עבודת גמר (תזה) לתואר
דוקטור לפילוסופיה

מוגשת למועצה המדעית של
מכון ויצמן למדע
רחובות, ישראל

מאת
חלי בן חמו

בינה יוצרת באמצעות מודלי זרימה רציפים ללא סימולציה

Thesis for the degree
Doctor of Philosophy

Submitted to the Scientific Council of the
Weizmann Institute of Science
Rehovot, Israel

By
Heli Ben Hamu

Simulation-Free Continuous Normalizing Flows for Generative Modeling

מנחה:
פרופ׳ ירון ליפמן

Advisor:
Prof. Yaron Lipman

תשרי התשפ״ה

October 2024

Abstract

Deep learning has revolutionized the way we build algorithms for analyzing, inferring, and modeling high-dimensional data. Long-standing challenges in fields such as image recognition, natural language understanding, and game playing have seen unprecedented success with deep learning models, often surpassing human-level performance. However, despite these advancements, generative models based on deep learning still faced limitations, particularly in their ability to efficiently model complex distributions with high-quality, diverse samples and likelihood estimation.

This dissertation focuses on continuous normalizing flows (CNFs), a powerful class of generative models, parameterized by a velocity field, flowing point samples to reshape densities. CNFs allow for likelihood computation, similar to autoregressive models, while offering a more scalable and efficient sampling procedure. We address the key challenge in naive maximum likelihood training of CNFs, which requires simulating an ordinary differential equation and differentiating through it, proposing novel simulation-free methods.

In the first part, we develop two new simulation-free approaches for efficient CNF training: (i) probability path matching, which optimizes the CNF to match a prescribed probability path connecting source and target distributions, and (ii) flow matching, which also matches prescribed probability paths, but, inspired by diffusion models, does so by reducing the generative modeling problem to a regression one onto conditional velocities that generate single data examples. We show that the flow matching framework scales up to high-dimensional data achieving state of the performance for image generation tasks.

In the second part, we introduce generalizations of flow matching for general joint distributions, extending the flow matching framework beyond generative modeling with applications to optimal transport which also enables improved sampling speed for generative tasks. Finally, we explore the dynamics of optimizing through the generation process, developing an approach for controlled generation with applications to inverse problems and conditional generation on domains such as images, audio, and molecular data.

Through these contributions, we enhanced the scalability, efficiency, and versatility of CNFs, expanding their potential applications in both generative modeling and broader fields involving complex data distributions.

Declaration

I hereby declare that this thesis summarizes my original research, performed under the guidance of my advisor Prof. Yaron Lipman. Wherever the contributions of others are involved, every effort was made to indicate this clearly by citing the relevant literature. In particular, [Ben+22; Poo+23] involved equal contributors, where all authors contributed to the idea, theoretic analysis, and implementation.

Acknowledgments

"'We did it!' cried Toad.
'Yes,' said Frog.
'If a running try did not work, and a running and waving try did not work,
and a running, waving, and jumping try did not work, I knew that a running,
waving, jumping, and shouting try just had to work.'"
— *Frog and Toad - The Kite, by Arnold Lobe*

As I read these lines to my 5 and 2 year old daughters, I realize that while we grow up, our challenges, by some mysterious conservation law, maintain a similar nature. In Frog and Toad, Frog and Toad repeatedly attempt to fly a kite. Frog holds the ball of string and sends Toad off running with the kite. Toad, with a head full of dreams and expectations, runs as fast as he can, but fails. He faces the laughter of three robins that sat nearby and returns discouraged and faithless to Frog. Luckily, Frog insists on trying again, and, despite numerous failed tries, they ultimately succeed with determination. This story resonates deeply with me, as it captures, among other things, the spirit of research—an ongoing series of attempts, adjustments, and leaps of faith.

Completing a PhD requires the resilience of Frog and the support of many Frog-spirited individuals to help lift you up during those inevitable Toad-like moments. I have been incredibly fortunate to have such people by my side throughout this journey. I owe my deepest gratitude to my advisor, Yaron Lipman, whose guidance and encouragement have been essential to my growth as a researcher. His belief in my abilities, his contagious passion, and his amazing ability to walk his students through hardship with great professional and emotional intelligence have been invaluable.

I am grateful for the mentorship and insights shared by my collaborators, whose perspectives, engaging discussions, and dedication have greatly shaped my path. My heartfelt thanks go to my lab mates at Weizmann: Itay Kezurer, Matan Atzmon, Niv Haim, Nimrod Segol, Hadar Serviansky, Amos Gropp, Omri Puny, and Neta Shaul. A special thanks are owed to Haggai Maron, who was my mentor in my early days at Yaron's lab, and to Lior Yariv, who has become a true friend, bound by the unspoken pact of upholding mental resilience in this challenging journey. I am also fortunate to have collaborated with many talented researchers and interns at Meta. Thank you to Ricky T. Q. Chen, Samuel Cohen, Itai Gat, Uriel Singer, Brian Karrer, Maximillian Nickel, Aram Alexander Pooladian, Carles Domingo Enrich, Brandon Amos, Joey Bose, Matt Le, and Aditya Grover for your support and partnership throughout my studies. And to my PhD committee members, Prof. Ronen Basri and Prof. Ohad Shamir, for your feedback and advice.

Finally, I would like to thank my family. To my parents, brothers, grandparents, aunts, and uncles, for being my role models growing up and raising me to be an aspiring female researcher. To my parents-in-law, for your support and countless babysitting hours. Last, and most importantly, to my three favorite people in the world. To my husband, Omer, my partner in the journey of life. Thank you for being the unique certain thing in this

uncertain voyage. For always believing in me, for carrying the weight during deadlines, and for being a shoulder to lean on in both joyful and hard moments. And to our daughters — Shahar, the morning's first ray of light, and Carmel, an evergreen mountain, for simply being a source of infinite playfulness and joy.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Generative modeling views data from a probabilistic perspective, aiming to model the distribution of observed samples. While originally motivated by the goal of estimating the probability density of an empirically observed distribution, modern, deep generative models have grown to be much more. Deep generative models have unlocked an amazing ability to generate novel samples from a learned distribution, estimate the probability density, and are also utilized as plug-and-play priors for solving inverse problems and for controlled generation. In this study, we take a journey intertwined with the developments that led to the success of generative artificial intelligence (GenAI) models.

We focus on a family of generative models called *continuous normalizing flows* (CNFs). CNFs build a generative model by parameterizing a continuous time generative process via its velocity field. Intuitively, the velocity field defines how particles sampled from a source distribution (typically one that is easy to sample from) move in time, moving mass so the reshaped probability density matches the target distribution. In contrast to latent deep generative models such as *generative adversarial networks* (GANs) and *variational auto-encoders* (VAEs), CNFs define a diffeomorphism between probability densities, which allow inverting the generation process and they allow computing model likelihoods. Similar to other likelihood-based generative methods such as *Normalizing Flows* (NFs) and *Auto-regressive models* (ARMs), CNFs are trained by direct maximum likelihood optimization. In this space, CNFs are appealing since they lift the architectural restrictions inherent to the design of NFs to ensure invertibility and cheap Jacobian computation, and compared to ARMs, they offer faster sampling procedures independent of data dimensionality. However, the common paradigm of maximum likelihood training made CNFs computationally heavy to train since it required backpropagating through the simulation of the ordinary differential equation (ODE) that defines the generation process.

The goal of this study is to introduce simulation-free training methods for CNFs as well as devise extensions and generalizations enabled by these new methods. In what follows, we describe the main results of our research addressing the challenges presented above and further explorations of CNFs.

1.1 Thesis Outline

This dissertation includes four papers to which I have contributed. In three of the four, I was a lead (or co-lead) author contributing to all aspects of the work, from theoretical to experimental. A full list of publications can be found in the Summary of Publications section. Below, we lay the outline of this dissertation.

The first part of the thesis introduces simulation-free methods for the training of CNFs:

- Chapter 3 is based on [Ben+22] and introduces *Probability Path Matching*, a first of its kind simulation-free training approach for CNFs that can match general probability paths.
  - In §3.3.1 we introduce the *Logarithmic Mass Conservation* formula, a PDE that couples between a flow model's velocity field and its generated log-probability path.
  - In §3.3.2, we derive the *Probability Path Divergence*, which serves as a training objective for training CNFs without having to simulate nor backpropagate through the solution of the ODE, by matching a target probability path.
  - In §3.3.3 we show how to build target probability paths. We are the first to introduce training of CNFs to match general probability paths between source and target distributions which are not tied to a stochastic process defined by a stochastic differential equation.
  - In §3.5 we experimentally demonstrate the capability of the proposed approach on higher dimensional manifolds compared to previous works, achieving state of the art results.
- Chapter 4 is based on [Lip+23] and introduces *Flow Matching*, the first simulation-free training approach for CNFs that has unbiased gradients, as opposed to *Probability Path Matching* proposed in §3, and that scales to high dimensional data.
  - §4.3 introduces the basic idea of flow matching – regressing a CNF 's velocity field to a predefined velocity field generating a target probability path.
  - In §4.3.1-§4.3.2 we propose an efficient way to match flows by regressing to conditional velocities generating conditional probability paths.
  - In §4.4 we construct target probability paths, showing that flow matching subsumes diffusion paths and we advocate a novel probability path choice inspired by optimal transport theory that results in a model that is easier to sample from, i.e.require fewer function evaluations to simulate the ODE.
  - In §4.6 we empirically show the effectiveness of the flow matching generative framework, achieving state of the art results on large scale image datasets.

In the subsequent two papers, we improve training and inference from flow matching models and propose a novel framework for controlled generation:

- Chapter 5 is based on [Poo+23] and introduces joint and *Multisample Flow Matching*, and extension to the flow matching framework for training flow models with non-trivial couplings between source and target distributions.
  - In §5.3 we introduce joint flow matching, generalizing independent couplings of source and target samples for a generally given joint, e.g.couples of low-resolution images and high-resolution images.
  - In §5.4 We propose the multi-sample framework for the construction of non-trivial joint distributions based on minibatch couplings.
  - In particular, in §5.4.2, we propose batch optimal transport multi-sample and show that in the limit of infinite batch size, the learned map will approach the optimal transport map.
  - Lastly, in §5.6, we demonstrate the benefit of multi-sample batch optimal transport for faster inference from flow matching models.
- Chapter 6 is based on [Ben+24] and introduces D-Flow. An optimization based controlled generation framework from flow and diffusion models.
  - In §6.2, we propose a simple optimization scheme for controlled generation from flow and diffusion models via differentiation through the generation process.
  - In §6.3, we provide a theoretical analysis of the proposed method, showing that for Gaussian affine probability paths, the optimization of the initial point of the ODE defined by the learned velocity field can be seen as projections on the data manifold along time.
  - Finally, in §6.5, we evaluate our method for linear and non-linear inverse problems on images and audio. We also use our method for conditional molecule generation, achieving state of the art performance across all applications and modalities.

Chapter 2

Preliminaries and Background

This chapter lays down the foundations for the research questions we answer in this dissertation. We begin by setting up the problem of generative modeling from a probabilistic point of view, then introduce the landscape of dominant approaches tackling this problem and discuss the challenges and shortcomings of each approach. At last, we focus our discussion on *Continuous Normalizing Flows*, the prime generative modeling paradigm researched in this study. We do not attempt to cover all necessary knowledge and assume the reader is familiar with key concepts in machine and deep learning (e.g., [GBC16]).

2.1 Generative Modeling

2.1.1 Problem Setup

Consider a finite observed dataset $\mathcal{D} \subseteq \mathbb{R}^d$, of size $|\mathcal{D}| = N$, where d is the data dimension, composed of independent, identically distributed (i.i.d) samples originating from an unknown data distribution $x \overset{\text{i.i.d.}}{\sim} p_{\text{data}}$ (e.g., natural images). The ultimate goal of generative modeling is to build a model that allows *efficient sampling* from the model distribution, p_{model}, given access only to $\mathcal{D}$, such that it approximates the data distribution:

$$p_{\text{model}} \approx p_{\text{data}}. \tag{2.1}$$

Toward this goal, the general scheme for learning such a model is to represent it by a function of free parameters, $\theta \in \mathbb{R}^p$, and seek for the optimal ones satisfying equation 2.1. Formally phrased as an optimization problem:

$$\min_{\theta} \quad \mathrm{D}\left(p_{\text{data}} \| p_\theta\right), \tag{2.2}$$

where $\mathrm{D}\left(\cdot\|\cdot\right)$ is a function that measures the difference between distributions and p_θ is the underlying model distribution, that is $p_\theta \coloneqq p_{\text{model}}$. Though this may seem abstract at this point, we will put all the pieces together and elaborate on possible optimization objectives, $\mathrm{D}\left(\cdot\|\cdot\right)$, and common successful approaches for modeling p_θ, either explicitly or implicitly that follow the scheme of problem 2.2.

2.1.2 Comparing Distributions: Metrics and Divergences

We framed the generative modeling problem as an optimization problem with the goal of minimizing the discrepancy between the model distribution and the data distribution. The literature on comparing statistical objects, particularly probability densities, is extensive and provides a range of metrics and divergences. We will review three prevalent notions for comparing probability distributions: integral probability metrics, Wasserstein distances, and f-divergences.

Integral probability metrics (IPMs). Consider probability density functions p and q over a set Ω. This family of metrics [Zol84; Mül97] uses the separation power of a class of real-valued functions on Ω, denoted as $\mathcal{F}$, to measure the difference between p and q. The metric is defined as follows:

$$\mathrm{D}_{\mathcal{F}}(p, q) = \sup_{f \in \mathcal{F}} \left| \mathbb{E}_{x \sim p} f(x) - \mathbb{E}_{y \sim q} f(y) \right| . \tag{2.3}$$

We note that different function classes, $\mathcal{F}$, define different metrics and for certain choices, $\mathrm{D}_{\mathcal{F}}(\cdot, \cdot)$ will actually be a pseudo-metric, violating the metric axiom that $\mathrm{D}_{\mathcal{F}}(p, q) = 0$ iff $p = q$. IPMs have been greatly studied as a theoretical tool in probability theory and statistics, but from a practical standpoint, efficient estimation of these metrics remains a challenge [Sri+12]. Nevertheless, there are a few instances, overlapping with the next two notions we will discuss, that are more widely known and used in practice.

Wasserstein distance. To define this family of metrics we will require that the domain over which p and q are defined is a metric space (Ω, d), where again Ω is a set and d is a metric on Ω. The Wasserstein ℓ-distance is defined as:

$$W_\ell(p, q) = \inf_{\pi \in \Pi(p,q)} \left[\mathbb{E}_{(x,y) \sim \pi} \mathrm{d}\,(x, y)^\ell \right]^{1/\ell}, \tag{2.4}$$

where $\Pi(p, q)$ is the set of all joint probability densities on $\Omega \times \Omega$ with marginals p and q with respect to the second and first factors respectively, that is:

$$\int_\Omega \pi(x, y) dy = p(x) \quad , \quad \int_\Omega \pi(x, y) dx = q(x). \tag{2.5}$$

A popular instance of the Wasserstein family is the W_1 Wasserstein distance which by the Kantorovich and Rubinstein duality theorem [KR58] is equivalent to an IPM, $\mathrm{D}_{\mathcal{F}}(\cdot, \cdot)$, with $\mathcal{F} = \{f \mid \text{continuous } f : \Omega \to \mathbb{R}, \mathrm{Lip}(f) \leq 1\}$, an equivalence that was used to improve generative adversarial networks (GANs) [ACB17]. The W_1 metric also shares connections to the optimal transport (OT) problem (see §5) and is equivalent to Kantorovich's formulation of the OT problem [Vil08] with the distance cost. Another well-known instance akin to OT is the W_2 Wasserstein distance, which has a dual dynamic representation [BB00]. We will revisit and elaborate on these connections to OT in Chapter 5, where we use OT-inspired techniques to improve generative modeling methods.

f-divergences. The f-divergences [Rén61; AS66; Csi67] of two probability densities p, q over Ω are defined by:

$$\mathrm{D}_f(p \,\|\, q) = \int_\Omega f\left(\frac{p(x)}{q(x)}\right) q(x)dx, \tag{2.6}$$

where $f : \mathbb{R}_{\geq 0} \to \mathbb{R}$ is a strictly convex function satisfying $f(1) = 0$. f-divergences satisfy the standard statistical divergence properties: $\mathrm{D}_f(p \,\|\, q) \geq 0$, and $\mathrm{D}_f(p \,\|\, q) = 0$ iff $p \equiv q$. f-divergences generalize standard divergences such as KL (with the choice $f(r) = r \log r$), reverse KL ($f(r) = -\log r$), total variation distance ($f(r) = |r - 1|$), and α-divergences ($f(r) = 1 - r^\alpha$ with $\alpha \neq 1, 0$).

KL Divergence and Maximum Likelihood Estimation

The last part of this section is dedicated to one of the most essential derivations in the literature of generative modeling, showing the equivalence of KL-divergence optimization to maximum likelihood estimation.

The KL-divergence, is an f-divergence with $f(r) = r \log r$:

$$\mathrm{D}_{KL}(p \,\|\, q) = \int_\Omega \log\left(\frac{p(x)}{q(x)}\right) p(x)dx = \mathbb{E}_{x\sim p}\left[\log\left(\frac{p(x)}{q(x)}\right)\right]. \tag{2.7}$$

When considering the generative modeling problem setup as in 2.2 with the KL-divergence, p_{data} takes the role of p and the model distribution, p_θ, takes the role of q in equation 2.7:

$$\mathrm{D}_{KL}(p_{\text{data}} \,\|\, p_\theta) = \mathbb{E}_{x\sim p_{\text{data}}}\left[\log\left(p_{\text{data}}(x)\right)\right] - \mathbb{E}_{x\sim p_{\text{data}}}\left[\log\left(p_\theta(x)\right)\right]. \tag{2.8}$$

Note that the first term in equation 2.8 does not depend on the model parameters, θ, and hence problem 2.2 with KL-divergence amounts to the *maximum likelihood estimation* approach:

$$\max_\theta \quad \mathbb{E}_{x\sim p_{\text{data}}}\left[\log\left(p_\theta(x)\right)\right]. \tag{2.9}$$

In practice, since we assume access to a finite set of samples from p_{data}, the expectation above is estimated empirically,

$$\max_\theta \quad \frac{1}{N}\sum_{i=1}^{N}\left[\log\left(p_\theta(x_i)\right)\right], \tag{2.10}$$

where $x_i \in \mathcal{D}$. The maximum likelihood estimator holds desirable statistical properties such as consistency and efficiency as sample size increases to infinity, that is $N \to \infty$, making it a go-to practice in statistics and is a core principle in the design of many popular generative modeling approaches, which we will now review.

2.2 Common Methodologies

In this section, we survey central methodologies in the landscape of deep generative models. The aim of this section is to give the reader a brief introduction to various approaches and ground our motivation to research and extend the use of continuous normalizing flows as our generative paradigm of choice in this study.

2.2.1 Likelihood Based Models

Recall that we framed the generative modeling problem as an optimization problem aiming to bring the model distribution close to the observed data distribution (see §2.1.1). In pursuit of this objective, models that directly represent the likelihood have been an appealing approach for designing generative models. Continuous normalizing flows belong to this class as well, yet we will discuss them with greater detail in §2.3.

Auto-Regressive Models

Auto-Regressive models (ARMs) historically date back to *belief networks* or *Bayesian networks* [Pea88; Nea92]. The key idea is to factorize the probability of a data point $x \in \mathbb{R}^d$ using the chain rule of probability over the dimensions, that is:

$$p_\theta(x) = \prod_{i=1}^{d} p_\theta(x_i | x_{<i}), \tag{2.11}$$

where x_i is the i-th coordinate of x and $x_{<i}$ denotes the $(i-1)$-dimensional vector holding the elements of x up to the $(i-1)$ coordinate, and abusing notation a bit, we define $x_{i<1}$ to be null. The product representation of one-dimensional distributions can be easily transformed to compute the log-likelihood as a sum of one-dimensional log densities, facilitating learning with the maximum likelihood principle derived in §2.1.2:

$$\max_\theta \quad \frac{1}{N} \sum_{j=1}^{N} \sum_{i=1}^{d} \log p_\theta(x_i^{(j)} | x_{<i}^{(j)}), \tag{2.12}$$

where $x^{(j)} \in \mathcal{D}$.

Modeling $\log p_\theta(x_i | x_{<i})$ with a neural network is, however, not trivial since density functions need to integrate to 1. For continuous distributions, one can either restrict the family of distributions represented by the model [TB15] or discretize the space [VKK16], which may not always be possible. Once a model is trained, generating samples with ARMs is done sequentially, constructing the sample coordinate by coordinate by sampling from $p_\theta(x_i | x_{<i})$.

ARMs have found great success in modeling sequential discrete data, in particular in language modeling [Rad+18], and are the foundational generative modeling framework driving the *large language models* (LLMs) revolution. Nevertheless, when the data does not have an inherent sequential structure, such as images, ARMs are a less intuitive choice.

Normalizing Flows

Normalizing Flows (NFs) transform continuous probability distributions by constructing a diffeomorphic map (smooth bijection with smooth inverse), called a *flow*, $\psi_\theta : \mathbb{R}^d \to \mathbb{R}^d$, mapping samples from a source distribution $z \sim p$ to the model distribution $\psi_\theta(z) \sim p_\theta$, which can be computed using the *change of variables* formula:

$$p_\theta(x) = p(\psi_\theta^{-1}(x)) \left|\det D_x \psi_\theta^{-1}(x)\right|, \tag{2.13}$$

where $D_x f$ denotes the Jacobian of a function f w.r.t x.

Early instantiations of such an approach appeared in [TV10; TT13]. Assuming we can easily compute the inverse and Jacobian of the flow, we can apply once again the maximum likelihood principle in our optimization objective by taking the log of equation 2.13:

$$\max_\theta \quad \frac{1}{N} \sum_{i=1}^{N} \log p(\psi_\theta^{-1}(x_i)) + \log \left|\det D_x \psi_\theta^{-1}(x_i)\right|, \tag{2.14}$$

where $x_i \in \mathcal{D}$. Sampling from NFs is cheap, compared to the sequential procedure required in ARMs, producing a sample in a single evaluation of the learned flow $\psi_\theta(z)$ with $z \sim p$.

In recent years, normalizing flows have been widely adopted in tasks like image generation and density estimation in scientific applications. Designing neural architectures that allow tractable computations of equation 2.14 has been a central area of research, aiming to achieve high expressivity with restricted architectures. Advances such as [RM15; DSB17; KD18] have significantly improved the ability of normalizing flows to handle high-dimensional data, nevertheless, NFs remained inferior in sample quality compared to the two generative modeling approaches we will discuss next.

2.2.2 Latent Variable Models

A different class of generative models takes a different perspective focusing on modeling the generative process. The key component is to introduce a latent variable $z \in \mathbb{R}^{d'}$, where typically $d' < d$, and learn a generator (deterministic or probabilistic), $\psi_\theta : \mathbb{R}^{d'} \to \mathbb{R}^d$, from latent space to data space. This approach is also related to a representation learning and dimensionality reduction perspective following the hypothesis that high dimensional data lies on a lower dimensional manifold.

Framing this in a probabilistic language, the generator ψ_θ is a sampler from the likelihood $p_\theta(x|z)$. Now, if one designs its latent variable model such that the prior $p(z)$ is easy to sample from, we achieve a computationally efficient generative process for sampling from the marginal likelihood (or the *evidence*), $p_\theta(x)$, by first sampling $z \sim p$ and then computing $x = \psi_\theta(z)$, where

$$p_\theta(x) = \int p_\theta(x|z)p(z)dz. \tag{2.15}$$

We remain with the question, how do we train such generator ψ_θ?

Variational Auto-Encoders

Variational Auto-Encoders (VAEs) [KW14] extend traditional autoencoders by introducing a probabilistic approach to latent space representation. Unlike standard autoencoders, which map data to a fixed latent space, VAEs learn to encode data into a probability distribution, typically Gaussian, from which latent variables are sampled.

The introductory part of this section sets out to learn a function, $\psi_\theta(z)$, that enables sampling from the likelihood $p_\theta(x|z)$. Using Bayes rule, this model defines the posterior:

$$p_\theta(z|x) = \frac{p_\theta(x|z)p(z)}{p_\theta(x)} \tag{2.16}$$

which is intractable. In VAEs, we define another function $\mathcal{E}_\phi : \mathbb{R}^d \to \mathbb{R}^{d'}$, with parameters ϕ, that maps data to latent space, and is used to model the approximate posterior $q_\phi(z|x)$. In analogy to autoencoders, $\mathcal{E}_\phi$ is the encoder, and ψ_θ is the decoder.

[KW14] then derive the *evidence lower bound* (ELBO) in terms of these models:

$$\begin{aligned}\log p_\theta(x) = \log \int p_\theta(x|z)p(z)dz = \log \mathbb{E}_{q_\phi(z|x)}\left[\frac{p_\theta(x|z)p(z)}{q_\phi(z|x)}\right] \\ \overset{\text{Jensen}}{\geq} \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p_\theta(x|z)p(z)}{q_\phi(z|x)}\right] = \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right] - \mathrm{D}_{KL}(q_\phi(z|x) \,\|\, p(z)),\end{aligned} \tag{2.17}$$

and train the model by maximizing the ELBO, indirectly maximizing the likelihood:

$$\max_{\theta,\phi} \quad \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{q_\phi(z|x_i)}\left[\log p_\theta(x_i|z)\right] - \mathrm{D}_{KL}(q_\phi(z|x_i) \,\|\, p(z)). \tag{2.18}$$

In practice, the first term is approximated, assuming that $p_\theta(x|z)$ is a Gaussian, resulting in a reconstruction-like type of loss while the second term is handled by a design trick, known as the parameterization trick, where the encoder function outputs the mean and variance of a Gaussian distribution, and together with a choice of Gaussian prior $p(z)$, we end up with a simple KL term. The training process optimizes a loss function that combines reconstruction accuracy with a regularization term (the Kullback-Leibler divergence) that encourages the latent space to conform to a known prior.

VAEs are widely used in tasks such as data generation, dimensionality reduction, and semi-supervised learning, as they offer a structured way of generating new, diverse data samples by manipulating the latent space. VAEs provide a robust framework for learning meaningful representations of complex data distributions, yet their generation quality is behind leading models such as generative adversarial networks, discussed below.

Generative Adversarial Networks

Generative Adversarial Networks (GANs), proposed by [Goo+14], cast the generative modeling problem as a two-player adversarial game, where the first player (the generator) tries to perfect in generating realistic samples from the target data distribution and the second player (the discriminator) aims to distinguish between generated and real samples.

Training a GAN involves two neural networks: (i) the generator, $\psi_\theta : \mathbb{R}^{d'} \to \mathbb{R}^d$, mapping latent variables $z \sim p$ to data space, modeling sampling from $p_\theta(x|z)$; and (ii) the discriminator, $h_\phi : \mathbb{R}^d \to \mathbb{R}$, outputting a score of realness for given samples. The adversarial training of these two networks can be put into the following minimax optimization:

$$\min_\theta \max_\phi \quad \mathbb{E}_{x\sim p_{\text{data}}} [\log h_\phi(x)] - \mathbb{E}_{z\sim p} [\log h_\phi(\psi_\theta(z))] . \tag{2.19}$$

Theoretically, when the models have sufficient capacity and the networks are trained in an alternating manner, where at each iteration the discriminator reaches optimum, the sampling procedure of $z \sim p$ and applying $\psi_\theta(z)$ is shown to converge to that of sampling from the data distribution.

In practice, however, optimizing the above game turned out to be very challenging and unstable. Over the years, various techniques for stabilizing and improving the training of GANs have been proposed [Sal+16; AB17; ACB17; Gul+17] achieving exceptional new abilities in generating realistic samples. A line of works by Karras et al. [KLA19; Kar+20; Kar+21] developed new architectures and designs that fully exploit the latent space of GANs and facilitate unsupervised representation learning and separation of high-level features.

GANs have served as the dominating deep generative paradigm for most data types, except perhaps for language modeling, until the re-introduction of diffusion models, discussed next in §2.2.3, around the year 2020. As a very efficient class of generative models, requiring a single network evaluation for sample generation compared to diffusion and flows that require multiple steps in generation, some recent works [Kan+23] question their decline by developing techniques to scale GANs to large scale text-2-image generation tasks.

2.2.3 Time-Dependent Generative Models

Time-dependent generative models view the generative process as a dynamic system sequentially transforming distributions along an auxiliary dimension (discrete or continuous), typically attributed to time, t (but could also be attached to a noise level σ, as in [SE19]). Under this definition, one can find Diffusion-based generative models, Score-based generative models, and CNFs. In this section, we will adopt the unifying perspective presented in [Son+21b] for describing diffusion and score-based models, and in the next section, we will provide a more delicate treatment of CNFs. Diffusion models for generative modeling were first introduced in [Soh+15], adopting principles from non-equilibrium physics, applying a forward diffusion process, turning data into Gaussian noise in an

iterative process, and learning the reverse process, serving as a generative model. This approach, however, did not gain the research community's attention until [HJA20] revisited the proposed method and suggested a simpler optimization objective achieving incredible results on image generation.

In concurrence, another rising generative modeling approach, called Score-matching, was introduced by [SE19]. Score matching proposes building a generative model by learning to estimate the score function of the data distribution, $\nabla \log p_{\text{data}}(x)$. Nonetheless, naively learning $\nabla \log p_{\text{data}}(x)$, one runs into various challenges, such as inaccurate score estimation in areas of low density as well as slow mixing of Langevin dynamics or any other method for sampling from un-normalized densities. These issues become even more pronounced as data dimensionality grows. To overcome these challenges, [SE19] proposed to learn a sequence of score functions of the data distributions convolved with a noising kernel (e.g., Gaussian), with increasing noise levels.

[Son+21b] unified both approaches under a stochastic process perspective. The forward diffusion process of [HJA20], known as Variance Preserving (VP), and the noising process proposed in [SE19], known as Variance Exploding (VE), were shown to be instances of forward processes described by a Stochastic Differential Equation (SDE) of the standard form

$$dx_t = f_t(x)dt + g_t(x)dw, \tag{2.20}$$

with time parameter $t \in [0, T]$, starting from data at $t = 0$, drift f_t, diffusion coefficient g_t, and dw is the Wiener process. The solution x_t to the SDE is a stochastic process, i.e., a continuous time-dependent random variable, the probability density of which, $p_t(x_t)$, is characterized by the Fokker-Planck equation:

$$\partial_t p_t(x) = -\text{div}(f_t(x)p_t(x)) + \frac{g_t^2(x)}{2}\Delta p_t(x) \tag{2.21}$$

where Δ represents the Laplace operator (in x), namely $\text{div}\nabla$, where ∇ is the gradient operator (also in x). At $T \to \infty$, p_T reaches the stationary distribution of the forward process.

The generative process is defined by the reverse process, and is also a diffusion process depending on the intermediate score functions [And82]:

$$dx_t = [f_t(x) - g_t^2(x)\nabla \log p_t(x)]dt + g_t(x)d\bar{w}, \tag{2.22}$$

where $d\bar{w}$ is a Wiener process and time flows backwards from T to 0. It is then apparent that if one can estimate the score function, a generative model is at hand by simulating equation 2.22 with initial condition $x_T \sim p_T$.

How can we learn the score efficiently then? Both [HJA20; SE19] proposed simulation-free regression objectives for learning the score (or an affine re-parameterization of it [HJA20]). While different points of view led to these objectives, they have been shown to be equivalent up to a time-dependent weighting. We will now describe the derivation from [SE19], which aligns with our perspective on CNFs and our proposed objective in §4.

We parameterize a time-dependent score function by a neural network $s_t(x;\theta) : [0,T] \times \mathbb{R}^d \to \mathbb{R}^d$ with parameters $\theta \in \mathbb{R}^p$. [SE19] used the useful result from [HD05], on the optimal score solving the following optimization problem:

$$\min_\theta \mathbb{E}_t \left[w(t) \mathbb{E}_{x_0 \sim p_{\text{data}}, x_t \sim p_t(x|x_0)} \left\| s_t(x_t;\theta) - \nabla \log p_t(x_t|x_0) \right\|^2 \right], \tag{2.23}$$

which is $s_t(x;\theta^*) = \nabla \log p_t(x)$ where $p_t(x) = \int p_t(x|x_0) p_{\text{data}}(x_0) dx_0$. A neat property of diffusion processes with affine drifts, f_t, is that $p_t(x|x_0)$ is a Gaussian with closed form mean and covariance, enabling simulation-free computation of the objective in equation 2.23. In the following chapters, we will revisit connections between CNFs and score and diffusion-based models. A noteworthy work, unifying simulation-free methods for diffusion and flows, was recently published by [ABV23].

2.3 Continuous Normalizing Flows

In this section, we introduce *continuous normalizing flows* (CNFs), or with a slight abuse of naming just *flows*. CNFs are a class of generative models that picture the generative modeling problem as a continuous time probability mass transport parameterized by a time-dependent velocity field. While they can also be classified under §2.2.1 and §2.2.3, we dedicate a deeper discussion on their foundations as they are the main generative paradigm this thesis builds upon.

In CNFs, a velocity field ,v_t, parameterizes a time-dependent diffeomorphic map, called a *flow*, $\psi : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$, defined via the ordinary differential equation (ODE):

$$\frac{d}{dt}\psi_t(x) = v_t(\psi_t(x)) \tag{2.24}$$

$$\psi_0(x) = x. \tag{2.25}$$

[Che+18] suggested modeling the velocity field v_t with a neural network, $v_t(x;\theta)$, where $\theta \in \mathbb{R}^p$ are its learnable parameters, which in turn leads to a deep parametric model of the flow ψ_t. A flow is used to reshape a simple prior density $p_0 \coloneqq p$ (e.g., pure noise) to a more complicated one, p_1, via the push-forward equation

$$p_t = [\psi_t]_* p_0, \tag{2.26}$$

where the push-forward (or change of variables, see equation 2.13) operator $*$ is defined by

$$[\psi_t]_* p_0(x) = p_0(\psi_t^{-1}(x)) \left| \det D_x \psi_t^{-1}(x) \right|. \tag{2.27}$$

2.3.1 The Continuity Equation

In physics, equation 2.24 is known as a Lagrangian description, where the change in density is implied by particle movement along trajectories, $\psi_t(x)$, obtained by integrating over the velocity field. The *continuity equation* presents a dual representation (see [Vil08]), called the Eulerian description, expressing the physical world via densities:

$$\partial_t p_t(x) + \operatorname{div}(p_t(x)v_t(x)) = 0, \tag{2.28}$$

where the divergence operator, div, is defined with respect to the spatial variable $x = (x^1, \ldots, x^d)$, i.e., $\operatorname{div} = \sum_{i=1}^{d} \frac{\partial}{\partial x^i}$, which is the trace of the Jacobian. The continuity equation is a fundamental principle in physics, particularly in fluid dynamics, and it describes the conservation of a physical quantity, which in our case is probability mass. We will use this principle to design a new CNF training approach in §3 and as a tool for proofs and derivations.

2.3.2 Likelihood Computation

Computing the log-likelihood a CNF generates at time t can be done by solving a system of ODEs. The instantaneous change of variables [Che+18] connects between the velocity field $v_t(x)$ and the log-likelihood:

$$\frac{d}{dt} \log p_t(\psi_t(x)) + \operatorname{div}(v_t(\psi_t(x)) = 0. \tag{2.29}$$

Integrating $t \in [0, s]$ gives:

$$\log p_s(\psi_s(x)) = \log p_0(\psi_0(x)) - \int_0^s \operatorname{div}(v_t(\psi_t(x)))dt \tag{2.30}$$

Assuming we choose p_0 such that we can compute $\log p_0(x)$, e.g., Gaussian, the log probability at (s, x) can be computed together with the flow trajectory by solving the ODE system backward in time:

$$\frac{d}{dt} \begin{bmatrix} x(t) \\ f(t) \end{bmatrix} = \begin{bmatrix} v_t(x(t)) \\ \operatorname{div}(v_t(x(t))) \end{bmatrix}, \tag{2.31}$$

given initial conditions

$$\begin{bmatrix} x(s) \\ f(s) \end{bmatrix} = \begin{bmatrix} x \\ 0 \end{bmatrix}, \tag{2.32}$$

and then plugging:

$$\log p_s(x) = \log p_0(x(0)) + f(0). \tag{2.33}$$

2.3.3 Maximum Likelihood Training

We have shown how one can compute the likelihood of a CNF model by solving a system of ODEs, and can, therefore, once again adopt the maximum likelihood principle in designing our objective:

$$\max_{\theta} \quad \frac{1}{N}\sum_{i=1}^{N}\left[\log\left(p_1(x_i)\right)\right], \tag{2.34}$$

where $x_i \in \mathcal{D}$ and $\log p_1$ is computed as in equation 2.33.

Hutchinson Trace Estimator. Optimizing equation 2.34 in high dimensions becomes computationally prohibitive as it requires evaluation of the divergence operator of high dimensional vector fields, namely taking the trace of the Jacobian, which scales quadratically with the dimension. [Gra+19] proposed overcoming this computational hurdle by turning to the Hutchinson trace estimator [Hut90], obtaining an unbiased estimator of the div operator using vector-Jacobian products which roughly scale linearly with dimension.

For a matrix, $Q \in \mathbb{R}^{d\times d}$, it can be shown that

$$\mathrm{Tr}\left(Q\right) = \mathbb{E}\left[\epsilon^T Q \epsilon\right] \tag{2.35}$$

where ϵ is a random variable with $\mathbb{E}[\epsilon] = 0$ and covariance $\mathrm{Cov}(\epsilon) = I$. Common choices for p_ϵ are Gaussian or Rademacher distributions. The Hutchinson trace estimator is realized by the Monte Carlo estimate of equation 2.35.

While significantly reducing the computational complexity required for evaluating the likelihood of a CNF, there is another axis in which the complexity can change – the amount of velocity field evaluations required for solving equation 2.31 with low discretization error.

ODE Solvers. In practice, solving and simulating ODEs on a computer amount to numerically evaluating integrals. For simplicity, let us consider the sampling procedure from a CNF, solving equation 2.24, namely

$$\psi_1(x_0) = x_0 + \int_0^1 v_t(\psi_t(x_0))dt. \tag{2.36}$$

The simplest numerical solver, approximating equation 2.36 is known as the *Euler Method* discretizing the solution time interval to equally distant time stamps ($t_0 = 0, t_1, \ldots, t_i, \ldots, t_N = 1$) and iteratively updating x_t:

$$x_{t_{i+1}} = x_{t_i} + \Delta t v_{t_i}(x_{t_i}) \tag{2.37}$$

where $\Delta t = 1/N$ is the discretization step. The Euler method can be derived from a first-order Taylor expansion of $\psi_t(x_0)$ around $t+\Delta t$, and its local error is linear in Δt, that is $o(\Delta t)$. Improving the dependency of

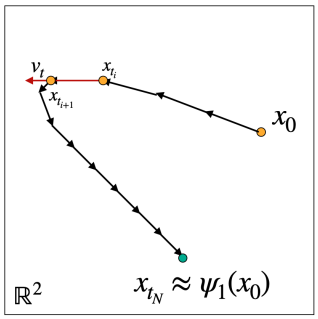


Figure 2.1: Euler Method.

the accumulated error in the size of the discretization step can be achieved by considering higher-order solvers. Furthermore, to save computation, one can adopt adaptive step size methods.

Nonetheless, for the likelihood estimation in equation 2.31 to produce meaningful gradients when backpropagating through the objective in equation 2.34 one needs to simulate the ODE with very low errors, and as training progresses it has been observed [Fin+20b] that the trajectories defined by the learned velocity become more complex and hence require a larger number of steps to simulate.

2.3.4 Degrees of Freedom

The above-mentioned issues with simulation-based training of CNFs have limited their applicability to large scales, e.g., images in a resolution higher than 32×32. In this last section of our introductory chapter, I would like to shape a degrees of freedom perspective on how CNFs are trained, a perspective that will accompany us in the next two chapters in designing simulation-free CNF training methods.

Maximum likelihood training, as done in FFJORD [Gra+19], directly supervises only the terminal log-likelihood, $\log p_1$, at time $t = 1$. There are, therefore, infinitely many intermediate p_t probability paths that can transform the source distribution p to the optimal p_1. Furthermore, for a given probability path p_t, there are infinitely many velocity fields that generate it, by adding a divergence-free term to the velocity field, see equation 2.28. This leads us to the question:

Can we reduce degrees of freedom to gain efficiency?

We will now discuss two strategies to do so: (i) regularization; and (ii) model restriction.

Regularization. The observation that with naive maximum likelihood training the velocity field becomes highly complex has led to several works that utilize optimal transport inspired penalties to regularize continuous normalizing flows for easier simulation. As optimal transport aims at finding a minimal cost transformation between distributions, that is, intuitively, point trajectories should not stray away when reshaping p into q. Indeed, in its dynamic formulation, optimal transport seeks to minimize the 2-Wasserstein distance between the source and target distribution:

$$W_2^2(p, q) = \min_{p_t, v_t} \int_0^1 \int_{\mathbb{R}^d} \|v_t(x)\|^2 \, p_t(x) \mathrm{d}x \mathrm{d}t. \tag{2.38}$$

A physical interpretation of this objective is that one seeks to minimize the kinetic energy. Various instances [Fin+20b; Onk+21b] used the above motivation to craft regularizing terms for maximum likelihood training of CNFs. While improving training and inference efficiency compared to FFJORD, these methods still lagged behind in both computational overhead and sample quality.

Table 2.1: Comparison of CNF training methods. FFJORD and RNODE maximize the log-likelihood, while Moser Flow maximizes the direct likelihood, which is a clear numerical disadvantage in high dimensions. The last column states the degrees of freedom in minimizing each method's objective: the probability path, p_t and/or velocity field, v_t.

Method	Maximum Likelihood Training	Simulation-Free Training	Does not require divergence	Degrees of Freedom
FFJORD [Gra+19]	$\log p_1$	✗	✗	p_t , v_t
RNODE [Fin+20b]	$\log p_1$	✗	✗	p_t , v_t
Moser Flow [Roz+21]	p_1	✓	✗	v_t
Probability Path Matching (§3) [Ben+22]	✗	✓	✗	v_t
Flow Matching (§4) [Lip+23]	✗	✓	✓	✗

Model Restriction. A recent work by [Roz+21], named Moser Flow, initiated a paradigm shift in CNF learning towards *simulation-free* training, refraining from solving an ODE during training. In Moser flow, the terminal likelihood p_1 is defined as $p_1 = p - \mathrm{div}(u)$ where u is a time-independent vector field. Parameterizing the velocity field to be $v_t(x) = \frac{u}{p_t}$ where $p_t = (1-t)p + tq$ restricts the resulting probability path generated by the learned CNF. This construction facilitated maximum likelihood training in a simulation-free manner.

However, while improving the computational costs of training, this work met other challenges in scaling CNFs to higher dimensions, such as intractable integrals computations and the use of the actual probability density rather than the log density. Table 2.1 summarizes the attributes of different CNF training paradigms, where the last two rows represent the methods introduced in this thesis in the following chapters. The methods we propose next are simulation-free, and they achieve this goal by introducing additional supervision and moving away from maximum likelihood training.

Part I

Simulation-Free Training of Continuous Normalizing Flows

Chapter 3

Probability Path Matching

> "I call it the law of the instrument, and it may be formulated as follows: Give a small boy a hammer, and he will find that everything he encounters needs pounding."
>
> — *Abraham Kaplan, 1964*

We concluded the previous chapter with a motivation to find approaches for restricting the degrees of freedom in CNF training, with the hope that these will, in turn, trade off with the need to simulate the ODE. In this chapter, we take our first step in this journey, seeking a tool that is not a hammer (i.e., log-likelihood) in the CNF toolkit. Inspired by score-based [SE19] and Diffusion models [HJA20], we conceptually change the way CNFs are trained. Rather than maximizing the log-likelihood of the trained CNF at $t = 1$ on data samples, we advocate matching it to an entire path of probability density functions over $t \in [0, 1]$ that connects between a source distribution p_0 and an approximation of the data distribution $p_1 \approx p_{\text{data}}$.

To this end, we introduce a novel family of divergences, *probability path divergence* (PPD), between the probability density path generated by the CNF and a target probability density path and apply them to generative tasks on manifolds. PPD is formulated using a logarithmic mass conservation formula, which is a linear first-order partial differential equation relating the log target probabilities and the CNF's defining velocity field. PPD has several key benefits over existing methods: it facilitates simulation-free training, readily applies to manifold data, scales to high dimensions, and is compatible with a large family of target paths interpolating pure noise and data in finite time. Theoretically, PPD is shown to bound classical probability divergences. Empirically, we show that CNFs learned by minimizing PPD achieve state-of-the-art results in likelihoods and sample quality on existing low-dimensional manifold benchmarks and are the first example of a generative model to scale to moderately high dimensional manifolds.

Remark. This chapter describes the published work [Ben+22]. Originally, the proposed method was given the acronym CNFM, standing for *Continous Normalizing Flow Matching*; however, in retrospect, in this dissertation, we rename it *Probability Path Matching* (PPM).

3.1 Motivation and Contributions

While early literature on generative modeling primarily focused on Euclidean data, the need to model data in non-Euclidean spaces arises in many scientific fields. For instance, occurrences of natural phenomena on earth can be modeled as a distribution on a sphere [MN20], protein structure prediction requires angle predictions [Mar+08], and motion and position of robots can be modeled with a product of Euclidean spaces and spheres. Therefore, constructing manifold-aware models which provide a geometric inductive bias might result in better and easier to train models with many potential applications.

Innate candidates for designing generative models on manifolds are Normalizing Flows (NFs) [RM15] and CNFs [Che+18]. In these approaches, the generator, ψ, is a diffeomorphism, i.e., a smooth bijection with a smooth inverse. Therefore, the model density can be expressed in terms of the prior density and the determinant of the Jacobian of ψ, also known as the change of variable formula, which can be naturally adapted to the manifold case. Recently, [Rez+20; Bos+20] devised NF models for sphere, tori and hyperbolic spaces. In a parallel line of works, [MN20; Lou+20; FF20] developed CNFs over Riemannian manifolds.

In this chapter, we aim to alleviate some of the limitations of previous approaches by introducing a novel family of Probability Path Divergences (PPD) used as an objective for training CNFs enabling *simulation-free* training. The PPD family is a divergence defined between an arbitrary target probability path (in time), p, and the probability path generated by the CNF, q. To define the PPD we first introduce the Logarithmic Mass Conservation (LMC) formula, a Partial Differential Equation (PDE), derived from the well known continuity equation (e.g.compressible fluid dynamics), that couples $\log q$ and the CNF's velocity field. Then, the PPD is defined as the extent to which $\log p$ and the CNF's velocity field fail to satisfy the LMC. PPD has the following desirable properties: (i) It is a proper divergence in the sense that it is non-negative, and zero iff $p \equiv q$. (ii) It does not require evaluating q during training; it is defined solely in terms of the parametric velocity field v_θ, its first order derivatives, and the target path's log density, $\log p$. This provides a speed up of $1-2$ orders of magnitude in evaluating the PPD and its derivatives, compared to, e.g., log likelihood. (iii) It is readily applicable to manifolds and higher dimensional data. (iv) The PPD has a single parameter $\ell \geq 1$. PPD with $\ell = 1$ upper bounds the total-variation distance comparing p and q at arbitrary times; PPD with $1 < \ell < \infty$ bounds their α-divergence; and PPD with $\ell = \infty$ bounds their reversed KL-divergence.

We call the minimization problem of the PPD between a target path p and a CNF density q, *Probability Path Matching* (PPM) and use it to train CNFs. The main design choice in PPM is the target path p. The requirements from p are: that it transforms a simple prior (pure noise) to an approximation of the unknown data distribution; that samples can be drawn from each p_t, where p_t represents the density at time t; and that we can compute or approximate the derivatives of $\log p_t$. Any p satisfying these requirements can be used to train a CNF in the Probability Path Matching framework. Other methods

that try to fit the generated probability density path to a target one are Score and Diffusion methods [SE19; HJA20; Son+21b]. However, these methods require target paths that are generated by stochastic differential equations (SDEs), which limits their applicability on manifolds.

We test our framework on several low and moderately high dimensional manifold data including Euclidean spaces, spheres/hyperspheres, and product of spheres, demonstrating state-of-the-art sample quality and likelihoods in standard low-dimensional manifold datasets. We demonstrate that PPM is considerably faster to optimize than state-of-the-art CNF training algorithms, allowing the scale of CNF training to considerably larger network architectures. Lastly, we demonstrate that PPM can train CNFs on moderately high dimensional manifolds, in contrast to previous methods of generative modeling on manifolds that mostly worked with low dimensional manifolds.

3.2 Notations

Let $\mathcal{M}$ be a d-dimensional smooth Riemannian manifold with a metric g and induced volume form dV, the volume of $\mathcal{M}$ is $|\mathcal{M}| = \int_{\mathcal{M}} dV_x$. We consider strictly positive, smooth probability densities over $\mathcal{M}$, $\mu : \mathcal{M} \to \mathbb{R}_{>0}$, satisfying $\int_{\mathcal{M}} \mu(x) dV_x = 1$. The tangent space at point $x \in \mathcal{M}$ is denoted $T_x\mathcal{M}$; the tangent bundle, which is the disjoint union of all tangent spaces of $\mathcal{M}$ is denoted $T\mathcal{M}$. The metric g defines an inner product for pairs of vectors $\xi, \eta \in T_x\mathcal{M}$ denoted by $\langle \xi, \eta \rangle$; a norm of a tangent vector is defined by $|\xi| = \langle \xi, \xi \rangle^{1/2}$. The Riemannian gradient of a smooth function $f : \mathcal{M} \to \mathbb{R}$ is denoted $\nabla f(x) \in T_x\mathcal{M}$. A time-dependent velocity field $v(t, x)$ is a smooth function $v : [0, 1] \times \mathcal{M} \to T\mathcal{M}$ such that $v(t, x) \in T_x\mathcal{M}$ for all $t \in [0, 1]$ and $x \in \mathcal{M}$. We denote the collection of bounded time-dependent smooth velocity fields over $\mathcal{M}$ by $\mathfrak{X}(\mathcal{M})$; by bounded we mean that for each $v \in \mathfrak{X}(\mathcal{M})$ there exists a constant $M > 0$ so that $|v(t, x)| \leq M$ for all $x \in \mathcal{M}$, $t \in [0, 1]$. The Riemannian divergence (w.r.t. x) of a smooth velocity field $v \in \mathfrak{X}(\mathcal{M})$ is denoted $\operatorname{div}(v)$. We denote by $\exp_x : T_x\mathcal{M} \to \mathcal{M}$, and $\log_x : \mathcal{M} \to T_x\mathcal{M}$ the Riemannian exponential and logarithmic maps. Note these should not be confused with the standard $\exp, \log$ that are written without subscript.

