma_{\max} - \sigma_{\min}})^2} \tag{39}$$
$$c_{\text{out}}(\sigma) = (\sigma - \sigma_{\min})\sigma_{\text{data}}c_{\text{in}}(\sigma), \tag{40}$$

which satisfy the condition $c_{\text{skip}}(\sigma_{\min}) = 1$ and $c_{\text{out}}(\sigma_{\min}) = 0$.

D Consistency Lower Bound

Proposition 1. *The following upper bound for negative log-density holds:*

$$\begin{aligned}-\log p_\theta(\boldsymbol{x}_0) &\leq \frac{1}{2\sigma^2}\,\mathbb{E}_{q_\phi(\boldsymbol{x}_1|\boldsymbol{x}_0)}\|\boldsymbol{x}_0 - \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_1, 1)\|^2 + \text{KL}\Big(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}_0)\,\Big\|\,p(\boldsymbol{z})\Big)\\ &\leq \frac{1}{2\sigma^2}\int_0^1 \mathbb{E}\left[\left\|\frac{d}{dt}\boldsymbol{f}_{\boldsymbol{\theta}}(\psi_t, t)\right\|^2\right]dt + \text{KL}\Big(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}_0)\,\Big\|\,p(\boldsymbol{z})\Big).\end{aligned}$$

This proposition establishes the connection between the minimization objective of VCT and that of the continuous-time CM.

Proof. For the triangle inequality, we have:

$$\|\boldsymbol{x}_0 - \boldsymbol{f_\theta}(\boldsymbol{x}_1, 1)\| \leq \sum_{i=0}^{N} \left\| \boldsymbol{f_\theta}(\boldsymbol{\psi}_{t_{i+1}}(\boldsymbol{x}_0; \boldsymbol{x}_1), t_{i+1}) - \boldsymbol{f}_{\boldsymbol{\theta}^-}(\boldsymbol{\psi}_{t_i}(\boldsymbol{x}_0; \boldsymbol{x}_1), t_i) \right\|. \tag{41}$$

We can now square both sides, obtaining:

$$\|\boldsymbol{x}_0 - \boldsymbol{f_\theta}(\boldsymbol{x}_1, 1)\|^2 \leq \left(\sum_{i=0}^{N} \left\| \boldsymbol{f_\theta}(\boldsymbol{\psi}_{t_{i+1}}(\boldsymbol{x}_0; \boldsymbol{x}_1), t_{i+1}) - \boldsymbol{f}_{\boldsymbol{\theta}^-}(\boldsymbol{\psi}_{t_i}(\boldsymbol{x}_0; \boldsymbol{x}_1), t_i) \right\| \right)^2. \tag{42}$$

Now, for the Cauchy-Schwarz Inequality, we can write the right hand side as:

$$\left(\sum_{i=0}^{N} \left\| \boldsymbol{f_\theta}(\boldsymbol{\psi}_{t_{i+1}}(\boldsymbol{x}_0; \boldsymbol{x}_1), t_{i+1}) - \boldsymbol{f}_{\boldsymbol{\theta}^-}(\boldsymbol{\psi}_{t_i}(\boldsymbol{x}_0; \boldsymbol{x}_1), t_i) \right\| \right)^2 \leq N \sum_{i=0}^{N} \left\| \boldsymbol{f_\theta}(\boldsymbol{\psi}_{t_{i+1}}(\boldsymbol{x}_0; \boldsymbol{x}_1), t_{i+1}) - \boldsymbol{f}_{\boldsymbol{\theta}^-}(\boldsymbol{\psi}_{t_i}(\boldsymbol{x}_0; \boldsymbol{x}_1), t_i) \right\|^2. \tag{43}$$

Since:

$$\log p_\theta(\boldsymbol{x}_0 \mid \boldsymbol{x}_1) \propto -\frac{1}{2\sigma^2} \|\boldsymbol{x}_0 - \boldsymbol{f_\theta}(\boldsymbol{x}_1, 1)\|^2, \tag{44}$$

we can write a lower bound on the log density as:

$$\log p_\theta(\boldsymbol{x}_0) \geq -\frac{1}{2\sigma^2} \mathbb{E}_{q_\phi(\boldsymbol{x}_1|\boldsymbol{x}_0)} \|\boldsymbol{x}_0 - \boldsymbol{f_\theta}(\boldsymbol{x}_1, 1)\|^2 - \mathrm{KL}\Big(q_\phi(\boldsymbol{x}_1 \mid \boldsymbol{x}_0) \Big\| p(\boldsymbol{x}_1)\Big). \tag{45}$$

