Efficient Flow Matching Using Latent Variables

Anirban Samaddar
Argonne National Laboratory

Yixuan Sun
Argonne National Laboratory

Viktor Nilsson
KTH Royal Institute of Technology

Sandeep Madireddy
Argonne National Laboratory

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 utilize the underlying clustering structure 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. To this end, we present `Latent-CFM`, which provides efficient training strategies by conditioning on the features extracted from data using pretrained deep latent variable models. Through experiments on synthetic data from multi-modal distributions and widely used image benchmark datasets, we show that `Latent-CFM` exhibits improved generation quality with significantly less training and computation than state-of-the-art flow matching models by adopting pretrained lightweight latent variable models. Beyond natural images, we consider generative modeling of spatial fields stemming from physical processes. Using a 2d Darcy flow dataset, we demonstrate that our approach generates more physically accurate samples than competing 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 about the underlying structures of the target data (such as low-dimensional clusters) is not explicitly included in the modeling, which can potentially lead to inefficiency in the FM training and convergence. One component of FM that could incorporate such information is the source distribution. Typically, an isotropic Gaussian Lipman et al. (2023); Tong et al. (2024) is used as the source, in an independent coupling with the target. 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 clustering 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

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300. Correspondence: `asamaddar@anl.gov`, `yixuan.sun@anl.gov`, `vikn@kth.se`, `smadireddy@anl.gov`.

lead to suboptimal performance for high-dimensional datasets while requiring customized training strategies that are dataset-dependent and expensive, thus restricting their broader applicability.

To address these limitations, we adapt advances in deep latent variable modeling Kingma and Welling (2022) to 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 low-dimensional clustering structures of the data 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 2d synthetic mixture and popular image benchmark datasets such as MNIST, CIFAR10, and ImageNet.
- 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 ability of our method in feature-conditional generation and extend it to compositional generation from the product of two feature-conditioned distributions. This allows generation-based low-frequency features learned from compressed latent representations, allowing regeneration of images with similar characteristics; see Fig. 6.

2 Related works

Continuous flow-based generative models, like diffusion and flow matching, evolve samples from a (typically) 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. Many works have considered conditional generation based on these models Dhariwal and Nichol (2021); Ho and Salimans (2022); Zheng et al. (2023), e.g. based on class labels, or text captions (prompts), by incorporating this information in the input of the network. This enables techniques such as classifier-free guidance, and typically leads to better overall performance.

Source sample structure A different approach to generating samples based on structural information is to incorporate this in the source distribution. 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 materials discovery. Diffusion Schödinger bridges De Bortoli et al. (2021) is an unsupervised framework for learning couplings π between the source and target distributions such that target samples are close to their source, in accordance with some reference corruption process, the low-noise limit being optimal transport (OT) couplings. Another line of work in this direction uses discrete OT-couplings computed on batches for training the flow network Pooladian et al. (2023); Tong et al. (2024). This similarly promotes OT-like proximity between source and target samples.

Latent variables Recent studies Guo and Schwing (2025); Jia et al. (2024) in incorporating data structures in diffusion and flow matching models have focused on a modeling approach where the flow network is conditioned on a latent variable. 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 conditional 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 supplementary material).

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

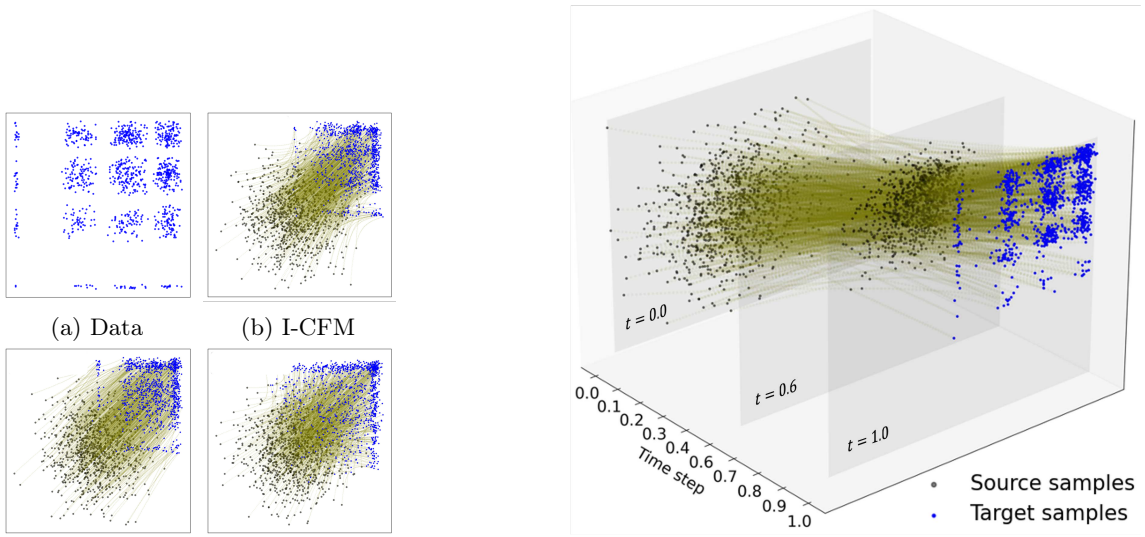
This section describes our proposed method `Latent-CFM`. First, we describe the notations that will be followed throughout the manuscript.

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,

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



(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.

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. We denote $p_t(.)$ as the generated probability path by Eq. 1, with $p_0(.), p_1(.)$ as the source and target distribution, respectively.

Flow Matching: Flow matching involves learning the vector field $u_t(.)$ that generates the flow from source to target distribution 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, $\mu_t' = \frac{d\mu_t}{dt}; \sigma_t' = \frac{d\sigma_t}{dt}$.

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 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:

$$p_t(x) = \int p_t(x|x_0, x_1)q(x_0, x_1)dx_0dx_1 \tag{4}$$

$$u_t(x) = \int u_t(x|x_0, x_1)\frac{p_t(x|x_0, x_1)q(x_0, x_1)}{p_t(x)}dx_0dx_1 \tag{5}$$

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 that 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 $p_t(x_0, x_1|x)$ involves 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. 2. 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.

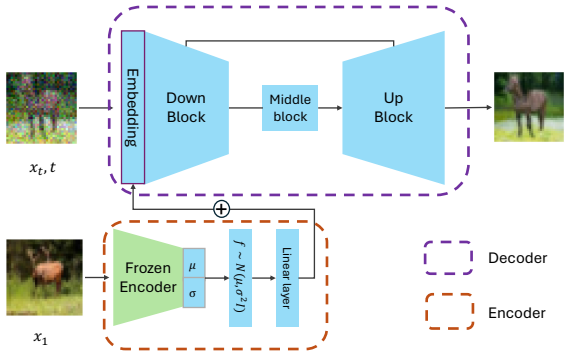


Figure 2: Schematic of `Latent-CFM` framework. Given a data x_1, `Latent-CFM` extracts latent features using a frozen encoder and a trainable stochastic layer. The features are embedded using a linear layer and added to the learned vector field. The framework resembles an encoder-decoder architecture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

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 `Latent-CFM` that combines recent developments of LVMs with flow matching for improved generative modeling.