Remark All derivations in this chapter are done for a general Riemannian manifold case. We note that the Euclidean case, where $\mathcal{M} \equiv \mathbb{R}^d$ is a particular case and all results trivially apply.

3.3 Matching Flows to Probability Paths

The central object we use in this chapter is the *probability path*. Let $\mathfrak{P}(\mathcal{M})$ denote all *probability paths* on $\mathcal{M}$, that is, functions $p : [0, 1] \times \mathcal{M} \to \mathbb{R}_{>0}$, smooth in t and satisfying $\int_{\mathcal{M}} p(t, x) dV_x = 1$. For brevity, we will position the time argument, t, as a subscript $p(t, x) \equiv p_t(x)$ and we will use the notation p_t to denote the density at time t, namely,

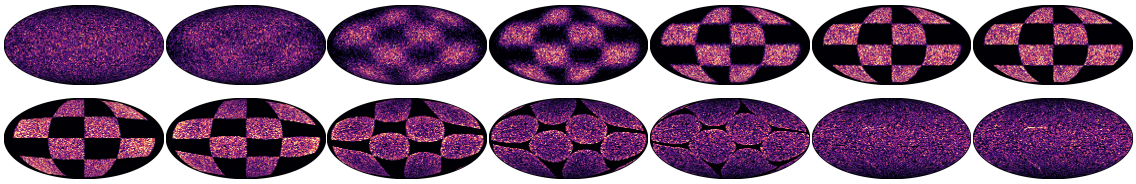


Figure 3.1: PPM on a manifold (sphere): the trained flow ψ_t is pushing noise $x \sim p_0$ to data $\psi_t(x)$ (top, from left $t = 0$ to right $t = 1$); and the reverse flow taking data $x \sim p_{\text{data}}$ to noise (bottom).

$p_t = p(t, \cdot)$. We are now ready to shift our attention to the entire path in time and view CNFs as probability path generators.

Definition 1. *We say that a flow* ψ_t generates *a probability density path* $q \in \mathfrak{P}(\mathcal{M})$ *if for all* $t \in [0, 1]$

$$q_t = \psi_{t*} q_0, \text{ or equivalently } \psi_t^* q_t = q_0, \tag{3.1}$$

With this perspective in mind, a natural question is: *can one train a CNF to generate a target probability path* $p \in \mathfrak{P}(\mathcal{M})$*?* The answer, as the name of the chapter hints, is yes, and perhaps the most interesting aspect of this different perspective is that the reduction of degrees of freedom by supervising the entire probability path, and not just the final density, q_1, will enable simulation-free training. With this prospect, we will now show how to match a CNF to a target probability path.

We approach probability path matching in a similar manner to how we built the generative modeling optimization formulation in equation 2.2, except now we wish to match a continuous evolution of density functions in time and not a single target density:

$$\min_{\theta} \quad \mathrm{D}(p \,\|\, q) \tag{3.2a}$$

$$\text{s.t.} \quad q_t = \psi_{t*} p_0, \quad t \in [0, 1], \tag{3.2b}$$

where D is a probability divergence between probability density *paths*. That is, for density paths $p, q \in \mathfrak{P}(\mathcal{M})$, $\mathrm{D}(p \,\|\, q) \geq 0$, and $\mathrm{D}(p \,\|\, q) = 0$ iff $p_t \equiv q_t$ for all $t \in [0, 1]$. To comply with the generative modeling goal, a typical target path p, will satisfy the boundary conditions: (i) for $t = 0$, p_0 will be some simple source distribution, e.g., a Gaussian; and (ii) for $t = 1$, p_1 will approximate p_{data}, e.g., delta functions on data samples.

At first glance, solving the problem 3.2 may seem more challenging than problem 2.2 and adapting existing CNF approaches to optimize equation 3.2 would require evaluating q_t, which is provided only through solutions to an ODE (see also the discussion in Section §3.4.1). Instead, we construct a novel divergence d, called the Probability Path Divergence (PPD), that does not require sampling of q or enforcing equation 3.2b explicitly, sidestepping the need for ODE simulation and backpropagation during training.

3.3.1 Logarithmic Mass Conservation

As a first step in constructing the PPD we derive a Partial Differential Equation (PDE) involving the log density path $\log p$ and a velocity field v, such that it is satisfied iff the flow

ψ_t, defined by v, generates p. We name this equation the Logarithmic Mass Conservation (LMC) formula.

Theorem 1. *Consider a flow $\psi_t : \mathcal{M} \to \mathcal{M}$ defined by a smooth, time-dependent velocity field $v \in \mathfrak{X}(\mathcal{M})$ as in equation 2.24, and a probability density path $p \in \mathfrak{P}(\mathcal{M})$. Then p is generated by ψ_t, i.e.,*

$$p_t = \psi_{t*} p_0, \quad \forall t \in [0, 1] \tag{3.3}$$

if and only if the LMC formula holds over $[0, 1] \times \mathcal{M}$:

$$\partial_t \log p_t + \langle \nabla \log p_t, v \rangle + \operatorname{div}(v) = 0 \tag{3.4}$$

The LMC formula can be proved with the aid of the mass conservation formula, also known as the continuity equation and equivalent to equation 3.4 [Vil08]:

$$\partial_t p_t + \operatorname{div}(p_t v) = 0, \tag{3.5}$$

where div denotes the divergence operator over the manifold $\mathcal{M}$. We assumed $p > 0$ and therefore dividing both sides by p_t leads to

$$\frac{\partial_t p_t}{p_t} + \frac{\langle \nabla p_t, v \rangle + p_t \operatorname{div}(v)}{p_t} = 0,$$

where we also used the fact that $\operatorname{div}(fv) = \langle \nabla f, v \rangle + f \operatorname{div}(v)$. Finally noting that $\partial_t \log p_t = \frac{\partial_t p_t}{p_t}$, and $\nabla_x \log p_t = \frac{\nabla_x p_t}{p_t}$ we get that equation 3.4 is equivalent to equation 3.5. See Appendix A.1.1 for more details. The benefit of using the LMC formula over the standard mass conservation formula is that it is formulated directly in terms of the log probability $\log p_t$, which reduces numerical issues for high dimensions.

3.3.2 Probability Path Divergence

Plugging a fixed target path $p \in \mathfrak{P}(\mathcal{M})$ in the LMC formula (equation 3.4) provides a necessary and sufficient condition for v to generate p via a CNF. Motivated by this observation, we define a family of probability path divergences (PPD), parameterized by an integer $\ell \geq 1$, comparing $p, q \in \mathfrak{P}(\mathcal{M})$ where $q_t = \psi_{t*} p_0$:

$$\mathrm{D}_\ell(p \,\|\, q) = \mathbb{E}_{t, x \sim p_t} \left| \partial_t \log p_t + \langle \nabla \log p_t, v \rangle + \operatorname{div}(v) \right|^\ell, \tag{3.6}$$

and t is distributed over $[0, 1]$, e.g., uniform $t \sim \mathcal{U}[0, 1]$. $\mathrm{D}_\ell(p \,\|\, q) \geq 0$ by construction, and Theorem 1 implies that $\mathrm{D}_\ell(p \,\|\, q) = 0$ iff $p_t \equiv q_t$ for all $t \in [0, 1]$.

Using this path divergence in the PPM problem (equation 3.2), we arrive at the following instantiation:

$$\min_\theta \mathbb{E}_{t, x \sim p_t} \left| \partial_t \log p_t + \langle \nabla \log p_t, v_\theta \rangle + \operatorname{div}(v_\theta) \right|^\ell \tag{3.7}$$

where v_θ is the learnable velocity field, parameterized by a neural network, defining the flow ψ_t generating q_t. Importantly, evaluating the PPD $\mathrm{D}_\ell(p \,\|\, q)$ and its derivatives with respect to θ does not require access to q and ψ_t and thus allows simulation-free training.

To further justify the PPD, we show that the PPD family bounds a family of standard divergences of probability densities in the following theorem (proof in Appendix A.1.2):

Theorem 2. *Consider paths $p, q \in \mathfrak{P}(\mathcal{M})$ where q is generated by a flow $\psi_t : \mathcal{M} \to \mathcal{M}$, and $q_0 = p_0$. Then for all $T \in [0, 1]$*

$$\mathrm{D}_\ell(p \,\|\, q)^{\frac{1}{\ell}} \geq \mathrm{D}_f(p_T \,\|\, q_T), \tag{3.8}$$

where $\mathrm{D}_\ell(p \,\|\, q)$ is an f-divergence with:

$$f(r) = \begin{cases} |r - 1| & \ell = 1 \quad \textit{(total variation)} \\ \ell\left(1 - r^{\frac{1}{\ell}}\right) & 1 < \ell < \infty \quad (\alpha) \\ -\log r & \ell = \infty \quad \textit{(reverse KL).} \end{cases}$$

Theorem 2 shows that the PPD D_ℓ bounds the respective f-divergences of p_T and q_T for all times $T \in [0, 1]$. Figure 3.2 visualizes four instances of f corresponding to different choices of ℓ. Note that with the exception of $\ell = 1$, all f are differentiable and have the same derivative at 1, which means they have similar values and derivatives when evaluating the divergence of nearby probability densities. As $\ell \to \infty$ we can see the f functions gets close to the $-\log r$ limit.

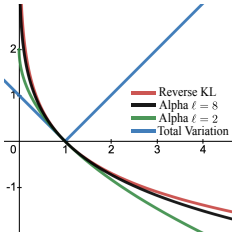


Figure 3.2: f instances, see Theorem 2.

Specifically, in the $\ell = \infty$ case of Theorem 2, we mean that the inequality equation 3.8 holds in the limit as $\ell \to \infty$, or more precisely,

$$\liminf_{\ell \to \infty} \mathrm{D}_\ell(p \,\|\, q)^{1/\ell} \geq \mathrm{D}_f(p_T \,\|\, q_T),$$

where we also assume that $\mathrm{D}_f(p_T \,\|\, q_T) < \infty$.

3.3.3 Target paths

The last ingredient needed for defining the probability path divergence (equation 3.6) is the target path $p \in \mathfrak{P}(\mathcal{M})$. In our framework, p should be defined satisfying the following requirement:

(i) p_0 is pure noise, e.g., a standard Gaussian or uniform.

(ii) p_1 approximates the unknown data distribution p_{data}.

(iii) We have an efficient generation procedure for $x \sim p_t$.

(iv) We have an approximation procedure for the time (∂_t) and space (∂_x) derivatives of $\log p_t(x)$.

Note that these requirements do not mean we know of an SDE, generating random variables distributed as p_t, nor a PDE (Fokker-Planck) with p_t as its solution. In fact, below we construct paths for which an SDE/PDE characterization is not known. In that context the target paths we consider are general; see discussion in Section 3.4.2.

In the following we construct target paths $p \in \mathfrak{P}(\mathcal{M})$ for several manifolds of interest. At the base of our construction is a kernel $p_\tau(x|y)$, namely a probability density in $x \in \mathcal{M}$, centered at $y \in \mathcal{M}$, with scale $\tau > 0$. We define our (ideal) target path $p \in \mathfrak{P}(\mathcal{M})$ by

$$p_t(x) = \int_{\mathcal{M}} p_\tau(x \,|\, \gamma_t(y)) p_{\text{data}}(y) dV_y, \tag{3.9}$$

where $\tau = \tau(t)$, $t \in [0,1]$, is a time-dependent scale function, and $\gamma : [0,1] \times \mathcal{M} \to \mathcal{M}$ is some differentiable in t map. In practice we do not know p_{data}, rather, we have an empirical sample $\{y_i\}_{i=1}^m$, drawn i.i.d. from p_{data}. Therefore we use the following approximation of equation 3.9

$$p_t(x) = \frac{1}{m} \sum_{i=1}^{m} p_\tau(x \,|\, \gamma_t(y_i)). \tag{3.10}$$

Note, that if we know how to compute or approximate $\log p_\tau(x|\gamma_t(y_i))$ then $\log p_t(x)$, required for the computation of the PPD, has the form

$$\log p_t(x) = \texttt{logsumexp}\{\log p_\tau(x|\gamma_t(y_i)\}_{i=1}^m - \log m.$$

Depending on the type of manifold, we consider two basic target path constructions that differ in their prior p_0: *Unimodal*, where the prior probability p_0 is centered around a single designated point in $\mathcal{M}$. Unimodal prior is mainly suitable to non-compact manifolds with infinite volume such as Euclidean or hyperbolic spaces. *Uniform*, where the prior p_0 is the uniform density over $\mathcal{M}$. A uniform prior is suitable to compact manifolds such as spheres.

Unimodal prior. Let $o \in \mathcal{M}$ be some designated point, $\sigma_0, \sigma_1 \geq 0$ initial and target scales. We define p according to equation 3.10 by making the choices:

$$\gamma_t(y) = \exp_o(t \log_o y) \quad , \quad \sigma(t) = \sigma_0^{1-t} \sigma_1^t \tag{3.11}$$

where $\tau = \sigma$ is the scaling function, and $\gamma_t(y)$ moves y to the center o along a geodesic (we assume the Riemannian $\exp_o, \log_o$ are defined in a sufficiently large neighborhood of o and $T_o\mathcal{M}$). With these choices, p_t starts with a single mode density p_0, centered at $o \in \mathcal{M}$, and then splits the unit mass, moving each $\frac{1}{m}$ part towards the empirical sample y_i along a geodesic while concentrating the density.

Euclidean. Let us instantiate the unimodal path for the Euclidean space, $\mathcal{M} = \mathbb{R}^d$, with the standard metric $\langle v, u \rangle = v^T u$, where $v, u \in \mathbb{R}^d$ are (always) column vectors. Our kernel in this case is the Gaussian, $p_\sigma(x|y) = \mathcal{N}(x|y, \sigma^2 \boldsymbol{I})$ with mean $y \in \mathbb{R}^d$ and covariance $\sigma^2 \boldsymbol{I}$. Furthermore, for $o \in \mathbb{R}^d$, $\exp_o(t \log_o y_i) = o + t(y_i - o) = (1-t)o + t y_i$. Therefore,

equation 3.10 takes the form

$$p_t(x) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{N}(x \,|\, (1-t)o + ty_i, \sigma^2 \boldsymbol{I}) \tag{3.12}$$

and we take $\sigma_0 = 1$ to represent a standard Gaussian prior, i.e., $\sigma(t) = \sigma_1^t$, and $\sigma_1 > 0$ is the (only) hyper-parameter.

Uniform prior. In this family of paths we consider compact manifolds $\mathcal{M}$ and start from the uniform density p_0. We assume in this case we have a kernel $p_\kappa(x|y)$ such that there exists a finite $\kappa_0 \geq 0$, for which $p_{\kappa_0}(x|y) \equiv |\mathcal{M}|^{-1}$ for all $y \in \mathcal{M}$, i.e., p_{κ_0} represents the uniform density. One way to construct such a kernel on compact submanifolds of $\mathbb{R}^{d+1}$, $\mathcal{M} \subset \mathbb{R}^{d+1}$, is by restricting an Euclidean Gaussian in $\mathbb{R}^{d+1}$ to $\mathcal{M}$; we discuss such a construction on the sphere below. In this case we define the target path using equation 3.10 again by making the choices

$$\gamma_t(y) = y \quad , \quad \kappa(t) = (1 - \kappa_0 + \kappa_1)^t + \kappa_0 - 1 \tag{3.13}$$

where $\tau = \kappa$ is the scaling function, and $\gamma_t(y)$ leaves samples at their original location.

Sphere. We instantiate the uniform prior paths to the unit spheres $\mathcal{M} = \mathcal{S}^d \subset \mathbb{R}^{d+1}$ with the induced metric from the Euclidean $\mathbb{R}^{d+1}$. The von Mises-Fisher (vMF) kernel [Mar14] is:

$$p_\kappa(x|y) = c_d(\kappa) \exp(\kappa x^T y), \tag{3.14}$$

where $c_d(\kappa)$ is the normalization constant detailed in Appendix B.1.2. vMF can be seen as a restricted Gaussian $\exp(-\kappa \|x - y\|_2^2)$ to the unit sphere $x, y \in \mathcal{S}^d$ with the relevant normalization constant. For $\kappa = 0$, $p_0(x|y)$ is uniform over the sphere for all $y \in \mathcal{S}^d$. Hence we take $\kappa_0 = 0$, which leaves $\kappa(t) = (1+\kappa_1)^t - 1$, and $\kappa_1 > 0$ is the (only) hyper-parameter in this case. The target path takes the form

$$p_t(x) = \frac{1}{m}\sum_{i=1}^{m} p_\kappa(x|y_i) \tag{3.15}$$

Paths on products of manifolds. We conclude the section with generalizing the target path construction to product of manifolds. Let $\mathcal{M} = \mathcal{M}^1 \times \ldots \times \mathcal{M}^N$. Each point $x \in \mathcal{M}$ is represented as a tuple $x = (x^1, \ldots, x^N)$, where $x^j \in \mathcal{M}^j$. For example, in robotics, a robot's state can be represented by the sequence of locations and/or rotations of its joints, i.e., each $\mathcal{M}^j$ is either a sphere ($\mathcal{S}^3$ for 3D rotations represented as quaternions; $\mathcal{S}^1$ for 2D rotations) or an Euclidean space (representing positions). Let p_{τ^j} be a kernel defined in $\mathcal{M}^j$, and γ^j is a deformation of $\mathcal{M}^j$. For example, for $\mathcal{M}^j$ being the Euclidean plane or a sphere we can use the above definitions for kernels p_{τ^j}. Let $\{y_i\}_{i=1}^m \subset \mathcal{M}$ be i.i.d. samples from p_{data} over $\mathcal{M}$. We define the kernel for $\mathcal{M}$ by

$$p_\tau(x|y_i) = \prod_{j=1}^{N} p_{\tau^j}(x^j|\gamma_t^j(y_i^j)) \tag{3.16}$$

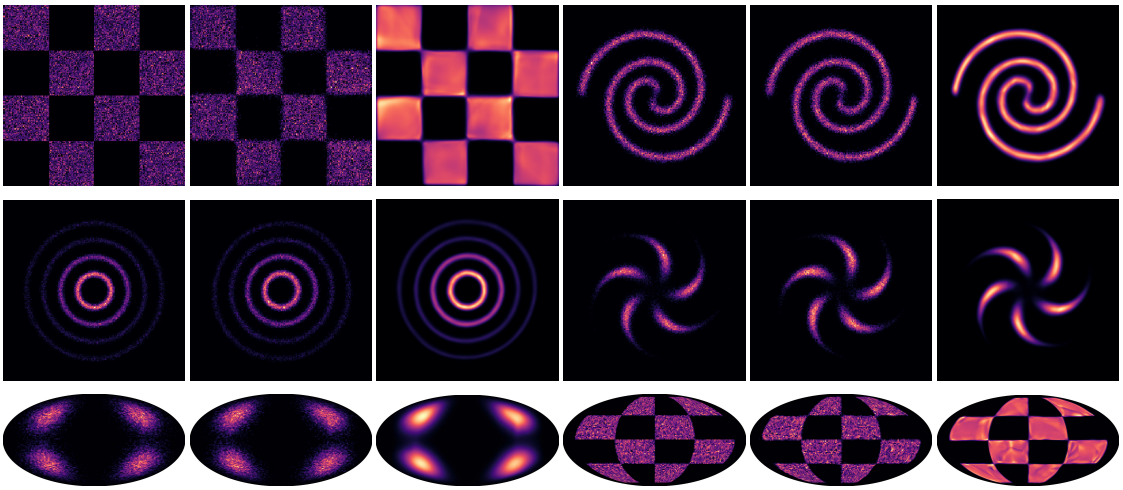


Figure 3.3: 2D toy densities. Each triplet shows (left to right): data samples, generate samples $x \sim q_1$, and learned model density q_1.

We note that $p_\tau(x|y)$ is a probability density in $x \in \mathcal{M}$, and if $p_{\tau^j}(x^j|y^j)$ is concentrated (as a function of $x^j \in \mathcal{M}^j$) around y^j for all j, then $p_\tau(x|y)$ is concentrated (as a function of $x \in \mathcal{M}$) around y. Lastly, and use equation 3.10 again to define out target path $p \in \mathfrak{P}(\mathcal{M})$. Further implementation details for the velocity field v_θ are in Appendix B.1.1.

3.4 Previous works

3.4.1 Relations to existing CNF models

The LMC formula (equation 3.4) is a linear first order PDE in $\log p_t$. Solving it using the method of characteristics [Eva97] provides a simple proof of the Instantaneous Change of Variables Theorem from [Che+18] and generalizes it to the manifold setting. Indeed, using the chain rule and the LMC we have

$$\begin{aligned} \partial_t \left[\log q_t(\psi_t)\right] &= \partial_t \log q_t(\psi_t) + \langle \nabla_x \log q_t(\psi_t), v(t, \psi_t) \rangle \\ &= -\operatorname{div} v(t, \psi_t), \end{aligned} \tag{3.17}$$

where $\partial_t \left[\log q_t(\psi_t)\right]$ denotes the total derivative w.r.t. t. Training a neural ODE by maximizing the likelihood of the data points $x_i \in \mathcal{M}$ entails computing $\log q_t(x_i)$ and its derivatives w.r.t. the parameters of the velocity field v_t. Using the characteristic method [Che+18; Lou+20; MN20; FF20] this amounts to solving an ODE for $(\log q_t(\psi_t), \psi_t)$ (equations 2.24 and 3.17) and differentiating the solution (which involves another ODE solve). In contrast, minimizing the PPD does not require solving an ODE during training.

Moser Flow (MF) [Roz+21] suggests to train a CNF by formulating the model density as $q_1 = p_0 - \operatorname{div}(u)$, where u is time independent velocity field over $\mathcal{M}$. Its relation to our method can be seen by making the choice $q_t = (1-t)p_0 + tq_1$, where p_0 and q_1 are prior and model probability densities, respectively. Indeed, plugging this path in the mass

conservation equation (equation 3.5) gives

$$q_1 - p_0 + \operatorname{div}(q_t v) = 0$$

which directly leads to MF by plugging $u = p_t v$ as a time independent solution to this equation. Although MF also avoids solving an ODE during training and generalizes to manifolds, it incorporates an additional loss term for keeping the model density q_1 positive; this loss term has high variance and does not scale to high dimensions. Furthermore, MF models probabilities rather than log-probabilities, which also hinders modeling high dimensional densities. Lastly, MF models a particular probability path (convex combinations of prior and model), while our framework can match more general paths.

3.4.2 Relations to score-based generative models

Another body of related work concerns score-based generative models [Son+21b], which encompasses both methods in score-matching [HD05; Vin11; SE19] and diffusion models [Soh+15; HJA20]. Score-based generative models use a stochastic differential equation (SDE) to define a time-dependent path between the prior and data densities as a solution to the Fokker Planck Equation (i.e., particles under Brownian motion). These models are typically trained by minimizing a weighted combination of score-matching losses, with recent extensions also exploring surrogate objectives based on variational approximations to the log-likelihood [Son+21a; HLC21; VKK21; Kin+21].

Consider the arguably simplest SDEs, describing Brownian motion over $\mathcal{M}$. The corresponding probability kernel $p_t(x|y)$ is the fundamental solution to the heat equation $\partial_t p = \Delta p$, where Δ is the Laplace-Beltrami operator on the manifold $\mathcal{M}$. Solutions to the heat equation are known in very few cases [Pen06], and even for the sphere the solution is only known as an infinite series of Legendre polynomials [TP01]. Therefore using the SDE framework on manifolds will often require some numerical solutions to the relevant SDE/ODE. In contrast, our LMC-based formulation provides the flexibility to specify arbitrary target probability paths between the prior and data densities. On the sphere for example, we use closed form paths defined by vMF distributions. For sampling, solving an ODE is generally easier than solving an SDE as ODE solvers have higher asymptotic convergence rates. For example, Euler's method has order 1 for ODE and only 0.5 for SDE [KPS12]. Furthermore, ODEs have simple higher order solvers like Runga-Kutta methods [DP80] with widely used open-source implementations.

3.5 Experiments

We have tested the PPM framework with the PPD for training CNFs on low and moderately high dimensional manifold data. In all experiments we generate the target path p according to Section 3.3.3 with input data samples $\{y_i\}_{i=1}^m \subset \mathcal{M}$. In general, we have found PPM to facilitate faster training of CNFs with larger models, often producing state of the art sampling and density estimation.

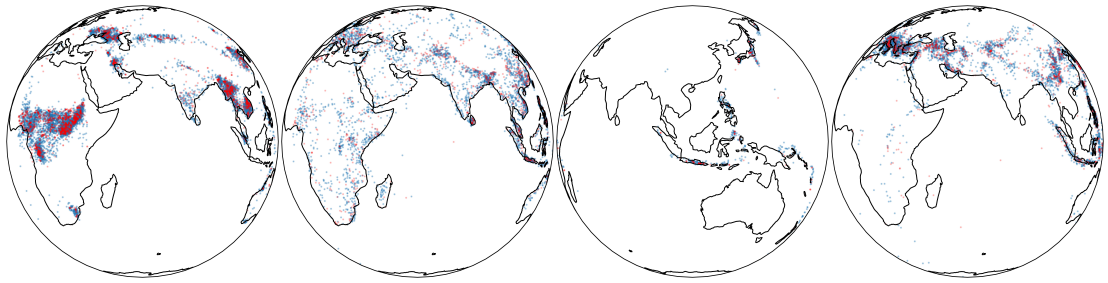


Figure 3.4: Earth and Climate dataset: generated samples from the trained PPM in blue, test samples in red. See table 3.1 for quantitative results.

Table 3.1: Negative log likelihood scores (↓) on the Earth and Climate Dataset [MN20].

Dataset	Earthquake	Flood	Fire	Volcano
Mixture vMF	0.59±0.01	1.09±0.01	−0.23±0.02	−0.31±0.07
Stereographic	0.43±0.04	0.99±0.04	−0.40±0.06	−0.64±0.20
Riemannian [MN20]	0.19±0.04	0.90±0.03	−0.66±0.05	−0.97±0.15
Moser Flow [Roz+21]	−0.09±0.02	0.62±0.04	−1.03±0.03	−2.02±0.42
PPM (ours)	−0.38±0.01	0.25±0.02	−1.40±0.02	−2.38±0.16

3.5.1 Toy densities on $\mathbb{R}^2$ and $\mathcal{S}^2$

In the first experiment we worked with samples drawn from standard toy distributions on the 2D Euclidean plane and sphere. For the Euclidean data we used the target path p defined in equation 3.12 with $p_0 \sim \mathcal{N}(x|0, \boldsymbol{I})$, the standard normal distribution, and $\sigma_1 = 0.01$. For the spherical data we used the target path p as defined in equation 3.15 with $\kappa_1 = 5000$. We used MLP of 3 layers of 256 neurons for the $\mathbb{R}^2$ data, and 6 layers of 512 neurons for $\mathcal{S}^2$. We used PPD with $\ell = 1$. Figure 3.3 depicts the data samples y_i along side samples generated from the learned model, and the model densities. Note the high similarity between the learned and ground truth (GT) densities; for sphere visualizations we use Mollweide projection.

3.5.2 Earth and climate dataset

In this experiment we considered the Earth and Climate dataset curated in [MN20]. This dataset contains locations of earthquakes, floods, fires, and volcano eruptions on earth, represented as point locations on the 2D sphere, $\mathcal{S}^2 \subset \mathbb{R}^3$. The target path p is defined as in equation 3.15 with $\kappa_1 = 55\text{K}$ (best out of $\kappa_1 \in \{5\text{K}, 55\text{K}, 500\text{K}\}$). We used the same architecture used in [Roz+21], a MLP with 6 layers of 512 neurons, PPD order $\ell = 2$. Table 3.1 depicts the negative log likelihoods (NLLs) scores, where PPM improves state of the art by a large margin, where the runner-up is Moser Flow [Roz+21]. Riemannian CNF and other baselines are taken from [MN20]. Figure 3.4 visualizes generated samples (blue) and test data samples (red).

3.5.3 Higher dimensional spheres

In this experiment we test the scaling of PPM to higher dimensional manifold data. We construct a family of challenging probability distributions, denoted r_k, on $\mathcal{S}^{15}$ and compare PPM to several baselines. We start by defining r_k over $\mathcal{S}^{15} \subset \mathbb{R}^{16}$: Henceforth, denote $d = 15$, and consider an orthogonal set $v_1, \ldots, v_k$, where $1 \leq k \leq d+1$. Let $s(x) = \prod_{i=1}^{k} \text{sign}(x^T v_i)$. Define the probability density:

$$r_k(x) = \frac{2}{|\mathcal{S}^d|} \begin{cases} 1 & \text{if } s(x) = 1 \\ 0 & \text{if } s(x) = -1. \end{cases} \tag{3.18}$$

To see r_k is indeed a probability density, note that the transformation $x = (x_1, \ldots, x_{d+1}) \mapsto (-x_1, \ldots, x_{d+1})$ is a volume preserving transformation of $\mathcal{S}^d$ and maps the set $\Omega_+ = \mathcal{S}^d \cap \left\{x \in \mathbb{R}^{d+1} | s(x) = 1\right\}$ to $\Omega_- = \mathcal{S}^d \cap \left\{x \in \mathbb{R}^{d+1} | s(x) = -1\right\}$, and vise versa. This means that $\int_{\Omega_+} dV_x = \int_{\Omega_-} dV_x$ and since $\mathcal{S}^d = \Omega_+ \cup \Omega_-$ we have that $\int_{\Omega_+} dV_x = \left|\mathcal{S}^d\right|/2$. Generating samples from r_k can be done by randomizing a uniform sample x over $\mathcal{S}^d$, if $s(x) = 1$, keep x, otherwise take $(-x_1, x_2, \ldots, x_{d+1})$. Figure 3.6-left depicts several examples of this density by visualizing random $\mathcal{S}^2$ cuts in $\mathcal{S}^{15}$; as k increases the complexity of density increases. We created datasets for $k = 2, 3, 4$ with 45K train samples and 5K test samples.

For baselines we use: vMF mixture models (vMF-MM) with 1K and 10K centers randomized from the training data, and scaling κ was chosen to be the optimal for the test set. This was done to compare to the best possible vMF-MM model. Furthermore, we compared to a version of manifold CNF [Lou+20; MN20; FF20]: We consider the stereographic projection of the sphere $\Phi : \mathbb{R}^d \to \mathcal{S}^d$, and used FFJORD [Gra+19] code adapted to the spherical case, denoted as S-FFJORD. In this baseline, computing log probabilities over the sphere is done by correcting for the stereographic projection, $\log p(\Phi(u)) = \log p(u) - \frac{1}{2} \log \det(D_\Phi(u)^T D_\Phi(u))$, where $u \in \mathbb{R}^d$, $\log p(u)$ is the Euclidean log probability learned by FFJORD, $D_\Phi(u) \in \mathbb{R}^{(d+1)\times d}$ is the matrix of partials of Φ. Table 3.2 reports the NLL scores of PPM and the baselines across this dataset. Figure 3.6-right depicts an example of random $\mathcal{S}^2$ cut of $\mathcal{S}^{15}$ for the $k = 3$ case.

Table 3.2: NLLs on $\mathcal{S}^{15}$.

Method	$k=2$	$k=3$	$k=4$
vMF-MM	1.23	1.31	1.33
S-FFJORD	0.77	0.97	1.04
PPM (ours)	0.73	0.83	0.95

The above results are reported using an MLP with 3 layers of 64 neurons for PPM and an equivalent architecture of S-FFJORD, with both methods running for about 4K seconds. In Figure 3.5 we compared typical epoch running times for this and larger architecture types for PPM and FFJORD training. Note that the time difference (in log scale) further increases for larger architectures.

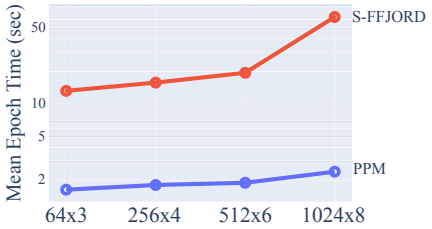


Figure 3.5: Timings.

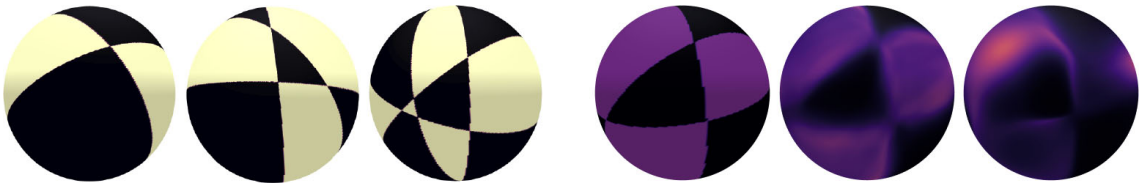


Figure 3.6: Left triplet shows the densities r_k for $k = 2, 3, 4$ on random cuts $\mathcal{S}^2 \subset \mathcal{S}^{15}$; right triplet visualizes the case $k = 3$ (on a different random cut) from Table 3.2 with PPM model density in the middle, and S-FFJORD density on the right.

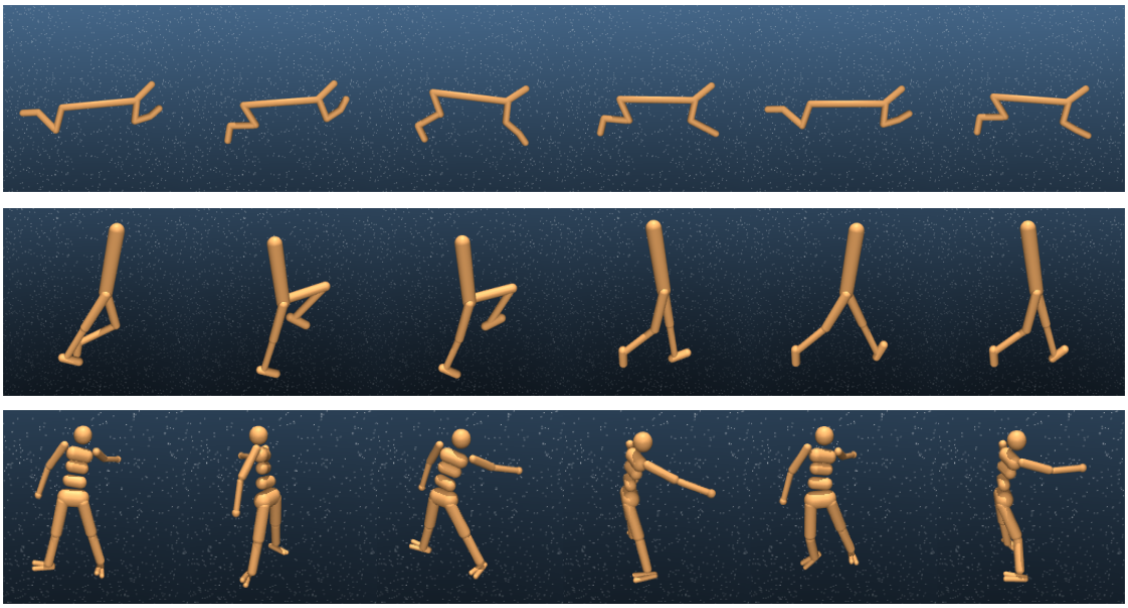


Figure 3.7: Uncurated samples computed with the trained PPM on product manifolds representing the robot's state space: Cheetah (top), Walker (middle), and Humanoid (bottom).

3.5.4 Product of manifolds - Robotics

In the last experiment we worked with robotics data generated with the physics and reinforcement learning engine MuJoCo [Tas+20]. For each of the three robot types, Walker (2D), Cheetah (2D), and Humanoid (3D), we randomized 17.5K samples from 50 simulated trajectories consisting of 500 observation each. The state space for the 2D robots is modeled as the product manifold $\mathcal{M} = \mathbb{R}^3 \times (\mathcal{S}^1)^6$, where $\mathbb{R}^3$ represents position, and $\mathcal{S}^1$ represents 2D rotations of a single joint. The state space for the 3D robot is $\mathcal{M} = \mathbb{R}^3 \times (\mathcal{S}^1)^8 \times (\mathcal{S}^3)^6$, where $\mathcal{S}^3$ represents 3D rotations of a joint (via quaternions). We used the target path p on the product manifold as described in Section 3.3.3. For each robot type, Figure 3.7 depicts uncurated samples from the trained PPM, and Figure 3.8 shows a path of noise to data, i.e., $\psi_t(x)$, $t \in [0, 1]$, where $x \sim p_0$. The generated samples are qualitatively similar to the data samples. More examples can be found in the published paper [Ben+22].

Figure 3.8: Noise to data paths computed with the trained PPM on product manifolds representing robot's state space: Cheetah (top), Walker (middle), and Humanoid (bottom).

Chapter 4

Flow Matching

In this chapter, we make our final step, further restricting the degrees of freedom in CNF training - reducing the generative modeling problem with CNFs to a regression problem. We propose the Flow Matching (FM) objective, a simple and intuitive training objective to regress onto a target velocity field that generates a desired probability path. We first show that we can construct such target velocity fields through per-example (i.e., conditional) formulations. Then, inspired by denoising score matching, we show that a per-example training objective, termed Conditional Flow Matching (CFM), provides equivalent gradients and does not require explicit knowledge of the intractable target velocity field. This chapter was published as [Lip+23].

4.1 Motivation and Contributions

The recent influx of amazing advances in generative modeling, e.g., for image generation [Ram+22; Rom+22b], is mostly facilitated by the scalable and relatively stable training of diffusion-based models [HJA20; Son+21b]. However, the restriction to simple diffusion processes leads to a rather confined space of sampling probability paths, resulting in very long training times and the need to adopt specialized methods (e.g., [SME21; ZC22]) for efficient sampling.

Although CNFs are capable of modeling arbitrary probability path and are in particular known to encompass the probability paths modeled by diffusion processes [Son+21a]. However, aside from diffusion that can be trained efficiently via, e.g., denoising score matching [Vin11], no scalable CNF training algorithms are known. Indeed, maximum likelihood training (e.g., [Gra+19]) requires expensive numerical ODE simulations, while existing simulation-free methods either involve intractable integrals [Roz+21] or biased gradients [Ben+22].

We propose Flow Matching (FM), an efficient simulation-free approach to training CNF models, allowing the adoption of general probability paths to supervise CNF training. Importantly, FM breaks the barriers for scalable CNF training beyond diffusion, and sidesteps the need to reason about diffusion processes to directly work with probability paths.

In particular, we propose the Flow Matching objective (Section 4.3), a simple and intuitive training objective to regress onto a target velocity field that generates a desired probability path. We first show that we can construct such target velocity fields through per-example (i.e., conditional) formulations. Then, inspired by denoising score matching, we show that a per-example training objective, termed Conditional Flow Matching (CFM), provides equivalent gradients and does not require explicit knowledge of the intractable target velocity field. Furthermore, we discuss a general family of per-example probability paths (Section 4.4) that can be used for Flow Matching, which subsumes existing diffusion paths as special instances. Even on diffusion paths, we find that using FM provides more robust and stable training, and achieves superior performance compared to score matching. Furthermore, this family of probability paths also includes a particularly interesting case: the velocity field that corresponds to an Optimal Transport (OT) displacement interpolant [McC97]. We find that conditional OT paths are simpler than diffusion paths, forming straight line trajectories whereas diffusion paths result in curved paths. These properties seem to empirically translate to faster training, faster generation, and better performance.

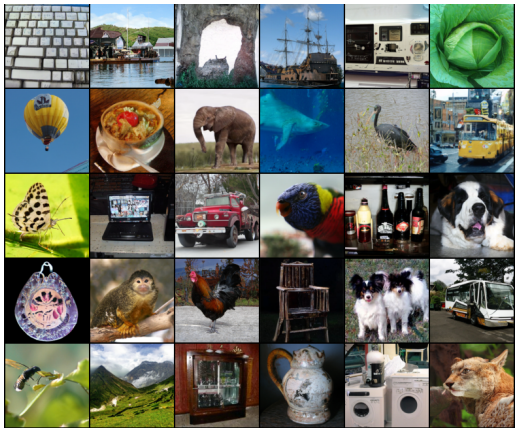


Figure 4.1: Unconditional ImageNet-128 samples of a CNF trained using Flow Matching with Optimal Transport probability paths.

We empirically validate Flow Matching and the Optimal Transport paths on ImageNet, a large and highly diverse image dataset. We find that we can easily train models to achieve favorable performance in both likelihood estimation and sample quality amongst competing diffusion-based methods. Furthermore, we find that our models produce better trade-offs between computational cost and sample quality compared to prior methods. Figure 4.1 depicts selected unconditional ImageNet 128×128 generated samples.

4.2 Notations

In this chapter, we only deal with the Euclidean case. We assume $\mathbb{R}^d$ is the data space, and to simplify cumbersome marginalization integrals, we denote the data distribution by $q \coloneqq p_{\text{data}}$, not to be confused with the previous chapter where q denoted the probability path generated by the flow.

Let x_1 denote a random variable distributed according to some unknown data distribution $q(x_1)$. We assume we only have access to data samples from $q(x_1)$ but have no access to the density function itself. Furthermore, we let p_t denote a target a probability path such that $p_0 = p$ is a simple distribution, e.g., the standard normal distribution $p(x) = \mathcal{N}(x|0, I)$, and let p_1 be approximately equal in distribution to q. We will later discuss how to construct such target paths following similar principles to [Ben+22].

4.3 Flow Matching

The Flow Matching objective is then designed to match this target probability path, which will allow us to flow from p_0 to p_1. Given a target probability density path $p_t(x)$ and a corresponding velocity field $u_t(x)$, which generates $p_t(x)$, we define the Flow Matching (FM) objective as

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t,p_t(x)}\|v_t(x) - u_t(x)\|^2, \tag{4.1}$$

where θ denotes the learnable parameters of the CNF velocity field v_t (as defined in §2.3), $t \sim \mathcal{U}[0,1]$ (uniform distribution), and $x \sim p_t(x)$. Simply put, the FM loss regresses the velocity field u_t with a neural network v_t. Upon reaching zero loss, the learned CNF model will generate $p_t(x)$ (See Definition 1).

Flow Matching is a simple and attractive objective, but naïvely on its own, it is intractable to use in practice since we have no prior knowledge for what an appropriate p_t and u_t are. There are many choices of probability paths that can satisfy $p_1(x) \approx q(x)$, and more importantly, we generally don't have access to a closed form u_t that generates the desired p_t. In this section, we show that we can construct both p_t and u_t using probability paths and velocity fields that are only defined *per sample*, and an appropriate method of aggregation provides the desired p_t and u_t. Furthermore, this construction allows us to create a much more tractable objective for Flow Matching.

4.3.1 The Marginalization Trick

A simple way to construct a target probability path is via a mixture of simpler probability paths similar to [Ben+22]. Given a particular data sample x_1 we denote by $p_t(x|x_1)$ a *conditional probability path* such that it satisfies $p_0(x|x_1) = p(x)$ at time $t = 0$, and we design $p_1(x|x_1)$ at $t = 1$ to be a distribution concentrated around $x = x_1$, e.g., $p_1(x|x_1) = \mathcal{N}(x|x_1, \sigma^2 I)$, a normal distribution with x_1 mean and a sufficiently small standard deviation $\sigma > 0$. Marginalizing the conditional probability paths over $q(x_1)$ give rise to *the marginal probability path*

$$p_t(x) = \int p_t(x|x_1)q(x_1)dx_1, \tag{4.2}$$

where in particular at time $t = 1$, the marginal probability p_1 is a mixture distribution that closely approximates the data distribution q,

$$p_1(x) = \int p_1(x|x_1)q(x_1)dx_1 \approx q(x). \tag{4.3}$$

Interestingly, we can also define a *marginal velocity field*, by "marginalizing" over the conditional velocity fields in the following sense (we assume $p_t(x) > 0$ for all t and x):

$$u_t(x) = \int u_t(x|x_1)\frac{p_t(x|x_1)q(x_1)}{p_t(x)}dx_1, \tag{4.4}$$

where $u_t(\cdot|x_1) : \mathbb{R}^d \to \mathbb{R}^d$ is a conditional velocity field that generates $p_t(\cdot|x_1)$. It may not

seem apparent, but this way of aggregating the conditional velocity fields actually results in the correct velocity field for modeling the marginal probability path.

Our first key observation is this:

The marginal velocity field (equation 4.4) generates the marginal probability path (equation 4.2).

This provides a surprising connection between the conditional VFs and the marginal VF. This connection allows us to break down the unknown and intractable marginal VF into simpler conditional VFs, which are much simpler to define as these only depend on a single data sample. We formalize this in the following theorem (proof in §A.2.1):

Theorem 3. *Given velocity fields $u_t(x|x_1)$ that generate conditional probability paths $p_t(x|x_1)$, for any distribution $q(x_1)$, the marginal velocity field u_t in equation 4.4 generates the marginal probability path p_t in equation 4.2, i.e., u_t and p_t satisfy the continuity equation (equation 2.28).*

4.3.2 Conditional Flow Matching

Unfortunately, due to the intractable integrals in the definitions of the marginal probability path and VF (equations 4.2 and 4.4), it is still intractable to compute u_t, and consequently, intractable to naïvely compute an unbiased estimator of the original Flow Matching objective. Instead, we propose a simpler objective, which surprisingly will result in the same optima as the original objective. Specifically, we consider the *Conditional Flow Matching* (CFM) objective,

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,q(x_1),p_t(x|x_1)}\left\|v_t(x) - u_t(x|x_1)\right\|^2, \tag{4.5}$$

where $t \sim \mathcal{U}[0,1]$, $x_1 \sim q(x_1)$, and now $x \sim p_t(x|x_1)$. Unlike the FM objective, the CFM objective allows us to easily sample unbiased estimates as long as we can efficiently sample from $p_t(x|x_1)$ and compute $u_t(x|x_1)$, both of which can be easily done as they are defined on a per-sample basis.

Our second key observation is therefore:

The FM (equation 4.1) and CFM (equation 4.5) objectives have identical gradients w.r.t. θ.

That is, optimizing the CFM objective is equivalent (in expectation) to optimizing the FM objective. Consequently, this allows us to train a CNF to generate the marginal probability path p_t—which in particular, approximates the unknown data distribution q at t=1— without ever needing access to either the marginal probability path or the marginal velocity field. We simply need to design suitable *conditional* probability paths and velocity fields. We formalize this property in the following theorem.

Theorem 4. *Assuming that $p_t(x) > 0$ for all $x \in \mathbb{R}^d$ and $t \in [0,1]$, then, up to a constant independent of θ, $\mathcal{L}_{CFM}$ and $\mathcal{L}_{FM}$ are equal. Hence, $\nabla_\theta \mathcal{L}_{FM}(\theta) = \nabla_\theta \mathcal{L}_{CFM}(\theta)$.*

4.4 Conditional Probability Paths and Velocity Fields

The Conditional Flow Matching objective works with any choice of conditional probability path and conditional velocity fields. In this section, we discuss the construction of $p_t(x|x_1)$ and $u_t(x|x_1)$ for a general family of Gaussian conditional probability paths:

$$p_t(x|x_1) = \mathcal{N}(x \,|\, \mu_t(x_1), \sigma_t(x_1)^2 I), \tag{4.6}$$

where $\mu : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ is the time-dependent mean of the Gaussian distribution, while $\sigma : [0,1] \times \mathbb{R} \to \mathbb{R}_{>0}$ describes a time-dependent scalar standard deviation (std). We set $\mu_0(x_1) = 0$ and $\sigma_0(x_1) = 1$, so that all conditional probability paths converge to the same standard Gaussian noise distribution at $t = 0$, $p(x) = \mathcal{N}(x|0, I)$. We then set $\mu_1(x_1) = x_1$ and $\sigma_1(x_1) = \sigma_{\min}$, which is set sufficiently small so that $p_1(x|x_1)$ is a concentrated Gaussian distribution centered at x_1.

There is an infinite number of velocity fields that generate any particular probability path (e.g., by adding a divergence free component to the continuity equation, see equation 2.28). We decide to use the simplest velocity field corresponding to a canonical transformation for Gaussian distributions. Specifically, consider the flow (conditioned on x_1)

$$\psi_t(x) = \sigma_t(x_1)x + \mu_t(x_1). \tag{4.7}$$

When x is distributed as a standard Gaussian, $\psi_t(x)$ is the affine transformation that maps to a normally-distributed random variable with mean $\mu_t(x_1)$ and std $\sigma_t(x_1)$. That is to say, according to equation 2.27, ψ_t pushes the noise distribution $p_0(x|x_1) = p(x)$ to $p_t(x|x_1)$, i.e.,

$$[\psi_t]_* \, p(x) = p_t(x|x_1). \tag{4.8}$$

This flow then provides a velocity field that generates the conditional probability path:

$$\frac{d}{dt}\psi_t(x) = u_t(\psi_t(x)|x_1). \tag{4.9}$$

Reparameterizing $p_t(x|x_1)$ in terms of just x_0 and plugging equation 4.9 in the CFM loss:

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,q(x_1),p(x_0)} \left\| v_t(\psi_t(x_0)) - \frac{d}{dt}\psi_t(x_0) \right\|^2. \tag{4.10}$$

Since ψ_t is a simple (invertible) affine map we can use equation 4.9 to solve for u_t in a closed form. Let f' denote the derivative with respect to time, i.e., $f' = \frac{d}{dt}f$, for a time-dependent function f.

Theorem 5. *Let $p_t(x|x_1)$ be a Gaussian probability path as in equation 4.6, and ψ_t its corresponding flow map as in equation 4.7. Then, the unique velocity field that defines ψ_t has the form:*

$$u_t(x|x_1) = \frac{\sigma_t'(x_1)}{\sigma_t(x_1)} \left(x - \mu_t(x_1)\right) + \mu_t'(x_1). \tag{4.11}$$

Consequently, $u_t(x|x_1)$ generates the Gaussian path $p_t(x|x_1)$.

4.4.1 Special Instances of Gaussian Conditional Probability Paths

Our formulation is general for arbitrary functions $\mu_t(x_1)$ and $\sigma_t(x_1)$, and we can set them to any differentiable functions satisfying the desired boundary conditions. We first discuss the special cases that recover probability paths corresponding to previously-used diffusion processes. Since we directly work with probability paths, we can simply depart from reasoning about diffusion processes altogether. Therefore, in the second example, we directly formulate a probability path based on the Wasserstein-2 optimal transport solution.

