Path Gradients after Flow Matching

Lorenz Vaitl
lorenz.vaitl@outlook.com

Leon Klein
Freie Universität Berlin
leon.klein@fu-berlin.de

Abstract

Boltzmann Generators have emerged as a promising machine learning tool for generating samples from equilibrium distributions of molecular systems using Normalizing Flows and importance weighting. Recently, Flow Matching has helped speed up Continuous Normalizing Flows (CNFs), scale them to more complex molecular systems, and minimize the length of the flow integration trajectories. We investigate the benefits of using Path Gradients to fine-tune CNFs initially trained by Flow Matching, in a setting where the target energy is known. Our experiments show that this hybrid approach yields up to a threefold increase in sampling efficiency for molecular systems, all while using the same model, a similar computational budget and without the need for additional sampling. Furthermore, by measuring the length of the flow trajectories during fine-tuning, we show that Path Gradients largely preserve the learned structure of the flow.

1 Introduction

Generative models, ranging from GANs (Goodfellow et al., 2014) and VAEs (Kingma & Welling, 2014) to Normalizing Flows (Rezende & Mohamed, 2015; Papamakarios et al., 2019) and Diffusion Models (Ho et al., 2020; Song et al., 2021), have advanced rapidly in recent years, driving progress both in media generation and in scientific applications such as simulation-based inference. While scientific workflows often incorporate domain-specific symmetries, they tend to under-exploit a crucial resource: the unnormalized target density.

Boltzmann Generators (Noé et al., 2019) are typically trained either via self-sampling (Boyda et al., 2021; Nicoli et al., 2020; Invernizzi et al., 2022; Midgley et al., 2023), leveraging gradients from the target distribution, or using samples from the target without incorporating gradient information (Nicoli et al., 2023; Klein et al., 2023b; Draxler et al., 2024; Klein & Noé, 2024). However, these approaches each ignore complementary parts of the training signal: either the data or its local geometry. Notably, first-order information evaluated at target samples remains underused, despite its potential to improve training. In this work, we close this gap by fine-tuning Continuous Normalizing Flows, initially trained with Flow Matching, using Path Gradients (Roeder et al., 2017; Vaitl, 2024) on samples from the target distribution. Furthermore, our approach requires computing target gradients only once per training sample, avoiding the potentially high cost of repeatedly computing gradients on newly generated samples for self-sampling.

Flow Matching, a method for training CNFs, is based on target samples. It is closely related to diffusion model training and has recently gained traction for its simulation-free training and strong empirical performance, both in generative modeling benchmarks (Lipman et al., 2023; Esser et al., 2024; Jin et al., 2024) and in scientific domains (Stark et al., 2024; Jing et al., 2024; Klein & Noé, 2024). Here, we investigate how incorporating Path Gradients on samples from the target distribution, enhances CNF performance post flow-matching.

Path gradients are low variance gradient estimators, which have strong theoretical guarantees close to the optimum (Roeder et al., 2017) and incorporate gradient information from both, the variational and the target distribution (Vaitl et al., 2024). While they have been adopted for training Normalizing Flows in the field of Lattice Gauge Theory (Bacchio et al., 2023; Kanwar, 2024; Abbott et al., 2023) and also variational inference (Agrawal & Domke, 2024; Andrade, 2023), they remain underused in Biochemistry. In this work, we explore the potential of Path Gradient fine-tuning for Boltzmann Generators with promising results. This indicates that even though training with Path Gradients is orders of magnitude slower *per iteration*, its use of additional information and low variance allows it to outperform Flow Matching *within the same computational constraints* in fine-tuning.

We make the following main contributions:

- We propose hybrid training using Flow Matching for pre-training and Path Gradients for fine-tuning. We do not require additional samples beyond the original force-labeled data, which were required already for generating the data. In order to ensure fast training, we are the first to optimize CNFs using Path Gradients for the forward KL and making use of the augmented adjoint method (Vaitl et al., 2022b).
- We show for many particle systems as well as alanine dipeptide that this fine-tuning approach can triple importance-sampling efficiency within the same computational budget.
- We investigate Path Gradients' impact on properties trained during Flow Matching, namely flow trajectory length and the MSE Flow Matching loss, and show that these are mostly unaffected by the fine-tuning.