We implicitly 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,

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

where $q(x_0, x_1|f)$ 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.

3.2.1 Choice of $q(.|x_1)$

The latent posterior distribution $q(.|x_1)$ should be easy to sample from and capture high-level structures in the data. 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 supplementary material.

In this study, we pretrain the VAEs optimizing the standard negative ELBO loss in 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 λ_{final}) that outputs $(\mu, \log(\sigma))$ and we regularize its learning with a KL-divergence term added to Eq. 10:

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

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 endpoints, which enables sample generation conditioned on the data features. Sec. B discusses the difference between the two loss functions in more detail.

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

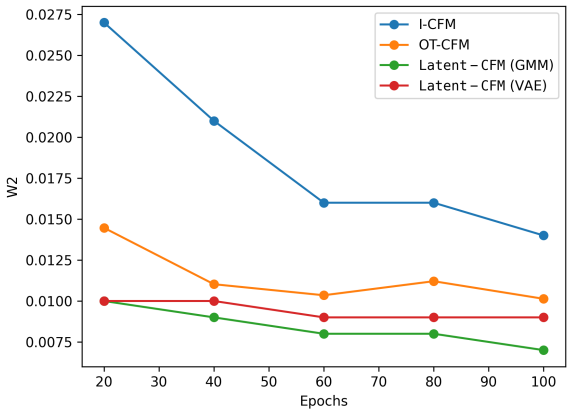


Figure 3: W2 vs epochs on 2d synthetic dataset

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.

3.2.2 Algorithm

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. Finetuning the last layer also enables regularizing the information learned in the latent space through the KL term in 11.

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 Experiments

We compare the proposed `Latent-CFM` against I-CFM Lipman et al. (2023), OT-CFM Tong et al. (2024), and VRFM Guo and Schwing (2025) on (a) unconditional data generation using synthetic 2d and high-dimensional image datasets and (b) physical data generation. We then analyze the learned latent space in the `Latent-CFM` model. Implementation details for all experiments are presented in the appendix.

Method	**W2** (↓)
OT-CFM	0.010 ± 0.0031
I-CFM	0.014 ± 0.0066
VRFM	0.050 ± 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.

4.1 Synthetic data sets

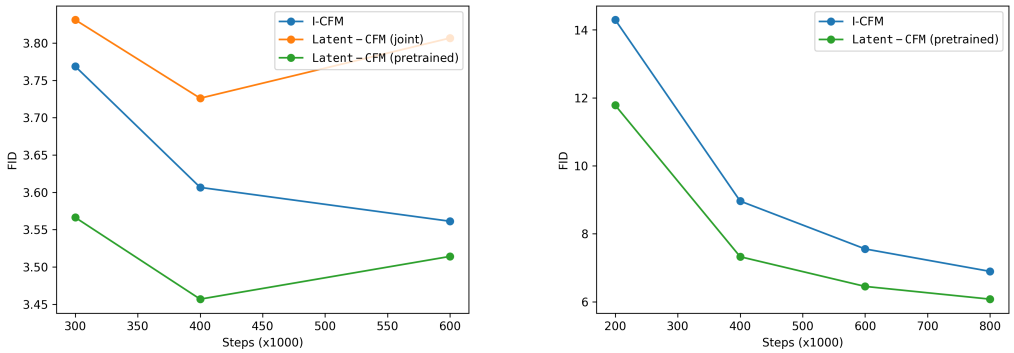
We use the 2d Triangle dataset Pichler et al. (2022); Nilsson et al. (2024) to benchmark the models' generation quality. The data distribution contains 16 modes (Fig. 1(a)) with different densities, and we generate 100K samples and divide them equally between the training and testing datasets.

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 network. For the `Latent-CFM`, we consider two variants differing in their pre-trained feature extractors: (1) a 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. For sampling, we have used the `dopri5` solver to solve the ODE in Eq. 1.

Training efficiency Fig. 3 shows the Wasserstein-2 (W2) distances with the test data vs the training epochs for the two `Latent-CFM` variants and the I-CFM and OT-CFM on the 2d synthetic dataset. We observe that `Latent-CFM` methods show lower W2 distances than the baseline methods for all training epochs. `Latent-CFM` with the GMM achieves a marginally better result than with the VAE encoder.

Generation quality 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 present all modes of the true data distribution. Table 1 shows the summary (mean ± standard deviation) of W2 metrics of all models, where the `Latent-CFM` variants exhibit lower W2 distances from the test samples than the competing methods. The GMM variant of `Latent-CFM` achieves the lowest W2 score.

The VRFM method shows the highest W2 distance among all methods. Fig. A.1 shows the effect of simplifying the VRFM input from (x_0, x_1, x_t, t) to x_1 for



(a) CIFAR10 (b) ImageNet

Figure 4: FID vs training steps show that `Latent-CFM` with pretrained VAE shows better generation quality compared to baseline methods on both CIFAR10 and ImageNet datasets early in the training process.

Methods	CIFAR10				MNIST				ImageNet		
	# Params.	FID (↓)			# Params.	FID (↓)			# Params.	FID (↓)	
		100	1000	Adaptive		100	1000	Adaptive		ODE	SDE
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	675.1M	6.893	6.745
`Latent-CFM` (joint)	47.3 M	4.675	3.931	3.807	1.58 M	**13.818**	14.572	**14.674**	-	-	-
`Latent-CFM` (pretrained)	36.1 M	**4.246**	3.575	**3.514**	1.58 M	13.848	**14.543**	14.694	675.2M	**6.076**	**5.955**

Table 2: Image generation performance of `Latent-CFM` compared to I-CFM and OT-FM on natural image datasets. 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 all datasets.

the encoder model (details in Sec. J). The plot shows a significant improvement in the generation with an improved final W2 distance of 0.015 vs the 0.050 in Table 1. This demonstrates that it is easier to learn with the `Latent-CFM` encoder model, $q(.|x_1)$, with the data as input, due to the underlying multimodal data distribution.

4.2 Unconditional Image Generation

For image generation, we train the OT-CFM, I-CFM, and `Latent-CFM` on MNIST, CIFAR10, and ImageNet 256×256 datasets. The details of the datasets are in the appendix. On MNIST and CIFAR10, 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 and implemented two VRFM variants with inputs (1) (x_1, x_0, x_t, t), and (2) (x_1, t) on CIFAR10. We train two variants of `Latent-CFM` with: (1) a **pretrained** encoder where only λ is updated during training of the vector field network, and (2) an encoder **jointly** whose full parameter set is trained from scratch together with the vector field network. We fix $\beta = 0.005$ for MNIST and $\beta = 0.001$ for CIFAR10 in Eq. 11. The analysis on the impact of β is in Sec. I. We train all models for 600K steps on CIFAR10 and 100K on MNIST. For both datasets, we use both a fixed-step Euler solver for 100 and 1000 steps and an adaptive `dopri5` solver for solving the ODE.

