IMPLICIT DYNAMICAL FLOW FUSION (IDFF) FOR GENERATIVE MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

Conditional Flow Matching (CFM) models can generate high-quality samples from a non-informative prior, but can require hundreds of network evaluations (NFE), making them computationally expensive. To address this limitation, we introduce Implicit Dynamical Flow Fusion (IDFF), which augments CFM's vector field with learnable momentum terms derived from higher-order derivatives of the log-density. These momentum terms provide geometric information about the probability landscape, enabling more efficient transport from noise to data while preserving the marginal distributions through compensating diffusion. As a result, IDFF reduces the NFEs by a factor of ten compared to CFMs without significantly sacrificing sample quality, allowing for rapid sampling and efficient handling of time-series data generation tasks with compatibility with any ODE solver. We evaluate IDFF on benchmarks for time series and image generation. IDFF demonstrates superior performance on time-series datasets, including molecular dynamics simulation and sea surface temperature (SST) forecasting, highlighting its versatility and effectiveness across diverse domains. For image generation, IDFF achieved an FID score of 2.78 on CIFAR-10, outperforming all existing CFM variants in efficiency while requiring only 10 NFEs compared to 100+ for standard methods.

1 INTRODUCTION

Diffusion models have emerged as powerful generative tools, iteratively transforming noise into structured data with state-of-the-art results in image generation (Song et al., 2020b), text production (Liu et al., 2024; Kim et al., 2019), and other domains (Cachay et al., 2023; Myers et al., 2022). However, a significant practical cost incurred is the number of function evaluations (NFEs) per sample (Ho et al., 2020), an issue which persists inspite of recent advances such as DPM-solvers (Lu et al., 2022b;c) and Denoising Diffusion Implicit Models (Song et al., 2020a).

Figure 1: Image generation using IDFF across datasets. Additional samples and analysis are provided in Appendix F.

While denoising diffusion models learn to generate images by modeling infinitesimal changes that map from noise to data, Conditional Flow Matching (CFM) (Liu et al., 2022; Albergo & Vanden-Eijnden, 2022; Albergo et al., 2023) offers an alternative; by instead learning deterministic vector fields, CFMs learn to transport probability distributions from source to target distributions. To minimize the length of the trajectories between noise and data distributions, OT-CFMs (Tong et al., 2023b) use optimal transport for learning. However, sampling data from CFMs still require ≥ 100 NFEs for high-quality generation (Dao et al., 2023), which is particularly problematic for time-series where sampling costs scale with sequence length.

Higher-order score matching in diffusion models improves sampling efficiency by incorporating second order gradient information about the log density, enabling sampling operations with larger step sizes. For example, Lu et al. (Lu et al., 2022a) demonstrate that second-order corrections significantly reduce discretization errors and accelerate convergence. Our work introduces a novel framework to adapt higher-order information while learning vector fields in CFM. We develop Implicit Dynamical Flow Fusion (IDFF), which successfully incorporates higher-order derivatives of the log-density into CFM's vector field through learnable momentum terms. These momentum terms provide rich geometric information about the probability landscape, enabling more efficient navigation from noise to data distributions while maintaining CFM's theoretical guarantees through carefully designed compensating diffusion. The contributions of our work are as follows:

1. We propose a new flow matching model, Implicit Dynamical Flow Fusion (IDFF), which uses learnable momentum terms from higher-order derivatives of the log-density to accelerate sampling while maintaining theoretical guarantees.
2. We prove that momentum-augmented vector fields preserve CFM's marginal distributions while accelerating convergence. Under mild boundary conditions, IDFF maintains the same marginals as standard CFM but traverses them more efficiently, guaranteeing correct sample generation despite modified dynamics governing the movement of particles.
3. Empirically, IDFF achieves substantial efficiency gains across diverse domains. For image generation (CIFAR-10, CelebA, ImageNet-64, CelebA-HQ, LSUN-Church, LSUN-Bedroom), IDFF produces high-quality samples with ≤ 5 NFEs compared to ≥ 100 NFEs required by current CFM variants. For time-series modeling—3D attractors, molecular dynamics, and sea surface temperature prediction—IDFF maintains superior quality while reducing computational costs by an order of magnitude.

Our contributions demonstrate momentum-augmented flows as a general principle for efficient generative modeling across static and dynamic domains.

2 Background And Preliminaries

Generative modeling aims to learn the underlying structure of complex high-dimensional data distributions from finite samples. Given empirical samples from a dataset $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{M}$ where each sample $\mathbf{x}^{(i)} \in \mathbb{R}^d$ is drawn from an unknown data distribution $p_{\text{data}}(\mathbf{x})$, the goal is to learn a model that can generate new samples that match p_{data}. For time-series data, $\mathcal{D} = \{\mathbf{x}_{1:N}^{(i)}\}_{i=1}^{M}$ where $\mathbf{x}_n^{(i)} \in \mathbb{R}^d$ for $n \in \{1, \ldots, N\}$, and the goal is to model either $p_{\text{data}}(\mathbf{x}_{1:N})$ or $p_{\text{data}}(\mathbf{x}_n|\mathbf{x}_{1:n-1})$.

2.1 Diffusion Models

Score-based generative models take a stochastic approach by learning to reverse a diffusion process that gradually destroys data structure. The forward diffusion follows the stochastic differential equation (SDE):

$$d\mathbf{x}_t = \mathbf{f}(\mathbf{x}_t, t)dt + g(t)d\mathbf{w}_t, \tag{1}$$

where $\mathbf{x}_t \in \mathbb{R}^d$, $\mathbf{f}(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^d$ is the drift, $g(t) \in \mathbb{R}_+$ is the diffusion coefficient, and $\mathbf{w}_t$ is Brownian motion. This transforms data $p_0 = p_{\text{data}}$ to prior $p_T \approx \mathcal{N}(\mathbf{0}, \mathbf{I})$. The reverse-time SDE is:

$$d\mathbf{x}_t = [\mathbf{f}(\mathbf{x}_t, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)]dt + g(t)d\bar{\mathbf{w}}_t, \tag{2}$$

where a neural network is tasked to learn the score function $\nabla_{\mathbf{x}} \log p_t(x_t)$. The probability density under the model evolves via the Fokker-Planck equation $\frac{\partial p_t}{\partial t} = -\nabla \cdot (\mathbf{f} p_t) + \frac{g(t)^2}{2} \Delta p_t$.

(Lu et al., 2022a) extend score matching beyond first-order gradients of the log density to leverage higher-order geometric information about the probability landscape and enable the model to sample from more complex probability distribution. The work leverages momentum variables $\hat{\boldsymbol{\epsilon}}_t^{(k)}(\mathbf{x}) = \nabla_{\mathbf{x}}^k \log p_t(\mathbf{x})$ as the k-th order derivative of the log-density. Under Gaussian transitions $p_{0t}(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \alpha_t \mathbf{x}_0, \sigma_t^2 \mathbf{I})$, the variables have closed forms: $\hat{\boldsymbol{\epsilon}}_{0t}^{(1)} = -\boldsymbol{\epsilon}_0/\sigma_t$, $\hat{\boldsymbol{\epsilon}}_{0t}^{(2)} = -\mathbf{I}/\sigma_t^2$, and $\hat{\boldsymbol{\epsilon}}_{0t}^{(k)} = 0$ for $k \geq 3$, where $\boldsymbol{\epsilon}_0 = (\mathbf{x}_t - \alpha_t \mathbf{x}_0)/\sigma_t$. The work learns separate neural networks $\hat{\boldsymbol{\epsilon}}^{(k)}(\mathbf{x}_t, t; \theta_k)$ to

match these closed forms over the trajectories of the SDE:

$$\mathcal{L}_k = \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}_0} \left\| \hat{\boldsymbol{\epsilon}}_{0t}^{(k)}(\mathbf{x}_t|\mathbf{x}_0) - \hat{\boldsymbol{\epsilon}}^{(k)}(\mathbf{x}_t, t; \theta_k) \right\|^2. \tag{3}$$

The resulting higher-order reverse-time SDE becomes:

$$d\mathbf{x}_t = \left[\mathbf{f}(\mathbf{x}_t, t) - g(t)^2 \hat{\boldsymbol{\epsilon}}_t^{(1)}(\mathbf{x}_t) - \sum_{k=2}^{K} \beta_k(t) \hat{\boldsymbol{\epsilon}}_t^{(k)}(\mathbf{x}_t) \right] dt + g(t) d\bar{\mathbf{w}}_t, \tag{4}$$

where $\beta_k(t)$ are time-dependent coefficients controlling the contribution of higher-order terms.

2.2 Conditional Flow Matching (CFM)

Conditional Flow Matching learns transport maps between probability distributions. Given a time-dependent vector field $\mathbf{v}_t : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$ we can transport samples from a base distribution $p_0 = \mathcal{N}(\mathbf{0}, \mathbf{I})$ to a target distribution $p_1 = p_{\text{data}}$ via the $\frac{d\mathbf{x}_t}{dt} = \mathbf{v}_t(\mathbf{x}_t)$ with $\mathbf{x}_0 \sim p_0$, this induces a probability path $p_t(\mathbf{x})$. The resulting continuity equation:

$$\frac{\partial p_t(\mathbf{x})}{\partial t} + \nabla \cdot (p_t(\mathbf{x}) \mathbf{v}_t(\mathbf{x})) = 0, \tag{5}$$

ensures that probability mass is conserved along the trajectories.

Learning objective and sampling. Instead of modeling the marginal flow directly, CFM's decompose the complex marginal flow into simpler conditional flows. Given pairs $\mathbf{z} = (\mathbf{x}_0, \mathbf{x}_1)$ where $\mathbf{x}_0$ is typically a sample from a normal distribution and $\mathbf{x}_1$ are samples from $\mathcal{D}$, we can define Gaussian conditional paths $p_t(\mathbf{x}_t|\mathbf{x}_0, \mathbf{x}_1) = \mathcal{N}(\mathbf{x}_t; \mu_t, \sigma_t^2 \mathbf{I})$ where $\mu_t = t\mathbf{x}_1 + (1-t)\mathbf{x}_0$ and $\sigma_t = \sigma_0 \sqrt{t(1-t)}$. Then, the conditional vector field becomes:

$$\mathbf{v}_t(\mathbf{x}_t|\mathbf{x}_0, \mathbf{x}_1) = \mathbf{x}_1 - \mathbf{x}_0 + \frac{\dot{\sigma}_t}{\sigma_t}(\mathbf{x}_t - \mu_t), \tag{6}$$

simplifying to $\mathbf{v}_t = \mathbf{x}_1 - \mathbf{x}_0$ when $\sigma_0 = 0$. CFMs train a neural network $\hat{\mathbf{v}}_t(\mathbf{x}; \theta)$ with parameters θ to approximate the conditional field via:

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}[0,1],(\mathbf{x}_0,\mathbf{x}_1)\sim q,\mathbf{x}_t\sim p_t(\cdot|\mathbf{x}_0,\mathbf{x}_1)} \left\| \hat{\mathbf{v}}_t(\mathbf{x}_t; \theta) - \mathbf{v}_t(\mathbf{x}_t|\mathbf{x}_0, \mathbf{x}_1) \right\|^2, \tag{7}$$

$\hat{\mathbf{v}}_t$ takes as input the current position $\mathbf{x}_t$ and time t and outputs a velocity vector in $\mathbb{R}^d$. The coupling $q(\mathbf{x}_0, \mathbf{x}_1)$ is a joint distribution over pairs of noise and data samples, which can be independent $q(\mathbf{x}_0, \mathbf{x}_1) = p_0(\mathbf{x}_0) p_{\text{data}}(\mathbf{x}_1)$ or use optimal transport couplings that minimize the expected transport cost $\mathbb{E}_q[\|\mathbf{x}_1 - \mathbf{x}_0\|^2]$. After training, samples are generated by integrating $\mathbf{x}_{t+\Delta t} = \mathbf{x}_t + \hat{\mathbf{v}}_t(\mathbf{x}_t; \theta)\Delta t$ from $\mathbf{x}_0 \sim p_0$ with step size $\Delta t = 1/\text{NFE}$.

The validity of the conditional vector field (CFM) rests on the following fundamental result:

Theorem 2.1 (CFM Marginal Consistency (Lipman et al., 2022))**.** *If conditional paths $p_t(\mathbf{x}|\mathbf{z})$ with fields $\mathbf{v}_t(\mathbf{x}|\mathbf{z})$ satisfy the continuity equation, then the marginal $p_t(\mathbf{x}) = \int p_t(\mathbf{x}|\mathbf{z}) q(\mathbf{z}) d\mathbf{z}$ with $\mathbf{v}_t(\mathbf{x}) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\mathbf{v}_t(\mathbf{x}|\mathbf{z})]$ also satisfies the continuity equation. C*

By Theorem 2.1, learning to match $\mathbf{v}_t(\mathbf{x}_t|\mathbf{x}_0, \mathbf{x}_1)$ is sufficient to recover the marginal flow and results in being able to sample from p_{data}.

Challenges in incorporating higher-order terms with CFM. Since the incorporation of momentum variables to explicitly capture higher order derivatives of the log density has shown promise in diffusion models, we anticipate that their incorporation in CFM could similarly yield value by combining the efficiency in learning deterministic transport maps with the acceleration in sampling probability manifolds conferred via higher-order information.

However the direct addition of momentum terms to the vector field for CFMs runs into several technical hurdles.

Consider a naive augmentation of the vector field as $\tilde{\mathbf{v}}_t = \mathbf{v}_t + \sum_{k=1}^{K} \gamma_t^k \hat{\boldsymbol{\epsilon}}^{(k)}$ where $\hat{\boldsymbol{\epsilon}}^{(k)} = \nabla_{\mathbf{x}}^k \log p_t$. Substituting into the continuity equation yields:

$$\frac{\partial p_t}{\partial t} + \nabla \cdot (p_t \tilde{\mathbf{v}}_t) = \frac{\partial p_t}{\partial t} + \nabla \cdot (p_t \mathbf{v}_t) + \sum_{k=1}^{K} \gamma_t^k \nabla \cdot (p_t \nabla_{\mathbf{x}}^k \log p_t). \tag{8}$$

Since the original CFM satisfies $\frac{\partial p_t}{\partial t} + \nabla \cdot (p_t \mathbf{v}_t) = 0$, the augmented field satisfies the continuity equation only if $\sum_{k=1}^{K} \gamma_t^k \nabla \cdot (p_t \nabla_{\mathbf{x}}^k \log p_t) = 0$. For the first-order term, $\nabla \cdot (p_t \nabla_{\mathbf{x}} \log p_t) = \nabla \cdot \nabla_{\mathbf{x}} p_t = \Delta p_t \neq 0$ in general. The addition terms result in a violation of Theorem 2.1's conditions and consequently means the augmented flow no longer transports p_0 to $p_1 = p_{\text{data}}$ exactly. Additionally, CFM's conditional paths only provide velocity supervision for training neural networks, not explicit targets for learning $\hat{\boldsymbol{\epsilon}}^{(k)}$ and computing derivatives during sampling requires k autodiff passes per step. These challenges necessitate the development of a principled framework that carefully balances acceleration with CFM's theoretical guarantees.

