# Efficient Flow Matching using Latent Variables

**Anirban Samaddar***
Argonne National Laboratory
Lemont, Illinois, USA

**Yixuan Sun**
Argonne National Laboratory
Lemont, Illinois, USA

**Viktor Nilsson**
KTH Royal Institute of Technology
Stockholm, Sweden

**Sandeep Madireddy***
Argonne National Laboratory
Lemont, Illinois, USA

## Abstract

Flow matching models have shown great potential in image generation tasks among probabilistic generative models. However, most flow matching models in the literature do not explicitly model the underlying structure/manifold in the target data when learning the flow from a simple source distribution like the standard Gaussian. This leads to inefficient learning, especially for many high-dimensional real-world datasets, which often reside in a low-dimensional manifold. Existing strategies of incorporating manifolds, including data with underlying multi-modal distribution, often require expensive training and hence frequently lead to suboptimal performance. To this end, we present `Latent-CFM`, which provides simplified training/inference strategies to incorporate multi-modal data structures using pretrained deep latent variable models. Through experiments on multi-modal synthetic data and widely used image benchmark datasets, we show that `Latent-CFM` exhibits improved generation quality with significantly less training (up to $\sim 50\%$ less) and computation than state-of-the-art flow matching models by incorporating extracted data features using pretrained lightweight latent variable models. Moving beyond natural images to generating fields arising from processes governed by physics, using a 2d Darcy flow dataset, we demonstrate that our approach generates more physically accurate samples than competitive approaches. In addition, through latent space analysis, we demonstrate that our approach can be used for conditional image generation conditioned on latent features, which adds interpretability to the generation process.

## 1 Introduction

Flow Matching (FM) is a generative modeling framework that directly learns a vector field that smoothly transports a simple source distribution to a target data distribution Lipman et al. [2023]. Compared to widely adopted diffusion generative models Song et al. [2020], Ho et al. [2020], FM provides a framework for building deterministic transport paths for mapping a simple noise distribution into a data distribution. In fact, the FM framework can also map between two arbitrary distributions Liu et al. [2022], Albergo et al. [2023]. The FM transport paths may be constructed to promote beneficial properties such as shortness and straightness of paths, providing significant computational benefits Tong et al. [2024]. FM has been shown to be applicable beyond Euclidean spaces, and extensible to incorporate optimal transport principles and novel conditioning mechanisms to improve sample quality and expressivity Tong et al. [2020], Albergo et al. [2023], Tong et al. [2024]. These developments have facilitated applications in various domains, including foundation models for video generation Polyak et al. [2025], molecular modeling Hassan et al. [2024], and discrete data generation Gat et al. [2024].

Despite these strengths, a limitation of current FM models stems from the fact that the prior knowledge or the data manifold/multi-modality in the target data are not explicitly included in the modeling, which can lead to inefficiency on the FM learning and convergence. One component of FM that could incorporate such information is the source distribution. Typically an isotropic Gaussian

*Correspondence: asamaddar@anl.gov, smadireddy@anl.gov

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop on Structured Probabilistic Inference & Generative Modeling.

Lipman et al. [2023], Tong et al. [2024] is used, which implies longer and inefficient probability flow trajectories compared to when the source distribution is closer to the target distribution structure, especially when the target data distribution is multi-modal. Only recently, FlowLLM Sriram et al. [2024] modified the source distribution as the Large Language Model generated response that FM subsequently refines to learn a transport to the target distribution. However, a data-driven source for high-dimensional image datasets is challenging to learn since transporting a custom source distribution to the target requires specialized loss functions and sampling processes Daras et al. [2022], Wang et al. [2023]. On the other hand, very few works have explored ways to incorporate the underlying manifold structure of the data Jia et al. [2024], Guo and Schwing [2025] during training. These works condition the flow from source to target distribution using latent variables. However, these works often lead to suboptimal performance for multi-modal datasets with a moderately high number of modes while requiring customized training strategies that are dataset-dependent and expensive, thus restricting their broader applicability.

To address these limitations, we adopt modern advancements in deep latent variable modeling Kingma and Welling [2022], Vahdat and Kautz [2020] into FM, and propose a simple and efficient training and inference framework to incorporate data structure in the generation process. Our contributions are as follows –

- We propose `Latent-CFM`, a FM training framework that efficiently incorporates and finetunes lightweight deep latent-variable models, enabling conditional FM networks to capture and leverage multi-modal data structures efficiently.
- We demonstrate the effectiveness of the proposed framework in significantly improving generation quality and training efficiency (up to $50\%$ fewer steps) compared to popular flow matching approaches in synthetic multi-modal and popular image benchmark datasets such as MNIST and CIFAR10.
- We show the superiority of `Latent-CFM` in generating physically consistent data with experiments on the 2d Darcy flow dataset, where consistency is measured through the residual of the governing partial differential equation, in contrast to the natural image datasets.
- We explore the natural connection of our method to the conditional generation to show that `Latent-CFM` can generate various images conditioned on the training data features. This allows generation-based high-level information from compressed latent representations, allowing regeneration of images with similar characteristics; see Fig. 6.

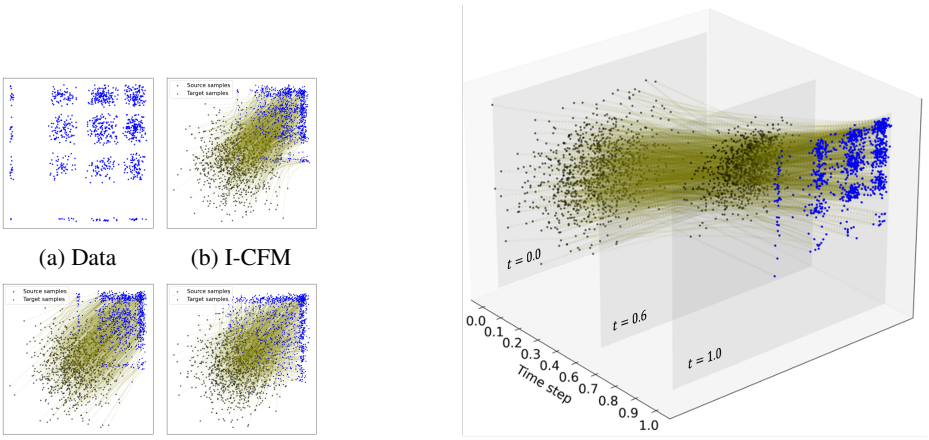
## 2 Related works

Flow-based generative models, like diffusion and flow matching, evolve samples from a simple source distribution to a complex target distribution Song et al. [2020], Lipman et al. [2023], Tong et al. [2024]. The most common choice for source distribution is Gaussian white noise. This choice often leads to longer transport paths, resulting in inefficiency during the sampling process Tong et al. [2024]. Few studies Daras et al. [2022], Wang et al. [2023] have attempted to solve this problem for diffusion models by directly modifying the source to have properties of the data distribution by injecting custom noise, e.g., blurring or masking. These works presents specialized training and sampling procedures needed to remove the effects of these noises. In flow matching, Kollovieh et al. [2024] attempts to model the source distribution as a Gaussian process (GP) to model the generation of time series data. In Sriram et al. [2024], the authors use a fine-tuned LLM to generate the source (noisy) samples to be transported by a discrete flow matching model for material discovery.

Another direction of improving efficiency of the flow-based models is by constructing specialized flows Tong et al. [2020, 2024], De Bortoli et al. [2021]. These works focus on solving for the optimal transport path from the source to the target data distribution, which "aligns" the random draws from the source distribution to the target samples during training to ensure straighter and more efficient flows. Popular approaches in this direction include optimal transport conditional flow matching (OT-CFM) Tong et al. [2024] and Schrödinger bridge conditional flow matching (SB-CFM).

Recent studies Guo and Schwing [2025], Jia et al. [2024] in incorporating data structures in diffusion and flow matching models have focused on a latent variable modeling approach. In diffusion models, Jia et al. [2024] proposed learning a Gaussian mixture model from data and conditioning the denoiser neural network with the learned cluster centers during training. The method shows promising results for 1-dimensional synthetic datasets but suboptimal results for high-dimensional datasets. In flow matching, Guo and Schwing [2025] proposes to adapt deep latent variable models Kingma and Welling [2022] to cluster the transport paths. The authors show promising results on high-dimensional datasets; however, the method demands expensive training. In Fig. 1, we show that the popular CFM models fail to capture the multi-modal data structure of the 2d triangle dataset with 16 modes (details in appendix E).

In this study, we present `Latent-CFM`, a framework for incorporating target data structures into the training/inference process of conditional flow matching models. Our approach enables adapting the




(a) Data (b) I-CFM (c) OT-CFM (d) VRFM (e) `Latent-CFM` (VAE)

Figure 1: (a)-(d) showing generation quality of different flow-based generative models on 2d triangle dataset with 16 modes. Popular conditional flow matching approaches generated samples fail to capture the multi-modal structure of the target data distribution. (e) `Latent-CFM` generates samples similar to the data samples, capturing the multi-modal data structure.

popular deep latent variable models Kingma and Welling [2022] for efficient training and high-quality sample generation. In addition, our approach can generate samples conditioned on data features, adding interpretability to the generated samples, which is uncommon for the standard flow matching approaches. Fig. 1e shows that our approach can generate samples capturing the multi-modal structure of the data.

# 3 `Latent-CFM`: Conditional Flow Matching with Latent variables

This section describes our proposed method `Latent-CFM`. First, we describe the notations that will be followed throughout the manuscript.

## 3.1 Background and Notations

We denote the unknown density of the data distribution over $\mathbb{R}^d$ by $p_1(x)$ and the source density, which is known and easy to sample from, by $p_0(x)$. Generative modeling involves finding a map from the simple source density $p_0(x)$ to the complex data distribution $p_1(x)$. We denote $x_0$, and $x_1$ as the random variables following the distributions $p_0(x)$ and $p_1(x)$ respectively.

**Probability flow ODE:** A time-dependent vector field $u_t : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$ defines an ordinary differential equation,

$$\frac{d\phi_t(x)}{dt} = u_t(\phi_t(x)); \phi_0(x) = x_0 \tag{1}$$

where $\phi_t(x)$ is the solution of the ODE or *flow* with the initial condition in Eq 1, and $u_t(.)$ (interchangeable with $u(., t)$) is the ground-truth vector field that transports the samples from the source to the target distribution. Given a source distribution $p_0(x)$ one can learn an invertible flow $\phi_1(x)$ using maximum likelihood using the change of variable formula Lipman et al. [2023]. This is the main motivation for learning normalizing flows (NF) where the source distribution is commonly assumed to be a standard Gaussian distribution.

