Learning Straight Flows by Learning Curved Interpolants

Shiv Shankar [1] **Tomas Geffner** [2]

Abstract

Flow matching models typically use linear interpolants to define the forward/noise addition process. This, together with the independent coupling between noise and target distributions, yields a vector field which is often non-straight. Such curved fields lead to a slow inference/generation process. In this work, we propose to learn flexible (potentially curved) interpolants in order to learn straight vector fields to enable faster generation. We formulate this via a multi-level optimization problem and propose an efficient approximate procedure to solve it. Our framework provides an end-to-end and simulation-free optimization procedure, which can be leveraged to learn straight line generative trajectories.

1. Introduction

In recent years, neural generative models have become remarkably successful in a variety of domains, including computer vision [52], data generation [23], robotics [13], and scientific applications [16]. This success can be broadly attributed to the development of generative models based on learning vector fields [20, 56, 30, 1]. Among these methods, flow matching [30, 1] is a recent simulation-free method that has gained traction due to its robust performance. Flow matching is a version of Continuous Flows (CNFs) [5] that uses deep neural networks to learn a velocity vector field that transports samples from a source distribution to the target distribution. Peluchetti [38], Lipman et al. [30] show that a regression objective on conditional probability paths between source and target samples can be used to learn a model of the velocity field without an explicit target field.

Despite their success, flow-matching models (as well as other vector-field-based models) often suffer from slow sampling. Specifically, one has to simulate an ODE (or SDE) with a numerical solver to generate samples. Since these models learn a vector field that needs to be numerically integrated, the overall process can be slow and rife with discretization errors [46]. This is because the learnt vector fields often have curved trajectories, and therefore small discretization step-sizes are needed for accurate simulation.

To alleviate this issue, many methods have been proposed to learn more easily integrable vector fields. One common approach tries to incorporate 'straightness' to the field [31, 32, 28, 48], as straight paths allow for 1-step and discretization-free inference. While few proposals learn straight approximations to the true field [57, 59, 29], most of these share a common flaw. Most of these these methods use linear conditional paths and independent coupling during training. While easy to implement, the underlying vector field in these scenarios is fundamentally curved, and any 'straight' version will be necessarily approximate.

In this work, we propose a new method to learn straight flows using flow matching by tuning interpolants. We propose explicitly training the conditional probability paths (or interpolants) used in flow matching to enforce straightness in the velocity field. While mathematically simple, the corresponding bi-level optimization problem is intractable. We address this by proposing an approximate approach to solve this optimization problem, which relies on an an analytic form for the target vector field of a Conditional Flow Matching objective [30], and enforcing straightness on the target field tuning the interpolant. Empirically, we observe that our method outperforms recent models on standard datasets.

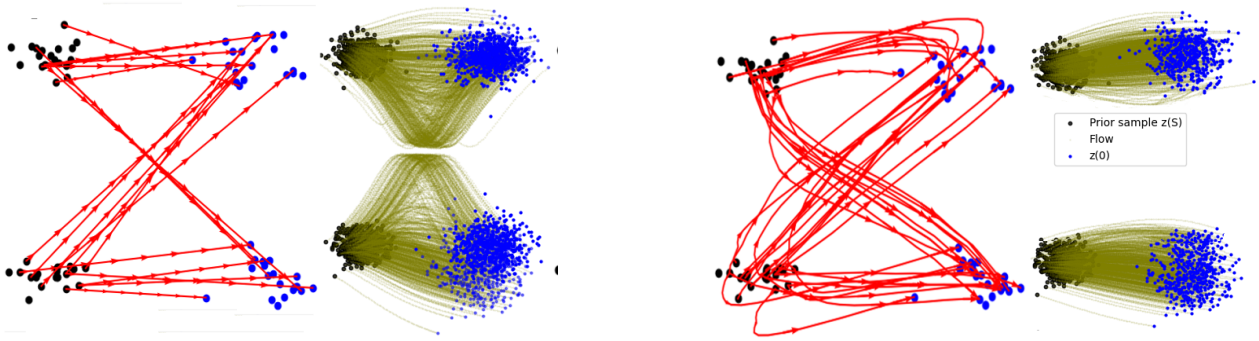
Contributions (a) We propose a new bi-level formulation to learn straight flows that explicitly forces straightness by learning interpolants, (b) We derive an analytic form of target vector field that allows training interpolants without differentiable optimization methods, (c) We present scalable parametric models for conditional paths that enable efficient training, (d) We show significant improvements for low-shot (i.e., low number of function evaluations) generation quality.

2. Preliminaries

Let p_0 and p_1 be two distributions on $\mathbb{R}^d$. Usually, p_1 is our target distribution, only known through samples, and p_0 is the source distribution, usually chosen to be tractable (e.g., a Gaussian). Our goal is to generate samples from p_1. One way to generate these samples is by transporting the initial

[1]University of Massachusetts [2]NVIDIA. Correspondence to: Shiv Shankar <>.

Submitted to the 40^{th} International Conference on Machine Learning, 2025.



FM with Linear Interpolants

FM with Learned Interpolants (ours)

Figure 1: Illustration of training difference flow matching (FM) [30] and our method on an illustrative 2 gaussian (source, black) to 2-gaussian (target, blue) problem. On the left we have standard FM with linear interpolants (red lines) and the resulting flow vector. On the right we show our approach. By allowing non-linear interpolants we are able to learn a flow velocity field with significantly less curvature.

distribution p_0 via a vector field to the target p_1. Specifically, consider a vector field $u(t,x) : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ such that the differential equation

$$\frac{d\gamma_x(t)}{dt} = u(t, \gamma_x(t)), \qquad \gamma_x(0) \sim p_0 \tag{1}$$

produces sample from p_1. Under some topological constraints on the distributions, the existence of such fields is well known [3]. For a given starting sample x, we denote the solution of Equation (1) as $\gamma_x(t)$. This solution, also called a flow, is the trajectory of point x as it evolves under u. We denote by p_t the probability distribution obtained at time t by moving the samples of p_0 via the field u. Note that for a given source and target distributions, multiple vector fields which satisfy the above equation exist.

Flow Matching

Flow matching (FM) [30] is a simulation-free method for learning one such vector field using a neural network $v_\theta(t,x)$. It does that by optimizing the FM objective

$$\mathcal{L}_{FM} = E_{t,p_t} ||v_\theta(t,x_t) - u(t,x_t)||_2^2. \tag{2}$$

Unfortunately, this objective is not tractable given only the source and target distributions, as the target vector field $u(t,x_t)$ is unknown. Lipman et al. [30] show that one can optimize the conditional flow matching objective instead

$$\mathcal{L}_{CFM} = E_{t,q(z)} E_{p_t(x_t|z)} ||v_\theta(t,x_t) - u(t,x_t|z)||_2^2, \tag{3}$$

where $u(t,x_t|x_1)$ is a conditional vector field and corresponding probability path $p_t(x_t|z)$. Lipman et al. [30] considered z to be a sample from the target x_1. While Lipman et al. [30] considered a family linear Gaussian paths, more generalized variants of this problem have been proposed [48]. Any suitable conditioning variable z can be chosen if the objective remains tractable [40, 48]. A related problem is that of the Schrodinger Bridge [11, 50], which seeks the vector field whose probability law is close to that of the standard Brownian diffusion process.