In summary, the ELBO bound for the Gaussian VAE is given by

$$-\log p_\theta(\boldsymbol{x}_0) \leq \frac{1}{2\sigma^2} \mathbb{E}_{q_\phi(\boldsymbol{x}_1|\boldsymbol{x}_0)} \|\boldsymbol{x}_0 - \boldsymbol{f_\theta}(\boldsymbol{x}_1, 1)\|^2 + \mathrm{KL}\Big(q_\phi(\boldsymbol{x}_1 \mid \boldsymbol{x}_0) \Big\| p(\boldsymbol{x}_1)\Big) := L(\boldsymbol{x}_0; \theta, \phi). \tag{46}$$

Combining the inequalities we derived in Eq. 42 and 43, below we consider the case when $N \to \infty$. First, we define:

$$S_N = N \sum_{i=0}^{N} \mathbb{E}_{q(x_1|x_0), p(t_i)} \left[\|\boldsymbol{f_\theta}(\psi_{t_{i+1}}(x_0; x_1), t_{i+1}) - \boldsymbol{f_\theta}(\psi_{t_i}(x_0; x_1), t_i)\|^2 \right]. \tag{47}$$

We assume that t_i is a partition of the interval $[0, 1]$ of:

$$\Delta t := t_{i+1} - t_i = \mathcal{O}\left(\frac{1}{N}\right). \tag{48}$$

For small $\Delta t = \mathcal{O}\left(\frac{1}{N}\right)$, we approximate the squared difference in function values using a first-order Taylor expansion[1]:

$$\|\boldsymbol{f_\theta}(\psi_{t_{i+1}}, t_{i+1}) - \boldsymbol{f_\theta}(\psi_{t_i}, t_i)\|^2 \approx \left\| \frac{d}{dt} \boldsymbol{f_\theta}(\psi_t, t) \Big|_{t=t_i} \right\|^2 \cdot \Delta t^2. \tag{49}$$

[1]Here, we assume $\psi_{t_{i+1}} - \psi_{t_i} = \mathcal{O}(t_{i+1} - t_i)$

Since $\Delta t = \mathcal{O}\left(\frac{1}{N}\right)$, we substitute:

$$\|\boldsymbol{f_\theta}(\psi_{t_{i+1}}, t_{i+1}) - \boldsymbol{f_\theta}(\psi_{t_i}, t_i)\|^2 \approx \frac{1}{N^2}\left\|\frac{d}{dt}\boldsymbol{f_\theta}(\psi_t, t)\right\|^2. \tag{50}$$

Multiplying by N:

$$S_N = N\sum_{i=0}^{N}\mathbb{E}\left[\|\boldsymbol{f_\theta}(\psi_{t_{i+1}}, t_{i+1}) - \boldsymbol{f_\theta}(\psi_{t_i}, t_i)\|^2\right] \approx \frac{1}{N}\sum_{i=0}^{N}\mathbb{E}\left[\left\|\frac{d}{dt}\boldsymbol{f_\theta}(\psi_t, t)\right\|^2\right]. \tag{51}$$

As $N \to \infty$,

$$\frac{1}{N}\sum_{i=0}^{N}\mathbb{E}\left[\left\|\frac{d}{dt}\boldsymbol{f_\theta}(\psi_t, t)\right\|^2\right] \to \int_0^1 \mathbb{E}\left[\left\|\frac{d}{dt}\boldsymbol{f_\theta}(\psi_t, t)\right\|^2\right] dt. \tag{52}$$

Thus, multiplying by N, we obtain:

$$\lim_{N\to\infty} S_N = \int_0^1 \mathbb{E}\left[\left\|\frac{d}{dt}\boldsymbol{f_\theta}(\psi_t, t)\right\|^2\right] dt. \tag{53}$$

Combining the above limit with the ELBO bound:

$$\begin{aligned}
-\log p_\theta(\boldsymbol{x}_0) &\leq \frac{1}{2\sigma^2}\mathbb{E}_{q_\phi(\boldsymbol{x}_1|\boldsymbol{x}_0)}\|\boldsymbol{x}_0 - \boldsymbol{f_\theta}(\boldsymbol{x}_1, 1)\|^2 + \mathrm{KL}\Big(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}_0)\,\Big\|\,p(\boldsymbol{z})\Big) && (54)\\
&\leq \frac{1}{2\sigma^2}\frac{1}{N}\sum_{i=0}^{N}\mathbb{E}\left[\left\|\frac{d}{dt}\boldsymbol{f_\theta}(\psi_t, t)\right\|^2\right] + \mathrm{KL}\Big(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}_0)\,\Big\|\,p(\boldsymbol{z})\Big). && (55)\\
&= \frac{1}{2\sigma^2}\int_0^1 \mathbb{E}\left[\left\|\frac{d}{dt}\boldsymbol{f_\theta}(\psi_t, t)\right\|^2\right] dt + \mathrm{KL}\Big(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}_0)\,\Big\|\,p(\boldsymbol{z})\Big), \quad \text{as } N \to \infty. && (56)
\end{aligned}$$