On ImageNet, we compare `Latent-CFM` with the SiT model Ma et al. (2024), which is an I-CFM model. We train both models for 800K steps. We pretrain a VAE for 200K steps and use the encoder for training `Latent-CFM`. We fix $\beta = 0.001$ and the latent dimension to be 128 for the `Latent-CFM` training. We follow the Stable Diffusion Rombach et al. (2022) architecture for training flows embedded in a latent space of smaller dimension. We were unable to find an open-source implementation for VRFM on ImageNet. We use an adaptive `dopri5` solver for ODE sampling and an Euler-Maruyama sampler for 250 steps for SDE sampling.

Training efficiency Fig. 4 shows the FID vs training steps for I-CFM and `Latent-CFM` on CIFAR10 and ImageNet. On both datasets, `Latent-CFM` with pretrained VAE exhibits significantly lower FID than I-CFM across training steps, demonstrating efficiency. On both datasets, compared to I-CFM, `Latent-CFM` achieves similar levels of FID (∼ 3.55 on CIFAR10, ∼ 7 on ImageNet) with 50% fewer training steps. Note that the best FID for `Latent-CFM` is ∼ 3.467, which is lower than the final I-CFM FID ∼ 3.561 and is the minimum across methods and solvers (Table 2) on CIFAR10. On CIFAR10, `Latent-CFM` with a pretrained encoder shows better (∼ 7% lower FID across steps) efficiency than a jointly trained model. On ImageNet, we observe a significant decrease in training speed to 1.3 steps/second for `Latent-CFM` with a jointly trained encoder, from 2.4 steps/second using a pretrained encoder. This further demonstrates the efficiency of `Latent-CFM` compared to VRFM (which jointly trains an encoder with a larger set of inputs) on large datasets. We add

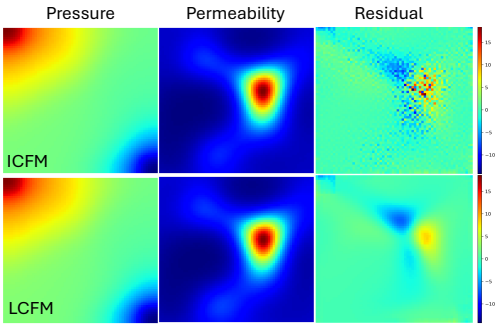


Figure 5: Plot showing a generated sample from I-CFM (top row) and `Latent-CFM` (bottom row) models trained on 10K samples generated by solving Darcy Flow equations Jacobsen et al. (2025). Visually generated samples from both models resemble true Pressure and Permeability fields. However, `Latent-CFM` exhibits a better fit to the Darcy flow equations as measured by the residuals.

the results for the `Latent-CFM` with a jointly trained encoder in Appendix O.

Generation quality 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, MNIST, and ImageNet for different solvers and different numbers of integration steps. All models are evaluated at their final training step. We observe that `Latent-CFM` variants consistently outperform the I-CFM and OT-CFM across datasets and solvers in terms of FID. On MNIST, two variants of `Latent-CFM` show similar FID. On CIFAR10, `Latent-CFM` with a pretrained encoder achieves lower FID (by $\sim 7\%$) compared to the jointly trained model, highlighting the benefit of a pretrained encoder model. `Latent-CFM` sampling uses features learned from the training data. Sec. K shows that this does not prohibit our approach from generalizing beyond the training data.

With our best efforts, we were unable to reproduce the VRFM FID numbers on CIFAR10 from Guo and Schwing (2025). Table. A.2 compares the final step FID with `dopri5` solver for the two VRFM models with our approach and with the two best VRFM models reported in Guo and Schwing (2025). We observe that `Latent-CFM` performs similarly to the best-performing VRFM model from Guo and Schwing (2025) (with $\sim$ 20% less time in terms of GPU hours). We also observe in our implementation of VRFM that simplifying the input to the model results in lower FID.

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 $(2, 64, 64)$. We pretrain an embedding VAE model following Rombach et al. (2022) for 100k steps. We set the latent dimension to be 2 and $\beta = 0.001$ for the feature extractor.

	I-CFM		`Latent-CFM`
# Params.	35.7M	68.8M	35.7M
Residual median ↓	5.922	4.921	**3.18**

Table 3: Comparison of the PDE residuals of the generated Darcy flow samples ($[K, p]$ pairs) from `Latent-CFM` and I-CFM with two model sizes. `Latent-CFM` model size is 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.

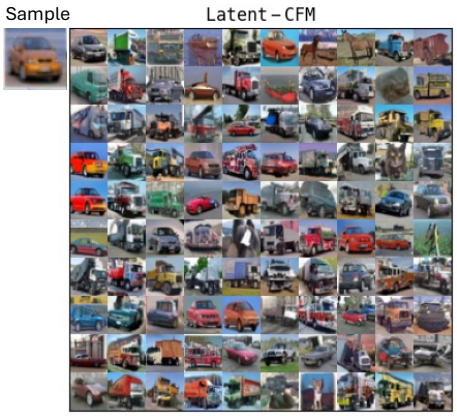
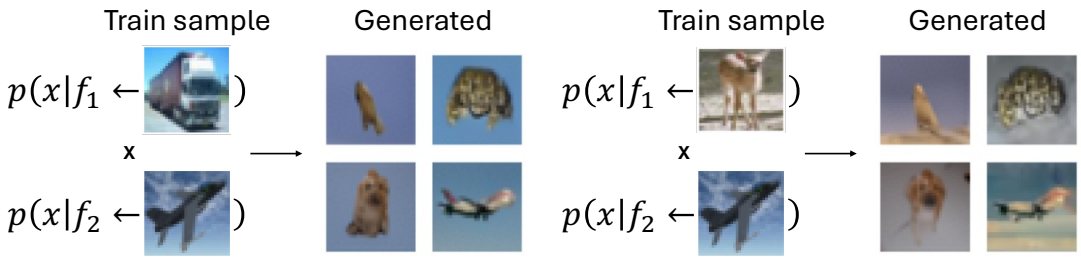


Figure 6: Conditioning the generation process of `Latent-CFM` on the features learned from the training samples shows that the framework generates samples by varying the objects while retaining properties like background, colors, object shape, etc. All samples shown have the same feature vector f respectively.

Generation quality Fig. 5 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 a smaller `Latent-CFM` (# parame-



(a) (b)

Figure 7: Plot showing a selected set of generated samples from the product of two feature-conditioned distributions using `Latent-CFM`. Changing one of the feature-conditioned distributions in the product, all generated samples share the high-frequency features (such as the skin pattern of the frog), varying the low-frequency features (like color and background).

ters 35.7M) significantly outperforms the large I-CFM model (# parameters 68.8M) in terms of the median residual. Using the 2d latent space, Fig. A.6 (details are in Sec. N) shows the latent traversal along each of the two latent coordinates. The figure shows that traversing the latent space generates physically consistent samples. The superiority of `Latent-CFM` in this dataset motivates us to investigate its performance for other scientific data generation tasks and examine the learned latent space in future work.