3. Learning Straight Flows by Tuning Interpolants

In this section we describe our method to learn straight flows. We start with the standard CFM objective, but introduce parametric interpolants (parameterized by ϕ), instead of the linear ones often used in flow matching. Once we do so, the learned flow model v_θ becomes dependent on the interpolant parameters ϕ. We thus formulate the problem of learning straight flows as a bi-level optimization problem, where the interpolants are tuned to optimize the straightness of v_θ (Section 3.1). Next, we show how one can solve this bilevel optimization without using differentiable optimization methods. For this purpose, we derive an analytic expression for the flow field in terms of the parameters ϕ (Section 3.2) together with a measure of the flow's straightness (Section 3.3). Finally, we describe a specific family of interpolants which enables scalable computation with high-dimensional datasets (Section 3.5).

3.1. Bi-Level Formulation for Straight Flows

We consider the commonly used CFM objective with $z = (x_0, x_1)$ [48]. While CFM models often use simple linear interpolants, the consistency of the CFM approach only requires smooth enough conditional fields. Crucially the linear interpolant x_t can be replaced by any other interpolant (stochastic or otherwise), as long as the conditional u (Equa-

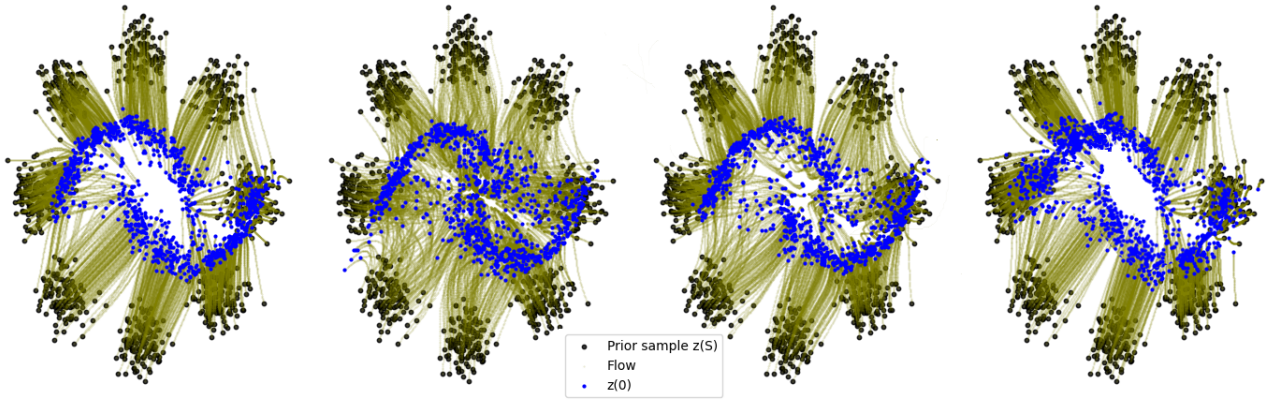


Figure 2: From left to right, the figure depicts flows learned by (1) OT Coupling, (2) Independent Coupling, (3) Sinkhorn Coupling and (4) Our method. For this example the models were trained to transform 8-gaussian (source) to spiral moons (target). We can see that using OT related coupling learns straighter fields. We also see that by using learnt interpolants, our approach learns straight fields despite using independent coupling .

tion (3)) is modified to be the corresponding tangent vector. Specifically, choosing the interpolant to be $x_t = \phi_{t,x_1}(x_0)$ we can rewrite the CFM objective from Equation (3) as

$$E_{t,p_0(x_0),p_1(x_1)}||v_\theta(t,x_t) - \partial_t \phi_{t,x_1}(x_0)||_2^2, \tag{4}$$

which recovers standard CFM for $\phi := tx_1 + (1-t)x_0$.

We propose learning straight flows by choosing ϕ such that the resulting vector field v is as straight as possible. We formulate this as a bi-level optimization problem

$$\min_{\phi} |v_{\theta^*}(.;\phi)|_{\text{Straight}} \qquad \text{s.t.} \tag{5}$$
$$v_{\theta^*} = \arg\min_{\theta} E_{t,p_0(x_0),p_1(x_1)}||v_\theta(t,x_t;\phi) - \partial_t \phi_{t,x_1}(x_0)||_2^2,$$

where $|\cdot|_{\text{Straight}}$ is a measure of straightness of the vector field. As an example, previous work by Pooladian et al. [40], Liu et al. [31] suggest measuring curvature using

$$|v|_{\text{Straight}} = \mathbb{E}_{\substack{t\sim U(0,1)\\ x_0\sim p_0}}\left[||v_t(\gamma_{x_0}(t))||^2 - ||\gamma_{x_0}(1) - x_0||^2\right],$$

where γ is the flow, that is, the solution to Equation 1.

Remark. *Note that v_θ is a function of θ, but also has an implicit dependence on ϕ. Due to the inner optimization, the optimal parameter θ^* depends on ϕ. We made this dependence explicit in Equation* (5)*.*

While mathematically straightforward, solving the optimization problem from Equation (5) is computationally intensive. This is a bi-level problem, where each inner objective requires solving a FM objective. Our insight is that one is not required to solve the FM objective to get v_{θ^*}, and can instead leverage the optimal vector field $v_{\theta*}$ in a different manner directly in terms of the interpolant functions ϕ.

3.2. Reformulating Flow Matching

This section shows how to connect v_θ to the interpolant function ϕ without solving the inner optimization iteratively, by presenting an analytic form for the optimal v_θ^* (Equation (6)) in terms of ϕ. We will then use this result in Section 3.3 and Section 3.5 to derive our final optimization algorithm.

Proposition 1. *Let $\mathcal{J}$ be the determinant of the Jacobian of the interpolant's inverse, i.e., $\left|\frac{d\phi^{-1}_{t,x_1}(x)}{dx}\right|_\Delta$. The optimal velocity field for an interpolant ϕ is given by*

$$v^*_\phi(t,x_t) = \int \frac{\partial_t \phi_{t,x_1}(\phi^{-1}_{t,x_1}(x_t)) p_0(\phi^{-1}_{t,x_1}(x_t))|\mathcal{J}|p_1(x_1)dx_1}{\int p_0(\phi^{-1}_{t,x_1}(x_t))|\mathcal{J}|p_1(x_1)dx_1}. \tag{6}$$

We include a derivation of Equation (6) *in Section 3.2.1.*

Remark. *A more compact version of Equation* (6) *for the standard linear interpolant $x_0 = \frac{x_t - t x_1}{1-t}$ has been presented earlier by Tong et al. [48]. However, since the linear interpolant has no free parameters, it is not suitable for learning.*

Specific simple parameterisations, like paths of the form $x_0 = \frac{x_t - g(t)x_1}{1-h(t)}$ with the coefficients g, h being non-linear in t, yield efficient versions of Equation (6). For such paths, Equation (6) simplifies as the Jacobians do not depend on x_1. However, such parameterisations, while simple to use, may not be powerful enough for complex datasets.

While exact, Equation (6) is intractable, as it involves complex integrals. In practice, our final algorithm will leverage empirical estimates for this quantity using samples from $p(x_1)$, and flexible interpolants ϕ parameterized by neural networks, as explained in Section 3.5.