□

E Ablation for different β

To see the effect of β on the generation performance, we compare the results for different values on the FashionMNIST dataset for iCT-VE-VC in Table 3. As expected, with small values of β, the coupling distribution deviates from the sampling distribution and the performance degenerates, while increasing β to high values reduces the benefits of the learned coupling.

FID for different β

	1 step	2 steps
$\beta = 5$	12.53	5.69
$\beta = 15$	4.97	**2.34**
$\beta = 30$	**3.88**	2.37
$\beta = 60$	3.90	2.67

Table 3: Comparison of FID performance (lower is better) for one and two sampling steps, for varying values of β. The models are iCT-VE-VC and trained on the FashionMNIST dataset with the same settings described in appendix F.2. Best entries in bold.

F Experimental details

F.1 Baselines

Here we recap the detials about the two baselines used in this work, the Improved Consistency Training from (Song & Dhariwal, 2024) and Easy Consistency Models from (Geng et al., 2024).

iCT: the training procedure uses a discretization of time steps between two values $\sigma_{\min} = 0.002$ and $\sigma_{\max} = 80$, with the equation from (Karras et al., 2022):

$$\sigma_i = \left(\sigma_{\min}^{1/\rho} + \frac{i-1}{N(k)-1}\left(\sigma_{\max}^{1/\rho} - \sigma_{\min}^{1/\rho}\right)\right)^{\rho}, \text{ where } i \in [[1, N(k)]], \tag{57}$$

where $\rho = 7$ and $N(k)$ is a scheduler that defines the number of discretization steps at the k-th training iteration. $N(k)$ is chosen to be an exponential schedule which starts from $s_0 = 10$ steps and reaches $s_1 = 1280$ steps at the end of the training, and is defined as:

$$N(k) = \min(2^{\lceil k/K' \rceil}, s_1) + 1, \quad K' = \left\lceil \frac{K}{\log_2(s_1/s_0) + 1} \right\rceil. \tag{58}$$

During training, time steps t_i (or equivalently σ_i) are sampled following a discrete lognormal distribution:

$$p(\sigma_i) \propto \text{erf}\left(\frac{\log(\sigma_{i+1}) - P_{\text{mean}}}{\sqrt{2}P_{\text{std}}}\right) - \text{erf}\left(\frac{\log(\sigma_i) - P_{\text{mean}}}{\sqrt{2}P_{\text{std}}}\right), \tag{59}$$

with $P_{\text{mean}} = -1.1$ and $P_{\text{std}} = 2.0$. Then, the steps t_i and t_{i+1} are used in the loss:

$$\mathcal{L}_{\text{ct}}(\boldsymbol{\theta}, \boldsymbol{\phi}) \leftarrow \lambda_{\text{ct}}(t_i) d(\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t_{i+1}}, t+1), \boldsymbol{f}_{\boldsymbol{\theta}^-}(\boldsymbol{x}_{t_i}, t_i)), \tag{60}$$

whith the time dependent weighting function $\lambda_{\text{ct}}(t_i) = \frac{1}{t_{i+1} - t_i}$, and $d(.,.)$ is the Pseudo-Huber loss:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{||\boldsymbol{x} - \boldsymbol{y}||_2^2 + c^2} - c. \tag{61}$$

ECM: ECM aims to simplify and improve the training procedure from iCT. We report here the main differences. Instead of using a discretized grid of time steps, it samples time steps t from a continuous lognormal distribution with $P_{\text{mean}} = -1.1$ and $P_{\text{std}} = 2.0$ (-0.8 and 1.6 for ImageNet). The second time step r used in the discretized training objective is then obtained with a mapping function

$$p(r|t, \text{iters}) = 1 - \frac{1}{q^a} n(t) = 1 - \frac{1}{q^{\lceil \text{iters}/d \rceil}} n(t), \tag{62}$$

where $n(t) = 1 + k\sigma(-bt) = 1 + \frac{k}{1+e^{bt}}$, $\sigma(.)$ is the sigmoid function, *iter* is the current training iteration, $k = 8$, $b = 1$, and $q = 2$ for all the models but ImageNet, where $q = 4$. The discretization step is made smaller for eight times over training (four times for ImageNet). The loss function is a generalization of the Pseudo-Huber loss, which consists of the L2 loss and an adaptive weighting function $w(\Delta)$. The models are initialized with the weights of pretrained diffusion models, which is shown to greatly improve stability during training and generation performance.