3 Implicit Dynamical Flow Fusion Model (IDFF)

IDFF augments the CFM vector field with a learnable momentum that accelerates convergence without breaking marginal consistency. We begin by defining a new augmented flow as:

$$\tilde{\mathbf{v}}_t(\mathbf{x}_t) = \gamma_t^0 \mathbf{v}_t(\mathbf{x}_t) + \gamma_t^1 \nabla_x \log p_t(\mathbf{x}_t), \tag{9}$$

where γ_t^0, γ_t^1 are time-dependent interpolation coefficients with $\gamma_t^0 + \gamma_t^1 = 1$ to interpolate between base flow and momentum variables. The momentum term $\boldsymbol{\xi}_t = \gamma_t^1 \nabla_x \log p_t(\mathbf{x}_t)$ guides particles along the log-density gradient, accelerating movement toward high-probability regions.

We can generalize the momentum to incorporate higher-order terms by defining: $\boldsymbol{\xi}_t(\mathbf{x}_t) = \sum_{k=1}^{K} \gamma_t^k \nabla_x^k \log p_t(\mathbf{x}_t)$. The second-order terms $\nabla_x^2 \log p_t$ capture local curvature information to enable sampling using step sizes that adapt to the local geometry, while higher orders enable efficient navigation through complex multi-modal landscapes. Both the base vector field $\mathbf{v}_t(\mathbf{x}_t)$ and momentum terms $\boldsymbol{\xi}_t(\mathbf{x}_t) = \sum_{k=1}^{K} \gamma_t^k \nabla_{\mathbf{x}}^k \log p_t(\mathbf{x}_t)$ can be modeled and parameterized by neural networks and to approximate the augmented vector field $\tilde{\mathbf{v}}_t(\mathbf{x}_t)$ in equation 10.

3.1 Central Theorem: K-th Order Marginal Preservation and Continuity

We now present the central theoretical result that establishes the validity of the K-th order IDFF approach.

Theorem 3.1 (K-th Order IDFF Marginal Preservation and Continuity)**.** *Let $\mathbf{v}_t(\mathbf{x}_t)$ be a vector field that generates marginal density $p_t(\mathbf{x}_t)$ via the continuity equation. Define the K-th order IDFF vector field:*

$$\tilde{\mathbf{v}}_t(\mathbf{x}_t) = \gamma_t^0 \mathbf{v}_t(\mathbf{x}_t) + \sum_{k=1}^{K} \gamma_t^k \nabla_{\mathbf{x}}^k \log p_t(\mathbf{x}_t). \tag{10}$$

Under the stochastic differential equation framework with diffusion coefficient σ_t, define the effective vector field:

$$\mathbf{w}_t(\mathbf{x}_t) = \gamma_t^0 \mathbf{v}_t(\mathbf{x}_t) + \left(\gamma_t^1 - \frac{\sigma_t^2}{2}\right) \nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t) + \sum_{k=2}^{K} \gamma_t^k \nabla_{\mathbf{x}}^k \log p_t(\mathbf{x}_t). \tag{11}$$

If the boundary conditions $\sigma_t \to 0$ and $\{\gamma_t^k\}_{k=1}^K \to 0$ as $t \to \{0, 1\}$ are satisfied, and the normalization condition $\sum_{k=0}^{K} \gamma_t^k = 1$ holds for all t, then:

1. ***Marginal Preservation**: The transformed vector field $\tilde{\mathbf{v}}_t(\mathbf{x}_t)$ preserves the same marginal distribution $p_t(\mathbf{x}_t)$ as the original CFM.*
2. ***Continuity**: The probability density evolves according to:*

$$\frac{\partial p_t(\mathbf{x}_t)}{\partial t} = -\nabla \cdot (\mathbf{w}_t(\mathbf{x}_t) p_t(\mathbf{x}_t)) \tag{12}$$

where $\mathbf{w}_t$ is the effective drift after accounting for the SDE formulation with diffusion σ_t.

3. ***Sampling Validity**: Samples generated by the SDE $d\mathbf{x}_t = \tilde{\mathbf{v}}_t(\mathbf{x}_t)dt + \sigma_t d\mathbf{w}$ follow the target distribution.*

Remark 3.2 (Effective Drift vs. Augmented Field). The distinction between $\tilde{\mathbf{v}}_t$ and $\mathbf{w}_t$ arises from the Fokker-Planck equation. When our SDE includes a diffusion term $\sigma_t d\mathbf{w}$, the probability density evolves according to:

$$\frac{\partial p_t}{\partial t} = -\nabla \cdot (\tilde{\mathbf{v}}_t p_t) + \frac{\sigma_t^2}{2}\Delta p_t = -\nabla \cdot \left(\tilde{\mathbf{v}}_t - \frac{\sigma_t^2}{2}\nabla \log p_t\right) p_t$$

Thus, the effective drift in the continuity equation is $\mathbf{w}_t = \tilde{\mathbf{v}}_t - \frac{\sigma_t^2}{2}\nabla \log p_t$, which compensates for the diffusion-induced spreading.

The boundary conditions ensure that $\tilde{\mathbf{v}}_t(\mathbf{x}_t)$ converges to $\mathbf{v}_t(\mathbf{x}_t)$ at $t = 0, 1$, preserving consistency at the endpoints. Thus, $\tilde{\mathbf{v}}_t(\mathbf{x}_t)$ can be used in place of $\mathbf{v}_t(\mathbf{x}_t)$ in the generative process to define a valid IDFF sampling path. A complete proof is provided in Appendix A.

Corollary 3.3 (Reduction to Standard CFM)**.** *When $K = 0$ and $\gamma_t^0 = 1$, Theorem 3.1 reduces to standard CFM where $\tilde{\mathbf{v}}_t = \mathbf{v}_t$.*

Corollary 3.4 (Connection to Score-Based Models)**.** *When $\gamma_t^0 = 0$ and $K = 1$ with $\gamma_t^1 = \alpha$ constant, IDFF reduces to score-based diffusion with $\tilde{\mathbf{v}}_t = \alpha \nabla_x \log p_t(\mathbf{x}_t)$. In the SDE formulation with diffusion σ_t, the effective drift becomes $\mathbf{w}_t = (\alpha - \frac{\sigma_t^2}{2})\nabla_x \log p_t(\mathbf{x}_t)$, recovering the score-based diffusion framework as shown in Albergo et al. (2023).*

3.2 Sample-Based Vector Field Learning

A key insight in IDFF is the reparameterization of vector field learning through direct sample prediction. Rather than directly learning $\mathbf{v}_t(\mathbf{x}_t)$, we predict the target sample $\mathbf{x}_1$ and use this to construct the velocity field. This approach offers computational advantages when combined with momentum terms and provides a unified framework for both denoising and flow learning.

Defining probability paths. Following OT-CFM, we define probability paths between the base distribution ($\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$) and data samples ($\mathbf{x}_1 \sim p_{\text{data}}$) via:

$$p_t(\mathbf{x}_t|\mathbf{x}_1) = \mathcal{N}\left(\mathbf{x}_t \middle| \mu_t; \sigma_t^2 \mathbf{I}\right), \quad \mu_t = t\mathbf{x}_1 + (1-t)\mathbf{x}_0, \quad \sigma_t = \sigma_0\sqrt{t(1-t)}. \tag{13}$$

This path induces the conditional velocity field:

$$\mathbf{v}_t(\mathbf{x}_t|\mathbf{x}_1) = \frac{\mathbf{x}_1 - \mathbf{x}_t}{1-t}. \tag{14}$$

Vector field reparameterization. Rather than directly learning the velocity field $\mathbf{v}_t$, we reparameterize it through sample prediction. Near $t = 1$, the velocity $\mathbf{v}_t = (\mathbf{x}_1 - \mathbf{x}_t)/(1-t)$ exhibits a singularity as $(1-t) \to 0$. To avoid this numerical instability, we instead learn $\hat{\mathbf{x}}_1(\mathbf{x}_t, t; \theta)$ to predict the clean target sample, then construct $\hat{\mathbf{v}}_t = (\hat{\mathbf{x}}_1 - \mathbf{x}_t)/(1-t)$. This reparameterization offers three key advantages: (i) numerical stability by avoiding the singularity, (ii) direct optimization for final generation quality rather than intermediate velocities, and (iii) a unified framework with denoising diffusion models. For the momentum terms $\nabla^k \log p_t$, we can either use dedicated network heads for each order (enabling constant-time inference) or compute them via automatic differentiation from the first-order score (requiring k backward passes per step but fewer parameters).

Classic CFM training minimizes the discrepancy $\|\hat{\mathbf{v}}_t - \mathbf{v}_t\|^2$. IDFF reparameterizes the velocity field as:

$$\hat{\mathbf{v}}_t(\mathbf{x}_t|\mathbf{x}_1; \theta) = \frac{\hat{\mathbf{x}}_1(\mathbf{x}_t, t; \theta) - \mathbf{x}_t}{1-t}, \tag{15}$$

where $\hat{\mathbf{x}}_1(\mathbf{x}_t, t; \theta)$ is a neural network that predicts the clean target sample $\mathbf{x}_1$ given the noisy observation $\mathbf{x}_t$ at time t. This approach is related to recent work on trajectory rectification (Zhang et al., 2024), but differs in the incorporation of momentum terms and higher-order corrections. Since the conditional path $p_t(\mathbf{x}_t|\mathbf{x}_1)$ connects $\mathbf{x}_1$ to $\mathbf{x}_t$ via a simple Gaussian interpolant, and the conditional velocity field has the closed form in Equation 14, this reparameterization allows us to train the entire model using tractable sample-based objectives.

Training objective. Because both $\hat{\mathbf{x}}_1$ and the transition density $p_t(\mathbf{x}_t|\mathbf{x}_1)$ are tractable, we formulate the entire loss using sample-based prediction (see Appendix B for a comprehensive derivation). Drawing $\mathbf{x}_1 \sim p_{\text{data}}$, $t \sim \mathcal{U}(0,1)$, and $\mathbf{x}_t \sim p_t(\mathbf{x}_t|\mathbf{x}_1)$, we define:

$$\mathcal{L}_{\text{IDFF}}(\theta, K) = \mathbb{E}_{t,\mathbf{x}_1,\mathbf{x}_t}\Big[\underbrace{\beta(t)^2\|\hat{\mathbf{x}}_1(\mathbf{x}_t,t;\theta)-\mathbf{x}_1\|^2}_{\mathcal{L}^{(0)}_{\text{IDFF}}(\theta)} + \underbrace{\sum_{k=1}^{K}\lambda_k(t)^2\left\|\hat{\boldsymbol{\epsilon}}^k(\mathbf{x}_t,t;\theta)-\nabla^k\log p_t(\mathbf{x}_t|\mathbf{x}_1)\right\|^2}_{\mathcal{L}^{(K)}_{\text{IDFF}}(\theta)}\Big]. \tag{16}$$

A single neural network outputs $K+1$ predictions: $\hat{\mathbf{x}}_1$ and $\{\hat{\boldsymbol{\epsilon}}^k\}_{k=1}^K$ where $\hat{\boldsymbol{\epsilon}}^k(\mathbf{x}_t,t;\theta) \approx \nabla^k_{\mathbf{x}_t}\log p_t(\mathbf{x}_t|\mathbf{x}_1)$. The positive schedules $\beta(t)$ and $\{\lambda_k(t)\}_{k=1}^K$ weight different time steps and derivative orders:

- $\mathcal{L}^{(0)}_{\text{IDFF}}$ trains the denoiser $\hat{\mathbf{x}}_1$ to recover clean samples
- $\mathcal{L}^{(K)}_{\text{IDFF}}$ trains the score and higher-order derivatives that drive momentum-based acceleration

Algorithm 1 IDFF **Training** (with/out OT)

Input: data distribution $p_1(\mathbf{x}_1)$, initial dist. $p_0(\mathbf{x}_0)$, bandwidth σ_0, weight schedules $\{\lambda_k(\cdot)\}$, $\beta(\cdot)$, solver order K; init. networks $\hat{\mathbf{x}}_1(\cdot;\theta)$ and $\{\hat{\boldsymbol{\epsilon}}^k(\cdot;\theta)\}_{k=1}^K$
while training **do**
  Sample $\mathbf{x}_1 \sim p_1, \mathbf{x}_0 \sim p_0$; $t \sim \mathcal{U}(0,1)$
  $\pi \leftarrow \text{OT}(\mathbf{x}_1,\mathbf{x}_0)$ (minibatch approx.)
  $(\mathbf{x}_0,\mathbf{x}_1) \sim \pi$
  $\boldsymbol{\mu}_t \leftarrow t\,\mathbf{x}_1 + (1-t)\,\mathbf{x}_0$
  $\sigma_t \leftarrow \sigma_0\sqrt{t(1-t)}$
  $\mathbf{x}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \sigma_t^2\mathbf{I})$
  $L \leftarrow \mathcal{L}^0_{\text{IDFF}}(\theta) + \mathcal{L}^{(\text{K})}_{\text{IDFF}}(\theta)$
  $\theta \leftarrow \text{Update}(\theta, \nabla_\theta L)$
end while
return $\{\hat{\mathbf{x}}_1(\cdot;\theta),\ \hat{\boldsymbol{\epsilon}}^k(\cdot;\theta)\}_{k=1}^K$

Through ablation studies, we demonstrate that IDFF can effectively replace OT-based paths with independently coupled paths, achieving comparable performance without relying on OT computations (Section 5.1). This substitution notably speeds up training and eliminates the computational burden of OT, especially with large batch sizes.