**Flow Matching:** Learning an invertible flow is restrictive in practice and thus serves as the motivation for flow matching. Flow matching involves learning the vector field $u_t(.)$ using a neural network $v_\theta(., t)$ by optimizing the loss:

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{t, p_t(x)} \left[ ||v_\theta(x, t) - u_t(x)||_2^2 \right] \tag{2}$$

Given the marginal probability path $p_t(x) = N(x|\mu_t, \sigma_t^2 I)$, the ODE flow that generates the path is not unique. However, a simple choice of the flow is $\phi_t(\epsilon) = \mu_t + \sigma_t \epsilon; \epsilon \sim N(0, I)$. [Lipman et al., 2023, Theorem 3] and [Tong et al., 2024, Theorem 2.1] show that the unique vector field that generates the flow has the following form:

$$u_t(x) = \frac{\sigma_t'}{\sigma_t}(x - \mu_t) + \mu_t' \tag{3}$$

where, $\mu_t' = \frac{d\mu_t}{dt}; \sigma_t' = \frac{d\sigma_t}{dt}$.

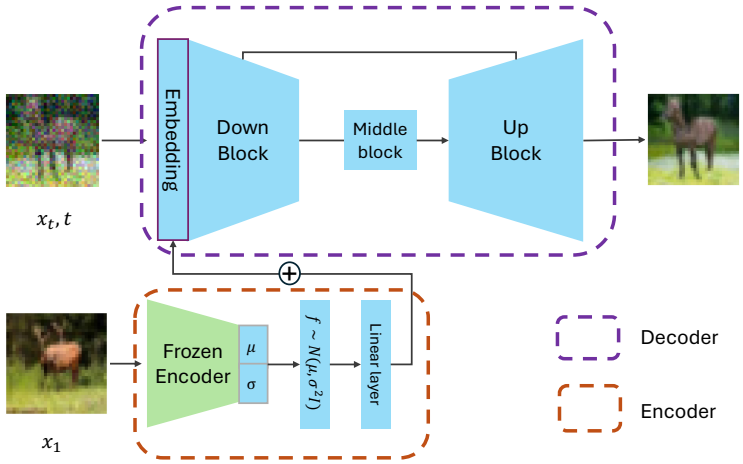



Figure 2: Schematic of `Latent-CFM` framework. Given a data $x_1$, `Latent-CFM` extracts latent features using a frozen encoder and a trainable stochastic layer. The features are embedded using a linear layer and added to the learned vector field. The framework resembles an encoder-decoder architecture like VAEs.

**Conditional Flow Matching:** The probability path $p_t$ is unknown for general source and target distributions. Lipman et al. [2023], Tong et al. [2024] proposed conditional flow matching (CFM) where the probability path $p_t(.|x_0, x_1)$ and the vector field $u_t(.|x_0, x_1)$ is learned conditioned on the end-point samples $(x_0, x_1)$ drawn from the distribution $q(x_0, x_1)$. The marginal vector field $u_t$ and the probability path $p_t$ are given by:

$$p_t(x) = \int p_t(x|x_0, x_1) q(x_0, x_1) dx_0 dx_1 \tag{4}$$

$$u_t(x) = \int u_t(x|x_0, x_1) \frac{p_t(x|x_0, x_1) q(x_0, x_1)}{p_t(x)} dx_0 dx_1 \tag{5}$$

Eq. 4 induces a mixture model on the marginal probability path $p_t(x)$ with the conditional probability paths $p_t(x|x_0, x_1)$ weighted according to the likelihood $q(x_0, x_1)$. Similarly the marginal vector field $u_t(x)$ is an weighted average of the conditional vector field $u_t(x|x_0, x_1)$ with the posterior likelihood $p_t(x_0, x_1|x) = \frac{p_t(x|x_0,x_1)q(x_0,x_1)}{p_t(x)}$ as weights.

Given we know the conditional vector field $u_t(x|x_0, x_1)$, it is not possible to derive the marginal $u_t(x)$ since the denominator of the posterior $p_t(x_0, x_1|x)$ involve the intractable marginal probability path $p_t(x)$. Therefore, Lipman et al. [2023], Tong et al. [2024] proposed conditional flow matching objective:

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{t, q(x_0,x_1), p_t(x|x_0,x_1)} \left[ ||v_\theta(x, t) - u_t(x|x_0, x_1)||_2^2 \right] \tag{6}$$

The CFM objective requires samples from $p_t(x|x_0, x_1)$, and $q(x_0, x_1)$ and the target conditional vector field $u_t(x|x_0, x_1)$. Lipman et al. [2023], Tong et al. [2020] show that under mild conditions $\nabla_\theta \mathcal{L}_{\text{CFM}}(\theta) = \nabla_\theta \mathcal{L}_{\text{FM}}(\theta)$. Therefore, the learned vector field $v_{\theta^*}(x, t)$ that minimizes Eq. 6 is also the minimizer of Eq. 4. As a consequence, we can directly solve Eq. 1 with replacing $u_t(.)$ by $v_{\theta^*}(., t)$ to transport the source distribution samples to the target distribution.

## 3.2 `Latent-CFM` framework

A key ingredient in CFM training is to specify the conditioning distribution $q(x_0, x_1)$ to sample for the CFM training. There are several choices, such as independent coupling Tong et al. [2024], optimal transport Tong et al. [2020], etc. Motivated by the deep latent variable models (LVMs) such as VAEs Kingma and Welling [2022], we propose `Latnet-CFM` that combines recent developments of LVMs with flow matching for improved generative modeling.

We model $q(x_0, x_1)$ as a mixture distribution with mixture weights given by a latent variable $f$ with density $q(f)$ defined over $\mathbb{R}^k$ where $k \leq d$,

$$q(x_0, x_1) = \int q(f) q(x_0, x_1|f) df \tag{7}$$

$q(x_0, x_1|f)$ is the likelihood of the conditioning distribution given the variables $f$. In this study, we model $p_1$ as independent of the source distribution $p_0$ where $p_1$ is assumed to be a mixture of

conditional distributions conditioned on the latent random variable $f$,

$$q(x_0, x_1|f) = p_0(x_0) \times p_1(x_1|f) \tag{8}$$

In Eq. 8, the random variable $f$ represents latent variables that can be used to model the data distribution efficiently. Eq. 7 aligns with the manifold hypothesis, which states that many real-world datasets tend to concentrate on low-dimensional manifolds. We aim to learn the latent variables from the data and augment them to aid the generative modeling.

We introduce the `Latent-CFM` loss function,

$$\mathcal{L}_{\texttt{Latent-CFM}} = \mathbb{E}_{t,q(f),q(x_0,x_1|f),p_t(x|x_0,x_1)} \, ||v_\theta(x, f, t) - u_t(x|x_0, x_1)||_2^2 \tag{9}$$

$q(x_0, x_1|f)$ is according to Eq. 8. During sampling, we can generate samples from the latent distribution $f \sim q(f)$ and the source distribution $x_0 \sim p_0(x)$ and solve Eq. 1 replacing the vector field by $v_\theta(x, f, t)$ fixing $f$. Computing Eq. 9 requires us to sample from $q(f)$, which is unobserved. Note that we can modify Eq. 9 into a more tractable objective,

$$\mathcal{L}_{\texttt{Latent-CFM}} = \mathbb{E}_{t,q(x_0,x_1),q(f|x_0,x_1),p_t(x|x_0,x_1)} \, ||v_\theta(x, f, t) - u_t(x|x_0, x_1)||_2^2 \tag{10}$$

We can apply Bayes theorem: $q(x_0, x_1)q(f|x_0, x_1) = q(f)q(x_0, x_1|f)$ to show the equivalence between Eq. 10 and 9. In Eq. 10, we can sample $(x_0, x_1) \sim q(x_0, x_1)$ and sample the posterior distribution $q(f|x_0, x_1)$ to compute the objective. Note that, the model in Eq.8 implies that $f$ is independent of the source $x_0$ and hence the posterior $q(f|x_0, x_1) = q(f|x_1)$. However, we keep the general notation $q(f|x_0, x_1)$, a more general model for the data where latent variables govern both source and target distributions. Proposition A.1 shows that if $v_\theta(x, f, t)$ has learned the minimum of $\mathcal{L}_{\texttt{Latent-CFM}}$, its flow generates the data distribution.

**Choice of** $q(.|x_1)$ The latent posterior distribution $q(.|x_1)$ should be easy to sample and capture high-level structures in the data, and can be pretrained or trained along with the CFM training. In this study, we set $q(.|x_1)$ to be the popular Variational AutoEncoders (VAE) Kingma and Welling [2022] for their success in disentangled feature extraction in high-dimensional datasets. The details about the VAEs are presented in the appendix C. In addition, we also explore Gaussian mixture models (GMM) Pichler et al. [2022] for their efficient training on small-dimensional generative modeling, which we describe in the appendix G.

In this study, we pretrain the VAEs optimizing the loss Eq. 22 on the datasets and use the pretrained encoder $q_{\hat{\lambda}}(f|x_1)$ as a feature extractor. When using the encoder in `Latent-CFM`, we finetune the final layer (parameterized by $\lambda_{final}$) that outputs $(\mu, \log(\sigma))$ and we regularize its learning with a KL-divergence term added to Eq. 10:

$$\begin{aligned}\mathcal{L}_{\texttt{Latent-CFM}} \geq \mathbb{E}_{q(x_0,x_1)} \big[ &\mathbb{E}_{t,q(f|x_0,x_1),p_t(x|x_0,x_1)} \, ||v_\theta(x, f, t) \\ &- u_t(x|x_0, x_1)||_2^2 + \beta D_{KL}(q_{\lambda_{final}}(f|x_0, x_1)||p(f))\big]\end{aligned} \tag{11}$$

where, $p(f) = N(0, I)$. Eq. 11 is an upper bound of Eq. 10 since KL-divergence is non-negative. In addition, we have empirically observed that the KL-divergence term resulted in the model learning beyond reconstruction of the data $x_1$ and increasing variability in unconditional generation. The loss in Eq. 11 is similar to VRFM loss Guo and Schwing [2025] in Eq. 20, however, in `Latent-CFM` the encoder $q_{\lambda_{final}}(.|x_0, x_1)$ only depends on the static endpoints, which enables sample generation conditioned on the data features.

Fig. 2 shows the schematic of the `Latent-CFM` model. Given data $x_1$, it passes through the frozen encoder layer to output the latent variable $z$. The latent variables are then embedded through a linear layer and added to the neural network $v_\theta(., ., z)$. The `Latent-CFM` model in Fig. 2 can be viewed as an encoder-decoder (red and purple boxes) architecture where the encoder extracts the features and the decoder reconstructs the sample conditioned on the features. However, `Latent-CFM` learns to predict the vector field conditioned on the features from the encoder, which is integrated to reconstruct the final image.

# 4 Experiments

We choose I-CFM Lipman et al. [2023], OT-CFM Tong et al. [2024], and VRFM Guo and Schwing [2025] as baselines and compare them with `Latent-CFM` on (a) unconditional data generation using synthetic 2d and high-dimensional image datasets, (b) physical data generation, and (c) analysis of the latent space. Implementation details for all experiments are presented in the appendix H.