4.4 Latent space analysis

This section analyzes the latent space learned by the `Latent-CFM` models trained on the CIFAR10 dataset. We investigate (1) the effect of conditioning the `Latent-CFM` generation on the data features $f \sim q(.|x_1^{train})$, (2) compositional generation composing multiple feature-conditioned distributions learned from multiple data points.

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 $f \sim q_\lambda(f|x_1^{car})$ 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.

Composing feature-conditioned distributions Recent works Du et al. (2024); Bradley et al. (2025) in diffusion models have explored generating high-fidelity samples from a composition of class-conditional generative models. We extend this idea to flow matching models for generating samples from a composition of feature-conditioned distributions using `Latent-CFM`. Given two feature-conditioned densities $p(.|f_1), p(.|f_2)$ where the features $f_1 \sim q(.|x_1^{train}), f_2 \sim q(.|x_2^{train})$ are extracted from two training samples $(x_1^{train}, x_2^{train})$, we want to sample from the product distribution $p^1 = \prod_{i=1,2} p(.|f_i)$, which has high likelihood under both feature conditioned distributions. Sec. M describes the details of our inference algorithm.

Fig. 7 shows a selected set of generated samples from two products of feature-conditioned distributions. The two products share one common set of features coming from the image of the airplane (bottom row). The Gaussian source samples vary within the generated samples from each product, but are the same between the two. We observe that the generated samples share high-frequency features (such as the skin pattern of the frog) between the two distributions. However, the low-frequency features vary in the generated samples (color and background changes from blue to a mixture of blue and yellow) between the two products. This indicates that the feature extractor helps the `Latent-CFM` to condition the generation on the low-frequency features from the training data, while the CFM model varies the high-frequency features to generate diverse samples. We provide an expanded set of 100 generated samples from the two product distributions in Fig. A.4.

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 that incorporates the underlying clustering structure 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 and compositional gener-

ation 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. It would also be interesting to disentangle the latent features, which can further improve control over the generation process. 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 Acknowledgments

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 the 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.

Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. In *International conference on machine learning*, pages 254–263. PMLR, 2018.

Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

Arwen Bradley, Preetum Nakkiran, David Berthelot, James Thornton, and Joshua M. Susskind. Mechanisms of projective composition of diffusion models, 2025. URL `https://arxiv.org/abs/2502.04549`.

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.

Yilun Du, Conor Durkan, Robin Strudel, Joshua B. Tenenbaum, Sander Dieleman, Rob Fergus, Jascha Sohl-Dickstein, Arnaud Doucet, and Will Grathwohl. Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and mcmc, 2024. URL `https://arxiv.org/abs/2302.11552`.

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`.

Xiangming Gu, Chao Du, Tianyu Pang, Chongxuan Li, Min Lin, and Ye Wang. On memorization in diffusion models. *arXiv preprint arXiv:2310.02664*, 2023.

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. Etflow: Equivariant flow-matching for molecular conformer generation, 2024. URL `https://arxiv.org/abs/2410.22388`.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022. URL `https://arxiv.org/abs/2207.12598`.

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

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large

language models, 2021. URL https://arxiv.org/abs/2106.09685.

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.

Nanye Ma, Mark Goldstein, Michael S. Albergo, Nicholas M. Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers, 2024. URL https://arxiv.org/abs/2401.08740.

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.

Aram-Alexandre Pooladian, Heli Ben-Hamu, Carles Domingo-Enrich, Brandon Amos, Yaron Lipman, and Ricky T. Q. Chen. Multisample Flow Matching: Straightening Flows with Minibatch Couplings, May 2023. URL http://arxiv.org/abs/2304.14772. arXiv:2304.14772 [cs].

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.

Gowthami Somepalli, Vasu Singla, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Understanding and mitigating copying in diffusion models. *Advances in Neural Information Processing Systems*, 36:47783–47803, 2023.

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

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.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

Qinqing Zheng, Matt Le, Neta Shaul, Yaron Lipman, Aditya Grover, and Ricky T. Q. Chen. Guided flows for generative modeling and decision making, 2023. URL `https://arxiv.org/abs/2311.13443`.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes/No/Not Applicable] Yes
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes/No/Not Applicable] Yes
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes/No/Not Applicable] Yes

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes/No/Not Applicable] Yes
   (b) Complete proofs of all theoretical results. [Yes/No/Not Applicable] Yes
   (c) Clear explanations of any assumptions. [Yes/No/Not Applicable] Yes

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes/No/Not Applicable] Yes
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes/No/Not Applicable] Yes
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes/No/Not Applicable] Yes for W2 distances in Table 1. FIDs have little variation.
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes/No/Not Applicable] Yes.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes/No/Not Applicable] Yes
   (b) The license information of the assets, if applicable. [Yes/No/Not Applicable] Not Applicable
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes/No/Not Applicable] Yes
   (d) Information about consent from data providers/curators. [Yes/No/Not Applicable] Not Applicable
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Yes/No/Not Applicable] Not Applicable

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Yes/No/Not Applicable] Not Applicable
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Yes/No/Not Applicable] Not Applicable
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Yes/No/Not Applicable] Not Applicable

Appendix

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 (μ, 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 μ_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 $g(t,\omega) \coloneqq h(tX_0(\omega) + (1-t)X_1(\omega))$, the conditions for doing so in $\frac{d}{dt}\int g(t,\omega)\,\mathbb{P}_f(d\omega)$ are fulfilled, see e.g. (Durrett, 2019, Ch. A5). Most importantly, $\frac{d}{dt}g(t,\omega) = \langle \nabla h(tX_0 + (1-t)X_1), X_1 - X_0\rangle_{\mathbb{R}^d}(\omega) = \langle \nabla h(X_t), \dot{X}_t\rangle_{\mathbb{R}^d}(\omega)$, 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 objective in Eq 21 is that the encoder model q_ϕ 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_ϕ 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 μ and a diagonal covariance matrix $\sigma^2 I$. The encoder network $q_\lambda(f|x_1)$ outputs the parameters μ and σ^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

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

where Ω 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

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

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

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

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 5 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 Ω 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)[1], we set the source term as

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

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 λ_i and $\phi_i(x)$ are eigenvalues and eigenfunctions of the covariance function sorted by decreasing λ_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 64×64 grids for model training. During training, we standardize both the permeability and pressure fields using $\mu_K = 1.1491$, $\sigma_K = 7.8154$, and $\mu_p = 0.0$, $\sigma_p = 0.0823$.

[1]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.1 `Latent-CFM` training

Given n sample $(x_1^1, ..., x_n^1)$ from $p_1(x)$, regularizer β;
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 λ_{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 θ, λ_{last}
end for
return $v_\theta(\cdot, \cdot, \cdot), q_{\hat{\lambda}}(.|x)$

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 $(x_1^1, ..., x_K^1)$

F `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.