There are two approaches for handling higher-order derivatives:

1. **Explicit network heads**: Predict each $\nabla^k_x \log p_t(\mathbf{x}_t)$ from dedicated output heads
2. **Automatic differentiation**: Predict only first-order score, then use automatic differentiation for higher orders

While the second approach saves model parameters, it introduces inference-time overhead: computing k-th order derivatives via automatic differentiation requires approximately k backward passes per time step, increasing both the effective NFE and memory consumption significantly. In contrast, using explicit heads allows constant-time inference, making it more efficient in low-latency or resource-constrained settings (see Appendix A.4 for more details). The choice depends on available memory, compute budget, and deployment requirements. Algorithm 1 summarizes the overall training procedure.

3.3 Sampling with IDFF

After training $\hat{\mathbf{x}}_1(\cdot;\theta)$ and $\{\hat{\boldsymbol{\epsilon}}^k(\cdot;\theta)\}_{k=1}^K$, we generate samples by integrating the continuity equation from Theorem 3.1. The probability density evolves according to:

$$\frac{\partial p_t(\mathbf{x}_t)}{\partial t} = -\nabla\cdot(\mathbf{w}_t(\mathbf{x}_t)p_t(\mathbf{x}_t)), \tag{17}$$

where the effective drift field $\mathbf{w}_t$ is:

$$\mathbf{w}_t(\mathbf{x}_t) = \gamma_t^0\frac{\hat{\mathbf{x}}_1(\mathbf{x}_t,t;\theta)-\mathbf{x}_t}{1-t} + \left(\gamma_t^1 - \frac{\sigma_t^2}{2}\right)\hat{\boldsymbol{\epsilon}}^1(\mathbf{x}_t,t;\theta) + \sum_{k=2}^{K}\gamma_t^k\hat{\boldsymbol{\epsilon}}^k(\mathbf{x}_t,t;\theta).$$

The $\mathbf{w}_t$ incorporates both the base flow and momentum corrections while accounting for the diffusion present in the underlying SDE formulation. This ensures the generated samples follow the correct probability path from source to target distribution. For numerical integration, we discretize the flow with time step $\Delta t = 1/\text{NFE}$ and evolve samples through:

$$\mathbf{x}_{t+\Delta t} = \mathbf{x}_t + \mathbf{w}_t(\mathbf{x}_t)\Delta t + \sigma_t\sqrt{\Delta t}\cdot\boldsymbol{\eta}, \quad \boldsymbol{\eta} \sim \mathcal{N}(0, \mathbf{I}), \tag{18}$$

where $\sigma_t = \sigma_0\sqrt{t(1-t)}$ provides controlled stochasticity that vanishes at the boundaries. Starting from $\mathbf{x}_0 \sim \mathcal{N}(0, \mathbf{I})$, we integrate forward in time until $t = 1$ to obtain samples from the target distribution. The momentum terms in $\mathbf{w}_t$ accelerate convergence, allowing for accurate sampling with significantly fewer NFEs compared to standard CFM, while Theorem 3.1 guarantees preservation of the marginal distribution $p_t(\mathbf{x}_t)$ throughout the trajectory.

Problem setup for time-series. For time-series data, we have sequences $\{\mathbf{x}_{1:N}^{(i)}\}_{i=1}^{M}$ where each sequence consists of N observations $\mathbf{x}_n^{(i)} \in \mathbb{R}^d$ for $n \in \{1, \ldots, N\}$. The goal is to model the joint distribution $p_{\text{data}}(\mathbf{x}_{1:N})$ or conditional distributions $p_{\text{data}}(\mathbf{x}_n|\mathbf{x}_{1:n-1})$. IDFF handles this by introducing a dual-time formulation: a discrete index $n \in \{1, \ldots, N\}$ for sequence position and continuous $t \in [0, 1]$ for flow interpolation. The networks become $\hat{\mathbf{x}}_1(\mathbf{x}_t, t, n; \theta)$ and $\hat{\boldsymbol{\epsilon}}^k(\mathbf{x}_t, t, n; \theta)$, conditioning on both temporal indices. For $n > 1$, we initialize $\mathbf{x}_0^n \sim \mathcal{N}(\mathbf{x}_1^{n-1}, \sigma_0^2\mathbf{I})$ rather than pure noise, creating a Markovian structure that maintains temporal coherence. When $N = 1$ (static data), this reduces to the standard formulation. The training and sampling procedures adapt accordingly, as detailed in Appendix D and Algorithms 2–3.

4 Related Work

CFM has revived interest in continuous-time generative modeling by removing the need for expensive simulation-based training procedures. The original *OT-CFM* method (Lipman et al., 2022) enforces an exact OT-path between noise and data, yielding strong likelihood performance but suffering from the cubic complexity of solving an OT problem at each minibatch. Follow-up works such as improved OT-CFM (Tong et al., 2023a) and $[\text{SF}]^2\text{M}$ mitigate this bottleneck by introducing minibatch-based OT approximations and introducing stochastic sampling using Schrödinger bridges. However, these models still require minibatch-approximation of OT, time-reversed ODEs/SDEs to construct sampling dynamics, and require a large number of NFEs to reach high-fidelity sample qualities. IDFF eliminates the need for minibatch-approximation of OT and time-reversed ODEs/SDEs through its direct formulation, departing fundamentally from these approaches by introducing a momentum-augmented vector field that embeds learnable momentum vectors $\boldsymbol{\xi}_t$ directly into the CFM vector field. This momentum-driven approach incorporates the acceleration dynamics of gradient flows while preserving marginal path consistency with the original CFM.

Trajectory Flow Matching (TFM) (Zhang et al., 2024) explores refinements to CFMs by injecting stochasticity and reparameterizing the vector field using a sample-space predictor similar to IDFF. However, TFM uses the same vector field as traditional CFMs and consequently still requires extensive function evaluations to generate high-fidelity samples. Unlike TFM's limited approach, IDFF combines flow-based generation with momentum-inspired acceleration for faster sampling without quality loss, as demonstrated in Section 5.2.

In parallel, the diffusion model community has pursued the NFE reduction challenge via tailored samplers. DDIM (Song et al., 2020a) transforms stochastic diffusion trajectories into deterministic flows, while distillation frameworks such as Flash (Kohler et al., 2024) compress many-step sampling into efficient student models. High-order ODE solvers like *DPM-Solver* (Lu et al., 2022b), DPM-Solver++ (Lu et al., 2022b), and DPM-Solver V3 (Zheng et al., 2023) leverage closed-form dynamics to produce high-quality generations in only 10–20 steps.

Higher-order score matching approaches. Prior work on higher-order derivatives in generative modeling focuses on improving score estimation accuracy. Lu et al. (Lu et al., 2022a) and Meng et al. (Meng et al., 2021) use higher-order score functions with separate objectives for each derivative order, requiring 100-1000 NFE within standard SDE solvers.

IDFF fundamentally differs by: (1) integrating momentum terms directly into velocity field dynamics for acceleration rather than accuracy enhancement; (2) using a unified objective that jointly opti-

mizes the denoiser and all momentum terms; (3) reimagining ODE integration through momentum-augmented vector fields, enabling 5-10 NFE versus traditional methods.

Unlike prior work that improves score approximation within existing solvers, IDFF constructs an entirely new flow inherently requiring fewer integration steps. By reparameterizing in the sample space, IDFF achieves better distribution alignment with significantly fewer function evaluations. Furthermore, IDFF extends to arbitrary K-th order derivatives while guaranteeing marginal preservation (Theorem 3.1), enabling adaptive momentum-based acceleration for complex data manifolds.

5 Experiments

We evaluate IDFF across two different generation paradigms: static image generation and time-series data generation, including simulated chaotic systems (Appendix G), molecular dynamics, and sea surface temperature forecasting.

5.1 Image Generation

We conducted comprehensive experiments on the CIFAR-10 dataset, with supplementary results spanning CelebA, ImageNet-64, CelebA-HQ, LSUN Bedrooms, and LSUN Church detailed in Appendix F and E. The image generation results on CIFAR-10 are presented in Table 1. The horizontal line separates diffusion and flow-based models. When juxtaposed with state-of-the-art models in CFMs including $[SF]^2M$, OT-CFM, IDFF consistently demonstrates superior results in both computational efficiency and generated image quality. IDFF attains superior FID scores while requiring only one-tenth the number of function evaluations compared to OT-CFM. With an FID of 2.78, IDFF sets a new state-of-the-art among all CFMs. Although it does not aim to outperform leading diffusion models such as DPM-S-V3-EDM Zheng et al. (2023) and iCTs Song & Dhariwal (2023) in general-purpose generative modeling, IDFF markedly enhances CFM performance and substantially closes the gap. This is further supported by the Feature Likelihood Divergence (FLD) results in Table 10 and wall-clock time comparisons in Table 9, which underscore IDFF's effectiveness in bridging this performance divide.

Table 1: Comparison of FID and NFE between IDFF and various methods on the CIFAR-10 dataset. We utilize the ScoreSDE Song et al. (2020b) backbone for the experiment. Additional results are provided in Appendix F.

Model	FID↓	NFE↓
DDIM	13.36	10
DDIM	6.84	20
DPM-S	6.03	12
DEIS	4.17	10
DPM-S++	4.01	10
UniPC	3.93	10
DPM-S-V3	3.40	10
DDPM	3.17	1000
iCT	2.83	1
DPM-S-V3-EDM	2.51	10
iCT	2.46	2
FM	14.36	10
OT-CFM	11.87	10
$[SF]^2$M	10.13	10
OT-CFM	6.35	142
IDFF (Ours)	**2.78**	**10**

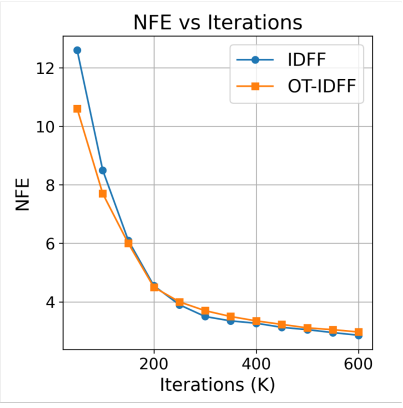

Figure 2: Evaluation of NFE across iterations for CIFAR-10 dataset.

The qualitative results, visualized in Figure 1 with additional samples in Appendix F, further substantiate our quantitative findings. The NFE directly corresponds to the number of neural network forward passes required during sampling. Reducing NFE from 100 to 10-5 translates to a $10\times - 20\times$ speedup in generation time. For CIFAR-10, this means generating 50,000 samples in 30 minutes on a single A6000 GPU. To evaluate the necessity of optimal transport in IDFF, we conducted experiments comparing the convergence speed and performance of the trained model with OT-based paths versus independently coupled paths. Figure 2 illustrates that IDFF maintains comparable performance without relying on computationally expensive OT calculations. This is possible because momentum terms naturally guide particles toward high-probability regions, reducing reliance on carefully initialized paths.

Ablations over NFE and time scheduling. Table 2 examines IDFF's performance across varying numbers of NFEs against state-of-the-art diffusion models. While performance improves with increased NFE, the most dramatic gains occur between NFE=6 and NFE=8, with diminishing returns beyond NFE=8. This exceptional efficiency stems from IDFF's momentum-driven dynamics, which enable more efficient traversal of the probability path compared to standard CFM approaches in fewer NFEs. Table 7 investigates the impact of different time step sampling strategies during training on IDFF's performance. Time sampling based on cosine functions produced optimal results (FID=2.78), probably because it places greater emphasis on the training loss near the $t = 1$ boundary, where sustaining the quality of generated samples is vital.

Table 2: FID↓ of the methods with different numbers of NFEs, evaluated on 50k samples. †We borrow the results reported in their original paper directly.

Method	NFE			
	5	6	8	10
†DEIS	15.37	–	–	4.17
DPM-S++	28.53	13.48	5.34	4.01
UniPC	23.71	10.41	5.16	3.93
DPM-S-v3	12.76	7.40	3.94	3.40
IDFF	**8.53**	**5.67**	**3.25**	**2.78**

5.2 Learning Efficient Surrogate Models for MD Simulation

Molecular dynamics simulations are essential for drug discovery and materials science but remain computationally prohibitive—simulating even microseconds of protein dynamics can require weeks of computation (Marx & Hutter, 2009; Car & Parrinello, 1985). Learning surrogate models that can generate realistic trajectories offers a path to accelerate scientific discovery (Shaw et al., 2008). We evaluate IDFF's ability to learn complex molecular dynamics from limited simulation data.

Table 3: MAE, RMSE, and CC results for the MD simulation.

Method	MAE↓	RMSE ↓	CC (%)↑
SRNN	$82.6_{\pm 28}$	$91.9_{\pm 25}$	$10.2_{\pm 0.27}$
DVAE	$78.1_{\pm 27}$	$88.1_{\pm 25}$	$30.4_{\pm 0.35}$
NODE	$25.3_{\pm 6.3}$	$28.8_{\pm 6.2}$	$10.5_{\pm 0.41}$
OT-CFM	$13.3_{\pm 1.1}$	$16.3_{\pm 2.4}$	$86.1_{\pm 0.1}$
TFM	$11.4_{\pm 1.1}$	$14.2_{\pm 2.7}$	$89.4_{\pm 0.4}$
IDFF-1st	$\mathbf{9.2}_{\pm 0.9}$	$\mathbf{12.5}_{\pm 2.8}$	$\mathbf{95.6}_{\pm 0.1}$
IDFF-2nd	$\mathbf{4.9}_{\pm 1.1}$	$\mathbf{9.7}_{\pm 2.8}$	$\mathbf{97.8}_{\pm 0.1}$

We employ traditional physics-based molecular dynamics (MD) simulations using the AMBER force field to generate training data. The system consists of a fully extended polyalanine structure containing 253 atoms and 46 dihedral angles, simulated for 400 picoseconds at 300K in vacuum conditions with dihedral angles recorded at regular intervals. IDFF learns to model the generative distribution of these MD trajectories, functioning as an efficient surrogate model for the underlying dynamical system rather than performing computationally expensive ab initio calculations.

IDFF demonstrates remarkable precision in predicting dynamics for complex molecular structures. Figure 3 illustrates the model's performance, presenting distributions of actual (A) and generated (B) dihedral angles. Panel (C) shows the dihedral angles for a single alanine molecule, while panel (D) provides a trajectory comparison between actual and generated angles over time. We rigorously assess performance using three key metrics: root mean squared error (RMSE), mean absolute error (MAE), and correlation coefficients (CC) between generated and actual trajectories.