Example I: Diffusion conditional VFs. Diffusion models start with data points and gradually add noise until it approximates pure noise. These can be formulated as stochastic processes, which have strict requirements in order to obtain closed form representation at arbitrary times t, resulting in Gaussian conditional probability paths $p_t(x|x_1)$ with specific choices of mean $\mu_t(x_1)$ and std $\sigma_t(x_1)$ [Soh+15; HJA20; Son+21b]. For example, the reversed (noise→data) Variance Exploding (VE) path has the form

$$p_t(x|x_1) = \mathcal{N}(x|x_1, \sigma_{1-t}^2 I), \tag{4.12}$$

where σ_t is an increasing function, $\sigma_0 = 0$, and $\sigma_1 \gg 1$. Next, equation 4.12 provides the choices of $\mu_t(x_1) = x_1$ and $\sigma_t(x_1) = \sigma_{1-t}$. Plugging these into equation 4.11 of Theorem 5

$$u_t(x|x_1) = -\frac{\sigma'_{1-t}}{\sigma_{1-t}}(x - x_1). \tag{4.13}$$

The reversed (noise→data) Variance Preserving (VP) diffusion path has the form

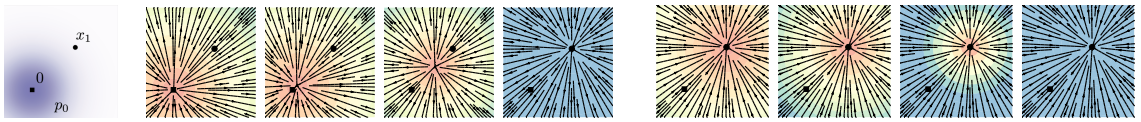
$$p_t(x|x_1) = \mathcal{N}(x \,|\, \alpha_{1-t}x_1, \left(1 - \alpha_{1-t}^2\right) I), \text{where } \alpha_t = e^{-\frac{1}{2}T(t)}, T(t) = \int_0^t \beta(s)ds, \tag{4.14}$$

and β is the noise scale function. Equation 4.14 provides the choices of $\mu_t(x_1) = \alpha_{1-t}x_1$ and $\sigma_t(x_1) = \sqrt{1 - \alpha_{1-t}^2}$. Plugging these into equation 4.11 of Theorem 5 we get

$$u_t(x|x_1) = \frac{\alpha'_{1-t}}{1 - \alpha_{1-t}^2}\left(\alpha_{1-t}x - x_1\right) = -\frac{T'(1-t)}{2}\left[\frac{e^{-T(1-t)}x - e^{-\frac{1}{2}T(1-t)}x_1}{1 - e^{-T(1-t)}}\right]. \tag{4.15}$$

Our construction of the conditional VF $u_t(x|x_1)$ does in fact coincide with the velocity field previously used in the deterministic probability flow ([Son+21b], equation 13) when restricted to these conditional diffusion processes; see details in Appendix B.2.1. Nevertheless, combining the diffusion conditional VF with the Flow Matching objective offers an attractive training alternative—which we find to be more stable and robust in our experiments—to existing score matching approaches.

Another important observation is that, as these probability paths were previously derived as solutions of diffusion processes, they do not actually reach a true noise distribution in finite time. In practice, $p_0(x)$ is simply approximated by a suitable Gaussian distribution for sampling and likelihood evaluation. Instead, our construction provides full control over the probability path, and we can just directly set μ_t and σ_t, as we will do next.



Diffusion path – conditional score function OT path – conditional velocity field

Figure 4.2: Compared to the diffusion path's conditional score function, the OT path's conditional velocity field has constant direction in time and is arguably simpler to fit with a parametric model. Note the blue color denotes larger magnitude while red color denotes smaller magnitude.

Example II: Optimal Transport conditional VFs. An arguably more natural choice for conditional probability paths is to define the mean and the std to simply change linearly in time, i.e.,

$$\mu_t(x_1) = tx_1, \text{ and } \sigma_t(x) = 1 - (1 - \sigma_{\min})t. \tag{4.16}$$

According to Theorem 5 this path is generated by the VF

$$u_t(x|x_1) = \frac{x_1 - (1 - \sigma_{\min})x}{1 - (1 - \sigma_{\min})t}, \tag{4.17}$$

which, in contrast to the diffusion conditional VF (equation 4.15), is defined for all $t \in [0, 1]$. The conditional flow that corresponds to $u_t(x|x_1)$ is

$$\psi_t(x) = (1 - (1 - \sigma_{\min})t)x + tx_1, \tag{4.18}$$

and in this case, the CFM loss (see equations 4.5, 4.10) takes the form:

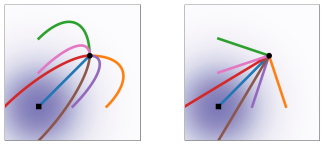
$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,q(x_1),p(x_0)} \left\| v_t(\psi_t(x_0)) - \Big(x_1 - (1 - \sigma_{\min})x_0\Big) \right\|^2. \tag{4.19}$$

Allowing the mean and std to change linearly not only leads to simple and intuitive paths, but it is actually also optimal in the following sense. The conditional flow $\psi_t(x)$ is in fact the Optimal Transport (OT) *displacement map* between the two Gaussians $p_0(x|x_1)$ and $p_1(x|x_1)$. The OT *interpolant*, which is a probability path, is defined to be (see Definition 1.1 in [McC97]):

$$p_t = [(1 - t)\text{id} + t\psi]_\star p_0 \tag{4.20}$$

where $\psi : \mathbb{R}^d \to \mathbb{R}^d$ is the OT map pushing p_0 to p_1, id denotes the identity map, i.e., $\text{id}(x) = x$, and $(1 - t)\text{id} + t\psi$ is called the OT displacement map. Example 1.7 in [McC97] shows, that in our case of two Gaussians where the first is a standard one, the OT displacement map takes the form of equation 4.18.

Intuitively, particles under the OT displacement map always move in straight line trajectories and with constant speed. Figure 4.3 depicts sampling paths for the diffusion and OT conditional VFs. Interestingly, we find that sampling trajectory from diffusion paths can "overshoot" the final sample, resulting in unnecessary backtracking, whilst the OT paths are guaranteed to stay straight.



Diffusion OT

Figure 4.3: Diffusion and OT conditional trajectories.

Figure 4.2 compares the diffusion conditional score function (the regression target in a typical diffusion methods), i.e., $\nabla \log p_t(x|x_1)$ with p_t defined as in equation 4.14, with the OT conditional VF (equation 4.17). The start (p_0) and end (p_1) Gaussians are identical in both examples. An interesting observation is that the OT VF has a constant direction in time, which arguably leads to a simpler regression task. This property can also be verified directly from equation 4.17 as the VF can be written in the form $u_t(x|x_1) = g(t)h(x|x_1)$. Figure B.1 in the Appendix shows a visualization of the Diffusion VF. Lastly, we note that although the conditional flow is optimal, this by no means imply that the marginal VF is an optimal transport solution. Nevertheless, we expect the marginal velocity field to remain relatively simple.

4.5 Related Work

Continuous Normalizing Flows were introduced in [Che+18] as a continuous-time version of Normalizing Flows (see e.g., [KPB20; Pap+21] for an overview). Originally, CNFs are trained with the maximum likelihood objective, but this involves expensive ODE simulations for the forward and backward propagation, resulting in high time complexity due to the sequential nature of ODE simulations. Although some works demonstrated the capability of CNF generative models for image synthesis [Gra+19], scaling up to very high dimensional images is inherently difficult. A number of works attempted to regularize the ODE to be easier to solve, e.g., using augmentation [DDT19], adding regularization terms [YK19; Fin+20b; Onk+21a; Ton+20; Kel+20], or stochastically sampling the integration interval [Du+22]. These works merely aim to regularize the ODE but do not change the fundamental training algorithm.

In order to speed up CNF training, some works have developed simulation-free CNF training frameworks by explicitly designing the target probability path and the dynamics. For instance, [Roz+21] consider a linear interpolation between the prior and the target density but involves integrals that were difficult to estimate in high dimensions, while [Ben+22] consider general probability paths similar to this work but suffers from biased gradients in the stochastic minibatch regime. In contrast, the Flow Matching framework allows simulation-free training with unbiased gradients and readily scales to very high dimensions.

Another approach to simulation-free training relies on the construction of a diffusion process to indirectly define the target probability path [Soh+15; HJA20; SE19]. [Son+21b] shows that diffusion models are trained using denoising score matching [Vin11], a conditional objective that provides unbiased gradients with respect to the score matching objective. Conditional Flow Matching draws inspiration from this result, but generalizes to matching vector fields directly. Due to the ease of scalability, diffusion models have received increased attention, producing a variety of improvements such as loss-rescaling [Son+21a], adding classifier guidance along with architectural improvements [DN21], and learning the noise schedule [ND21; Kin+21]. However, [ND21] and [Kin+21] only consider a restricted setting of Gaussian conditional paths defined by simple diffusion processes with a single

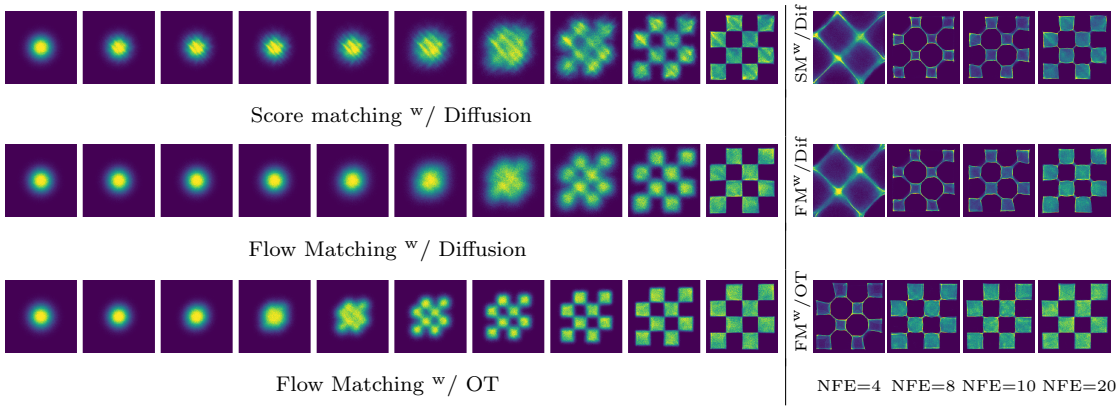


Figure 4.4: (*left*) Trajectories of CNFs trained with different objectives on 2D checkerboard data. The OT path introduces the checkerboard pattern much earlier, while FM results in more stable training. (*right*) FM with OT results in more efficient sampling, solved using the midpoint scheme.

parameter—in particular, it does not include our conditional OT path.

In another line of works, [De +21; Wan+21; Pel21] proposed finite time diffusion constructions via diffusion bridges theory resolving the approximation error incurred by infinite time denoising constructions.

While existing works make use of a connection between diffusion processes and continuous normalizing flows with the same probability path [MRO20a; Son+21b; Son+21a], our work allows us to generalize beyond the class of probability paths modeled by simple diffusion. With our work, it is possible to completely sidestep the diffusion process construction and reason directly with probability paths, while still retaining efficient training and log-likelihood evaluations. Lastly, concurrently to our work [LGL23; AV23] arrived at similar conditional objectives for simulation-free training of CNFs, while [NSM23] derived an implicit objective when u_t is assumed to be a gradient field.

4.6 Experiments

We explore the empirical benefits of using Flow Matching on the image datasets of CIFAR-10 [KH+09] and ImageNet at resolutions 32, 64, and 128 [CLH17; Den+09a]. We also ablate the choice of diffusion path in Flow Matching, particularly between the standard variance preserving diffusion path and the optimal transport path. We discuss how sample generation is improved by directly parameterizing the generating vector field and using the Flow Matching objective. Lastly we show Flow Matching can also be used in the conditional generation setting. Unless otherwise specified, we evaluate likelihood and samples from the model using `dopri5` [DP80] at absolute and relative tolerances of 1e-5. Generated samples can be found in the Appendix, and all implementation details are in Appendix B.2.2.

4.6.1 Density Modeling and Sample Quality on ImageNet

We start by comparing the same model architecture, i.e., the U-Net architecture from [DN21] with minimal changes, trained on CIFAR-10, and ImageNet 32/64 with different

Table 4.1: Likelihood (BPD), quality of generated samples (FID), and evaluation time (NFE) for the same model trained with different methods.

	CIFAR-10			ImageNet 32×32			ImageNet 64×64		
Model	NLL↓	FID↓	NFE↓	NLL↓	FID↓	NFE↓	NLL↓	FID↓	NFE↓
Ablations									
DDPM	3.12	7.48	274	3.54	6.99	262	3.32	17.36	264
Score Matching	3.16	19.94	242	3.56	5.68	178	3.40	19.74	441
ScoreFlow	3.09	20.78	428	3.55	14.14	195	3.36	24.95	601
Ours									
FM ^{w}/ Diffusion	3.10	8.06	183	3.54	6.37	193	3.33	16.88	187
FM ^{w}/ OT	2.99	6.35	142	3.53	5.02	122	3.31	14.45	138

	ImageNet 128×128	
Model	NLL↓	FID↓
MGAN [Hoa+18]	–	58.9
PacGAN2 [Lin+18]	–	57.5
Logo-GAN-AE [Sag+18]	–	50.9
Self-cond. GAN [Luč+19]	–	41.7
Uncond. BigGAN [Luč+19]	–	25.3
PGMGAN [Arm+21]	–	21.7
FM ^{w}/ OT	2.90	20.9

popular diffusion-based losses: DDPM from [HJA20], Score Matching (SM) [Son+21b], and Score Flow (SF) [Son+21a]; see Appendix B.2.2 for exact details. Table 4.1 (left) summarizes our results alongside these baselines reporting negative log-likelihood (NLL) in units of bits per dimension (BPD), sample quality as measured by the Frechet Inception Distance (FID; [Heu+17]), and averaged number of function evaluations (NFE) required for the adaptive solver to reach its a prespecified numerical tolerance, averaged over 50k samples. All models are trained using the same architecture, hyperparameter values and number of training iterations, where baselines are allowed more iterations for better convergence. Note that these are *unconditional* models. On both CIFAR-10 and ImageNet, FM-OT consistently obtains best results across all our quantitative measures compared to competing methods. We are noticing a higher that usual FID performance in CIFAR-10 compared to previous works [HJA20; Son+21b; Son+21a] that can possibly be explained by the fact that our used architecture was not optimized for CIFAR-10.

Secondly, Table 4.1 (right) compares a model trained using Flow Matching with the OT path on ImageNet at resolution 128×128. Our FID is state-of-the-art with the exception of IC-GAN [Cas+21] which uses conditioning with a self-supervised ResNet50 model, and therefore is left out of this table. Figures B.4, B.5, B.6 in the Appendix show non-curated samples from these models.

Faster training. While existing works train diffusion models with a very high number of iterations (e.g., 1.3m and 10m iterations are reported by Score Flow and VDM, respectively), we find that Flow Matching generally converges much faster. Figure 4.5 shows FID curves during training of Flow Matching and all baselines for ImageNet 64×64; FM-OT is able to lower the FID faster and to a greater extent than the alternatives. For ImageNet-128 [DN21] train for 4.36m iterations with batch size 256, while FM (with 25% larger model) used 500k iterations with batch size 1.5k, i.e., 33% less image throughput; see Table B.1 for exact details. Furthermore, the cost of sampling from a model can drastically change during training for score matching, whereas the sampling cost stays constant when training with Flow Matching (Figure B.3 in Appendix).

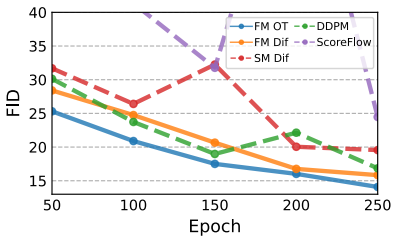
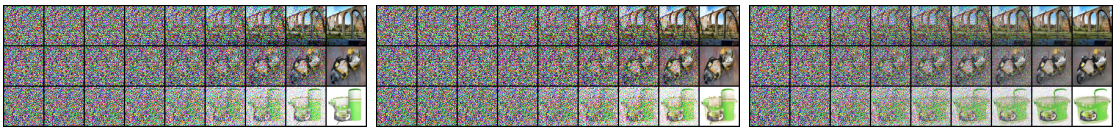
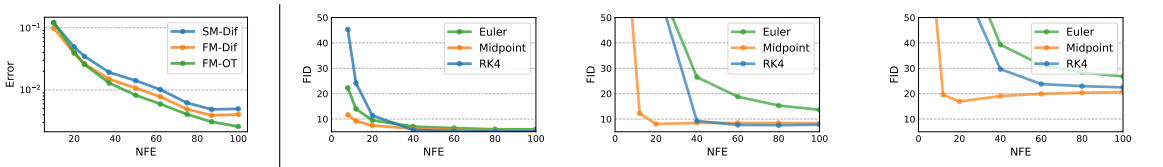


Figure 4.5: Image quality during training, ImageNet 64×64.



Score Matching w/ Diffusion | Flow Matching w/ Diffusion | Flow Matching w/ OT

Figure 4.6: Sample paths from the same initial noise with models trained on ImageNet 64×64. The OT path reduces noise roughly linearly, while diffusion paths visibly remove noise only towards the end of the path. Note also the differences between the generated images.



Error of ODE solution | Flow matching w/ OT | Flow matching w/ Diffusion | Score matching w/ Diffusion

Figure 4.7: Flow Matching, especially when using OT paths, allows us to use fewer evaluations for sampling while retaining similar numerical error (left) and sample quality (right). Results are shown for models trained on ImageNet 32×32, and numerical errors are for the midpoint scheme.

4.6.2 Sampling Efficiency

For sampling, we first draw a random noise sample $x_0 \sim \mathcal{N}(0, I)$ then compute $\psi_1(x_0)$ by solving equation 2.24 with the trained VF, v_t, on the interval $t \in [0, 1]$ using an ODE solver. While diffusion models can also be sampled through an SDE formulation, this can be highly inefficient and many methods that propose fast samplers (e.g., [SME21; ZC22]) directly make use of the ODE perspective (see Appendix B.2.1). In part, this is due to ODE solvers being much more efficient—yielding lower error at similar computational costs [KPS12]—and the multitude of available ODE solver schemes. When compared to our ablation models, we find that models trained using Flow Matching with the OT path always result in the most efficient sampler, regardless of ODE solver, as demonstrated next.

Sample paths. We first qualitatively visualize the difference in sampling paths between diffusion and OT. Figure 4.6 shows samples from ImageNet-64 models using identical random seeds, where we find that the OT path model starts generating images sooner than the diffusion path models, where noise dominates the image until the very last time point. We additionally depict the probability density paths in 2D generation of a checkerboard pattern, Figure 4.4 (left), noticing a similar trend.

Low-cost samples. We next switch to fixed-step solvers and compare low ($\leq$100) NFE samples computed with the ImageNet-32 models from Table 4.1. In Figure 4.7 (left), we compare the per-pixel MSE of low NFE solutions compared with 1000 NFE solutions (we use 256 random noise seeds), and notice that the FM with OT model produces the best numerical error, in terms of computational cost, requiring roughly only 60% of the NFEs to reach the same error threshold as diffusion models. Secondly, Figure 4.7 (right) shows how FID changes as a result of the computational cost, where we find FM with OT is able to achieve decent FID even at very low NFE values, producing better trade-off between sample quality and cost compared to ablated models. Figure 4.4 (right) shows low-cost sampling effects for the 2D checkerboard experiment.

4.6.3 Conditional Sampling from Low-Resolution Images

Lastly, we experimented with Flow Matching for conditional image generation. In particular, upsampling images from 64×64 to 256×256. We follow the evaluation procedure in [Sah+22b] and compute the FID of the upsampled validation images; baselines include reference (FID of original validation set), and regression. Results are in Table 4.2. Upsampled image samples are shown in Figures B.7, B.8 in the Appendix. FM-OT achieves similar PSNR and SSIM values to [Sah+22b] while considerably improving on FID and IS, which as argued by [Sah+22b] is a better indication of generation quality.

Model	FID↓	IS↑	PSNR↑	SSIM↑
Reference	1.9	240.8	–	–
Regression	15.2	121.1	27.9	0.801
SR3 [Sah+22b]	5.2	180.1	26.4	0.762
FM ^{w}/ OT	3.4	200.8	24.7	0.747

Table 4.2: Image super-resolution on the ImageNet validation set.

Part II

Extensions and Applications

Chapter 5

Multisample Flow Matching

Simulation-free methods for training continuous-time generative models construct probability paths that go between noise distributions and individual data samples. Recent works, such as Flow Matching, derived paths that are optimal for each data sample. However, these algorithms rely on independent data and noise samples, and do not exploit underlying structure in the data distribution for constructing probability paths. We propose Multisample Flow Matching, a more general framework that uses non-trivial couplings between data and noise samples while satisfying the correct marginal constraints. At very small overhead costs, this generalization allows us to (i) reduce gradient variance during training, (ii) obtain straighter flows for the learned vector field, which allows us to generate high-quality samples using fewer function evaluations, and (iii) obtain transport maps with lower cost in high dimensions, which has applications beyond generative modeling. Importantly, we do so in a completely simulation-free manner with a simple minimization objective. We show that our proposed methods improve sample consistency on downsampled ImageNet data sets, and lead to better low-cost sample generation. This chapter was publishhed as [Poo+23].

5.1 Motivation and Contributions

Deep generative models offer an attractive family of paradigms that can approximate a data distribution and produce high quality samples, with impressive results in recent years [Ram+22; Sah+22a; Gaf+22]. In particular, these works have made use of simulation-free training methods for diffusion models [HJA20; Son+21b]. A number of works have also adopted and generalized these simulation-free methods [Lip+23; AV23; LGL23; NSM22] for continuous normalizing flows (CNF; [Che+18]).

In Chapter §4, we proposed *Flow Matching* (FM), a method to train CNFs based on constructing explicit *conditional probability paths* between the noise distribution (at time $t = 0$) and each data sample (at time $t = 1$). Furthermore, we showed that these conditional probability paths can be taken to be the optimal transport path when the noise distribution is a standard Gaussian, a typical assumption in generative modeling. However, this does not imply that the *marginal probability path* (marginalized over the

data distribution) is anywhere close to the optimal transport path between the noise and data distributions.

Most existing works, including diffusion models and Flow Matching, have only considered conditional sample paths where the endpoints (a noise sample and a data sample) are sampled independently. However, this results in non-zero gradient variances even at convergence, slow training times, and in particular limits the design of probability paths. In turn, it becomes difficult to create paths that are fast to simulate, a desirable property for both likelihood evaluation and sampling.

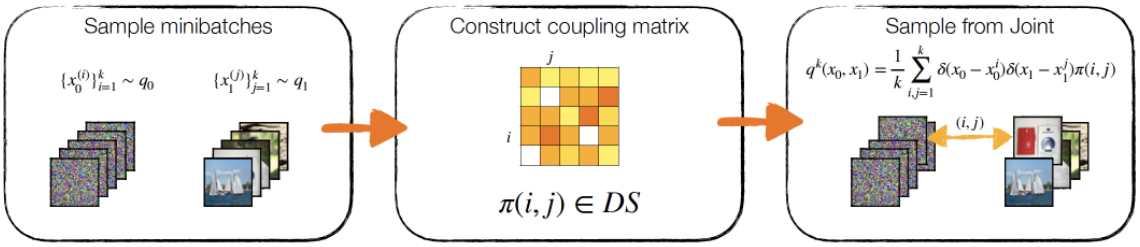


Figure 5.1: *The Multisample Flow Matching Algorithm.* We randomly sample noise and data samples, then re-arrange the pairing to be either optimal, or stable, within the current minibatch.

Contributions: We present a tractable instance of Flow Matching with joint distributions, which we call *Multisample Flow Matching*. Our proposed method generalizes the construction of probability paths by considering non-independent couplings of k-sample empirical distributions.

Among other theoretical results, we show that if an appropriate optimal transport (OT) inspired coupling is chosen, then sample paths become straight as the batch size $k \to \infty$, leading to more efficient simulation. In practice, we observe both improved sample quality on ImageNet using adaptive ODE solvers and using simple Euler discretizations with a low budget number of function evaluations. Empirically, we find that on ImageNet, we can *reduce the required sampling cost by 30% to 60%* for achieving a low Fréchet Inception Distance (FID) compared to a baseline Flow Matching model, while introducing only 4% more training time. This improvement in sample efficiency comes at no degradation in performance, e.g.log-likelihood and sample quality.

Within the deep generative modeling paradigm, this allows us to regularize towards the optimal velocity field in a *completely simulation-free manner* (unlike e.g.[Fin+20b; LGL23]), and avoids adversarial formulations (unlike e.g.[Mak+20; AV23]). In particular, we are the first work to be able to make use of solutions from optimal solutions on minibatches while preserving the correct marginal distributions, whereas prior works would only fit to the barycentric average (see detailed discussion in §5.5.1). Beyond generative modeling, we also show how our method can be seen as a new way to compute approximately optimal transport maps between arbitrary distributions in settings where the cost function is completely unknown and only minibatch optimal transport solutions are provided.

5.2 Preliminaries

5.2.1 Flow Matching

In this chapter, we extend flow matching (see §4) to joint distributions, considering non-trivial coupling between the source and target distributions. We first slightly change some notations to make the treatment of the source and target distributions more symmetric as well as simplify a little the notations of equation 4.9 to make the connection between the conditional flow and conditional velocity field clearer.

Given two marginal distributions $q_0(x_0)$ (source) and $q_1(x_1)$ (target) for which we would like to learn a CNF to transport between, Flow Matching proposes the *Conditional Flow Matching* (CFM) objective:

$$\mathbb{E}_{t,q_1(x_1),p_t(x|x_1)} \left\| v_t(x;\theta) - u_t(x_t|x_1) \right\|^2, \tag{5.1}$$

with $x_t \coloneqq \psi_t(x_0|x_1)$ and $\psi_t(x_0|x_1)$ is the conditional flow defined by the conditional velocity field $u_t(x|x_1)$ and equation 2.24, generating the conditional probability path satisfying:

$$p_{t=0}(x|x_1) = q_0(x) \quad \text{and} \quad p_{t=1}(x|x_1) = \delta(x - x_1), \tag{5.2}$$

where $\delta(x-a)$ is a Dirac mass centered at $a \in \mathbb{R}^d$. By construction, $p_t(x|x_1)$ now satisfies both marginal constraints.

Conditional OT (CondOT) path

One particular choice of conditional path $p_t(x|x_1)$ is to use the flow that corresponds to the optimal transport displacement interpolant [McC97] when $q_0(x_0)$ is the standard Gaussian. The corresponding velocity field is

$$u_t(x_t|x_1) = \frac{x_1 - x}{1 - t}. \tag{5.3}$$

Using this conditional velocity field in equation 2.24, this gives the conditional flow

$$x_t = \psi_t(x_0|x_1) = (1-t)x_0 + tx_1\,. \tag{5.4}$$

Substituting equation 5.4 into equation 5.3, one can also express the value of this velocity field using a simpler expression,

$$u_t(x_t|x_1) = x_1 - x_0\,. \tag{5.5}$$

It is evident that this results in conditional flows that *(i)* tranports all points x_0 from $t = 0$ to x_1 at exactly $t = 1$ and *(ii)* are straight paths between the samples x_0 and x_1. This particular case of straight paths was also studied by [LGL23] and [AV23], where the conditional flow equation 5.4 is referred to as a stochastic interpolant. In §4 we showed that the conditional construction can be applied to a large class of Gaussian

conditional probability paths, namely when $p_t(x|x_1) = \mathcal{N}(x|\mu_t(x_1), \sigma_t(x_1)^2 I)$. This family of probability paths encompasses most prior diffusion models where probability paths are induced by simple diffusion processes with linear drift and constant diffusion (e.g.[HJA20; Son+21b]). However, existing works mostly consider settings where $q_0(x_0)$ and $q_1(x_1)$ are sampled independently when computing training objectives such as equation 5.1.

5.2.2 Optimal Transport: Static & Dynamic

Optimal transport generally considers methodologies that define some notion of distance on the space of probability measures [Vil08; Vil03; San15]. Letting $\mathcal{P}(\mathbb{R}^d)$ be the space of probability measures over $\mathbb{R}^d$, we define the Wasserstein distance with respect to a cost function $c : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$ between two measures $q_0, q_1 \in \mathcal{P}(\mathbb{R}^d)$ as [Kan42]

$$W_c(q_0, q_1) \coloneqq \min_{q \in \Gamma(q_0, q_1)} \mathbb{E}_{q(x_0, x_1)}[c(x_0, x_1)]\,, \tag{5.6}$$

where $\Gamma(q_0, q_1)$ is the set of joint measures with left marginal equal to q_0 and right marginal equal to q_1, called the set of *couplings*. The minimizer to equation 5.6 is called the optimal coupling, which we denote by q_c^*. In the case where $c(x_0, x_1) \coloneqq \|x_0 - x_1\|^2$, the squared-Euclidean distance, equation 5.6 amounts to the (squared) 2-Wasserstein distance $W_2^2(q_0, q_1)$, and we simply write the optimal transport plan as q^*.

Considering again the squared-Euclidean cost, in the case where q_0 exhibits a density over $\mathbb{R}^d$ (e.g. if q_0 is the standard normal distribution), [BB00] states that $W_2^2(q_0, q_1)$ can be equivalently expressed as a *dynamic* formulation,

$$W_2^2(q_0, q_1) = \min_{p_t, u_t} \int_0^1 \int_{\mathbb{R}^d} \|u_t(x)\|^2 \, p_t(x) \mathrm{d}x_0 \mathrm{d}t. \tag{5.7}$$

where u_t generates p_t, and p_t satisfies boundary conditions $p_{t=0} = q_0$ and $p_{t=1} = q_1$. The optimality condition ensures that sample paths x_t are straight lines, i.e. minimize the length of the path, and leads to paths that are much easier to simulate. Some prior approaches have sought to regularize the model using this optimality objective (e.g.[Ton+20; Fin+20b]). In contrast, instead of directly minimizing equation 5.7, we will discuss an approach based on using solutions of the optimal coupling q^* on minibatch problems, while leaving the marginal constraints intact.

5.3 Flow Matching with Joint Distributions

While Conditional Flow Matching in equation 5.1 leads to an unbiased gradient estimator for the Flow Matching objective, it was designed with independently sampled x_0 and x_1 in mind. We generalize the framework from Subsection 5.2.1 to a construction that uses arbitrary joint distributions of $q(x_0, x_1)$ which satisfy the correct marginal constraints, i.e.

$$\int q(x_0, x_1) \mathrm{d}x_1 = q_0(x_0)\,, \quad \int q(x_0, x_1) \mathrm{d}x_0 = q_1(x_1). \tag{5.8}$$

We will show in §5.4 that this can potentially lead to lower gradient variance during training and allow us to design more optimal marginal vector fields $u_t(x)$ with desirable properties such as improved sample efficiency.

Building on top of Flow Matching, we propose modifying the conditional probability path construction equation 5.2 so that at $t = 0$, we define

$$p_{t=0}(x_0|x_1) = q(x_0|x_1). \tag{5.9}$$

where $q(x_0|x_1)$ is the conditional distribution $\frac{q(x_0,x_1)}{q_1(x_1)}$. Using this construction, we still satisfy the marginal constraint,

$$p_0(x) = \int p_0(x|x_1)q_1(x_1)dx_1 = \int q(x, x_1)dx_1 = q_0(x)$$

i.e., $p_{t=0}(x) = \int q(x, x_1)dx_1 = q_0(x)$ by the assumption made in equation 5.8. Then similar to [CL23], we note that the conditional probability path $p_t(x|x_1)$ *need not be explicitly formulated* for training, and that only an appropriate conditional velocity field $u_t(x|x_1)$ needs to be chosen such that all points arrive at x_1 at $t = 1$, which ensures $p_{t=1}(x|x_1) = \delta(x - x_1)$. As such, we can make use of the same conditional velocity field as prior works, e.g., the choice in equations 5.3 to 5.5.

We then propose the **Joint CFM** objective as

$$\mathcal{L}_{\text{JCFM}} = \mathbb{E}_{t,q(x_0,x_1)} \left\| v_t(x_t; \theta) - u_t(x_t|x_1) \right\|^2, \tag{5.10}$$

where $x_t = \psi_t(x_0|x_1)$ is the conditional flow. Training only involves sampling from $q(x_0, x_1)$ and does not require explicitly knowing the densities of $q(x_0, x_1)$ or $p_t(x|x_1)$. Note that equation 5.10 reduces to the original CFM objective equation 5.1 when $q(x_0, x_1) = q_0(x_0)q_1(x_1)$.

A quick sanity check shows that this objective can be used with any choice of joint distribution $q(x_0, x_1)$.

Lemma 1. *The optimal velocity field $v_t(\cdot; \theta)$ in equation 5.10, which is the marginal velocity field u_t, maps between the marginal distributions $q_0(x_0)$ and $q_1(x_1)$.*

In the remainder of the section, we highlight some motivations for using joint distributions $q(x_0, x_1)$ that are different from the independent distribution $q_0(x_0)q_1(x_1)$.

Variance reduction Choosing a good joint distribution can be seen as a way to reduce the variance of the gradient estimate, which improves and speeds up training. We develop the gradient covariance at a fixed x and t, and bound its total variance:

Lemma 2. *The total variance (i.e. the trace of the covariance) of the gradient at a fixed x and t is bounded as:*

$$\begin{aligned} \sigma^2_{t,x} &= \operatorname{Tr}\left[\operatorname{Cov}_{p_t(x_1|x)}\left(\nabla_\theta \left\| v_t(x;\theta) - u_t(x|x_1)\right\|^2\right)\right] \\ &\leq \left\|\nabla_\theta v_t(x;\theta)\right\|^2 \mathbb{E}_{p_t(x_1|x)} \left\| u_t(x) - u_t(x|x_1)\right\|^2 \end{aligned} \tag{5.11}$$

Then $\mathbb{E}_{t,p_t(x)}[\sigma^2_{t,x}]$ *is bounded above by:*

$$\max_{t,x} \|\nabla_\theta v_t(x;\theta)\|^2 \times \mathcal{L}_{\text{JCFM}} \tag{5.12}$$

This proves that $\mathbb{E}_{t,p_t(x)}[\sigma^2_{t,x}]$, which is the average gradient variance at fixed x and t, is upper bounded in terms of the Joint CFM objective. That means that minimizing the Joint CFM objective helps in decreasing $\mathbb{E}_{t,p_t(x)}[\sigma^2_{t,x}]$. Note also that $\mathbb{E}_{t,p_t(x)}[\sigma^2_{t,x}]$ is not the gradient variance and is always smaller, as it does not account for variability over x and t, but it is a good proxy for it. The proof is in App. A.3.2.

Sampling x_0 and x_1 independently generally cannot achieve value zero for $\mathbb{E}_{t,p_t(x)}[\sigma^2_{t,x}]$ *even at the optimum*, since there are an infinite number of pairs (x_0, x_1) whose conditional path crosses any particular x at a time t. As shown in equation 5.12, having a low optimal value for the Joint CFM objective is a good proxy for low gradient variance and hence a desirable property for choosing a joint distribution $q(x_0, x_1)$. In §5.4, we show that certain joint distributions have optimal Joint CFM values close to zero.

Straight flows Ideally, the flow ψ_t of the marginal vector field u_t (and of the learned v_θ by extension) should be close to a straight line. The reason is that ODEs with straight trajectories can be solved with high accuracy using fewer steps (i.e. function evaluations), which speeds up sample generation. The quantity

$$S = \mathbb{E}_{t,q_0(x_0)}\left[\|u_t(\psi_t(x_0))\|^2 - \|\psi_1(x_0) - x_0\|^2\right], \tag{5.13}$$

which we call the *straightness* of the flow and was also studied by [Liu22], measures how straight the trajectories are. Namely, we can rewrite it as

$$S = \mathbb{E}_{t,q_0(x_0)}\left[\|u_t(\psi_t(x_0)) - \mathbb{E}_{t'}\left[u_{t'}(\psi_{t'}(x_0))\right]\|^2\right], \tag{5.14}$$

which shows that $S \geq 0$ and only zero if $u_t(\psi_t(x_0))$ is constant along t, which is equivalent to $\psi_t(x_0)$ being a straight line.

When x_0 and x_1 are sampled independently, the straightness is in general far from zero. This can be seen in the CondOT plots in Figure 5.2 (right); if flows were close to straight lines, samples generated with one function evaluation (NFE=1) would be of high quality. In §5.4, we show that for certain joint distributions, the straightness of the flow is close to zero.

Near-optimal transport cost By Lemma 1, the flow ψ_t corresponding to the optimal u_t satisfies that $\psi_0(x_0) = x_0 \sim q_0$ and $\psi_1(x_0) \sim q_1$. Hence, $x_0 \mapsto \psi_1(x_0)$ is a transport map between q_0 and q_1 with an associated transport cost

$$\mathbb{E}_{q_0(x_0)}\|\psi_1(x_0) - x_0\|^2. \tag{5.15}$$

There is no reason to believe that when x_0 and x_1 are sampled independently, the transport cost $\mathbb{E}_{q_0(x_0)}\|\psi_1(x_0) - x_0\|^2$ will be anywhere near the optimal transport cost $W_2^2(p_0, p_1)$. Yet, in §5.4 we show that for well chosen q, the transport cost for ψ_1 does approach its optimal value. Computing optimal (or near-optimal) transport maps in high dimensions is a challenging task [Mak+20; Amo23] that extends beyond generative modeling and into the field of optimal transport, and it has applications in computer vision [Fey+17; Sol+15; Sol+16; Liu+23a] and computational biology [Lüb+22; Bun+21; BKc22; Sch+19], for instance. Hence, Joint CFM may also be viewed as a practical way to obtain approximately optimal transport maps in this context.

5.4 Multisample Flow Matching

Constructing a joint distribution satisfying the marginal constraints is difficult, especially since at least one of the marginal distributions is based on empirical data. We thus discuss a method to construct the joint distribution $q(x_0, x_1)$ implictly by designing a suitable sampling procedure that leaves the marginal distributions invariant. Note that training with equation 5.10 only requires sampling from $q(x_0, x_1)$.

We use a multisample construction for $q(x_0, x_1)$ in the following manner:

1. Sample $\{x_0^{(i)}\}_{i=1}^k \sim q_0(x_0)$ and $\{x_1^{(i)}\}_{i=1}^k \sim q_1(x_1)$.

2. Construct a doubly-stochastic matrix with probabilities $\pi(i, j)$ dependent on the samples $\{x_0^{(i)}\}_{i=1}^k$ and $\{x_1^{(i)}\}_{i=1}^k$.

3. Sample from the discrete distribution,
$q^k(x_0, x_1) = \frac{1}{k}\sum_{i,j=1}^k \delta(x_0 - x_0^i)\delta(x_1 - x_1^j)\pi(i, j)$.

Marginalizing $q^k(x_0, x_1)$ over samples from Step 1, we obtain the implicitly defined $q(x_0, x_1)$. By choosing different *couplings* $\pi(i, j)$, we induce different joint distributions. In this work, we focus on couplings that induce joint distributions which approximates, or at least partially satisfies, the optimal transport joint distribution. The following result, proven in App. A.3.3, guarantees that q has the right marginals.

Lemma 3. *The joint distribution $q(x_0, x_1)$ constructed in Steps [1-3] has marginals $q_0(x_0)$ and $q_1(x_1)$.*

That is, the marginal constraints equation 5.8 are satisfied and consequently we are allowed to use the framework of §5.3.

5.4.1 CondOT is Uniform Coupling

The aforementioned multisample construction subsumes the independent joint distribution used by prior works, when the joint coupling is taken to be uniformly distributed, i.e.$\pi(i, j) = \frac{1}{k}$. This is precisely the coupling we used in §4 under our introduced notion of Multisample Flow Matching, and acts as a natural reference point.

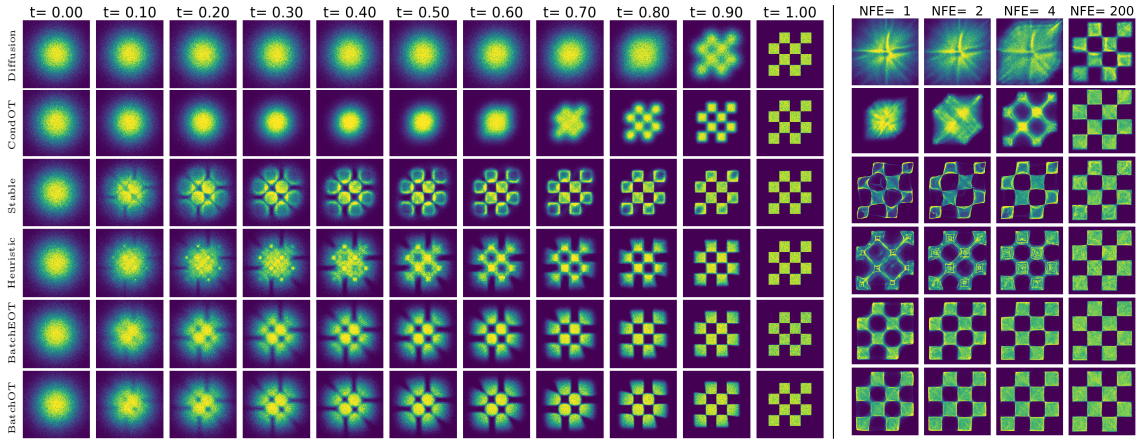


Figure 5.2: Multisample Flow Matching learn probability paths that are much closer to an optimal transport path than baselines such as Diffusion and CondOT paths. (*Left*) Exact marginal probability paths. (*Right*) Samples from trained models at $t = 1$ for different numbers of function evaluations (NFE), using Euler discretization. Furthermore, the final values of the Joint CFM objective equation 5.10—upper bounds on the variance of u_t at convergence—are: CondOT: 10.72; Stable: 1.60, Heuristic: 1.56; BatchEOT: 0.57, BatchOT: 0.24.

5.4.2 Batch Optimal Transport (BatchOT) Couplings

The natural connections between optimal transport theory and optimal sampling paths in terms of straight-line interpolations, lead us to the following pseudo-deterministic coupling, which we call Batch Optimal Transport (BatchOT). While it is difficult to solve equation 5.6 at the population level, it can efficiently solved on the level of samples. Let $\{x_0^{(i)}\}_{i=1}^k \sim q_0(x_0)$ and $\{x_1^{(i)}\}_{i=1}^k \sim q_1(x_1)$. When defined on batches of samples, the OT problem equation 5.6 can be solved exactly and efficiently using standard solvers, as in `POT` [Fla+21, Python Optimal Transport]. On a batch of k samples, the runtime complexity is well-understood via either the Hungarian algorithm or network simplex algorithm, with an overall complexity of $\mathcal{O}(k^3)$ [PC19, Chapter 3]. The resulting coupling $\pi^{k,*}$ from the algorithm is a *permutation matrix*, which is a type of doubly-stochastic matrix that we can incorporate into Step 3 of our procedure.

We consider the effect that the sample size k has on the marginal vector field $u_t(x)$. The following theorem shows that in the limit of $k \to \infty$, BatchOT satisfies the three criteria that motivate Joint CFM: variance reduction, straight flows, and near-optimal transport cost.

Theorem 6 (Informal)**.** *Suppose that Multisample Flow Matching is run with BatchOT. Then, as $k \to \infty$,*

(i) The value of the Joint CFM objective (equation 5.10) for the optimal u_t converges to 0.

(ii) The straightness S for the optimal marginal vector field u_t (equation 5.13) converges to zero.

(iii) The transport cost $\mathbb{E}_{q_0(x_0)}\|\psi_1(x_0) - x_0\|^2$ (equation 5.15) associated to u_t converges to the optimal transport cost $W_2^2(p_0, p_1)$.

Result *(i)* implies that the gradient variance both during training and at convergence

is reduced due to equation 5.12; result *(ii)* implies the optimal model will be easier to simulate between t=0 and t=1; result *(iii)* implies that Multisample Flow Matching can be used as a simulation-free algorithm for approximating optimal transport maps.

The full version of Theorem 6 can be found in Appendix A.3.4, and it makes use of standard, weak technical assumptions which are common in the optimal transport literature. While Theorem 6 only analyzes asymptotic properties, we provide theoretical evidence that the transport cost decreases with k, as summarized by a monotonicity result in Theorem 11.

5.4.3 Batch Entropic OT (BatchEOT) Couplings

For k sufficiently large, the cubic complexity of the BatchOT approach is not always desirable, and instead one may consider approximate methods that produce couplings sufficiently close to BatchOT at a lower computational cost. A popular surrogate, pioneered in [Cut13], is to incorporate an entropic penalty parameter on the doubly stochastic matrix $\pi(i, j)$, pulling it closer to the independent coupling:

$$\min_{q^k \in \Gamma(q_0, q_1)} \mathbb{E}_{q^k(x_0, x_1)} \|x_0 - x_1\|^2 + \epsilon H(q^k)\,,$$

where $H(q^k) = -\sum_{i,j} \pi_{i,j}(\log(\pi_{i,j}) - 1)$ is the entropy of the doubly stochastic matrix π, and $\epsilon > 0$ is some finite regularization parameter. The optimality conditions of this strictly convex program leads to Sinkhorn's algorithm, which has a runtime of $\tilde{\mathcal{O}}(k^2/\epsilon)$ [AWR17].

The output of performing Sinkhorn's algorithm is a doubly-stochastic matrix. The two limiting regimes of the regularization parameter are well understood (c.f. [PC19], Proposition 4.1, for instance): as $\epsilon \to 0$, BatchEOT recovers the BatchOT permutation matrix from §5.4.2; as $\epsilon \to \infty$, BatchEOT recovers the independent coupling on the indices from §5.4.1.

5.4.4 Stable and Heuristic Couplings

An alternative approach is to consider faster algorithms that satisfy at least some desirable properties of an optimal coupling. In particular, an optimal coupling is *stable*. A permutation coupling is stable if *no pair of* $x_0^{(i)}$ *and* $x_1^{(j)}$ *favor each other over their assigned pairs based on the coupling.* Such a problem can be solved using the Gale-Shapeley algorithm [GS62] which has a compute cost of $\mathcal{O}(k^2)$ given the cross set ranking of all samples. Starting from a random assignment, it is an iterative algorithm that reassigns pairs if they violate the stability property and can terminate very early in practice. Note that in a cost-based ranking, one has to sort the coupling costs of each sample with all samples in the opposing set, resulting in an overall $\mathcal{O}(k^2 \log(k))$ compute cost.

The Gale-Shapeley algorithm is agnostic to any particular costs, however, as stability is only defined in terms of relative rankings of individual samples. We design a modified version of this algorithm based on a heuristic for satisfying the cyclical monotonicity

property of optimal transport, namely that should pairs be reassigned, the reassignment should not increase the total cost of already matched pairs. We refer to the output of this modified algorithm as a *heuristic coupling* and discuss the details in Appendix B.3.1.

5.5 Related Work

Generative modeling and optimal transport are inherently intertwined topics, both often aiming to learn a transport between two distributions but with very different goals. Optimal transport is widely recognized as a powerful tool for large-scale generative modeling as it can be used to stabilize training [ACB17]. In the context of continuous-time generative modeling, optimal transport has been used to regularize continuous normalizing flows for easier simulation [Fin+20b; Onk+21a], and increase interpretability [Ton+20]. However, the existing methods for encouraging optimality in a generative model generally require either solving a potentially unstable min-max optimization problem (e.g.[ACB17; Mak+20; AV23]) or require simulation of the learned vector field as part of training (e.g.[Fin+20b; LGL23]). In contrast, the approach of using batch optimal couplings can be used to avoid the min-max optimization problem, but has not been successfully applied to generative modeling as they do not satisfy marginal constraints—we discuss this further in the following §5.5.1. On the other hand, neural optimal transport approaches are mainly centered around the quadratic cost [Mak+20; Amo23; Fin+20a] or rely heavily on knowing the exact cost function [Fan+22; Asa+24]. Being capable of using batch optimal couplings allows us to build generative models to approximate optimal maps under any cost function, and even when the cost function is unknown.

5.5.1 Minibatch Couplings for Generative Modeling

Among works that use optimal transport for training generative models are those that make use of batch optimal solutions and their gradients such as [Li+17; GPC18; Fat+20; LGS19]. However, *naïvely using solutions to batches only produces, at best, the barycentric map*, i.e.the map that fits to average of the batch couplings [Fer+14; Seg+17; PN21], and does not correctly match the true marginal distribution. This is a well-known problem and while multiple works (e.g.[Fat+21; Ngu+22]) have attempted to circumvent the issue through alternative formulations of optimality, the lack of marginal preservation has been a major downside of using batch couplings for generative modeling as they do not have the ability to match the target distribution for finite batch sizes. This is due to the use of building models within the *static* setting, where the map is parameterized directly with a neural network. In contrast, we have shown in §3 that in our *dynamic* setting, where we parameterize the map as the solution of a neural ODE, it is possible to preserve the marginal distribution exactly. Furthermore, we have shown in Proposition 3 (App. A.3.5) that our method produces a map that is no higher cost than the joint distribution induced from BatchOT couplings.

Concurrently, [Ton+24] motivates the use of BatchOT solutions within a similar frame-

work as our Joint CFM, but from the perspective of obtaining accurate solutions to dynamic optimal transport problems. Similarly, [LKY23] propose to explicitly learn a joint distribution, parameterized with a neural network, with the aim of minimizing trajectory curvature; this is done using through an auxiliary VAE-style objective function. In contrast, we propose a family of couplings that all satisfy the marginal constraints, all of which are easy to implement and have negligible cost during training. Our construction allow us to focus on (i) fixing consistency issues within simulation-free generative models, and (ii) using Joint CFM to obtain more optimal solutions than the original BatchOT solutions.

5.6 Experiments

We empirically investigate Multisample Flow Matching on a suite of experiments. First, we show how different couplings affect the model on a 2D distribution. We then turn to benchmark, high-dimensional datasets, namely ImageNet [Den+09b]. We use the official *face-blurred* ImageNet data and then downsample to 32×32 and 64×64 using the open source preprocessing scripts from [CLH17]. Finally, we explore the setting of unknown cost functions while only batch couplings are provided. Full details on the experimental setting can be found in Appendix B.3.2.

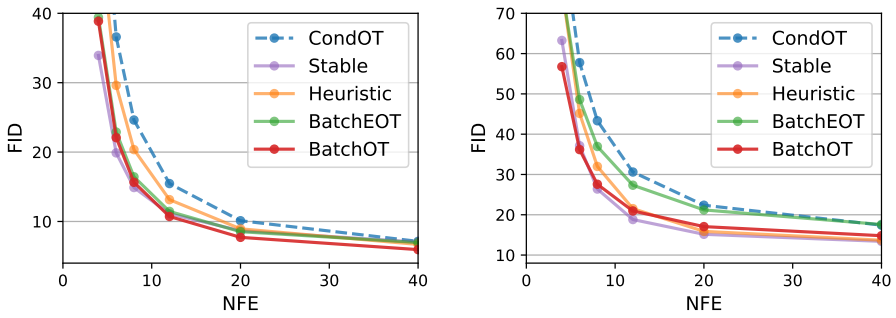
5.6.1 Insights from 2D experiments

Figure 5.2 shows the proposed Multisample Flow Matching algorithm on fitting to a checkboard pattern distribution in 2D. We show the marginal probability paths induced by different coupling algorithms, as well as low-NFE samples of trained models on these probability paths.