3.2.1. Derivation of Equation (6)

A key challenge in analyzing the FM objective is that both terms involved in the $L2$ loss (Equation (4)) depend on x_0 and x_1. This can be addressed by writing out the distribution over x_t explicitly

$$E_{t,p_0(x_0),p_1(x_1)}||v_\theta(t,x_t) - \partial_t \phi_{t,x_1}(x_0)||_2^2 \tag{7}$$
$$= E_{t,p(x_0,x_1)}||v_\theta(t,x_t) - \partial_t \phi_{t,x_1}(x_0)||_2^2 \tag{8}$$
$$= E_{t,p_t(x_t),p(x_0,x_1|x_t,t)}||v_\theta(t,x_t) - \partial_t \phi_{t,x_1}(x_0)||_2^2, \tag{9}$$

where it can be observed that the global minimizer is

$$v^*(t,x_t) = \mathbb{E}_{p(x_0,x_1|x_t,t)}[\partial_t \phi_t(x_0,x_1)]. \tag{10}$$

This expression can also be directly obtained from the generalized CFM objective [40], using $z = x_t$.

While simple, Equation (10) cannot be used in practice since we do not have access to the required distributions. For instance, x_1 is only available via samples, and the posterior $p(x_0,x_1|x_t)$ is intractable. This can be addressed by some analysis and a few judicious choices conditional paths. First we write the conditional distribution as:

$$p(x_0,x_1|x_t,t) = \frac{p(x_0,x_1,x_t|t)}{p(x_t|t)} = \frac{p(x_0,x_1,x_t|t)}{\int p(x_0,x_1,x_t|t)dx_0dx_1}. \tag{11}$$

Specifically, using $x_t = \phi_{t,x_1}(x_0)$, Equation (11) becomes

$$p(x_0,x_1|x_t,t) = \frac{p(x_0,x_1,\phi_{t,x_1}(x_0)|t)}{\int p(x_0,x_1,\phi_{t,x_1}(x_0)|t)dx_0dx_1}. \tag{12}$$

Since the maps are deterministic, we get

$$p(x_0,x_1 \mid x_t,t) = \frac{\delta(\tilde{x}_t - x_t)\, p_0(x_0)\, p_1(x_1)}{p(x_t|t)}, \tag{13}$$

where $\tilde{x}_t = \phi_{t,x_1}(x_0)$ is an auxiliary variable introduced to distinguish the conditioning variable from the conditioned value, δ is the Dirac function [12], and $p(x_t)$ is the marginal likelihood (normalization constant), given by

$$p(x_t|t) = \int p(x_0,x_1,x_t|t)dx_0dx_1 \tag{14}$$
$$= \int \delta(\tilde{x}_t - x_t)\, p_0(x_0)\, p_1(x_1)\, dx_0\, dx_1. \tag{15}$$

While the above integral can be estimated with infinite data, the expression is not directly practically useful, since the estimator will often just be zero when relying on empirical samples (due to the Dirac function). However, a useful expression can be obtained noting that the Dirac delta function allows us to integrate over x_0 using the change in variables formula [18], yielding

$$p(x_t|x_1,t) = p_0(\phi_{t,x_1}^{-1}(x_t))\mathcal{J}, \tag{16}$$

where $\mathcal{J}$ is the determinant of the Jacobian of the interpolant's inverse, i.e., $\left|\frac{d\phi_{t,x_1}^{-1}(x)}{dx}\right|_\Delta$. Substituting this in Equation (15) and using the definition of v^* (the expectation of $\partial_t \phi_{t,x_1}$ under $p(x_0,x_1|x_t)$) yields Equation (6).

3.3. Straightening the Flow

Equation (6) yields powerful way to manipulate the target velocity field v^* to enforce desired properties on the flow. When we want the flows to be straight, the vector field v^* should be constant everywhere on a trajectory. This implies that a straight velocity field u_{str} must satisfy

$$u_{\text{str}}(t,\gamma_{x_0}(t)) = u_{\text{str}}(s,\gamma_{x_0}(s)) = u_{\text{str}}(0,\gamma_{x_0}(0))\ \forall s,t,$$

where γ_{x_0} is the flow generated by u_{str} from point x_0. Since the vector field is constant, we can differentiate the above expression with respect to t to get

$$\partial_t u_{\text{str}} = -\nabla_x u_{\text{str}} \cdot u_{\text{str}} \tag{17}$$

for all points on the trajectory, where $\cdot$ refers to the matrix-vector product. Therefore, since we know the optimal field is given by v^*, we can learn straight flows by optimizing

$$\phi^* = \arg\min_\phi ||\partial_t v_\phi^* + \nabla_x v_\phi^* \cdot v_\phi^*||^2. \tag{18}$$

We note here that this criteria for straight flows has been observed before Liu et al. [31], Yang et al. [57].

While the above expression is correct, optimizing it is computationally intensive, as we need to differentiate through v_ϕ^*, which in turns requires the Jacobian of ϕ_{t,x_1}^{-1}. However, using a flexible neural network for v_θ, under ideal training we expect $v_\theta \approx v_\phi^*$. Therefore, we propose to replace the derivatives of v^* with derivatives of v_θ, which yields

$$\min_{\theta,\phi} \mathbb{E}_t \mathbb{E}_{x_0,x_1}||v_\theta(x_t,t) - \text{sg}(v^*(x_t))||^2 + \lambda||\partial_t v_\theta(x_t',t) + \nabla_x v_\theta(x_t',t)\cdot v^*(x_t',t)||^2. \tag{19}$$

Here "sg" refers to stop gradient and $\lambda > 0$ is a hyperparameter. We suppress the dependence of v^* on ϕ for notational convenience.

Remark. *Technically, Equation 17 holds only for points x_t on the trajectory obtained using v_ϕ^*. Hence, ideally, one should compute this loss only via sampling from v_ϕ^*. However, if the source and target are diffused enough, then such trajectories will likely be spread around and one-step approximations for $x_t' = x_0 + t\, v(x_0,t)$ can be used.*

3.4. Segmented Objective

One issue with the aforementioned approach is that a chosen parameterization of ϕ may not be powerful enough to

ensure straight flows. An alternative is to ensure piecewise straightness. Since even the standard curved flow under linear interpolants can be well approximated if the number of pieces is high enough, this can address the situation when ϕ is difficult to learn. Forcing piecewise straightness also provides a natural way to balance the computational resource vs straightness, and has been explored by Yang et al. [57]. We follow their approach for enforcing piecewise straightness. The time interval $[0, 1]$ is divided into K uniform pieces, and straightness is only enforced between points in the same segment. This objective has the additional advantage that unlike Equation (17) this can be enforced at a more general x_t.

$$||f_\phi(t, x_t) - f_\phi(t + \Delta t, x_{t+\Delta t}||^2 \tag{20}$$

where $f_\phi(t, x_t) = x_t + (\frac{i+1}{K} - t)v^*_\phi(t, x_t)$ and both $t, t+\Delta t$ belong to the same segment $[i/K, (i + 1)/K]$. We also perform experiments by replacing the regularizer in Equation (19) with Equation (20).

3.5. Scalable Computation

While Equation (19) is 'analytical', in general, it has poor scaling in the dimensionality of the output, specifically due to the presence of the inverse-derivative as well as the Jacobian, both of which do not generally scale with the dimension of the data. Additionally, we need to ensure that the function ϕ_{t,x_1} remains invertible. Fortunately, researchers have developed models that are amenable to these considerations. Specifically, the literature on normalising flows and invertible models [37] has proposed several families of expressive neural networks that support such operations. In this work we use GLOW/1x1 convolution model [27] to parameterise the function ϕ. This family of neural networks learns any linear transform's parameters in the "PLU" format. It initializes any matrix parameter W of a linear layer, finds its PLU decomposition, fixes P and optimizes the lower and upper diagonal matrices L and U. Normalizations and activations are also chosen in a way that ensures easy inversion and memory efficiency.