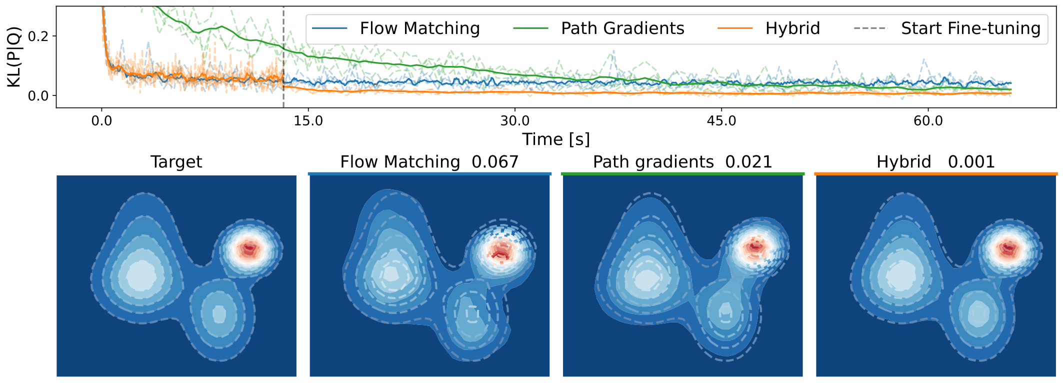

Figure 1: Training a CNF on a simple 2D Gaussian Mixture Model. Comparison between pure training with Flow Matching, pre-training with Flow Matching and fine-tuning with Path Gradients and pure training with Path Gradients. We see that given the same wall-time hybrid training performs best in terms of forward KL divergence. The bottom row shows the target and the final model after training.

2 Method

Generative models have recently garnered significant interest for biochemical applications (Noé et al., 2019; Jumper et al., 2021; Abramson et al., 2024). In this work, we focus on the application of enhancing or replacing molecular dynamics simulations.

2.1 Boltzmann Generators

Boltzmann Generators (Noé et al., 2019) aim to asymptotically generate unbiased samples from the equilibrium Boltzmann distribution

$$p(x) = \frac{1}{Z}\exp(-U(x)) \tag{1}$$

with an energy $U(x)$ and an unknown normalization factor $Z = \int \exp(-U(x))dx$. The Boltzmann distribution describes the probability of states in a physical system at thermal equilibrium. Traditional sampling methods, such as Markov Chain Monte Carlo (MCMC) and Molecular Dynamics (MD) simulations, generate samples sequentially. In contrast, Boltzmann Generators are designed to produce independent samples directly from the target distribution. There are many different instances of Boltzmann Generators, many focus on molecular systems (Dibak et al., 2022; Köhler et al., 2021; Midgley et al., 2023; Ding & Zhang, 2021b,a; Kim et al., 2024; Rizzi et al., 2023; Tamagnone et al., 2024; Schönle et al., 2024; Klein & Noé, 2024; Tan et al., 2025), while others are trained to sample lattice and other many particle systems (Wirnsberger et al., 2020; Ahmad & Cai, 2022; Nicoli et al., 2020, 2021; Schebek et al., 2024; Abbott et al., 2023; Kanwar, 2024).