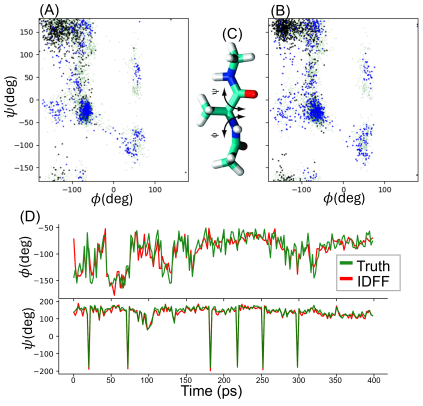

Figure 3: (A) True and (B) generated dihedral angles. (C) The angles for an alanine molecule. (D) True and generated angle trajectories by IDFF-1st order.

When benchmarked against established dynamical models including Sequential Recurrent Neural Networks (SRNN), Variational Recurrent Neural Networks (VRNN), and Neural Ordinary Differential Equations (NODE), the proposed methods—OT-CFM, TFM, and IDFF—demonstrate superior performance across all metrics (Table 3). Notably, IDFF-2nd achieves the best results, highlighting the method's strong potential for accurate molecular dynamics simulations.

5.3 SEA SURFACE TEMPERATURE FORECASTING

Forecasting sea surface temperature (SST) is vital for weather forecasting and climate modeling (Haghbin et al., 2021). We apply IDFF to forecast ea surface temperature (SST) using a daily dataset from 1982-2021, with data split into training (1982-2019, 15,048 samples), validation (2020, 396 samples), and testing (2021, 396 samples). Following (Cachay et al., 2023), we transform the global data into 60×60 (latitude$\times$longitude) tiles, selecting 11 patches in the eastern tropical Pacific Ocean for forecasting horizons of 1-7 days.

We use two complementary metrics: Mean Squared Error (MSE) measures deterministic forecast accuracy, while Continuous Ranked Probability Score (CRPS) evaluates probabilistic forecast quality by comparing the predicted distribution to observations (we did the evaluations for forecasts extending up to 7 days). CRPS is particularly important for weather forecasting as it captures uncertainty quantification. CRPS calculations involve generating a 20-member ensemble to capture forecast uncertainty, while MSE is computed on the ensemble mean predictions to assess deterministic accuracy. We comprehensively compare IDFF against multiple state-of-the-art baseline methods, including DDPM (Ho et al., 2020), MCVD (Voleti et al., 2022), DDPM variants (DDPM-D (Gal & Ghahramani, 2016) and DDPM-P (Pathak et al., 2022)), Dyffusion (Cachay et al., 2023), Alternator Rezaei & Dieng (2024), and OT-CFM (Tong et al., 2023b).

Table 4: Results for sea surface temperature forecasting 1 to 7 days ahead, averaged over the evaluation horizon.

Method	CRPS↓	MSE↓
Perturb.	0.281 ± 0.004	0.180 ± 0.011
Dropout	0.267 ± 0.003	0.164 ± 0.004
DDPM	0.246 ± 0.005	0.177 ± 0.005
MCVD	0.216	0.161
Dyffusion	0.224 ± 0.001	0.173 ± 0.001
Alternator	0.221 ± 0.031	0.144 ± 0.045
OT-CFM	0.231 ± 0.005	0.175 ± 0.006
IDFF (Ours)	$\mathbf{0.180 \pm 0.024}$	$\mathbf{0.105 \pm 0.029}$

While competing methods require up to 1000 neural function evaluations (NFEs) during inference, IDFF achieves superior performance with merely 5 NFEs, representing a significant computational advantage. The model substantially outperforms all baselines across both CRPS and MSE metrics. Representative forecasting results are visualized in Figure 12, demonstrating the model's ability to capture complex spatiotemporal patterns in ocean temperature dynamics. In IDFF, each forecasting step builds on the preceding prediction instead of starting from noise, which minimizes transport distance. The incorporated momentum terms represent common evolution patterns, facilitating quick passage through likely future scenarios. IDFF's dual-time formulation (t for flow, n for sequence position) enables parameter sharing across time steps while maintaining temporal consistency. The networks $\hat{\mathbf{x}}_1(\mathbf{x}_t, t, n; \theta)$ and $\hat{\boldsymbol{\epsilon}}^k(\mathbf{x}_t, t, n; \theta)$ learn both the flow dynamics and temporal evolution jointly.

6 CONCLUSION AND DISCUSSION

IDFF models present a substantial advancement in generative modeling, particularly excelling in both image and time-series data generation. By learning a new vector field, IDFF achieves significantly improved computational efficiency by a factor of ten compared to traditional CFMs without compromising sample quality. This efficiency, combined with compatibility with various ODE solvers, makes IDFF a versatile approach for generative modeling tasks. Looking ahead, the efficiency and flexibility of IDFF open promising avenues for further exploration in diverse domains such as music generation (Briot et al., 2017), speech synthesis (Yu & Deng, 2016), and biological time-series modeling (Anumanchipalli et al., 2019; Golshan et al., 2020; Rezaei et al., 2021; 2023; Gracco et al., 2005). Developing new architectures and improved training techniques will further improve the application and performance of IDFF in these areas.

Limitations and Future Work: While IDFF dramatically reduces NFEs, it requires training additional network heads for momentum terms, increasing model size by 10%. For extremely high-dimensional data ($d \geq 10000$), storing higher-order terms becomes prohibitive, limiting us to $K \leq 2$. Additionally, the optimal choice of γ_t^k schedules remains dataset-dependent, requiring hyperparameter tuning.

REFERENCES

Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022.

Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

Gopala K Anumanchipalli, Josh Chartier, and Edward F Chang. Speech synthesis from neural decoding of spoken sentences. *Nature*, 568(7753):493–498, 2019.

Jean-Pierre Briot, Gaëtan Hadjeres, and François-David Pachet. Deep learning techniques for music generation–a survey. *arXiv preprint arXiv:1709.01620*, 2017.

Salva Rühling Cachay, Bo Zhao, Hailey James, and Rose Yu. Dyffusion: A dynamics-informed diffusion model for spatiotemporal forecasting. *arXiv preprint arXiv:2306.01984*, 2023.

Richard Car and Mark Parrinello. Unified approach for molecular dynamics and density-functional theory. *Physical review letters*, 55(22):2471, 1985.

Quan Dao, Hao Phung, Binh Nguyen, and Anh Tran. Flow matching in latent space. *arXiv preprint arXiv:2307.08698*, 2023.

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

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059. PMLR, 2016.

Hosein M Golshan, Adam O Hebb, and Mohammad H Mahoor. Lfp-net: A deep learning framework to recognize human behavioral activities using brain stn-lfp signals. *Journal of neuroscience methods*, 335:108621, 2020.

Vincent L Gracco, Pascale Tremblay, and Bruce Pike. Imaging speech production using fmri. *Neuroimage*, 26(1):294–301, 2005.

Masoud Haghbin, Ahmad Sharafati, Davide Motta, Nadhir Al-Ansari, and Mohamadreza Hosseinian Moghadam Noghani. Applications of soft computing models for predicting sea surface temperature: a comprehensive review and assessment. *Progress in earth and planetary science*, 8: 1–19, 2021.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Marco Jiralerspong, Joey Bose, Ian Gemp, Chongli Qin, Yoram Bachrach, and Gauthier Gidel. Feature likelihood divergence: evaluating the generalization of generative models using samples. *Advances in Neural Information Processing Systems*, 36:33095–33119, 2023.

Dong Wook Kim, Hye Young Jang, Kyung Won Kim, Youngbin Shin, and Seong Ho Park. Design characteristics of studies reporting the performance of artificial intelligence algorithms for diagnostic analysis of medical images: results from recently published papers. *Korean journal of radiology*, 20(3):405–410, 2019.

Jonas Kohler, Albert Pumarola, Edgar Schönfeld, Artsiom Sanakoyeu, Roshan Sumbaly, Peter Vajda, and Ali Thabet. Imagine flash: Accelerating emu diffusion models with backward distillation. *arXiv preprint arXiv:2405.05224*, 2024.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Yixin Liu, Kai Zhang, Yuan Li, Zhiling Yan, Chujie Gao, Ruoxi Chen, Zhengqing Yuan, Yue Huang, Hanchi Sun, Jianfeng Gao, et al. Sora: A review on background, technology, limitations, and opportunities of large vision models. *arXiv preprint arXiv:2402.17177*, 2024.

Cheng Lu, Kaiwen Zheng, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Maximum likelihood training for score-based diffusion odes by high order denoising score matching. In *International conference on machine learning*, pp. 14429–14460. PMLR, 2022a.

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, 2022b.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022c.

Dominik Marx and Jürg Hutter. *Ab initio molecular dynamics: basic theory and advanced methods*. Cambridge University Press, 2009.

Chenlin Meng, Yang Song, Wenzhe Li, and Stefano Ermon. Estimating high order gradients of the data distribution by denoising. *Advances in Neural Information Processing Systems*, 34: 25359–25369, 2021.

Catherine E Myers, Alejandro Interian, and Ahmed A Moustafa. A practical introduction to using the drift diffusion model of decision-making in cognitive psychology, neuroscience, and health sciences. *Frontiers in Psychology*, 13:1039172, 2022.

Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, et al. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

W Peebles and S Xie. Scalable diffusion models with transformers. arxiv e-prints, art. *arXiv preprint arXiv:2212.09748*, 2022.

Mohammad R Rezaei, Haseul Jeoung, Ayda Gharamani, Utpal Saha, Venkat Bhat, Milos R Popovic, Ali Yousefi, Robert Chen, and Milad Lankarany. Inferring cognitive state underlying conflict choices in verbal stroop task using heterogeneous input discriminative-generative decoder model. *Journal of Neural Engineering*, 20(5):056016, 2023.

Mohammad Reza Rezaei and Adji Bousso Dieng. Alternators for sequence modeling. *arXiv preprint arXiv:2405.11848*, 2024.

Mohammad Reza Rezaei, Kensuke Arai, Loren M Frank, Uri T Eden, and Ali Yousefi. Real-time point process filter for multidimensional decoding problems using mixture models. *Journal of neuroscience methods*, 348:109006, 2021.

David E Shaw, Martin M Deneroff, Ron O Dror, Jeffrey S Kuskin, Richard H Larson, John K Salmon, Cliff Young, Brannon Batson, Kevin J Bowers, Jack C Chao, et al. Anton, a special-purpose machine for molecular dynamics simulation. *Communications of the ACM*, 51(7):91–97, 2008.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020a.

Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. *arXiv preprint arXiv:2310.14189*, 2023.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020b.

Alexander Tong, Nikolay Malkin, Kilian Fatras, Lazar Atanackovic, Yanlei Zhang, Guillaume Huguet, Guy Wolf, and Yoshua Bengio. Simulation-free schr\" odinger bridges via score and flow matching. *arXiv preprint arXiv:2307.03672*, 2023a.

Alexander Tong, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Kilian Fatras, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *arXiv preprint arXiv:2302.00482*, 2023b.

Vikram Voleti, Alexia Jolicoeur-Martineau, and Chris Pal. Mcvd-masked conditional video diffusion for prediction, generation, and interpolation. *Advances in Neural Information Processing Systems*, 35:23371–23385, 2022.

Dong Yu and Li Deng. *Automatic speech recognition*, volume 1. Springer, 2016.

Xi Nicole Zhang, Yuan Pu, Yuki Kawamura, Andrew Loza, Yoshua Bengio, Dennis Shung, and Alexander Tong. Trajectory flow matching with applications to clinical time series modelling. *Advances in Neural Information Processing Systems*, 37:107198–107224, 2024.

Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. Unipc: A unified predictor-corrector framework for fast sampling of diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024.

Kaiwen Zheng, Cheng Lu, Jianfei Chen, and Jun Zhu. Dpm-solver-v3: Improved diffusion ode solver with empirical model statistics. *Advances in Neural Information Processing Systems*, 36: 55502–55542, 2023.

CONTENTS

A K-TH ORDER IDFF MARGINAL PRESERVATION AND CONTINUITY

In this section, we provide a complete proof of Theorem 3.1, which establishes that the K-th order IDFF vector field preserves marginal distributions and satisfies the continuity equation.

A.1 PROOF OF THEOREM 3.1

Statement: Let $\mathbf{v}_t(\mathbf{x}_t)$ be a vector field that generates marginal density $p_t(\mathbf{x}_t)$ via the continuity equation. Define the K-th order IDFF vector field:

$$\tilde{\mathbf{v}}_t(\mathbf{x}_t) = \gamma_t^0 \mathbf{v}_t(\mathbf{x}_t) + \sum_{k=1}^{K} \gamma_t^k \nabla_x^k \log p_t(\mathbf{x}_t). \tag{19}$$

If the boundary conditions $\sigma_t \to 0$ and $\{\gamma_t^k\}_{k=1}^K \to 0$ as $t \to 0, 1$ are satisfied, and the normalization condition $\sum_{k=0}^K \gamma_t^k = 1$ holds for all t, then the three stated properties hold.

Part 1: Marginal Preservation **Goal**: Show that $\tilde{\mathbf{v}}_t(\mathbf{x}_t)$ generates the same marginal distribution $p_t(\mathbf{x}_t)$ as $\mathbf{v}_t(\mathbf{x}_t)$.

Proof: We proceed by extending the conditional flow matching framework to higher-order derivatives.

STEP 1: CONSISTENCY CONDITION Following the conditional flow matching paradigm, the original vector field satisfies:

$$\mathbf{v}_t(\mathbf{x}) = \mathbb{E}_{q(\mathbf{z})}\left[\frac{\mathbf{v}_t(\mathbf{x}|\mathbf{z})p_t(\mathbf{x}|\mathbf{z})}{p_t(\mathbf{x})}\right], \tag{A}$$

where $\mathbf{z}$ is an auxiliary random variable distributed according to $q(\mathbf{z})$ (typically uniform).