F.2 Training details

We report the training details for our models in Tables 4 and 5. Note that the baselines are the ones from our reimplementation. The models have the same number of parameters and training hyperparameters regardless of the transition kernel used. In the following, we report additional information important for reproducing out experiments:

ECM-LI: In ECM, the time steps t ar sampled from a lognormal distribution, as done in (Karras et al., 2022). This means that time steps $t > \sigma_{\max}$ can be sampled during training. While this works well when using the variance exploding Kernel, in the linear interpolantion case the time step t cannot exceed $\sigma_{\max}$, and we therefore clip t to be at most $\sigma_{\max}$.

Random seeds: All the training runs are initialized with random seed 42. For sampling and FID computation, we always set the random seed to 32, which was randomly chosen. This differs from what commonly done in EDM, where three different seeds are used to evaluate FID and the best result is reported. While our evaluation can lead to slightly worse results, the evaluation is consistent between our models and reimplemented baselines.

2-steps generation: Like in the original iCT baseline, all the models use $t = 0.821$ for CIFAR10 and all the other datasets but ImageNet, where $t = 1.526$ is used insetad.

Data augmentation: We scale all the images to have values between -1 and 1. For CIFAR10 we apply random horizontal flip with 50% probability.

Differences for ImageNet: The training procedure for ECM on ImageNet differs slightly from the one for the other datasets. The Adam optimizer is used instead of RAdam, with betas= $(0.9, 0.99)$, and inverse square root learning rate decay defined as a function of the current training iteration i:

$$\alpha(i) = \frac{\alpha_{\text{ref}}}{\sqrt{\max(i/i_{\text{ref}}, 1)}}, \tag{63}$$

with $\alpha_{\text{ref}} = 0.001$ (the initial learning rate) and $i_{\text{ref}} = 2000$ iterations. The Exponential Moving Average uses the power function averaging profile introduced in (Karras et al., 2024). In ECM, three different EMA profiles are tracked during training, with rates 0.01, 0.05, and 0.1. In our reimplementation, we only use the rate 0.1. The number of times in which the discretization interval changes is reduced from 8 to 4, and the loss constant c is set to 0.06.

Model Setups	**FashionMNIST**	**CIFAR10**
Model Architecture	DDPM++	DDPM++
Model Channels	64	128
N° of ResBlocks	4	4
Attention Resolution	-	16
Channel multiplyer	$[2, 2, 2]$	$[2, 2, 2]$
Model capacity	13.6M	55.7M
Training Details		
Minibatch size	128	1024
Batch per device	128	512
Iterations	400k	400k
Dropout probability	30%	30%
Optimizer	RAdam	RAdam
Learning rate	0.0001	0.0001
EMA rate	0.9999	0.9999
Training Cost		
Number of GPUs	1	2
GPU types	H100	H100
Training time (hours)	28	92
Training time with OT (hours)	29	95
Encoder Details		
Model Architecture	DDPM++	DDPM++
Model Channels	32	32
N° of ResBlocks	1	1
Attention Resolution	-	16
Channel multiplyer	$[2, 2, 2]$	$[2, 2, 2]$
β regularizer	30	30
Encoder Params	1.5M	1.6M
Training time with Encoder (hours)	34	102

Table 4: Model Configurations and Training Details for iCT on FashionMNIST and CIFAR10