Boltzmann Generators employ a Normalizing Flow to map samples from a simple base distribution into a learned sampling distribution q_θ, which is trained to approximate the target Boltzmann distribution. We can use Boltzmann Generators to obtain asymptotically unbiased estimators for observables over $p(x)$ using the sampling distribution q_θ and (self-normalized) importance sampling (Noé et al., 2019; Nicoli et al., 2020),

$$\mathbb{E}_{p(x)}\left[\mathcal{O}(x)\right] = \mathbb{E}_{q_\theta(x)}\left[\frac{p(x)}{q_\theta(x)}\mathcal{O}(x)\right], \tag{2}$$

for an observable $\mathcal{O} : \mathbb{R}^d \to \mathbb{R}$. For self-normalized importance sampling, we still have to estimate the normalization constant

$$Z = \int \exp(-U(x))dx = \mathbb{E}_{q_\theta(x)}\left[\frac{p(x)}{q_\theta(x)}\right] \tag{3}$$

with an MC estimator. This separates Boltzmann Generators from related generative methods, which only generate approximate samples from the Boltzmann distribution (Jing et al., 2022; Abdin & Kim, 2023; Klein et al., 2023a; Schreiner et al., 2023; Jing et al., 2024; Lewis et al., 2024).

We can estimate the efficiency of importance sampling using the (relative) effective sampling size (ESS). The ESS

$$\text{ESS} := \left(\mathbb{E}_{q_\theta}\left[\left(\frac{p}{q_\theta}(x)\right)^2\right]\right)^{-1} = \left(\mathbb{E}_{p}\left[\frac{p}{q_\theta}(x)\right]\right)^{-1} \in [0,1] \tag{4}$$

can be estimated either on samples from the model q_θ or the target distribution p. Roughly speaking, the ESS tells us how efficiently we are able to sample from the target distribution. The estimator ESS_q, based on samples from q_θ, might fail to detect missing modes in the target distribution, while ESS_p requires samples from the target distribution. Both estimators exhibit high variance and might overestimate the performance if too few samples are being used. For more information confer (Nicoli et al., 2023).

2.2 Continuous Normalizing Flows

Neural ODEs, introduced by Chen et al. (2018), parameterize a continuous-time transformation through a neural-network-defined vector field v_θ. By employing the adjoint sensitivity method (Pontryagin, 1987), gradients are computed in a backpropagation-like fashion with constant memory. By computing the ODE of the divergence, we can compute the change in probability, which lets us use a Neural ODE as a Normalizing Flow (Chen et al., 2018; Grathwohl et al., 2019).

Using a simple base distribution $q_0(x_0)$ and the transformation

$$T_\theta(x_0) = x_0 + \int_0^1 v_\theta(x_t, t)dt\,, \tag{5}$$

we obtain a distribution q_θ, which we can sample from and compute the probability

$$\begin{aligned}\log q_\theta(T_\theta(x_0)) &= \log q_0(x_0) - \log\det\left(\frac{\partial T_\theta(x_0)}{\partial x_0}\right)\\ &= \log q_0(x_0) - \int_0^1 \text{Tr}\left(\frac{\partial v_\theta(x_t,t)}{\partial x_t}\right)dt\,.\end{aligned} \tag{6}$$

For training samples from the target distribution, a straightforward way of approximating the target density is by maximum likelihood training, i.e. by minimizing

$$\mathcal{L}_{\text{ML}}(\theta) = -\mathbb{E}_{p(x)}\left[\log q_\theta(x)\right]. \tag{7}$$

Minimizing Equation (7) is equivalent to minimizing the forward KL divergence up to a constant

$$KL(p|q_\theta) = \mathbb{E}_{p(x)}\left[\log p(x) - \log q_\theta(x)\right] \overset{c}{=} \mathcal{L}_{\text{ML}}(\theta)\,. \tag{8}$$

As Köhler et al. (2020) showed, we can construct equivariant CNFs by using an equivariant network as the vector field.

2.3 Flow Matching

Inspired by advances in diffusion models, Flow Matching (Lipman et al., 2023; Albergo et al., 2023; Liu et al., 2022) has emerged as a promising competitor to diffusion models (Esser et al., 2024). Flow Matching enables us to train CNFs in a simulation free manner, i.e. training without the need of the expensive integration of the ODEs (5) and (6).