For $\tilde{\mathbf{v}}_t$ to preserve the marginal, it must satisfy:

$$\tilde{\mathbf{v}}_t(\mathbf{x}) = \mathbb{E}_{q(\mathbf{z})}\left[\frac{\tilde{\mathbf{v}}_t(\mathbf{x}|\mathbf{z})p_t(\mathbf{x}|\mathbf{z})}{p_t(\mathbf{x})}\right]. \tag{B}$$

STEP 2: DEFINE THE CONDITIONAL MOMENTUM TERMS Define the conditional K-th order IDFF vector field:

$$\tilde{\mathbf{v}}_t(\mathbf{x}|\mathbf{z}) = \gamma_t^0 \mathbf{v}_t(\mathbf{x}|\mathbf{z}) + \sum_{k=1}^{K} \gamma_t^k \nabla_x^k \log p_t(\mathbf{x}|\mathbf{z}). \tag{C}$$

Define the momentum terms:

$$\boldsymbol{\xi}_t(\mathbf{x}|\mathbf{z}) = \sum_{k=1}^{K} \gamma_t^k \nabla_x^k \log p_t(\mathbf{x}|\mathbf{z}), \quad \boldsymbol{\xi}_t = \sum_{k=1}^{K} \gamma_t^k \nabla_x^k \log p_t(\mathbf{x}). \tag{D}$$

STEP 3: VERIFY THE CONSISTENCY CONDITION FOR MOMENTUM TERMS From equations (A) and (B), subtracting γ_t^0 times equation (A) from equation (B), we require:

$$\boldsymbol{\xi}_t = \mathbb{E}_{q(\mathbf{z})}\left[\frac{\boldsymbol{\xi}_t(\mathbf{x}|\mathbf{z})p_t(\mathbf{x}|\mathbf{z})}{p_t(\mathbf{x})}\right]. \tag{E}$$

Substituting the definition of $\boldsymbol{\xi}_t(\mathbf{x}|\mathbf{z})$:

$$\sum_{k=1}^{K} \gamma_t^k \nabla_x^k \log p_t(\mathbf{x}) = \mathbb{E}_{q(\mathbf{z})}\left[\frac{\sum_{k=1}^K \gamma_t^k \nabla_x^k \log p_t(\mathbf{x}|\mathbf{z}) p_t(\mathbf{x}|\mathbf{z})}{p_t(\mathbf{x})}\right]. \tag{F}$$

STEP 4: VERIFY EACH ORDER SEPARATELY For each order k, we need to show:

$$\nabla_x^k \log p_t(\mathbf{x}) = \mathbb{E}_{q(\mathbf{z})}\left[\frac{\nabla_x^k \log p_t(\mathbf{x}|\mathbf{z}) p_t(\mathbf{x}|\mathbf{z})}{p_t(\mathbf{x})}\right]. \tag{G}$$

Using the property that for exponential family distributions (including Gaussians):

$$\nabla_x^k p_t(\mathbf{x}|\mathbf{z}) = p_t(\mathbf{x}|\mathbf{z}) \nabla_x^k \log p_t(\mathbf{x}|\mathbf{z}) + \text{lower-order terms}. \tag{H}$$

For Gaussian distributions used in CFM, the lower-order terms have a specific structure that allows clean marginalization. We have:

$$\mathbb{E}_{q(\mathbf{z})}\left[\frac{\nabla_x^k p_t(\mathbf{x}|\mathbf{z})}{p_t(\mathbf{x})}\right] = \frac{1}{p_t(\mathbf{x})} \int \nabla_x^k p_t(\mathbf{x}|\mathbf{z}) q(\mathbf{z}) d\mathbf{z}. \tag{I}$$

Since $p_t(\mathbf{x}) = \int p_t(\mathbf{x}|\mathbf{z}) q(\mathbf{z}) d\mathbf{z}$, taking the k-th derivative:

$$\nabla_x^k p_t(\mathbf{x}) = \int \nabla_x^k p_t(\mathbf{x}|\mathbf{z}) q(\mathbf{z}) d\mathbf{z}. \tag{J}$$

Therefore:

$$\mathbb{E}_{q(\mathbf{z})}\left[\frac{\nabla_x^k p_t(\mathbf{x}|\mathbf{z})}{p_t(\mathbf{x})}\right] = \frac{\nabla_x^k p_t(\mathbf{x})}{p_t(\mathbf{x})} = \nabla_x^k \log p_t(\mathbf{x}). \tag{K}$$

For exponential family distributions, substituting equation (H) into the expectation:

$$\mathbb{E}_{q(\mathbf{z})}\left[\frac{p_t(\mathbf{x}|\mathbf{z}) \nabla_x^k \log p_t(\mathbf{x}|\mathbf{z})}{p_t(\mathbf{x})}\right] = \mathbb{E}_{q(\mathbf{z})}\left[\frac{\nabla_x^k p_t(\mathbf{x}|\mathbf{z})}{p_t(\mathbf{x})}\right] \tag{20}$$

$$= \nabla_x^k \log p_t(\mathbf{x}). \tag{L}$$

This verifies equation (G) for each order k, thus confirming that $\tilde{\mathbf{v}}_t$ preserves the marginal distribution.

Part 2: Continuity Equation **Goal**: Show that $p_t(\mathbf{x}_t)$ evolves according to the continuity equation under the SDE framework.

Proof: Consider the stochastic differential equation:

$$d\mathbf{x}_t = \tilde{\mathbf{v}}_t(\mathbf{x}_t) dt + \sigma_t d\mathbf{w}. \tag{M}$$

The associated Fokker-Planck equation is:

$$\frac{\partial p_t(\mathbf{x}_t)}{\partial t} = -\nabla \cdot (\tilde{\mathbf{v}}_t(\mathbf{x}_t) p_t(\mathbf{x}_t)) + \frac{\sigma_t^2}{2} \Delta p_t(\mathbf{x}_t). \tag{N}$$

Using the identity $\Delta p_t = \nabla \cdot (\nabla p_t) = \nabla \cdot (p_t \nabla \log p_t)$:

$$\frac{\partial p_t(\mathbf{x}_t)}{\partial t} = -\nabla \cdot \left(\tilde{\mathbf{v}}_t(\mathbf{x}_t) p_t(\mathbf{x}_t) - \frac{\sigma_t^2}{2} p_t(\mathbf{x}_t) \nabla \log p_t(\mathbf{x}_t)\right). \tag{O}$$

This can be rewritten as:

$$\frac{\partial p_t(\mathbf{x}_t)}{\partial t} = -\nabla \cdot (\mathbf{w}_t(\mathbf{x}_t) p_t(\mathbf{x}_t)), \tag{P}$$

where the effective drift is:

$$\mathbf{w}_t(\mathbf{x}_t) = \tilde{\mathbf{v}}_t(\mathbf{x}_t) - \frac{\sigma_t^2}{2} \nabla \log p_t(\mathbf{x}_t). \tag{Q}$$

Substituting the definition of $\tilde{\mathbf{v}}_t$:

$$\mathbf{w}_t(\mathbf{x}_t) = \gamma_t^0 \mathbf{v}_t(\mathbf{x}_t) + \sum_{k=1}^{K} \gamma_t^k \nabla_x^k \log p_t(\mathbf{x}_t) - \frac{\sigma_t^2}{2} \nabla \log p_t(\mathbf{x}_t) \tag{21}$$

$$= \gamma_t^0 \mathbf{v}_t(\mathbf{x}_t) + \left(\gamma_t^1 - \frac{\sigma_t^2}{2}\right) \nabla_x \log p_t(\mathbf{x}_t) + \sum_{k=2}^{K} \gamma_t^k \nabla_x^k \log p_t(\mathbf{x}_t). \tag{R}$$

This establishes the continuity equation for the K-th order IDFF with the effective drift $\mathbf{w}_t$.

Part 3: Boundary Conditions and Sampling Validity **Goal**: Show that the boundary conditions ensure convergence to the original CFM at endpoints.

Proof: As $t \to 0$ and $t \to 1$:

1. $\sigma_t \to 0$ by construction (typically $\sigma_t = \sigma_0\sqrt{t(1-t)}$ or similar)
2. $\{\gamma_t^k\}_{k=1}^K \to 0$ by assumption
3. $\gamma_t^0 \to 1$ by the normalization condition $\sum_{k=0}^K \gamma_t^k = 1$

Therefore:

$$\lim_{t\to 0,1} \tilde{\mathbf{v}}_t(\mathbf{x}_t) = \lim_{t\to 0,1} \left[\gamma_t^0 \mathbf{v}_t(\mathbf{x}_t) + \sum_{k=1}^K \gamma_t^k \nabla_x^k \log p_t(\mathbf{x}_t)\right] = \mathbf{v}_t(\mathbf{x}_t). \tag{S}$$

Similarly for the effective drift:

$$\lim_{t\to 0,1} \mathbf{w}_t(\mathbf{x}_t) = \mathbf{v}_t(\mathbf{x}_t). \tag{T}$$

This ensures that the IDFF flow matches the original CFM at the boundaries, preserving:

- The source distribution at $t = 0$: $p_0(\mathbf{x}_0) = q_0(\mathbf{x}_0)$
- The target distribution at $t = 1$: $p_1(\mathbf{x}_1) = q_1(\mathbf{x}_1)$

Since marginal preservation holds throughout $[0, 1]$ (Part 1) and boundary conditions are satisfied, samples generated by the SDE in equation (M) follow the correct probability path from source to target distribution.

A.2 Corollary Proofs

Corollary 1: Reduction to Standard CFM When $K = 0$ and $\gamma_t^0 = 1$:

$$\tilde{\mathbf{v}}_t(\mathbf{x}_t) = 1 \cdot \mathbf{v}_t(\mathbf{x}_t) + \sum_{k=1}^0 \gamma_t^k \nabla_x^k \log p_t(\mathbf{x}_t) = \mathbf{v}_t(\mathbf{x}_t). \tag{22}$$

The effective drift becomes:

$$\mathbf{w}_t(\mathbf{x}_t) = \mathbf{v}_t(\mathbf{x}_t) - \frac{\sigma_t^2}{2}\nabla \log p_t(\mathbf{x}_t). \tag{23}$$

As $\sigma_t \to 0$ at the boundaries, we recover $\mathbf{w}_t = \mathbf{v}_t$, which is standard CFM.

Corollary 2: Connection to Score-Based Models When $\gamma_t^0 = 0$, $K = 1$, and $\gamma_t^1 = \alpha$ (constant due to normalization):

$$\tilde{\mathbf{v}}_t(\mathbf{x}_t) = 0 \cdot \mathbf{v}_t(\mathbf{x}_t) + \alpha \nabla_x \log p_t(\mathbf{x}_t) = \alpha \nabla_x \log p_t(\mathbf{x}_t). \tag{24}$$

The effective drift in the SDE formulation becomes:

$$\mathbf{w}_t(\mathbf{x}_t) = \alpha \nabla_x \log p_t(\mathbf{x}_t) - \frac{\sigma_t^2}{2}\nabla_x \log p_t(\mathbf{x}_t) = \left(\alpha - \frac{\sigma_t^2}{2}\right)\nabla_x \log p_t(\mathbf{x}_t). \tag{25}$$

This recovers the score-based diffusion framework where the drift is proportional to the score function, establishing the connection to stochastic interpolants as shown in Albergo et al. (2023).

Mathematical connection to momentum optimization. Consider the optimization problem $\min_{\mathbf{x}} f(\mathbf{x})$ where $f(\mathbf{x}) = -\log p(\mathbf{x})$. The momentum update rule:

$$\mathbf{p}_{t+1} = \beta \mathbf{p}_t + \nabla f(\mathbf{x}_t) = \beta \mathbf{p}_t - \nabla \log p(\mathbf{x}_t) \tag{26}$$

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \mathbf{p}_{t+1} \tag{27}$$

In the continuous-time limit with appropriate scaling, this yields:

$$\dot{\mathbf{x}}_t = \mathbf{v}_t(\mathbf{x}_t) + \alpha \nabla \log p_t(\mathbf{x}_t) \tag{28}$$

which directly motivates our augmented vector field formulation.

A.3 Remarks on Implementation

For practical implementation with Gaussian probability paths $p_t(\mathbf{x}_t|\mathbf{x}_1) = \mathcal{N}(\mu_t, \sigma_t^2\mathbf{I})$ where $\mu_t = t\mathbf{x}_1 + (1-t)\mathbf{x}_0$:

1. First-order: $\nabla_x \log p_t(\mathbf{x}_t|\mathbf{x}_1) = -\frac{\mathbf{x}_t - \mu_t}{\sigma_t^2}$
2. Second-order: $\nabla_x^2 \log p_t(\mathbf{x}_t|\mathbf{x}_1) = -\frac{1}{\sigma_t^2}\mathbf{I}$
3. Higher orders: $\nabla_x^k \log p_t(\mathbf{x}_t|\mathbf{x}_1) = 0$ for $k \geq 3$

This simplification makes K-th order IDFF particularly efficient for Gaussian paths, as only the first two orders contribute non-zero terms. □

A.4 Practical Implementation Notes

For practical implementation, the higher-order derivatives $\nabla_x^k \log p_t(\mathbf{x}_t|\mathbf{x}_1)$ can be computed through two main approaches, each with distinct computational trade-offs.

For the standard CFM probability path $p_t(\mathbf{x}_t|\mathbf{x}_1) = \mathcal{N}(\boldsymbol{\mu}_t, \sigma_t^2\mathbf{I})$ with $\boldsymbol{\mu}_t = t\mathbf{x}_1 + (1-t)\mathbf{x}_0$, the higher-order derivatives have closed-form expressions:

$$\nabla_x \log p_t(\mathbf{x}_t|\mathbf{x}_1) = -\frac{\mathbf{x}_t - \boldsymbol{\mu}_t}{\sigma_t^2} = -\frac{\boldsymbol{\epsilon}_0}{\sigma_t}, \tag{29}$$

$$\nabla_x^2 \log p_t(\mathbf{x}_t|\mathbf{x}_1) = -\frac{1}{\sigma_t^2}\mathbf{I}, \tag{30}$$

$$\nabla_x^k \log p_t(\mathbf{x}_t|\mathbf{x}_1) = 0 \quad \text{for } k \geq 3, \tag{31}$$

where $\boldsymbol{\epsilon}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is the noise added during the forward process. This property significantly simplifies implementation for Gaussian paths, as only first and second-order terms are non-zero.

Higher-order scores are computed using dedicated network heads that output tensors of appropriate dimensions. For first-order terms $\nabla_x \log p_t(\mathbf{x}_t|\mathbf{x}_1)$, the output dimension equals the data dimension d. Full k-th order tensors generally require d^k parameters, growing exponentially with both order and dimensionality.

The parameter scaling $O(d^k)$ for k-th order terms can become prohibitive for high-dimensional data. We address this computational challenge through diagonal approximations for second-order and higher terms, requiring only d parameters per order regardless of k, which significantly reduces the computational burden. Alternatively, low-rank factorizations can approximate the k-th order tensor as a sum of rank-1 components, reducing the parameter count from $O(d^k)$ to $O(rd^{k-1})$ where r is the rank. In practice, we use $K = 1$ or $K = 2$ for most applications, reserving higher orders for specific cases where the computational cost can be justified.

Our experimental validation follows a computational hierarchy based on data complexity. For toy examples with $d = 2$ dimensional data, we use $K > 2$ with full tensors to demonstrate the conceptual benefits of higher-order momentum terms. For real datasets, we employ $K = 2$ with diagonal approximations, providing substantial benefits while remaining computationally tractable. For high-dimensional applications, we default to $K = 1$ (first-order momentum) to achieve an optimal efficiency-performance trade-off.