Model Setups	**CIFAR10**	**FFHQ** 64×46	**ImageNet** 64×64
Model Architecture	DDPM++	DDPM++	EDM2-S
Model Channels	64	128	192
N° of ResBlocks	4	4	3
Attention Resolution	[16]	[16]	[16, 8]
Channel multiplyer	$[2, 2, 2]$	$[1, 2, 2, 2]$	$[1, 2, 3, 4]$
Model capacity	55.7M	61.8M	280M
Training Details			
Minibatch size	128	128	128
Batch per device	128	128	128
Iterations	400k	400k	200k
Dropout probability	20%	20%	40% (res ≤ 16)
Optimizer	RAdam	RAdam	Adam
Learning rate	0.0001	0.0001	0.001
EMA rate	0.9999	0.9999	0.1
Training Cost			
Number of GPUs	1	1	1
GPU types	H100	H100	H100
Training time (hours)	37	95	51
Training time with OT (hours)	38	96	52
Encoder Details			
Model Architecture	DDPM++	DDPM++	EDM2-S
Model Channels	32	32	32
N° of ResBlocks	1	1	2
Attention Resolution	[16]	[16]	[16, 8]
Channel multiplyer	$[2, 2, 2]$	$[1, 2, 2, 2]$	$[1, 2, 3, 4]$
β regularizer	10	10	100 (VE), 90 (LI)
Encoder Params	1.6M	1.6M	6M
Training time with Encoder (hours)	49	110	58

Table 5: Model Configurations and Training Details for ECM on CIFAR10 , FFHQ 64×46 and ImageNet 64×64

G Toy experiments

To gain a visual understanding of the benefits of the variational coupling, we use the model to learn the distribution of a mixture of two Gaussians, with means $\mu_1 = (0, 0.5)$ and $\mu_2 = (0, -0.5)$, and standard deviation $\sigma = 0.05$. We use iCT-LI so that the perturbed data reaches the prior even with small $\sigma_{\max}$, with $\sigma_{\min} = 0.002$, $\sigma_{\max} = 0.1$ and $\sigma_{\text{data}} = 0.05$. The models are trained for 40k iterations, with $s_0 = 10$ and $s_1 = 80$, and EMA rate 0.999. We use a simple four-layers MLP with GeLU activation and Positional time embedding, with batch size 256 and learning rate $1e^{-4}$. For iCT-LI-VC we use $\beta = 0.001$. The results are shown in figures 1 and 3 (one and two step generation respectively).

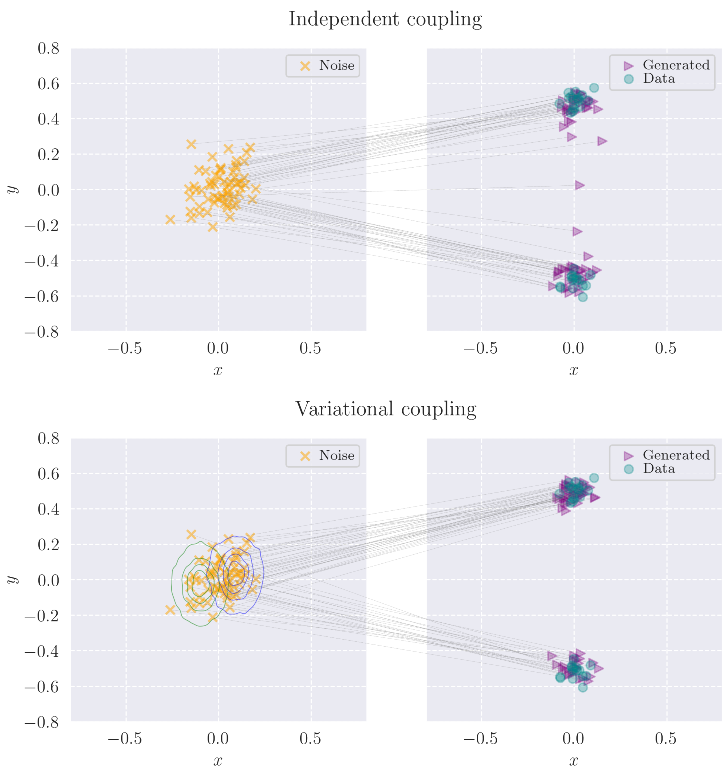


Figure 3: 2-step generation result on the toy data, with $t = 0.07$.

H Model diagram

In figure 4 we show the difference between the forward process for standard Consistency Training and for our method with learned noise coupling, for a given time step t and transition kernel characterized by the coefficients a_t and b_t.