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.

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 θ
end for
return $v_\theta(\cdot, \cdot, \cdot), q_{\hat{\lambda}}(.)$

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

Given sample size K, trained $v_{\hat{\theta}}(., ., .)$ and $q_{\hat{\lambda}}(.)$, number of ODE steps n_{ode}
Select K random cluster memberships $(c_1, ..., c_K)$ from the distribution $Categorical(\hat{w}_1, ..., \hat{w}_K)$
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**
  $x_i^{t+h} = \text{ODEstep}(v_{\hat{\theta}}(x_i^t, c_i, t), x_i^t)$
end for
return Samples $(x_1^1, ..., x_K^1)$

G 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_CFM-66CF/README.md`. We closely follow the implementation in the Tong et al. (2024) repository[2].

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[3], and CIFAR10[4]. We take the latent encodings from the last encoder layer and add a trainable MLP layer to output the mean and log-variance of

[2]https://github.com/atong01/conditional-flow-matching.git
[3]https://github.com/csinva/gan-vae-pretrained-pytorch
[4]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.

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. For encoder joint training with `Latent-CFM`, we keep the same architecture as in the above codebases.

We have also implemented the VRFM model on the CIFAR10 dataset. Following Guo and Schwing (2025), we adopted the downsampling layers of the vector field network to reduce the input to a latent of size $(512, 1)$. The addition of the latents to the vector field network follows the same architecture as described in Fig. 2.

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 $\beta = 0.001$.

ImageNet We follow the same SiT model architecture from Ma et al. (2024) for our experiments in ImageNet. SiT is a latent flow matching model. For the `Latent-CFM`, we pretrained a VAE following Rombach et al. (2022) as described above on the latents of the training data for 200K steps. For reproducibility, we fixed all hyperparameters of the CFM training the same as Ma et al. (2024).

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.

H Metrics

H.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`[5] Python library.

H.2 Residual metric on Darcy Flow data

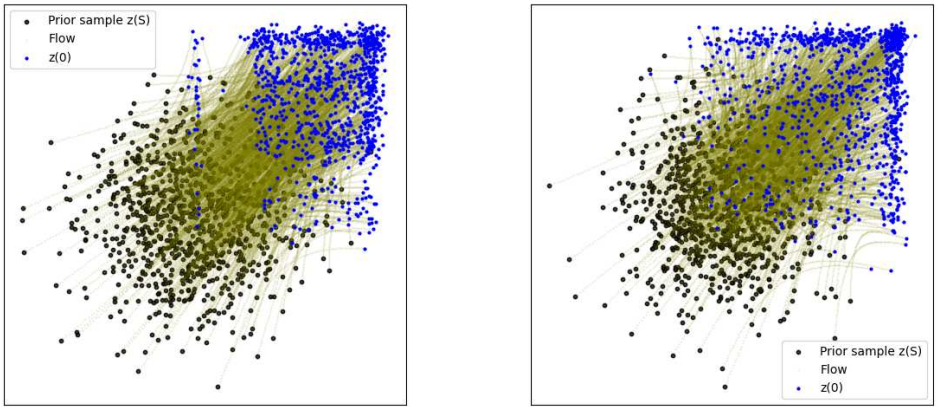
After rescaling to the original space, we compute the spatially averaged squared residuals of the governing equation across the domain to evaluate the sample quality in the 2D Darcy flow experiment. In particular, for each generated sample, we compute

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

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.

I Selection of β

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



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

Figure A.1: 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.

An important hyperparameter of `Latent-CFM` training is the KL regularization parameter β. It controls the information compression of the latent space Alemi et al. (2019). In `Latent-CFM`, too high β 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.2, we plot the KL term evaluated on the MNIST test dataset vs three β 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 β for `Latent-CFM` as an interesting future research direction.

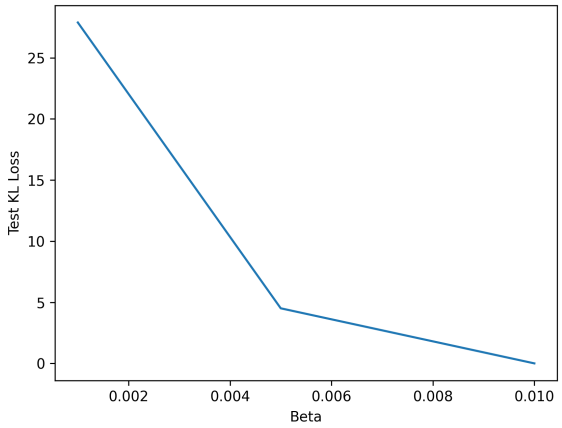


Figure A.2: Plot of the regularization parameter β and the KL divergence on test set shows an optimal β around $5e-3$ on MNIST.

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 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.1 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.

Method	CIFAR10 FID (↓)
VRFM-1 Guo and Schwing (2025)	3.561
VRFM-2 Guo and Schwing (2025)	**3.478**
VRFM (x_0, x_1, x_t, t)	16.276
VRFM (x_1, t)	6.480
`Latent-CFM` (pretrained)	**3.514**

Table A.2: Comparison between VRFM and `Latent-CFM` in terms of FID on CIFAR10. We were unable to reproduce the FID numbers from Guo and Schwing (2025) presented in the top two rows. `Latent-CFM` shows a similar FID to the best VRFM model from Guo and Schwing (2025). Additionally, we observe performance improvements as we simplify the input to VRFM in our implementation.

Methods	# Params.	Sinkhorn (↑)	Energy (↑)	Gaussian (↑)	Laplacian (↑)
ICFM	35.8M	675.613	12.690	**0.0020**	**0.0018**
`Latent-CFM`	36.1M	**680.500**	**12.698**	**0.0020**	**0.0018**

Table A.3: Table shows the generalization for ICFM and `Latent-CFM` in terms of different distances (average over 30 batches) from the train dataset on CIFAR10. In terms of most metrics, the two approaches show similar distance from the train dataset.

K Generalization of `Latent-CFM`

`Latent-CFM` sampling uses latent features f learned from training data samples $q(.|x^{train})$. This is uncommon for flow-based modeling and most other deep generative model frameworks, since the model has an additional, non-parametric component that models random features of the data. Note that there are many non-parametric models that directly use the training set during sampling, e.g. kernel density estimation and Gaussian processes Bishop and Nasrabadi (2006), while neural networks are sometimes seen as non-parametric, due to their overparametrization and their memorization capabilities, manifested by compressing/memorizing the training data into their weights Zhang et al. (2016); Arora et al. (2018). In practice, deep generative models may also memorize features (or even entire samples in the overfitting regime) Gu et al. (2023); Somepalli et al. (2023). To assess that our model does not cheat, i.e. copy too much information from the training sample, we perform additional comparisons between generated samples, their conditioning training sample, and the train/test sets at large.