## 4.1 Synthetic data sets

We use the 2d Triangle dataset Pichler et al. [2022], Nilsson et al. [2024] to benchmark the models' generation quality. The data distribution contains 16 modes (Fig. 1(a)) with different densities, and we generate 100K samples and divide them equally between the training and testing datasets.

All evaluated models share the same neural network architecture for the learned vector field. For VRFM, we fix the latent dimension to 2 and the encoder architecture to be similar to the vector field. For the `Latent-CFM`, we consider two variants differing in their feature extractors: (1) a pre-trained 16-component Gaussian mixture model (GMM), and (2) a continuous VAE with 2d latent space with $\beta = 0.1$. The training/inference algorithms for `Latent-CFM` with GMM are described in the appendix G. For sampling, we have used the `dopri5` solver with 100 steps to solve the ODE in Eq. 1.

Fig. 1(b)-(d) show the generation trajectories (yellow lines) of I-CFM, VRFM, and OT-CFM from the source to the target samples on the 2d triangle dataset. Fig. 1e shows a 3d trajectory plot for `Latent-CFM` (VAE) with the samples from the time steps $[0, 0.6, 1]$ highlighted. Compared to the other models, `Latent-CFM` generates samples that exhibit the multi-modality of the true data distribution. We compute the Wasserstein-2 (W2) metric between the generated and the test samples (details in the appendix I) to quantify the generation quality. Table 1 shows the summary (mean $\pm$ standard deviation) of W2 metrics of all models, where The `Latent-CFM` variants exhibit a lower W2 distance from the test samples than the competing method. The GMM variant of `Latent-CFM` achieves the lowest W2 score.

| **Method** | **W2** ($\downarrow$) |
|---|---|
| OT-CFM | $0.010 \pm 0.0031$ |
| I-CFM | $0.014 \pm 0.0066$ |
| VRFM | $0.050 \pm 0.0344$ |
| `Latent-CFM` (VAE) | $\mathbf{0.009 \pm 0.0013}$ |
| `Latent-CFM` (GMM) | $\mathbf{0.007 \pm 0.0008}$ |

Table 1: Wasserstein-2 distance between the generated samples from the models and the test samples on 2d Triangle datasets. The mean and the standard deviations are calculated across 5 random data density shapes. `Latent-CFM` shows the most similarity with the test samples.

## 4.2 Unconditional Image Generation

For images generation, we train the OT-CFM, I-CFM, and `Latent-CFM` on MNIST and CIFAR10 datasets. The details of the datasets are in the appendix E. We followed the network architectures and hyperparameters from Tong et al. [2024] to train OT-CFM and I-CFM. We did not find an open-source implementation for VRFM. Therefore, on CIFAR10, we add the FID of the top two VRFM models from Guo and Schwing [2025] Table 1. `Latent-CFM` shares the same neural network architecture for the vector field as the other models. In addition, we use open-sourced pretrained VAE on MNIST[2] and CIFAR10[3] as feature extractors as described in Alg. A.1. We fix the latent dimension to be 20 for MNIST and 256 for CIFAR10. We fix $\beta = 0.005$ for MNIST and $\beta = 0.001$ for CIFAR10 in Eq. 11. The parameter $\beta$ was selected to prevent `Latent-CFM` from only reconstructing the training set during sampling with Alg. A.2. The analysis on the impact of $\beta$ is in the appendix K. We train all models for 600K steps on CIFAR10 and 100K on MNIST. We use a fixed-step Euler solver for 100 and 1000 steps and the adaptive `dopri5` solver for sampling.

We evaluate the samples using the Fréchet inception distance (FID) Parmar et al. [2022], which quantifies the quality and diversity of the generated images. Table 2 shows the parameter count and the FID of the unconditional generation for the models on CIFAR10 and MNIST for different solvers and different numbers of integration steps. All models are evaluated at their final training step. `Latent-CFM` achieves similar (marginally higher FID) performance to the best-performing VRFM method with adaptive solver on CIFAR10. Meanwhile, `Latent-CFM` shows the lowest FID with a 100-step Euler solver and is similar to the best FID (with I-CFM) obtained with 1000 steps. On MNIST, across solvers, `Latent-CFM` exhibits the lowest FID values.

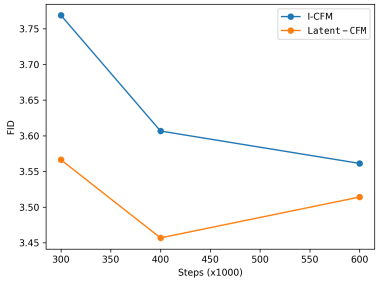



Figure 3: FID vs training steps on CIFAR10 shows that `Latent-CFM` shows better generation quality compared to I-CFM early in the training process.

Fig. 3 shows the FID vs training steps for I-CFM and `Latent-CFM` on CIFAR10. `Latent-CFM` exhibits significantly lower FID than I-CFM across training steps, demonstrating efficiency. In addition, compared to I-CFM, `Latent-CFM` achieves similar levels of FID ($\sim 3.55$) with 50% fewer training steps. Note that the best FID for `Latent-CFM` is $\sim 3.467$, which is significantly lower than the final I-CFM FID $\sim 3.561$ and is the minimum across methods and solvers (Table 2) on CIFAR10.

[2]https://github.com/csinva/gan-vae-pretrained-pytorch
[3]https://github.com/Lightning-Universe/lightning-bolts/

| Methods | CIFAR10 | | | | MNIST | | | |
|---|---|---|---|---|---|---|---|---|
| | # Params. | FID (↓) | | | # Params. | FID (↓) | | |
| | | 100 | 1000 | Adaptive | | 100 | 1000 | Adaptive |
| VRFM-1 Guo and Schwing [2025] | 37.2 M | 4.349 | 3.582 | 3.561 | - | - | - | - |
| VRFM-2 Guo and Schwing [2025] | 37.2 M | 4.484 | 3.614 | **3.478** | - | - | - | - |
| OT-FM | 35.8 M | 4.661 | 3.862 | 3.727 | 1.56 M | 15.101 | 15.880 | 16.012 |
| I-CFM | 35.8 M | 4.308 | **3.573** | 3.561 | 1.56 M | 14.272 | 14.928 | 15.050 |
| `Latent-CFM` | 36.1 M | **4.246** | 3.575 | 3.514 | 1.58 M | **13.848** | **14.543** | **14.694** |

Table 2: Image generation performance of `Latnet-CFM` compared to VRFM, I-CFM and OT-FM on CIFAR10 and MNIST. Our method exhibits improved (or similar) FID over the state-of-the-art methods using both fixed-step Euler and the adaptive `dopri5` solver for both datasets.

| | I-CFM | | `Latent-CFM` | | |
|---|---|---|---|---|---|
| **# Params.** | 34M | 65M | (34 + 16)M | (34 +.07)M | (34+1)M |
| **Residual median ↓** | 5.922 | 4.921 | **3.305** | **3.619** | **3.615** |

Table 3: Comparison of the PDE residuals of the generated Darcy flow samples ($[K, p]$ pairs) from I-CFM and `Latent-CFM` with various model sizes. `Latent-CFM` model sizes are a sum of the vector field parameters and the VAE encoder parameters. The samples generated using `Latent-CFM`, despite the smaller models, present lower PDE residuals.

## 4.3 Generation of 2D Darcy Flow

Beyond the image space, generative models have great potential in advancing scientific computing tasks. Different from images, scientific data must satisfy specific physical laws on top of visual correctness. As a result, generated samples from the unconditional models usually present non-physical artifacts due to the lack of physics-based structure imposed during learning Jacobsen et al. [2025], Cheng et al. [2024]. We use `Latent-CFM` to explore its performance of generating permeability and pressure fields ($K$ and $p$) in 2D Darcy flow and compute the residuals of the governing equations to evaluate generated samples. Details of data generation and parameter choices are in Appendix D.

We train I-CFM and `Latent-CFM` on the 10K samples of the 2d Darcy flow process. We adopted the network architecture for the vector field from our CIFAR10 experiments for this data by changing the input convolution to adapt to the Darcy flow data size, which is $(2, 64, 64)$. For this data, we don't have a pretrained VAE. Therefore, in `Latent-CFM`, we train a VAE encoder model following Rombach et al. [2022] architecture along with the CFM training. We also add the results of a `Latent-CFM` training with a pretrained VAE (Alg. A.1) in the appendix L.

Fig. 4 shows the generated samples from I-CFM and `Latent-CFM`. Both models generate visually plausible $[K, p]$ pairs. However, `Latent-CFM` resulted in samples with lower residuals, making them more physically aligned with the governing equations. Table. 3 shows the median mean-squared-residuals for the methods calculated on 500 generated samples and also their parameter counts. We observe that `Latent-CFM` with a small encoder (# parameters $34.07$M) significantly outperforms the large I-CFM model (# parameters $65$M) in terms of the median residual. The superiority of `Latent-CFM` in this case motivates us to investigate its performance for other scientific data generation tasks and examine the learned latent space in future work. Although `Latent-CFM` exhibits lower median residual, it generates more extreme outliers as shown in appendix Fig. A.1.

## 4.4 Latent space analysis

This section analyzes the latent space learned by the `Latent-CFM` models trained on the MNIST and CIFAR10 datasets. We investigate the effect of the two sources of stochasticity in the `Latent-CFM` model during inference by (1) perturbing the latent features while fixing the samples from the source distribution and (2) using Gaussian white noise as the source distribution samples while keeping the latent feature constant. All results presented for the `Latent-CFM` are from the model at the end of training. We have used an adaptive `dopri5` solver for inference.

**Latent space traversal on MNIST** Fig. 5 shows the latent space traversal for the pre-trained VAE model and the `Latent-CFM` model, which augments the feature learned by the VAE encoder according to Alg. A.1. We obtain the latent variables from data samples and generate new samples by perturbing the odd coordinates $(1, 3, ..., 19)$ of the 20-dimensional latent space within the range $[\mu - 5\sigma, \mu + 5\sigma]$, where $(\mu, \sigma)$ represent the encoded mean and variance. For each row, the new samples cor-

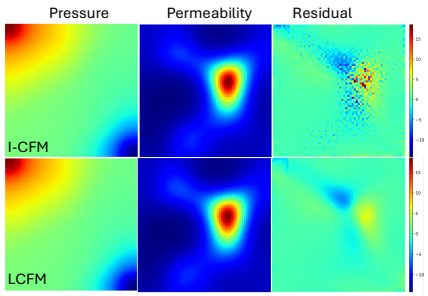



Figure 4: Plot showing a generated sample from I-CFM (top row) and `Latent-CFM` (bottom row) models trained on 10K samples generated by solving Darcy Flow equations Jacobsen et al. [2025]. Visually generated samples from both models resemble true Pressure and Permeability fields. However, `Latent-CFM` exhibits a better fit to the Darcy flow equations as measured by the residuals.