The diffusion and CondOT probability paths do not capture intricate details of the data distribution until it is almost at the end of the trajectory, whereas Multisample Flow Matching approaches provide a gradual transition to the target distribution along the flow. We also see that with a fixed step solver, the BatchOT method is able to produce an accurate target distribution in just one Euler step in this low-dimensional setting, while the other coupling approaches also get pretty close. Finally, it is interesting that both Stable and Heuristic exhibit very similar probability paths to optimal transport despite only satisfying weaker conditions.

5.6.2 Image Datasets

We find that Multisample Flow Matching retains the performance of Flow Matching while improving on sample quality, compute cost, and variance. In Table B.4, we report sample quality using the standard Fréchet Inception Distance (FID), negative log-likelihood values using bits per dimension (BPD), and compute cost using number of function evaluations (NFE); these are all standard metrics throughout the literature. Additionally, we report the variance of $u_t(x|x_0, x_1)$, estimated using the Joint CFM loss equation 5.10 which is



ImageNet 32×32 ImageNet 64×64

Figure 5.3: Sample quality (FID) vs compute cost (NFE) using Euler discretization. CondOT has significantly higher FID at lower NFE compared to proposed methods.

an upper bound on the variance. We do not observe any performance degradations while simulation efficiency improves significantly, even with small batch sizes.

Additionally, in Appendix B.3.3, we include runtime comparisons between Flow Matching and Multisample Flow Matching. On ImageNet32, we only observe a 0.8% relative increase in runtime compared to Flow Matching, and a 4% increase on ImageNet64.

Interestingly, we find that the Stable coupling actually performs on par, and some times better than the BatchOT coupling, despite having a smaller asymptotic compute cost and only satisfying a weaker condition within each batch.

As FID is computed over a full set of samples, it does not show how varying NFE affects individual sample paths. We discuss a notion of consistency next, where we analyze the similarity between low-NFE and high-NFE samples.

Higher sample quality on a compute budget We observe that with a fixed NFE, models trained using Multisample Flow Matching generally achieve better sample quality. For these experiments, we draw $x_0 \sim \mathcal{N}(0, I_d)$ and simulate $v_t(\cdot, \theta)$ up to time $t = 1$ using a fixed step solver with a fixed NFE. Figure 5.3 show that even on high dimensional data distributions, the sample quality of of multisample methods improves over the naïve CondOT approach as the number of function evaluations drops. We compare to the FID of diffusion baseline methods in Table 5.2, and provide additional results in Appendix B.3.3.

Table 5.1: Derived results shown in Figure 5.3, we can determine the approximate NFE required to achieve a certain FID across our proposed methods. The baseline diffusion-based methods (e.g. ScoreFlow and DDPM) require more than 40 NFE to achieve these FID values.

	ImageNet 32×32 NFE @ FID = 10	**ImageNet 64×64** NFE @ FID = 20
Diffusion	≥40	≥40
FM ^{w}/ CondOT	20	29
MultisampleFM ^{w}/ Heuristic	18	12
MultisampleFM ^{w}/ Stable	14	11
MultisampleFM ^{w}/ BatchOT	14	12

Table 5.2: FID (↓) of model samples on ImageNet 32×32 using varying number of function evaluations (NFE) using Euler discretization.

NFE	DDPM	ScoreSDE	BatchOT	Stable
Adaptive	5.72	6.84	4.68	5.79
40	19.56	16.96	5.94	7.02
20	63.08	58.02	7.71	8.66
8	232.97	218.66	15.64	14.89
6	275.28	266.76	22.08	19.88
4	362.37	340.17	38.86	33.92

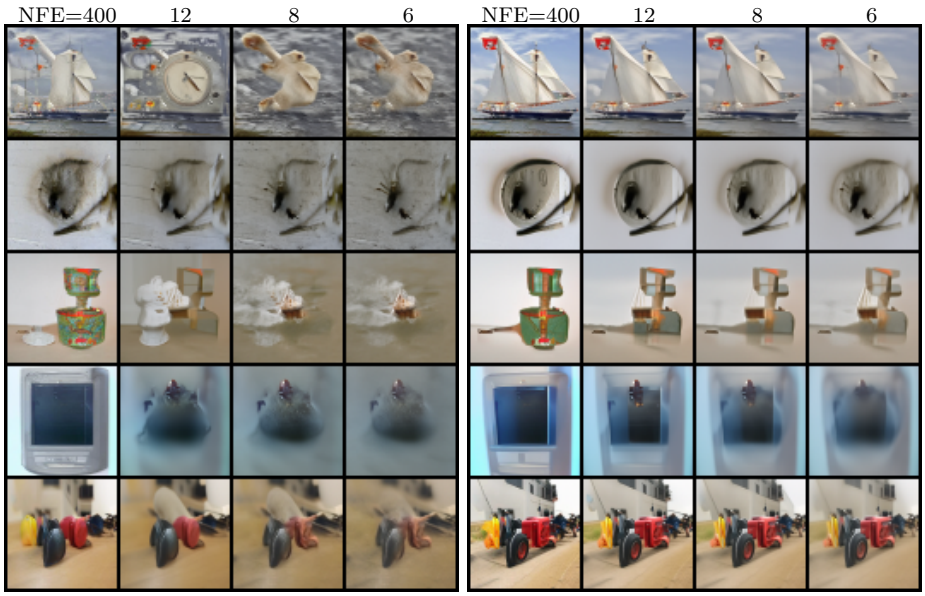


Figure 5.4: Multisample Flow Matching trained with batch optimal couplings produces more consistent samples across varying NFEs. Note that both flows on each row start from the same noise sample.

Consistency of individual samples In Figure 5.4 we show samples at different NFEs, where it can be qualitatively seen that BatchOT produces samples that are more consistent between high- and low-NFE solutions than CondOT, despite achieving similar FID values.

To evaluate this quantitatively, we define a metric for establishing the *consistency* of a model with respect to an integration scheme: let $x^{(m)}$ be the output of a numerical solver initialized at x using m function evalutions to reach $t = 1$, and let $x^{(*)}$ be a near-exact sample solved using a high-cost solver starting from x_0 as well. We define

$$\text{Consistency}(m) = \tfrac{1}{D}\mathbb{E}_{x\sim q_0}\|\mathcal{F}(x^{(m)}) - \mathcal{F}(x^{(*)})\|^2 \tag{5.16}$$

where $\mathcal{F}(\cdot)$ outputs the hidden units from a pretrained InceptionNet[1], and D is the number of hidden units. These kinds of perceptual losses have been used before to check the content alignment between two image samples (e.g.[GEB15; JAF16]). We find that Multisample Flow Matching has better consistency at all values of NFE, shown in Table 5.3.

[1]We take the same layer as used in standard FID computation.

Table 5.3: BatchOT produces samples with more similar content to its true samples at low NFEs (using midpoint discretization). Visual examples of this consistency are shown in Figure 5.4.

	ImageNet 32×32		**ImageNet 64×64**	
	CondOT	BatchOT	CondOT	BatchOT
Consistency(m=4)	0.141	0.101	0.174	0.157
Consistency(m=6)	0.105	0.071	0.151	0.134
Consistency(m=8)	0.079	0.052	0.132	0.115
Consistency(m=12)	0.046	0.030	0.106	0.085

Training efficiency Figure 5.5 shows the convergence of Multisample Flow Matching with BatchOT coupling compared to Flow Matching with CondOT and diffusion-based methods. We see that by choosing better joint distributions, we obtain faster training. This is in line with our variance estimates reported in Table B.6 and supports our hypothesis that gradient variance is reduced by using non-trivial joint distributions.

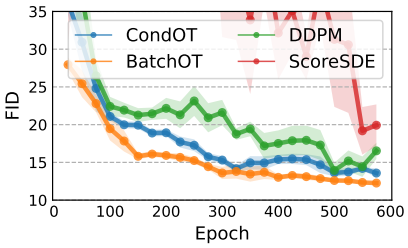


Figure 5.5: FID vs. epochs on ImageNet64.

Table 5.4: Matching couplings from an oracle BatchOT solver with unknown costs. Multisample Flow Matching is able to match the marginal distribution correctly while being at least a optimal as the oracle, but static maps fail to preserve the marginal distribution.

	2-D Cost			2-D KL		32-D Cost			32-D KL		64-D Cost			64-D KL	
Cost Fn. $c(x_0, x_1)$	B	B-ST	B-FM	B-ST	B-FM	B	B-ST	B-FM	B-ST	B-FM	B	B-ST	B-FM	B-ST	B-FM
$\|x_1 - x_0\|_2^2$	0.90	0.60	0.72	0.07	4E-3	41.08	31.58	38.73	151.47	0.06	92.90	65.57	87.97	335.38	0.14
$\|x_1 - x_0\|_1$	1.09	0.86	0.98	0.18	4E-3	27.92	24.51	27.26	254.59	0.08	60.27	50.49	58.38	361.16	0.16
$1 - \frac{\langle x_0, x_1 \rangle}{\|x_0\| \|x_1\|}$	0.03	2E-4	3E-3	5.91	4E-3	0.62	0.53	0.58	179.48	0.06	0.71	0.60	0.68	337.63	0.12
$\|A(x_1 - x_0)\|_2^2$	0.91	0.54	0.65	0.07	4E-3	32.66	24.61	30.13	256.90	0.06	78.70	58.11	78.50	529.09	0.19

5.6.3 Improved Batch Optimal Couplings

We further explore the usage of Multisample Flow Matching as an approach to improve upon batch optimal solutions. Here, we experiment with a different setting, where the cost is unknown and only samples from a batch optimal coupling are provided. In the real world, it is often the case that the preferences of each person are not known explicitly, but when given a finite number of choices, people can more easily find their best assignments. This motivates us to consider the case of unknown cost functions, and information regarding the optimal coupling is only given by a weak oracle that acts on finite samples, denoted $q^k_{OT,c}$. We consider two baselines: (i) the BatchOT cost (B) which corresponds to $\mathbb{E}_{q^k_{OT,c}(x_0,x_1)}[c(x_0, x_1)]$, and (ii) learning a static map that mimics the BatchOT couplings (B-ST) by minimizing the following objective:

$$\mathbb{E}_{q^k_{OT,c}(x_0,x_1)} \|x_1 - \psi_\theta(x_0)\|^2 . \tag{5.17}$$

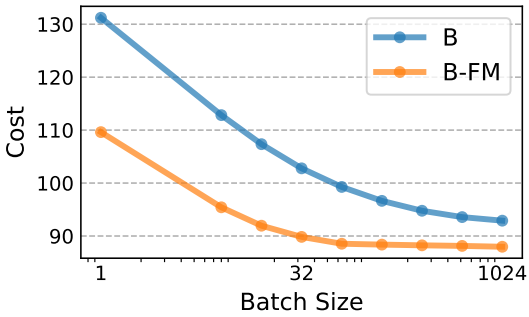


Figure 5.7: Transport cost vs. batch size (k) for computing couplings on the 64D synthetic dataset. The number of samples used for performing gradient steps during training and the resulting KL divergences were kept the same.

This can be viewed as learning the barycentric projection [Fer+14; Seg+17], i.e.$\psi^*(x_0) = E_{q^k_{OT,c}(x_1|x_0)}[x_1]$, a well-studied quantity but is known to not preserve the marginal distribution [Fat+20].

We experiment with 4 different cost functions on three synthetic datasets in dimensions $\{2, 32, 64\}$ where both q_0 and q_1 are chosen to be Gaussian mixture models. In Table 5.4 we report both the transport cost and the KL divergence between q_1 and the distribution induced by the learned map, i.e.$[\psi_1]_\sharp q_0$. We observe that while B-ST always results in lower transport costs compared to B-FM, its KL divergence is always very high, meaning that the pushed-forward distribution by the learned static map poorly approximates q_1. Another interesting observation is that B-FM always reduces transport costs compared to B, providing experimental support to the theory (Theorem 11).

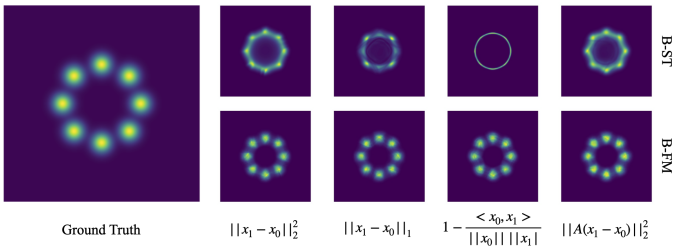


Figure 5.6: 2D densities on the 8-Gaussians target distribution. (Left) Ground truth density. (Right) Learned densities with static maps in the top row and Multisample Flow Matching dynamic maps in the bottom row. Models within each column were trained using batch optimal couplings with the corresponding cost function.

Flow Matching improves optimality Figure 5.7 shows the cost of the learned model as we vary the batch size for computing couplings, where the models are trained sufficiently to achieve the same KL values as reported in Table 5.4. We see that our approach decreases the cost compared to the BatchOT oracle for any fixed batch size, and furthermore, converges to the OT solution faster than the batchOT oracle. Thus, since Multisample Flow Matching retains the correct marginal distributions, it can be used to better approximate optimal transport solutions than simply relying on a minibatch solution.

Chapter 6

Differentiating through Flows for Controlled Generation

Controlled generation from generative priors is of great interest in many domains. Various problems such as conditional generation, inverse problems, sample editing, etc., can all be framed as a controlled generation problem. In this chapter, we explore controlled generation from diffusion/flow generative prior [SE19; HJA20; Lip+23] as they are the central research paradigm in this study, as well as being the current state-of-the-art generative approaches across various data modalities.

In [Ben+24] we introduce a framework for adding controlled generation to a pre-trained Diffusion or Flow-Matching (FM) model based on *differentiation through the ODE sampling process*. Our key observation is that for Diffusion/FM models trained with standard Gaussian probability paths, differentiating an arbitrary loss $\mathcal{L}(x)$ through the generation process of x with respect to the initial point, x_0, projects the gradient $\nabla_x \mathcal{L}$ onto the "data manifold", i.e., onto major data directions at x, implicitly injecting a valuable prior. Based on this observation we advocate a simple general algorithm that minimizes an arbitrary cost function $\mathcal{L}(x)$, representing the desired control, as a function of the source noise point x_0 used to generate x.

6.1 Motivation and Contributions

Controlled generation from diffusion/flow models can be roughly classified to three main approaches: (i) conditional training, where the model receives the condition as an additional input during training [Son+21b; DN21; HS21], although performing very well this approach requires task specific training of a generative model which in cases may be prohibitive; (ii) training-free approaches that modify the generation process of a pre-trained model, adding additional guidance [Bar+23; Yu+23]. The guidance is usually built upon strong assumptions on the generation process that can lead to errors in the generation and mostly limit the method to observations that are linear in the target [Kaw+22; Chu+22; Son+23a; Pok+23]; lastly, (iii) adopt a variational perspective, framing the controlled generation as an optimization problem [Gra+22; Mar+23; Wal+23; Sam+23b], requiring

only a differentiable cost to enforce the control. This paper belongs to this third class.

The goal of this paper is to introduce a framework for adding controlled generation to a pre-trained Diffusion or Flow-Matching (FM) model based on *differentiation through the ODE sampling process*. Our key observation is that for Diffusion/FM models trained with standard Gaussian probability paths, differentiating an arbitrary loss $\mathcal{L}(x)$ through the generation process of x with respect to the initial point, x_0, projects the gradient $\nabla_x \mathcal{L}$ onto the "data manifold", i.e., onto major data directions at x, implicitly injecting a valuable prior. Based on this observation we advocate a simple general algorithm that minimizes an arbitrary cost function $\mathcal{L}(x)$, representing the desired control, as a function of the source noise point x_0 used to generate x. That is,

$$\min_{x_0} \mathcal{L}(x). \tag{6.1}$$

Differentiating through a generator of a GAN or a normalizing flow was proven generally useful for controlled generation [Bor+17; Asi+20; WLD21] and counterfactual examples [DGK21; Dom+24]. Recently, [Wal+23; Sam+23b] have been suggesting to differentiate through a discrete diffusion solver for the particular tasks of incorporating classifier guidance and generating rare concepts. In this paper we generalize this idea in two ways: (i) we consider general flow models trained with Gaussian probability paths, including Diffusion and Flow-Matching models; and (ii) we demonstrate, both theoretically and practically, that the inductive bias injected by differentiating through the flow is applicable to a much wider class of problems modeled by general cost functions.

We experiment with our method on a variety of settings and applications: Inverse problems on images using conditional ImageNet and text-2-image (T2I) generative priors, conditional molecule generation with QM9 unconditional generative priors, and audio inpainting and super-resolution with unconditional generative prior. In all application we were able to achieve state of the art performance without carefully tuning the algorithm across domains and applications. One drawback of our method is the relative long time for generation (usually $5-15$ minutes on ImageNet-128 on an NVidia V100 GPU) compared to some baselines, however the method's simplicity and its superior results can justify its usage and adaptation in many use cases. Furthermore, we believe there is great room for speed improvement.

To summarize, our contributions are:

- We formulate the controlled generation problem as a simple source point optimization problem using general flow generative models.
- We show that source point optimization of flows trained with Gaussian probability paths inject an implicit bias exhibiting a data-manifold projection behaviour to the cost function's gradient.
- We empirically show the generality and the effectiveness of the proposed approach for different domains.

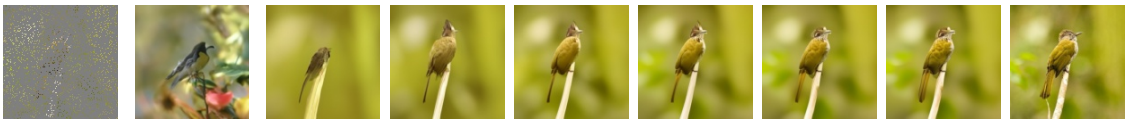


Figure 6.1: Intermediate $x(1)$ during optimization. Given a distorted image and randomly initialized x_0 defining the initial $x(1)$, our optimization travels close to the natural image manifold passing through in-distribution images on its way to the GT sample from the face-blurred ImageNet-128 validation set.

6.2 Controlled Generation via Source Point Optimization

Our method considers generative flow models, including Continuous Normalizing Flows (CNFs) [Che+18; Lip+23] and (deterministic sampling of) Diffusion Models [Son+21b]. These models generate samples $x(1) \in \mathbb{R}^d$ by first sampling from some source (noise) distribution $x(0) \sim p_0(x_0)$ and then solving an Ordinary Differential Equation (ODE),

$$\dot{x}(t) = u_t(x(t)), \tag{6.2}$$

from time $t = 0$ to time $t = 1$, using a predetermined velocity field $u : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$. We denote by p_1 the distribution and density function of $x(1)$ given $x(0) \sim p_0(x_0)$.

Given a pre-trained (frozen) flow model, $u_t(x)$, represented by a neural network and some cost function $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}_+$, our goal is to find likely samples x that provide low cost $\mathcal{L}(x)$ and are likely under the flow model's distribution p_1. We advocate a general framework formulating this problem as the following optimization problem

$$\min_{x_0} \quad \mathcal{L}(x(1)), \tag{6.3}$$

where in general $\mathcal{L}$ can also incorporate multiple costs including potentially a regularization term that can depend on x_0 and u,

$$\tilde{\mathcal{L}}(x) = \mathcal{L}(x) + \mathcal{R}(x_0, u). \tag{6.4}$$

In this formulation, the sample $x(1)$ is constrained to be a solution of the ODE equation 6.2, with initial boundary condition $x(0) = x_0$, where x_0 is the only optimized quantity and $\mathcal{L}$ is the desired cost function.

Optimizing equation 6.3 is done by computing the gradients of the loss w.r.t. the optimized variable x_0 as listed in Algorithm 1. We call this method *D-Flow*. To better understand the generality of this framework we next consider several instantiations of equation 6.3.

6.2.1 Cost Functions

Reversed sampling. First, consider the simple case where $\mathcal{L}(x) = \|x - y\|^2$. In this case, the solution of 6.3 will be the x_0 that that has an ODE trajectory that reaches y at $t = 1$, i.e., $x(1) = y$. Note that since (under some mild assumptions on $u_t(x)$) equation 6.2 defines a diffeomorphism $\mathbb{R}^d \to \mathbb{R}^d$, for an arbitrary $y \in \mathbb{R}^d$, there exists a unique solution $x_0 \in \mathbb{R}^d$ to equation 6.3.

Inverse problems. In this case we have access to some known corruption function $H : \mathbb{R}^d \to \mathbb{R}^n$ and a corrupted sample from an unknown ground truth signal x_*,

$$y = H(x_*) + \epsilon, \tag{6.5}$$

where $\epsilon \sim \mathcal{N}(\epsilon)$ is an optional additive noise. The goal is to recover an x that produces y and the cost function is usually

$$\mathcal{L}(x) = \|H(x) - y\|^2, \tag{6.6}$$

where the norm can be some arbitrary L_p norm or even a general loss $\ell(H(x), y)$ comparing $H(x)$ and y. Specific choices of the corruption function H can lead to common applications: *Image inpainting* corresponds to choosing the corruption function H to sub-sample known $n < d$ pixels out of d total pixels; *Image deblurring* corresponds to taking $H : \mathbb{R}^d \to \mathbb{R}^d$ to be a blurring function, e.g., a convolution with a blurring kernel; *Super-resolution* corresponds to $H : \mathbb{R}^d \to \mathbb{R}^{d/k}$ lowering the dimension by a factor of k.

Conditional sampling. Another important application is to guide the sampling process to satisfy some condition y. In this case, we can take $\mathcal{L}(x)$ to encourage a classifier or some energy function to reach a particular class or energy y. For example, let $\mathcal{F} : \mathbb{R}^d \to \mathbb{R}$ and we would like to generate a sample from a certain level set $c \in \mathbb{R}$ we can use the loss

$$\mathcal{L}(x) = (\mathcal{F}(x) - c)^2. \tag{6.7}$$

6.2.2 Initialization

The initialization of x_0 can have a great impact on the convergence of the optimization of equation 6.3. A natural choice will be to initialize x_0 with a sample from the source distribution $p_0(x_0)$. We find that for cases when an observed signal y provides a lot of information about the desired x, one can improve the convergence speed of the optimization. For example, in linear inverse problems on images, where the observed y has a strong prior on the structure of the image, it is beneficial to initialize x_0 with a blend of a sample from the source distribution and the backward solution of the ODE from $t = 1$ to $t = 0$:

$$x_0 = \sqrt{\alpha} \cdot y(0) + \sqrt{1 - \alpha} \cdot z, \tag{6.8}$$

where $z \sim p_0(x_0)$ and $y(0) = y + \int_1^0 u(t, y(t))dt$.

6.2.3 Regularizations

The formulation in equation 6.3 allows including different regularizations $\mathcal{R}$ (equation 6.4) discussed next. Maybe the most intriguing of these regularizations, and the main point of this paper, is the *implicit regularization*, i.e., corresponding to $\mathcal{R} \equiv 0$, discussed last in what follows.

Regularizing the target $x(1)$. Maybe the most natural is incorporating the negative log likelihood (NLL) of the sample $x(1)$, i.e., $\mathcal{R} = -\log p_1(x(1))$ in equation 6.4. This prior can be incorporated by augmenting $x(t) \in \mathbb{R}^d$ with an extra coordinate $z \in \mathbb{R}$ and formulate equation 6.3 as

$$\min_{x_0} \quad \mathcal{L}(x(1)) - z(1) \tag{6.9a}$$

$$\text{s.t.} \quad \dot{x}(t) = u_t(x(t)), \qquad x(0) = x_0 \tag{6.9b}$$

$$\dot{z}(t) = -\operatorname{div} u_t(x(t)), \quad z(0) = \log p_0(x_0) \tag{6.9c}$$

Indeed, solving the ODE system defined by equations 6.9b and 6.9c for times $t \in [0, 1]$ provides $z(1) = \log p_1(x(1))$, see [Che+18]. However, aside from the extra complexity introduced by the divergence term in the ODE in equation 6.9c (see e.g., [Gra+19] for ways to deal with this type of ODE) it is not clear whether likelihood is a good prior in deep generative models in high dimensions [Nal+19]; In Figure 6.2 we compare bits-per-dimension (BPD) of a test image of ImageNet-128 and a masked version of this image, providing a more likely image according to our flow model trained on ImageNet.

BPD=2.02 BPD=1.84

Figure 6.2: BPD of two images in an ImageNet-128 model.

Regularizing the source $x(0) = x_0$. Another option is to regularize the source point $x(0) = x_0$. The first choice would again be to incorporate the NLL of the noise sample, i.e., $\mathcal{R} = -\log p_0(x_0)$, which for standard noise $p_0(x_0) = \mathcal{N}(x_0|0, I)$ would reduce to $\mathcal{R} = c + \frac{1}{2}\|x_0\|^2$, where c is a constant independent of x_0. This however, would attract x_0 towards the most likely all zero mean but far from most of the probability mass at norm $\sqrt{d}$.

Following [Sam+23a] we instead prefer to make sure x_0 stays in the area where most mass of p_0 is concentrated and therefore use the χ^d distribution, which is defined as the probability distribution $p(r)$ of the random variable $r = \|x_0\|$ where $x_0 \sim \mathcal{N}(x_0|0, I)$ is again the standard normal distribution. The NLL of r in this case is

$$\mathcal{R} = -\log p(r) = c + (d-1)\log\|x_0\| - \frac{\|x_0\|^2}{2}, \tag{6.10}$$

where c is a constant independent of x_0.

Implicit regularization. Maybe the most interesting and potentially useful regularization in our formulation (equation 6.3) comes from the choice of optimizing the cost $\mathcal{L}(x(1))$ as a function of the source point $x(0) = x_0$. For standard diffusion/flow models that are trained to zero loss:

Optimizing the cost $\mathcal{L}(x(1))$ with respect to x_0 follows the data distribution $p_1(x_1)$ by projecting the gradient $\nabla_{x(1)}\mathcal{L}(x(1))$ with the local data covariance matrix.

This is intuitively illustrated in Figure 6.3: while moving in direction of the gradient $\nabla_{x(1)}\mathcal{L}(x(1))$ generally moves away from the data distribution (in pink), differentiating w.r.t. $x(0)$ projects this gradient onto high variance data directions and consequently staying close to the data distribution. To exemplify this phenomena we show in Figure 6.1 optimization steps $x^{(0)}(1), x^{(2)}(1), x^{(4)}(1), \ldots$ of a loss $\mathcal{L}(x) = \|H(x) - H(x_*)\|^2$, where H is a linear matrix that subsamples a (random) subset of the image's pixels consisting of 90% of the total number of pixels, and x_* is a target image (different from the initial $x^{(0)}(1)$). The sampling process here is using an ImageNet trained flow model with the class condition 'bulbul'. As can be seen in this sequence of images, the intermediate steps of the optimization stay close to the distribution and pass through different sub-species of the bulbul bird. In the next section we provide a precise mathematical statement supporting this claim but for now let us provide some intuitive explanation.

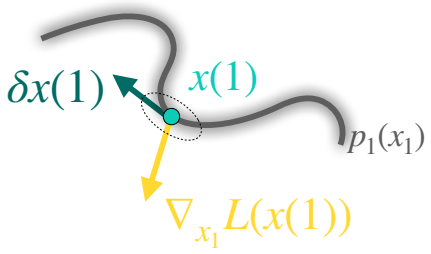


Figure 6.3: Implicit bias in differentiating through the solver.

6.2.4 Practical Implementation

The practical implementation of Algorithm 1 requires three algorithmic choices. First, one needs to decide how to initialize x_0. In all experiments we either initialize x_0 as a sample from the source distribution, i.e., normal Gaussian, or we use a variance preserving blend of a normal Gaussian with the backward solution from $t = 1$ to $t = 0$ of the observed signal when possible. Second, we need to choose the solver used to parameterize $x(1)$. To this end we utilize the `torchdiffeq` package [Che18], providing a wide class of differentiable ODE solvers. Backpropagating through the solver can be expensive in memory and we therefore use gradient checkpointing to reduce memory consumption at the cost of runtime. In most of our experiments we use the midpoint method with 6 function evaluations. Lastly, we need to choose the optimizer for the gradient step. Since the optimization we perform is not stochastic we choose to use the LBFGS algorithm with line search in all experiments. The runtime of the optimization depends on the problem but typically ranges from $5 - 15$ minutes per sample. For large text-2-image and text-2-audio models run times are higher and can reach $30 - 40$ minutes.

Algorithm 1 D-Flow Algorithm.

input : condition y, cost $\mathcal{L}$, pre-trained flow model $u_t(x)$
Initialize $x_0^{(0)} \leftarrow \texttt{Init}(y)$; ▷ Initialize source point
for $i = 1, \ldots, N$ **do**
  $x^{(i)}(1) \leftarrow \texttt{solve}(x_0^{(i)}, u_t)$; ▷ Solve ODE numerically
  $x_0^{(i+1)} \leftarrow \texttt{optimize\_step}(x_0^{(i)}, \nabla_{x_0}\mathcal{L}(x^{(i)}(1)))$; ▷ Gradient descent step
end
output: $x^N(1)$

6.3 Optimization Dynamics Analysis

In this section we provide the theoretical support to the implicit regularization claim made in the previous section. First, we revisit the family of Affine Gaussian Probability Paths (AGPP) taking noise to data that are used to supervise diffusion/flow models. When diffusion/flow models reach zero loss they reproduce these probability paths and we will therefore use them to analyze the implicit bias. Second, we use the method of adjoint dynamics to provide an explicit formula for the gradient $\nabla_{x_0}\mathcal{L}(x(1))$ under the AGPP assumption, and consequently derive the variation in $x(1)$. Lastly, we interpret this variation to demonstrate why it is pointing in the direction of the data distribution.

Affine Gaussian probability paths. Diffusion and recent flow based models use Affine Gaussian Probability Path (AGPP) to supervise their training. In particular, denoting $p_0 = \mathcal{N}(0, \sigma_0^2 I)$ the Gaussian noise (source) distribution and p_1 data (target) distribution, an AGPP is defined by

$$p_t(x) = \int p_t(x|x_1)p_1(x_1)dx_1, \tag{6.11}$$

where $p_t(x|x_1) = \mathcal{N}(x|\alpha_t x_1, \sigma_t^2 I)$ is a Gaussian kernel and $\alpha_t, \sigma_t : [0,1] \to [0,1]$ are called the *scheduler*, satisfying $\alpha_0 = 0$, $\sigma_1 \approx 0$, and $\alpha_1 = 1 = \sigma_0$, consequently guaranteeing that p_t interpolates (exactly or approximately) the source and target distributions at times $t = 0$ and $t = 1$, respectively. The velocity field that generates this probability path and coincide with the velocity field trained by diffusion/flow models at zero loss is [Sha+23]

$$u_t(x) = \int \left[a_t x + b_t x_1\right] p_t(x_1|x)dx_1 \tag{6.12}$$

where using Bayes' Theorem

$$p_t(x_1|x) = \frac{p_t(x|x_1)p_1(x_1)}{p_t(x)}, \tag{6.13}$$

and

$$a_t = \frac{\dot{\sigma}_t}{\sigma_t}, \quad b_t = \dot{\alpha}_t - \alpha_t \frac{\dot{\sigma}_t}{\sigma_t}. \tag{6.14}$$

One can also simplify the integral in equation 4.4 and write the marginal velocity field in terms of the denoiser [Kar+22], $\hat{x}_{1|t}(x) = \int x_1 p_t(x_1|x)dx_1$:

$$u_t(x) = a_t x + b_t \hat{x}_{1|t}(x) \tag{6.15}$$

which for AGPP possesses a useful property we will use in Theorem 7, stated in the following proposition (proof in Appendix A.4.1):

Proposition 1. *For AGPP, the gradient of the denoiser $\hat{x}_{1|t}(x)$ w.r.t x is proportional to the variance of the random variable defined by $p_t(x_1|x)$, formally:*

$$D_x\hat{x}_{1|t}(x) = \frac{\alpha_t}{\sigma_t^2}\mathrm{Var}_{1|t}(x) \tag{6.16}$$

where

$$\mathrm{Var}_{1|t}(x) = \mathbb{E}_{p_t(x_1|x)}\left[x_1 - \hat{x}_{1|t}(x)\right]\left[x_1 - \hat{x}_{1|t}(x)\right]^T \tag{6.17}$$

Differentiating through the solver. When diffusion/flow models are optimized to a minimal loss they perfectly reproduce the AGPP velocity field, i.e., equation 4.4 [Lip+23]. For this velocity field we begin with an analysis of the differential of a solution (sample) $D_{x_0}x(1)$ for the continuous time exact case and a discrete time approximation.

Theorem 7. *For AGPP velocity field u_t (see equation 4.4) and $x(t)$ defined via equation 6.2 the differential of $x(1)$ as a function of x_0 is*

$$D_{x_0}x(1) = \sigma_1\mathcal{T}\exp\left[\int_0^1 \gamma_t\mathrm{Var}_{1|t}(x(t))dt\right], \tag{6.18}$$

where $\mathcal{T}\exp[\cdot]$ stands for a time-ordered exponential, $\gamma_t = \frac{1}{2}\frac{d}{dt}\mathrm{snr}(t)$ and we define $\mathrm{snr}(t) = \frac{\alpha_t^2}{\sigma_t^2}$.

The proof is given in Appendix A.4.2. In the exact case where $\sigma_1 = 0$ we also have $\int_0^1 \gamma_t dt = \infty$, nevertheless we show in Appendix A.4.2 that $D_{x_0}x(1)$ is the time-ordered exponential of a bounded time-dependent matrix. While a closed form expression to this integral is unknown, we note that the matrix-vector product $D_{x_0}x(1)v$ corresponds to an infinite sum of powers of the matrices $\gamma_t\mathrm{Var}_{1|t}(x(t))$ applied to v.

To gain better intuition and align our theory with practice, where discrete ODE solvers are used to obtain $x(1)$, we will now analyze the discrete time solver case. Let us consider an Euler ODE solver with N uniform steps of size $h = \frac{1}{N}$, then the differential of $x(1)$ as a function of x_0 is:

$$D_{x_0}x(1) = \prod_{m=0}^{N-1}\Big((1 + ha_{mh})I + h\gamma_{mh}\mathrm{Var}_{1|mh}(x_{mh})\Big), \tag{6.19}$$

note that the product is a time-ordered product, with m decreasing from right to left (derivation in Appendix A.4.3). The form of equation 6.19, consisting of powers of $\mathrm{Var}_{1|t}(x)$, provides insights as to why D-Flow works even with a low number of solver steps. Intuitively, the vector-matrix multiplication $\mathrm{Var}_{1|t}(x)v$ projects v on the major axes of the distribution of the data conditioned on x. As we will soon see, $D_{x_0}x(1)$ is key to understanding the implicit bias claim.

The dynamics of $x(1)$. Consider an optimization step updating the optimized variable x_0 with a gradient step, i.e., $x_0^\tau = x_0 - \tau \nabla_{x_0} \mathcal{L}(x(1))$, where the gradient $\nabla_{x_0} \mathcal{L}(x(1))$ can be now computed with the chain rule and equation 6.18,

$$\nabla_{x_0} \mathcal{L}(x(1)) = D_{x_0} x(1)^T \nabla_{x(1)} \mathcal{L}(x(1)), \tag{6.20}$$

We can now ask: *How is the sample $x(1)$ changing infinitesimally under this gradient step?* Denote by $\Psi : \mathbb{R}^d \to \mathbb{R}^d$ the map taking initial conditions x_0 to solutions of equation 6.2 at $t = 1$, i.e., $\Psi(x_0) = x(1)$. The *variation* of $x(1)$ is

$$\begin{aligned} \delta x(1) &= \frac{d}{d\tau}\Big|_{\tau=0} \Psi\left(x_0 - \tau \nabla_{x_0} \mathcal{L}(x(1))\right) \\ &= -\left[D_{x_0} x(1) D_{x_0} x(1)^T\right] \nabla_{x(1)} \mathcal{L}(x(1)), \end{aligned}$$

where the first equality is the definition of variation and the second equality is using chain rule and equation 6.20. Indeed, the dynamics of $x(1)$ follow the projection of the gradient $\nabla_{x(1)} \mathcal{L}(x(1))$ with the operator $D_{x_0} x(1)$ that iteratively applies projection by the covariance matrix $\mathrm{Var}_{1|t}(x(t))$ at different times t (equations 6.18 and 6.19).

6.4 Related Work

Inverse Problems. A new line of works alter the diffusion generation process for training-free solutions of inverse problems. Most works can be viewed as building guidance strategies to the generation process of diffusion models. [Kaw+22] takes a variational approach deriving a solver for linear inverse problems. Similarly, [Chu+22; WYZ23] modify the generation process by enforcing consistency with the observations either via cost functions or projections [Cho+21; WYZ23; Lug+22]. Other approaches guide the sampling process with derivatives through the diffusion model at each denoising step [Ho+22; Chu+23; Son+23a; Pok+23]. A recent work by [Rou+23] extends the ideas for latent diffusion models by chained applications of encoder-decoder. Similar to our approach [Mar+23] performs optimization of a reconstruction loss with score matching regularization.

Conditional sampling. Conditional sampling from diffusion models can be achieved by training an additional noise-aware condition predictor model [Son+21b] or by incorporating the condition into the training process [DN21; HS21]. These approaches however require task specific training. Plug-and-play approaches, on the other hand, utilize a pretrained unconditional generative model as a prior. [Gra+22] perform constrained generation via optimization of a reconstruction term regularized by the diffusion loss. [Liu+23b] seeks for optimal control optimizing through the generation process to learn guiding controls. Our method formulates a similar optimization problem like earlier works on GANs [Bor+17] and normalizing flows [Asi+20; DGK21; Chá22]. While [Asi+20] provides an analysis of a simplified linear model, [DGK21] analyzes the manifold preserving properties of diffeomorphic generative models. Our work provides a novel theoretical analysis of the gradient of differentiable functionals with respect to initial values of diffusion/flow

Table 6.1: Quantitative evaluation of linear inverse problems on face-blurred ImageNet-128.

	Inpainting-Center				Super-Resolution X2				Gaussian deblur			
Method	FID ↓	LPIPS ↓	PSNR ↑	SSIM ↑	FID ↓	LPIPS ↓	PSNR ↑	SSIM ↑	FID ↓	LPIPS ↓	PSNR ↑	SSIM ↑
$\sigma_y = 0$												
ΠGDM [Son+23a]	5.73	0.096	36.89	0.908	6.01	0.104	34.31	0.911	4.27	0.066	37.61	0.961
OT-ODE [Pok+23]	5.65	0.094	37.00	0.893	4.28	0.097	33.88	0.903	2.04	0.048	37.44	0.959
RED-Diff [Mar+23]	5.40	0.068	38.91	0.928	3.05	0.091	33.74	0.900	1.62	0.055	35.18	0.937
Ours	4.14	0.072	37.67	0.922	2.50	0.069	34.88	0.924	2.37	0.035	39.47	0.976
$\sigma_y = 0.05$												
ΠGDM [Son+23a]	7.99	0.122	34.57	0.867	4.38	0.148	32.07	0.831	30.30	0.328	29.96	0.606
OT-ODE [Pok+23]	6.25	0.119	35.01	0.882	4.61	0.149	32.59	0.862	4.84	0.175	31.94	0.821
RED-Diff [Mar+23]	14.63	0.171	32.42	0.820	10.54	0.182	31.82	0.852	21.43	0.229	31.41	0.807
Ours	4.76	0.102	34.609	0.890	4.26	0.146	32.35	0.858	5.35	0.167	31.99	0.820

generative processes with affine Gaussian paths. Our analysis unravels a fresh perspective on the implicit regularization implemented by differentiating through the generation process, even with a few number of steps (Appendix A.4.3), that aligns with the denoising attributes of diffusion/flow models. We note that using gradients through the solver for the case of discrete diffusion models was first used by [Wal+23] for classifier guidance and by [Sam+23b] to generate rare samples.

6.5 Experiments

We test D-Flow on the tasks: linear inverse problems on images, inverse problems with latent flow models and conditional molecule generation. For all the inverse problems experiments, where the observed signal provides structural information, we use a blend initialization to our algorithm speeding up convergence and often improving performance. Furthermore, in most experiments we find that there is no need in adding an explicit regularizing term in the optimization. The only cases where we found regularization helpful was in the noisy case for linear inverse problems and molecule generation. Additional details are in Appendix B.4.1.

6.5.1 Linear Inverse Problems on Images

We validate our method on standard linear inverse problems with a known degradation model on images. The tasks we consider are center-crop inpainting, super-resolution and Gaussian deblurring both in the noiseless and noisy case. In all cases we stop the optimization at a task dependent target PSNR. For the noisy case we choose the target PSNR to be the PSNR corresponding to the known added noise.

Tasks. We follow the same settings as in [Pok+23]: (i) For center-crop inpainting, we use a 40×40 centered mask; (ii) for super-resolution we use bicubic interpolation to downsample the images by $\times 2$; and lastly (iii) for Gaussian deblur we apply a Gaussian blur kernel of size 61×61 with intensity 1. For each task we report results for the noiseless and noisy (Gaussian noise of $\sigma_y = 0.05$, see equation 6.5) cases. Further implementation details can be found in the Appendix B.4.1.

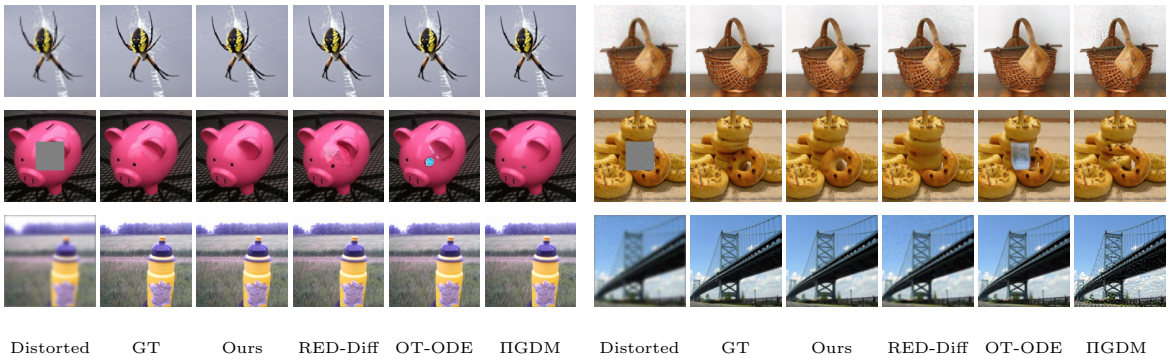


Figure 6.4: Qualitative comparison for linear inverse problems on ImageNet-128. GT samples from ImageNet-128 validation.

Metrics. Following the evaluation protocol of prior works [Chu+22; Kaw+22] we report Fréchet Inception Distance (FID) [Heu+17], Learned Perceptual Image Patch Similarity (LPIPS) [Zha+18], peak signal-to-noise ratio (PSNR), and structural similarity index (SSIM).

Datasets and baselines. We use the face-blurred ImageNet-128 dataset and report our results on the $10k$ split of the face-blurred ImageNet dataset used by [Pok+23]. We compare our method to three recent state of the art methods: ΠGDM [Son+23a], OT-ODE [Pok+23] and RED-Diff [Mar+23]. We use the implementation of [Pok+23] for all the baselines. All methods, including ours, are evaluated with the same Cond-OT flow-matching class conditioned model trained on the face-blurred ImageNet-128 unless the reults we produced were inferior to the ones reported in [Pok+23]. In that case, we use the reported numbers from [Pok+23].

Results. As shown in Table 6.1, our method shows strong performance across all tasks, Figure 6.4 shows samples for each type of distortion. For inpainting and super-resolution our method improves upon state of the art in most metrics. We believe that our method's ability to reach images with higher fidelity to the ground truth is attributed to the source point optimization, which, differently from guided sampling approaches such as [Son+23a; Pok+23], iteratively correct the sampling trajectory to better match the observed signal. We further note that compared to RED-Diff, which is also an optimization approach, our method does not struggle in the noisy case and achieves SOTA performance. We show more samples in Figures B.14,B.15.

6.5.2 Inverse Problems with Latent Flow Models

Image Inpainting

We demonstrate the capability of our approach for non-linear inverse problems by applying it to the task of free-form inpainting using a latent T2I FM model.

Metrics. To quantitatively assess our results we report standard metrics used in T2I generation: PSNR, FID [Heu+17], and Clip score [Ram+22].

Datasets and baselines. The T2I model we use was trained on a proprietary

Table 6.2: Quantitative evaluation of free-form inpainting on MS-COCO with T2I latent model.

	Inpainting-Free-Form				
Method	FID ↓	LPIPS ↓	PSNR ↑	SSIM ↑	Clip score ↑
RED-Diff [Mar+23]	23.31	0.327	33.28	0.813	0.882
Ours	16.92	0.327	32.34	0.759	0.892

Table 6.3: Quantitative evaluation of music generation with latent flow models.

	Inpainting (10%)		Inpainting (20%)		Super-Resolution X2		Super-Resolution X4		Super-Resolution X8	
Method	FAD ↓	PSNR ↑	FAD ↓	PSNR ↑	FAD ↓	PSNR ↑	FAD ↓	PSNR ↑	FAD ↓	PSNR ↑
In-domain										
RED-Diff [Mar+23]	0.75	31.19	0.78	29.99	0.93	35.27	1.63	33.51	1.73	29.12
Ours	0.22	31.02	0.49	29.57	0.22	44.51	0.50	42.64	1.01	36.50
MusicCaps										
RED-Diff [Mar+23]	3.59	32.81	3.72	30.39	3.07	37.13	3.51	34.99	3.97	30.49
Ours	1.19	31.78	1.31	31.08	1.25	38.93	1.42	35.83	2.09	32.20

dataset of $330m$ image-text pairs. It was trained on the latent space of an autoencoder as in [Rom+22a]. The architecture is based on GLIDE [Nic+22] and uses a T5 text encoder [Raf+23]. We evaluate on a subset of $1k$ samples from the validation set of the COCO dataset [Lin+15]. We compare our method to RED-Diff [Mar+23] as it is also not limited to linear inverse problems like the other baselines we used in the previous section. We tested different hyper-parameters for RED-Diff and report results with the best.

Results. Table 6.2 reports metrics for the baseline and our method. The metrics indicate that while RED-Diff better matches the unmasked areas, achieving superior performance for structural metrics (PSNR, SSIM) our method produces more semantically plausible image completion winning in perceptual metrics. We do observe that RED-Diff often produces artifacts for this task. Results are visualized in Figure B.16.

Audio Inpainting and Super-Resolution

We evaluate our method on the tasks of music inpainting and super-resolution, utilizing a latent flow-matching music generation model. For this, we used a trained Cond-OT flow-matching text conditioned model with a transformer architecture of 325m parameters that operates on top of EnCodec representation [Déf+23]. The model's performance aligns with the current state-of-the-art scores in text-conditional music generation, achieving a Fréchet Audio Distance (FAD) score of 3.13 [Kil+18] on MusicCaps and FAD of 0.72 on in-domain data. The model is trained to generate ten-seconds samples. In the following, we evaluate the performance of inpainting and super-resolution using our method and RED-Diff as baseline, we report FAD and PSNR metrics.

Datasets and baselines. For evaluation, we use the MusicCaps benchmark, which comprises of 5.5K pairs of music and a textual description and an internal (in-domain) evaluation set of 202 samples, similar to [Cop+23; Ziv+24]. Similar to prior work, we compute FAD metric using VGGish. We compare our method to RED-Diff [Mar+23].

Results. Table 6.3 studies our method in inapinting and super resolution tasks. This experiment demosntrates the ability of our method to work in non-linear setup, where the flow model is trained over a neural representation and the cost function is evaluated on

52.22 63.60 72.63 79.36 81.33 88.88 95.13 102.40

Figure 6.5: Qualitative visualization of controlled generated molecules for various polarizability (α) levels.

the post-decoded signal (neural representation after decoding). In the inpainting task, we center crop the signal by 10% and 20%, i.e., for a ten-seconds signal, we mask out two and four seconds respectively. In the super-resolution task we upscale a signal by factors of two, four, and eight,i.e., from 4kHz, 8kHz, 16kHz to 32kHz respectively. Overall, our method improves upon the baseline. Specifically, in all experiments, our method obtain the lowest FAD metric. In the inpainting task our method obtains a slightly lower PSNR from the baseline. Audio samples are attached in a supplementray material. Additional implementation details appear in Appendix B.4.1.

6.5.3 Conditional Molecule Generation on QM9

In this experiment we illustrate the application of our method for controllable molecule generation, which is of practical significance in the fields of material and drug design. The properties targeted for conditional generation (c in equation 6.7) include polarizability α, orbital energies $\varepsilon_{HOMO}, \varepsilon_{LUMO}$ and their gap $\Delta\varepsilon$, Diople moment μ, and heat capacity C_v. To assess the properties of the molecules generated, we used a property classifier ($\mathcal{F}$ in equation 6.7) for each property. Those classifiers were trained following the methodology outlined in [Hoo+22]. Further details are in Appendix B.4.1.

Metrics. To assess conditional generation, we calculate the Mean Absolute Error (MAE) between the predicted property value of the generated molecule by the property classifier, [Sat+22], and the target property value. According to the conditional training protocol from [Hoo+22], the property classifier is trained over half of the QM9 train set (50K) while the remaining half is used for training the conditional generative models. Additionally, we appraise the quality of the generated molecules by evaluating atom stability (the percentage of atoms with correct valency), molecule stability (the percentage of molecules where all atoms are stable), validity (as defined in RDKit [Lan16]), and the uniqueness of the generated molecules.

Dataset and baselines. The generative models used for this experiment are trained using the QM9 dataset [Ram+14], a commonly used molecular dataset containing small molecules with up to 29 atoms. The model we use as prior in these experiments is an unconditional equivariant Flow-Matching model with CondOT path [Lip+23], trained on the train set half used in [Hoo+22] for conditional training. We compare our method to several state of the art *conditional models*: conditional EDM, Equivariant Flow-Matching (EquiFM) [Son+23b], and Geometric Latent Diffusion Model (GeoLDM)[Xu+23] an equivariant latent diffusion model. Additionally, we report the test MAE of each property classifier (denoted as QM9* in Table 6.4), which serves as an empirical lower bound. It is

Table 6.4: Quantitative evaluation of conditional molecule generation. Values reported in the table are MAE (over 10K samples) for molecule property predictions (lower is better).