We follow a similar approach but modify the parameterization further. Specifically, we require the diagonal of the matrix U to be non-zero. This is because the Jacobian determinant of the corresponding transform is directly dependent on the diagonal product of U. As these terms appear in the denominator of v^* in Equation 6, we require that the determinants should remain far from zero for stability.

The expression for v^* from Equation (6) computes the inverse function ϕ^{-1} for the same x_t and varying x_1. Thus, it is more natural to use neural networks to directly parameterize the inverse function ϕ^{-1}, compared to ϕ. However we still need an efficient inverse function. GLOW models form a natural family of such functions. In our experiments, we do the same using the GLOW model to map from x_t to x_0.

Boundary Conditions Besides invertibility and efficient Jacobian computation, the interpolant ϕ must satisfy certain boundary conditions: (i) $\phi^{-1}_{t',x_1}(x_t) \to x_t$ for $t' \to t$, and (ii) $\phi^{-1}_{1,x_1}(x_t) = x_1$. To ensure both of these, we parameterize the model as $F((t' - t)x_t + (1 - t')x_1)$ where F is the aforementioned GLOW model, and we choose the model weights such that the function becomes the identity at $t' = t$ and $t' = 1$. We achieve this using the Bernstein polynomials $b_{n,k}$ for $k = 1 \cdots n - 1$. Specifically, we parameterize the L matrix as $L = I + \sum_{k=1}^{n-1} b_{n,k}(\frac{t'-t}{1-t})L_k$, where L_k are a set of trainable lower diagonal matrices, and k is a hyperparameter specifying how many transformations are allowed. The weight matrices U also follow this parameterization.

Computing Expectations The expression for v^* from Equation (6) requires computing expectations (or integrals) over the data distribution $p(x_1)$. Since ϕ is given by the neural network and p_0 is Gaussian (or a similar tractable distribution), the value of $p_0(\phi_{t,x_1}(x_t))$ can be computed exactly. Therefore, one can estimate both the numerator and the denominator as empirical expectations over the target distribution. While the numerator of v^* admits unbiased estimation through this approach, using an empirical estimate in the denominator introduces bias. However, the resulting estimator is consistent, with the bias going to zero as we aggregate more data samples. Additionally, this estimation process, which combines vector fields from multiple paths at the same time, often produces a lower variance estimate of the optimal v^*. To see this, note that the all the terms in Equation (6) except $\partial_t(\phi)$ are related to probabilities (p_0, p_1 or the Jacobian). As such when replaced by empirical expectation, this can be seen as a form of weighted importance sampling, which often has lower variance [47, 17], thus leading to more efficient optimization. Such estimators often work well and are common in both probabilistic inference [47] and reinforcement learning [41]. In fact, Xu et al. [54] used this idea to reduce the variance of the gradient estimates used to train diffusion models, observing that it often led to more robust optimization, thanks to the variance reduction, despite the bias introduced.

In our framework, a balance between low bias and low variance can be achieved by interpolating v^*_ϕ with the original conditional target field $\partial_t\phi$ in the FM objective. Additionally, the bias introduced by using an empirical estimate of v^*_ϕ to learn v_θ can be countered by using the conditional vector $\partial_t\phi$ instead of using v^*_ϕ in the first term of the FM loss (Equation (19)). In such a case the bias only has an indirect influence on v_θ, as it drives the optimization of ϕ . Consequently, this bias predominantly affects the straightness metric rather than the core FM objective. However this may slow down optimization due to the higher variance

[54]. In our experiments we use v^*_ϕ, and leave managing the bias-variance tradeoff for future work.

4. Related Work

FM has been proven to work with arbitrary couplings between the source and target distributions [40, 48]. Multisample FM [40] proposes to generalize the independent coupling of the data distribution $p_1(x_1)$ and prior distribution $p_0(x_0)$ to the optimal transport coupling plan $\pi(x_0, x_1)$. Under the optimal transport plan, the learned trajectory of ODE are straight [40]. However, this requires constructing the optimal transport plan for the data which is computationally prohibitive. Minibatch OT and Minibatch Sinkhorn coupling have been suggested to lower the cost of computing such couplings [48]. Minibatch OT can ensure straight fields asymptotically in the limit of full data optimization.

Liu et al. [31] suggested a rectified flow matching method which uses a pretrained FM to learn a straighter approximation. Based on this insight, other methods to learn straight flows have also been proposed [31, 32, 29]. These methods however are not simulation-free (requiring sampling during training) and often require iteratively distilling from flows, both of which are computationally intensive.

Non-Linear Interpolants Rectified and distilled flow methods [29, 31] rely on the coupling given by a "teacher" (pre-trained) flow. These methods often rely on linear interpolants, and require multiple training stages, where each stage requires generating training pairs of noise and data samples by simulating the ODE with the vector field learnt in the previous stage. Our method, on the other hand, is simulation-free and uses the independent coupling, though with learned interpolants. Kapuśniak et al. [21] also proposed using non-linear interpolants. They learn an interpolant such that the conditional paths stay close to a manifold, but do not directly optimize any property of v^*. Another recent work using non-linear interpolant is Davis et al. [10] to improve generation of discrete data.

Recently, Bartosh et al. [4] proposed learning the forward process in diffusion models, akin to learning the interpolant in flow matching, and also rely on interpolants parameterized via neural networks and adaptations of normalizing flows. While closely related to our method, the approach proposed in prior work differs in its formulation for learning the interpolants. Specifically, they employ a specific parameterization for both the interpolant ϕ and the flow model v_θ, which share parameters. This design choice, can limit the flexibility of the model. For instance, as illustrated in Appendix C, there exist certain interpolants ϕ for which, under their paramaterization, no θ can yield the optimal field v^*. Additionally, since they do not use a bi-level objective, the optimization procedure does not account for the relationship between the optimal θ and optimal ϕ. In other words, the dependence of θ on ϕ is ignored. Our method does not have these restrictions. Empirically, we observe that our approach achieves improved performance (Section 5), highlighting the benefits of explicitly modeling this relationship.

Other Related Works Xu et al. [54] proposed a self-normalized estimator like the one we use derived from Equation (6) in the context of diffusion models to reduce the variance of the gradient estimates used during training; a problem which had also been identified by Karras et al. [24]. Yang et al. [57] proposed regularizing with a loss similar to Equation (18), except they consider the paths taken with respect to the conditional vector fields. However, this family of methods can only learn approximate velocity fields, which can be seen from the analytic form of v^* (Equation (6)). If one takes a linear interpolant (so the Jacobian determinant $\mathcal{J} = 1$ and ϕ_{t,x_1} is linear), the optimal field v^* is exactly determined and does not have any free variables to adjust. As such, if one fixes both the coupling function and the interpolant, the underlying ground truth velocity field is fully determined. If this field is not straight, any objective which tries to straighten the field sacrifices accuracy.

Consistency Models This family of models [57, 55, 46] aims to learn a function that directly solves Equation (1). One can choose to solve the ODE numerically and learn the function using a distilled dataset consisting of pairs of noise and the corresponding data-samples, though more common and scalable approaches use a simulation free approach. We solve the problem of learning straight fields, which is orthogonal to the problem that consistency models aim to solve, which involves learning a function that correctly integrates the vector field. In principle, these ideas could be combined to produce more efficient sampling methods.