To investigate the generalization of the generated samples beyond the training data, Fig. A.3 compares the 10 nearest neighbors of 10 `Latent-CFM` generated samples from the train and test data, and the training images used to generate the samples on CIFAR10. In most cases, we observe that the generated samples, although they share features with, significantly differ from nearest neighbors in the train and test data. This demonstrates that the generated samples don't reconstruct the training data. In addition, for most samples, the training data point used for feature extraction does not appear (except for the red box) within the 10 nearest neighbor samples in the train dataset, demonstrating generalization beyond the training set.

To quantify generalization beyond training data, Table A.3 shows the distance metrics (averaged over 30 batches of size 500) between generated samples using ICFM and `Latent-CFM` and the training dataset on CIFAR10. We have used the `geomloss` Python package to calculate the distances. In terms of most distances, both ICFM and `Latent-CFM` show similar generalization. Additionally, Table A.4 shows FID for ICFM and `Latent-CFM` for three training checkpoints calculated on the CIFAR10 test dataset. We observed that our approach shows better FID than ICFM across all checkpoints.

L Latent space traversal on MNIST

Fig. A.5 shows the latent space traversal for the pretrained 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 (μ, σ) represent the encoded mean and variance. For each row, the new samples correspond 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.

M Composition of feature-conditioned distributions

This section describes the algorithm and derivations for our sampling algorithm from the product of two feature-conditioned `Latent-CFM` models. The key ingredient involves the relation between the vector field $u_t(.)$

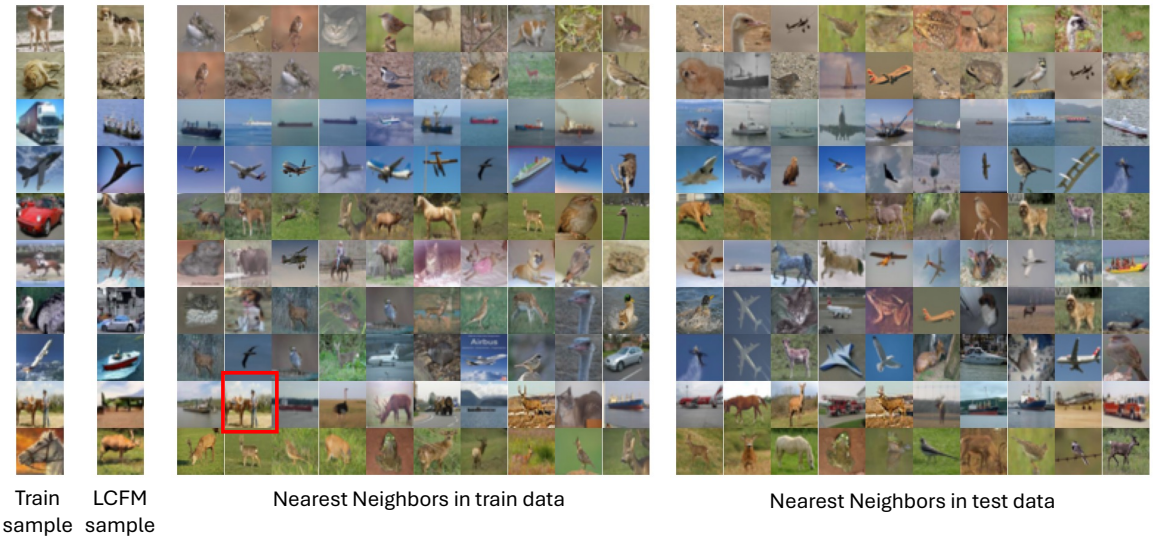


Figure A.3: Plot shows 10 nearest neighbors of `Latent-CFM` samples on the train and test dataset with the training images used for feature extraction (on left). `Latent-CFM` generated samples generalize well, although they share features with the nearest neighbors in the train data. For one sample, one of the nearest neighbors (marked red) matches the training samples used for feature extraction.

(a) $f_1 \sim q(.|x^{truck}), f_1 \sim q(.|x^{airplane})$ (b) $f_1 \sim q(.|x^{deer}), f_1 \sim q(.|x^{airplane})$

Figure A.4: Expanded set of 100 samples from the two product distributions.

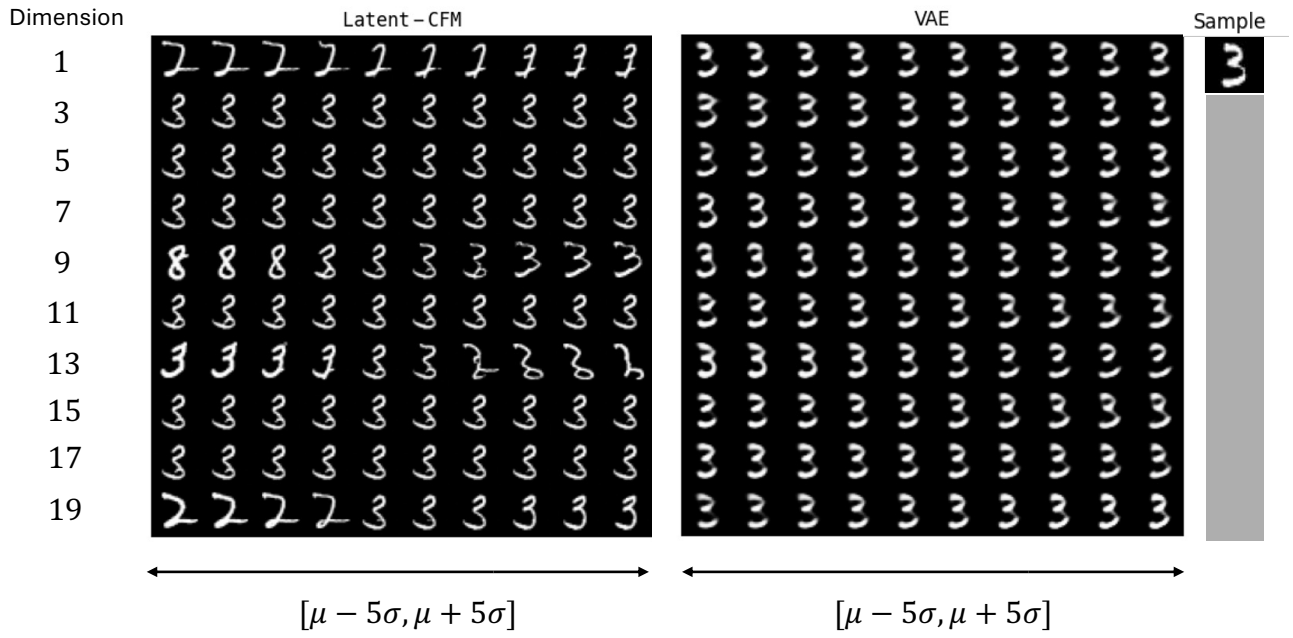


Figure A.5: Traversal of the latent space f shows that `Latent-CFM` generates a diverse set of samples than the pretrained VAE. All generated images for `Latent-CFM` share the same source samples $x_0 \sim N(0, I)$. The latent space traversal shows we can generate different digits with similar latent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

Methods	**CIFAR10 testset FID** (↓)		
	300K	**400K**	**600K**
ICFM	5.769	5.742	5.652
`Latent-CFM` (pretrained)	**5.710**	**5.563**	**5.631**

Table A.4: `Latent-CFM` compared to I-CFM in terms of testset FID on CIFAR10 for three training checkpoints. Our method exhibits improved FID over ICFM for all checkpoints.

and the score $\nabla \log p_t(.)$ where $p_t(.)$ is the probability path at time t. The following Lemma from Zheng et al. (2023) describes this relation for the Gaussian probability paths.