Property	α	$\Delta\varepsilon$	ε_{HOMO}	ε_{LUMO}	μ	C_v
Units	Bohr^2	meV	meV	meV	D	$\frac{\text{cal}}{\text{mol}}$K
QM9*	0.10	64	39	36	0.043	0.040
EDM	2.76	655	356	584	1.111	1.101
EQUIFM	2.41	591	337	530	1.106	1.033
GEOLDM	2.37	587	340	522	1.108	1.025
Ours	1.39	344	182	330	0.300	0.784

Table 6.5: Stability and validity evaluation of D-flow on conditional molecule generation (10K samples).

Property	α	$\Delta\varepsilon$	ε_{HOMO}	ε_{LUMO}	μ	C_v
Molecule Stability (%)	56.2	59.4	60.2	59.4	60.7	57.9
Atom Stability (%)	93.6	93.9	94.1	93.8	94.2	93.6
Validity (%)	77.4	79.4	80.2	79.4	81.1	78.9
Validity & Uniqueness (%)	77.4	79.4	80.2	79.4	81.1	78.9

important to note that for each specific property of conditional generation, the baseline methods utilized a distinct conditional model, each individually trained for generating that particular property while we used a single unconditional model.

Results. Table 6.4 demonstrates that our approach significantly outperforms all other baseline methods in the quality of conditional molecule generation. This superior performance is attributed to our direct optimization of the conditional generation. Table 6.5 presents the stability and validity metrics for our method. In comparison with conditional EDM, which achieves an average molecular stability of 82.1% across different properties, our method reveals a disparity in the stability of the generated molecules. This gap is a consequence of two factors. First, the trained Flow Matching unconditional model achieved inferior performance compared to EDM reaching molecular stability of 72.2%. Second, the optimization with respect to the property predictor does not achieve the same quality of generation as regular sampling. We further verify that the gain in MAE that D-Flow presents is not due to the degradation in the percentage of stable molecules and report both MAE values for stable and non-stable molecules withing the $10k$ generated sample, in Table B.11. The MAE values for both stable and non-stable molecules are on par and improve by a large margin the existing baselines. Figure 6.5 visualize the controlled generation for different polarizability α values; all molecules in the figure are valid and stable with a classifier error lower than 1.

Part III

Discussion and Conclusions

Chapter 7

Discussion and Conclusions

This thesis has introduced several innovative methods addressing fundamental challenges in generative modeling. Our primary focus has been on Continuous Normalizing Flows, enhancing their scalability for high-dimensional data and extending their application beyond traditional generative modeling into fields like optimal transport and controlled generation.

In the first part of this dissertation, we devised simulation-free methods for CNF training, notably through Probability Path Matching and Flow Matching. These simulation-free approaches allow CNFs to scale more effectively and reduce the computational load associated with standard training procedures, making them viable for higher-dimensional data.

In Probability Path Matching (PPM), we built a framework for matching a target density path and the density path generated by a CNF. The PPM is based on minimizing a novel Probability Path Divergence (PPD) that does not require sampling of model densities and, therefore, is easier to train and apply to manifolds. The PPD is shown to upper bound standard divergences and can work with a rather flexible family of target paths on manifolds. Empirically, PPM was shown to facilitate CNF training, scaling for the first time to manifolds of moderate dimension, improving training time, and producing state-of-the-art samplings and log-likelihoods.

At the time of publication, our work achieved state-of-the-art performance on the tasks of generative modeling on manifolds and was the first method to apply to high-dimensional manifolds. The main drawback of the PPD was that the approximation of $\log p_t(x)$ introduced bias, and it required the evaluation of the divergence of the velocity field. In our next work, Flow Matching, we proposed an alternative CNF training approach via simple velocity field regression that also matches probability paths by matching the flow and is unbiased.

Flow Matching (FM) relies on conditional constructions to effortlessly scale to very high dimensions. Furthermore, the FM framework provides an alternative view on diffusion models, and suggests forsaking the stochastic/diffusion construction in favor of more directly specifying the probability path, allowing us to, e.g., construct paths that allow faster sampling and/or improve generation. We experimentally showed the ease of training and sampling when using the Flow Matching framework. Since published, the flow

matching framework has been widely adopted by the research community for large-scale applications such as text-2-image and video generation, as well as generative tasks for molecular and biological data.

In the second part of the thesis, we introduced generalizations and extensions of the flow matching as a generative paradigm. We first proposed Multisample Flow Matching. While most prior works make use of training algorithms where data and noise samples are sampled independently, Multisample Flow Matching allows the use of more complex joint distribution. This introduces a new approach to designing probability paths. Our framework increases sample efficiency and sample quality when using low-cost solvers. Unlike prior works, our training method does not rely on simulation of the learned velocity field during training and does not introduce any min-max formulations. We also note that our method of fitting to batch optimal couplings is the first to preserve the marginal distributions, an important property in both generative modeling and solving transport problems.

Finally, we presented a simple and general framework for controlled generation from pre-trained diffusion/flow models and demonstrated its efficacy on a wide range of problems from various domains and data types ranging from images, and audio to molecules. The main limitation of our approach is in its relatively long runtimes (see Section 6.2.4, and Appendix B.4.1) which stems from the need to back-propagate through multiple compositions of the velocity field (equivalently, the diffusion model). Our theoretical analysis and empirical evidence show however that computing gradients through the ODE solution have a desirable implicit bias, producing state of the art results on common conditional generation tasks. Consequently, an interesting future direction is to utilize the implicit bias but with potentially cheaper computational overhead, and draw connections to other biases used in other controlled generation paradigms.

Bibliography

[AB17] Martin Arjovsky and Leon Bottou. "Towards Principled Methods for Training Generative Adversarial Networks". In: *International Conference on Learning Representations*. 2017 (cit. on p. 10).

[ABV23] Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. "Stochastic interpolants: A unifying framework for flows and diffusions". In: *arXiv preprint arXiv:2303.08797* (2023) (cit. on p. 12).

[ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein generative adversarial networks". In: *International conference on machine learning*. PMLR. 2017, pp. 214–223 (cit. on pp. 5, 10, 54).

[AHK19] Brandon M. Anderson, Truong-Son Hy, and Risi Kondor. "Cormorant: Covariant Molecular Neural Networks." In: *NeurIPS*. 2019, pp. 14510–14519 (cit. on p. 144).

[Amo23] Brandon Amos. "On amortizing convex conjugates for optimal transport". In: *International Conference on Learning Representations (ICLR)* (2023) (cit. on pp. 51, 54).

[And82] Brian DO Anderson. "Reverse-time diffusion equation models". In: *Stochastic Processes and their Applications* 12.3 (1982), pp. 313–326 (cit. on p. 11).

[Arm+21] Mohammadreza Armandpour, Ali Sadeghian, Chunyuan Li, and Mingyuan Zhou. "Partition-guided gans". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 5099–5109 (cit. on p. 41).

[AS66] Syed Mumtaz Ali and Samuel D Silvey. "A general class of coefficients of divergence of one distribution from another". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 28.1 (1966), pp. 131–142 (cit. on p. 6).

[Asa+24] Arip Asadulaev, Alexander Korotin, Vage Egiazarian, Petr Mokrov, and Evgeny Burnaev. "Neural Optimal Transport with General Cost Functionals." In: *ICLR*. 2024 (cit. on p. 54).

[Asi+20] Muhammad Asim, Max Daniels, Oscar Leong, Ali Ahmed, and Paul Hand. "Invertible generative models for inverse problems: mitigating representation error and dataset bias." In: *ICML*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 399–409 (cit. on pp. 61, 68).

[AV23] Michael Samuel Albergo and Eric Vanden-Eijnden. "Building Normalizing Flows with Stochastic Interpolants". In: *The Eleventh International Conference on Learning Representations (ICLR)*. 2023 (cit. on pp. 40, 45–47, 54).

[AWR17] Jason Altschuler, Jonathan Weed, and Philippe Rigollet. "Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration". In: *Advances in Neural Information Processing Systems 30*. 2017 (cit. on p. 53).

[Bar+23] Omer Bar-Tal, Lior Yariv, Yaron Lipman, and Tali Dekel. "MultiDiffusion: Fusing Diffusion Paths for Controlled Image Generation." In: *ICML*. Vol. 202. Proceedings of Machine Learning Research. PMLR, 2023, pp. 1737–1752 (cit. on p. 60).

[BB00] Jean-David Benamou and Yann Brenier. "A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem". In: *Numerische Mathematik* 84.3 (2000), pp. 375–393 (cit. on pp. 5, 48, 110).

[Ben+22] Heli Ben-Hamu*, Samuel Cohen*, Joey Bose, Brandon Amos, Maximillian Nickel, Aditya Grover, Ricky T. Q. Chen, and Yaron Lipman. "Matching Normalizing Flows and Probability Paths on Manifolds". In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 1749–1763 (cit. on pp. ii, 2, 16, 18, 30, 32–34, 39, 95).

[Ben+24] Heli Ben-Hamu, Omri Puny, Itai Gat, Brian Karrer, Uriel Singer, and Yaron Lipman. "D-Flow: Differentiating through Flows for Controlled Generation". In: *Proceedings of the 41st International Conference on Machine Learning*. Vol. 235. Proceedings of Machine Learning Research. PMLR, 21–27 Jul 2024, pp. 3462–3483 (cit. on pp. 3, 60, 95).

[BKc22] Charlotte Bunne, Andreas Krause, and marco cuturi. "Supervised Training of Conditional Monge Maps". In: *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho. 2022 (cit. on p. 51).

[Bog07] Vladimir I Bogachev. *Measure theory*. Vol. 1. Springer Science & Business Media, 2007 (cit. on p. 101).

[Bor+17] Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G. Dimakis. "Compressed sensing using generative models". In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. JMLR.org, 2017, pp. 537–546 (cit. on pp. 61, 68).

[Bos+20] Avishek Joey Bose, Ariella Smofsky, Renjie Liao, Prakash Panangaden, and William L. Hamilton. "Latent variable modelling with hyperbolic normalizing flows". In: *Proceedings of the 37th International Conference on Machine Learning*. ICML'20. JMLR.org, 2020 (cit. on p. 19).

[Bun+21] Charlotte Bunne, Stefan G Stark, Gabriele Gut, Jacobo Sarabia del Castillo, Kjong-Van Lehmann, Lucas Pelkmans, Andreas Krause, and Gunnar Rätsch. "Learning single-cell perturbation responses using neural optimal transport". In: *bioRxiv* (2021) (cit. on p. 51).

[Caf92] Luis Caffarelli. "The regularity of mappings with a convex potential". In: *Journal of the American Mathematical Society* 5 (1992), pp. 99–104 (cit. on p. 107).

[Cas+21] Arantxa Casanova, Marlene Careil, Jakob Verbeek, Michal Drozdzal, and Adriana Romero Soriano. "Instance-conditioned gan". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 27517–27529 (cit. on p. 41).

[Chá22] José A. Chávez. "Generative Flows as a General Purpose Solution for Inverse Problems". In: *CVPR*. 2022 (cit. on p. 68).

[Che+18] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. "Neural ordinary differential equations". In: *Advances in neural information processing systems* 31 (2018) (cit. on pp. 12, 13, 19, 26, 39, 45, 62, 64).

[Che18] Ricky T. Q. Chen. *torchdiffeq*. 2018. URL: https://github.com/rtqichen/torchdiffeq (cit. on pp. 65, 128, 137).

[Cho+21] Jooyoung Choi, Sungwon Kim, Yonghyun Jeong, Youngjune Gwon, and Sungroh Yoon. "ILVR: Conditioning Method for Denoising Diffusion Probabilistic Models". In: *ICCV* (2021) (cit. on p. 68).

[Chu+22] Hyungjin Chung, Byeongsu Sim, Dohoon Ryu, and Jong Chul Ye. "Improving Diffusion Models for Inverse Problems using Manifold Constraints". In: *Advances in Neural Information Processing Systems*. 2022 (cit. on pp. 60, 68, 70).

[Chu+23] Hyungjin Chung, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye. "Diffusion Posterior Sampling for General Noisy Inverse Problems". In: *The Eleventh International Conference on Learning Representations*. 2023 (cit. on p. 68).

[CL23] Ricky T. Q. Chen and Yaron Lipman. "Riemannian flow matching on general geometries". In: *International Conference on Machine Learning* (2023) (cit. on p. 49).

[CLH17] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. "A downsampled variant of imagenet as an alternative to the cifar datasets". In: *arXiv preprint arXiv:1707.08819* (2017) (cit. on pp. 40, 55, 126).

[CMT97] J.A. Cuesta-Albertos, C. Matrán, and A. Tuero-Diaz. "Optimal Transportation Plans and Convergence in Distribution". In: *Journal of Multivariate Analysis* 60.1 (1997), pp. 72–83 (cit. on p. 113).

[Cop+23] Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez. "Simple and Controllable Music Generation". In: *NeurIPS*. 2023 (cit. on p. 71).

[Csi67] Imre Csiszár. "Information-type measures of difference of probability distributions and indirect observation". In: *studia scientiarum Mathematicarum Hungarica* 2 (1967), pp. 229–318 (cit. on p. 6).

[Cut13] Marco Cuturi. "Sinkhorn distances: Lightspeed computation of optimal transport". In: *Advances in neural information processing systems* 26 (2013) (cit. on p. 53).

[DDT19] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. "Augmented Neural ODEs". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019 (cit. on p. 39).

[De +21] Valentin De Bortoli, James Thornton, Jeremy Heng, and Arnaud Doucet. "Diffusion Schrödinger Bridge with Applications to Score-Based Generative Modeling". In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 2021, pp. 17695–17709 (cit. on p. 40).

[Déf+23] Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. "High Fidelity Neural Audio Compression". In: *Transactions on Machine Learning Research* (2023). ISSN: 2835-8856 (cit. on p. 71).

[Den+09a] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848 (cit. on p. 40).

[Den+09b] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255 (cit. on p. 55).

[Det+22] Nicki Skafte Detlefsen, Jiri Borovec, Justus Schock, Ananya Harsh Jha, Teddy Koker, Luca Di Liello, Daniel Stancl, Changsheng Quan, Maxim Grechkin, and William Falcon. "TorchMetrics - Measuring Reproducibility in PyTorch". In: *Journal of Open Source Software* 7.70 (2022), p. 4101. DOI: 10.21105/joss.04101 (cit. on p. 142).

[DGK21] Ann-Kathrin Dombrowski, Jan E Gerken, and Pan Kessel. "Diffeomorphic Explanations with Normalizing Flows". In: *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*. 2021 (cit. on pp. 61, 68).

[DN21] Prafulla Dhariwal and Alexander Quinn Nichol. "Diffusion Models Beat GANs on Image Synthesis". In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan. 2021 (cit. on pp. 39–41, 60, 68, 126, 128, 136).

[Dom+24] Ann-Kathrin Dombrowski, Jan E. Gerken, Klaus-Robert Müller, and Pan Kessel. "Diffeomorphic Counterfactuals With Generative Models". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46.5 (2024), pp. 3257–3274. DOI: 10.1109/TPAMI.2023.3339980 (cit. on p. 61).

[DP80] John R Dormand and Peter J Prince. "A family of embedded Runge-Kutta formulae". In: *Journal of computational and applied mathematics* 6.1 (1980), pp. 19–26 (cit. on pp. 27, 40).

[DSB17] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using Real NVP". In: *International Conference on Learning Representations*. 2017 (cit. on pp. 8, 137).

[Du+22] Shian Du, Yihong Luo, Wei Chen, Jian Xu, and Delu Zeng. "TO-FLOW: Efficient Continuous Normalizing Flows with Temporal Optimization adjoint with Moving Speed". In: *CVPR* (2022) (cit. on p. 39).

[Eva05] Lawrence C Evans. "An introduction to mathematical optimal control theory". In: *Lecture Notes, University of California, Department of Mathematics, Berkeley* 3 (2005), pp. 15–40 (cit. on p. 116).

[Eva97] Lawrence C Evans. "Partial differential equations and Monge-Kantorovich mass transfer". In: *Current developments in mathematics* 1997.1 (1997), pp. 65–126 (cit. on p. 26).

[Fan+22] Jiaojiao Fan, Shu Liu, Shaojun Ma, Yongxin Chen, and Hao-Min Zhou. "Scalable Computation of Monge Maps with General Costs". In: *ICLR Workshop on Deep Generative Models for Highly Structured Data*. 2022 (cit. on p. 54).

[Fat+20] Kilian Fatras, Younes Zine, Rémi Flamary, Rémi Gribonval, and Nicolas Courty. "Learning with minibatch Wasserstein: asymptotic and gradient properties". In: *AISTATS* (2020) (cit. on pp. 54, 59).

[Fat+21] Kilian Fatras, Younes Zine, Szymon Majewski, Rémi Flamary, Rémi Gribonval, and Nicolas Courty. "Minibatch optimal transport distances; analysis and applications". In: *arXiv preprint arXiv:2101.01792* (2021) (cit. on p. 54).

[Fer+14] Sira Ferradans, Nicolas Papadakis, Gabriel Peyré, and Jean-François Aujol. "Regularized discrete optimal transport". In: *SIAM Journal on Imaging Sciences* 7.3 (2014), pp. 1853–1882 (cit. on pp. 54, 59).

[Fey+17] Jean Feydy, Benjamin Charlier, François-Xavier Vialard, and Gabriel Peyré. "Optimal transport for diffeomorphic registration". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2017, pp. 291–299 (cit. on p. 51).

[FF20] Luca Falorsi and Patrick Forré. "Neural Ordinary Differential Equations on Manifolds". In: *Second workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models (ICML), Virtual Conference* (2020) (cit. on pp. 19, 26, 29).

[Fin+20a] Chris Finlay, Augusto Gerolin, Adam M Oberman, and Aram-Alexandre Pooladian. "Learning normalizing flows from Entropy-Kantorovich potentials". In: *arXiv preprint arXiv:2006.06033* (2020) (cit. on p. 54).

[Fin+20b] Chris Finlay, Jörn-Henrik Jacobsen, Levon Nurbekyan, and Adam Oberman. "How to train your neural ODE: the world of Jacobian and kinetic regularization". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 3154–3164 (cit. on pp. 15, 16, 39, 46, 48, 54).

[Fla+21] Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z. Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, Léo Gautheron, Nathalie T.H. Gayraud, Hicham Janati, Alain Rakotomamonjy, Ievgen Redko, Antoine Rolet, Antony Schutz, Vivien Seguy, Danica J. Sutherland, Romain Tavenard, Alexander Tong, and Titouan Vayer. "POT: Python Optimal Transport". In: *Journal of Machine Learning Research* 22.78 (2021), pp. 1–8 (cit. on p. 52).

[Gaf+22] Oran Gafni, Adam Polyak, Oron Ashual, Shelly Sheynin, Devi Parikh, and Yaniv Taigman. "Make-A-Scene: Scene-Based Text-to-Image Generation with Human Priors". In: *Computer Vision – ECCV 2022: 17th European Conference*. Springer-Verlag, 2022, pp. 89–106. ISBN: 978-3-031-19783-3 (cit. on p. 45).

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016 (cit. on p. 4).

[GEB15] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. "A neural algorithm of artistic style". In: *arXiv preprint arXiv:1508.06576* (2015) (cit. on p. 57).

[Goo+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Vol. 27. 2014 (cit. on p. 10).

[GPC18] Aude Genevay, Gabriel Peyré, and Marco Cuturi. "Learning generative models with Sinkhorn divergences". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2018, pp. 1608–1617 (cit. on p. 54).

[Gra+19] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, and David Duvenaud. "Scalable Reversible Generative Models with Free-form Continuous Dynamics". In: *International Conference on Learning Representations (ICLR)*. 2019 (cit. on pp. 14–16, 29, 32, 39, 64).

[Gra+22] Alexandros Graikos, Nikolay Malkin, Nebojsa Jojic, and Dimitris Samaras. "Diffusion models as plug-and-play priors". In: *Proceedings of the 36th International Conference on Neural Information Processing Systems*. 2022 (cit. on pp. 60, 68).

[GS62] David Gale and Lloyd S Shapley. "College admissions and the stability of marriage". In: *The American Mathematical Monthly* 69.1 (1962), pp. 9–15 (cit. on p. 53).

[Gul+17] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. "Improved Training of Wasserstein GANs". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017 (cit. on p. 10).

[Hai+19] Niv Haim, Nimrod Segol, Heli Ben-Hamu, Haggai Maron, and Yaron Lipman. "Surface networks via general covers". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 632–641 (cit. on p. 96).

[HD05] Aapo Hyvärinen and Peter Dayan. "Estimation of non-normalized statistical models by score matching." In: *Journal of Machine Learning Research* 6.4 (2005) (cit. on pp. 12, 27).

[Heu+17] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. "Gans trained by a two time-scale update rule converge to a local nash equilibrium". In: *Advances in neural information processing systems* 30 (2017) (cit. on pp. 41, 70).

[HJA20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6840–6851 (cit. on pp. 11, 18, 20, 27, 32, 37, 39, 41, 45, 48, 60, 126, 127, 139).

[HLC21] Chin-Wei Huang, Jae Hyun Lim, and Aaron Courville. "A Variational Perspective on Diffusion-Based Generative Models and Score Matching". In: *Advances in Neural Information Processing Systems*. 2021 (cit. on p. 27).

[Ho+22] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. *Video Diffusion Models*. 2022. arXiv: 2204.03458 [cs.CV] (cit. on p. 68).

[Hoa+18] Quan Hoang, Tu Dinh Nguyen, Trung Le, and Dinh Phung. "MGAN: Training generative adversarial nets with multiple generators". In: *International conference on learning representations (ICLR)*. 2018 (cit. on p. 41).

[Hoo+22] Emiel Hoogeboom, Víctor Garcia Satorras, Clément Vignac, and Max Welling. "Equivariant Diffusion for Molecule Generation in 3D". In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 8867–8887 (cit. on p. 72).

[HS21] Jonathan Ho and Tim Salimans. "Classifier-Free Diffusion Guidance". In: *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*. 2021 (cit. on pp. 60, 68).

[Hut90] M.F. Hutchinson. "A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines". In: *Communications in Statistics - Simulation and Computation* 19.2 (1990), pp. 433–450 (cit. on p. 14).

[JAF16] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. "Perceptual losses for real-time style transfer and super-resolution". In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*. Springer. 2016, pp. 694–711 (cit. on p. 57).

[Kan+23] Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, and Taesung Park. "Scaling up GANs for Text-to-Image Synthesis". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023 (cit. on p. 10).

[Kan42] L. Kantorovitch. "On the translocation of masses". In: *C. R. (Doklady) Acad. Sci. URSS (N.S.)* 37 (1942), pp. 199–201 (cit. on p. 48).

[Kar+20] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. "Analyzing and Improving the Image Quality of StyleGAN". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 8107–8116. DOI: 10.1109/CVPR42600.2020.00813 (cit. on p. 10).

[Kar+21] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. "Alias-Free Generative Adversarial Networks". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan. Vol. 34. Curran Associates, Inc., 2021, pp. 852–863 (cit. on p. 10).

[Kar+22] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. "Elucidating the design space of diffusion-based generative models". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 26565–26577 (cit. on p. 66).

[Kaw+22] Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. "Denoising Diffusion Restoration Models". In: *Advances in Neural Information Processing Systems*. 2022 (cit. on pp. 60, 68, 70).

[KD18] Durk P Kingma and Prafulla Dhariwal. "Glow: Generative flow with invertible 1x1 convolutions". In: *Advances in neural information processing systems* 31 (2018) (cit. on p. 8).

[Kel+20] Jacob Kelly, Jesse Bettencourt, Matthew J Johnson, and David K Duvenaud. "Learning differential equations that are easy to solve". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 4370–4380 (cit. on p. 39).

[KH+09] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images". In: (2009) (cit. on p. 40).

[Kil+18] Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi. "Fréchet Audio Distance: A Metric for Evaluating Music Enhancement Algorithms". In: *arXiv preprint arXiv:1812.08466* (2018) (cit. on p. 71).

[Kin+21] Diederik P Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. "Variational Diffusion Models". In: *Advances in Neural Information Processing Systems*. 2021 (cit. on pp. 27, 39, 126).

[KLA19] Tero Karras, Samuli Laine, and Timo Aila. "A Style-Based Generator Architecture for Generative Adversarial Networks". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 4396–4405. DOI: 10.1109/CVPR.2019.00453 (cit. on p. 10).

[KPB20] Ivan Kobyzev, Simon JD Prince, and Marcus A Brubaker. "Normalizing flows: An introduction and review of current methods". In: *IEEE transactions on pattern analysis and machine intelligence* 43.11 (2020), pp. 3964–3979 (cit. on p. 39).

[KPS12] Peter Eris Kloeden, Eckhard Platen, and Henri Schurz. *Numerical solution of SDE through computer experiments*. Springer Science & Business Media, 2012 (cit. on pp. 27, 42).

[KR58] Leonid Vasilevich Kantorovich and SG Rubinshtein. "On a space of totally additive functions". In: *Vestnik of the St. Petersburg University: Mathematics* 13.7 (1958), pp. 52–59 (cit. on p. 5).

[KW14] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes." In: *ICLR*. 2014 (cit. on p. 9).

[Lan16] Greg Landrum. "RDKit: Open-Source Cheminformatics Software". In: (2016) (cit. on p. 72).

[LGL23] Xingchao Liu, Chengyue Gong, and Qiang Liu. "Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow". In: *The Eleventh International Conference on Learning Representations (ICLR)*. 2023 (cit. on pp. 40, 45–47, 54, 113, 137, 139).

[LGS19] Huidong Liu, Xianfeng Gu, and Dimitris Samaras. "Wasserstein GAN with quadratic transport cost". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 4832–4841 (cit. on p. 54).

[Li+17] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. "MMD GAN: Towards deeper understanding of moment matching network". In: *Advances in neural information processing systems* 30 (2017) (cit. on p. 54).

[Lin+15] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV] (cit. on p. 71).

[Lin+18] Zinan Lin, Ashish Khetan, Giulia Fanti, and Sewoong Oh. "Pacgan: The power of two samples in generative adversarial networks". In: *Advances in neural information processing systems* 31 (2018) (cit. on p. 41).

[Lip+23] 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 (ICLR)*. 2023 (cit. on pp. 2, 16, 32, 45, 60, 62, 67, 72, 95, 115, 139).

[Liu+23a] Guan-Horng Liu, Arash Vahdat, De-An Huang, Evangelos A Theodorou, Weili Nie, and Anima Anandkumar. "I^2SB: Image-to-Image Schr\" odinger Bridge". In: *International Conference on Machine Learning* (2023) (cit. on p. 51).

[Liu+23b] Xingchao Liu, Lemeng Wu, Shujian Zhang, Chengyue Gong, Wei Ping, and Qiang Liu. "FlowGrad: Controlling the Output of Generative ODEs With Gradients". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2023, pp. 24335–24344 (cit. on p. 68).

[Liu22] Qiang Liu. *Rectified Flow: A Marginal Preserving Approach to Optimal Transport*. 2022 (cit. on p. 50).

[LKY23] Sangyun Lee, Beomsu Kim, and Jong Chul Ye. "Minimizing Trajectory Curvature of ODE-based Generative Models". In: *arXiv preprint arXiv:2301.12003* (2023) (cit. on p. 55).

[Lou+20] Aaron Lou, Derek Lim, Isay Katsman, Leo Huang, Qingxuan Jiang, Ser-Nam Lim, and Christopher De Sa. "Neural Manifold Ordinary Differential Equations". In: (2020) (cit. on pp. 19, 26, 29).

[Lüb+22] Frederike Lübeck, Charlotte Bunne, Gabriele Gut, Jacobo Sarabia del Castillo, Lucas Pelkmans, and David Alvarez-Melis. "Neural unbalanced optimal transport via cycle-consistent semi-couplings". In: *arXiv preprint arXiv:2209.15621* (2022) (cit. on p. 51).

[Luč+19] Mario Lučić, Michael Tschannen, Marvin Ritter, Xiaohua Zhai, Olivier Bachem, and Sylvain Gelly. "High-fidelity image generation with fewer labels". In: *International conference on machine learning*. PMLR. 2019, pp. 4183–4192 (cit. on p. 41).

[Lug+22] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. *RePaint: Inpainting using Denoising Diffusion Probabilistic Models*. 2022. arXiv: 2201.09865 [cs.CV] (cit. on p. 68).

[Mag54] Wilhelm Magnus. "On the exponential solution of differential equations for a linear operator". In: *Communications on Pure and Applied Mathematics* 7.4 (1954), pp. 649–673. DOI: https://doi.org/10.1002/cpa.3160070404. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpa.3160070404 (cit. on p. 117).

[Mak+20] Ashok Makkuva, Amirhossein Taghvaei, Sewoong Oh, and Jason Lee. "Optimal transport mapping via input convex neural networks". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 6672–6681 (cit. on pp. 46, 51, 54).

[Mar+08] Kanti V. Mardia, Gareth Hughes, Charles C. Taylor, and Harshinder Singh. "A Multivariate Von Mises Distribution with Applications to Bioinformatics". In: *The Canadian Journal of Statistics / La Revue Canadienne de Statistique* 36.1 (2008), pp. 99–109 (cit. on p. 19).

[Mar+19a] Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. "Invariant and Equivariant Graph Networks". In: *International Conference on Learning Representations (ICLR)*. 2019 (cit. on p. 95).

[Mar+19b] Haggai Maron*, Heli Ben-Hamu*, Hadar Serviansky*, and Yaron Lipman. "Provably Powerful Graph Networks". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 2156–2167 (cit. on p. 95).

[Mar+23] Morteza Mardani, Jiaming Song, Jan Kautz, and Arash Vahdat. *A Variational Perspective on Solving Inverse Problems with Diffusion Models*. 2023. arXiv: 2305.04391 [cs.LG] (cit. on pp. 60, 68–71, 145).

[Mar14] Kantilal Varichand Mardia. *Statistics of directional data*. Academic press, 2014 (cit. on p. 25).

[McC97] Robert J McCann. "A convexity principle for interacting gases". In: *Advances in mathematics* 128.1 (1997), pp. 153–179 (cit. on pp. 33, 38, 47).

[MN20] Emile Mathieu and Maximilian Nickel. "Riemannian continuous normalizing flows". In: *arXiv preprint arXiv:2006.10605* (2020) (cit. on pp. 19, 26, 28, 29).

[MRO20a] Dimitra Maoutsa, Sebastian Reich, and Manfred Opper. "Interacting particle solutions of Fokker–Planck equations through gradient–log–density estimation". In: *Entropy* 22.8 (2020), p. 802 (cit. on p. 40).

[MRO20b] Dimitra Maoutsa, Sebastian Reich, and Manfred Opper. "Interacting Particle Solutions of Fokker–Planck Equations Through Gradient–Log–Density Estimation". In: *Entropy* 22.8 (July 2020), p. 802. DOI: 10.3390/e22080802 (cit. on p. 124).

[Mül97] Alfred Müller. "Integral Probability Metrics and Their Generating Classes of Functions". In: *Advances in Applied Probability* 29.2 (1997), pp. 429–443. ISSN: 00018678 (cit. on p. 5).

[Nal+19] Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. *Do Deep Generative Models Know What They Don't Know?* 2019. arXiv: 1810.09136 [stat.ML] (cit. on p. 64).

[ND21] Alexander Quinn Nichol and Prafulla Dhariwal. "Improved denoising diffusion probabilistic models". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8162–8171 (cit. on p. 39).

[Nea92] Radford M Neal. "Connectionist learning of belief networks". In: *Artificial intelligence* 56.1 (1992), pp. 71–113 (cit. on p. 7).

[Ngu+22] Khai Nguyen, Dang Nguyen, Tung Pham, Nhat Ho, et al. "Improving mini-batch optimal transport via partial transportation". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 16656–16690 (cit. on p. 54).

[Nic+22] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. *GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models*. 2022. arXiv: 2112.10741 [cs.CV] (cit. on p. 71).

[NIO20] Levon Nurbekyan, Alexander Iannantuono, and Adam M. Oberman. "No-Collision Transportation Maps". In: *Journal of Scientific Computing* 82.2 (2020) (cit. on p. 108).

[NSM22] Kirill Neklyudov, Daniel Severo, and Alireza Makhzani. "Action Matching: A Variational Method for Learning Stochastic Dynamics from Samples". In: *arXiv preprint arXiv:2210.06662* (2022) (cit. on p. 45).

[NSM23] Kirill Neklyudov, Daniel Severo, and Alireza Makhzani. *Action Matching: A Variational Method for Learning Stochastic Dynamics from Samples*. 2023 (cit. on p. 40).

[OAP19] Changyong Oh, Kamil Adamczewski, and Mijung Park. "Radial and directional posteriors for bayesian neural networks". In: *arXiv preprint arXiv:1902.02603* (2019) (cit. on p. 122).

[Onk+21a] Derek Onken, Samy Wu Fung, Xingjian Li, and Lars Ruthotto. "Ot-flow: Fast and accurate continuous normalizing flows via optimal transport". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 10. 2021, pp. 9223–9232 (cit. on pp. 39, 54).

[Onk+21b] Derek Onken, Samy Wu Fung, Xingjian Li, and Lars Ruthotto. "OT-Flow: Fast and Accurate Continuous Normalizing Flows via Optimal Transport". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.10 (2021), pp. 9223–9232 (cit. on p. 15).

[Pap+21] George Papamakarios, Eric T Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. "Normalizing Flows for Probabilistic Modeling and Inference." In: *J. Mach. Learn. Res.* 22.57 (2021), pp. 1–64 (cit. on p. 39).

[PBL20] Omri Puny, Heli Ben-Hamu, and Yaron Lipman. "Global Attention Improves Graph Networks Generalization". In: *Workshop on: "Graph Representation Learning and Beyond"*. 2020 (cit. on p. 96).

[PC19] Gabriel Peyré and Marco Cuturi. "Computational optimal transport". In: *Foundations and Trends® in Machine Learning* 11.5-6 (2019), pp. 355–607 (cit. on pp. 52, 53).

[Pea88] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier, 1988 (cit. on p. 7).

[Pel21] Stefano Peluchetti. "Non-Denoising Forward-Time Diffusions". In: (2021) (cit. on pp. 40, 103).

[Pen06] Xavier Pennec. "Intrinsic statistics on Riemannian manifolds: Basic tools for geometric measurements". In: *Journal of Mathematical Imaging and Vision* 25.1 (2006), p. 127 (cit. on p. 27).

[PN21] Aram-Alexandre Pooladian and Jonathan Niles-Weed. "Entropic estimation of optimal transport maps". In: *arXiv preprint arXiv:2109.12004* (2021) (cit. on p. 54).

[Pok+23] Ashwini Pokle, Matthew J. Muckley, Ricky T. Q. Chen, and Brian Karrer. *Training-free Linear Image Inversion via Flows*. 2023. arXiv: 2310.04432 [cs.CV] (cit. on pp. 60, 68–70, 142, 145).

[Poo+23] 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". In: *International Conference on Machine Learning*. 2023 (cit. on pp. ii, 3, 45, 95).

[Pun+21] Omri Puny*, Matan Atzmon*, Heli Ben-Hamu*, Edward J Smith, Ishan Misra, Aditya Grover, and Yaron Lipman. "Frame Averaging for Invariant and Equivariant Network Design". In: *Tenth International Conference on Learning Representations (ICLR)* (2021) (cit. on p. 96).

[Rad+18] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. "Improving language understanding with unsupervised learning". In: (2018) (cit. on p. 7).

[Raf+23] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. 2023. arXiv: 1910.10683 [cs.LG] (cit. on p. 71).

[Ram+14] Raghunathan Ramakrishnan, Pavlo Dral, Matthias Rupp, and Anatole von Lilienfeld. "Quantum chemistry structures and properties of 134 kilo molecules". In: *Scientific Data* 1 (Aug. 2014). DOI: 10.1038/sdata.2014.22 (cit. on pp. 72, 144).

[Ram+22] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. "Hierarchical text-conditional image generation with clip latents". In: *arXiv preprint arXiv:2204.06125* (2022) (cit. on pp. 32, 45, 70).

[Rén61] Alfréd Rényi. "On measures of entropy and information". In: *Proceedings of the fourth Berkeley symposium on mathematical statistics and probability, volume 1: contributions to the theory of statistics*. Vol. 4. University of California Press. 1961, pp. 547–562 (cit. on p. 6).

[Rez+20] Danilo Jimenez Rezende, George Papamakarios, Sébastien Racaniere, Michael Albergo, Gurtej Kanwar, Phiala Shanahan, and Kyle Cranmer. "Normalizing flows on tori and spheres". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 8083–8092 (cit. on p. 19).

[RM15] Danilo Rezende and Shakir Mohamed. "Variational inference with normalizing flows". In: *International Conference on Machine Learning*. PMLR. 2015, pp. 1530–1538 (cit. on pp. 8, 19).

[Rom+22a] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2022. arXiv: 2112.10752 [cs.CV] (cit. on p. 71).

[Rom+22b] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. "High-resolution image synthesis with latent diffusion models". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 10684–10695 (cit. on p. 32).

[Rou+23] Litu Rout, Negin Raoof, Giannis Daras, Constantine Caramanis, Alexandros G. Dimakis, and Sanjay Shakkottai. *Solving Linear Inverse Problems Provably via Posterior Sampling with Latent Diffusion Models*. 2023. arXiv: 2307.00619 [cs.LG] (cit. on p. 68).

[Roz+21] Noam Rozen, Aditya Grover, Maximilian Nickel, and Yaron Lipman. "Moser Flow: Divergence-based Generative Modeling on Manifolds". In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan. 2021 (cit. on pp. 16, 26, 28, 32, 39, 121).

[RS16] Diego Ruiz-Antolín and Javier Segura. "A new type of sharp bounds for ratios of modified Bessel functions". In: *Journal of Mathematical Analysis and Applications* 443.2 (2016), pp. 1232–1246 (cit. on p. 122).

[Sag+18] Alexander Sage, Eirikur Agustsson, Radu Timofte, and Luc Van Gool. "Logo synthesis and manipulation with clustered generative adversarial networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 5879–5888 (cit. on p. 41).

[Sah+22a] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding". In: *arXiv preprint arXiv:2205.11487* (2022) (cit. on p. 45).

[Sah+22b] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. "Image super-resolution via iterative refinement". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022) (cit. on p. 43).

[Sal+16] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. "Improved Techniques for Training GANs". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016 (cit. on p. 10).

[Sam+23a] Dvir Samuel, Rami Ben-Ari, Nir Darshan, Haggai Maron, and Gal Chechik. "Norm-guided latent space exploration for text-to-image generation". In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023 (cit. on p. 64).

[Sam+23b] Dvir Samuel, Rami Ben-Ari, Simon Raviv, Nir Darshan, and Gal Chechik. *Generating images of rare concepts using pre-trained diffusion models*. 2023. arXiv: 2304.14530 [cs.CV] (cit. on pp. 60, 61, 69).

[San15] Filippo Santambrogio. "Optimal transport for applied mathematicians". In: (2015) (cit. on p. 48).

[Sat+22] Victor Garcia Satorras, Emiel Hoogeboom, Fabian B. Fuchs, Ingmar Posner, and Max Welling. *E(n) Equivariant Normalizing Flows*. 2022. arXiv: 2105.09016 [cs.LG] (cit. on p. 72).

[Sch+19] Geoffrey Schiebinger, Jian Shu, Marcin Tabaka, Brian Cleary, Vidya Subramanian, Aryeh Solomon, Joshua Gould, Siyan Liu, Stacie Lin, Peter Berube, et al. "Optimal-transport analysis of single-cell gene expression identifies developmental trajectories in reprogramming". In: *Cell* 176.4 (2019), pp. 928–943 (cit. on p. 51).

[SE19] Yang Song and Stefano Ermon. "Generative Modeling by Estimating Gradients of the Data Distribution". In: *Advances in Neural Information Processing Systems*. 2019, pp. 11895–11907 (cit. on pp. 10–12, 18, 20, 27, 39, 60, 126, 127).

[Seg+17] Vivien Seguy, Bharath Bhushan Damodaran, Rémi Flamary, Nicolas Courty, Antoine Rolet, and Mathieu Blondel. "Large-scale optimal transport and mapping estimation". In: *arXiv preprint arXiv:1711.02283* (2017) (cit. on pp. 54, 59).

[Sha+23] Neta Shaul, Ricky TQ Chen, Maximilian Nickel, Matthew Le, and Yaron Lipman. "On kinetic optimal probability paths for generative models". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 30883–30907 (cit. on p. 66).

[SME21] Jiaming Song, Chenlin Meng, and Stefano Ermon. "Denoising Diffusion Implicit Models". In: *International Conference on Learning Representations*. 2021 (cit. on pp. 32, 42).

[SN20] J. J. Sakurai and Jim Napolitano. *Modern Quantum Mechanics*. 3rd ed. Cambridge University Press, 2020 (cit. on pp. 116, 117).

[Soh+15] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. "Deep unsupervised learning using nonequilibrium thermodynamics". In: *International Conference on Machine Learning*. PMLR. 2015, pp. 2256–2265 (cit. on pp. 10, 27, 37, 39).

[Sol+15] Justin Solomon, Fernando De Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas Guibas. "Convolutional Wasserstein distances: Efficient optimal transportation on geometric domains". In: *ACM Transactions on Graphics (TOG)* 34.4 (2015), p. 66 (cit. on p. 51).

[Sol+16] Justin Solomon, Gabriel Peyré, Vladimir G. Kim, and Suvrit Sra. "Entropic metric alignment for correspondence problems". In: *ACM Trans. Graph.* 35.4 (2016), 72:1–72:13 (cit. on p. 51).

[Son+21a] Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. "Maximum Likelihood Training of Score-Based Diffusion Models". In: *Thirty-Fifth Conference on Neural Information Processing Systems*. 2021 (cit. on pp. 27, 32, 39–41, 126, 127, 139).

[Son+21b] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. "Score-Based Generative Modeling through Stochastic Differential Equations". In: *International Conference on Learning Representations (ICLR)*. 2021 (cit. on pp. 10, 11, 20, 27, 32, 37, 39–41, 45, 48, 60, 62, 68, 124, 127, 139).

[Son+23a] Jiaming Song, Arash Vahdat, Morteza Mardani, and Jan Kautz. "Pseudoinverse-Guided Diffusion Models for Inverse Problems". In: *International Conference on Learning Representations (ICLR)*. 2023 (cit. on pp. 60, 68–70, 145).

[Son+23b] Yuxuan Song, Jingjing Gong, Minkai Xu, Ziyao Cao, Yanyan Lan, Stefano Ermon, Hao Zhou, and Wei-Ying Ma. "Equivariant Flow Matching with Hybrid Probability Transport for 3D Molecule Generation". In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023 (cit. on p. 72).

[Sri+12] Bharath K. Sriperumbudur, Kenji Fukumizu, Arthur Gretton, Bernhard Schölkopf, and Gert R. G. Lanckriet. "On the empirical estimation of integral probability metrics". In: *Electronic Journal of Statistics* 6.none (2012), pp. 1550–1599. DOI: 10.1214/12-EJS722 (cit. on p. 5).

[Tas+20] Yuval Tassa, Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, and Nicolas Heess. *dm_control: Software and Tasks for Continuous Control*. 2020. arXiv: 2006.12983 [cs.RO] (cit. on p. 30).

[TB15] Lucas Theis and Matthias Bethge. "Generative Image Modeling Using Spatial LSTMs." In: *NIPS*. 2015, pp. 1927–1935 (cit. on p. 7).

[Ton+20] 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*. PMLR. 2020, pp. 9526–9536 (cit. on pp. 39, 48, 54).

[Ton+24] 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". In: *Transactions on Machine Learning Research* (2024). ISSN: 2835-8856 (cit. on p. 54).

[TP01] Vladimir Tulovsky and Lech Papiez. "Formula for the fundamental solution of the heat equation on the sphere". In: *Applied mathematics letters* 14.7 (2001), pp. 881–884 (cit. on p. 27).

[TT13] Esteban G. Tabak and Cristina Vilma Turner. "A Family of Nonparametric Density Estimation Algorithms". In: *Communications on Pure and Applied Mathematics* 66 (2013) (cit. on p. 8).

[TV10] Esteban G. Tabak and Eric Vanden-Eijnden. "Density estimation by dual ascent of the log-likelihood". In: *Communications in Mathematical Sciences* 8.1 (2010), pp. 217–233 (cit. on p. 8).

[Var58] V. S. Varadarajan. "On the Convergence of Sample Probability Distributions". In: *Sankhyā: The Indian Journal of Statistics (1933-1960)* 19.1/2 (1958), pp. 23–26 (cit. on p. 112).

[Vil03] C. Villani. *Topics in Optimal Transportation*. Graduate studies in mathematics. American Mathematical Society, 2003 (cit. on pp. 48, 108).

[Vil08] C. Villani. *Optimal Transport: Old and New*. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 2008 (cit. on pp. 5, 13, 22, 48, 98, 107, 112, 136).

[Vin11] Pascal Vincent. "A connection between score matching and denoising autoencoders". In: *Neural computation* 23.7 (2011), pp. 1661–1674 (cit. on pp. 27, 32, 39).

[VKK16] Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. "Pixel recurrent neural networks". In: *International conference on machine learning*. PMLR. 2016, pp. 1747–1756 (cit. on p. 7).

[VKK21] Arash Vahdat, Karsten Kreis, and Jan Kautz. "Score-based Generative Modeling in Latent Space". In: *arXiv preprint arXiv:2106.05931* (2021) (cit. on p. 27).

[Wal+23] Bram Wallace, Akash Gokul, Stefano Ermon, and Nikhil Naik. *End-to-End Diffusion Latent Optimization Improves Classifier Guidance*. 2023. arXiv: 2303.13703 [cs.CV] (cit. on pp. 60, 61, 69).

[Wan+21] Gefei Wang, Yuling Jiao, Qian Xu, Yang Wang, and Can Yang. "Deep Generative Learning via Schrödinger Bridge". In: arXiv:2106.10410 (July 2021). arXiv:2106.10410 [cs]. DOI: 10.48550/arXiv.2106.10410 (cit. on p. 40).

[WLD21] Jay Whang, Qi Lei, and Alexandros G. Dimakis. *Solving Inverse Problems with a Flow-based Noise Model*. 2021. arXiv: 2003.08089 [cs.LG] (cit. on p. 61).

[Wol20] Gershon Wolansky. "Semi-discrete optimal transport". In: *arXiv preprint arXiv:1911.04348* (Sept. 2020) (cit. on pp. 135, 136).

[WYZ23] Yinhuai Wang, Jiwen Yu, and Jian Zhang. "Zero-Shot Image Restoration Using Denoising Diffusion Null-Space Model". In: *The Eleventh International Conference on Learning Representations* (2023) (cit. on p. 68).

[Xu+23] Minkai Xu, Alexander S Powers, Ron O. Dror, Stefano Ermon, and Jure Leskovec. "Geometric Latent Diffusion Models for 3D Molecule Generation". In: *Proceedings of the 40th International Conference on Machine Learning*. Vol. 202. Proceedings of Machine Learning Research. PMLR, 2023, pp. 38592–38610 (cit. on p. 72).

[YK19] Liu Yang and George E. Karniadakis. "Potential Flow Generator with L_2 Optimal Transport Regularity for Generative Models". In: *CoRR* (2019). arXiv: 1908.11462 (cit. on p. 39).

[Yu+23] Jiwen Yu, Yinhuai Wang, Chen Zhao, Bernard Ghanem, and Jian Zhang. "FreeDoM: Training-Free Energy-Guided Conditional Diffusion Model". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (2023) (cit. on p. 60).

[ZC22] Qinsheng Zhang and Yongxin Chen. "Fast Sampling of Diffusion Models with Exponential Integrator". In: *arXiv preprint arXiv:2204.13902* (2022) (cit. on pp. 32, 42).

[Zha+18] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric". In: *CVPR* (2018) (cit. on p. 70).

[Ziv+24] Alon Ziv, Itai Gat, Gael Le Lan, Tal Remez, Felix Kreuk, Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. "Masked Audio Generation using a Single Non-Autoregressive Transformer". In: (2024) (cit. on p. 71).

[Zol84] V. M. Zolotarev. "Probability Metrics". In: *Theory of Probability & Its Applications* 28.2 (1984), pp. 278–302. DOI: 10.1137/1128025 (cit. on p. 5).

Summary of Publications

Throughout my PhD, I was fortunate to research and contribute to two central fields of modern machine learning: *Deep Generative Modeling* and *Geometric Deep Learning*. This thesis elaborates on my contributions to the field of *deep generative modeling* and is distributed as described below.

The content of Chapter 3 is based on:

> [Ben+22] Heli Ben-Hamu*, Samuel Cohen*, Joey Bose, Brandon Amos, Maximillian Nickel, Aditya Grover, Ricky T. Q. Chen, and Yaron Lipman. "Matching Normalizing Flows and Probability Paths on Manifolds". In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 1749–1763

The content of Chapter 4 is based on:

> [Lip+23] 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 (ICLR)*. 2023

The content of Chapter 5 is based on:

> [Poo+23] 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". In: *International Conference on Machine Learning*. 2023

The content of Chapter 6 is based on:

> [Ben+24] Heli Ben-Hamu, Omri Puny, Itai Gat, Brian Karrer, Uriel Singer, and Yaron Lipman. "D-Flow: Differentiating through Flows for Controlled Generation". In: *Proceedings of the 41st International Conference on Machine Learning*. Vol. 235. Proceedings of Machine Learning Research. PMLR, 21–27 Jul 2024, pp. 3462–3483

Non-Thesis Publications: I have also contributed to the following research publications in the field of *geometric deep learning*, which are not included in this thesis.

> Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. "Invariant and Equivariant Graph Networks". In: *International Conference on Learning Representations (ICLR)*. 2019
>
> Haggai Maron*, Heli Ben-Hamu*, Hadar Serviansky*, and Yaron Lipman. "Provably Powerful Graph Networks". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 2156–2167

Niv Haim, Nimrod Segol, Heli Ben-Hamu, Haggai Maron, and Yaron Lipman. "Surface networks via general covers". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 632–641

Omri Puny, Heli Ben-Hamu, and Yaron Lipman. "Global Attention Improves Graph Networks Generalization". In: *Workshop on: "Graph Representation Learning and Beyond"*. 2020

Omri Puny*, Matan Atzmon*, Heli Ben-Hamu*, Edward J Smith, Ishan Misra, Aditya Grover, and Yaron Lipman. "Frame Averaging for Invariant and Equivariant Network Design". In: *Tenth International Conference on Learning Representations (ICLR)* (2021)

(*) denotes co-first author.