For non-Gaussian distributions, higher-order derivatives can be computed through automatic differentiation by repeatedly applying gradient operators, requiring k backward passes per evaluation, or through dedicated network heads that enable constant-time inference but require additional parameters. The choice between these approaches depends on available memory, computational budget, and deployment requirements. Our theoretical framework supports arbitrary orders, but practical implementations must balance computational efficiency with performance gains.

B Training Objective Derivations

As described in Theorem 2 of (Lipman et al., 2022), the FM loss is defined as:

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t,\mathbf{x}_1,p_t(\mathbf{x}_t|\mathbf{x}_1)} \left\| \mathbf{v}_t(\mathbf{x}_t) - \hat{\mathbf{v}}_t(\mathbf{x}_t) \right\|^2 \tag{32}$$

The CFM loss is equivalent to the FM loss up to a constant value, implying that $\mathcal{L}_{\text{FM}}(\theta) = \mathcal{L}_{\text{CFM}}(\theta)$. Based on this equivalence, we can express the training objective for approximating the vector field in equation 11 as:

$$\mathcal{L}^0_{\text{IDFF}}(\theta) = \mathbb{E}_{t,\mathbf{x}_1,p_t(\mathbf{x}_t|\mathbf{x}_1)} \left\| \hat{\mathbf{v}}_t(\mathbf{x}_t \mid \mathbf{x}_1; \theta) - \mathbf{v}_t(\mathbf{x}_t|\mathbf{x}_1) \right\|^2 \tag{33}$$

To recover unbiased samples, we use the true conditional vector field $\mathbf{v}_t(\mathbf{x}_t|\mathbf{x}_1) = \frac{\mathbf{x}_1 - \mathbf{x}_t}{1-t}$ and our reparameterization $\hat{\mathbf{v}}_t(\mathbf{x}_t|\mathbf{x}_1;\theta) = \frac{\hat{\mathbf{x}}_1(\mathbf{x}_t,t;\theta) - \mathbf{x}_t}{1-t}$, resulting in:

$$\mathcal{L}^0_{\text{IDFF}}(\theta) = \mathbb{E}_{t,\mathbf{x}_1,p_t(\mathbf{x}_t|\mathbf{x}_1)} \left[\left\| \frac{\hat{\mathbf{x}}_1(\mathbf{x}_t,t;\theta) - \mathbf{x}_t}{1-t} - \frac{\mathbf{x}_1 - \mathbf{x}_t}{1-t} \right\|^2 \right] \tag{34}$$

Simplifying:

$$= \mathbb{E}_{t,\mathbf{x}_1,p_t(\mathbf{x}_t|\mathbf{x}_1)} \left[\frac{1}{(1-t)^2} \left\| \hat{\mathbf{x}}_1(\mathbf{x}_t,t;\theta) - \mathbf{x}_1 \right\|^2 \right] \tag{35}$$

Setting $\beta(t) = \frac{1}{1-t}$, we obtain:

$$\mathcal{L}^0_{\text{IDFF}}(\theta) = \mathbb{E}_{t,\mathbf{x}_1,p_t(\mathbf{x}_t|\mathbf{x}_1)} \left[\beta(t)^2 \left\| \hat{\mathbf{x}}_1(\mathbf{x}_t,t;\theta) - \mathbf{x}_1 \right\|^2 \right] \tag{36}$$

To approximate $\nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t|\mathbf{x}_1)$, we employ a time-dependent score-based model. For Gaussian paths with $p_t(\mathbf{x}_t|\mathbf{x}_1) = \mathcal{N}(\mu_t, \sigma_t^2 \mathbf{I})$ where $\mu_t = t\mathbf{x}_1 + (1-t)\mathbf{x}_0$, we have:

$$\nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t|\mathbf{x}_1) = -\frac{\mathbf{x}_t - \mu_t}{\sigma_t^2} = -\frac{\boldsymbol{\epsilon}_0}{\sigma_t} \tag{37}$$

where $\boldsymbol{\epsilon}_0 = (\mathbf{x}_t - \mu_t)/\sigma_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is the standardized noise.

Since $\nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t|\mathbf{x}_1)$ approaches infinity as $t \to 0$ or $t \to 1$, we standardize by predicting the noise $\boldsymbol{\epsilon}_0$ directly. This leads to the score loss:

$$\mathcal{L}^1_{\text{IDFF}}(\theta) = \mathbb{E}_{t,\mathbf{x}_1,p_t(\mathbf{x}_t|\mathbf{x}_1)} \left[\lambda_1(t)^2 \left\| \hat{\boldsymbol{\epsilon}}^1(\mathbf{x}_t,t;\theta) - \nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t|\mathbf{x}_1) \right\|^2 \right] \tag{38}$$

where $\lambda_1(t)$ is a positive weighting function. With proper choice of $\lambda_1(t)$, this loss is equivalent to the original DDPM loss up to a constant value, according to Theorem 1 of (Song et al., 2020a).

Combining the two components, the first-order IDFF loss function is:

$$\mathcal{L}_{\text{IDFF}}(\theta, 1) = \mathcal{L}^0_{\text{IDFF}}(\theta) + \mathcal{L}^1_{\text{IDFF}}(\theta) \tag{39}$$

B.1 Higher-Order IDFF: Extended Loss Derivation

We extend the IDFF training objective to incorporate higher-order gradients of the log-density, $\nabla^k \log p_t(\mathbf{x}_t|\mathbf{x}_1)$. The auxiliary term $\boldsymbol{\xi}_t(\mathbf{x}_t)$ now includes higher-order derivatives:

$$\boldsymbol{\xi}_t(\mathbf{x}_t) = \sum_{k=1}^{K} \gamma_t^k \, \nabla^k \log p_t(\mathbf{x}_t), \tag{40}$$

where each γ_t^k controls the contribution of the k-th order derivative.

For Gaussian paths, the higher-order derivatives have closed forms:

$$\nabla \log p_t(\mathbf{x}_t|\mathbf{x}_1) = -\frac{\mathbf{x}_t - \mu_t}{\sigma_t^2} \tag{41}$$

$$\nabla^2 \log p_t(\mathbf{x}_t|\mathbf{x}_1) = -\frac{1}{\sigma_t^2}\mathbf{I} \tag{42}$$

$$\nabla^k \log p_t(\mathbf{x}_t|\mathbf{x}_1) = 0 \quad \text{for } k \geq 3 \tag{43}$$

The generalized higher-order score loss penalizes all orders up to K:

$$\mathcal{L}^{(K)}_{\text{IDFF}}(\theta) = \mathbb{E}_{t,\mathbf{x}_1,\,p_t(\mathbf{x}_t|\mathbf{x}_1)} \left[\sum_{k=1}^{K} \lambda_k(t)^2 \left\| \hat{\boldsymbol{\epsilon}}^k(\mathbf{x}_t,t;\theta) - \nabla^k \log p_t(\mathbf{x}_t|\mathbf{x}_1) \right\|^2 \right], \tag{44}$$

where $\lambda_k(t)$ is chosen separately for each order k to accommodate different scaling behaviors.

Combining the flow-matching term with the higher-order score-matching terms yields the complete K-th order IDFF loss:

$$\mathcal{L}_{\text{IDFF}}(\theta, K) = \mathcal{L}^0_{\text{IDFF}}(\theta) + \mathcal{L}^{(K)}_{\text{IDFF}}(\theta) \tag{45}$$

$$= \mathbb{E}_{t,\mathbf{x}_1,\, p_t(\mathbf{x}_t|\mathbf{x}_1)} \left[\beta(t)^2 \left\| \hat{\mathbf{x}}_1(\mathbf{x}_t, t; \theta) - \mathbf{x}_1 \right\|^2 \right]$$

$$+ \mathbb{E}_{t,\mathbf{x}_1,\, p_t(\mathbf{x}_t|\mathbf{x}_1)} \left[\sum_{k=1}^{K} \lambda_k(t)^2 \left\| \hat{\boldsymbol{\epsilon}}^k(\mathbf{x}_t, t; \theta) - \nabla^k \log p_t(\mathbf{x}_t|\mathbf{x}_1) \right\|^2 \right] \tag{46}$$

Why sample prediction over velocity learning. The reparameterization $\hat{\mathbf{v}}_t = (\hat{\mathbf{x}}_1 - \mathbf{x}_t)/(1-t)$ offers several advantages:

1. **Numerical stability**: Near $t = 1$, directly learning $\mathbf{v}_t$ suffers from the $(1-t)^{-1}$ singularity. Sample prediction avoids this by learning a bounded target.
2. **Interpretability**: The network directly optimizes for reconstruction quality, providing clearer training signals.
3. **Score computation**: Given $\hat{\mathbf{x}}_1$, we can derive the score via $\nabla \log p_t(\mathbf{x}_t|\hat{\mathbf{x}}_1) = -(\mathbf{x}_t - t\hat{\mathbf{x}}_1 - (1-t)\mathbf{x}_0)/\sigma_t^2$, enabling efficient momentum computation.

Practical Considerations

- **Dimensionality:** Higher-order derivatives become large tensors in high-dimensional spaces. For data dimension d, the k-th order gradient has $O(d^k)$ components, making $K > 2$ challenging without special structure. We provide a solution for this in more detail in Appendix A.4. In practice, we find that $K = 2$ offers the best trade-off between acceleration and computational overhead, with diminishing returns for $K > 2$.
- **Weight Schedules:** Each $\lambda_k(t)$ handles potential singularities in $\nabla^k \log p_t$ near $t = 0$ or $t = 1$. For Gaussian paths, $\lambda_1(t) = \sigma_t$ and $\lambda_2(t) = \sigma_t^2$ provide proper normalization. We parameterize the interpolation coefficients as $\gamma_t^k = \alpha_k t^{a_k}(1-t)^{b_k}$ where α_k controls the strength and (a_k, b_k) control the temporal weighting. Typical values are $a_k = b_k = 1$ for symmetric weighting, though asymmetric schedules can be beneficial for certain distributions.
- **Model Architecture:** One may output all $\hat{\boldsymbol{\epsilon}}^k$ from a single network with multiple heads or treat them as separate networks, depending on memory constraints and desired efficiency. We adopt a shared backbone with separate projection heads for each order, allowing feature reuse while maintaining flexibility. The backbone uses standard U-Net or transformer architectures, with the final layer split into $K+1$ heads: one for $\hat{\mathbf{x}}_1$ and K for $\{\hat{\boldsymbol{\epsilon}}^k\}_{k=1}^K$.
- **Numerical Stability:** Near boundaries ($t \approx 0$ or $t \approx 1$), the denominator $(1-t)$ in the velocity field can cause instabilities. We clip the integration range to $[t_{\min}, t_{\max}] = [0.001, 0.999]$ in practice, with negligible impact on sample quality. Additionally, we use gradient clipping during training to prevent exploding gradients from higher-order terms.
- **Solver Choice:** While our experiments use Euler integration for simplicity, IDFF is compatible with higher-order solvers (e.g., Heun, RK4). The combination of IDFF with adaptive step-size solvers remains an interesting direction for future work.
- **Hyperparameter Selection:** The key hyperparameters are the order K, the momentum strengths $\{\alpha_k\}$, and the noise scale σ_0. We find $K \in \{1, 2, 3\}$, $\alpha_k \in [0.1, 1.0]$, and $\sigma_0 \in [0.01, 0.2]$ work well across diverse datasets. Grid search over these ranges typically yields good results, though adaptive schemes that adjust α_k based on training dynamics show promise.

Likelihood evaluation. Following the standard approach for continuous normalizing flows, we evaluate likelihood using the instantaneous change of variables formula:

$$\log p_1(\mathbf{x}_1) = \log p_0(\mathbf{x}_0) - \int_0^1 \nabla \cdot \mathbf{w}_t(\mathbf{x}_t) dt, \tag{47}$$

where the divergence of $\mathbf{w}_t$ can be computed efficiently using automatic differentiation or estimated via the Hutchinson trace estimator.

C IDFF Training Algorithm for Static Data

In the case of static datasets where samples are drawn i.i.d. from a fixed distribution, the IDFF training algorithm simplifies yet retains its key features. The generative process constructs a flow from a base distribution $p_0(\mathbf{x}_0) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ to the data distribution $p_1(\mathbf{x}_1) = p_{\text{data}}$, with intermediate states $\mathbf{x}_t$ sampled via the Gaussian conditional path defined in Equation equation 13.

At each training step, we sample $\mathbf{x}_0 \sim p_0$ and $\mathbf{x}_1 \sim p_1$. These can be coupled independently as $(\mathbf{x}_0, \mathbf{x}_1) \sim p_0(\mathbf{x}_0)p_1(\mathbf{x}_1)$ or through an optimal transport plan π^* that minimizes $\mathbb{E}[\|\mathbf{x}_1 - \mathbf{x}_0\|^2]$. We then sample an interpolation time $t \sim \mathcal{U}[0, 1]$ and construct the intermediate point $\mathbf{x}_t \sim \mathcal{N}(\mu_t, \sigma_t^2\mathbf{I})$ where $\mu_t = t\mathbf{x}_1 + (1-t)\mathbf{x}_0$ and $\sigma_t = \sigma_0\sqrt{t(1-t)}$, with bandwidth parameter σ_0 controlling path stochasticity.

The model jointly learns to: (i) predict the clean sample through the denoiser network $\hat{\mathbf{x}}_1(\mathbf{x}_t, t; \theta)$, and (ii) approximate the k-th order derivatives of the log-density through momentum networks $\hat{\boldsymbol{\epsilon}}^k(\mathbf{x}_t, t; \theta)$ for $k \in \{1, \ldots, K\}$. The training uses the IDFF loss objective defined in Equation equation 16, which combines the denoising loss with momentum matching terms. For the Gaussian conditional path, the ground truth momentum terms are $\nabla^k \log p_t(\mathbf{x}_t|\mathbf{x}_1)$, which have closed-form expressions as described in Section 3. This loss operates entirely in sample space, ensuring compatibility with either explicit network heads for each derivative order or automatic differentiation for computing higher-order terms. The full training procedure is summarized in Algorithm 1.

D Training and Sampling with IDFF for Time-Series Data

Adapting IDFF to time-series data requires extending the framework to handle sequential dependencies while maintaining the momentum-augmented flow structure. The key innovation is a dual-time formulation: a discrete index $n \in \{1, \ldots, N\}$ represents the sequence position, while the continuous variable $t \in [0, 1]$ parameterizes the flow from noise to data at each sequence step.