Lemma M.1. *Let $p_t(x|x_1) = N(x|\alpha_t x_1, \sigma_t^2 I)$ be a Gaussian Path defined by a scheduler (α_t, σ_t), then its generating vector field $u_t(x|x_1)$ is related to the score function $\nabla \log p_t(x|x_1)$ by,*

$$u_t(x|x_1) = a_t x + b_t \nabla \log p_t(x|x_1) \tag{31}$$

where,

$$a_t = \frac{\dot{\alpha}_t}{\alpha_t}, b_t = (\dot{\alpha}_t \sigma_t - \alpha_t \dot{\sigma}_t)\frac{\sigma_t}{\alpha_t} \tag{32}$$

In this paper, we used linear interpolation paths in `Latent-CFM` $x_t = tx_1 + (1-t)x_0$, where $x_0 \sim N(0, I)$. It is easy to show that, $a_t = \frac{1}{t}, b_t = \frac{1-t}{t}$. Therefore, we can use Eq. 31 to convert the learned vector field $v_{\hat{\theta}}(.;.,t)$ to score estimator $s_{\hat{\theta}}(.;.,t)$.

We aim to draw samples from the product probability path $p_t^1 = \prod_{i=1,2} p_t(x|f_i)$ where $f_i \sim q(.|x_i)$ and (x_1, x_2) are two images. Using Lemma M.1, we can derive the vector field underlying the product, $u_t^1(x) = -a_t x + u_t(x|f_1) + u_t(x|f_2)$. During the reverse ODE sampling, we replace the true vector fields with their learned networks,

$$v^1(x;.,t) = -a_t x + v_{\hat{\theta}}(x; f_1, t) + v_{\hat{\theta}}(x; f_2, t) \tag{33}$$

With these derivations, we perform the predictor-corrector sampling using Langevin dynamics from Song et al. (2020) to sample from the product distribution. Algorithm A.5 describes the steps of the sampling process.

As stated in Section 4.4, assuming conditionally independent latent variables f_1 and f_2 given x, we can construct a new distribution. Repeatedly using Bayes theorem, we can obtain

$$p(x|f_1, f_2) = \frac{p(x|f_1)p(x|f_2)}{p(x)} \cdot \frac{p(f_1)p(f_2)}{p(f_1, f_2)} \propto \frac{p(x|f_1)p(x|f_2)}{p(x)}.$$

Algorithm A.5 Sampling from product of feature-conditioned models

trained $v_{\hat{\theta}}(., f_i, .)$ where $f_i \sim q_{\hat{\lambda}}(.|x_i)$, number of ODE steps n_{ode}, number of Langevin steps n_ℓ, drift and diffusion schedulers $(\epsilon_t^{drift}, \epsilon_t^{diffusion})_{t\in[0,1]}$
Sample $x_0 \sim N(0, I)$
$h \leftarrow \frac{1}{n_{ode}}$
for $t = 0, h, ..., 1-h$ **do**
  $x_{t+h} = \text{ODEstep}(v^1(x_t; ., t), x_t)$ ▷ **Predictor step**
  for $j = 1, ..., n_\ell$ **do**
    $z \sim N(0, I)$
    $x_{t+h} = x_{t+h} + \epsilon_{t+h}^{drift} s_{\hat{\theta}}(x_{t+h}; ., t) + \sqrt{2\epsilon_{t+h}^{diffusion}} z$ ▷ **Corrector step**
  end for
end for
return x_1

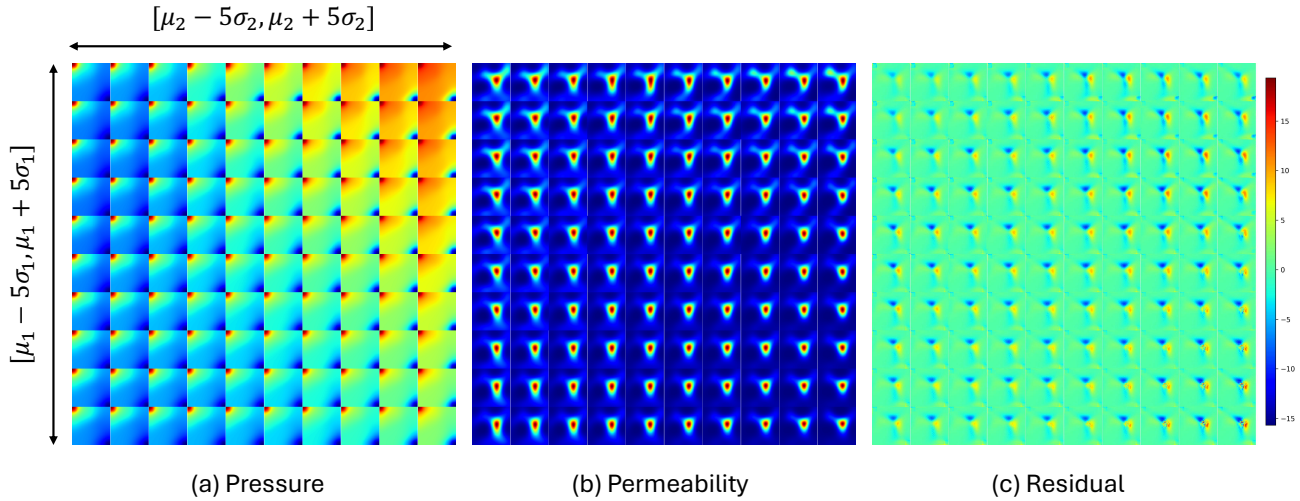


Figure A.6: Plot shows the generated pressure and permeability fields and the residual measuring physical accuracy of generation from varying the two latent dimensions within $[\mu - 5\sigma, \mu + 5\sigma]$. We observe that traversing the latent space produces a variety of samples for both fields. In 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.

Such construction requires a marginal data distribution $p(x)$ where $p(x|f_1, f_2)$ is a proper conditional. This means such $p(x)$ allows f_1 and f_2 to exist simultaneously for some $x = x'$. However, without a clear semantic meaning or disentanglement of the latent space, $p(x|f_1, f_2)$ is not well-defined. This might explain the unresolved center objects in the samples from the composed distribution.

To perform an effective composition in this case, Bradley et al. (2025) shows that $p(x|f_1)$ and $p(x|f_2)$ need to be independent. Therefore, it asks the latent variables to represent orthogonal concepts in x, which prompts the future direction of learning a latent space encapsulating mutually independent features of the input data for better compositional generation.