Appendices

Appendix A

Proofs

A.1 Proofs of Chapter 3

A.1.1 Proof of Theorem 1

Since $v \in \mathfrak{X}(\mathcal{M})$, it is locally Lipschitz. Since v is bounded it satisfies in particular

$$\int_0^1 \int_{\mathcal{M}} |v(t,x)|\, p_t(x)\, dV_x dt < +\infty.$$

Therefore, according to the Mass Conservation Formula Theorem (see e.g., [Vil08]) equation 3.3 holds iff

$$\partial_t p_t + \operatorname{div}(p_t v) = 0, \tag{A.1}$$

where div denotes the divergence operator over the manifold $\mathcal{M}$. We assumed $p_t > 0$ and therefore dividing both sides with p_t leads to

$$\frac{\partial_t p_t}{p_t} + \frac{\langle \nabla_x p_t, v\rangle + p_t \operatorname{div}(v)}{p_t} = 0.$$

where we used that $\operatorname{div}(fv) = \langle \nabla_x f, v\rangle + f\operatorname{div}(v)$ for $f \in \mathfrak{P}(\mathcal{M})$ and $v \in \mathfrak{X}(\mathcal{M})$. Finally noting that $\partial_t \log p_t = \frac{\partial_t p_t}{p_t}$, and $\nabla_x \log p_t = \frac{\nabla_x p_t}{p_t}$ we get that equation 3.4 is equivalent to equation A.1. □

A.1.2 Proof of Theorem 2

To prove Theorem 2 we use the following Lemma proved in A.1.3:

Lemma 4. *Consider paths $p, q \in \mathfrak{P}(\mathcal{M})$ where q is generated by a flow $\psi_t : \mathcal{M} \to \mathcal{M}$, and $q_0 = p_0$. Then the following holds:*

$$\mathrm{D}_\ell(p \,\|\, q) = \mathbb{E}_{x\sim p_0} \int_0^1 \frac{p_t(\psi_t(x))}{q_t(\psi_t(x))} \left| \partial_t \left[\log \frac{p_t(\psi_t(x))}{q_t(\psi_t(x))} \right] \right|^\ell dt$$

We now use Lemma 4 to prove each case of Theorem 2:

$\ell = 1$ case. For $\ell = 1$, Lemma 4 provides the following form for D_1:

$$\mathrm{D}_1(p \,\|\, q) = \mathbb{E}_{x\sim p_0} \int_0^1 \left| \partial_t \left[\frac{p_t(\psi_t(x))}{q_t(\psi_t(x))} \right] \right| dt \tag{A.2}$$

which shows that for $\ell = 1$ the path divergence is equivalent to the Total Variation norm of the density ratio p_t/q_t along trajectories of the flow. Second, Jensen's inequality with the convex function $|\cdot|$ provides for every $T \in [0, 1]$

$$\begin{aligned}
\mathrm{D}_1(p \,\|\, q) &\geq \mathbb{E}_{x\sim p_0} \int_0^T \left| \partial_t \left[\frac{p_t(\psi_t(x))}{q_t(\psi_t(x))} \right] \right| dt \\
&\geq \mathbb{E}_{x\sim p_0} \left| \frac{p_T(\psi_T(x))}{q_T(\psi_T(x))} - 1 \right| = \mathbb{E}_{x\sim q_T} \left| \frac{p_T(x)}{q_T(x)} - 1 \right| \\
&= \mathrm{D}_f(p_T, q_T)
\end{aligned}$$

where the first inequality is due to the fact that we integrate over the smaller interval $[0, T]$, in the first equality we used the fact that $\psi_T(x) \sim q_T$ if $x \sim p_0$, and in the last equality we took $f(r) = |r - 1|$.

$1 < \ell < \infty$ case. Lemma 4 again with Jensen's inequality of the convex function $|\cdot|^\ell$ provides

$$\begin{aligned}
\mathrm{D}_\ell(p \,\|\, q)^{\frac{1}{\ell}} &\geq \left| \mathbb{E}_{x\sim p_0} \int_0^T \left[\frac{p_t(\psi_t(x))}{q_t(\psi_t(x))} \right]^{\frac{1}{\ell}} \partial_t \left[\log \frac{p_t(\psi_t(x))}{q_t(\psi_t(x))} \right] dt \right| \\
&= \left| \mathbb{E}_{x\sim p_0} \int_0^T \ell \partial_t \left[\frac{p_t(\psi_t(x))}{q_t(\psi_t(x))} \right]^{\frac{1}{\ell}} dt \right| \\
&= \left| \mathbb{E}_{x\sim p_0} \ell \left(\left[\frac{p_T(\psi_T(x))}{q_T(\psi_T(x))} \right]^{\frac{1}{\ell}} - 1 \right) \right| \\
&= \left| \mathbb{E}_{x\sim q_T} \ell \left(\left[\frac{p_T(x)}{q_T(x)} \right]^{\frac{1}{\ell}} - 1 \right) \right| \\
&= \mathrm{D}_f(p_T \,\|\, q_T)
\end{aligned} \tag{A.3}$$

with $f(r) = \ell(1 - r^{\frac{1}{\ell}})$.

$\ell = \infty$ case. We need to consider equation A.3 and move to the limit $\ell \to \infty$. To this end, we prove that for any $r > 0$, $\ell(1 - r^{\frac{1}{\ell}}) \nearrow -\log(r)$, that is monotonically increasing and converging to $-\log(r)$ as $\ell \to \infty$. Fix $r > 0$, and define the function

$$f(s) = \frac{(1 - r^s)}{s}$$

where $s \in (0, 1)$. Now, using L'Hôpital's rule:

$$\lim_{s\downarrow 0} f(s) = \lim_{s\downarrow 0} \frac{-\log(r) r^s}{1} = -\log(r)$$

Therefore in particular $\lim_{\ell\to\infty} \ell(1 - r^{\frac{1}{\ell}}) = -\log(r)$. Monotonicity follows from the fact that for all $s \in (0, 1)$ and $r > 0$

$$f'(s) = \frac{-\log(r)sr^s + r^s - 1}{s^2} = \frac{r^s}{s}\left(\frac{1 - r^{-s}}{s} - \log(r)\right) \leq 0$$

The inequality can be justified by first noting that $r^s/s > 0$. Second, let $0 < r = \exp(w)$ we get that

$$\frac{1 - r^{-s}}{s} - \log(r) \leq 0$$

which is true iff

$$\frac{1 - \exp(-ws)}{s} - w \leq 0$$

which is true iff

$$1 - ws \leq \exp(-ws)$$

which is true iff for all $u \in \mathbb{R}$

$$1 - u \leq \exp(-u)$$

which is true since $1 - u$ is tangent to $\exp(-u)$ at $u = 0$, and $\exp(u)$ is convex. Since $f'(s)$ is monotonically decreasing in s, $\ell(1 - r^{\frac{1}{\ell}})$ is increasing as $\ell \to \infty$.

Let

$$f_\ell(x) = \ell\left(1 - \left[\frac{p_T(x)}{q_T(x)}\right]^{\frac{1}{\ell}}\right)$$

We showed that $f_\ell(x) \nearrow f(x) = -\log\frac{p_T(x)}{q_T(x)}$. Furthermore, f_ℓ are all integrable since

$$\begin{aligned}
&\int_{\mathcal{M}} \ell\left|1 - \left[\frac{p_T(x)}{q_T(x)}\right]^{\frac{1}{\ell}}\right| q_T(x)dV_x \\
&\leq \ell\int_{\mathcal{M}} q_T(x)dV_x + \ell\int_{\mathcal{M}} p_T(x)^{\frac{1}{\ell}} q_T^{1-\frac{1}{\ell}} dV_x \\
&\leq \ell + \ell\left[\int_{\mathcal{M}} p_T(x)dV_x\right]^{\frac{1}{\ell}}\left[\int_{\mathcal{M}} q_T(x)dV_x\right]^{1-\frac{1}{\ell}} \\
&= 2\ell
\end{aligned}$$

Where in the first inequality we used the triangle inequality, and in the second inequality we used Holder inequality with $\frac{1}{\ell} + \frac{\ell-1}{\ell} = 1$.

We assume that

$$\mathrm{D}_f(p_T \,\|\, q_T) = \int_{\mathcal{M}} f(x)q_T(x)dV_x < \infty.$$

Since $f_\ell(x) \leq f(x)$ and both f_ℓ, f are integrable we have that

$$\int_{\mathcal{M}} f_\ell(x) q_T(x) dV_x \leq \int_{\mathcal{M}} f(x) q_T(x) dV_x < \infty$$

for all ℓ. Therefore, the Monotone Convergence Theorem (see Theorem 2.8.2 in [Bog07]) implies that

$$\lim_{\ell \to \infty} \int_{\mathcal{M}} f_\ell(x) q_T(x) dV_x = \int_{\mathcal{M}} f(x) q_T(x) dV_x$$

Namely,

$$\lim_{\ell \to \infty} \mathbb{E}_{x \sim q_T} \ell \left(1 - \left[\frac{p_T(x)}{q_T(x)} \right]^{\frac{1}{\ell}} \right) = -\mathbb{E}_{x \sim q_T} \log \frac{p_T(x)}{q_T(x)}$$

We are now ready to move to the limit $\ell \to \infty$ in equation A.3:

$$\begin{aligned} \liminf_{\ell \to \infty} \mathrm{D}_\ell(p \,\|\, q)^{\frac{1}{\ell}} &\geq \lim_{\ell \to \infty} \left| \mathbb{E}_{x \sim q_T} \ell \left(1 - \left[\frac{p_T(x)}{q_T(x)} \right]^{\frac{1}{\ell}} \right) \right| \\ &= -\mathbb{E}_{x \sim q_T} \log \frac{p_T(x)}{q_T(x)} \\ &= \mathrm{D}_f(p_T \,\|\, q_T) \end{aligned} \tag{A.4}$$

with $f(r) = -\log(r)$. □

A.1.3 Proof of Lemma 4

Given a time dependent velocity field $v \in \mathfrak{X}(\mathcal{M})$, a diffeomorphism two parameter family $\Psi_{t,t_0} : \mathcal{M} \to \mathcal{M}$ can be defined via the following Ordinary Differential Equation (ODE):

$$\begin{cases} \frac{d}{dt} \Psi_{t,t_0}(x) = v(t, \Psi_{t,t_0}(x)) \\ \Psi_{t_0,t_0}(x) = x \end{cases} \tag{A.5}$$

The CNF diffeomorphism is defined by $\psi_t = \Psi_{t,0}$. Now, consider a smooth function $u(t,x)$, $u : [0,1] \times \mathcal{M} \to \mathbb{R}$, then

$$\partial_t|_{t=t_0} \left[u(t, \Psi_{t,t_0}(x)) \right] = \partial_t u(t_0, x) + \langle \nabla_x u(t_0, x), v(t_0, x) \rangle \,. \tag{A.6}$$

From Theorem 1 we have that

$$\partial_t \log q_t + \langle \nabla_x \log q_t, v \rangle + \mathrm{div}(v) = 0$$

for all $t \in [0,1]$ and $x \in \mathcal{M}$. Subtracting that in our loss, we get

$$\mathrm{D}_\ell(p \,\|\, q) = \mathbb{E}_{t, x \sim p_t} \left| \partial_t \log \frac{p_t}{q_t} + \left\langle \nabla_x \log \frac{p_t}{q_t}, v \right\rangle \right|^\ell \tag{A.7}$$

Now using equation A.6 with $u(t,x) = \log \frac{p_t(x)}{q_t(x)}$, we get

$$\partial_t|_{t=t_0} \log \frac{p_t(\Psi_{t,t_0}(x))}{q_t(\Psi_{t,t_0}(x))} =$$
$$\partial_t \log \frac{p_{t_0}(x)}{q_{t_0}(x)} + \left\langle \nabla_x \log \frac{p_{t_0}(x)}{q_{t_0}(x)}, v(t_0,x) \right\rangle$$

Plugging this in equation A.7 with $t = s$ and $t_0 = t$ we get:

$$\begin{aligned}
\mathrm{D}_\ell(p \,\|\, q) &= \mathbb{E}_{t,x\sim p_t} \left| \partial_s|_{s=t} \log \frac{p_s(\Psi_{s,t}(x))}{q_s(\Psi_{s,t}(x))} \right|^\ell \\
&= \mathbb{E}_{t,x\sim q_t} \frac{p_t(x)}{q_t(x)} \left| \partial_s|_{s=t} \log \frac{p_s(\Psi_{s,t}(x))}{q_s(\Psi_{s,t}(x))} \right|^\ell \\
&= \mathbb{E}_{t,x\sim p_0} \frac{p_t(\psi_t(x))}{q_t(\psi_t(x))} \left| \partial_s|_{s=t} \log \frac{p_s(\Psi_{s,t}(\psi_t(x)))}{q_s(\Psi_{s,t}(\psi_t(x)))} \right|^\ell \\
&= \mathbb{E}_{t,x\sim p_0} \frac{p_t(\psi_t(x))}{q_t(\psi_t(x))} \left| \partial_s|_{s=t} \log \frac{p_s(\psi_s(x))}{q_s(\psi_s(x))} \right|^\ell \\
&= \mathbb{E}_{x\sim p_0} \int_0^1 \frac{p_t(\psi_t(x))}{q_t(\psi_t(x))} \left| \partial_t \log \frac{p_t(\psi_t(x))}{q_t(\psi_t(x))} \right|^\ell dt
\end{aligned}$$

where in the third equality we used the fact that $\psi_t(x) \sim q_t$ if $x \sim p_0$; in the fourth equality we used the fact that $\Psi_{s,t}(\psi_t(x)) = \Psi_{s,t}(\Psi_{t,0}(x)) = \Psi_{s,0}(x) = \psi_s(x)$. □

A.2 Proofs of Chapter 4

A.2.1 Proof of Theorem 3

To verify this, we check that p_t and u_t satisfy the continuity equation (equation 2.28):

$$\begin{aligned}\frac{d}{dt}p_t(x) &= \int \Big(\frac{d}{dt}p_t(x|x_1)\Big)q(x_1)dx_1 = -\int \mathrm{div}\Big(u_t(x|x_1)p_t(x|x_1)\Big)q(x_1)dx_1 \\ &= -\mathrm{div}\Big(\int u_t(x|x_1)p_t(x|x_1)q(x_1)dx_1\Big) = -\mathrm{div}\Big(u_t(x)p_t(x)\Big),\end{aligned}$$

where in the second equality we used the fact that $u_t(\cdot|x_1)$ generates $p_t(\cdot|x_1)$, in the last equality we used equation 4.4. Furthermore, the first and third equalities are justified by assuming the integrands satisfy the regularity conditions of the Leibniz Rule (for exchanging integration and differentiation). Theorem 3 can also be derived from the Diffusion Mixture Representation Theorem in [Pel21] that provides a formula for the marginal drift and diffusion coefficients in diffusion SDEs. □

A.2.2 Proof of Theorem 4

To ensure existence of all integrals and to allow the changing of integration order (by Fubini's Theorem) in the following we assume that $q(x)$ and $p_t(x|x_1)$ are decreasing to zero at a sufficient speed as $\|x\| \to \infty$, and that $u_t, v_t, \nabla_\theta v_t$ are bounded.

First, using the standard bilinearity of the 2-norm we have that

$$\begin{aligned}\|v_t(x) - u_t(x)\|^2 &= \|v_t(x)\|^2 - 2\left\langle v_t(x), u_t(x)\right\rangle + \|u_t(x)\|^2 \\ \|v_t(x) - u_t(x|x_1)\|^2 &= \|v_t(x)\|^2 - 2\left\langle v_t(x), u_t(x|x_1)\right\rangle + \|u_t(x|x_1)\|^2\end{aligned}$$

Next, remember that u_t is independent of θ and note that

$$\begin{aligned}\mathbb{E}_{p_t(x)}\|v_t(x)\|^2 &= \int \|v_t(x)\|^2 p_t(x)dx = \int \|v_t(x)\|^2 p_t(x|x_1)q(x_1)dx_1dx \\ &= \mathbb{E}_{q(x_1),p_t(x|x_1)}\|v_t(x)\|^2,\end{aligned}$$

where in the second equality we use equation 4.2, and in the third equality we change the order of integration. Next,

$$\begin{aligned}\mathbb{E}_{p_t(x)}\left\langle v_t(x), u_t(x)\right\rangle &= \int \left\langle v_t(x), \frac{\int u_t(x|x_1)p_t(x|x_1)q(x_1)dx_1}{p_t(x)}\right\rangle p_t(x)dx \\ &= \int \left\langle v_t(x), \int u_t(x|x_1)p_t(x|x_1)q(x_1)dx_1\right\rangle dx \\ &= \int \left\langle v_t(x), u_t(x|x_1)\right\rangle p_t(x|x_1)q(x_1)dx_1dx \\ &= \mathbb{E}_{q(x_1),p_t(x|x_1)}\left\langle v_t(x), u_t(x|x_1)\right\rangle,\end{aligned}$$

where in the last equality we change again the order of integration. □

A.2.3 Proof of Theorem 5

For notational simplicity let $w_t(x) = u_t(x|x_1)$. Now consider equation 2.24:

$$\frac{d}{dt}\psi_t(x) = w_t(\psi_t(x)).$$

Since ψ_t is invertible (as $\sigma_t(x_1) > 0$) we let $x = \psi^{-1}(y)$ and get

$$\psi_t'(\psi^{-1}(y)) = w_t(y), \tag{A.8}$$

where we used the apostrophe notation for the derivative to emphasize that ψ_t' is evaluated at $\psi^{-1}(y)$. Now, inverting $\psi_t(x)$ provides

$$\psi_t^{-1}(y) = \frac{y - \mu_t(x_1)}{\sigma_t(x_1)}.$$

Differentiating ψ_t with respect to t gives

$$\psi_t'(x) = \sigma_t'(x_1)x + \mu_t'(x_1).$$

Plugging these last two equations in equation A.8 we get

$$w_t(y) = \frac{\sigma_t'(x_1)}{\sigma_t(x_1)}\left(y - \mu_t(x_1)\right) + \mu_t'(x_1)$$

as required. □

A.3 Proofs of Chapter 5

A.3.1 Proof of Lemma 1

We need only prove that the marginal probability path interpolates between q_0 and q_1.

$$p_0(x) = \int p_0(x|x_1)q_1(x_1)dx_1 = \int q(x|x_1)q_1(x_1)dx_1 = q_0(x). \tag{A.9}$$

Then since $u_t(x|x_1)$ transports all points $x \in \mathbb{R}^D$ to x_1 at time $t = 1$, we satisfy $p_{t=1}(x|x_1) = \delta(x - x_1)$.

$$p_1(x) = \int p_1(x|x_1)q_1(x_1)dx = \int \delta(x - x_1)q_1(x_1)dx_1 = q_1(x). \tag{A.10}$$

Theorems 3 and 4 can then be used to prove that (i) the marginal velocity field $u_t(x)$ transports between $p_0 = q_0$ and $p_1 = q_1$, and (ii) the Joint CFM objective has the same gradient in expectation as the Flow Matching objective and is uniquely minimized by $v_t(x; \theta) = u_t(x)$.

A.3.2 Proof of Lemma 2

Note that

$$\begin{aligned}
&\mathrm{Cov}_{p_t(x_1|x)}\left(\nabla_\theta \left\|v_t(x;\theta) - u_t(x|x_1)\right\|^2\right) = \mathrm{Cov}_{p_t(x_1|x)}\left(\nabla_\theta \left\|v_t(x;\theta)\right\|^2 - \left(\nabla_\theta v_t(x;\theta)\right)^{\mathsf{T}} u_t(x|x_1)\right) \\
&= \left(\nabla_\theta v_t(x;\theta)\right)^{\mathsf{T}} \mathrm{Cov}_{p_t(x_1|x)}\left(u_t(x|x_1)\right)\left(\nabla_\theta v_t(x;\theta)\right),
\end{aligned} \tag{A.11}$$

and that

$$\mathrm{Cov}_{p_t(x_1|x)}\left(u_t(x|x_1)\right) = \mathbb{E}_{p_t(x_1|x)}\left(u_t(x|x_1) - u_t(x)\right)\left(u_t(x|x_1) - u_t(x)\right)^\top. \tag{A.12}$$

Here, we used that $u_t(x) = \mathbb{E}_{p_t(x_1|x)}\left[u_t(x|x_1)\right]$ by equation 4.4. If we take the trace on both sides of equation A.11, we get

$$\begin{aligned}
&\mathrm{Tr}\left[\mathrm{Cov}_{p_t(x_1|x)}\left(\nabla_\theta \left\|v_t(x;\theta) - u_t(x|x_1)\right\|^2\right)\right] = \mathrm{Tr}\left[\left(\nabla_\theta v_t(x;\theta)\right)^{\mathsf{T}} \mathrm{Cov}_{p_t(x_1|x)}\left(u_t(x|x_1)\right)\left(\nabla_\theta v_t(x;\theta)\right)\right] \\
&= \mathrm{Tr}\left[\mathrm{Cov}_{p_t(x_1|x)}\left(u_t(x|x_1)\right)\left(\nabla_\theta v_t(x;\theta)\right)\left(\nabla_\theta v_t(x;\theta)\right)^{\mathsf{T}}\right] \\
&= \left\langle \mathrm{Cov}_{p_t(x_1|x)}\left(u_t(x|x_1)\right), \left(\nabla_\theta v_t(x;\theta)\right)\left(\nabla_\theta v_t(x;\theta)\right)^{\mathsf{T}} \right\rangle_F \\
&\leq \left\|\mathrm{Cov}_{p_t(x_1|x)}\left(u_t(x|x_1)\right)\right\|_F \left\|\left(\nabla_\theta v_t(x;\theta)\right)\left(\nabla_\theta v_t(x;\theta)\right)^{\mathsf{T}}\right\|_F \\
&\leq \mathbb{E}_{p_t(x_1|x)}\left\|\left(u_t(x|x_1) - u_t(x)\right)\left(u_t(x|x_1) - u_t(x)\right)^\top\right\|_F \left\|\left(\nabla_\theta v_t(x;\theta)\right)\left(\nabla_\theta v_t(x;\theta)\right)^\top\right\|_F \\
&= \left\|\nabla_\theta v_t(x;\theta)\right\|^2 \mathbb{E}_{p_t(x_1|x)}\left\|u_t(x|x_1) - u_t(x)\right\|^2.
\end{aligned} \tag{A.13}$$

The second equality holds because $\mathrm{Tr}(AB) = \mathrm{Tr}(BA)$ when both expressions are well defined, and the third equality holds by the definition of the Frobenius inner product $\langle \cdot, \cdot \rangle_F$. The first inequality holds by the Cauchy-Schwarz inequality. The second inequality holds

by equation equation A.12 and by the triangle inequality. In the last equality we used that for any vector v, $\|vv^\top\|_F = (\mathrm{Tr}(vv^\top, vv^\top))^{1/2} = \|v\|^2$. This proves equation 5.11.

To prove equation 5.12, we write:

$$\begin{aligned}
&\mathbb{E}_{t,p_t(x)}[\sigma_{t,x}^2] \\
&\leq \mathbb{E}_{t,p_t(x)}[\|\nabla_\theta v_t(x;\theta)\|^2 \mathbb{E}_{p_t(x_1|x)}\|u_t(x|x_1) - u_t(x)\|^2] \\
&\leq \max_{x,t} \|\nabla_\theta v_t(x;\theta)\|^2 \times \mathbb{E}_{t,p_t(x)}[\mathbb{E}_{p_t(x_1|x)}\|u_t(x|x_1) - u_t(x)\|^2] \\
&= \max_{x,t} \|\nabla_\theta v_t(x;\theta)\|^2 \times \mathbb{E}_{t,q(x_0,x_1)}[\|u_t(x_t|x_1) - v_t(x_t;\theta)\|^2] \leq \max_{t,x} \|\nabla_\theta v_t(x;\theta)\|^2 \times \mathcal{L}_{\mathrm{JCFM}}
\end{aligned} \tag{A.14}$$

Here, the first inequality holds by equation 5.11, and the last inequality holds because $u_t(x)$ is the minimizer of $\mathcal{L}_{\mathrm{JCFM}}$.

A.3.3 Proof of Lemma 3

For an arbitrary test function f, by the construction of q we write

$$\mathbb{E}_{q(x_0,x_1)} f(x_0) = \mathbb{E}_{\{x_0^{(i)}\}_{i=1}^k \sim q_0, \{x_1^{(i)}\}_{i=1}^k \sim q_1} \mathbb{E}_{q^k(x_0,x_1)} f(x_0). \tag{A.15}$$

Since q^k has marginal $\frac{1}{k}\sum_{i=1}^k \delta(x_0 - x_0^{(i)})$ because π is a doubly stochastic matrix, we obtain that $\mathbb{E}_{q^k(x_0,x_1)} f(x_0) = \frac{1}{k}\sum_{i=1}^k f(x_0^{(i)})$ and then the right-hand side is equal to

$$\mathbb{E}_{\{x_0^{(i)}\}_{i=1}^k \sim q_0, \{x_1^{(i)}\}_{i=1}^k \sim q_1} \frac{1}{k}\sum_{i=1}^k f(x_0^{(i)}) = \mathbb{E}_{q_0(x_0)} f(x_0), \tag{A.16}$$

which proves that the marginal of q for x_0 is q_0. The same argument works for the x_1 marginal.

A.3.4 Proof of Theorem 6

Notation We begin by recalling and introducing some additional notation. Let $\boldsymbol{X}_0 = (x_0^i)_{i=1}^{+\infty}$, $\boldsymbol{X}_1 = (x_1^i)_{i=1}^{+\infty}$ be sequences of i.i.d. samples from the distributions q_0 and q_1, and denote by $\boldsymbol{X}_0^k = (x_0^i)_{i=1}^k$, $\boldsymbol{X}_1^k = (x_1^i)_{i=1}^k$ the finite sequences containing the initial k samples. We denote by q_0^k and q_1^k the empirical distributions corresponding to $\boldsymbol{X}_0^k$ and $\boldsymbol{X}_1^k$, i.e. $q_0^k = \frac{1}{k}\sum_{i=1}^k \delta_{x_0^i}$, $q_1^k = \frac{1}{k}\sum_{i=1}^k \delta_{x_1^i}$. Let q^k be the distribution over $\mathbb{R}^d \times \mathbb{R}^d$, which is output by the matching algorithm; q^k has marginals that are equal to q_0^k and q_1^k. Let q^* be the optimal transport plan between q_0 and q_1, and let $\tilde{q}^k$ be the optimal transport plan between q^k and q under the quadratic cost. Using this additional notation, we rewrite some of the objects that were defined in the main text in a lengthier, more precise way:

(i) The marginal vector field corresponding to sample size k:

$$u_t^k(x) = \mathbb{E}_{\boldsymbol{X}_0^k \overset{\text{iid}}{\sim} q_0, \boldsymbol{X}_1^k \overset{\text{iid}}{\sim} q_1, (x_0,x_1)\sim q^k}[x_1 - x_0 | x = tx_1 + (1-t)x_0], \qquad \forall t \in [0,1]. \tag{A.17}$$

We made the dependency on k explicit, and we used that $\psi_t(x_0|x_1) = tx_1 + (1-t)x_0$. Note that equivalently, we can write u_t^k as the solution of a simple variational problem.

$$u_t^k = \arg\min_{u_t} \mathbb{E}_{\boldsymbol{X}_0^k \overset{\text{iid}}{\sim} q_0, \boldsymbol{X}_1^k \overset{\text{iid}}{\sim} q_1, (x_0,x_1)\sim q^k} \|x_1 - x_0 - u_t(tx_1 + (1-t)x_0)\|^2, \qquad \forall t \in [0,1]. \tag{A.18}$$

(ii) The flow $\psi_t^k(x_0)$ corresponding to u_t^k, i.e. the solution of $\frac{dx_t}{dt} = u_t^k(x_t)$ with initial condition x_0. We made the dependency on k explicit.

(iii) The straightness of the flow ψ_t^k:

$$S^k = \mathbb{E}_{t\sim \mathrm{U}(0,1), x_0 \sim q_0}\left[\|u_t^k(\psi_t^k(x_0))\|^2 - \|\psi_1^k(x_0) - x_0\|^2\right]. \tag{A.19}$$

Assumptions We will use the following three assumptions, which allow us to potentially extend our result beyond BatchOT:

(A1) The distributions q_0 and q_1 over $\mathbb{R}^d$ have bounded supports, i.e. there exists $C > 0$ such that for any $x \in \mathrm{supp}(q_0) \cup \mathrm{supp}(q_1)$, $\|x\| \leq C$.

(A2) q_0 admits a density and the optimal transport map T between q_0 and q_1 under the quadratic cost is continuous.

(A3) We assume that almost surely w.r.t. the draw of $\boldsymbol{X}_0$ and $\boldsymbol{X}_1$, q^k converges weakly to q as $k \to \infty$.

Some comments are in order as to when assumptions **(A2)**, **(A3)** hold, since they are not directly verifiable. By the Caffarelli regularity theorem (see [Vil08], Ch. 12, originally in [Caf92]), a sufficient condition for **(A2)** to hold is the following:

(A2') q_0 and q_1 have a common support Ω which is compact and convex, have α-Hölder densities, and they satisfy the lower bound $q_0, q_1 > \gamma$ for some $\gamma > 0$.

Assumption **(A3)** holds when the matching algorithm is BatchOT, that is, when q^k is the optimal transport plan between q_0^k and q_1^k, as shown by the following proposition, which is proven in App. A.3.4.

Proposition 2. *Let q^k be the optimal transport plan between q_0^k and q_1^k under the quadratic cost (i.e. the result of Steps [1-3] under BatchOT). We have that almost surely w.r.t. the draws of $\boldsymbol{X}_0$ and $\boldsymbol{X}_1$, the sequence $(q_k)_{k\geq 0}$ converges weakly to q^*, i.e. assumption **(A3)** holds.*

Proof structure We split the proof of Theorem 6 into two parts: in Subsubsec. A.3.4 we prove that the optimal value of the Joint CFM objective equation 5.10 converges to

zero as $k \to \infty$. In Subsubsec. A.3.4, we prove that the straightness converges to zero and the transport cost converges to the optimal transport cost as $k \to \infty$.

Convergence of the optimal value of the CFM objective

Theorem 8. *Suppose that assumptions **(A1)**, **(A2)** and **(A3)** hold. We have that*

$$\lim_{k\to\infty} \mathbb{E}_{t\sim \mathrm{U}(0,1), \boldsymbol{X}_0^k \overset{\text{iid}}{\sim} q_0, \boldsymbol{X}_1^k \overset{\text{iid}}{\sim} q_1, (x_0,x_1)\sim q^k} \|x_1 - x_0 - u_t^k(tx_1 + (1-t)x_0)\|^2 = 0, \tag{A.20}$$

where u_t^k is the marginal vector field as defined in equation A.17.

Proof. The transport plan q^* satisfies the non-crossing paths property, that is, for each $x \in \mathbb{R}^d$ and $t \in [0,1]$, there exists at most one pair (x_0, x_1) such that $x = tx_1 + (1-t)x_0$ [NIO20; Vil03]. Consequently, when such a pair (x_0', x_1') exists, we have that the analogue of the vector field in equation A.17 admits a simple expression:

$$u_t^*(x) := \mathbb{E}_{(x_0,x_1)\sim q^*}[x_1 - x_0 | x = tx_1 + (1-t)x_0] = x_1' - x_0' \tag{A.21}$$

This directly implies that

$$\mathbb{E}_{(x_0,x_1)\sim q^*} \|x_1 - x_0 - u_t^*(tx_1 + (1-t)x_0)\|^2 = 0. \tag{A.22}$$

Applying this, we can write

$$\begin{aligned} &\mathbb{E}_{t\sim \mathrm{U}(0,1),(x_0,x_1)\sim q^k} \|x_1 - x_0 - u_t^*(tx_1 + (1-t)x_0)\|^2 \\ &= |\mathbb{E}_{(x_0,x_1)\sim q^k}[\mathbb{E}_{t\sim \mathrm{U}(0,1)} \|x_1 - x_0 - u_t^*(tx_1 + (1-t)x_0)\|^2] \\ &\quad - \mathbb{E}_{(x_0,x_1)\sim q^*}[\mathbb{E}_{t\sim \mathrm{U}(0,1)} \|x_1 - x_0 - u_t^*(tx_1 + (1-t)x_0)\|^2]|. \end{aligned} \tag{A.23}$$

Now, define the function $f : \mathrm{supp}(q_0) \times \mathrm{supp}(q_1) \to \mathbb{R}$ as

$$f(x_0, x_1) = \mathbb{E}_{t\sim \mathrm{U}(0,1)} \|x_1 - x_0 - u_t^*(tx_1 + (1-t)x_0)\|^2. \tag{A.24}$$

By Lemma 5, which holds under **(A1)** and **(A2)**, we have that f is bounded and continuous. Assumption **(A3)** states that almost surely w.r.t. the draws of $\boldsymbol{X}_0$ and $\boldsymbol{X}_1$, the measure q_k converges weakly to q^*. We apply the definition of weak convergence of measures, which implies that almost surely,

$$\lim_{k\to\infty} \mathbb{E}_{(x_0,x_1)\sim q^k}[f(x_0, x_1)] = \mathbb{E}_{(x_0,x_1)\sim q}[f(x_0, x_1)]. \tag{A.25}$$

Equivalently, the right-hand side of equation A.23 converges to zero as k tends to infinity. Hence, $\mathbb{E}_{t\sim \mathrm{U}(0,1),(x_0,x_1)\sim q^k} \|x_1 - x_0 - u_t^*(tx_1 + (1-t)x_0)\|^2 \to 0$ almost surely. Almost sure convergence implies convergence in probability, which means that

$$\Pr(\mathbb{E}_{t\sim \mathrm{U}(0,1),(x_0,x_1)\sim q^k} \|x_1 - x_0 - u_t^*(tx_1 + (1-t)x_0)\|^2 > \epsilon) \xrightarrow{k\to\infty} 0, \quad \forall \epsilon > 0. \tag{A.26}$$

Here, the randomness comes only from drawing the random variables $\boldsymbol{X}_0^k, \boldsymbol{X}_1^k$. Also, using

again that f is bounded, say by the constant $C > 0$, we can write $\mathbb{E}_{t\sim \mathrm{U}(0,1),(x_0,x_1)\sim q^k}\|x_1 - x_0 - u_t^*(tx_1 + (1-t)x_0)\|^2 \leq C$, for all $k \geq 0$. A crude bound yields

$$\mathbb{E}_{t\sim \mathrm{U}(0,1),\boldsymbol{X}_0^k,\boldsymbol{X}_1^k,(x_0,x_1)\sim q^k}\|x_1 - x_0 - u_t^*(tx_1 + (1-t)x_0)\|^2 \tag{A.27}$$

$$\leq \epsilon + C\mathrm{Pr}(\mathbb{E}_{t\sim \mathrm{U}(0,1),(x_0,x_1)\sim q^k}\|x_1 - x_0 - u_t^*(tx_1 + (1-t)x_0)\|^2 > \epsilon). \tag{A.28}$$

In this equation and from now, we write $\boldsymbol{X}_0^k, \boldsymbol{X}_1^k$ instead of $\boldsymbol{X}_0^k \overset{\text{iid}}{\sim} q_0, \boldsymbol{X}_1^k \overset{\text{iid}}{\sim} q_1$ for shortness. We can take ϵ arbitrarily small, and for a given ϵ we can make the second term in the right-hand side arbitrarily small by taking k large enough. Hence, we obtain that

$$\lim_{k\to\infty} \mathbb{E}_{t\sim \mathrm{U}(0,1),\boldsymbol{X}_0^k,\boldsymbol{X}_1^k,(x_0,x_1)\sim q^k}\|x_1 - x_0 - u_t^*(tx_1 + (1-t)x_0)\|^2 = 0. \tag{A.29}$$

To conclude the proof, we use the variational characterization of u_t^k given in equation A.18, which implies that

$$\begin{aligned}&\mathbb{E}_{t\sim \mathrm{U}(0,1),\boldsymbol{X}_0^k,\boldsymbol{X}_1^k,(x_0,x_1)\sim q^k}\|x_1 - x_0 - u_t^k(tx_1 + (1-t)x_0)\|^2 \\ &\leq \mathbb{E}_{t\sim \mathrm{U}(0,1),\boldsymbol{X}_0^k,\boldsymbol{X}_1^k,(x_0,x_1)\sim q^k}\|x_1 - x_0 - u_t^*(tx_1 + (1-t)x_0)\|^2 \to 0.\end{aligned} \tag{A.30}$$

□

Definition 2 (Weak convergence of probability measures, convergence of random variables in distribution)**.** *A sequence of probability measures $(p_k)_{k=0}^{+\infty}$ on a common metric space and Borel σ-algebra converges weakly to the probability measure p if for any bounded continuous function f, $\lim_{k\to\infty} \mathbb{E}_{p_k}[f] = \mathbb{E}_p[f]$. A sequence of random variables $(X_k)_{k=0}^{+\infty}$ converges in distribution to the random variable X if the sequence of their laws $(p_k)_{k=0}^{+\infty}$ converges weakly to the law p of X.*

Lemma 5. *Let f be the function defined in equation equation A.24. Suppose that assumptions **(A1)** and **(A2)** hold. Then, f is bounded and continuous.*

Proof. First, we show that the function u_t^* defined in equation equation A.21 is bounded and continuous wherever it is defined. It is bounded because $u_t^*(x) = x_1' - x_0'$ for some x_0' in $\mathrm{supp}(q_0)$ and x_1' in $\mathrm{supp}(q_1)$, which are both bounded by assumption.

To show that u_t^* is continuous, we use that q_0 is absolutely continuous and that consequently a transport map T exists. Moreover, we have that $x_1' = T(x_0')$. Consider the transport map T_t at time t, defined as $T_t(x) = tT(x) + (1-t)x$. Thus, we can write that $u_t^*(T_t(x_0)) = T(x_0) - x_0$. The non-crossing paths property implies that T_t is invertible, which means that an inverse T_t^{-1} exists. We can write

$$u_t^*(x) = T(T_t^{-1}(x)) - T_t^{-1}(x). \tag{A.31}$$

By assumption **(A2)**, the transport map T is continuous, and so is T_t. It is a well-known fact that if E, E' are metric spaces, E is compact, and $f : E \to E'$ a continuous bijective function, then $f^{-1} : E' \to E$ is continuous. Thus, T_t^{-1} is also continuous. From equation equation A.31, we conclude that u_t^* is continuous.

The rest of the proof is straightforward: $(x_1, x_0) \mapsto \|x_1 - x_0 - u_t^*(tx_1 + (1-t)x_0)\|^2$ is bounded and continuous on the bounded supports of q_0 and q_1 for all $t \in [0,1]$, and then f is also continuous and bounded since it is an average of continuous bounded functions, applying the dominated convergence theorem. □

Convergence of the straightness and the transport cost

Theorem 9. *Suppose that assumptions **(A1)** and **(A3)** hold. Then,*

(i) We have that $\lim_{k\to\infty} S^k = 0$*, where* S^k *is the straightness defined in equation A.19.*

(ii) We also have that

$$\mathbb{E}_{t\sim \mathrm{U}(0,1), x_0\sim q_0}\|u_t^k(\psi_t^k(x_0))\|^2 \geq \mathbb{E}_{x_0\sim q_0}\|\psi_1^k(x_0) - x_0\|^2 \geq W_2^2(q_0, q_1), \tag{A.32}$$

$$\lim_{k\to\infty} \mathbb{E}_{t\sim \mathrm{U}(0,1), x_0\sim q_0}\|u_t^k(\psi_t^k(x_0))\|^2 = \lim_{k\to\infty} \mathbb{E}_{x_0\sim q_0}\|\psi_1^k(x_0) - x_0\|^2 = W_2^2(q_0, q_1). \tag{A.33}$$

Proof. We begin with the proof of (i). We introduce some additional notation. We define the quantity S^* in analogy with S^k:

$$S^* = \mathbb{E}_{t\sim \mathrm{U}(0,1), x_0\sim q_0}\left[\|u_t^*(\psi_t^*(x_0))\|^2 - \|\psi_1^*(x_0) - x_0\|^2\right], \tag{A.34}$$

and $\psi_t^*(x_0)$ as the solution of the ODE $\frac{dx_t}{dt} = u_t^*(x_t)$. Since the trajectories for the optimal transport vector field are straight lines, we deduce from the alternative expression of the straightness (equation equation 5.14) that $S^* = 0$. An alternative way to see this is by the Benamou-Brenier theorem [BB00], which states that the dynamic optimal transport cost $\mathbb{E}_{t\sim \mathrm{U}(0,1), x_0\sim q_0}\|u_t^*(\psi_t^*(x_0))\|^2$ is equal to the static optimal transport cost $\mathbb{E}_{t\sim \mathrm{U}(0,1), x_0\sim q_0}\|\psi_1^*(x_0) - x_0\|^2$.

We will first prove that $\mathbb{E}_{t\sim \mathrm{U}(0,1), x_0\sim q_0}\|u_t^k(\psi_t^k(x_0))\|^2$ converges to $\mathbb{E}_{t\sim \mathrm{U}(0,1), x_0\sim q_0}\|u_t^*(\psi_t^*(x_0))\|^2$ and then that $\mathbb{E}_{t\sim \mathrm{U}(0,1), x_0\sim q_0}\|\psi_1^k(x_0) - x_0\|^2$ converges to $\mathbb{E}_{t\sim \mathrm{U}(0,1), x_0\sim q_0}\|\psi_1^*(x_0) - x_0\|^2$.

For given instances of $\boldsymbol{X}_0^k$ and $\boldsymbol{X}_1^k$, let $\tilde{q}_k$ be the optimal transport plan between the optimal transport plans q and q^k. In other words, $\tilde{q}_k$ is a measure over the variables x_0, x_1, x_0', x_1', and is such that its marginal w.r.t. x_0, x_1 is q, while its marginal w.r.t. x_0', x_1' is q^k. That is, we will use that for all $t \in [0,1]$, the random variable $tx_1 + (1-t)x_0$, with $(x_0, x_1) \sim q^k$, and q^k built randomly from $\boldsymbol{X}_0^k \overset{\text{iid}}{\sim} q_0, \boldsymbol{X}_1^k \overset{\text{iid}}{\sim} q_1$, has the same distribution as the random variable $\psi_t^k(x_0)$, with $x_0 \sim q_0$. This is a direct consequence of Lemma 1, i.e. the marginal vector field u_t generates the marginal probability path p_t. An analogous statement holds for q, i.e. the random variable $tx_1 + (1-t)x_0$, with $(x_0, x_1) \sim q$, has the same distribution as the random variable $\psi_t^*(x_0)$, with $x_0 \sim q_0$. However, in this case it can be obtained immediately by the non-crossing paths property of the optimal transport plan. Hence,

$$\begin{aligned}
\mathbb{E}_{t\sim \mathrm{U}(0,1), x_0\sim q_0}\|u_t^*(\psi_t^*(x_0))\|^2 &= \mathbb{E}_{t\sim \mathrm{U}(0,1), (x_0,x_1)\sim q}\|u_t^*(tx_1 + (1-t)x_0)\|^2, \\
\mathbb{E}_{t\sim \mathrm{U}(0,1), x_0\sim q_0}\|u_t^k(\psi_t^k(x_0))\|^2 &= \mathbb{E}_{t\sim \mathrm{U}(0,1), \boldsymbol{X}_0^k, \boldsymbol{X}_1^k, (x_0,x_1)\sim q^k}\|u_t^k(tx_1 + (1-t)x_0)\|^2.
\end{aligned} \tag{A.35}$$

Using this and the definition of $\tilde{q}^k$, and applying Jensen's inequality, the Cauchy-Schwarz inequality and the triangle inequality, we can write

$$
\begin{aligned}
&\left|\mathbb{E}_{t\sim\mathrm{U}(0,1),x_0\sim q_0}\|u_t^*(\psi_t^*(x_0))\|^2 - \mathbb{E}_{t\sim\mathrm{U}(0,1),x_0\sim q_0}\|u_t^k(\psi_t^k(x_0))\|^2\right| \\
&= \left|\mathbb{E}_{t\sim\mathrm{U}(0,1),(x_0,x_1)\sim q}\|u_t^*(tx_1+(1-t)x_0)\|^2 - \mathbb{E}_{t\sim\mathrm{U}(0,1),\boldsymbol{X}_0^k,\boldsymbol{X}_1^k,(x_0',x_1')\sim q^k}\|u_t^k(tx_1'+(1-t)x_0')\|^2\right| \\
&= \left|\mathbb{E}_{t\sim\mathrm{U}(0,1),\boldsymbol{X}_0^k,\boldsymbol{X}_1^k,(x_0,x_1,x_0',x_1')\sim\tilde{q}^k}\left[\|u_t^*(tx_1+(1-t)x_0)\|^2 - \|u_t^k(tx_1'+(1-t)x_0')\|^2\right]\right| \\
&= \Big|\mathbb{E}_{t\sim\mathrm{U}(0,1),\boldsymbol{X}_0^k,\boldsymbol{X}_1^k,(x_0,x_1,x_0',x_1')\sim\tilde{q}^k}\Big[\left(\|u_t^*(tx_1+(1-t)x_0)\| - \|u_t^k(tx_1'+(1-t)x_0')\|\right) \\
&\qquad\qquad \times \left(\|u_t^*(tx_1+(1-t)x_0)\| + \|u_t^k(tx_1'+(1-t)x_0')\|\right)\Big]\Big| \\
&\leq \left(\mathbb{E}_{t\sim\mathrm{U}(0,1),\boldsymbol{X}_0^k,\boldsymbol{X}_1^k,(x_0,x_1,x_0',x_1')\sim\tilde{q}^k}\left[\left(\|u_t^*(tx_1+(1-t)x_0)\| - \|u_t^k(tx_1'+(1-t)x_0')\|\right)^2\right]\right)^{1/2} \\
&\qquad\qquad \times \left(\mathbb{E}_{t\sim\mathrm{U}(0,1),\boldsymbol{X}_0^k,\boldsymbol{X}_1^k,(x_0,x_1,x_0',x_1')\sim\tilde{q}^k}\left[\left(\|u_t^*(tx_1+(1-t)x_0)\| + \|u_t^k(tx_1'+(1-t)x_0')\|\right)^2\right]\right)^{1/2} \\
&\leq \left(\mathbb{E}_{t\sim\mathrm{U}(0,1),\boldsymbol{X}_0^k,\boldsymbol{X}_1^k,(x_0,x_1,x_0',x_1')\sim\tilde{q}^k}\|u_t^*(tx_1+(1-t)x_0) - u_t^k(tx_1'+(1-t)x_0')\|^2\right)^{1/2} \\
&\qquad\qquad \times \left(\mathbb{E}_{t\sim\mathrm{U}(0,1),\boldsymbol{X}_0^k,\boldsymbol{X}_1^k,(x_0,x_1,x_0',x_1')\sim\tilde{q}^k}\left[\left(\|u_t^*(tx_1+(1-t)x_0)\| + \|u_t^k(tx_1'+(1-t)x_0')\|\right)^2\right]\right)^{1/2}.
\end{aligned}
\tag{A.36}
$$

Remark that the second factor in the right-hand side is bounded because u_t^* and u_t^k are bounded. Using Lemma 6, we obtain that the first factor in the right-hand side tends to zero as k grows. Thus,

$$
\left|\mathbb{E}_{t\sim\mathrm{U}(0,1),x_0\sim q_0}\|u_t^*(\psi_t^*(x_0))\|^2 - \mathbb{E}_{t\sim\mathrm{U}(0,1),x_0\sim q_0^k}\|u_t^k(\psi_t^k(x_0))\|^2\right| \xrightarrow{k\to\infty} 0. \tag{A.37}
$$

Now, since $\mathbb{E}_{x_0\sim q_0}\|\psi_1^*(x_0)-x_0\|^2$ is the optimal cost and $S^*=0$, we write

$$
\begin{aligned}
\left|\mathbb{E}_{x_0\sim q_0}\|\psi_1^*(x_0)-x_0\|^2 - \mathbb{E}_{x_0\sim q_0}\|\psi_1^k(x_0)-x_0\|^2\right| &= \mathbb{E}_{x_0\sim q_0}\|\psi_1^k(x_0)-x_0\|^2 - \mathbb{E}_{x_0\sim q_0}\|\psi_1^*(x_0)-x_0\|^2 && \text{(A.38)} \\
&= \mathbb{E}_{x_0\sim q_0}\|\psi_1^k(x_0)-x_0\|^2 - \mathbb{E}_{t\sim\mathrm{U}(0,1),x_0\sim q_0}\|u_t^*(\psi_t^*(x_0))\|^2. && \text{(A.39)}
\end{aligned}
$$

Since ψ_t^k is the flow of u_t^k and by Jensen's inequality, we have that

$$
\begin{aligned}
\mathbb{E}_{x_0\sim q_0}\|\psi_1^k(x_0)-x_0\|^2 &= \mathbb{E}_{x_0\sim q_0}\left\|\int_0^1 u_s^k(\psi_s^k(x_0'))\,ds\right\|^2 \\
&\leq \mathbb{E}_{x_0\sim q_0}\int_0^1 \|u_s^k(\psi_s^k(x_0))\|^2\,ds = \mathbb{E}_{t\sim U(0,1),x_0\sim q_0}\|u_t^k(\psi_t^k(x_0))\|^2.
\end{aligned}
$$

Plugging this into equation A.38, we get that

$$
\begin{aligned}
&\left|\mathbb{E}_{x_0\sim q_0}\|\psi_1^*(x_0)-x_0\|^2 - \mathbb{E}_{x_0\sim q_0}\|\psi_1^k(x_0)-x_0\|^2\right| && \text{(A.40)} \\
&\leq \mathbb{E}_{t\sim U(0,1),x_0\sim q_0}\|u_t^k(\psi_t^k(x_0))\|^2 - \mathbb{E}_{t\sim\mathrm{U}(0,1),x_0\sim q_0}\|u_t^*(\psi_t^*(x_0))\|^2 \xrightarrow{k\to\infty} 0, && \text{(A.41)}
\end{aligned}
$$

where the limit holds by equation A.40. Putting together equation A.37 and equation A.40, we end up with $S^k = |S^* - S^k| \xrightarrow{k\to\infty} 0$, which proves (i).