5. Experiments

5.1. Toy Datasets

We first consider three small toy datasets to evaluate our method in visually illustrative scenarios. We compare our method against FM trained on the optimal transport map [40], the Sinkhorn coupling [48], which attempts to approximate the OT solution, and standard FM [30]. For these we only considered the objective in Equation (19) and not the segmented one. In Figure 2 we present an example of generating spirals (target) from 8 Gaussians (source). We can see that training with the optimal transport map, as expected, produces straight vector fields. We can also see that the Sinkhorn coupling yields reasonably straight fields, while the standard FM approach produces significantly more curved paths. We also see that our approach produces very straight paths, similar to the ones obtained using the opti-

Algorithm 1 Vector field model training algorithm

Require: Sampler for p_0 (usually Gaussian), Empirical samples from p_1, batch size N, averaging size $M(\geq N)$ (to estimate Equation (6)); models $v_\theta(x,t)$ and ϕ

while not converged **do**
  Sample N points $\{t^i\}_{i=1}^N$ from $\mathcal{U}[0,1]$
  Sample N pairs $\{x_0^i, x_1^i\}_{i=1}^n$ from $p(x_0,x_1) = p_0(x_0)p_1(x_1)$
  Sample $M-N$ points $\{\hat{x}_1^j\}_{i=N}^{M-N}$ from p_1
  Compute N interpolants x_t^i from (t^i, x_0^i, x_1^i)
  Estimate $v^*(x_t^i)$ (Eq. 6) replacing integrals $\int f(x_t,x_1,t)p_1(x_1)dx_1$ by empirical estimates $\frac{1}{M}\sum_{j=1}^M f(x_t^i, x_1^j, t^i)$
  Calculate the empirical loss using Equation (19)
  Update parameters θ, ϕ (e.g., using SGD or Adam [26])
end while

CIFAR-10 ImageNet 32x32 CelebA 256

Figure 3: Generated samples from CIFAR, ImageNet and CelebA.

mal coupling. We show an additional example on another synthetic dataset, transforming a Gaussian (source) to the 4-square target, in Appendix B, with similar conclusions.

In Figure 1 we show the learnt conditional paths (i.e., interpolants) for a simple 2-dimensional dataset with both the source and target being a mixture of 2 gaussians. Regular FM uses linear interpolants (shown in red, left plot Figure 1), which leads to a curved velocity field (second plot in Figure 1). In the same figure we can observe that our method, by learning curved interpolants (third plot in Figure 1), yields a velocity field with very little curvature. Intuitively, this is achieved thanks to the curved interpolants, with the curvature canceling out once the curved interpolants are mixed to get the final velocity field.

5.2. Real Data

In this section we present experimental results on image generation [1]. We conduct two sets of experiments: one on low-resolution datasets, including CIFAR-10 [2] and ImageNet 32x32 [8], and another on higher-resolution datasets, namely CelebA-HQ [22] and AFHQ-Cat [7]. Following the methodology of Song et al. [46], we evaluate the model across varying numbers of function evaluations (NFE). The flow field is learned using a U-Net architecture based on DDPM++ [45]. To assess the quality of generated images, we employ the Fréchet Inception Distance (FID) score [19].

Remark. *We use an annealing schedule for the regularization term in Equation* (19) *for non-toy datasets. The straightness penalty, as we use in this work, is reasonable once v_θ is roughly aligned with v^*. For non-toy datasets, it may take many training iterations before a good vector field v is learnt. We thus use a multi-stage approach where we set $\lambda = 0$ in the first stage, and anneal λ slowly over time. Furthermore, we observed that learning ϕ too fast sometimes led to unstable optimization, as the conditional paths are more complex than v^*. We therefore use a smaller learning rate for ϕ (5e-6) than for θ (1e-4).*

Baselines We follow Song et al. [46], Yang et al. [57] and compare our method against several baselines comprising of representative diffusion models and flow models, as well as the recent approaches that focus on learning straight vector fields. The baseline models include Consistency

[1] Our code is based upon the code provided by Yang et al. [57]

Models [46], Rectified Flow [31], Rectified Flow with Bellman Sampling [35], Neural Flow Diffusion Models [4], and Consistency-FM [57]. We did not run these baselines ourselves and have reported results from literature. Since not all earlier works have reported results on all the datasets, for each dataset the set of baselines is not always identical.

5.2.1. RESULTS

The results for the CIFAR dataset are presented in Table 1. We also report the Inception-Score (IS) [43] comparison with models that have reported IS scores in Appendix B. Our method demonstrates superior performance compared to models such as Consistency FM [57], Rectified Flow [31], and Consistency Model [46]. We also see that our model matches or outperforms mainstream diffusion models while using a low number of function evaluations (NFE).

Table 3 shows results on the ImageNet 32x32 dataset, where we compare against plain flow matching [30] (with a large number of NFEs), multisample flow matching [40], and Neural Flow Diffusion models [4]. As for the other datasets, we observe that our approach yields better performance than competing approaches using a low number of NFEs, and that using 12 steps during generation it outperforms plain flow matching with 120 steps, while remaining highly competitive when using only 4 steps.

Table 1: Comparison with baseline models on CIFAR-10. Results for other models are obtained from previous work.

Method	NFE (↓)	FID (↓)
Score SDE [45]	2000	2.20
DDPM [20]	1000	3.17
LSGM [49]	147	2.10
PFGM [53]	110	2.35
EDM [24]	35	**2.04**
1-Rectified Flow /ReFlow[31]	1	378
Glow [27]	1	48.9
Residual Flow [6]	1	46.4
GLFlow [51]	1	44.6
DenseFlow [14]	1	34.9
Consistency Model [46]	2	5.83
Consistency Flow Matching [57]	2	5.34
Ours	2	**4.61**
Ours (segmented)	2	**4.19**

We further evaluate our method on high-resolution image generation tasks, specifically 256×256 images from AFHQ-Cat and CelebA-HQ. Following Yang et al. [57], we compare against baseline methods, including Consistency FM [57], ReFlow [31], and ReFlow with Bellman sampling [35]. Results from CelebA are summarized in Table 2 while AFHQ can be found in Appendix B. All baseline results are taken from Yang et al. [57].

Our method outperforms baseline approaches such as Rectified Flow [31] and Rectified Flow with Bellman sampling [35] by a significant margin, even with the same number of function evaluations (NFEs). Additionally, compared to its performance on CIFAR-10, our method demonstrates even greater improvements in generating high-resolution images.

Table 2: Comparison with flow matching models on CelebA.

Method	NFE (↓)	FID (↓)
ReFlow [31]	8	109.4
	6	127.0
ReFlow + Bellman Sampling [35]	8	49.8
	6	72.5
Consistency Flow Matching [57]	6	36.4
Ours	6	**28.6**
Ours (segmented)	6	**24.2**

Table 3: Results on ImageNet-32x32.

Method	NFE (↓)	FID (↓)
Flow Matching [30]	120	5.0
MultiSample FM [40]	4	17.3
	12	7.2
NFDM [4]	4	6.1
	12	4.1
Ours	2	8.32
	4	5.58
	12	3.84
Ours (segmented)	2	7.32
	4	5.41
	12	3.56

6. Conclusion

We propose a novel approach to learn flow matching vector fields that; unlike existing methods, which try to learn straight approximations to a curved vector field; learns a straight vector field directly. We use non-linear interpolants in the CFM objective and show how one can optimize the corresponding vector field solutions. In the process, we provide analytical expressions for the general solution to a CFM model and show how it can be tuned to adjust the "straightness" of the vector field. We present a way to parametrize the interpolants using a GLOW model, allowing fast inversion and determinant computations. Our approach outperforms recent methods when using a low number of NFEs.