N Additional results on Darcy Flow data

In Fig. A.6, 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 (μ_i, σ_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.6, the mean and median residual MSE were 3.614 and 3.467, respectively. This demonstrates that traversing the latent space helps to generate physically accurate samples that share semantic similarities.

O `Latent-CFM` joint training results on ImageNet-256

In this section, we compare the two training strategies of `Latent-CFM`, (1) with a **pretrained** VAE encoder with the final layer parameters λ updated with the CFM training, and (2) an encoder **jointly trained** with the learned vector field on the ImageNet-256 dataset. For both models, we use the same architectures for VAE Rombach et al. (2022), and the vector field network Tong et al. (2024). Other hyperparameters, such as per-GPU batch size and learning rate, follow from Ma et al. (2024) and are kept the same for both models.

`Latent-CFM` with a pretrained encoder significantly improves ($\sim$ 2.4 steps/second) the training speed from the jointly trained model ($\sim$ 1.3 steps/second). Table A.5 shows the FID over training steps for both models. We observe that `Latent-CFM` with a pretrained encoder outperforms the jointly trained model across all training steps.

Steps	`Latent-CFM` (pretrained) FID (↓)	`Latent-CFM` (joint) FID (↓)
200K	**11.781**	44.259
400K	**7.324**	27.186
600K	**6.450**	20.608

Table A.5: Comparison between `Latent-CFM` with a **pretrained** and **jointly trained** encoder in terms of FID on ImageNet-256. `Latent-CFM` with a pretrained encoder performed significantly better than the jointly trained model across all training steps.

P Computational cost

For our experiments, we used between 1, 4, and 8 NVIDIA A100 GPUs to train all models. On MNIST, CIFAR10, and Darcy Flow datasets, it took approximately 3, 16, and 3.5 hours to complete $100K$, $600K$, and $100K$ steps of the `Latent-CFM` training, respectively. On ImageNet-256, it took 4.5 days to train `Latent-CFM` with a pretrained encoder model for $800K$ steps. On Darcy Flow and ImageNet data, we pretrained a VAE model, which took approximately 1.7 and 5.5 hours to complete $100K$ and $200K$ training steps, respectively.

Q Ablation Studies

In this section, we present ablation studies for `Latent-CFM` on CIFAR10 data with respect to (1) encoder architecture, (2) latent space dimension, and (3) finetuning strategy.

VAE Architectures We ran an ablation study comparing the generation quality of `Latent-CFM` with two pretrained VAE architectures on the CIFAR10 dataset. In the paper, we showed results for `Latent-CFM` with a frozen $\sim$ 20M parameter resnet18 VAE model[6]. In addition, we pretrained a $\sim$ 26M CNN VAE model following the Stable-Diffusion repository Rombach et al. (2022) for 100K steps. We fixed all other `Latent-CFM` parameters except the frozen VAE encoder. Table A.6 shows the FID from both models for three training checkpoints on CIFAR10. We observe that both models perform similarly across the training checkpoints, with LCFM-CNN achieving slightly better FID after 600K steps.

Training steps	FID LCFM-resnet18 (↓)	FID LCFM-CNN (↓)
300K	**3.566**	3.572
400K	**3.457**	3.481
600K	3.514	**3.506**

Table A.6: FID comparison across training steps

Latent dimension	FID (↓)
16	3.916
32	3.747
64	**3.514**
128	3.584
256	**3.517**
512	3.556

Table A.7: FID across latent dimensions

[6] https://github.com/Lightning-Universe/lightning-bolts/

Latent dimension We ran an ablation study for `Latent-CFM` w.r.t the latent dimension on the CIFAR10 dataset. Table A.7 shows FIDs from varying the dimension of the trainable stochastic layer in `Latent-CFM` within the range 16-512. We observe that FID increases for small latent spaces (≤ 32). The larger latent spaces tend to have similar FIDs (~ 3.5).

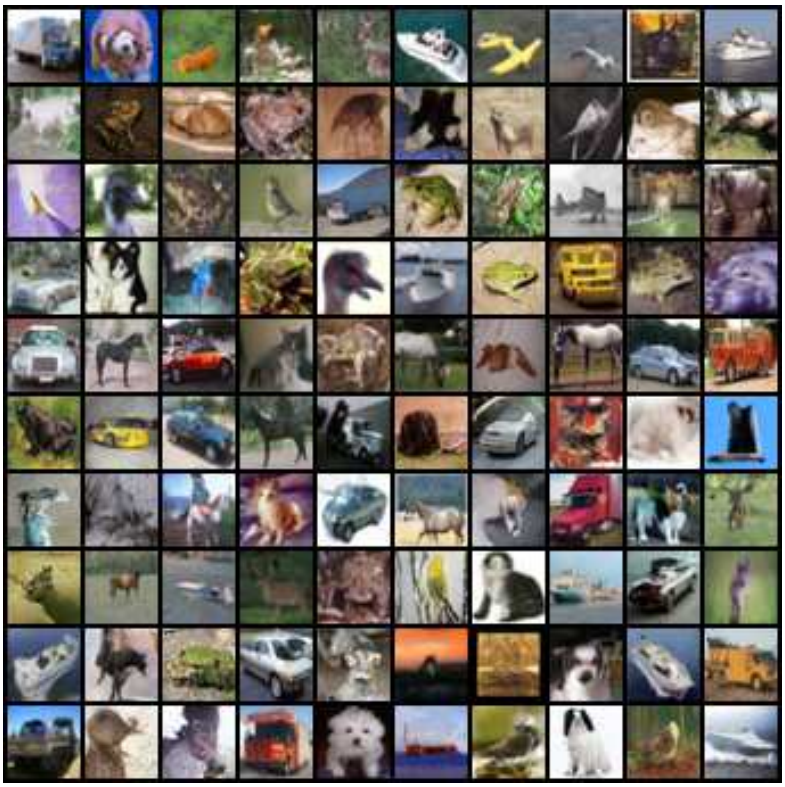
VAE finetuning with an adapter In the paper, to train `Latent-CFM` model with a pretrained VAE encoder, we freeze the encoder parameters and add a trainable MLP head to map to the mean and variance of the latent space. However, we can choose a different strategy where we add a low-rank adapter (LoRA) Hu et al. (2021) to the final layer of the VAE encoder and train only the adapter parameters. Table A.8 shows the FID resulting from `Latent-CFM` training with the two finetuning strategies on CIFAR10. Both strategies introduce very few ($< 1\%$) parameters to the model. We observe that the trainable MLP head for the frozen VAE encoder produces lower FID across training steps on CIFAR10.

Training steps	FID LCFM w MLP finetuning (↓)	FID LCFM w LoRA (↓)
300K	**3.566**	3.713
400K	**3.457**	3.727
600K	**3.514**	3.743

Table A.8: FID comparison with MLP finetuning and LoRA

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

=



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