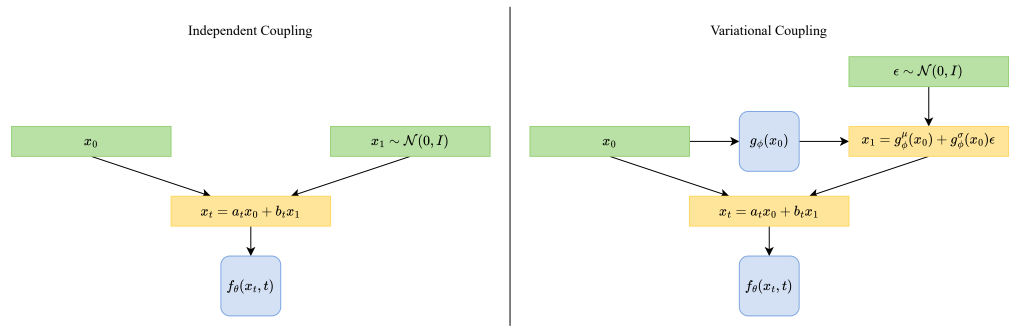

Independent Coupling
x_0
$x_1 \sim \mathcal{N}(0, I)$
$x_t = a_t x_0 + b_t x_1$
$f_\theta(x_t, t)$
Variational Coupling
$\epsilon \sim \mathcal{N}(0, I)$
x_0
$g_\phi(x_0)$
$x_1 = g^\mu_\phi(x_0) + g^\sigma_\phi(x_0)\epsilon$
$x_t = a_t x_0 + b_t x_1$
$f_\theta(x_t, t)$

Figure 4: Diagram for Consistency Training with independent coupling (left) and variational coupling (right).

I Qualitative Results

Here we report samples from our best models, iCT-LI-VC for FashionMNIST (figure 7), iCT-VE-VC and ECM-VE-VC on CIFAR10 (figures 8 and 9), ECM-VE-VC on FFHQ 64×64 (figure 10) and ECM-LI-VC on class conditional Imagenet 64×64 (figure 11). In figure 5, we show the mean and standard deviation learned by the encoder for some images from the CIFAR10 dataset. While the values are very close to a standard Gaussian, the model still retains information from the original input. We empirically compare the variance of the gradients for iCT-VE and iCT-VE-VC trained on CIFAR10, and show in Figure 6 how the resulting reduced variance corresponds to improved FID score. In particular, in early training the model with VC exhibits higher variance, which can be attributed to the fact that the encoder is still learning the coupling. As training continues, the coupling becomes effective at providing better data-noise pairs to the model, which results in reduced gradient variance and improved generative performance.

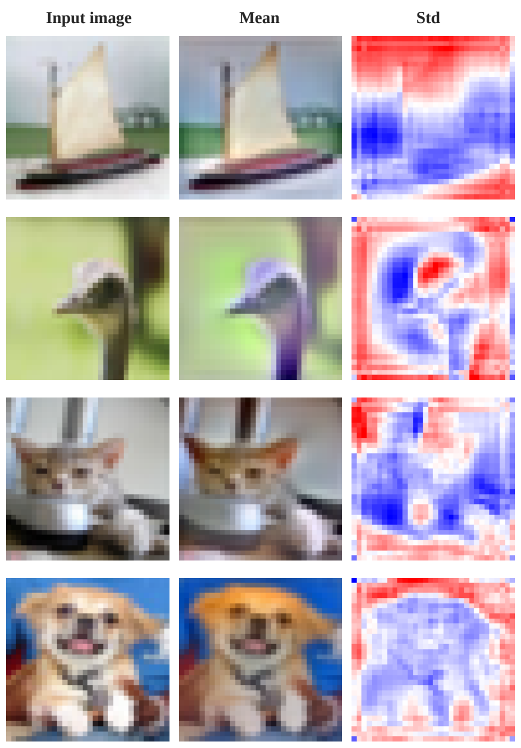
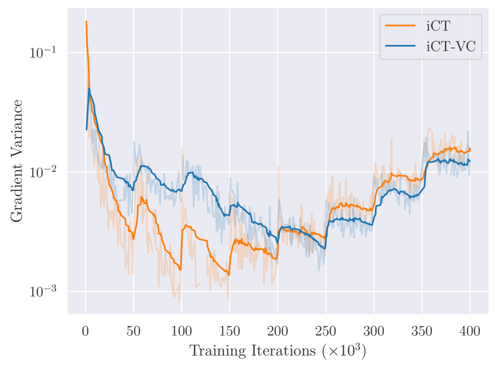
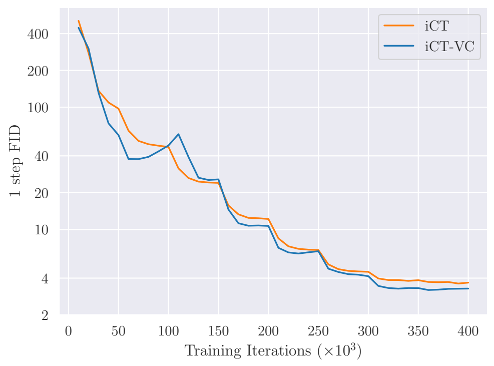


Figure 5: Visualization of the predicted mean and standard deviation for a trained iCT-VE-VC model for different input images. For visualization purpose, we perform min-max rescaling for the predicted mean and standard deviation, as they tend to have most values close to zero and one respectively. We also turn the predicted 3 channels standard deviations to a single channel with grayscale transform.