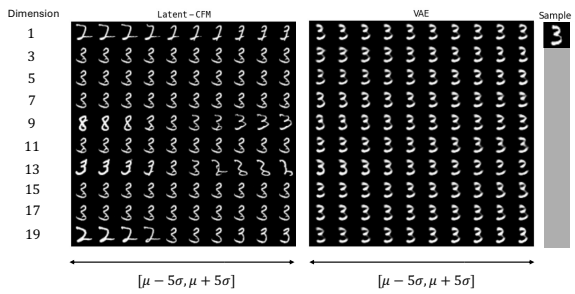



Figure 5: Traversal of the latent space $f$ shows that `Latent-CFM` generates a diverse set of samples than the pretrained VAE. All generated images for `Latent-CFM` share the same source samples $x_0 \sim N(0, I)$. The latent space traversal shows we can generate different digits with similar latent structures.

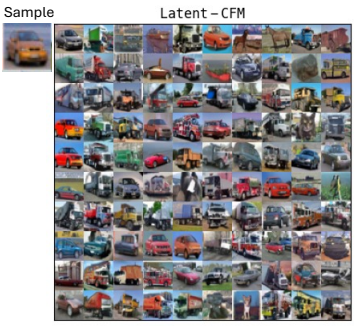



Figure 6: Conditioning the generation process of `Latent-CFM` on the features learned from the training samples shows that the framework generates samples by varying the objects while retaining properties like background, colors, object shape, etc. All samples shown have the same feature vector $f$ respectively.

respond to perturbing only one coordinate of the data sample latent. For all generated images with `Latent-CFM`, we fix the samples $x_0$ from the source distribution. We observe that `Latent-CFM` generates images with significantly more variability and better quality than the baseline VAE model. `Latent-CFM` has generated the same digit (the digit 3) but with different styles. In addition, we observe that perturbing certain latent coordinates (for example, coordinate 9,19) generates different digits with similar structure.

**Conditional generation by image features** An important property of the `Latent-CFM` is the ability to generate samples from the distribution $p_1(x|f)$ conditioned on the features $f$. Fig. 6 shows 100 generated samples conditioned on the CIFAR10 data sample of a car image (on the left). For all generations, we fix the same latent sample $f \sim q_{\hat{\lambda}}(f|x_1^{car})$ where $x_1^{car}$ denotes the selected CIFAR10 image and vary the source samples $x_0 \sim N(0, I)$. We observe that `Latent-CFM` generates different images while retaining properties like color schemes and object shape, etc. Note that `Latent-CFM` provides access to an approximate posterior likelihood $q_{\hat{\lambda}}(.|x)$, which can be used for classifier guidance Dhariwal and Nichol [2021] techniques to improve generation quality. We leave this as a future direction.

# 5 Conclusion

Flow matching models generalize the transport paths of diffusion models, thus unifying flow-based generative models. However, existing flow matching and diffusion studies often do not consider the structure of the data explicitly when constructing the flow from source to target distribution. In this study, we present `Latent-CFM`, a framework capable of incorporating additional structure/manifold of the data as latent variables in conditional flow matching. We present training/inference algorithms to adapt popular deep latent variable models into the CFM framework. Using experiments on synthetic and benchmark image datasets, we show that our approach improves (or shows similar) generation quality (FID $\sim 3.5$ on CIFAR10) compared to state-of-the-art CFM models, especially with significantly fewer training steps (with $\sim 50\%$ in CIFAR10). In addition, we demonstrate the utility of `Latent-CFM` in generating more physically consistent Darcy flow data than I-CFM. Finally, through latent space analysis, we explore the natural connection of our approach to conditional image generation conditioned on image features like background, color, etc.

One interesting direction for future research could be to tighten the upper bound in Eq. 11 with a data-driven learned prior $\hat{p}(f)$. Based on recent advances in estimating information-theoretic bounds Nilsson et al. [2024], one can train a learned prior alongside `Latent-CFM` to learn better latent representations with lower loss value. This approach can bring about a computational advantage during sampling, where we can avoid reusing the encoder and directly sample from $\hat{p}(f)$. In addition, it will be interesting to explore the application of our approach in scientific machine learning. An area of application could be in multifidelity modeling, where using `Latent-CFM`, one can inform a CFM model trained on a high-fidelity (expensive) simulation dataset (example, fluid dynamics simulations)

with latents learned from inexpensive low-fidelity simulated data. Based on our experiment with the Darcy Flow dataset, it could be a promising approach to improve generation performance by satisfying underlying physics constraints.

# 6 Acknowledgements

The computations were enabled by the computational resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357, Laboratory Computing Resource Center (LCRC) at the Argonne National Laboratory. AS, YS, and SM were supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research, through the SciDAC-RAPIDS2 institute under Contract DE-AC02-06CH11357, and additionally, SM was supported by Competitive Portfolios For Advanced Scientific Computing Research Project, Energy Efficient Computing: A Holistic Methodology, under Contract DE-AC02-06CH11357.

**References**

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.

Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. Deep variational information bottleneck, 2019. URL https://arxiv.org/abs/1612.00410.

Luigi Ambrosio, Nicola Gigli, and Giuseppe Savaré. *Gradient flows: in metric spaces and in the space of probability measures*. Springer Science & Business Media, 2008.

Chaoran Cheng, Boran Han, Danielle C. Maddix, Abdul Fatir Ansari, Andrew Stuart, Michael W. Mahoney, and Yuyang Wang. Hard Constraint Guided Flow Matching for Gradient-Free Generation of PDE Solutions, December 2024. URL http://arxiv.org/abs/2412.01786. arXiv:2412.01786 [cs].

Giannis Daras, Mauricio Delbracio, Hossein Talebi, Alexandros G Dimakis, and Peyman Milanfar. Soft diffusion: Score matching for general corruptions. *arXiv preprint arXiv:2209.05442*, 2022.

Valentin De Bortoli, James Thornton, Jeremy Heng, and Arnaud Doucet. Diffusion schrödinger bridge with applications to score-based generative modeling. *Advances in Neural Information Processing Systems*, 34:17695–17709, 2021.

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

Rick Durrett. *Probability: theory and examples*, volume 49. Cambridge university press, 2019.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching, 2024. URL https://arxiv.org/abs/2407.15595.

Pengsheng Guo and Alexander G Schwing. Variational rectified flow matching. *arXiv preprint arXiv:2502.09616*, 2025.

Majdi Hassan, Nikhil Shenoy, Jungyoon Lee, Hannes Stark, Stephan Thaler, and Dominique Beaini. Et-flow: Equivariant flow-matching for molecular conformer generation, 2024. URL https://arxiv.org/abs/2410.22388.

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

Christian Jacobsen, Yilin Zhuang, and Karthik Duraisamy. Cocogen: Physically consistent and conditioned score-based generative models for forward and inverse problems. *SIAM Journal on Scientific Computing*, 47(2):C399–C425, 2025.

Nanshan Jia, Tingyu Zhu, Haoyu Liu, and Zeyu Zheng. Structured diffusion models with mixture of gaussians as prior distribution. *arXiv preprint arXiv:2410.19149*, 2024.

Olav Kallenberg. *Foundations of modern probability, 3rd edition*. Springer, 2021.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022. URL https://arxiv.org/abs/1312.6114.

Marcel Kollovieh, Marten Lienen, David Lüdke, Leo Schwinn, and Stephan Günnemann. Flow matching with gaussian process priors for probabilistic time series forecasting, 2024. URL https://arxiv.org/abs/2410.03024.

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

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

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.

Viktor Nilsson, Anirban Samaddar, Sandeep Madireddy, and Pierre Nyquist. Remedi: Corrective transformations for improved neural entropy estimation. In *Forty-first International Conference on Machine Learning*. PMLR, 2024.

Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. On aliased resizing and surprising subtleties in gan evaluation. In *CVPR*, 2022.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. URL https://arxiv.org/abs/1912.01703.

Georg Pichler, Pierre Colombo, Malik Boudiaf, Günther Koliander, and Pablo Piantanida. A differential entropy estimator for training neural networks, 2022. URL https://arxiv.org/abs/2202.06618.

Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, David Yan, Dhruv Choudhary, Dingkang Wang, Geet Sethi, Guan Pang, Haoyu Ma, Ishan Misra, Ji Hou, Jialiang Wang, Kiran Jagadeesh, Kunpeng Li, Luxin Zhang, Mannat Singh, Mary Williamson, Matt Le, Matthew Yu, Mitesh Kumar Singh, Peizhao Zhang, Peter Vajda, Quentin Duval, Rohit Girdhar, Roshan Sumbaly, Sai Saketh Rambhatla, Sam Tsai, Samaneh Azadi, Samyak Datta, Sanyuan Chen, Sean Bell, Sharadh Ramaswamy, Shelly Sheynin, Siddharth Bhattacharya, Simran Motwani, Tao Xu, Tianhe Li, Tingbo Hou, Wei-Ning Hsu, Xi Yin, Xiaoliang Dai, Yaniv Taigman, Yaqiao Luo, Yen-Cheng Liu, Yi-Chiao Wu, Yue Zhao, Yuval Kirstain, Zecheng He, Zijian He, Albert Pumarola, Ali Thabet, Artsiom Sanakoyeu, Arun Mallya, Baishan Guo, Boris Araya, Breena Kerr, Carleigh Wood, Ce Liu, Cen Peng, Dimitry Vengertsev, Edgar Schonfeld, Elliot Blanchard, Felix Juefei-Xu, Fraylie Nord, Jeff Liang, John Hoffman, Jonas Kohler, Kaolin Fire, Karthik Sivakumar, Lawrence Chen, Licheng Yu, Luya Gao, Markos Georgopoulos, Rashel Moritz, Sara K. Sampson, Shikai Li, Simone Parmeggiani, Steve Fine, Tara Fowler, Vladan Petrovic, and Yuming Du. Movie gen: A cast of media foundation models, 2025. URL https://arxiv.org/abs/2410.13720.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022. URL https://arxiv.org/abs/2112.10752.

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*, 2020.

Anuroop Sriram, Benjamin Miller, Ricky TQ Chen, and Brandon Wood. Flowllm: Flow matching for material generation with large language models as base distributions. *Advances in Neural Information Processing Systems*, 37:46025–46046, 2024.

Alexander Tong, Jessie Huang, Guy Wolf, David Van Dijk, and Smita Krishnaswamy. Trajectorynet: A dynamic optimal transport network for modeling cellular dynamics. In *International conference on machine learning*, pages 9526–9536. PMLR, 2020.

Alexander Tong, Kilian FATRAS, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=CD9Snc73AW. Expert Certification.

Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *Advances in neural information processing systems*, 33:19667–19679, 2020.