We prove (ii). The first inequality in equation A.32 holds because $S^k \geq 0$ since it can

be written in a form analogous to equation 5.14. The second inequality in equation A.32 holds because $\mathbb{E}_{x_0\sim q_0}\|\psi_1^k(x_0)-x_0\|^2$ is the squared transport cost for the map $x\mapsto\psi_1^k(x)$, which must be at least as large as the optimal cost. The first equality in equation A.32 is a direct consequence of (i). To prove the second equality in equation A.32, we remark that $W_2^2(q_0,q_1)=\mathbb{E}_{x_0\sim q_0}\|\psi_1^*(x_0)-x_0\|^2$. Then, equation equation A.40 readily implies that $|\mathbb{E}_{x_0\sim q_0^k}\|\psi_1^k(x_0)-x_0\|^2-W_2^2(q_0,q_1)|\xrightarrow{k\to\infty}0$. □

Lemma 6. *Suppose that assumptions **(A1)** and **(A3)** hold. Let $\tilde{q}^k$ be the optimal transport plan between the optimal transport plans q and q^k. We have that*

$$\lim_{k\to\infty}\mathbb{E}_{t\sim\mathrm{U}(0,1),\boldsymbol{X}_0^k\overset{\text{iid}}{\sim}q_0,\boldsymbol{X}_1^k\overset{\text{iid}}{\sim}q_1,(x_0,x_1,x_0',x_1')\sim\tilde{q}^k}\left[\|u_t^*(tx_1+(1-t)x_0)-u_t^k(tx_1'+(1-t)x_0')\|^2\right]=0 \tag{A.42}$$

Proof. For given instances of $\boldsymbol{X}_0^k$ and $\boldsymbol{X}_1^k$, we can write

$$\begin{aligned}
&\mathbb{E}_{t\sim\mathrm{U}(0,1),(x_0,x_1,x_0',x_1')\sim\tilde{q}^k}\left[\|u_t^*(tx_1+(1-t)x_0)-u_t^k(tx_1'+(1-t)x_0')\|^2\right]\\
&=\mathbb{E}_{t\sim\mathrm{U}(0,1),(x_0,x_1,x_0',x_1')\sim\tilde{q}^k}\Big[\|\mathbb{E}_{\tilde{x}_0,\tilde{x}_1\sim q}[\tilde{x}_1-\tilde{x}_0|tx_1+(1-t)x_0=t\tilde{x}_1+(1-t)\tilde{x}_0]\\
&\qquad\qquad-\mathbb{E}_{\tilde{x}_0',\tilde{x}_1'\sim q^k}[\tilde{x}_1'-\tilde{x}_0'|tx_1'+(1-t)x_0'=t\tilde{x}_1'+(1-t)\tilde{x}_0']\|^2\Big]\\
&\leq\mathbb{E}_{(x_0,x_1,x_0',x_1')\sim\tilde{q}^k}\left[\left\|x_1-x_0-(x_1'-x_0')\right\|^2\right]\leq 2\mathbb{E}_{(x_0,x_1,x_0',x_1')\sim\tilde{q}^k}\left[\left\|x_1-x_1'\right\|^2+\left\|x_0-x_0'\right\|^2\right]\\
&=2\mathbb{E}_{(x_0,x_1,x_0',x_1')\sim\tilde{q}^k}\left[\left\|(x_0,x_1)-(x_0',x_1')\right\|^2\right]=2W_2^2(q,q^k)
\end{aligned} \tag{A.43}$$

Assumption **(A3)** implies that almost surely, q^k converges to q weakly. For distributions on a bounded domain, weak convergence is equivalent to convergence in the Wasserstein distance [Vil08, Thm. 6.8], and this means that $W_2^2(q,q^k)\xrightarrow{k\to\infty}0$ almost surely. Almost sure convergence implies convergence in probability, which means that

$$\Pr(W_2^2(q,q^k)>\epsilon)\xrightarrow{k\to\infty}0,\quad\forall\epsilon>0. \tag{A.44}$$

Note that $W_2^2(q,q^k)$ is a bounded random variable because q and q^k have bounded support as q_0,q_1,q_0^k and q_1^k have bounded support. Suppose that $W_2^2(q,q^k)$ is bounded by the constant C. Hence, we can write

$$\mathbb{E}_{\boldsymbol{X}_0^k,\boldsymbol{X}_1^k}\mathbb{E}_{t\sim\mathrm{U}(0,1),(x_0,x_1,x_0',x_1')\sim\tilde{q}^k}\left[\|u_t^*(tx_1+(1-t)x_0)-u_t^k(tx_1'+(1-t)x_0')\|^2\right] \tag{A.45}$$

$$\leq 2\mathbb{E}_{\boldsymbol{X}_0^k,\boldsymbol{X}_1^k}W_2^2(q,q^k)\leq 2\big(\epsilon+C\Pr(W_2^2(q,q^k)>\epsilon)\big). \tag{A.46}$$

We can take ϵ arbitrarily small, and for a given ϵ we can make the second term in the right-hand side arbitrarily small by taking k large enough. The final result follows. □

Proof of Proposition 2

We have that almost surely, the empirical distributions q_0^k, resp. q_1^k, converge weakly to q_0, resp. q_1 [Var58]. Hence, we can apply Theorem 10. Since convergence in distribution of random variables is equivalent to weak convergence of their laws, and the law of an optimal

coupling is the optimal transport plan, we conclude that $(q_k)_{k\geq 0}$ converges weakly to q^*.

Theorem 10 ([CMT97], Theorem 3.2)**.** *Let $(P_n)_n$, $(Q_n)_n$, P, Q be probability measures in $\mathcal{P}_2$ (the space of Borel probability measures with bounded second order moment) such that $P \ll \lambda_p$ (P is absolutely continuous with respect to the Lebesgue measure) and $P_n \xrightarrow{w} P$, $Q_n \xrightarrow{w} Q$, where $\xrightarrow{w}$ denotes weak convergence of probability measures. Let (X_n, Y_n) be an optimal coupling between P_n and Q_n, $n \in \mathbb{N}$, and (X, Y) an optimal coupling between P and Q. Then, $(X_n, Y_n) \xrightarrow{\mathcal{L}} (X, Y)$, where $\xrightarrow{\mathcal{L}}$ denotes convergence of random variables in distribution.*

A.3.5 Bounds on the transport cost and monotone convergence results

The following result shows that for an arbitrary joint distribution $q(x_0, x_1)$, we can upper-bound the transport cost associated to the marginal vector field u_t to a quantity that depends only $q(x_0, x_1)$.

Proposition 3. *For an arbitrary joint distribution $q(x_0, x_1)$ with marginals $q_0(x_0)$ and $q_1(x_1)$, let ψ_t be the flow corresponding to the marginal vector field u_t. We have that*

$$\mathbb{E}_{q_0(x_0)}\|\psi_1(x_0) - x_0\|^2 \leq \mathbb{E}_{q(x_0,x_1)}\|x_1 - x_0\|^2, \tag{A.47}$$

Proof. We make use of the notation introduced in App. A.3.4. We will rely on the fact that for all $t \in [0, 1]$, the random variable $tx_1 + (1-t)x_0$, with $(x_0, x_1) \sim q$ has the same distribution as the random variable $\psi_t(x_0)$, with $x_0 \sim q_0$. This is a direct consequence of Lemma 1. Using that ψ_t is the flow for u_t and Jensen's inequality twice, we have that

$$\begin{aligned}
&\mathbb{E}_{x_0\sim q_0}\|\psi_1(x_0) - x_0\|^2 \\
&= \mathbb{E}_{x_0\sim q_0}\left\| \int_0^1 u_s(\psi_s(x_0))\, ds \right\|^2 \leq \mathbb{E}_{t\sim \mathrm{U}(0,1), x_0\sim q_0}\|u_t(\psi_t(x_0))\|^2 \\
&= \mathbb{E}_{t\sim \mathrm{U}(0,1),(x_0,x_1)\sim q}\|u_t(tx_1 + (1-t)x_0)\|^2 \\
&= \mathbb{E}_{t\sim \mathrm{U}(0,1),(x_0,x_1)\sim q}\|\mathbb{E}_{(x_0',x_1')\sim q}\left[u_t(tx_1 + (1-t)x_0 | x_0', x_1')| tx_1 + (1-t)x_0 = tx_1' + (1-t)x_0'\right]\|^2 \\
&\leq \mathbb{E}_{t\sim \mathrm{U}(0,1),(x_0,x_1)\sim q}\mathbb{E}_{(x_0',x_1')\sim q}\left[\|u_t(tx_1 + (1-t)x_0 | x_0', x_1')\|^2 | tx_1 + (1-t)x_0 = tx_1' + (1-t)x_0'\right] \\
&= \mathbb{E}_{t\sim \mathrm{U}(0,1),(x_0,x_1)\sim q}\mathbb{E}_{(x_0',x_1')\sim q}\left[\|x_1' - x_0'\|^2 | tx_1 + (1-t)x_0 = tx_1' + (1-t)x_0'\right] \\
&= \mathbb{E}_{t\sim \mathrm{U}(0,1),(x_0,x_1)\sim q}\|x_1 - x_0\|^2
\end{aligned} \tag{A.48}$$

as needed. □

Note that that the statement and proof of this proposition is equivalent to Theorem 3.5 of [LGL23], although the language and notation that we use is different, which is why we though convenient to include it.

For the case of BatchOT, the following theorem shows that the quantity in the upper bound of equation A.47 is monotonically decreasing in k. The combination of Proposition 3 and Theorem 11 provides a weak guarantee that for BatchOT, the transport cost should not get much higher when k increases.

Theorem 11. *Suppose that Multisample Flow Matching is run with BatchOT. For clarity, we make the dependency on the sample size k explicit and let*[1] $q^{(k)}(x_0, x_1) := q(x_0, x_1)$, *and* $\psi_t^k(x_0) := \psi_t(x_0)$. *Then, for any $k \geq 1$, we have that*

$$\begin{aligned}
&\mathbb{E}_{q_0(x_0)}\|\psi_1^k(x_0) - x_0\|^2 \leq \mathbb{E}_{q^{(k)}(x_0,x_1)}\|x_1 - x_0\|^2, \\
&\mathbb{E}_{q^{(k+1)}(x_0,x_1)}\|x_1 - x_0\|^2 \leq \mathbb{E}_{q^{(k)}(x_0,x_1)}\|x_1 - x_0\|^2.
\end{aligned} \tag{A.49}$$

Proof. We write

$$\begin{aligned}
&\mathbb{E}_{t\sim\mathrm{U}(0,1),\boldsymbol{X}_0^{k+1},\boldsymbol{X}_1^{k+1}}\mathbb{E}_{(x_0,x_1)\sim q^{k+1}}\|x_1 - x_0\|^2 = \frac{1}{k}\mathbb{E}_{t\sim\mathrm{U}(0,1),\boldsymbol{X}_0^{k+1},\boldsymbol{X}_1^{k+1}}\left[\sum_{i=1}^{k}\|x_1^{(i)} - x_0^{(\sigma_{k+1}(i))}\|^2\right] \\
&= \frac{1}{k}\frac{1}{k+1}\mathbb{E}_{t\sim\mathrm{U}(0,1),\boldsymbol{X}_0^{k+1},\boldsymbol{X}_1^{k+1}}\left[\sum_{j=1}^{k+1}\sum_{i\in[k+1]\setminus\{j\}}\|x_1^{(i)} - x_0^{(\sigma_{k+1}(i))}\|^2\right] \\
&\leq \frac{1}{k}\frac{1}{k+1}\mathbb{E}_{t\sim\mathrm{U}(0,1),\boldsymbol{X}_0^{k+1},\boldsymbol{X}_1^{k+1}}\left[\sum_{j=1}^{k+1}\sum_{i\in[k+1]\setminus\{j\}}\|x_1^{(i)} - x_0^{(\sigma_k^{-j}(i))}\|^2\right] \\
&= \mathbb{E}_{t\sim\mathrm{U}(0,1),\boldsymbol{X}_0^{k},\boldsymbol{X}_1^{k}}\left[\frac{1}{k}\sum_{j=1}^{k}\|x_1^{(i)} - x_0^{(\sigma_k(i))}\|^2\right] = \mathbb{E}_{t\sim\mathrm{U}(0,1),\boldsymbol{X}_0^{k},\boldsymbol{X}_1^{k}}\mathbb{E}_{(x_0,x_1)\sim q^{k}}\|x_1 - x_0\|^2.
\end{aligned} \tag{A.50}$$

In the first equality, we used that the optimal transport map between the empirical distributions q_0^k and q_1^k can be encoded as a permutation, which we denote by σ_{k+1}. In the inequality, we introduced the notation σ_k^{-j} to denote the optimal permutation within $\{x_0^{(i)}\}_{i\in[k+1]\setminus\{j\}}$. The inequality holds because using the optimality of σ_{k+1}:

$$\begin{aligned}
&\sum_{j=1}^{k+1}\sum_{i\in[k+1]\setminus\{j\}}\|x_1^{(i)} - x_0^{(\sigma_{k+1}(i))}\|^2 \leq \sum_{j=1}^{k+1}\sum_{i\in[k+1]}\|x_1^{(i)} - x_0^{(\sigma_{k+1}(i))}\|^2 \\
&\leq \sum_{j=1}^{k+1}\left(\sum_{i\in[k+1]\setminus\{j\}}\|x_1^{(i)} - x_0^{(\sigma_k^{-j}(i))}\|^2 + \|x_1^{(j)} - x_0^{(j)}\|^2\right) \leq \sum_{j=1}^{k+1}\sum_{i\in[k+1]\setminus\{j\}}\|x_1^{(i)} - x_0^{(\sigma_k^{-j}(i))}\|^2.
\end{aligned} \tag{A.51}$$

□

[1] Note that here $q^{(k)} := q$ is a marginalized distribution and is different from q^k defined in Step 3.

A.4 Proofs of Chapter 6

A.4.1 Proof of Proposition 1

We recall a general affine Gaussian path is defined by

$$p_t(x|x_1) = \mathcal{N}(x|\alpha_t x_1, \sigma_t^2 I), \qquad \text{conditional probability path} \tag{A.52}$$

$$p_t(x) = \int p_t(x|x_1)q(x_1)dx_1, \qquad \text{marginal probability path} \tag{A.53}$$

where (α_t, σ_t) define the scheduler and q is the dataset probability density. The velocity fields defining these paths are [Lip+23]:

$$u_t(x|x_1) = a_t x + b_t x_1, \qquad a_t = \frac{\dot{\sigma}_t}{\sigma_t}, \quad b_t = \dot{\alpha}_t - \alpha_t \frac{\dot{\sigma}_t}{\sigma_t} \qquad \text{conditional velocity field} \tag{A.54}$$

$$u_t(x) = \int u_t(x|x_1)p_t(x_1|x)dx_1, \qquad p_t(x_1|x) = \frac{p_t(x|x_1)q(x_1)}{p_t(x)} \qquad \text{marginal velocity field} \tag{A.55}$$

The differential of the denoiser is then:

$$D_x\hat{x}_{1|t}(x) = D_x \int x_1 p_t(x_1|x)dx_1 = \int x_1 \nabla_x p_t(x_1|x)dx_1 \tag{A.56}$$

Since $p_t(x|x_1)$ is a Gaussian:

$$\nabla_x p_t(x|x_1) = \frac{\alpha_t x_1 - x}{\sigma_t^2} p_t(x|x_1) \tag{A.57}$$

and plugging into A.53, we have:

$$\nabla_x p_t(x) = \int \frac{\alpha_t x_1 - x}{\sigma_t^2} p_t(x|x_1)q(x_1)dx_1 \tag{A.58}$$

using A.55, we get:

$$\nabla_x p_t(x_1|x) = p_t(x_1|x)\frac{\alpha_t}{\sigma_t^2}\left(x_1 - \hat{x}_{1|t}(x)\right) \tag{A.59}$$

therefore, A.56 takes the form:

$$\begin{aligned} D_x\hat{x}_{1|t}(x) &= \int \frac{\alpha_t}{\sigma_t^2} p_t(x_1|x) x_1 (x_1 - \hat{x}_{1|t}(x))^T dx_1 = \\ &= \int \frac{\alpha_t}{\sigma_t^2} p_t(x_1|x)(x_1 - \hat{x}_{1|t}(x))(x_1 - \hat{x}_{1|t}(x))^T dx_1 = \frac{\alpha_t}{\sigma_t^2}\mathrm{Var}_{1|t}(x) \end{aligned} \tag{A.60}$$

□

A.4.2 Proof of Theorem 7

To compute the differential of $x(1)$ w.r.t the initial point x_0 we utilize adjoint dynamics.

Let us define the adjoint $p(t) = D_{x(t)}x(1)$. The dynamics of $p(t)$ are defined by the following ODE [Eva05]:

$$\dot{p}(t) = -D_x u_t(x(t))^T p(t) \tag{A.61}$$
$$p(1) = D_{x(1)}x(1) = I. \tag{A.62}$$

To compute $D_{x_0}x(1)$ we solve A.61 from time $t = 1$ back to time $t = 0$. Then,

$$p(0) = D_{x_0}x(1). \tag{A.63}$$

First, we will use the properties of AGPPs to further analyze the differential of the velocity field, $D_x u_t(x(t))$, that defines the adjoint dynamics in equation A.61.

We write the velocity field in terms of the denoiser by plugging A.54 into A.55:

$$u_t(x) = a_t x + b_t \hat{x}_{1|t}(x) \tag{A.64}$$

and using equation A.60, the differential of the velocity field is:

$$D_x u_t(x) = a_t I + b_t D_x \hat{x}_{1|t}(x) = a_t I + b_t \frac{\alpha_t}{\sigma_t^2}\mathrm{Var}_{1|t}(x) \tag{A.65}$$

The adjoint dynamics are then given by:

$$\dot{p}(t) = A(t)p(t) \tag{A.66}$$
$$p(1) = D_{x(1)}x(1) = I. \tag{A.67}$$

where

$$A(t) = -\left(a_t I + \gamma_t \mathrm{Var}_{1|t}(x)\right)^T. \tag{A.68}$$

and we define:

$$\gamma_t = b_t \frac{\alpha_t}{\sigma_t^2} = \left(\dot{\alpha}_t - \alpha_t \frac{\dot{\sigma}_t}{\sigma_t}\right)\frac{\alpha_t}{\sigma_t^2} = \frac{\dot{\alpha}_t \alpha_t \sigma_t^2 - \alpha_t^2 \dot{\sigma}_t \sigma_t}{\sigma_t^4} = \frac{1}{2}\frac{d}{dt}\left(\frac{\alpha_t^2}{\sigma_t^2}\right) = \frac{1}{2}\frac{d}{dt}\mathrm{snr}(t) \tag{A.69}$$

The adjoint ODE A.66 is a non-autonomous linear ODE and together with the initial condition equation A.67 its solution is given by [SN20]:

$$p(t) = \mathcal{T}\exp\left[-\int_t^1 A(s)ds\right]p(1) \tag{A.70}$$

known as the time-ordered exponential, defined as follows:

$$\mathcal{T}\exp\left[\int_t^1 A(s)ds\right] = \sum_{n=1}^{\infty}\frac{(-1)^n}{n!}\int_t^1\cdots\int_t^1 \mathcal{T}\{A(s_1)A(s_2)\dots A(s_n)\}ds_1ds_2\dots ds_n \quad \text{(A.71)}$$

where $\mathcal{T}\{A(s_1)A(s_2)\dots A(s_n)\}$ orders the product of matrices such that the value of s_i decreases from right to left. Alternative equivalent solutions to this differential equation are the dyson series [SN20] and the Magnus expansion [Mag54].

When the matrices $\{A(s)\}_{s\in[1,t]}$ commute, i.e., $[A(s), A(s')] = A(s)A(s') - A(s')A(s) = 0$, the time-ordered exponential A.71 reduces to the first term in the sum for $n = 1$ in A.71. We can therefore separate the first term in A.68, since I commutes with every matrix, and arrive at a simplified solution for $t = 0$:

$$D_{x_0}x(1) = \sigma_1\mathcal{T}\exp\left[\int_0^1 \gamma_t \mathrm{Var}_{1|t}(x)dt\right], \quad \text{(A.72)}$$

where $\exp\left[\int_0^1 a_t dt\right] = \sigma_1$ under the assumption that a_t is integrable, concluding the proof. □

Next we show that the integral in equation 6.18 is defined also for $\sigma_1 = 0$.

Lemma 7. *For a Lipschitz function $f : \mathbb{R}^d \to \mathbb{R}$ we have that $\int \mathcal{N}(x|y, \sigma^2 I)f(x)dx = f(y) + \mathcal{O}(\sigma)$.*

Proof.

$$\begin{aligned}\left|\int \mathcal{N}(x|y,\sigma^2 I)f(x)dx - f(y)\right| &\le \int \mathcal{N}(x|y,\sigma^2 I))\,|f(x) - f(y)|\,dx \\ &= \int \mathcal{N}(z|0,I)\,|f(\sigma z + y) - f(y)|\,dz \\ &\le K\sigma\int \mathcal{N}(z|0,I)\,|z|\,dz \\ &= \mathcal{O}(\sigma),\end{aligned}$$

where in the first equality we performed a change of variable $z = \frac{x-y}{\sigma}$, and in the second inequality we used the fact that f is Lipschitz with constant $K > 0$. □

Using this Lemma we prove (under the assumption that $p_1(x)$ and its derivatives is Lipschitz):

Proposition 4. *The denoiser asymptotics at $t \to 1$ is*

$$\hat{x}_{1|t}(x) = \frac{x}{\alpha_t} + \mathcal{O}(\sigma_t) \quad \text{(A.73)}$$

Proof. First we note that we assume $\sigma_t \to 0$ and $\alpha_t \to 1$ as $t \to 1$,

$$\mathcal{N}(x|\alpha_t x_1, \sigma_t^2 I) = c_t\mathcal{N}\left(x_1\middle|\frac{x}{\alpha_t}, \left(\frac{\sigma_t}{\alpha_t}\right)^2 I\right), \quad \text{(A.74)}$$

where c_t is some normalization constant such that $c_1 = 1$. Now,

$$p_t(x) = \int \mathcal{N}(x|\alpha_t x_1, \sigma_t^2 I) p_1(x_1) dx_1 \tag{A.75}$$

$$= c_t \int \mathcal{N}\left(x_1 \middle| \frac{x}{\alpha_t}, \left(\frac{\sigma_t}{\alpha_t}\right)^2 I\right) p_1(x_1) dx_1 \tag{A.76}$$

$$= c_t p_1\left(\frac{x}{\alpha_t}\right) + \mathcal{O}(\sigma_t), \tag{A.77}$$

where in second equality we used equation A.74 and the last equality Lemma 7.

$$\hat{x}_{1|t}(x) = \int x_1 p_t(x_1|x) dx_1 \tag{A.78}$$

$$= \int x_1 \frac{\mathcal{N}(x|\alpha_t x_1, \sigma_t^2 I) p_1(x_1)}{p_t(x)} dx_1 \tag{A.79}$$

$$= \int x_1 \frac{c_t \mathcal{N}\left(x_1 \middle| \frac{x}{\alpha_t}, \left(\frac{\sigma_t}{\alpha_t}\right)^2 I\right) p_1(x_1)}{p_t(x)} dx_1 \tag{A.80}$$

$$= \frac{c_t \frac{x}{\alpha_t} p_1\left(\frac{x}{\alpha_t}\right) + \mathcal{O}(\sigma_t)}{c_t p_1\left(\frac{x}{\alpha_t}\right) + \mathcal{O}(\sigma_t)} \tag{A.81}$$

$$= \frac{x}{\alpha_t} + \mathcal{O}(\sigma_t), \tag{A.82}$$

where in the second equality we used the definition of $p_t(x_1|x)$, in the third equality we used equation A.74, and in the fourth equality we used Lemma 7. □

Now we can show that $D_x u_t(x(t))$ is bounded as $t \to 1$

$$D_x u_t(x(t)) = a_t I + b_t D_x \hat{x}_{1|t}(x) \tag{A.83}$$

$$= a_t I + b_t \left(\frac{1}{a_t} I + \mathcal{O}(\sigma_t)\right) \tag{A.84}$$

$$= \frac{\dot{\alpha}_t}{\alpha_t} I + \mathcal{O}(1), \tag{A.85}$$

where in the first equality we used equation A.65, in the second Proposition 4 (and the fact that the derivatives of p_1 are Lipschitz for the derivation of the asymptotic rule), and in the last equality equation A.54. Furthermore $D_x u_t(x(t))$ is bounded as $t \to 0$ as both a_0, b_0 are well defined. This means that $D_x u_t(x(t))$ is integrable over $[0, 1]$.

A.4.3 Discrete Time Analysis

Theorem 7 analyzes the continuous time case, providing intuition about the behavior of the dynamics of $x(1)$ when changing the initial condition x_0. The final expression 6.18, however, involves a time-ordered exponential which may be hard to interpret. Furthermore, our experiments show that even with a small number of discrete steps, differentiating $x(1)$ with respect to x_0 yields meaningful gradients, performing well in practice (see B.4.1).

Let us consider Euler solver with N uniform steps of size $h = \frac{1}{N}$, with initial point x_0. An intermediate point at time mh, x_{mh} is given by:

$$x_{(m+1)h} = x_{mh} + hu_{mh}(x_{mh}) \tag{A.86}$$

We are interested at the derivative of $x_{Nh} = x(1)$ w.r.t x_0.

By the chain rule and equation A.65, one can write:

$$D_{x_0}x_1 = \prod_{m=0}^{N-1} D_{x_{mh}}x_{(m+1)h} = \prod_{m=0}^{N-1} \Big((1 + ha_{mh})I + h\gamma_{mh}\mathrm{Var}_{1|mh}(x_{mh})\Big) \tag{A.87}$$

note that this is also a time-ordered product, with m decreasing from left to right.

For the CondOT probability path, where $\alpha_t = t, \sigma_t = 1 - t$, A.87 takes the form:

$$D_{x_0}x_1 = \prod_{m=0}^{N-1} \Big(\frac{1 - (m+1)h}{1 - mh} I + \frac{mh^2}{(1 - mh)^3}\mathrm{Var}_{1|mh}(x_{mh})\Big) \tag{A.88}$$

A.4.4 On Flow-Matching, Denoisers and Noise Prediction

Consider a general affine conditional probability path defined by the following transport map:

$$x_t = \sigma_t x_0 + \alpha_t x_1$$

where $x_0 \sim p_0$ and $x_1 \sim p_1$.

For different choices of σ_t, α_t we can parametrize known diffusion and flow-matching paths. The corresponding conditional vector field on x_1 is:

$$u_t(x|x_1) = \frac{\dot\sigma_t}{\sigma_t}(x - \alpha_t x_1) + \dot\alpha_t x_1 = \frac{\dot\sigma_t}{\sigma_t}x - \left(\frac{\dot\sigma_t \alpha_t}{\sigma_t} - \dot\alpha_t\right) x_1$$

and the conditional vector field on x_0 is:

$$u_t(x|x_0) = \dot\sigma_t x_0 + \frac{\dot\alpha_t}{\alpha_t}(x - \sigma_t x_0) = \frac{\dot\alpha_t}{\alpha_t}x - \left(\frac{\dot\alpha_t \sigma_t}{\alpha_t} - \dot\sigma_t\right) x_0$$

where $\dot f = \frac{d}{dt}f$.

Consider the marginal velocity field:

$$u_t(x) = \int u_t(x|x_1)p_t(x_1|x)dx_1 = \int u_t(x|x_0)p_t(x_0|x)dx_0$$

One can express it in terms of the optimal *denoiser* function, $\hat x_{1|t}(x)$:

$$u_t(x) = \frac{\dot{\sigma}_t}{\sigma_t} \int x p_t(x_1|x) dx_1 - \left(\frac{\dot{\sigma}_t \alpha_t}{\sigma_t} - \dot{\alpha}_t \right) \int x_1 p_t(x_1|x) dx_1 = \frac{\dot{\sigma}_t}{\sigma_t} x - \left(\frac{\dot{\sigma}_t \alpha_t}{\sigma_t} - \dot{\alpha}_t \right) \hat{x}_{1|t}(x) \tag{A.89}$$

For Cond-OT:

$$u_t(x) = \frac{\hat{x}_{1|t}(x) - x}{1 - t} \tag{A.90}$$

Or, in terms of the optimal noise predictor, $\epsilon_t(x)$, like in DDPM:

$$u_t(x) = \frac{\dot{\alpha}_t}{\alpha_t} \int x p_t(x_0|x) dx_0 - \left(\frac{\dot{\alpha}_t \sigma_t}{\alpha_t} - \dot{\sigma}_t \right) \int x_0 p_t(x_0|x) dx_0 = \frac{\dot{\alpha}_t}{\alpha_t} x - \left(\frac{\dot{\alpha}_t \sigma_t}{\alpha_t} - \dot{\sigma}_t \right) \epsilon_t(x)$$

and for Cond-OT:

$$u_t(x) = \frac{x - \epsilon_t(x)}{t} \tag{A.91}$$

Appendix B

Additional Details and Figures

B.1 Additions for Chapter 3

B.1.1 Velocity Field Representation

We describe how we represent the parametric part of our system, namely the velocity field $v_\theta \in \mathfrak{P}(\mathcal{M})$, for the different manifold types we consider: Euclidean space, spheres and product manifolds. In the *Euclidean* case we use an MLP $v_\theta : \mathbb{R}^{d+1} \to \mathbb{R}^d$, where d is the dimension of $\mathcal{M}$. We use standard Euclidean inner product, gradient ∇, and divergence $\mathrm{div} = \nabla\cdot$ for computing the PPD (equation 3.6), where the path p defined in equation 3.12. In the *sphere* case $\mathcal{S}^d \subset \mathbb{R}^{d+1}$, similarly to [Roz+21] we define v to be constant in the normal direction to the sphere and produce tangent vectors via the tangent projection operator

$$v_\theta(t, x) = \left(\boldsymbol{I} - \frac{xx^T}{\|x\|^2}\right) w_\theta\left(t, \frac{x}{\|x\|}\right), \tag{B.1}$$

where $w_\theta : \mathbb{R}^{d+2} \to \mathbb{R}^{d+1}$ is an MLP. The inner product on the sphere is the induced Euclidean one, i.e., for $v, u \in T_x\mathcal{S}^d$ we have $\langle u, v\rangle = u^T v$. For v_θ defined in equation B.1, the Riemannian gradient and divergence coincide with the Euclidean gradient ∇ and divergence $\nabla\cdot$. p is defined as in equation 3.15. For notational simplicity we explain the *product manifold* implementation for two manifolds $\mathcal{M} = \mathbb{R}^{d_1} \times \mathcal{S}^{d_2}$, where the extension to product of N manifolds is similar. The tangent velocity field is a function of the form $v_\theta : \mathbb{R}^{d_1+1} \times \mathbb{R}^{d_2+2} \to \mathbb{R}^{d_1} \times \mathbb{R}^{d_2+1}$. For $(t_1, x_1, t_2, x_2) \in \mathbb{R}^{d_1+1} \times \mathbb{R}^{d_2+2}$ we let

$$v_\theta(t_1, x_1, t_2, x_2) = \begin{bmatrix} v_1\left(t_1, x_1, t_2, \frac{x_2}{\|x_2\|}\right) \\ \left(\boldsymbol{I} - \frac{x_2 x_2^T}{\|x_2\|^2}\right) v_2\left(t_1, x_1, t_2, \frac{x_2}{\|x_2\|}\right) \end{bmatrix}$$

where v_1, v_2 are MLPs. The inner product $(v_1, v_2), (u_1, u_2) \in T_x\mathcal{M}$ is defined by $\langle v_1, u_1\rangle + \langle v_2, u_2\rangle$; the gradient as $\nabla = (\nabla_1, \nabla_2)^T$, where ∇_1 is the Euclidean gradient w.r.t. x_1, and ∇_2 is the Euclidean gradient w.r.t. x_2; the divergence $\mathrm{div}(v) = \nabla_1 \cdot v_1 + \nabla_2 \cdot v_2$. Lastly, p is defined as in equation 3.10 with kernel equation 3.16.

B.1.2 Numerically Stable Derivative of the Normalizing Constant of vMF

The log of the normalizing constant of the vMF has the form

$$\log C_p(\kappa) = \left(\frac{p}{2} - 1\right) \log \kappa - \frac{p}{2} \log(2\pi) - \left[\kappa + \log \texttt{ive}(\frac{p}{2} - 1, \kappa)\right]$$

where $\texttt{ive}(\nu, \kappa) = \texttt{iv}(\nu, \kappa) \exp(-\kappa)$. Now, the $\log \texttt{ive}(\frac{p}{2} - 1, \kappa)$ is stable but its derivative is not. Therefore we will define a new function and its derivative: $\texttt{logive}(\nu, \kappa)$. Its forward will be defined by;

$$\texttt{logive}(\nu, \kappa) = \log\left(\texttt{ive}(\nu, \kappa)\right),$$

and for its derivative we first note:

$$\begin{aligned}\partial_\kappa \log \texttt{ive}(\nu, \kappa) &= \frac{\partial_\kappa \texttt{ive}(\nu, \kappa)}{\texttt{ive}(\nu, \kappa)} = \frac{\texttt{ive}(\nu - 1, \kappa) - \texttt{ive}(\nu, \kappa)\left[\frac{\nu+\kappa}{\kappa}\right]}{\texttt{ive}(\nu, \kappa)} \\ &= \frac{\texttt{ive}(\nu - 1, \kappa)}{\texttt{ive}(\nu, \kappa)} - \left[\frac{\nu + \kappa}{\kappa}\right] = \frac{\texttt{ive}(\nu - 1, \kappa)}{\texttt{ive}(\nu, \kappa)} - \frac{\nu}{\kappa} - 1\end{aligned}$$

For high dimensions the $\texttt{ive}$ ratio is numerically unstable and several approximations have been suggested. In particular [RS16] suggest the following lower and upper bounds:

$$\frac{\nu - \frac{1}{2} + \sqrt{(\nu + \frac{1}{2})^2 + \kappa^2}}{\kappa} > \frac{\texttt{ive}(\nu - 1, \kappa)}{\texttt{ive}(\nu, \kappa)} > \frac{\nu - 1 + \sqrt{(\nu + 1)^2 + \kappa^2}}{\kappa}$$

Similar to [OAP19] we take the average of the higher and lower bound (see [OAP19] for empirically demonstrating the quality of this approximation):

$$\partial_\kappa \log \texttt{ive}(\nu, \kappa) = \frac{\texttt{ive}(\nu - 1, \kappa)}{\texttt{ive}(\nu, \kappa)} - \frac{\nu}{\kappa} - 1 \approx \frac{-1.5 + \sqrt{(\nu + 1)^2 + \kappa^2} + \sqrt{(\nu + \frac{1}{1})^2 + \kappa^2}}{2\kappa} - 1$$

and this is defined as the derivative of $\texttt{logive}$.

B.1.3 Experimental Details

Toy densities on $\mathbb{R}^2$ and $\mathcal{S}^2$

For the $\mathbb{R}^2$ datasets we used a 3 layer MLP with hidden dimension 256. We trained with Adam optimizer with learning rate $1e-4$, batch size 1000, $\sigma_1 = 0.01$ and $\ell = 1$. The searched parameters across learning rates are $\{1e-3, 5e-4, 1e-4\}$ and $\sigma_1 \in \{0.005, 0.01, 0.05\}$. For the $\mathcal{S}^2$ datasets we used a 6 layer MLP with hidden dimension 512. We trained with Adam optimizer with learning rate $1e-4$, batch size 1000, $\kappa = 5000$ and $\ell = 1$.

Earth and climate datasets

For the earth and climate datasets we used a 6 layer MLP with hidden dimension 512. We trained with Adam optimizer with learning rate $1e-4$, batch size 1000, $\kappa = 55K$ and

$\ell = 2$. The searched parameters across κ are $\{5K, 55K, 500k\}$.

Higher dimensional spheres

We ran experiments on $\mathcal{S}^{15}$ for different $k = 2, 3, 4$ values. The architecture used was a 3 layer MLP with hidden dimension 64, Adam optimizer with learning rate $1e-3$, batch size 7000, $\kappa = 5K$ and $\ell = 2$. We searched over learning rates $\{1e-3, 1e-4, 1e-5\}$.

The S-FFJORD baseline is as described in the paper. We used the architecture used for the 2D toy experiments in the FFJORD paper, as published in the official FFJORD code repository. We run both PPM and S-FFJORD with approximate divergence computation using the Hutchinson estimator.

Product of manifolds - Robotics

For the robotics datasets we used a 6 layer MLP with hidden dimension 512. We trained with Adam optimizer with learning rate $1e-4$, batch size 1000, $\kappa = 55K$ and $\ell = 1$. The searched parameters across κ are $\{5K, 55K\}$.

B.2 Additions for Chapter 4

B.2.1 Diffusion Conditional Velocity Fields

We derive the velocity field governing the Probability Flow ODE (equation 13 in [Son+21b]) for the VE and VP diffusion paths (equation 4.14) and note that it coincides with the conditional velocity fields we derive using Theorem 5, namely the velocity fields defined in equations 4.12 and 4.15.

We start with a short primer on how to find a conditional velocity field for the probability path described by the Fokker-Planck equation, then instantiate it for the VE and VP probability paths.

Since in the diffusion literature the diffusion process runs from data at time $t = 0$ to noise at time $t = 1$, we will need the following lemma to translate the diffusion VFs to our convention of $t = 0$ corresponds to noise and $t = 1$ corresponds to data:

Lemma 8. *Consider a flow defined by a velocity field $u_t(x)$ generating probability density path $p_t(x)$. Then, the velocity field $\tilde{u}_t(x) = -u_{1-t}(x)$ generates the path $\tilde{p}_t(x) = p_{1-t}(x)$ when initiated from $\tilde{p}_0(x) = p_1(x)$.*

Proof. We use the continuity equation (equation 2.28):

$$\begin{aligned}\frac{d}{dt}\tilde{p}_t(x) = \frac{d}{dt}p_{1-t}(x) &= -p'_{1-t}(x) \\ &= \operatorname{div}(p_{1-t}(x)u_{1-t}(x)) \\ &= -\operatorname{div}(\tilde{p}_t(x)(-u_{1-t}(x)))\end{aligned}$$

and therefore $\tilde{u}_t(x) = -u_{1-t}(x)$ generates $\tilde{p}_t(x)$. □

Conditional VFs for Fokker-Planck probability paths Consider a Stochastic Differential Equation (SDE) of the standard form

$$dy = f_t dt + g_t dw \tag{B.2}$$

with time parameter t, drift f_t, diffusion coefficient g_t, and dw is the Wiener process. The solution y_t to the SDE is a stochastic process, i.e., a continuous time-dependent random variable, the probability density of which, $p_t(y_t)$, is characterized by the Fokker-Planck equation:

$$\frac{dp_t}{dt} = -\operatorname{div}(f_t p_t) + \frac{g_t^2}{2}\Delta p_t \tag{B.3}$$

where Δ represents the Laplace operator (in y), namely $\operatorname{div}\nabla$, where ∇ is the gradient operator (also in y). Rewriting this equation in the form of the continuity equation can be done as follows [MRO20b]:

$$\frac{dp_t}{dt} = -\operatorname{div}\Big(f_t p_t - \frac{g^2}{2}\frac{\nabla p_t}{p_t}p_t\Big) = -\operatorname{div}\Big((f_t - \frac{g_t^2}{2}\nabla \log p_t)p_t\Big) = -\operatorname{div}\Big(w_t p_t\Big)$$

where the velocity field

$$w_t = f_t - \frac{g_t^2}{2}\nabla \log p_t \tag{B.4}$$

satisfies the continuity equation with the probability path p_t, and therefore generates p_t.

Variance Exploding (VE) path The SDE for the VE path is

$$dy = \sqrt{\frac{d}{dt}\sigma_t^2}dw,$$

where $\sigma_0 = 0$ and increasing to infinity as $t \to 1$. The SDE is moving from data, y_0, at $t = 0$ to noise, y_1, at $t = 1$ with the probability path

$$p_t(y|y_0) = \mathcal{N}(y|y_0, \sigma_t^2 I).$$

The conditional VF according to equation B.4 is:

$$w_t(y|y_0) = \frac{\sigma_t'}{\sigma_t}(y - y_0)$$

Using Lemma 8 we get that the probability path

$$\tilde{p}_t(y|y_0) = \mathcal{N}(y|y_0, \sigma_{1-t}^2 I)$$

is generated by

$$\tilde{w}_t(y|y_0) = -\frac{\sigma_{1-t}'}{\sigma_{1-t}}(y - y_0),$$

which coincides with equation 4.13.

Variance Preserving (VP) path The SDE for the VP path is

$$dy = -\frac{T'(t)}{2}y + \sqrt{T'(t)}dw,$$

where $T(t) = \int_0^t \beta(s)ds$, $t \in [0, 1]$. The SDE coefficients are therefore

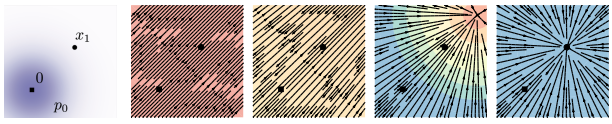
$$f_s(y) = -\frac{T'(s)}{2}y, \quad g_s = \sqrt{T'(s)}$$

and

$$p_t(y|y_0) = \mathcal{N}(y|e^{-\frac{1}{2}T(t)}y_0, (1 - e^{-T(t)})I).$$

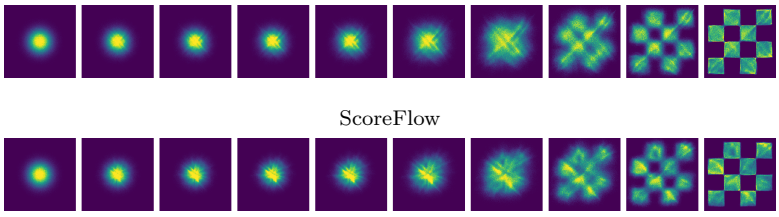
Plugging these choices in equation B.4 we get the conditional VF

$$w_t(y|y_0) = \frac{T'(t)}{2}\left(\frac{y - e^{-\frac{1}{2}T(t)}y_0}{1 - e^{-T(t)}} - y\right) \tag{B.5}$$



Diffusion path – conditional velocity field

Figure B.1: VP Diffusion path's conditional velocity field. Compare to Figure 4.2.



ScoreFlow

DDPM

Figure B.2: Trajectories of CNFs trained with ScoreFlow [Son+21a] and DDPM [HJA20] losses on 2D checkerboard data, using the same learning rate and other hyperparameters as Figure 4.4.

Using Lemma 8 to reverse the time we get the conditional VF for the reverse probability path:

$$\begin{aligned}\tilde{w}_t(y|y_0) &= -\frac{T'(1-t)}{2}\left(\frac{y - e^{-\frac{1}{2}T(1-t)}y_0}{1-e^{-T(1-t)}} - y\right) \\ &= -\frac{T'(1-t)}{2}\left[\frac{e^{-T(1-t)}y - e^{-\frac{1}{2}T(1-t)}y_0}{1-e^{-T(1-t)}}\right],\end{aligned}$$

which coincides with equation 4.15.

B.2.2 Implementation details

For the 2D example we used an MLP with 5-layers of 512 neurons each, while for images we used the UNet architecture from [DN21]. For images, we center crop images and resize to the appropriate dimension, whereas for the 32×32 and 64×64 resolutions we use the same pre-processing as [CLH17]. The three methods (FM-OT, FM-Diffusion, and SM-Diffusion) are always trained on the same architecture, same hyper-parameters, and for the same number of epochs.

Diffusion baselines

Losses. We consider three options as diffusion baselines that correspond to the most popular diffusion loss parametrizations [SE19; Son+21a; HJA20; Kin+21]. We will assume general Gaussian path form of equation 4.6, i.e.,

$$p_t(x|x_1) = \mathcal{N}(x|\mu_t(x_1), \sigma_t^2(x_1)I).$$

Score Matching loss is

$$\mathcal{L}_{\text{SM}}(\theta) = \mathbb{E}_{t,q(x_1),p_t(x|x_1)}\lambda(t)\left\|s_t(x) - \nabla \log p_t(x|x_1)\right\|^2 \tag{B.6}$$

$$= \mathbb{E}_{t,q(x_1),p_t(x|x_1)}\lambda(t)\left\|s_t(x) - \frac{x-\mu_t(x_1)}{\sigma_t^2(x_1)}\right\|^2. \tag{B.7}$$

Taking $\lambda(t) = \sigma_t^2(x_1)$ corresponds to the original Score Matching (SM) loss from [SE19], while considering $\lambda(t) = \beta(1-t)$ (β is defined below) corresponds to the Score Flow (SF) loss motivated by an NLL upper bound [Son+21a]; s_t is the learnable score function. DDPM (Noise Matching) loss from [HJA20] (equation 14) is

$$\mathcal{L}_{\text{NM}}(\theta) = \mathbb{E}_{t,q(x_1),p_t(x|x_1)}\left\|\epsilon_t(x) - \frac{x-\mu_t(x_1)}{\sigma_t(x_1)}\right\|^2 \tag{B.8}$$

$$= \mathbb{E}_{t,q(x_1),p_0(x_0)}\left\|\epsilon_t(\sigma_t(x_1)x_0 + \mu_t(x_1)) - x_0\right\|^2 \tag{B.9}$$

where $p_0(x) = \mathcal{N}(x|0,I)$ is the standard Gaussian, and ϵ_t is the learnable noise function.

Diffusion path. For the diffusion path we use the standard VP diffusion (equation 4.15), namely,

$$\mu_t(x_1) = \alpha_{1-t}x_1, \quad \sigma_t(x_1) = \sqrt{1-\alpha_{1-t}^2}, \quad \text{where } \alpha_t = e^{-\frac{1}{2}T(t)}, \quad T(t) = \int_0^t \beta(s)ds,$$

with, as suggested in [Son+21b], $\beta(s) = \beta_{\min} + s(\beta_{\max} - \beta_{\min})$ and consequently

$$T(s) = \int_0^s \beta(r)dr = s\beta_{\min} + \frac{1}{2}s^2(\beta_{\max} - \beta_{\min}),$$

where $\beta_{\min} = 0.1$, $\beta_{\max} = 20$ and time is sampled in $[0, 1-\epsilon]$, $\epsilon = 10^{-5}$ for training and likelihood and $\epsilon = 10^{-5}$ for sampling.

Sampling. Score matching samples are produced by solving the ODE (equation 2.24) with the velocity field

$$u_t(x) = -\frac{T'(1-t)}{2}\left[s_t(x) - x\right]. \tag{B.10}$$

DDPM samples are computed with equation B.10 after setting $s_t(x) = \epsilon_t(x)/\sigma_t$, where $\sigma_t = \sqrt{1-\alpha_{1-t}^2}$.

Training & evaluation details

We report the hyper-parameters used in Table B.1. We use full 32 bit-precision for training CIFAR10 and ImageNet-32 and 16-bit mixed precision for training ImageNet-64/128/256. All models are trained using the Adam optimizer with the following parameters: $\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay $= 0.0$, and $\epsilon = 1e{-}8$. All methods we trained (i.e., FM-OT, FM-Diffusion, SM-Diffusion) using identical architectures, with the same parameters for the the same number of Epochs (see Table B.1 for details). We use either a constant

	CIFAR10	ImageNet-32	ImageNet-64	ImageNet-128
Channels	256	256	192	256
Depth	2	3	3	3
Channels multiple	1,2,2,2	1,2,2,2	1,2,3,4	1,1,2,3,4
Heads	4	4	4	4
Heads Channels	64	64	64	64
Attention resolution	16	16,8	32,16,8	32,16,8
Dropout	0.0	0.0	0.0	0.0
Effective Batch size	256	1024	2048	1536
GPUs	2	4	16	32
Epochs	1000	200	250	571
Iterations	391k	250k	157k	500k
Learning Rate	5e-4	1e-4	1e-4	1e-4
Learning Rate Scheduler	Polynomial Decay	Polynomial Decay	Constant	Polynomial Decay
Warmup Steps	45k	20k	-	20k

Table B.1: Hyper-parameters used for training each model

learning rate schedule or a polynomial decay schedule (see Table B.1). The polynomial decay learning rate schedule includes a warm-up phase for a specified number of training steps. In the warm-up phase, the learning rate is linearly increased from $1e{-}8$ to the peak learning rate (specified in Table B.1). Once the peak learning rate is achieved, it linearly decays the learning rate down to $1e{-}8$ until the final training step.

When reporting negative log-likelihood, we dequantize using the standard uniform dequantization. We report an importance-weighted estimate using

$$\log \frac{1}{K} \sum_{k=1}^{K} p_t(x + u_k), \text{ where } u_k \sim \mathcal{U}(0, 1), \tag{B.11}$$

with x is in $\{0, \ldots, 255\}$ and solved at $t = 1$ with an adaptive step size solver `dopri5` with `atol=rtol=1e-5` using the `torchdiffeq` [Che18] library. Estimated values for different values of K are in Table B.2.

When computing FID/Inception scores for CIFAR10, ImageNet-32/64 we use the TensorFlow GAN library [1]. To remain comparable to [DN21] for ImageNet-128 we use the evaluation script they include in their publicly available code repository [2].

[1] https://github.com/tensorflow/gan
[2] https://github.com/openai/guided-diffusion

B.2.3 Additional tables and figures

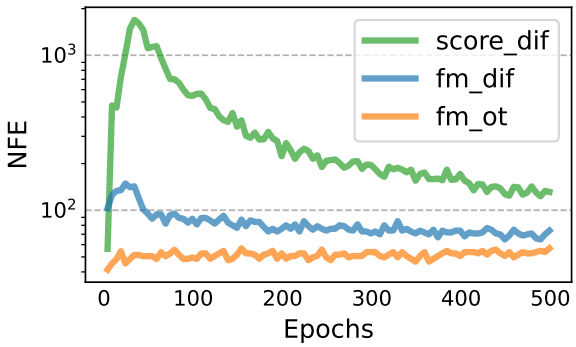


Figure B.3: Function evaluations for sampling during training, for models trained on CIFAR-10 using `dopri5` solver with tolerance $1e^{-5}$.

	CIFAR-10			**ImageNet 32×32**			**ImageNet 64×64**		
Model	K=1	K=20	K=50	K=1	K=5	K=15	K=1	K=5	K=10
Ablation									
DDPM	3.24	3.14	3.12	3.62	3.57	3.54	3.36	3.33	3.32
Score Matching	3.28	3.18	3.16	3.65	3.59	3.57	3.43	3.41	3.40
ScoreFlow	3.21	3.11	3.09	3.63	3.57	3.55	3.39	3.37	3.36
Ours									
FM ^{w}/ Diffusion	3.23	3.13	3.10	3.64	3.58	3.56	3.37	3.34	3.33
FM ^{w}/ OT	3.11	3.01	2.99	3.62	3.56	3.53	3.35	3.33	3.31

Table B.2: Negative log-likelihood (↓, in bits per dimension) on the test set with different values of K using uniform dequantization.

Figure B.4: Non-curated unconditional ImageNet-32 generated images of a CNF trained with FM-OT.