D.1 Time-Series Formulation

For sequences $\mathbf{x}_{1:N} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ where $\mathbf{x}_n \in \mathbb{R}^d$, we model the joint distribution $p(\mathbf{x}_{1:N})$ through sequential conditional flows. The networks receive both temporal indices:

$$\hat{\mathbf{x}}_1(\mathbf{x}_t, t, n; \theta), \quad \hat{\boldsymbol{\epsilon}}^k(\mathbf{x}_t, t, n; \theta) \quad \text{for } k \in \{1, \ldots, K\}. \tag{48}$$

The crucial modification for temporal coherence is the initialization strategy. For the first sequence step ($n = 1$), we sample from the standard base distribution: $\mathbf{x}_0^1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. For subsequent steps ($n > 1$), we initialize from a distribution centered at the previous output:

$$\mathbf{x}_0^n \sim \mathcal{N}(\mathbf{x}_1^{n-1}, \sigma_0^2\mathbf{I}), \tag{49}$$

creating a Markovian structure that maintains temporal continuity while allowing controlled stochasticity through σ_0.

D.2 Training Procedure

During training, we randomly sample sequence positions and construct conditional paths accordingly. For position $n > 1$, the base distribution shifts from $p_0 = \mathcal{N}(\mathbf{0}, \mathbf{I})$ to $p_0^n = p(\mathbf{x}_1^{n-1})$, the distribution of the previous step's output. The training objective remains:

$$\mathcal{L}_{\text{IDFF}} = \mathbb{E}_{n,t,\mathbf{x}_0,\mathbf{x}_1,\mathbf{x}_t}\left[\beta(t)^2\|\hat{\mathbf{x}}_1(\mathbf{x}_t, t, n; \theta) - \mathbf{x}_1\|^2 + \sum_{k=1}^{K}\lambda_k(t)^2\|\hat{\boldsymbol{\epsilon}}^k(\mathbf{x}_t, t, n; \theta) - \nabla^k \log p_t(\mathbf{x}_t|\mathbf{x}_1)\|^2\right], \tag{50}$$

where the conditional path $p_t(\mathbf{x}_t|\mathbf{x}_0, \mathbf{x}_1) = \mathcal{N}(\mathbf{x}_t; \mu_t, \sigma_t^2\mathbf{I})$ with $\mu_t = t\mathbf{x}_1 + (1-t)\mathbf{x}_0$ and $\sigma_t = \sigma_0\sqrt{t(1-t)}$.

For the Gaussian path, the ground truth targets are:

$$\nabla \log p_t(\mathbf{x}_t|\mathbf{x}_1) = -\frac{\mathbf{x}_t - \mu_t}{\sigma_t^2} = -\frac{\boldsymbol{\epsilon}_0}{\sigma_t}, \tag{51}$$

$$\nabla^2 \log p_t(\mathbf{x}_t|\mathbf{x}_1) = -\frac{1}{\sigma_t^2}\mathbf{I}, \tag{52}$$

$$\nabla^k \log p_t(\mathbf{x}_t|\mathbf{x}_1) = 0 \quad \text{for } k \geq 3, \tag{53}$$

where $\boldsymbol{\epsilon}_0 = (\mathbf{x}_t - \mu_t)/\sigma_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

D.3 Sampling Procedure

Sequential generation proceeds autoregressively. For each sequence position n, we integrate the momentum-augmented flow from $t = 0$ to $t = 1$:

$$\mathbf{w}_t^n = \gamma_t^0 \frac{\hat{\mathbf{x}}_1(\mathbf{x}_t^n, t, n; \theta) - \mathbf{x}_t^n}{1-t} + \left(\gamma_t^1 - \frac{\sigma_t^2}{2}\right)\hat{\boldsymbol{\epsilon}}^1(\mathbf{x}_t^n, t, n; \theta) + \sum_{k=2}^{K} \gamma_t^k \hat{\boldsymbol{\epsilon}}^k(\mathbf{x}_t^n, t, n; \theta), \tag{54}$$

where $\mathbf{w}_t^n$ is the effective drift incorporating momentum corrections and compensating diffusion.

The integration follows:

$$\mathbf{x}_{t+\Delta t}^n = \mathbf{x}_t^n + \mathbf{w}_t^n \Delta t + \sigma_t\sqrt{\Delta t} \cdot \boldsymbol{\eta}, \quad \boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{55}$$

with step size $\Delta t = 1/\text{NFE}$. After completing integration for position n, the final sample $\mathbf{x}_1^n$ serves as the mean for initializing the next position's flow.

Algorithm 2 IDFF Training for Time-Series

Input: Data distribution p_1, sequence length N, order K, bandwidth σ_0, weights $\{\lambda_k\}$, β
while training **do**
  Sample $n \sim \mathcal{U}\{1, \ldots, N\}$
  if $n = 1$ **then**
    $\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\mathbf{x}_1 \sim p_1(\mathbf{x}_1^1)$
  else
    $\mathbf{x}_0 \sim p_1(\mathbf{x}_1^{n-1})$, $\mathbf{x}_1 \sim p_1(\mathbf{x}_1^n)$
  end if
  Optionally apply OT coupling: $(\mathbf{x}_0, \mathbf{x}_1) \sim \pi^*$
  Sample $t \sim \mathcal{U}[0, 1]$
  $\mu_t \leftarrow t\mathbf{x}_1 + (1-t)\mathbf{x}_0$, $\sigma_t \leftarrow \sigma_0\sqrt{t(1-t)}$
  Sample $\mathbf{x}_t \sim \mathcal{N}(\mu_t, \sigma_t^2\mathbf{I})$
  Compute loss $\mathcal{L}_{\text{IDFF}}$ and update θ
end while

Algorithm 3 IDFF Sampling for Time-Series

Input: Trained networks, sequence length N, NFE, coefficients $\{\gamma_t^k\}_{k=0}^K$
for $n = 1$ to N **do**
  if $n = 1$ **then**
    Initialize $\mathbf{x}_0^n \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  else
    Initialize $\mathbf{x}_0^n \sim \mathcal{N}(\mathbf{x}_1^{n-1}, \sigma_0^2\mathbf{I})$
  end if
  for $i = 0$ to NFE $- 1$ **do**
    $t \leftarrow i/\text{NFE}$
    Compute $\mathbf{w}_t^n$ using momentum-augmented drift
    Update $\mathbf{x}_t^n \leftarrow \mathbf{x}_t^n + \mathbf{w}_t^n \Delta t + \sigma_t\sqrt{\Delta t} \cdot \boldsymbol{\eta}$
  end for
  Store $\mathbf{x}_1^n \leftarrow \mathbf{x}_t^n$
end for
Return $\{\mathbf{x}_1^n\}_{n=1}^N$

D.4 Computational Considerations

The time-series extension scales computational cost by the sequence length N. For order K, the cost is $O(N \cdot \text{NFE} \cdot K)$ using explicit network heads for derivatives, or $O(N \cdot \text{NFE} \cdot K^2)$ using automatic differentiation. The Markovian initialization creates implicit temporal dependencies without requiring recurrent architectures, maintaining efficiency while capturing sequential structure.

When $N = 1$, the framework reduces exactly to static IDFF, providing a unified approach across data modalities. The bandwidth parameter σ_0 controls the trade-off between temporal coherence (small σ_0) and diversity (large σ_0) in sequential generation.

E Implementation Details

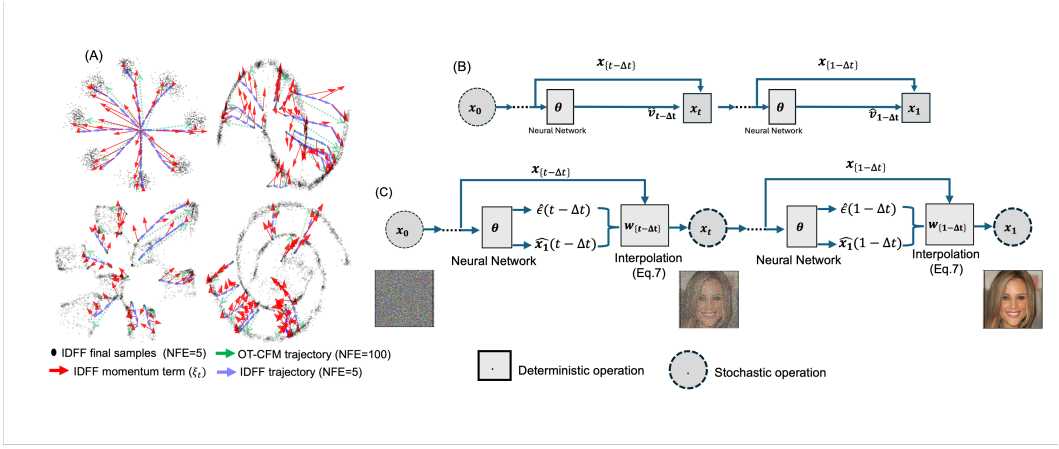

Figure 4: A) Comparison of trajectory sampling between 1st-order IDFF and OT-CFMs: The figure displays 4096 final samples generated by IDFF. As shown, IDFF takes larger steps toward the target distribution, guided by the momentum term. B) OT-CFMs sampling process. C) IDFF sampling process. In this process, $\hat{\mathbf{x}}_1(.)$ approximates the data sample $\mathbf{x}_1$, $\hat{\epsilon}(.)$ approximates the scores associated with $\boldsymbol{\xi}_t$, and $\mathbf{w}_t(.)$ is the calculated vector field by equation 11. The key difference between IDFF and OT-CFMs is the vector field generation: 1st-order IDFF generates $\hat{\mathbf{x}}_1(.)$ and $\hat{\epsilon}(.)$ in sample space and then reconstructs the vector field, and uses momentum to guide the sampling process.

For most image datasets, we employed a CNN-based UNet (Dhariwal & Nichol, 2021) to simultaneously model both $\hat{\mathbf{x}}_1(., t; \theta)$ and $\boldsymbol{\epsilon}(., t; \theta)$, while ImageNet-64 utilized a DiT architecture ('DiT-L/4')(Peebles & Xie, 2022). Implementation involved doubling the network input channels and feeding the augmented input $(\mathbf{x}_t, \mathbf{x}_t)$, then splitting the outputs into $(\hat{\mathbf{x}}_1(., t; \theta), \boldsymbol{\epsilon}(., t; \theta))$.

For fair comparison with existing models, we evaluated performance using standard metrics: negative log-likelihood (NLL) using equation 47 measured in bits per dimension (BPD) Lipman et al. (2022), Frechet Inception Distance (FID) for sample quality, and average number of function evaluations (NFE) required for the reported metrics, averaged over 50k samples.

Network configuration: For our experiments on image generation (except the ImageNet-64) and SST forecasting, we utilize the ScoreSDE model architecture as described in Song et al. (2020b). Detailed configurations of the networks tailored for various datasets are provided in Table 5.

Training hyper-params. In Table 6, we provide training hyperparameters for unconditional image generation and SST forecasting problem.

Table 5: ScoreSDE network configuration for different datasets.

	CIFAR-10	CelebA64	CelebA 256	Church & Bed	SST
# of ResNet blocks	2/4	2	2	2	2
Base channels	128	128	128	256	128
Channel multiplier	1,2,2,2	1,2,2,4,4	1,1,2,2,4,4	1,1,2,2,4,4	1,2,4
Attention resolutions	16	16	16	16	16
Label dimensions	1	1	1	1	10
Params (M)	65.6	102.14	453.45	108.41	55.39

Table 6: Hyper-parameters of ScoreSDE network.

	ImageNet-64	CIFAR-10	CelebA64	CelebA 256	Church & Bed	SST
lr	1e-4	1e-4	1e-5	1e-5	1e-5	1e-4
Batch size	64	128	128	16	16	8
# of iterations	3M	700K	1.2M	1.5M	1.5M	700K
# of GPUs	4	1	1	1	1	1

F ADDITIONAL RESULTS FOR IMAGE GENERATION EXPERIMENT

Table 7: Effect of time scheduling strategies on IDFF performance for CIFAR-10.

Time Sampling Strategy	FID↓
Linear	3.22
Logarithmic	3.11
Beta Schedule	2.98
Cosine	**2.78**

We also assessed IDFF performance in generating images against fast diffusion process models with NFE=5. As Table 9 IDFF achieves a significantly better FID score (8.53) compared to UniPC (23.71) and DPM-Solver-v3 (12.76), while also boasting the fastest wall-clock time (0.34 seconds) among all solvers. This highlights IDFF's ability to generate high-quality samples with minimal computational overhead, making it ideal for real-time applications. Even at NFE=10, IDFF remains superior with a FID (2.78) and the fastest wall-clock time (0.52 seconds), demonstrating its efficiency and scalability. These results suggest that IDFF hits a balance between sample quality and computational speed that lends itself to speed.

Table 8: Comparison of FID and NFE metrics between IDFF and various methods on the CelebA (64×64) dataset.

Model	FID↓	NFE↓
DDPM	45.20	100
DDIM	13.73	20
DDIM	17.33	10
FastDPM	12.83	50
IDFF (Ours)	**11.83**	**10**

Table 9: Comparison of FID↓ performance and sampling times (Wall-clock) between IDFF and fast diffusion sampling methods for NFE=5 and NFE=10, evaluated on 50k samples.

Method	FID (NFE=5)	Wall-clock (sec, NFE=5)	FID (NFE=10)	Wall-clock (sec, NFE=10)
UniPC (Zhao et al., 2024)	23.71	0.62	3.93	1.05
DPM-Solver-v3 (Zheng et al., 2023)	12.76	0.49	3.40	0.92
IDFF	**8.53**	**0.34**	**2.78**	**0.52**

Table 10: Summary of FLD performance metrics for various generative models, based on the results reported in Jiralerspong et al. (2023). For this experiment, IDFF utilized the ScoreSDE model to generate 10K samples with NFE=10. Results for the other models are adapted from Table 1 of Jiralerspong et al. (2023).