(a) Gradient Variance



(b) FID Comparison

Figure 6: The left-hand side graph shows a comparison of gradient variance during training for iCT-VE and iCT-VE-VC on CIFAR10. We plot the variance for each epoch (shaded) and its exponential moving average with smoothing factor 0.9. Especially later during training, the model with learned coupling exhibits lower variance, which results in improved performance, shown in terms of 1-step FID in the right-hand side graph. For a fair comparison, we did not use gradient clipping for iCT-VC in this run.

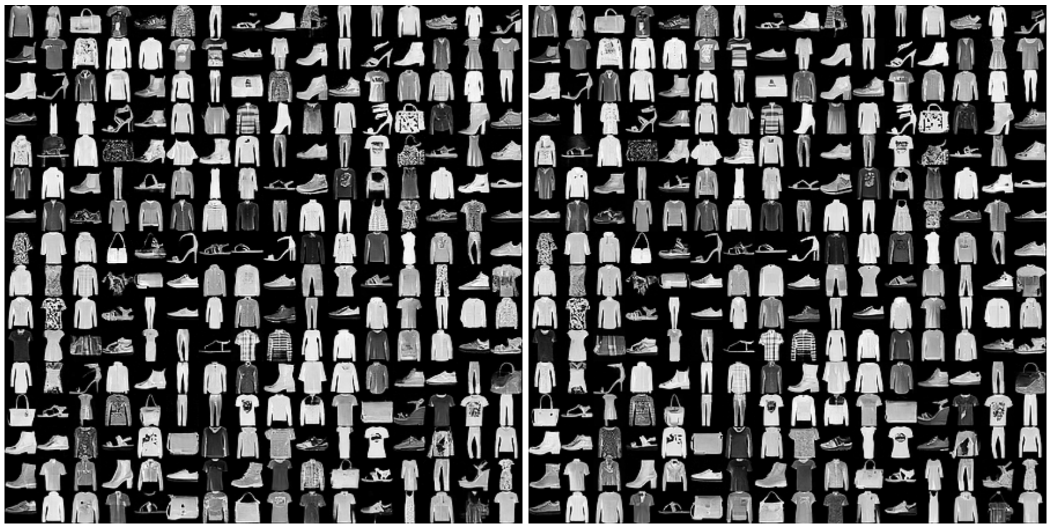


Figure 7: 1-step (FID=3.62, left) and 2-step (FID=2.22, right) generation from iCT-LI-VC trained on FashionMNIST.

Figure 8: 1-step (FID=2.86, left) and 2-step (FID=2.32, right) generation from iCT-VE-VC trained on CIFAR10.

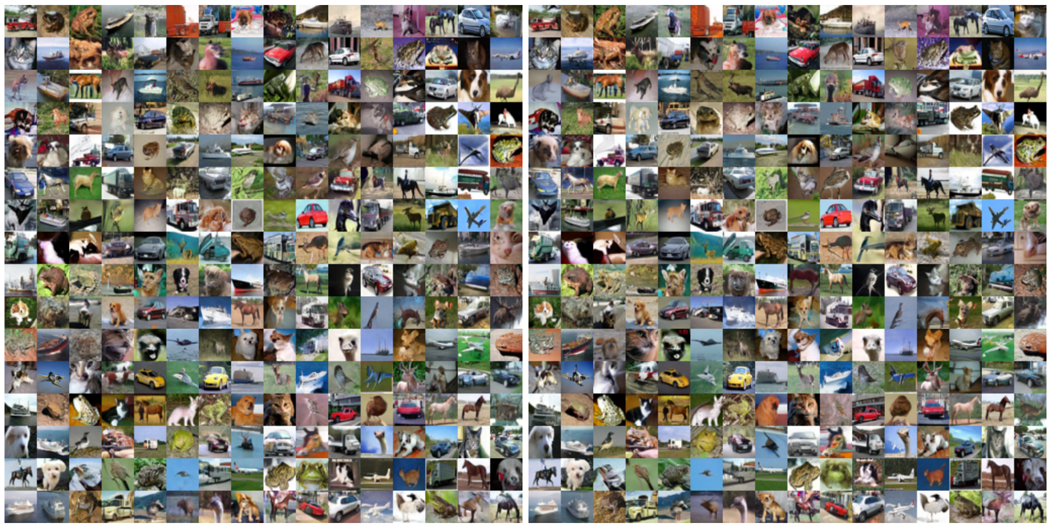


Figure 9: 1-step (FID=3.26, left) and 2-step (FID=2.02, right) generation from ECM-VE-VC trained on CIFAR10.