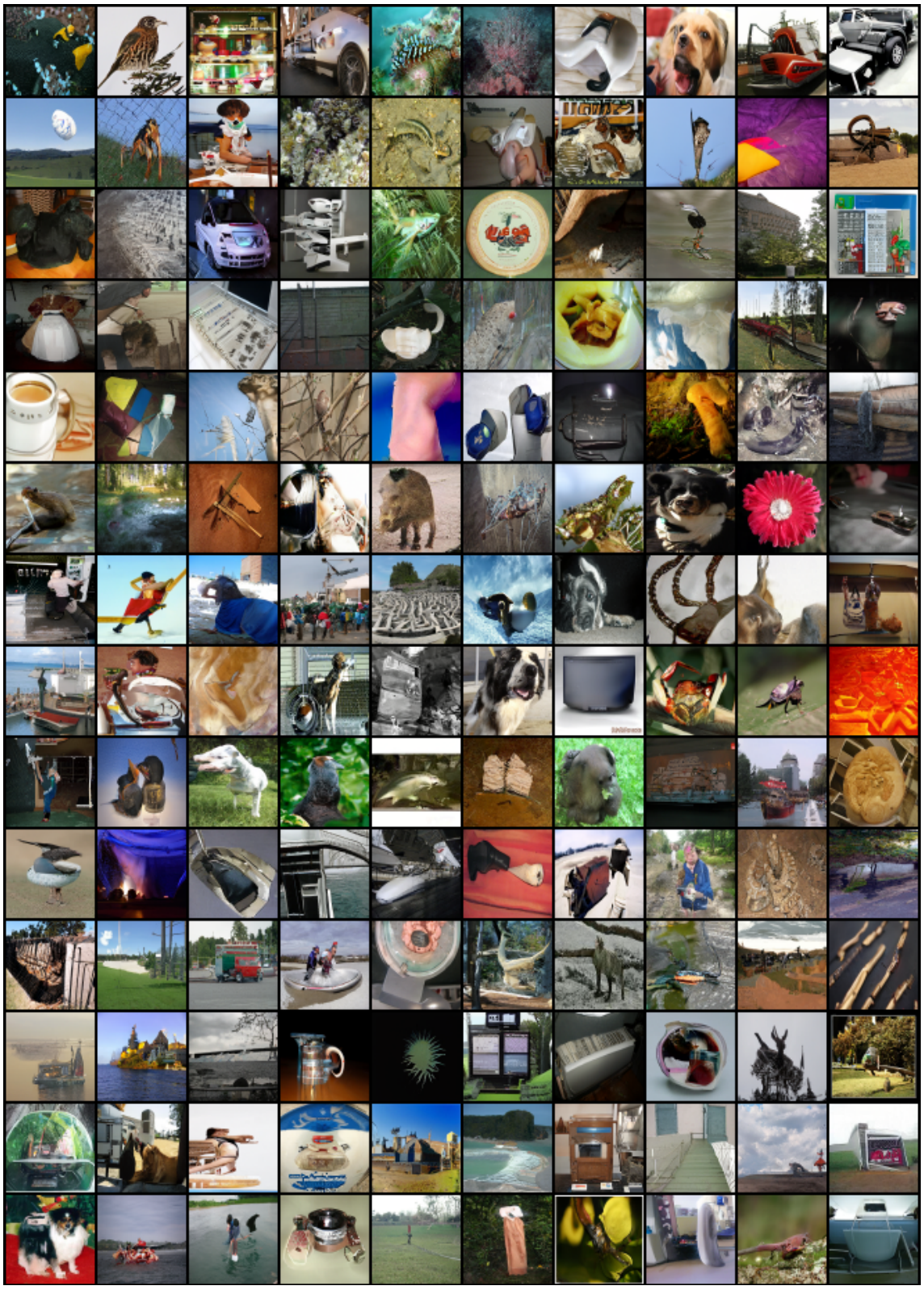


Figure B.5: Non-curated unconditional ImageNet-64 generated images of a CNF trained with FM-OT.

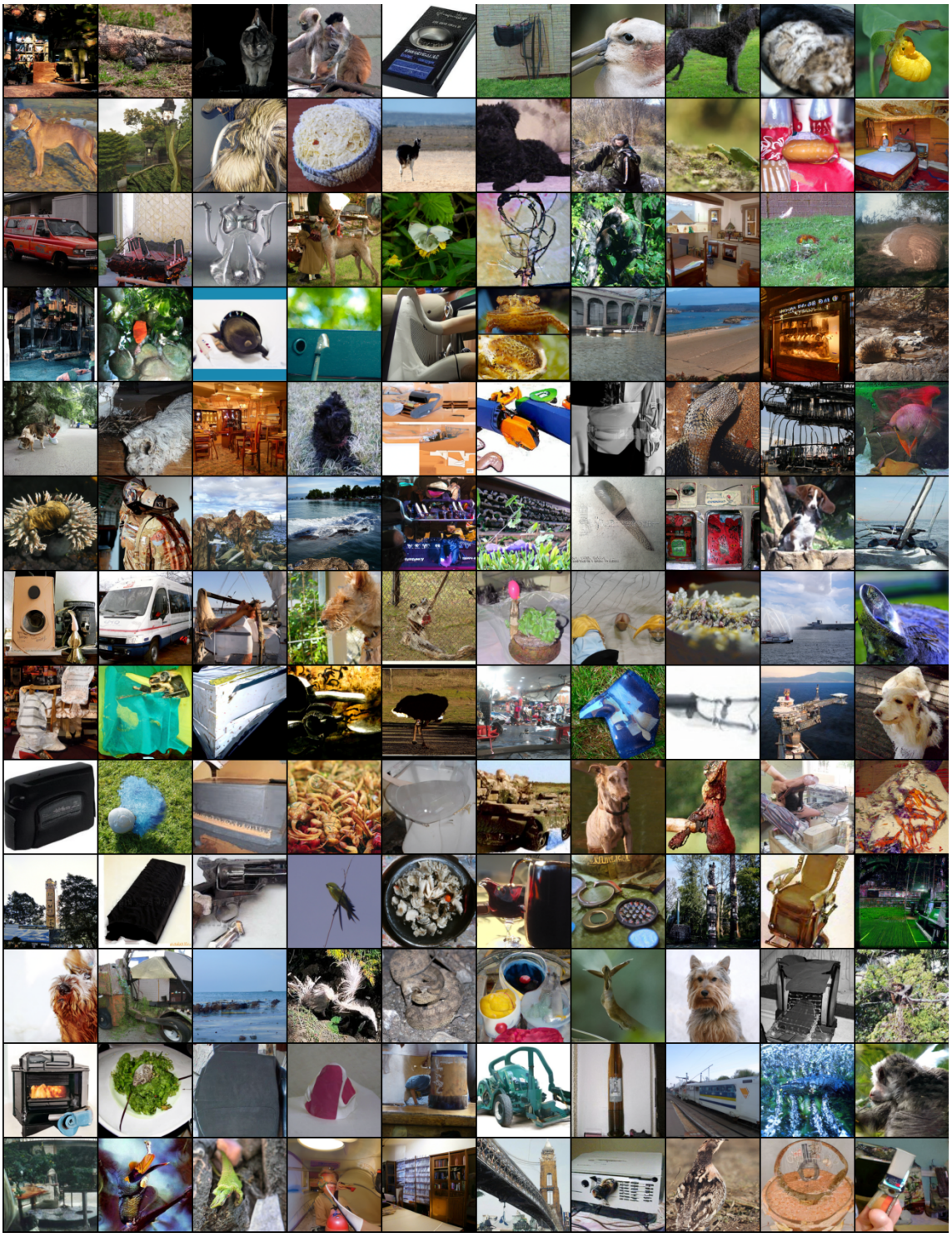


Figure B.6: Non-curated unconditional ImageNet-128 generated images of a CNF trained with FM-OT.

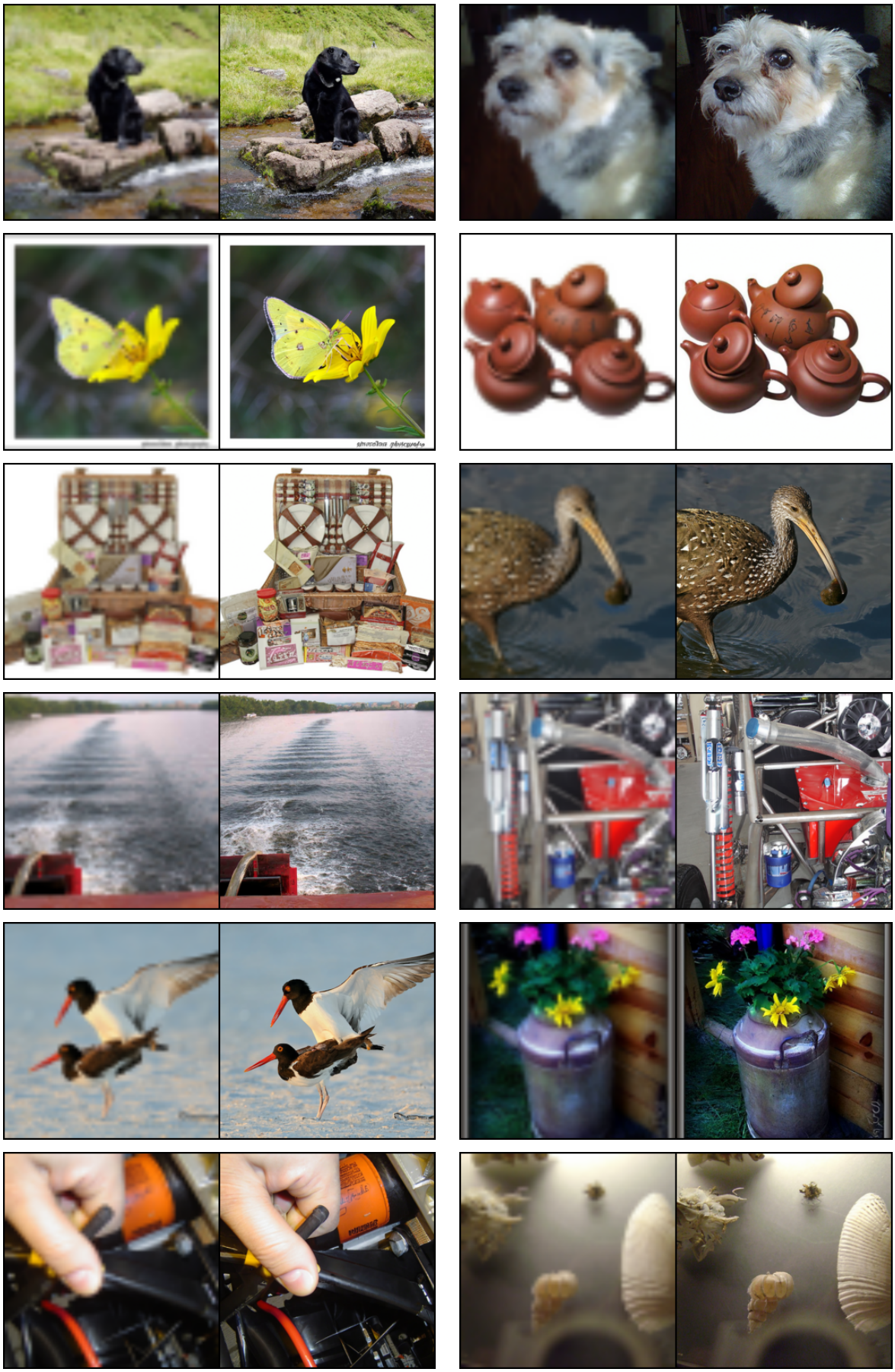


Figure B.7: Conditional generation 64×64→256×256. Flow Matching OT upsampled images from validation set.

Figure B.8: Conditional generation 64×64→256×256. Flow Matching OT upsampled images from validation set.

B.3 Additions for Chapter 5

B.3.1 Coupling algorithms

Multisample FM makes use of batch coupling algorithms to construct an implicit joint distribution satisfying the marginal constraints. While BatchOT coupling is motivated by approximating the OT map, we consider other lower complexity coupling algorithms which produce coupling that satisfy some desired property of optimal couplings. In Table B.3 we summarize the runtime complexities for the different algorithms used in this work. We will now describe in detail the Stable and Heuristic coupling algorithms.

Table B.3: Runtime complexities of the different coupling algorithms as a function of the batch size k.

	CondOT	BatchOT	BatchEOT	Stable	Heuristic
Runtime Complexity	$\mathcal{O}(1)$	$\mathcal{O}(k^3)$	$\tilde{\mathcal{O}}(k^2/\epsilon)$	$\mathcal{O}(k^2 \log(k))$	$\mathcal{O}(k^2 \log(k))$

Stable couplings

[Wol20] surveys discrete optimal transport from a stable coupling perspective proving that stability is a necessary condition for OT couplings. Although stable couplings are not OT, they are cheaper to compute and are therefore an appealing approach to pursue. For completeness we formulate the Gale Shapely Algorithm in our setting in Algorithm 2. The rankings R_0, R_1 hold the preferences of the samples in $\{x_0^{(i)}\}_{i=1}^k$ and $\{x_1^{(i)}\}_{i=1}^k$ respectively. Where $R_0(i,j)$ is the rank of $x_1^{(j)}$ in $x_0^{(i)}$'s preferences and $R_1(i,j)$ is the rank of $x_0^{(j)}$ in $x_1^{(i)}$'s preferences.

Algorithm 2 Stable Coupling (Gale Shapely)

input : $\{x_0^{(i)}\}_{i=1}^k \sim q_0(x_0)$, $\{x_1^{(i)}\}_{i=1}^k \sim q_1(x_1)$, rankings R_0, R_1
initialization: σ empty assignment
while $\exists\, i \in [k]$ *s.t.* $\sigma(i)$ *is empty* **do**
  $j \leftarrow$ first sample in $R_0(i,\cdot)$ whom $x_0^{(i)}$ has not tried to match with yet
  if $\exists\, i'$ *s.t.* $\sigma(i') = j$ **then**
    if $R_1(j,i) < R_1(j,i')$ **then**
      $\sigma(i') \leftarrow$ empty
      $\sigma(i) \leftarrow j$
    end
  else
    $\sigma(i) \leftarrow j$
  end
end
output: assignment σ

Heuristic couplings

The stable coupling is agnostic to the cost of pairing samples and only takes into account the ranks. Therefore, reassignments during the Gale Shapely algorithms might increase the total cost although the rankings of assigned samples are improved. We draw inspiration from the cyclic monotonicity of OT couplings [Vil08] and from the marriage with sharing formulation in [Wol20] and modify the reassignment condition in the Gale Shapely algorithm (see Algorithm 3). The modified condition encourages "local" monotonicity between the reassigned pairs only, reassigning a pair only if the potentially newly assigned pairs have a lower cost.

Algorithm 3 Heuristic Coupling

input : $\{x_0^{(i)}\}_{i=1}^k \sim q_0(x_0)$, $\{x_1^{(i)}\}_{i=1}^k \sim q_1(x_1)$, rankings R_0, R_1, cost matrix C
initialization: σ empty assignment
while $\exists\, i \in [k]$ *s.t.* $\sigma(i)$ *is empty* **do**
  $j \leftarrow$ first sample in $R_0(i, \cdot)$ whom $x_0^{(i)}$ has not tried to match with yet
  if $\exists\, i'$ *s.t.* $\sigma(i') = j$ **then**
    $j' \leftarrow$ first sample in $R_0(i', \cdot)$ whom $x_0^{(i')}$ has not tried to match with yet
    $l \leftarrow$ second sample in $R_0(i, \cdot)$ whom $x_0^{(i)}$ has not tried to match with yet
    if $C(i,j) + C(i',j') < C(i,l) + C(i',j)$ **then**
      $\sigma(i') \leftarrow$ empty
      $\sigma(i) \leftarrow j$
    end
  else
    $\sigma(i) \leftarrow j$
  end
end
output: assignment σ

B.3.2 Experimental & evaluation details

Image datasets

We report the hyper-parameters used in Table B.4. We use the architecture from [DN21] but with much lower attention resolution. We use full 32 bit-precision for training ImageNet-32 and 16-bit mixed precision for training ImageNet-64. All models are trained using the Adam optimizer with the following parameters: $\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay $= 0.0$, and $\epsilon = 1e{-}8$. All methods we trained using identical architectures, with the same parameters for the the same number of epochs (see Table B.4 for details), with the exception of Rectified Flow, which we trained for much longer starting from the fully trained CondOT model. We use either a constant learning rate schedule or a polynomial decay schedule (see Table B.4). The polynomial decay learning rate schedule includes a warm-up phase for a specified number of training steps. In the warm-up phase, the learning rate is linearly increased from $1e{-}8$ to the peak learning rate (specified in Table B.4). Once the peak learning rate is achieved, it linearly decays the learning rate down to $1e{-}8$ until the final training step.

When reporting negative log-likelihood, we dequantize using the standard uniform dequantization [DSB17]. We report an importance-weighted estimate using

$$\text{BPD}(K) = -\frac{1}{D}\log_2 \frac{1}{K}\sum_{k=1}^{K} p_t(x + u_k), \text{ where } u_k \sim [U(0,1)]^D, \tag{B.12}$$

with x is in $\{0, \ldots, 255\}^D$. We solve for p_t at exactly $t = 1$ with an adaptive step size solver `dopri5` with `atol=rtol=1e-5` using the `torchdiffeq` [Che18] library. We used K=15 for ImageNet32 and K=10 for ImageNet64.

When computing FID, we use the TensorFlow-GAN library `https://github.com/tensorflow/gan`.

We run coupling algorithms only within each GPU. We also ran coupling algorithms across all GPUs (using the "Effective Batch Size") in preliminary experiments, but did not see noticeable gains in sample efficiency while obtaining slightly worse performance and sample quality, so we stuck to the smaller batch sizes for running our coupling algorithms.

For Rectified Flow, we use the finalized FM-CondOT model, generate 50000 noise and sample pairs, then train using the same FM-CondOT algorithm and hyperparameters on these sampled pairs. This is equivalent to their 2-Rectified Flow approach [LGL23]. For the rectification process, we train for 300 epochs.

Table B.4: Hyper-parameters used for training each model.

	ImageNet-32	ImageNet-64
Channels	256	192
Depth	3	3
Channels multiple	1,2,2,2	1,2,3,4
Heads	4	4
Heads Channels	64	64
Attention resolution	4	8
Dropout	0.0	0.1
Batch size / GPU	256	50
GPUs	4	16
Effective Batch size	1024	800
Epochs	350	575
Effective Iterations	438k	957k
Learning Rate	1e-4	1e-4
Learning Rate Scheduler	Polynomial Decay	Constant
Warmup Steps	20k	-

B.3.3 Additional tables and figures

Runtime per iteration is not significantly affected by solving for couplings

Table B.5: Absolute and relative runtime comparisons between CondOT, BatchOT and Stable matching. "It./s" denotes the number of iterations per second, and "Rel. increase" is the relative increase with respect to CondOT. Note that these are on relatively standard batch sizes (refer to §B.3.2 for exact batch sizes).

	ImageNet 32×32		**ImageNet 64×64**	
	It./s	Rel. increase	It./s	Rel. increase
CondOT (reference)	1.16	—	1.31	—
BatchOT	1.15	0.8%	1.26	3.9%
Stable	1.15	0.8%	1.26	3.9%

How batch size affects the marginal probability paths on 2D checkerboard data

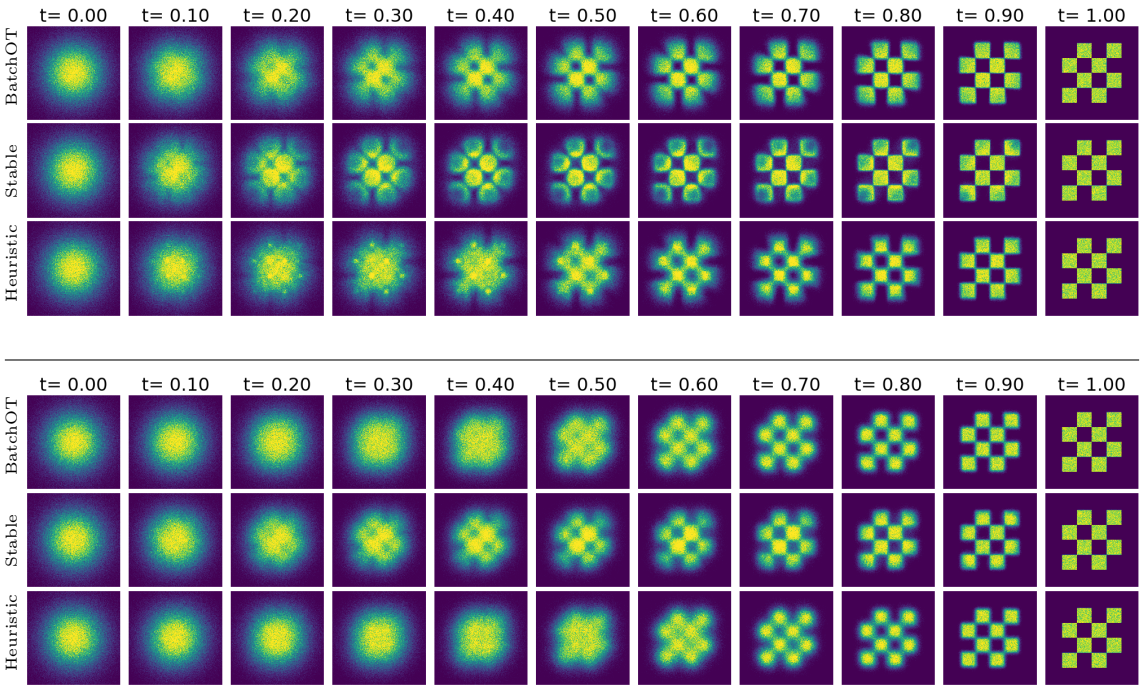


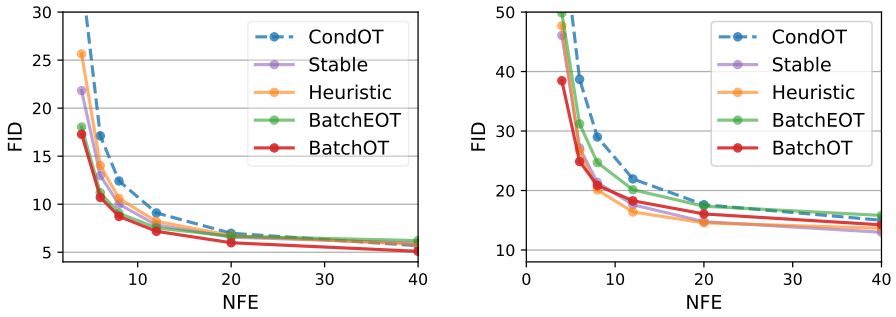
Figure B.9: Marginal probability paths. (*Top*) Batch size 64. (*Bottom*) Batch size 8.

Full results on ImageNet data

Table B.6: Multisample Flow Matching improves on sample quality and sample efficiency while not trading off performance at all compared to Flow Matching. †Reproduction using the same training hyperparameters (architecture, optimizer, training iterations) as our methods.

	ImageNet 32×32				ImageNet 64×64			
Model	NLL	FID	NFE	Var(u_t)	NLL	FID	NFE	Var(u_t)
Ablations†								
DDPM [HJA20]	3.61	5.72	330		3.27	13.80	323	
ScoreSDE [Son+21b]	3.61	6.84	198		3.30	26.64	365	
ScoreFlow [Son+21a]	3.61	9.53	189		3.34	32.78	554	
Flow Matching w/ Diffusion [Lip+23]	3.60	6.36	165		3.35	15.11	162	
Rectified Flow [LGL23]	3.59	5.55	111		3.31	13.02	129	
Flow Matching w/ CondOT [Lip+23]	3.58	5.04	139	594	3.27	13.93	131	1880
Ours								
Multisample Flow Matching w/ StableCoupling	3.59	5.79	148	523	3.27	11.82	132	1782
Multisample Flow Matching w/ HeuristicCoupling	3.58	5.29	133	555	3.26	13.37	110	1816
Multisample Flow Matching w/ BatchEOT	3.58	6.14	132	508	3.26	14.92	141	1736
Multisample Flow Matching w/ BatchOT	3.58	4.68	146	507	3.27	12.37	135	1733

FID vs NFE using midpoint discretization scheme



ImageNet 32×32 ImageNet 64×64

Figure B.10: Sample quality (FID) vs compute cost (NFE); midpoint discretization.

Comparison of FID vs NFE for baseline methods DDPM and ScoreSDE

Table B.7: Comparing the FID vs. NFE on ImageNet32 for two baselines and two of our methods.

ImageNet32 FID (Euler)				
NFE	DDPM	ScoreSDE	BatchOT	Stable
Adaptive	5.72	6.84	**4.68**	5.79
40	19.56	16.96	**5.94**	7.02
20	63.08	58.02	**7.71**	8.66
12	152.59	140.95	**10.72**	11.10
8	232.97	218.66	15.64	**14.89**
6	275.28	266.76	22.08	**19.88**
4	362.37	340.17	38.86	**33.92**

ImageNet32 FID (Midpoint)				
NFE	DDPM	ScoreSDE	BatchOT	Stable
Adaptive	5.72	6.84	**4.68**	5.79
40	6.68	6.48	**5.09**	5.94
20	7.80	8.96	**5.98**	6.57
12	14.87	16.22	**7.18**	7.84
8	56.41	56.73	**8.73**	9.99
6	188.08	168.99	**10.71**	12.98
4	319.41	279.06	**17.28**	21.82

Table B.8: Comparing the FID vs. NFE on ImageNet64 for two baselines and two of our methods.

ImageNet64 FID (Euler)				
NFE	DDPM	ScoreSDE	BatchOT	Stable
Adaptive	13.80	26.64	12.37	**11.82**
40	25.83	44.16	14.79	**13.39**
20	66.42	82.97	17.06	**15.15**
12	158.46	141.79	20.94	**18.81**
8	258.49	210.29	27.56	**26.38**
6	321.04	262.20	**36.17**	37.14
4	373.08	335.54	**56.75**	63.25

ImageNet64 FID (Midpoint)				
NFE	DDPM	ScoreSDE	BatchOT	Stable
Adaptive	13.80	26.64	12.37	**11.82**
40	15.3	26.67	14.22	**12.97**
20	15.05	25.73	16.05	**14.76**
12	18.91	29.99	18.27	**17.60**
8	53.15	67.83	**20.85**	21.36
6	179.79	155.91	**24.87**	27.15
4	330.53	279.00	**38.45**	46.08

Convergence improves when using larger coupling sizes

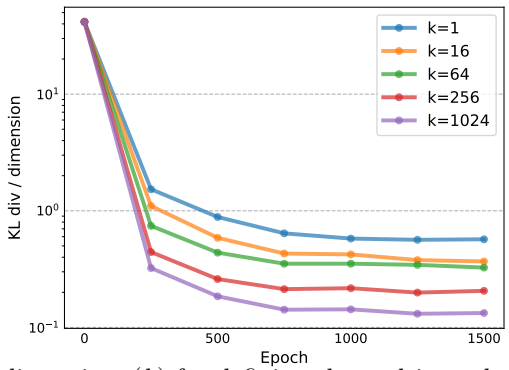


Figure B.11: Larger couplings sizes (k) for defining the multisample coupling results in faster and more stable convergence. This is done on the 64-D experiments in §5.6.3. The batch size (number of samples) for training is kept thestr same and only k is varied for solving the couplings.

B.3.4 Generated samples

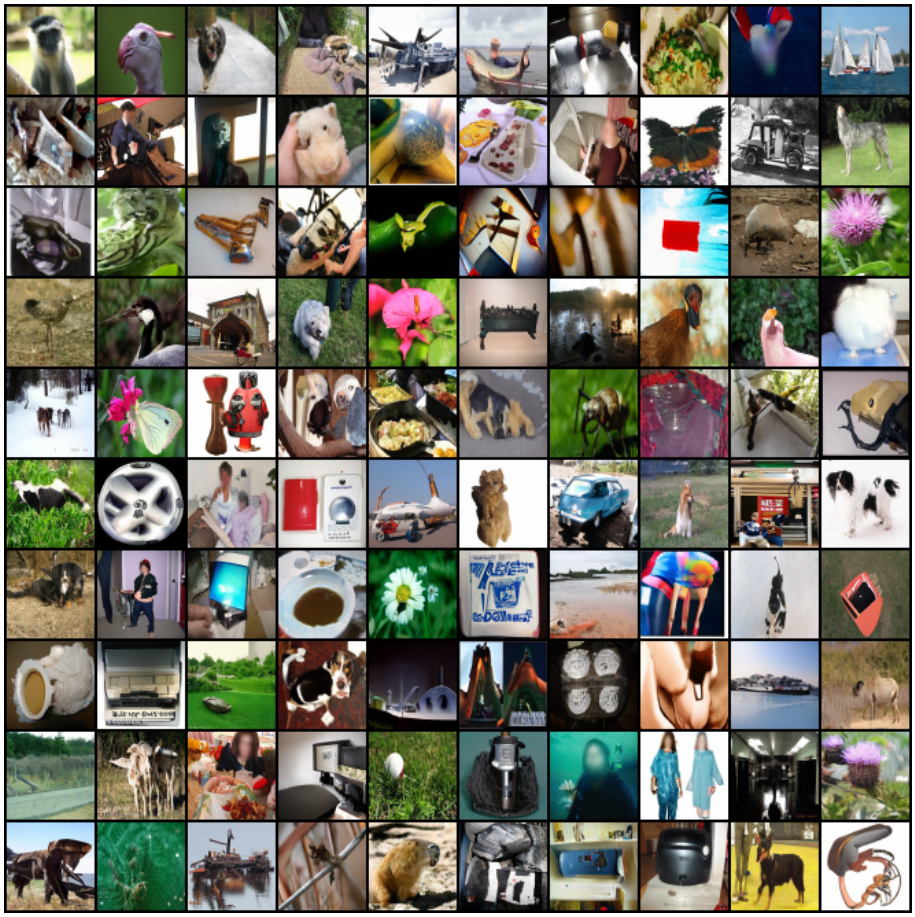


Figure B.12: Non-curated generated images for ImageNet64 using Multisample Flow Matching with BatchOT coupling.

B.4 Additions for Chapter 6

B.4.1 Implementation details

Linear Inverse Problems on Images

Optimization details. For all experiments in this section, we used the LBFGS optimizer with 20 inner iterations for each optimization step with line search. The stopping criterion was set by a target PSNR value, varying for different tasks. The solver used was midpoint with 6 function evaluations. The losses, regularizations, initializations, and stopping criteria of our algorithm for the linear inverse problems are listed in Table B.9. In the Table χ^d regularization corresponds to equation 6.10 and λ denotes the coefficients used.

Table B.9: Algorithmic choices for the ImageNet-128 linear inverse problems tasks.

	Inpainting-Center		**Super-Resolution X2**		**Gaussian Deblur**	
	$\sigma_y = 0$	$\sigma_y = 0.05$	$\sigma_y = 0$	$\sigma_y = 0.05$	$\sigma_y = 0$	$\sigma_y = 0.05$
Loss	$-\text{PSNR}(Hx, y)$		$-\text{PSNR}(Hx, y)$		$-\text{PSNR}(H^\dagger Hx, H^\dagger y)$	$-\text{PSNR}(Hx, y)$
Regularization	None	χ^d, with $\lambda = 0.01$	None	χ^d, with $\lambda = 0.01$	None	χ^d, with $\lambda = 0.01$
Initialization	0.1 blend		0.1 blend		0.1 blend	
Target PSNR	45	32	55	32	55	32

Runtimes. For noiseless tasks: inpainting center crop took on avarage 10 minutes per image, super resolution took 12.5 minutes per image and Gaussian deblurring took 15.5 minutes per image. For the noisy tasks: inpainting center crop took on avarage 4 minutes per image, super resolution took 2.5 minute per image and Gaussian deblurring took 3.5 minutes per image. Experiments ran on 32GB NVIDIA V100 GPU.

Metrics are computed using the open source TorchMetrics library [Det+22].

RED-Diff baseline. To use the RED-Diff baseline with a FM cond-OT trained model we transform the velocity field to epsilon prediction according to A.91. We searched for working parameters and reported results that outperformed the results that were produced by [Pok+23] with an epsilon prediction model, otherwise we kept the number from [Pok+23].

Inpainting with Latent Flow Models

Image inpainting

Optimization details. In this experiment, we used the LBFGS optimizer with 20 inner iterations for each optimization step with line search. The stopping criterion was set by a runtime limit of 30 minutes, but optimization usually convergences before. The solver used was midpoint with 6 function evaluations and the loss was negative PSNR without regularization. We initialized the algorithm with a backward blend with $\alpha = 0.25$. To

facilitate the backpropagation through a large T2I model, we use gradient checkpointing.

The validation set of the COCO dataset, used for evaluation, was downloaded from http://images.cocodataset.org/zips/val2017.zip.

RED-Diff baseline. To adapt RED-Diff to a latent space diffusion model, let us recall the loss used in RED-Diff:

$$\ell(\mu) = \|y - f(\mu)\|^2 + \lambda_t(\texttt{sg}\left[\epsilon(x(t), t) - \epsilon\right])^T \mu \tag{B.13}$$

where f can be any differentiable function. In the latent diffusion/flow model for inverse problems, we can model f as $f = H(\texttt{decode}(\mu))$, where `decode` applies the decoder of the autoencoder used in the latent diffusion/flow model and H is the corruption operator. We use $\text{lr} = 0.25, \lambda = 0.25$.

Audio inpainting

Optimization details. We follow the same setup described in B.4.1. Differently, we use 10 inner iterations and stop after 100 global iterations. We initialize the algorithm with a backward blend with $\alpha = 0.1$.

RED-Diff baseline. We follow the same adaptation described above in B.4.1. We use $\text{lr} = 0.05, \lambda = 0.5$.

Conditional Molecule Generation on QM9

Optimization details. In this section, we describe how Algorithm 1 was practically applied in the QM9 experiment. We initialized $x_0 \in \mathbb{R}^{n \times 9}$ for the experiment, where n represents the molecule's atom count and 9 the number of attributes per atom, using a standard Gaussian distribution. To enhance optimization process stability, we ensured x_0 had a feature-wise mean of zero and a standard deviation of one by normalizing it after every optimization step. We employed the midpoint method for the ode solver, with a total of 100 function evaluations, i.e.step size of 1/50. The optimization technique utilized was LBFGS with line search, configured with 5 optimization steps and a limit of 5 inner iterations for each step. The learning rate was set to 1. On average, generating a single molecule took approximately 2.5 minutes using a single NVIDIA Quadro RTX8000 GPU.

Table B.10: Comparison of generated molecules quality using different solvers and D-Flow.

Sample Method	NFE (#)	Molecule Stability (%)	Atom Stability (%)	Validity (%)	Validity & Uniqueness (%)
Dopri5 Adaptive Solver	-	72.03	96.14	85.00	83.84
Midpoint (50 steps)	100	72.10	96.18	85.56	84.39
Midpoint (50 steps) + optimization	100	58.97	93.87	79.38	79.38

In Table B.11 below, we report MAE values over the split to stable and non-stable molecules within the $10k$ generated samples. Other baselines and the result denoted as 'Ours' in the table report MAE on the entire $10k$ set of molecules without distinguishing between stable and non-stable ones. Since our method produces lower stability percentage, we also report the MAE on the stable and non-stable splits. It can be seen that our improved MAE performance is not due to producing non-stable molecules with lower MAE, but performance is also SOTA on generated stable molecules.

Table B.11: Quantitative evaluation of conditional molecule generation. Values reported in the table are MAE (over 10K samples) for molecule property predictions (lower is better).

Property	α	$\Delta\varepsilon$	ε_{HOMO}	ε_{LUMO}	μ	C_v
Units	Bohr^2	meV	meV	meV	D	$\frac{\text{cal}}{\text{mol}}$K
QM9*	0.10	64	39	36	0.043	0.040
EDM	2.76	655	356	584	1.111	1.101
EQUIFM	2.41	591	337	530	1.106	1.033
GEOLDM	2.37	587	340	522	1.108	1.025
Ours	1.39	344	182	330	0.300	0.784
Ours-stable	1.40	347	180	337	0.287	0.835
Ours-non stable	1.38	340	186	318	0.321	0.714

QM9. The QM9 dataset [Ram+14], a widely recognized collection, encompasses molecular characteristics and atomic positions for 130K small molecules, each containing no more than 9 heavy atoms (up to 29 atoms when including hydrogens). The train/-validation/test partitions used are according to [AHK19] and consists of 100K/18K/13 samples per partition. We provide additional details regarding the properties used in the experiment:

- α Polarizabilty - Tendency of a molecule to acquire an electric dipole moment when subjected to an external electric field.
- ε_{HOMO} - Highest occupied molecular energy.
- ε_{LUMO} - Lowest unoccupied molecular energy.
- $\Delta\varepsilon$ - The difference between HOMO and LUMO.
- μ - Dipole moment.
- C_v - Heat capacity at 298.15K.

B.4.2 Additional Experiment: Image Denoising

In this experiment, we consider the task of denoising. The corrupted signal, y, is given by $y = x + \epsilon,\ \epsilon \sim \mathcal{N}(0, \sigma_y I)$, with $\sigma_y = 0.05$. Hyperparameters are the same as in Table B.9 for the noisy experiments. Reported metrics are in Table B.12.

Table B.12: Quantitative evaluation of denoising inverse problem on face-blurred ImageNet-128.

	Denoising			
Method	FID ↓	LPIPS ↓	PSNR ↑	SSIM ↑
$\sigma_y = 0.05$				
ΠGDM [Son+23a]	9.60	0.107	35.11	0.903
OT-ODE [Pok+23]	3.14	0.062	37.34	0.964
RED-Diff [Mar+23]	9.19	0.105	32.52	0.895
Ours	2.83	0.060	36.05	0.952

B.4.3 Ablation: Regularization Coefficient

As reported in Table B.9, on the tasks of linear inverse problems on images, we used the source point χ^d regularization, equation 6.10, for the noisy case. In the plot below, B.13, we report the evaluation metrics (FID, LPIPS, PSNR, SSIM) for varying regularization coefficient values, λ, on the task of noisy super-resolution. All other hyperparameters are as reported in Table B.9.

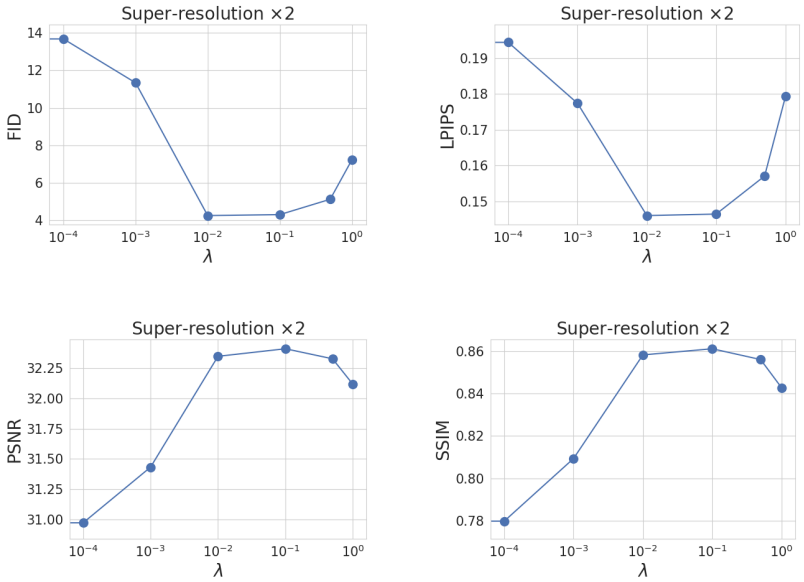


Figure B.13: Evaluation metrics vs. regularization coefficient λ of χ^d regularization over x_0 for noisy super-resolution on ace-blurred ImageNet-128.

B.4.4 Additional Qualitative Results

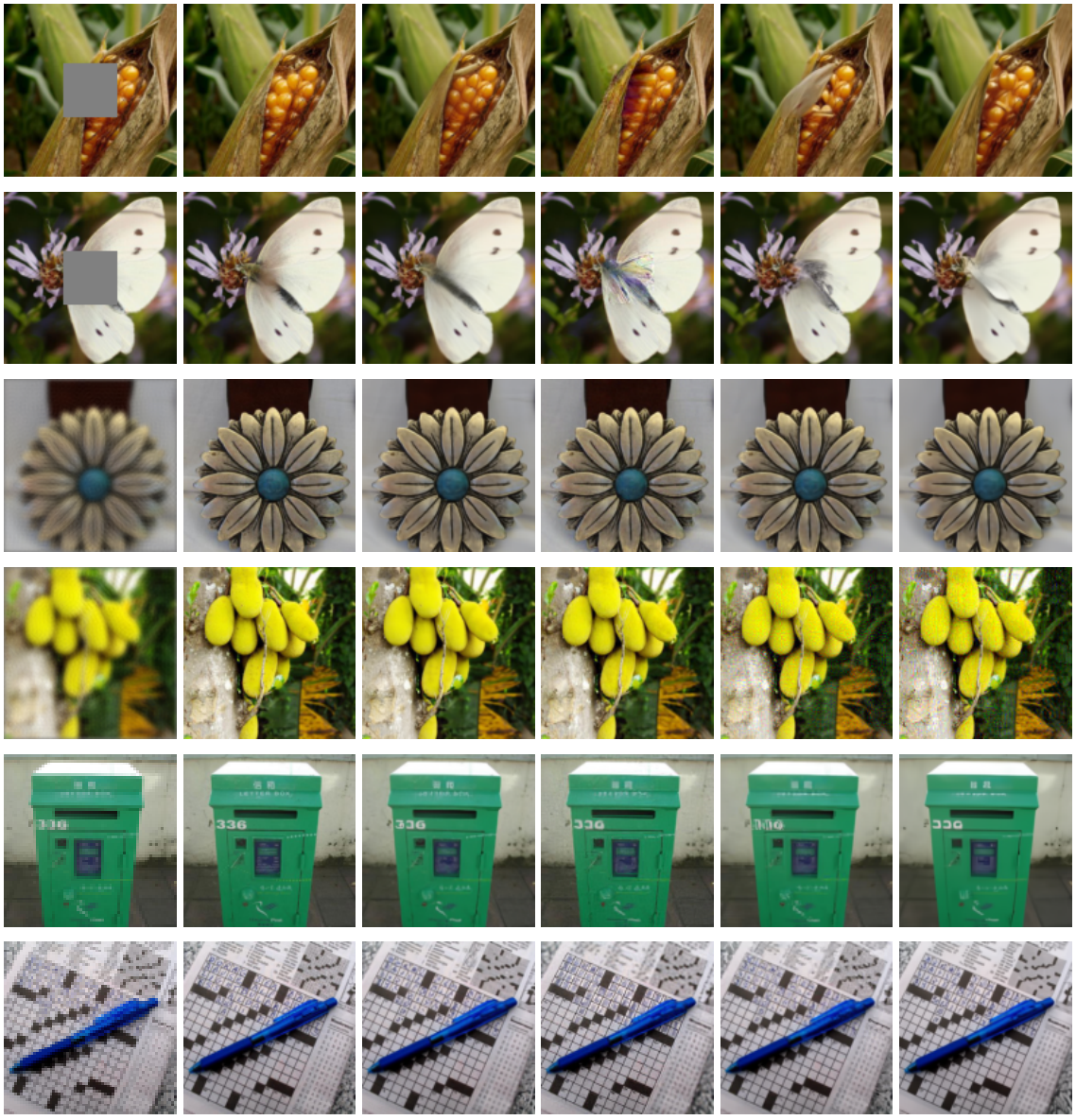


Figure B.14: Qualitative comparison for linear inverse problems on ImageNet-128 for the noiseless case. GT samples come from the face-blurred ImageNet-128 validation set.

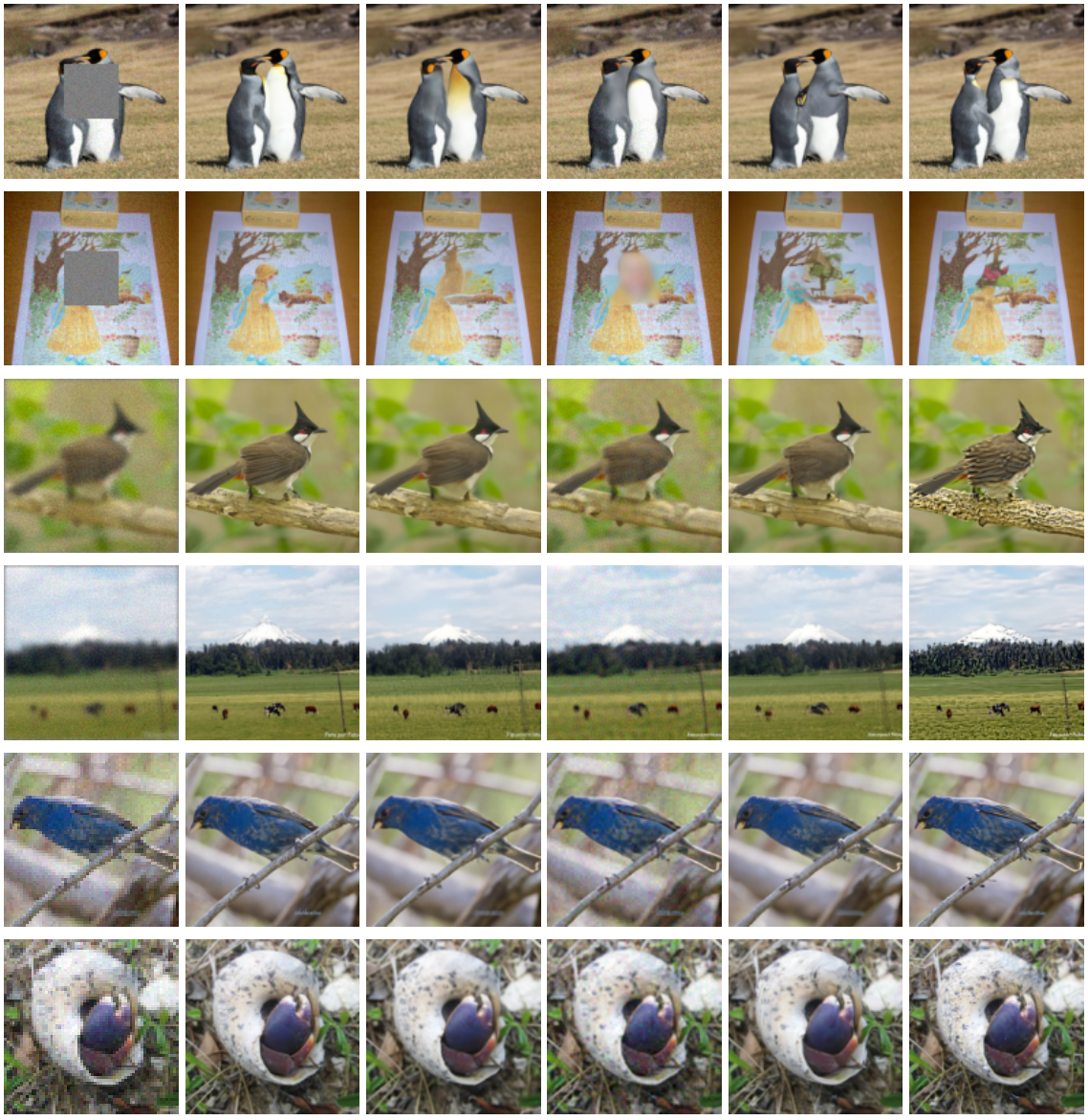


Figure B.15: Qualitative comparison for linear inverse problems on ImageNet-128 for the noisy case. GT samples come from the face-blurred ImageNet-128 validation set.

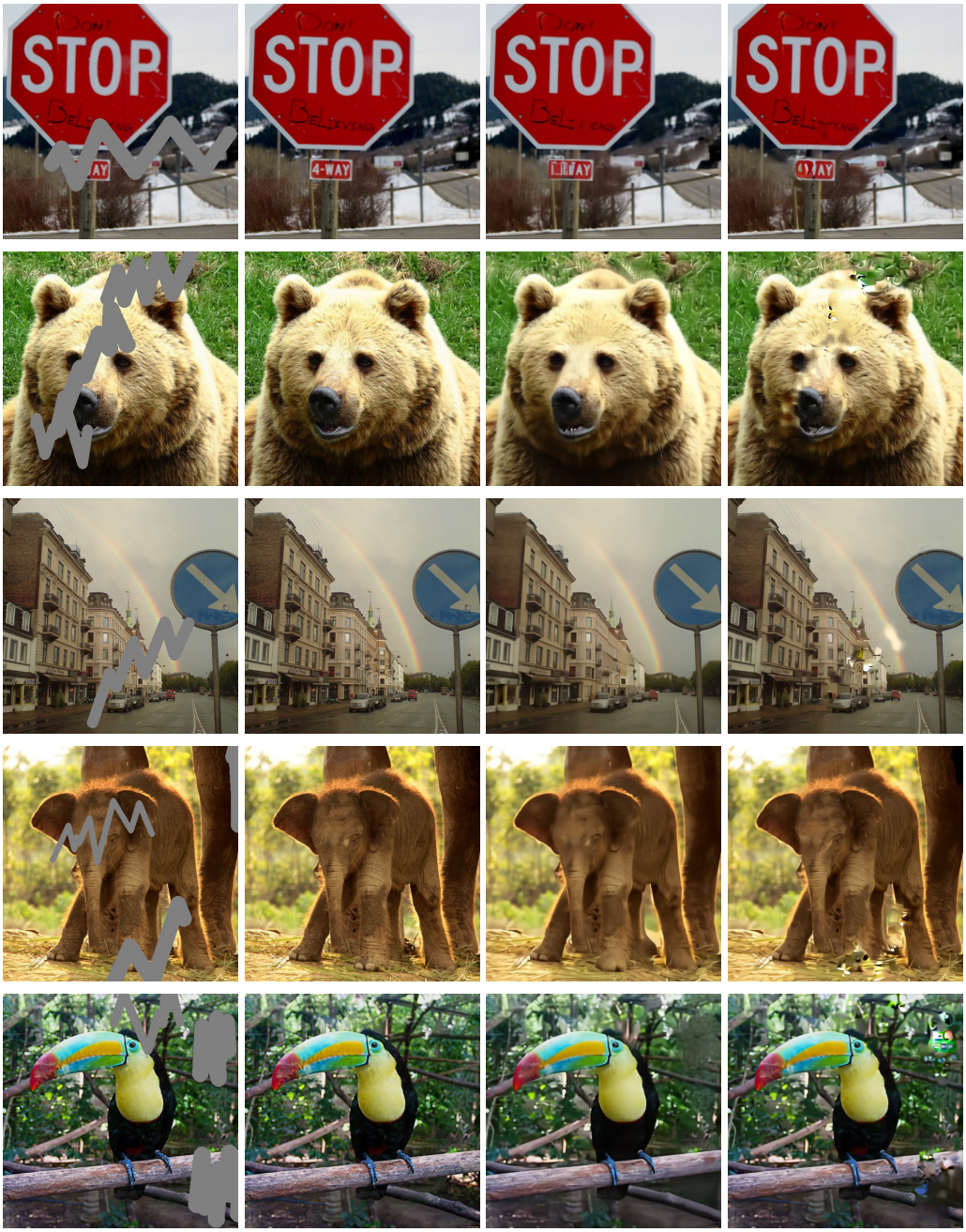


Figure B.16: Qualitative comparison for free-form inpainting on the MS-COCO dataset using a T2I latent FM model. GT samples come from the MS-COCO validation set.