7. Impact Statement

This paper presents work whose goal is to advance the field of machine learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

[1] Albergo, M. S. and Vanden-Eijnden, E. Building normalizing flows with stochastic interpolants. In *The Eleventh International Conference on Learning Representations*, 2022.

[2] Alex, K. Learning multiple layers of features from tiny images. *https://www. cs. toronto. edu/kriz/learning-features-2009-TR. pdf*, 2009.

[3] Ambrosio, L. and Crippa, G. Continuity equations and ode flows with non-smooth velocity. *Proceedings of the Royal Society of Edinburgh Section A: Mathematics*, 144(6):1191–1244, 2014.

[4] Bartosh, G., Vetrov, D., and Naesseth, C. A. Neural flow diffusion models: Learnable forward process for improved diffusion modelling. *arXiv preprint arXiv:2404.12940*, 2024.

[5] Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[6] Chen, R. T., Behrmann, J., Duvenaud, D. K., and Jacobsen, J.-H. Residual flows for invertible generative modeling. *Advances in Neural Information Processing Systems*, 32, 2019.

[7] Choi, Y., Uh, Y., Yoo, J., and Ha, J.-W. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8188–8197, 2020.

[8] Chrabaszcz, P., Loshchilov, I., and Hutter, F. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017.

[9] Daras, G., Delbracio, M., Talebi, H., Dimakis, A. G., and Milanfar, P. Soft diffusion: Score matching for general corruptions. *arXiv preprint arXiv:2209.05442*, 2022.

[10] Davis, O., Kessler, S., Petrache, M., Ceylan, I. I., Bronstein, M., and Bose, A. J. Fisher flow matching for generative modeling over discrete data. *arXiv preprint arXiv:2405.14664*, 2024.

[11] De Bortoli, V., Thornton, J., Heng, J., and Doucet, A. Diffusion schrödinger bridge with applications to score-based generative modeling. *Advances in Neural Information Processing Systems*, 34:17695–17709, 2021.

[12] Dirac, P. A. M. *The principles of quantum mechanics*. Number 27. Oxford university press, 1981.

[13] Firoozi, R., Tucker, J., Tian, S., Majumdar, A., Sun, J., Liu, W., Zhu, Y., Song, S., Kapoor, A., Hausman, K., et al. Foundation models in robotics: Applications, challenges, and the future. *The International Journal of Robotics Research*, pp. 02783649241281508, 2023.

[14] Grcić, M., Grubišić, I., and Šegvić, S. Densely connected normalizing flows. *Advances in Neural Information Processing Systems*, 34:23968–23982, 2021.

[15] Gu, J., Zhai, S., Zhang, Y., Bautista, M. A., and Susskind, J. f-dm: A multi-stage diffusion model via progressive signal transformation. *arXiv preprint arXiv:2210.04955*, 2022.

[16] Guo, Z., Liu, J., Wang, Y., Chen, M., Wang, D., Xu, D., and Cheng, J. Diffusion models in bioinformatics and computational biology. *Nature reviews bioengineering*, 2(2):136–154, 2024.

[17] Hájek, J. Comment on "an essay on the logical foundations of survey sampling, part one". *The foundations of survey sampling*, 236, 1971.

[18] Halperin, I. and Schwartz, L. *Introduction to the Theory of Distributions*. University of Toronto Press, 1952.

[19] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[20] Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In *NeurIPS*, volume 33, pp. 6840–6851, 2020.

[21] Kapuśniak, K., Potaptchik, P., Reu, T., Zhang, L., Tong, A., Bronstein, M., Bose, A. J., and Di Giovanni, F. Metric flow matching for smooth interpolations on the data manifold. *arXiv preprint arXiv:2405.14780*, 2024.

[22] Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[23] Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. Analyzing and improving the image quality of stylegan. In *CVPR*, pp. 8110–8119, 2020.

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

[25] Kim, D., Na, B., Kwon, S. J., Lee, D., Kang, W., and Moon, I.-C. Maximum likelihood training of implicit

nonlinear diffusion model. *Advances in neural information processing systems*, 35:32270–32284, 2022.

[26] Kingma, D. P. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[27] Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

[28] Kornilov, N., Gasnikov, A., and Korotin, A. Optimal flow matching: Learning straight trajectories in just one step. *arXiv preprint arXiv:2403.13117*, 2024.

[29] Lee, S., Kim, B., and Ye, J. C. Minimizing trajectory curvature of ode-based generative models. In *International Conference on Machine Learning*, pp. 18957–18973. PMLR, 2023.

[30] Lipman, Y., Chen, R. T., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2022.

[31] Liu, X., Gong, C., et al. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2022.

[32] Liu, X., Zhang, X., Ma, J., Peng, J., and Liu, Q. Instaflow: One step is enough for high-quality diffusion-based text-to-image generation. *arXiv preprint arXiv:2309.06380*, 2023.

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

[34] Nachmani, E., Roman, R. S., and Wolf, L. Non gaussian denoising diffusion models. *arXiv preprint arXiv:2106.07582*, 2021.

[35] Nguyen, B., Nguyen, B., and Nguyen, V. A. Bellman optimal stepsize straightening of flow-matching models. In *The Twelfth International Conference on Learning Representations*, 2024.

[36] Nielsen, B. M., Christensen, A., Dittadi, A., and Winther, O. Diffenc: Variational diffusion with a learned encoder. *arXiv preprint arXiv:2310.19789*, 2023.

[37] Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

[38] Peluchetti, S. Non-denoising forward-time diffusions, 2022. URL `https://openreview.net/forum?id=oVfIKuhqfC`.

[39] Phung, H., Dao, Q., and Tran, A. Wavelet diffusion models are fast and scalable image generators. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10199–10208, 2023.

[40] Pooladian, A.-A., Ben-Hamu, H., Domingo-Enrich, C., Amos, B., Lipman, Y., and Chen, R. T. Multisample flow matching: Straightening flows with minibatch couplings. 2023.

[41] Precup, D. Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, pp. 80, 2000.

[42] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *CVPR*, pp. 10684–10695, 2022.

[43] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

[44] Shaul, N., Perez, J., Chen, R. T., Thabet, A., Pumarola, A., and Lipman, Y. Bespoke solvers for generative flow models. *arXiv preprint arXiv:2310.19075*, 2023.

[45] Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2020.

[46] Song, Y., Dhariwal, P., Chen, M., and Sutskever, I. Consistency models. In *International Conference on Machine Learning*, pp. 32211–32252. PMLR, 2023.

[47] Tokdar, S. T. and Kass, R. E. Importance sampling: a review. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(1):54–60, 2010.

[48] Tong, A., Malkin, N., Huguet, G., Zhang, Y., Rector-Brooks, J., FATRAS, K., Wolf, G., and Bengio, Y. Improving and generalizing flow-based generative models with minibatch optimal transport. In *ICML Workshop on New Frontiers in Learning, Control, and Dynamical Systems*, 2023.

[49] Vahdat, A., Kreis, K., and Kautz, J. Score-based generative modeling in latent space. *Advances in neural information processing systems*, 34:11287–11302, 2021.

[50] Wang, G., Jiao, Y., Xu, Q., Wang, Y., and Yang, C. Deep generative learning via schrödinger bridge. In *International conference on machine learning*, pp. 10794–10804. PMLR, 2021.

[51] Xiao, Z., Yan, Q., and Amit, Y. Generative latent flow. *arXiv preprint arXiv:1905.10485*, 2019.

[52] Xing, Z., Feng, Q., Chen, H., Dai, Q., Hu, H., Xu, H., Wu, Z., and Jiang, Y.-G. A survey on video diffusion models. *ACM Computing Surveys*, 2023.