Zhendong Wang, Yifan Jiang, Huangjie Zheng, Peihao Wang, Pengcheng He, Zhangyang Wang, Weizhu Chen, Mingyuan Zhou, et al. Patch diffusion: Faster and more data-efficient training of diffusion models. *Advances in neural information processing systems*, 36:72137–72154, 2023.

# A Theoretical analysis

In the following proposition, we adopt a slightly different formalism where we denote between the random variables with capital letters ($X_0, X_1, F$, etc.) and the values they take ($x_0, x_1, f$). For a random variable $Y$, $\mathcal{L}(Y)$ denotes its law.

**Proposition A.1.** *Let $(X_0, X_1)$ be drawn under some probability measure $\mathbb{P}$ on a measurable space $(\Omega, \mathcal{F})$, which also carries the latent random variable $F$ in $\mathbb{R}^{d_f}$. Assume that $X_1 - X_0$ is integrable and define the process $\{X_t\}_t$ by*

$$X_t = tX_0 + (1-t)X_1, \tag{12}$$

*and let $\{\mu_t^f\}_t = \{\mathcal{L}(X_t \mid F = f)\}_t$ be its ($F$-conditional) marginal probability path, under $\mathbb{P}$. Given an optimally learned $F$-conditional vector field $v^* = v^*_{t,f}(x)$, minimizing the loss function in* (11)*, and its $F$-conditional flow $\phi^{v^*,f}$, we have that its marginal probability path is equal to that of the ground truth process in* (12) *$\mathbb{P}$-a.s. in $f$, i.e. $\{\mathcal{L}(\phi_t^{v^*,f}(X_0))\}_t = \{\mu_t^f\}_t$ ($F_\#\mathbb{P}$-a.s. in $f$). In particular, $\mathcal{L}(\phi_1^{v^*,f}(X_0)) = \mathcal{L}(X_1 \mid F = f)$ ($F_\#\mathbb{P}$-a.s. in $f$) and $\phi_1^{v^*,F}(X_0) \overset{\mathcal{L}}{=} X_1$.*

*Proof.* We follow the proof ideas in Liu et al. [2022] and Guo and Schwing [2025]. We want to see that $(\mu, v^*)$ satisfies the continuity equation

$$\dot{\mu}_t^f + \nabla \cdot (\mu_t^f v^*_{t,f}) = 0. \tag{13}$$

The meaning of (13) is only formal, as $\mu_t^f$ may not even have a density. The **definition** of (13) is that for any *test function* $h : \mathbb{R}^d \to \mathbb{R}$, i.e. $h$ is smooth and compactly supported in $\mathbb{R}^d$,

$$\frac{d}{dt}\int_{\mathbb{R}^d} h \, d\mu_t^f = \int_{\mathbb{R}^d} \langle \nabla h, v^*_{t,f} \rangle \, d\mu_t^f \tag{14}$$

holds (in the sense of distributions on $(0,1)$) [Ambrosio et al., 2008, p. 169-170]. Under some regularity conditions on $v^*$ [Ambrosio et al., 2008, Proposition 8.1.8], we know that the theorem follows if we can show (14).

Here, in fact, the derivative on the left hand side of (14) exists in the classical sense, and the differentiation can be moved under the integral sign. This is seen by noting that with $f(t,\omega) \coloneqq h(tX_0(\omega) + (1-t)X_1(\omega))$, the conditions for doing so in $\frac{d}{dt}\int f(t,\omega)\,\mathbb{P}_f(d\omega)$ are fulfilled, see e.g. [Durrett, 2019, Ch. A5]. Most importantly, $\frac{d}{dt}f(t,\omega) = \langle \nabla h(tX_0 + (1-t)X_1), X_1 - X_0\rangle_{\mathbb{R}^d}(\omega) = \langle \nabla h(X_t), \dot{X}_t\rangle_{\mathbb{R}^d}(\omega)$, where $h$ and its derivatives are bounded, and $\dot{X}_t = X_1 - X_0$ is integrable and thus conditionally integrable for almost all $f$. We get, taking $\mathbb{P}_f$ to be a regular conditional probability measure for $F = f$, known to exist via the disintegration theorem [Kallenberg, 2021, Theorem 3.4],

$$\frac{d}{dt}\int_{\mathbb{R}^d} h \, d\mu_t^f = \int_{\mathbb{R}^d} \langle \nabla h(X_t), \dot{X}_t\rangle_{\mathbb{R}^d}(\omega)\mathbb{P}_f(d\omega) = \mathbb{E}^{\mathbb{P}_f}[\langle \nabla h(X_t), \dot{X}_t\rangle_{\mathbb{R}^d}] = \dots \tag{15}$$

Further, by using the tower property to condition on $X_t$,

$$\dots = \mathbb{E}^{\mathbb{P}_f}[\mathbb{E}^{\mathbb{P}_f}[\langle \nabla h(X_t), \dot{X}_t\rangle_{\mathbb{R}^d} \mid X_t]] = \mathbb{E}^{\mathbb{P}_f}[\langle \nabla h(X_t), \mathbb{E}^{\mathbb{P}_f}[\dot{X}_t \mid X_t]\rangle_{\mathbb{R}^d}] \tag{16}$$

But for almost all $f$, we have that $\mathbb{E}^{\mathbb{P}_f}[\dot{X}_t \mid X_t] = v^*_{t,f}(X_t)$, since $v^*$ is optimal for (11), whose minimizer (for a fixed encoder $q_{\lambda_{final}}$) $v^*_{t,f}(x)$ is

$$\mathbb{E}[u_t(x|X_0, X_1) \mid X_t = x, F = f] = \mathbb{E}[X_1 - X_0 \mid X_t = x, F = f] = \mathbb{E}^{\mathbb{P}_f}[\dot{X}_t \mid X_t = x]. \tag{17}$$

This in (15) and (16) gives

$$\frac{d}{dt}\int_{\mathbb{R}^d} h\, d\mu_t^f = \mathbb{E}^{\mathbb{P}_f}[\langle \nabla h(X_t), v_t^*(X_t)\rangle_{\mathbb{R}^d}] = \int_{\mathbb{R}^d} \langle \nabla h, v_{t,f}^*\rangle_{\mathbb{R}^d}\, d\mu_t^f, \tag{18}$$

which finishes the proof of the main statement. The last part of the proposition can be seen from:

$$\mathbb{P}(\phi_1^{v^*,F}(X_0) \in A) = \mathbb{E}[\mathbb{P}(\phi_1^{v^*,F}(X_0) \in A \mid F)] = \mathbb{E}[\mathbb{P}(X_1 \in A \mid F)] = \mathbb{P}(X_1 \in A). \tag{19}$$

□

# B Relation with Variational Rectified Flow

Recent work in variational rectified flow matching (VRFM) Guo and Schwing [2025] has studied the effect of a mixture model of the vector field $u_t(x)$ induced by a latent variable. In Eq.6, we observe that the CFM objective function can be viewed as a log-likelihood of a Gaussian distribution model for $u_t(x) \sim N(u_t; v_\theta(x,t), I)$. In VRFM, we model the vector field by a mixture model induced by a latent variable $z \sim p(z)$,

$$p(u_t|x_t,t) = \int p_\theta(u_t|x_t,t,z)p(z)df \tag{20}$$

In VRFM, given $z$, the conditional density $p_\theta(u_t|x_t,t,z)$ are assumed to be $N(u_t; v_\theta(x,t,z), I)$. To learn the latent variable $z$, the authors use a recognition model $q_\phi(z|x_0,x_1,x_t,t)$ or encoder. The parameters of the encoder and the learned vector field are learned jointly by optimizing a VAE objective,

$$\begin{aligned}\log p(u_t|x_t,t) \geq\ & \mathbb{E}_{z\sim q_\phi}\left[\log p_\theta(u_t|x_t,t,z)\right] \\ & - D_{\mathrm{KL}}(q_\phi(z|x_0,x_1,x_t,t)||q(z))\end{aligned} \tag{21}$$

A key observation in the VRFM loss in Eq 21 is that the encoder model $q_\phi$ depends on $(x_0,x_1,x_t,t)$ which dynamically changes with time $t$. However, the generative model $q(z)$ is static and equal to $N(0,I)$. To generate samples from VRFM, we sample $f \sim N(0,I)$ once and solve Eq. 1 using the learned vector field $v_\theta(x,t,z)$ fixing $z$.

In `Latent-CFM`, we propose learning a static encoder model $q_\phi$ from the data $x_1$ and optimize the Eq 21 w.r.t the encoder and the vector field parameters. This change has the following advantages, (1) We can generate samples conditioned on the variables learned from the data distribution $p_1(x)$, and (2) We can use pre-trained feature extractors and fine-tune them to optimize the CFM loss.

# C Variational AutoEncoders

VAE is a popular deep latent variable model that assumes a latent variable $f$ is governing the data distribution $p_1(x_1) = \int p(f)p(x_1|f)df$. The posterior distribution $p(f|x)$ is intractable and hence is approximated by a variational distribution $q_\lambda(f|x)$ which is then learned by optimizing an ELBO:

$$\mathcal{L}_{VAE} = -\,\mathbb{E}_{p_{data}(x_1)}\left[\,\mathbb{E}_{q_\lambda(f|x_1)}\log p_\psi(x_1|f) + D_{KL}(q_\lambda(f|x_1)||p(f))\right] \tag{22}$$

where, $q_\lambda(f|x_1)$ and $p_\psi(x_1|f)$ are parameterized by an encoder and a decoder neural network respectively and $p(f)$ is assumed to be $N(0,I)$. The variational distribution is commonly assumed to be multivariate Gaussian with mean $\mu$ and a diagonal covariance matrix $\sigma^2 I$. The encoder network $q_\lambda(f|x_1)$ outputs the parameters $\mu$ and $\sigma^2$. VAEs are successful in generative modeling applications in a variety of domains. However, they often suffer from low generation quality in high-dimensional problems.