The idea behind Flow Matching is to posit a base probability p_0 and a vector field $u_t(x)$, which generates p_t from the base density p_0, such that the final probability p_1 equals the target density p. Given these components, Flow Matching aims to minimize the mean squared error between the target vector field $u_t(x)$ and the learned vector field $v_{\theta,t}(x)$

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}(0,1),x\sim p_t(x)}\left[||v_\theta(x,t) - u_t(x)||^2\right]\,. \tag{9}$$

since p_t and u_t are not known, it is intractable to compute the objective. Nevertheless, Lipman et al. (2023) showed that we can construct a conditional probability path $p(x|z)$ and corresponding vector field $u_t(\cdot|z)$ conditioned on $z = (x_0, x_1)$, which has identical gradients to Eq. (9). This yields the conditional Flow Matching objective

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}(0,1),x\sim p_t(x|z)}\left[||v_\theta(x,t) - u_t(x|z)||^2\right]\,. \tag{10}$$

In this framework, there are different ways of constructing the conditional probability path. We here focus on the parametrization introduced in Tong et al. (2023), which results in optimal integration paths

$$z = (x_0, x_1) \quad \text{and} \quad p(z) = \pi(x_0, x_1) \tag{11}$$

$$u_t(x|z) = x_1 - x_0 \quad \text{and} \quad p_t(x|z) = \mathcal{N}(x|t\cdot x_1 + (1-t)\cdot x_0, \sigma^2), \tag{12}$$

where $\pi(x_0, x_1)$ denotes the optimal transport map between p_0 and p_1. As this map is again intractable, it is only evaluated per batch during training. We refer to this parametrization as *OT FM*. In Klein et al. (2023b); Song et al. (2023) the authors show that we can extend this to highly symmetric systems, such as many particle systems and molecules, and still obtain optimal transport paths. The approach enforces the system's symmetries in the OT map by computing the Wasserstein distance over optimally aligned samples with respect to those symmetries. Hereafter, we denote this parameterization as *EQ OT FM*. For more details on optimal transport Flow Matching refer to Tong et al. (2023) and Klein et al. (2023b); Song et al. (2023). Note, that we obtain the original formulation in Lipman et al. (2023) by using $p(z) = p_0(x_0)p_1(x_1)$, we will refer to it as *standard FM*.

2.4 Path Gradients

Introduced in the context of variational auto-encoders (Roeder et al., 2017; Tucker et al., 2019), Path Gradient estimators have improved performance, especially when applied to normalizing flows (Agrawal et al., 2020; Vaitl et al., 2022a,b; Agrawal & Domke, 2024). In the field of simulating Lattice Field Theories, they have become a go-to tool for training flows (Bacchio et al., 2023; Kanwar, 2024; Abbott et al., 2023). In these applications, the reverse Kullback-Leibler divergence $KL(q_\theta|p)$ is minimized by self-sampling, i.e. without existing samples from the target density, but only on samples from q_θ generated while training.
Path gradients are unbiased gradient estimators which have low variance close to the optimum. We obtain Path Gradients by separating the total derivative of a reparametrized gradient for the reverse KL

$$\frac{d}{d\theta}KL(q_\theta|p) = \mathbb{E}_{q_0(x_0)}\left[\frac{d}{d\theta}\log\frac{q_\theta(T_\theta(x_0))}{p(T_\theta(x_0))}\right] \tag{13}$$

into two partial derivatives

$$\mathbb{E}_{q_0(x_0)}\left[\underbrace{\frac{\partial}{\partial x_1}\log\frac{q_\theta(x_1)}{p(x_1)}\cdot\frac{\partial T_\theta(x_0)}{\partial\theta}}_{\text{Path Gradient of } KL(q_\theta|p)}+\underbrace{\frac{\partial \log q_\theta(x_1)}{\partial\theta}\Big|_{x_1=T_\theta(x_0)}}_{\text{Score term}}\right], \tag{14}$$

called the Path Gradient and the score term [1].