[53] Xu, Y., Liu, Z., Tegmark, M., and Jaakkola, T. Poisson flow generative models. *Advances in Neural Information Processing Systems*, 35:16782–16795, 2022.

[54] Xu, Y., Tong, S., and Jaakkola, T. Stable target field for reduced variance score estimation in diffusion models. *arXiv preprint arXiv:2302.00670*, 2023.

[55] Yang, L., Liu, J., Hong, S., Zhang, Z., Huang, Z., Cai, Z., Zhang, W., and Bin, C. Improving diffusion-based image synthesis with context prediction. In *NeurIPS*, 2023.

[56] Yang, L., Zhang, Z., Song, Y., Hong, S., Xu, R., Zhao, Y., Zhang, W., Cui, B., and Yang, M.-H. Diffusion models: A comprehensive survey of methods and applications. *ACM Computing Surveys*, 56(4):1–39, 2023.

[57] Yang, L., Zhang, Z., Zhang, Z., Liu, X., Xu, M., Zhang, W., Meng, C., Ermon, S., and Cui, B. Consistency flow matching: Defining straight flows with velocity consistency. *arXiv preprint arXiv:2407.02398*, 2024.

[58] Yin, T., Gharbi, M., Zhang, R., Shechtman, E., Durand, F., Freeman, W. T., and Park, T. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6613–6623, 2024.

[59] Zhang, D., Han, Q., Xiong, Y., and Du, H. Mutlimodal straight flow matching for accelerated mr imaging. *Computers in Biology and Medicine*, pp. 108668, 2024.

[60] Zheng, H., Nie, W., Vahdat, A., Azizzadenesheli, K., and Anandkumar, A. Fast sampling of diffusion models via operator learning. In *International Conference on Machine Learning*, pp. 42390–42402. PMLR, 2023.

A. Additional Related Work

Standard vector field models use linear interpolants for the forward process [30, 20]. While mathematically sound, such paths might be suboptimal for modeling the reverse process [4]. Non-linear interpolants are a natural solutions, and other methods with different forms of forward process have been explored [1, 25]. These methods include merging blurring with Gaussian noise [9], diffusions applied in wavelet spectral domains [39], and non-Gaussian forward diffusions [34]. Daras et al. [9] can be considered as a specific form of this family of works which modify the forward processes, where components like noise schedules or data transformations (static [25, 42] or dynamic [36, 15]) are learned rather than fixed. Our works lies in the broader umbrella of learning these time-dependent dynamic transformations.

A different thread on these models purely focuses on increasing sampling efficiency. These include methods on changing the sampling process [44, 33, 60], and those based on distillation [46, 58], though the latter demands auxiliary models. These innovations are orthogonal to our work and can be potentially combined for enhanced performance. The specific problem of learning straighter generative trajectories has received some interest. However, these approaches requires distillation [32, 31] or solving adversarial optimization problems [1].

B. Additional results

B.1. Synthetic Data

We show results on an additional synthetic example consisting on a Gaussian source and the 4-square target in Figure 4.

B.2. Real Data

We show additional results for CIFAR-10, including the Inception score, in Table 4, and results for the AFHQ-Cat dataset in Table 5.

Table 4: IS score comparison with baseline models on CIFAR-10. Results for baselines are from literature.

Method	NFE (↓)	FID (↓)	IS (↑)
Score SDE [45]	2000	2.20	**9.89**
DDPM [20]	1000	3.17	9.46
PFGM [53]	110	2.35	9.68
EDM [24]	35	**2.04**	9.84
1-Rectified Flow [31]	1	378	1.13
Glow [27]	1	48.9	3.92
Consistency Model [46]	2	5.83	8.85
Consistency FM [57]	2	5.34	8.70
Ours	2	**4.61**	**9.18**
Ours (segmented)	2	**4.19**	**9.31**

Table 5: Comparison with baseline models on AFHQ-Cat.

Method	**AFHQ-Cat 256 × 256**	
	NFE (↓)	FID (↓)
Rectified Flow [31]	8	57.0
	6	61.5
Rectified Flow + Bellman [35]	8	33.9
	6	36.2
Consistency FM [57]	6	22.5
Ours	6	**20.8**
Ours (Segmented)	6	**20.2**

C. Counterexample for [4]

Consider a situation visually similar to to the setup from Figure 1, but with both the source and target being a mixture of two delta distributions in two dimensions. We'll denote the source distribution as $p(x_0) = 0.5\,\delta(x-x_0^1)+0.5\,\delta(x-x_0^2)$, and similarly the target distribution as $p(x_1) = 0.5\,\delta(x - x_1^1) + 0.5\,\delta(x - x_1^2)$, where $x_i^j \in \mathbb{R}^2$. Additionally, we will index each component of a point through parantheses, that is, $x_i^j = [x_i^j(0), x_i^j(1)]$ (for instance, $x_0^1 = [x_0^1(0), x_0^1(1)]$).

In our notation, Bartosh et al. [4] consider a forward process (or interpolant) given $x_t := \phi(x_0, x_1, t)$ and fit a neural model to the reverse process (generative direction) as

$$v_{\phi,\theta}(x_t) = \partial_t \phi(t, \hat{x}_0^\theta(x_t), \hat{x}_1^\theta(x_t)), \tag{21}$$

where $\hat{x}_0^\theta$ and $\hat{x}_1^\theta$ are neural networks parameterized by θ (we sometimes suppress the dependence on θ for notational convenience). They then train $v_{\phi,\theta}(x_t)$ by minimizing

$$\mathbb{E}_t \mathbb{E}_{x_0,x_1} ||v_{\phi,\theta}(x_t) - \partial_t \phi(x_0, x_1, t)||^2. \tag{22}$$

Contrasting this with the case of linear interpolants $x_t := tx_1 + (1-t)x_0$, we can see that the corresponding $v_{\phi,\theta}(x_t) = \hat{x}_1(x_t) - \hat{x}_0(x_t)$ corresponds to a neural network to predict $x_1 - x_0$.

We know the ground truth reverse process corresponds to $v^* = \mathbb{E}[\partial_t \phi(x_0, x_1, t)]$. On the other, given the chosen parameterization for $v_{\phi,\theta}$ from Equation (21), we must have

$$\forall x_t, t \quad v_{\phi,\theta}(x_t) := \partial_t \phi(t, \hat{x}_0(x_t), \hat{x}_1(x_t)) = \mathbb{E}[\partial_t \phi(x_0, x_1, t)], \tag{23}$$

with $\hat{x}_0(x_t), \hat{x}_1(x_t)$ being θ-parameterized neural networks.

We now show that, even with infinitely powerful predictors for $x_i(x_t)$, there exists some interpolant ϕ for which the above conditional cannot hold. More specifically, we construct a ϕ, x_t, t such that no functions $\hat{x}_0, \hat{x}_1$ can make Equation (23) true.