# D Darcy Flow Dataset

The Darcy flow equation describes the fluid flowing through porous media. With a given permeability field $K(x)$ and a source function $f_s(x)$, the pressure $p(x)$ and velocity $u(x)$ of the fluid, according to Darcy's law, are governed by the following equations

$$\begin{aligned} u(x) &= -K(x)\nabla p(x), \quad x \in \Omega \\ \nabla \cdot u(x) &= f_s(x), \quad x \in \Omega \\ u(x)\cdot n(x) &= 0, \quad x \in \partial\Omega \\ \int_\Omega p(x)dx &= 0, \end{aligned} \tag{23}$$

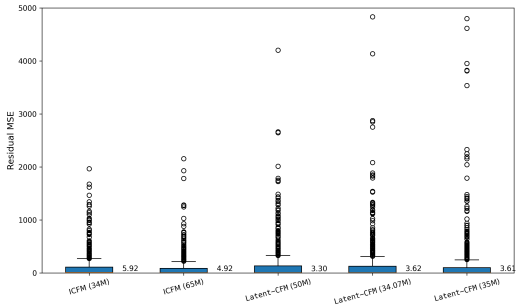



Figure A.1: Distribution of the residual MSE of I-CFM and `Latent-CFM` shows that the latter produces the overall lower median residual MSE (numbers beside each box plot). However, `Latent-CFM` seems to generate more extreme outlier residuals than I-CFM.

where $\Omega$ denotes the problem domain and $n(x)$ is the outward unit vector normal to the boundary. Following the problem set up in Jacobsen et al. [2025], we set the source term as

$$f_s(x) = \begin{cases} r, & \mid x_i - 0.5w \mid \leq 0.5w,\ i = 1, 2 \\ -r, & \mid x_i - 1 + 0.5w \mid \leq 0.2w,\ i = 1, 2 \\ 0, & \text{otherwise} \end{cases}, \tag{24}$$

and sample $K(x)$ from a Gaussian random field, $K(x) = \exp(G(x))$, $G(\cdot) \sim \mathcal{N}(\mu, k(\cdot, \cdot))$, where the covariance function is $k(x, x') = \exp(-\frac{\|x-x'\|_2}{l})$.

Using the finite difference solver, we created a dataset containing 10,000 pairs of $[K, p]$ and trained I-CFM and `Latent-CFM` models to generate new pairs. The generated samples are expected to follow (26). Therefore, we use the residual of the governing equation to evaluate a sample quality. In particular, for each generated sample, we compute

$$\begin{aligned} R(x) &= f_s(x) + \nabla \cdot [K(x)\nabla p(x)] \\ &= f_s(x) + K(x)\frac{\partial^2 p(x)}{\partial x_1^2} + \frac{\partial K(x)}{\partial x_1}\frac{\partial p(x)}{\partial x_1} \\ &\quad + K(x)\frac{\partial^2 p(x)}{\partial x_2^2} + \frac{\partial K(x)}{\partial x_2}\frac{\partial p(x)}{\partial x_2}. \end{aligned} \tag{25}$$

The partial derivatives in (30) are approximated using central finite differences.

We generated 500 $[K, p]$ pairs from trained I-CFM and `Latent-CFM` models and computed the residual according to (30) for evaluation. Figure 4 shows sample examples of Darcy flow from the two models, and Table 3 presents the comparison of the median of sample residuals between the two models.

# E Datasets

We describe the details of the datasets used in this study.

## E.1 Triangle dataset

The triangular dataset shares the same structural design as in Pichler et al. [2022], Nilsson et al. [2024]. For any dimension $d > 1$, it is constructed as the $d$-fold product of a multimodal distribution with $k$ modes (as illustrated in Fig. 2 of Pichler et al. [2022] for $k = 10$), resulting in a distribution with $k^d$ modes. For the experiments in the main paper, we set $k = 4$ and $d = 2$, creating 16 modes over the 2d plane.

## E.2 MNIST and CIFAR10

We download and use MNIST LeCun et al. [1998] and CIFAR10 Krizhevsky et al. [2009] datasets using the classes `torchvision.datasets.MNIST`, and `torchvision.datasets.CIFAR10` from the PyTorch library Paszke et al. [2019] respectively. On MNIST, we normalize the data using the mean $[0.5, 0.5]$ and the standard deviation $[0.5, 0.5]$. On CIFAR10, we use random horizontal flipping of the data and normalize using the mean $[0.5, 0.5, 0.5]$, and the standard deviation $[0.5, 0.5, 0.5]$.

## E.3 Darcy Flow data

The Darcy flow equation describes the fluid flowing through porous media. With a given permeability field $K(x)$ and a source function $f_s(x)$, the pressure $p(x)$ and velocity $u(x)$ of the fluid,

according to Darcy's law, are governed by the following equations

$$\begin{aligned} u(x) &= -K(x)\nabla p(x), \quad x \in \Omega \\ \nabla \cdot u(x) &= f_s(x), \quad x \in \Omega \\ u(x) \cdot n(x) &= 0, \quad x \in \partial\Omega \\ \int_\Omega p(x)dx &= 0, \end{aligned} \tag{26}$$

where $\Omega$ denotes the problem domain and $n(x)$ is the outward unit vector normal to the boundary. Following the problem set up in Jacobsen et al. [2025][4], we set the source term as

$$f_s(x) = \begin{cases} r, & \mid x_i - 0.5w \mid \leq 0.5w,\ i = 1,2 \\ -r, & \mid x_i - 1 + 0.5w \mid \leq 0.2w,\ i = 1,2 \\ 0, & \text{otherwise} \end{cases}, \tag{27}$$

and sample $K(x)$ from a Gaussian random field, $K(x) = \exp(G(x))$, $G(\cdot) \sim \mathcal{N}(\mu, k(\cdot,\cdot))$, where the covariance function is $k(x,x') = \exp(-\frac{\|x-x'\|_2}{l})$ and $G(x) = \mu + \sum_{i=1}^{s} \sqrt{\lambda_i}\theta_i\phi_i(x)$, where where $\lambda_i$ and $\phi_i(x)$ are eigenvalues and eigenfunctions of the covariance function sorted by decreasing $\lambda_i$, and $\theta_i \sim \mathcal{N}(0, \mathrm{I})$.

We sample permeability fields and solve for the pressure fields which results in 10,000 $[K, p]$ pairs with $r = 10$, $w = 0.125$, and $s = 16$ on $64 \times 64$ grids for model training. During training, we standardize both the permeability and pressure fields using $\mu_K = 1.1491$, $\sigma_K = 7.8154$, and $\mu_p = 0.0$, $\sigma_p = 0.0823$.

# F `Latent-CFM` algorithms

The training algorithm for `Latent-CFM` is described in Alg. A.1. In this study, following I-CFM Tong et al. [2024], we adopt the conditional probability flow $p_t(x|x_0, x_1) = N(x|tx_1 + (1-t)x_0, \sigma^2 I)$, and the vector field $u_t(x|x_0, x_1) = x_1 - x_0$. We propose to pretrain a VAE model before the CFM training loop using the same training set. However, one can run a training loop where the VAE encoder and the vector field parameters are updated jointly at each step. We omit this training algorithm since it is similar to the VRFM Guo and Schwing [2025] method with the key difference in choosing a static encoder $q_\lambda(.|x_0, x_1)$ whose parameters are trained along with the vector field. In our experiments, separating the two steps has produced better results. We have observed that directly using the latents, without finetuning, from the pretrained VAE leads to suboptimal results. Finetuning the last layer also enables regularizing the information learned in the latent space through the KL term in 11.

**Algorithm A.1** `Latent-CFM` training

Given $n$ sample $(x_1^1, ..., x_n^1)$ from $p_1(x)$, regularizer $\beta$;
**if** no pretrained VAE available **then**
  Train `VAE` using $(x_1^1, ..., x_n^1)$ optimizing Eq. 22
  Save the encoder $q_{\hat{\lambda}}(.|x_1)$
**end if**
Initialize $v_\theta(\cdot,\cdot,\cdot)$ and last encoder layer parameters $\lambda_{last}$
**for** $k$ steps **do**
  Sample latent variables $f_i \sim q_{\lambda_{last}}(f|x_i^1)$ for all $i = 1, ..., n$
  Sample $(x_1^0, ..., x_n^0)$ from $\mathcal{N}(0, I)$ and noise levels $(t_1, ..., t_n)$ from $Unif(0, 1)$ and compute $(u_{t_1}(.|x_0, x_1), ..., u_{t_n}(.|x_0, x_1))$
  compute $v_\theta(x_i^{t_i}, f_i, t_i)$ where $x_i^{t_i}$ is the corrupted $i$-th data at noise level $t_i$
  Compute $\nabla\mathcal{L}_{\texttt{Latent-CFM}}$ and update $\theta, \lambda_{last}$
**end for**
**return** $v_\theta(\cdot,\cdot,\cdot), q_{\hat{\lambda}}(.|x)$

Alg. A.2 describes the inference procedure. During inference, we need to draw samples from the estimated marginal distribution of the variables $\hat{p}(f) = \int p_1(x) q_{\hat{\lambda}}(f|x)dx$. For a moderately high-dimensional latent space, sampling from the marginal is difficult. Therefore, we reuse the empirical training samples $(x_1^{train}, ..., x_K^{train})$, for a given sample size $K$, to draw samples $f_i \sim q_{\hat{\lambda}}(f|x_i^{train})$ for all $i = 1, ..., K$.

[4]We use the same data generation code available at `https://github.com/christian-jacobsen/CoCoGen/blob/master/data_generation/darcy_flow/generate_darcy.py`

**Algorithm A.2** `Latent-CFM` inference

Given sample size $K$, trained $v_{\hat{\theta}}(.,.,.)$ and $q_{\hat{\lambda}}(.|x_1)$, number of ODE steps $n_{ode}$
Select $K$ training samples $(x_1^{train}, ..., x_K^{train})$
Sample latent variables $f_i \sim q_{\hat{\lambda}}(f|x_i^{train})$ for all $i = 1, ..., K$
Sample $(x_1^0, ..., x_n^0)$ from $\mathcal{N}(0, I)$
$h \leftarrow \frac{1}{n_{ode}}$
**for** $t = 0, h, ..., 1 - h$ and $i = 1, ..., K$ **do**
    $x_i^{t+h} = \text{ODEstep}(v_{\hat{\theta}}(x_i^t, f_i, t), x_i^t)$
**end for**
**return** Samples $(x_1^1, ..., x_K^1)$

# G `Latent-CFM` with Gaussian Mixture Models

Gaussian mixture models (GMM) use a mixture of Gaussian kernels to model the data distribution. An $M$ component GMM is defined as,

$$q_\lambda(x) = \sum_{j=1}^{M} w_j N(x; \mu_j, \Sigma_j), \quad \sum_{j=1}^{M} w_j = 1 \tag{28}$$

where, $\lambda = \{\mu_j, \Sigma_j, w_j : j = 1, ..., M\}$ are the GMM parameters. GMMs are popular in density estimation tasks and are consistent estimators of the entropy of a probability distribution under certain assumptions (Theorem 1 in Pichler et al. [2022]). Since the mixture components are Gaussian, one can easily sample from a fitted GMM. However, estimation of GMMs requires a very large number of training samples for moderately large dimensional data and is prone to overfitting (Fig.1 in Nilsson et al. [2024]).

We explore GMM as an alternative to the VAE as a feature extractor in `Latent-CFM`. We follow Pichler et al. [2022] to train the GMMs using the cross-entropy loss function,

$$\mathcal{L}_{GMM} = -\,\mathbb{E}_{p_1(x)} \log q_\lambda(x) \tag{29}$$

Alg. A.3 describes the `Latent-CFM` training using GMMs. First, we pretrain a GMM by optimizing Eq. 29. Following Jia et al. [2024], during the CFM training, we assign each sample $x_i^1$ a cluster membership id $c_i$ based on the mixture component, which shows the maximum likelihood calculated for the data sample. These ids are passed to the learned vector field $v_\theta(.,.,.)$ as the conditioning variables. The rest of the training is similar to Alg. 1 of the main paper. Alg. A.3 does not involve a finetuning of the encoder during the CFM training, therefore, we drop the KL term in the `Latent-CFM` loss and optimize Eq. 10 of the main paper.