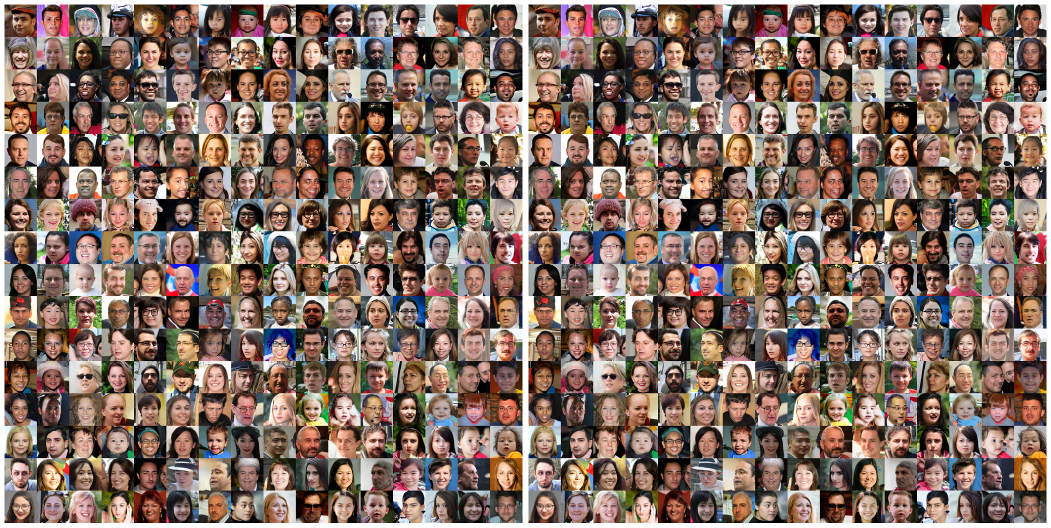


Figure 10: 1-step (FID=5.47, left) and 2-step (FID=4.16, right) generation from ECM-VE-VC trained on FFHQ64×64.

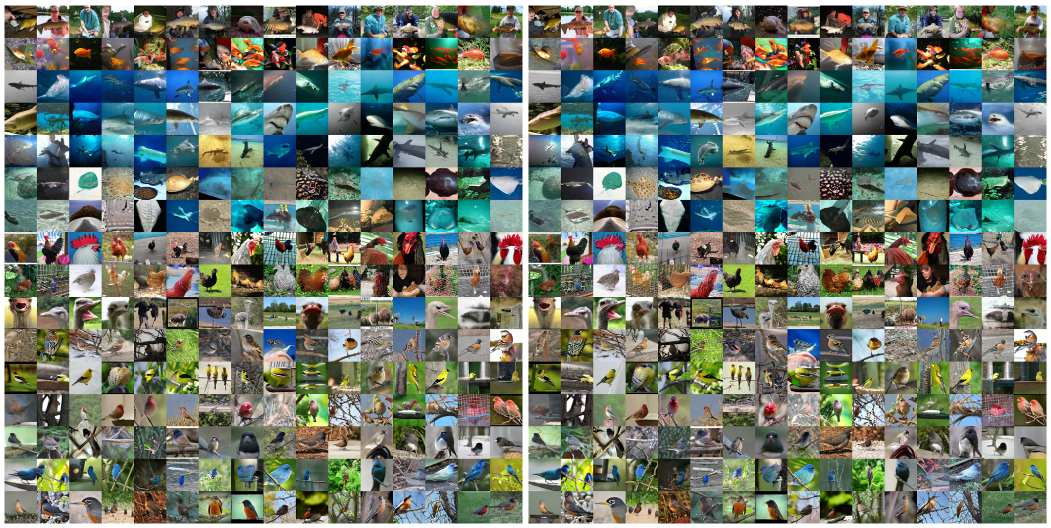


Figure 11: 1-step (FID=4.93, left) and 2-step (FID=3.07, right) generation from ECM-LI-VC trained on class conditional ImageNet 64×64.