We consider paths parameterized by t given by

$$x_t = \phi(x_0, x_1, t) = tx_1 + (1-t)x_0 + kt(1-t)(t-0.5)g(x_1, x_0), \quad (24)$$

where k is a scalar and g is a function to be specified. We choose the points from the source and target distributions x_0^1 and x_0^2 such that

$$x_1^1 - x_1^2 = x_0^1 - x_0^2$$

We then consider the interpolants from x_0^1 to x_1^2 and from x_0^2 to x_1^1. These interpolants intersect at $t = 0.5$, giving the point:

$$x_{0.5} = \frac{x_0^1 + x_1^2}{2} = \frac{x_1^1 + x_0^2}{2}.$$

At this point, the optimal field is given by

$$\frac{\partial_t \phi(x_0^1, x_1^2, t = 0.5) + \partial_t \phi(x_0^2, x_1^1, t = 0.5)}{2}.$$

For the chosen ϕ, this field can be written as

$$\frac{x_1^2 - x_0^1 + k \cdot 0.25 \cdot g(x_1^2, x_0^1) + x_1^1 - x_0^2 + k \cdot 0.25 \cdot g(x_1^1, x_0^2)}{2}.$$

By choosing the points symmetrically, the linear terms $x_1^2 - x_0^1 + x_1^1 - x_0^2$ cancel out, leading to:

$$v^*(x_{0.5}) = \frac{k \cdot 0.25 \cdot g(x_1^2, x_0^1) + k \cdot 0.25 \cdot g(x_1^1, x_0^2)}{2}.$$

On the other hand, $v_{\phi,\theta}$ is given by

$$v_{\phi,\theta} = \phi'(\hat{x}_0, \hat{x}_1, t = 0.5) = \hat{x}_1 - \hat{x}_0 + k \cdot 0.25 \cdot g(\hat{x}_1, \hat{x}_0).$$

We now define $h(a, b) = g(a, b) + a - b$, and thus $g(a, b) = b - a + h(a, b)$. Substituting this in $v^*(x_{0.5})$ and $v_{\phi,\theta}$ yields

$$v^*(x_{0.5}) = \frac{k \cdot 0.25 \cdot h(x_1^2, x_0^1) + k \cdot 0.25 \cdot h(x_1^1, x_0^2)}{2}$$

and

$$v_{\phi,\theta} = k \cdot 0.25 \cdot h(\hat{x}_1, \hat{x}_0).$$

respectively.

Therefore, we need the following equation to be true

$$h(\hat{x}_1, \hat{x}_0) = \frac{h(x_1^1, x_0^2) + h(x_1^2, x_0^1)}{2}.$$

However, it is simple to find a function h for which this cannot hold. For instance, this is the case for

$$h(x_1, x_0) = [\cos(x_1(0)), \sin(x_1(0))].$$

If $x_1^1(0) = 0$ and $x_1^2(0) = \pi$, then:

$$h(x_1^1, x_0^2) = [\cos(0), \sin(0)] = [1, 0]$$

and

$$h(x_1^2, x_0^1) = [\cos(\pi), \sin(\pi)] = [-1, 0].$$

Therefore

$$\frac{h(x_1^1, x_0^2) + h(x_1^2, x_0^1)}{2} = \frac{[1, 0] + [-1, 0]}{2} = [0, 0].$$

However $h(\hat{x}_1, \hat{x}_0) = [\cos(\hat{x}_1(0)), \sin(\hat{x}_1(0))]$, which can never be $[0, 0]$.

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]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Not Applicable]
2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Not Applicable]
3. For all figures and tables that present empirical results, check if you include:
   (a) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (b) 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). [Not Applicable]
   (c) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [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]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]

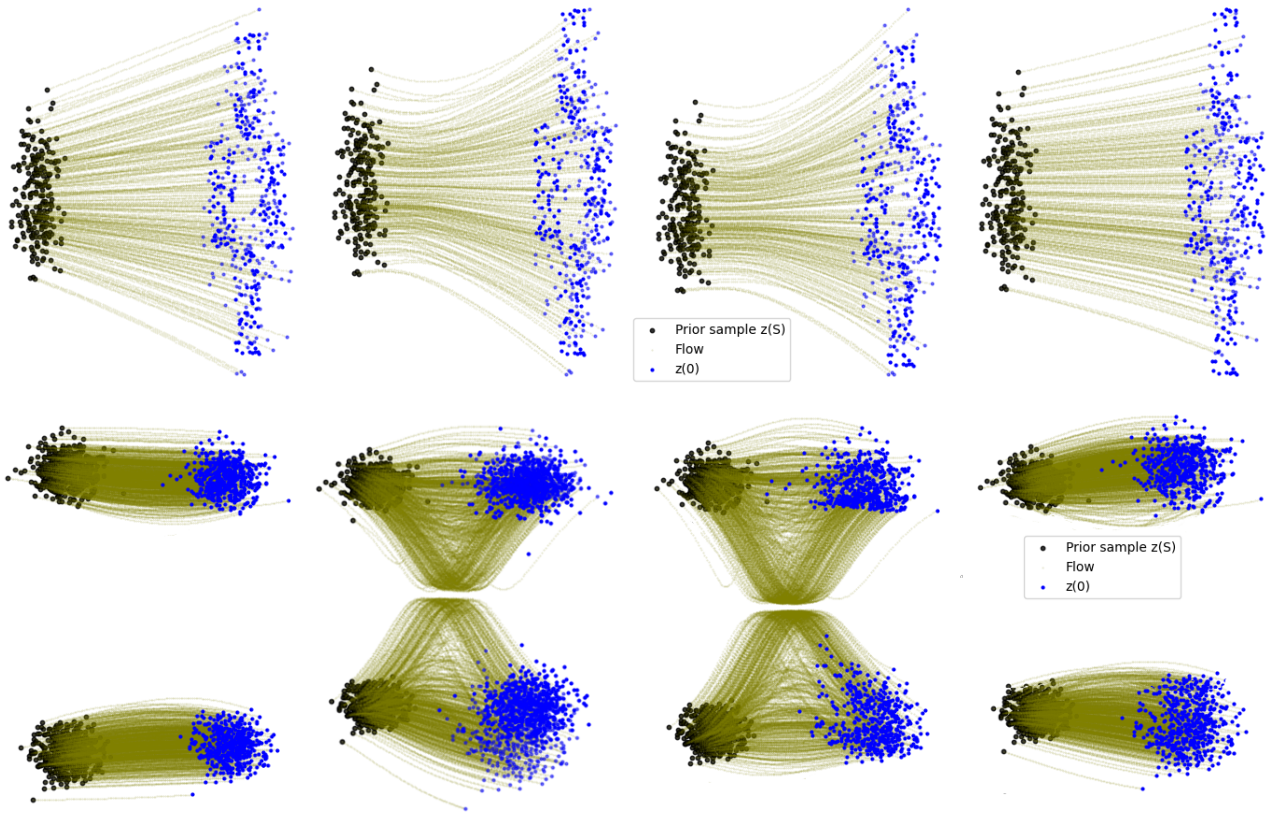


Figure 4: From left to right, the figure depicts flows learned by (1) OT Coupling, (2) Independent Coupling, (3) Sinkhorn Coupling and (4) Our method. For this example, the models were trained to transform Gaussian (source) to 4-square (target). We can see that, as expected, the OT map yields a perfectly straight field. We also see that by using learnt interpolants, our approach gets close to producing straight fields as well, despite using the independent coupling between the source and target distributions.

(d) Information about consent from data providers/curators. [Not Applicable]

(e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [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. [Not Applicable]

(b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]

(c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

C.1. Training Details

	CIFAR 10	ImageNet 32	CelebA	AFHQ-Cat
Channel Multipliers	1,2,2,2	1,2,2,2	1, 1, 2, 2, 2, 2, 2	1, 1, 2, 2, 2, 2, 2
BlockType	biggan	biggan	biggan	biggan
Nonlinearity	swish	swish	swish	swish
Attention resolution	16	16	16	16
GPUS	2 A100	4 A100	4 A100	4 A100
Iterations	500k	300k	350k	350k

Table 6: Training details for different datasets