**Algorithm A.3** `Latent-CFM` w GMM training

Given $n$ sample $(x_1^1, ..., x_n^1)$ from $p_1(x)$;
**if** no pretrained GMM available **then**
    Train GMM using $(x_1^1, ..., x_n^1)$ optimizing Eq. 29
    Save the GMM $q_{\hat{\lambda}}(.)$
**end if**
Initialize $v_\theta(\cdot, \cdot, \cdot)$
**for** $k$ steps **do**
    Calculate the cluster memberships $c_i = \underset{j=1,..,M}{argmax}(N(x_i^1; \hat{\mu}_j, \hat{\Sigma}_j)), i = 1, ..., n$
    Sample $(x_1^0, ..., x_n^0)$ from $\mathcal{N}(0, I)$ and noise levels $(t_1, ..., t_n)$ from $Unif(0, 1)$ and compute $(u_{t_1}(.|x_0, x_1), ..., u_{t_n}(.|x_0, x_1))$
    compute $v_\theta(x_i^{t_i}, c_i, t_i)$ where $x_i^{t_i}$ is the corrupted $i$-th data at noise level $t_i$
    Compute $\nabla \mathcal{L}_{\texttt{Latent-CFM}}$ in Eq. 10 (main paper) and update $\theta$
**end for**
**return** $v_\theta(\cdot, \cdot, \cdot), q_{\hat{\lambda}}(.)$

Alg. A.4 describes the inference steps using the `Latent-CFM` with GMM. Given a budget of $K$ samples, we draw the cluster membership ids $(c_1, ..., c_K)$ from the distribution $Categorical(w_1, ..., w_K)$. This step helps maintain the relative proportion of the clusters in the generated sample set according to the estimated GMM. The rest of the inference steps are similar to Alg. 2 from the main paper.

# H Implementation details

We provide implementation details of `Latent-CFM` and other methods used in the experiments section. The full codebase is available at `https://anonymous.4open.science/r/Latent_`

**Algorithm A.4** `Latent-CFM` w GMM inference

1: Given sample size $K$, trained $v_{\hat{\theta}}(.,.,.)$ and $q_{\hat{\lambda}}(.)$, number of ODE steps $n_{ode}$
2: Select $K$ random cluster memberships $(c_1, ..., c_K)$ from the distribution $Categorical(\hat{w}_1, ..., \hat{w}_K)$
3: Sample $(x_1^0, ..., x_n^0)$ from $\mathcal{N}(0, I)$
4: $h \leftarrow \frac{1}{n_{ode}}$
5: **for** $t = 0, h, ..., 1 - h$ and $i = 1, ..., K$ **do**
6: $\quad x_i^{t+h} = \text{ODEstep}(v_{\hat{\theta}}(x_i^t, c_i, t), x_i^t)$
7: **end for**
8: **return** Samples $(x_1^1, ..., x_K^1)$

`CFM-66CF/README.md`. We closely follow the implementation in the Tong et al. [2024] repository[5].

**Synthetic data** All CFM training on the 2d triangle dataset has the same neural network architecture for the learned vector field. The architecture is a multi-layered perceptron (MLP) with three hidden layers with `SELU` activations. For I-CFM and OT-CFM, the input to the network is $(x_t, t)$, and it outputs the learned vector field value with the same dimension as $x_t$.

`Latent-CFM` and VRFM training have an additional VAE encoder that learns the latent representations during training. For VRFM, we consider the same MLP architecture used for the vector field as the VAE encoder, with the final layer outputs 2d mean and 2d log-variance vectors. The input to the encoder involves the tuple $(x_0, x_1, x_t, t)$ as recommended by Guo and Schwing [2025], which outputs the latent variable $z_t$ and is added to the input tuple $(x_t, z_t, t)$ and passed to the vector field network. The pretrained VAE in `Latent-CFM` has the same encoder and a one-hidden-layer MLP with `SELU` activation as a decoder. In the CFM training in `Latent-CFM`, we fix the pretrained VAE encoder and add a trainable layer, which predicts the mean and the log-variance. The VAE encoder in `Latent-CFM` takes the input $x_1$ and outputs the latent variable $f$, which is then added to the input tuple $(x_t, f, t)$ in the CFM training. For both VRFM and `Latent-CFM` we fix the KL regularization parameter $\beta = 0.01$.

In `Latent-CFM` with GMM, we consider a diagonal covariance matrix $\Sigma_j = \sigma_j^2 I$ for each Gaussian component and set the number of components $K = 16$. The CFM architecture is the same as above.

**MNIST and CIFAR10** All models used the same U-Net architecture from Tong et al. [2024] on MNIST and CIFAR10. The hyperparameters of the learned vector field are changed depending on the dataset. For I-CFM and OT-CFM, the model takes the input $(x_t, t)$ where both variables are projected onto an embedding space and concatenated along the channel dimension and passed through the U-Net layers to output the learned vector field.

In `Latent-CFM`, we use the pretrained VAE model available for MNIST[6], and CIFAR10[7]. We take the latent encodings from the last encoder layer and add a trainable MLP layer to output the mean and log-variance of the latent space. Using the reparameterization trick Kingma and Welling [2022], we sample the latent variable and project it to the embedding space of the CFM model using a single trainable MLP layer. These feature embeddings are added (see Fig. 2 of the main paper) to the time embeddings and passed to the U-Net.

**Darcy Flow** We follow the same model architecture used for CIFAR10 for the I-CFM training in the Darcy Flow dataset. On this dataset, for `Latent-CFM`, we train a VAE encoder following Rombach et al. [2022] along with the CFM training. The VAE encoder has 3 downsampling layers, bringing down the spatial dimension from $[64, 64]$ to $[8, 8]$ in the latent space. The channel dimension was successively increased from 2 to 128 using the sequence $[16, 32, 64, 128]$ and then reduced to 8 in the final encoder layer. The final latent encodings are flattened and added to the vector field, similar to the CIFAR10 training. We fix the KL regularization parameter $\beta = 0.001$.

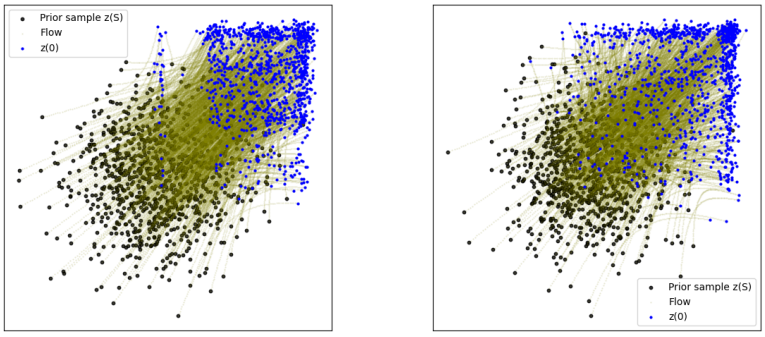
Additional hyperparameter details are presented in Table A.1. In addition, following Tong et al. [2024], we set the variance of the simulated Gaussian probability path $\sigma_t = 0.01$ for MNIST and 0 for CIFAR10 and Darcy Flow dataset.

**Computational cost** All models were trained using NVIDIA A100 GPUs. On MNIST, CIFAR10, and Darcy Flow data, training `Latent-CFM` took approximately $3, 16$, and $3.5$ hours, respectively.

[5] https://github.com/atong01/conditional-flow-matching.git
[6] https://github.com/csinva/gan-vae-pretrained-pytorch
[7] https://github.com/Lightning-Universe/lightning-bolts/

| Hyperparameters | MNIST | CIFAR-10 | Darcy Flow |
|---|---|---|---|
| Train set size | 60,000 | 50,000 | 10,000 |
| # steps | 100K | 600K | 100K |
| Training batch size | 128 | 128 | 128 |
| Optimizer | Adam | Adam | Adam |
| Learning rate | 2e-4 | 2e-4 | 2e-4 |
| Latent dimension | 20 | 256 | 256 |
| number of model channels | 32 | 128 | 128 |
| number of residual blocks | 2 | 2 | 2 |
| channel multiplier | [1,2,2] | [1, 2, 2, 2] | [1, 2, 2, 2] |
| number of attention heads | 1 | 4 | 4 |
| dropout | 0 | 0.1 | 0.1 |

Table A.1: Hyperparameter settings used for the experiments on benchmark datasets.




(a) `Latent-CFM` (b) VRFM

Figure A.2: Plot shows the generated trajectories from the source to target distribution for (a) `Latent-CFM`, and (b) VRFM model. By changing the input to the encoder, `Latent-CFM` generates samples with an improved multimodal structure similar to the data than the VRFM.

# I Metrics

## I.1 Wasserstein metric

Given two batches of samples $(x, y)$, the Wasserstein distance was calculated using the `SamplesLoss` function with the sinkhorn algorithm from the `geomloss`[8] Python library.

## I.2 Residual metric on Darcy Flow data

After rescaling to the original space, we compute the spatially averaged squared residuals of the governing equation across the domain to evaluate the sample quality in the 2D Darcy flow experiment. In particular, for each generated sample, we compute

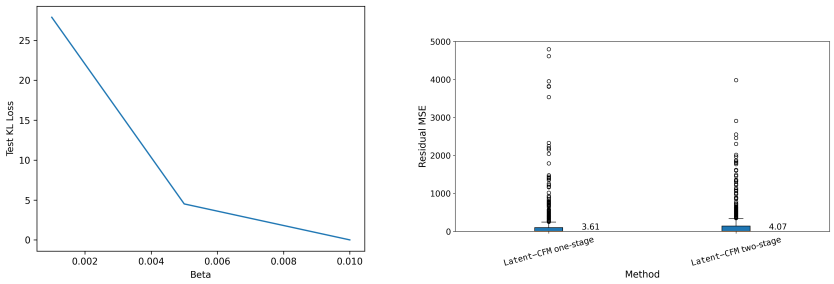
$$\begin{aligned} R(x) &= \frac{1}{N^2}\|f_s(x) + \nabla \cdot [K(x)\nabla p(x)]\|_2^2 \\ &= \frac{1}{N^2}\|f_s(x) + K(x)\frac{\partial^2 p(x)}{\partial x_1^2} + \frac{\partial K(x)}{\partial x_1}\frac{\partial p(x)}{\partial x_1} + K(x)\frac{\partial^2 p(x)}{\partial x_2^2} + \frac{\partial K(x)}{\partial x_2}\frac{\partial p(x)}{\partial x_2}\|_2^2, \end{aligned} \tag{30}$$

where $N$ is the number of discretization locations in each spatial dimension, $x_1$ and $x_2$ represent the spatial coordinates of the domain, and the partial derivatives in (30) are approximated using central finite differences.