Model	FLD↓
ACGAN-Mod	24.22
LOGAN	18.94
BigGAN-Deep	9.28
MHGAN	8.84
StyleGAN2-ada	6.86
iDDPM-DDIM	5.63
IDFF (Ours)	5.62
StyleGAN-XL	5.58
PFGMPP	4.58

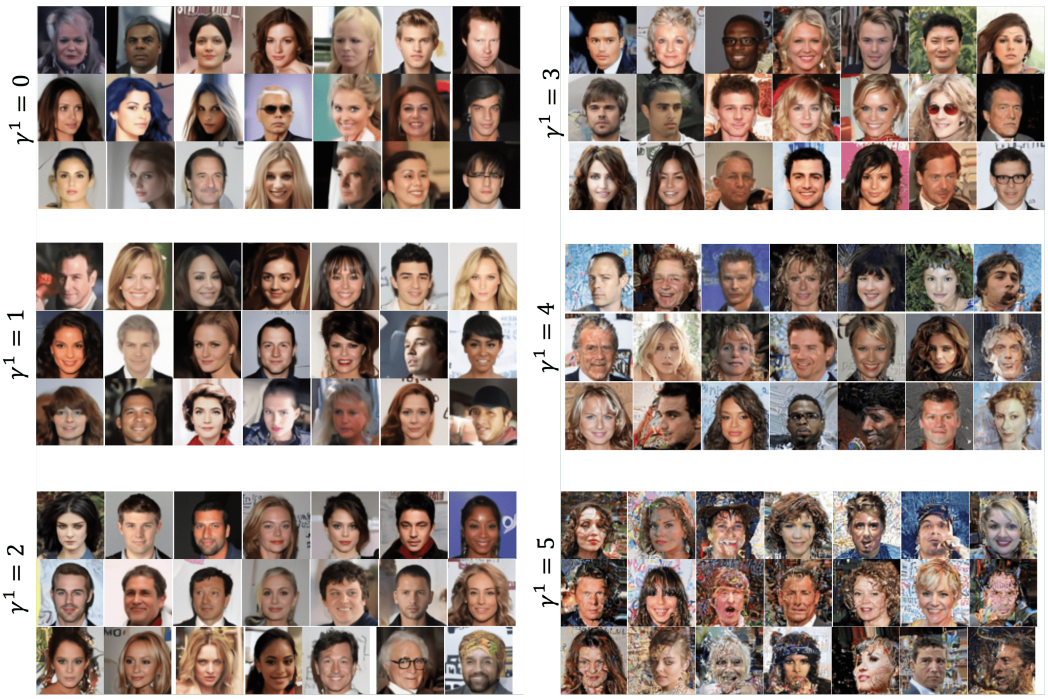

Figure 5: Generated samples for CelebA64 (64×64) dataset with different $\gamma_t^1 = \sigma_t^2 \gamma^1$s and $NFE = 10$.

G 3D-ATTRACTORS

In this experiment, we assess IDFF's performance in generating trajectories of chaotic systems from scratch. We generate trajectories with $K = 2000$ samples in each trajectory from 3D attractors, specifically the Lorenz and Rössler attractors, which are chaotic systems with nonlinear dynamics. The parameters for the Lorenz and Rössler models are set to $\sigma = 10, \rho = 28, \beta = 8/3$ and $a = .2, b = .2, c = 5.7$, respectively, to produce complex trajectories in 3D space. We then train the IDFF model based on these trajectories. We used the training 2 and sampling 3 algorithms suggested for time-series data.

To model each attractor, we use an MLP with two hidden layers of 128 dimensions and two separate heads for $\hat{\mathbf{x}}_1(., t, k; \theta)$ and $\epsilon(\mathbf{x}_t, t, k; \theta)$. Additionally, we incorporate two separate embedding layers for embedding t and k, which are directly concatenated with the first hidden layer of the MLP. The

Figure 6: Generated samples for CIFAR-10 (32×32) dataset with different $\gamma_t^1 = \sigma_t^2 \gamma^1$s and $NFE = 10$.

optimized IDFF successfully generates samples of these trajectories from scratch. The generated trajectories are shown in Figure 11.

The quality of the results demonstrates that IDFF can successfully simulate the behaviors of highly nonlinear and nonstationary systems such as attractors.

H SST FORECASTING VISUALIZATION

For this task, we employed a class-conditional UNet, with the class encoder handling long-range time dependencies to enable continuous forecasting. Network structure and hyperparameters are detailed in Appendix E. We utilized the training 2 and sampling 3 algorithms designed for time-series data.

I 2D-SIMULATED STATIC DATA AND TIME-SERIES

Additional results for 2D simulations for both static and time-series generation are shown in Figure14.

Figure 7: Generated samples for CelebA-HQ (256×256) dataset with $\sigma_0 = 0.2$ and $NFE = 10$.

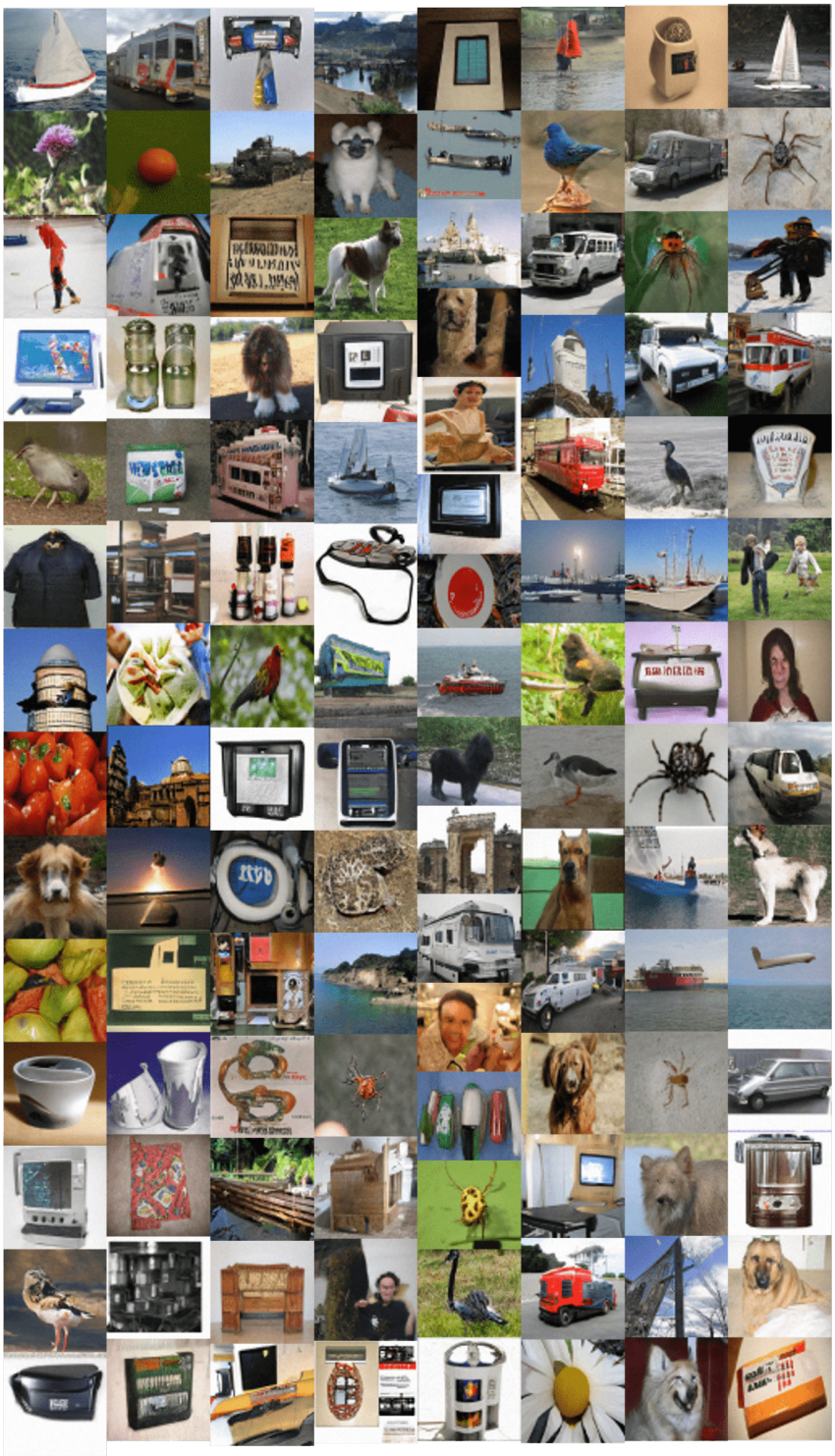

Figure 8: Generated samples for ImageNet64 dataset with $\sigma_0 = 0.2$ and $NFE = 10$.

Figure 9: Generated samples for LSUN-church (256×256) dataset with different $\gamma_t^1 = \sigma_t^2 \gamma^1$s and $NFE = 10$.

Figure 10: Generated samples for LSUN-bed (256×256) dataset with different $\gamma_t^1 = \sigma_t^2 \gamma^1$s and $NFE = 10$.

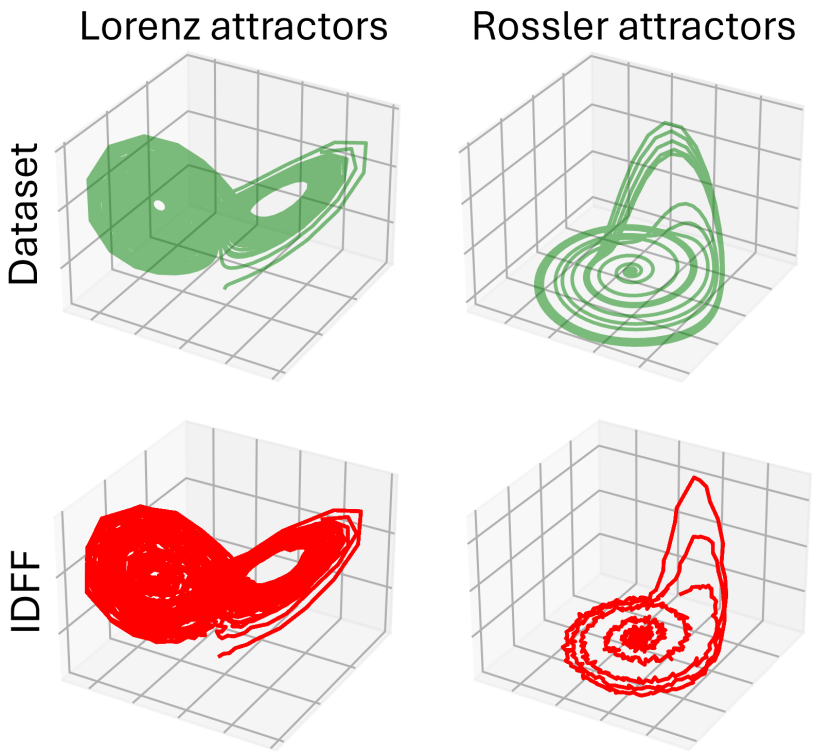

Figure 11: Time-series simulation. IDFF trajectory generation for the chaotic systems.

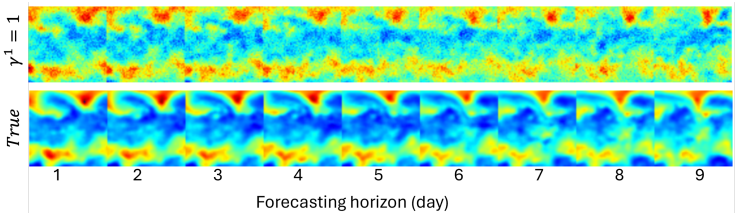

Figure 12: SST forecasting result conditioned on day 1st for 9 days with $\gamma^1 = 1$ and fixed $NFE = 5$. Same results for different $\gamma_t^1 = \sigma_t^2 \gamma^1$s is shown in Figure13

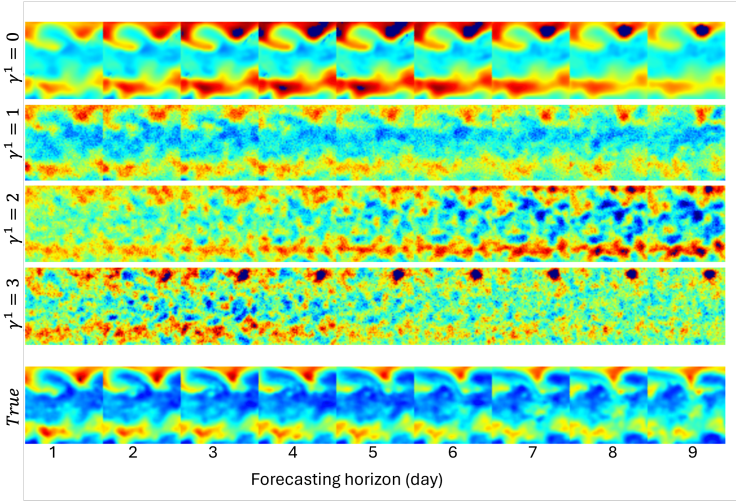

Figure 13: SST forecasting result conditioned on day 1st for 9 days for different values of $\gamma_t^1 = \sigma_t^2 \gamma^1$ and fixed $NFE = 5$.

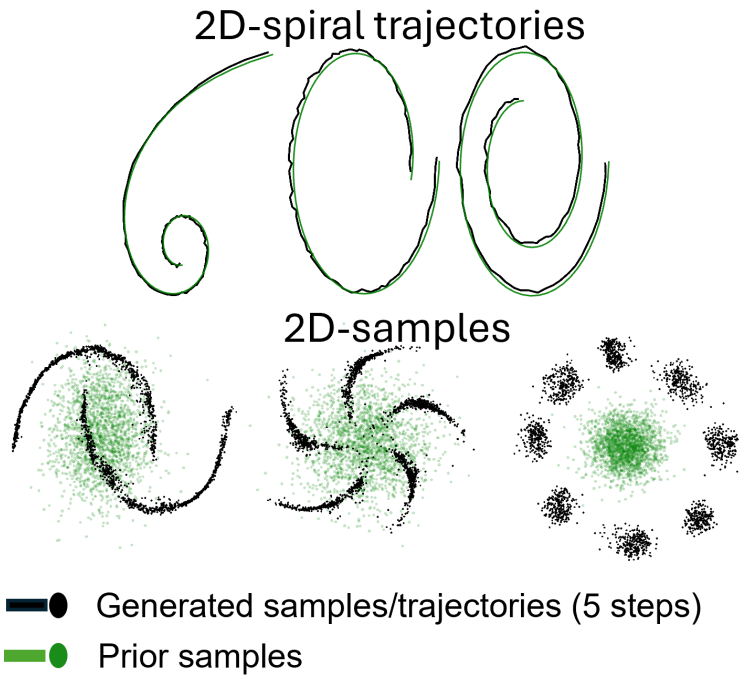

Figure 14: 2D synthetic simulation.