# J Comparison with VRFM

`Latent-CFM` loss function is similar to the recently proposed variational rectified flow matching (VRFM) Guo and Schwing [2025]. We like to highlight a subtle but key difference between the two methods. VRFM training requires the encoder model to learn the latent $z_t$ from a time-varying input tuple $(x_0, x_1, x_t)$. This is required since the modeling assumption in VRFM is that the vector field $u_t$

[8]https://www.kernel-operations.io/geomloss/




(a) (b)

Figure A.3: (a) Plot of the regularization parameter $\beta$ and the KL divergence on test set shows an optimal $\beta$ around $5e-3$ on MNIST. (b)Boxplot of mean-squared residual for `Latent-CFM` with one-stage and two-stage training shows that having a pretraining stage helps to reduce the extreme residuals. However, the overall median residual is better for a one-stage `Latent-CFM` model.

is a mixture model for all $t$. `Latent-CFM` simplifies the model training by modeling the data $x_1$ as a mixture model Eq.7 in the main paper. The encoder only requires the data $x_1$ as input to extract the latent features.

Fig. A.2 shows the generated trajectories from the source to target distribution for (a) `Latent-CFM`, and (b) VRFM model trained on the 2d triangle dataset. For ease of comparison, we train the VAE encoder in `Latent-CFM` along with the CFM training, similar to VRFM. The only difference between the two models lies in the input to the encoder, which for VRFM is $(x_0, x_1, x_t, t)$, and for `Latent-CFM` is $x_1$. We observe that this change helps `Latent-CFM` to learn the multimodal structure of the data better than the VRFM. This is perhaps due to the violation of the VRM modeling assumption that the vector field random variable $u_t$ is multimodal for all $t$, which is approximately true when $t$ is close to 1.

# K Selection of $\beta$

An important hyperparameter of `Latent-CFM` training is the KL regularization parameter $\beta$. It controls the information compression of the latent space Alemi et al. [2019]. In `Latent-CFM`, too high $\beta$ results in the posterior collapsing to the prior (KL term $= 0$), and too low a value results in the model only learning to reconstruct the training data. In Fig. A.3a, we plot the KL term evaluated on the MNIST test dataset vs three $\beta$ values $[0.001, 0.005, 0.010]$. We observe an optimal region around $0.005$. On CIFAR10 and Darcy Flow, we observe a good tradeoff between the generation and reconstruction for $\beta = 0.001$. We leave a formal strategy for selecting good $\beta$ for `Latent-CFM` as an interesting future research direction.

# L Additional results on Darcy Flow data

This section provides the results of running the `Latent-CFM` with a pretrained VAE model following Alg. 1. We pretrain a VAE following the architecture in Rombach et al. [2022] for 100K steps. For ease of comparison, we keep the encoder size the same ($\sim$ 1M parameters) as one of the `Latent-CFM` described in Sec. 4.3 of the main paper. We train the `Latent-CFM` following the CIFAR10 training in Sec H, where we freeze the pretrained VAE encoder and add another trainable linear layer on top that outputs the mean and log-variance of the encoder Gaussian distribution. We fix the $\beta = 0.001$ for both the base VAE training and during `Latent-CFM` training.

## L.1 Effect of pretraining

We compare the `Latent-CFM` model trained using a pretrained VAE with the one-stage model described in Sec H with the same encoder size ($\sim$ 1M parameters). Fig. A.3b shows the boxplot of residual MSE for the two models trained on the Darcy Flow dataset. We observe that having a pretrained VAE helps to smooth the extreme outlier samples in terms of the residual MSE. However, the one-stage model performs better with a lower median residual MSE. Both models show a lower median residual MSE than the I-CFM model (Table 3 in the main paper) on this dataset.

## L.2 2d Latent space traversal

To study the learning of latent space, we trained a `Latent-CFM` model with a 2d latent space. We use the same pretrained VAE encoder ($\sim$ 1M parameters) as the previous section, and fix the output

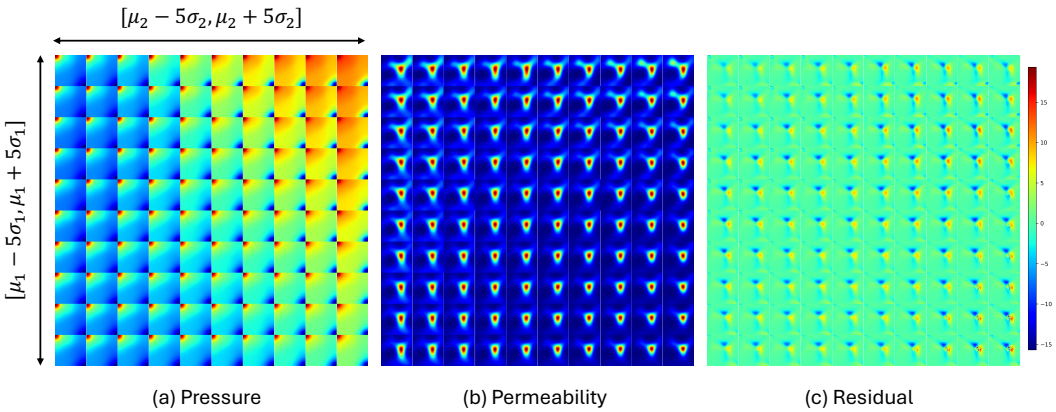
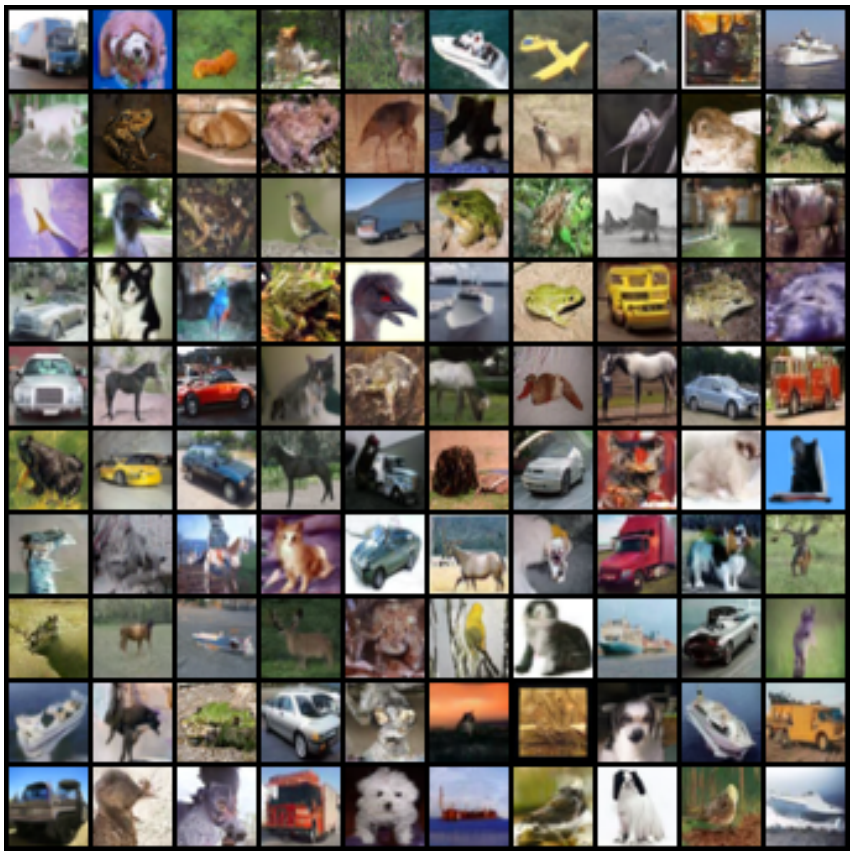



Figure A.4: Plot shows the generated pressure and permeability fields and the residual measuring physical accuracy of generation from varying the two latent dimensions within $[\mu - 5\sigma, \mu + 5\sigma]$. We observe that traversing the latent space produces a variety of samples for both fields. In addition, the variation in the latent space seems to maintain the physical consistency of the samples in terms of the residual. All images share the same source samples $x_0$.

dimension of the final trainable linear layer to produce mean and log-variance of a 2d Gaussian distribution.

In Fig. A.4, we plot generated fields and the residual metric by varying the two dimensions of the latent variable within the range $[\mu_i - 5\sigma_i, \mu_i + 5\sigma_i], i = 1, 2$, where $i$ denotes the dimension, and $(\mu_i, \sigma_i)$ denotes the mean and standard deviation respectively. We fix the same sample $x_0 \sim N(0, I)$ from the source distribution for all generations. We observe that traversing the latent space manifold has produced a variety of generated fields for both variables. In addition, `Latent-CFM` with varying latent variables seems to have generated physically consistent samples in terms of the residual. For the 2d grid of latent variables in Fig. A.4, the mean and median residual MSE were $3.614$ and $3.467$, respectively. In addition, with this model, we observed a median residual RMSE of $3.18$ across $500$ generated samples, which is the lowest across methods in Table 3 of the main paper. This demonstrates that traversing the latent space helps to generate physically accurate samples that share semantic similarities.

(a) Generated images for MNIST using `Latent-CFM`



(b) Generated images for CIFAR10 using `Latent-CFM`

# NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We presented evidence regarding the accuracy and efficiency claims in the abstract and introduction of the `Latent-CFM` by latent variables experimentally on benchmark datasets.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have highlighted in the conclusion that the sampling process of `Latent-CFM` can further be improved by learning a data-dependent prior.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We give the proof of Proposition A.1 (correct to the best of our knowledge) in the appendix.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provided a necessary snapshot of training `Latent-CFM` in the Experiments section and other parts of the main paper. We have provided additional implementation details in the appendix and also released the code.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We used publicly available image benchmark datasets. We provided details for generating the Darcy flow data. Also, we have made our codebase available.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provided details about all important hyperparameters in `Latent-CFM`.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We provided standard deviations for the measures, like W2 metric and the residual metric for evaluating physical accuracy.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provided the size of our models, the type of hardware used, and the computational cost.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We confirm that our research follows the NeurIPS Code of Ethics. We will ensure anonymity in all materials released.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: The paper emphasizes technical contributions without delving into broader societal implications. Although the application domains, such as energy-efficient generative modeling, suggest possible positive outcomes, these are not explicitly discussed, nor are any potential negative consequences addressed.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper introduces a new algorithm/framework. Based on the model and the benchmark data employed, there does not appear to be a significant risk of misuse that would require safeguards beyond standard open-source protocols.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All the papers and repositories used in this study are properly cited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The primary new asset is the source code for the proposed method, which is open source. Documentation is assumed to be provided alongside the code in its repository.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.