Roeder et al. (2017) observed, that the martingale score term vanishes in expectation, but has a non-zero variance. Vaitl et al. (2022a) showed that its variance is $\frac{1}{N}I_\theta$ for a batch of size N and the Fisher Information I_θ of the variational distribution q_θ. The low variance close to the optimum allows Path Gradient estimators to "stick-the-landing" (Roeder et al., 2017), i.e. having zero variance at the optimum, making it an ideal tool for fine-tuning. Vaitl et al. (2024) showed that Path Gradient estimators incorporate additional information about the derivative of both target and variational distribution, opposed to the reparametrized gradient gradient estimator (Kingma & Welling, 2014; Mohamed et al., 2020) (see Appendix A.11.1 for a detailed explanation). They hypothesized that this leads to less overfitting and a generally better fit to the target density (see Vaitl (2024)). These theoretical aspects make Path Gradients a promising suitor for fine-tuning Boltzmann Generators.

Path Gradients on given samples Recently, Path Gradient estimators have been proposed for samples from distributions different from the variational density q_θ (Bauer & Mnih, 2021). By viewing the forward KL$(p_1|q_\theta)$ at $t=1$ as a reverse KL$(p_{0,\theta}|q_0)$ at $t=0$, we can straightforwardly apply Path Gradients (Vaitl et al., 2024). Here, we assume $p_{0,\theta}$ to be a (Continuous) Normalizing Flow from p_1 as its base density and T_θ^{-1} to be the parametrized diffeomorphism.

$$\begin{aligned}
KL(p_1|q_\theta) &= E_{p_1(x_1)}[\log p_1(x_1) - \log q_\theta(x_1)] \\
&= E_{p_1(x_1)}\Big[\log p_1(x_1) - \underbrace{\left(\log q_0(T_\theta^{-1}(x_1)) - \log\det\left|\frac{\partial T_\theta^{-1}(x_1)}{\partial x_1}\right|\right)}_{=\log q_\theta(x_1)}\Big] \qquad (15)\\
&= E_{p_1(x_1)}\Big[\underbrace{\left(\log p_1(x_1) - \log\det\left|\frac{\partial T_\theta^{-1}(x_1)}{\partial x_1}\right|\right)}_{=\log p_{0,\theta}(x_0)} - \log q_0(T_\theta^{-1}(x_1))\Big] \\
&= E_{p_{0,\theta}(x_0)}\Big[\log p_{0,\theta}(x_0) - \log q_0(x_0)\Big] = KL(p_{0,\theta}|q_0)\,. \qquad (16)
\end{aligned}$$

This different view lets us compute Path Gradients with the same ease as for the reverse KL, but on given samples from the target distribution.

While Vaitl et al. (2022a) and the present work both use Path Gradients, both minimize different losses on different samples. Our work optimizes the Forward KL$(p_1|q_\theta)$ with path gradients on existing samples from p, as proposed in Vaitl et al. (2024) using the algorithm proposed in Vaitl et al. (2022a). Vaitl et al. (2022a) minimize the reverse KL$(q_\theta|p_1)$ via self-sampling, i.e. on samples from the model q_θ. Training via self-sampling has many drawbacks, mainly: modes of the target p can be entirely missed, which invalidates all asymptotic guarantees and breaks importance sampling, see e.g. Nicoli et al. (2023). This becomes increasingly likely in higher dimensions. Further, the forces for the newly generated samples have to be evaluated, which can increase the cost of training or lead to unstable optimization behavior.

Since the publication Vaitl et al. (2022b), Flow Matching has been established as the de facto training method for CNFs. Our work investigates and combines the performance of Flow Matching and Path Gradients. Specifically, we investigate the effect of Path Gradients on the inner workings of the CNF, like e.g. trajectory length, while Vaitl et al. (2022b) simply looked at the performance compared to standard self-sampling losses.

In Appendix A.11, we zoom in on the properties of the different estimators. We show that while Path Gradients exhibit zero variance at the optimum, generically, Flow Matching does not. We further

[1]The score term $\frac{\partial \log q_\theta(x)}{\partial\theta}$ is not to be confused with the force term $\frac{\partial \log q_\theta(x)}{\partial x}$, which is often called the score in the context of diffusion models (Song et al., 2021).

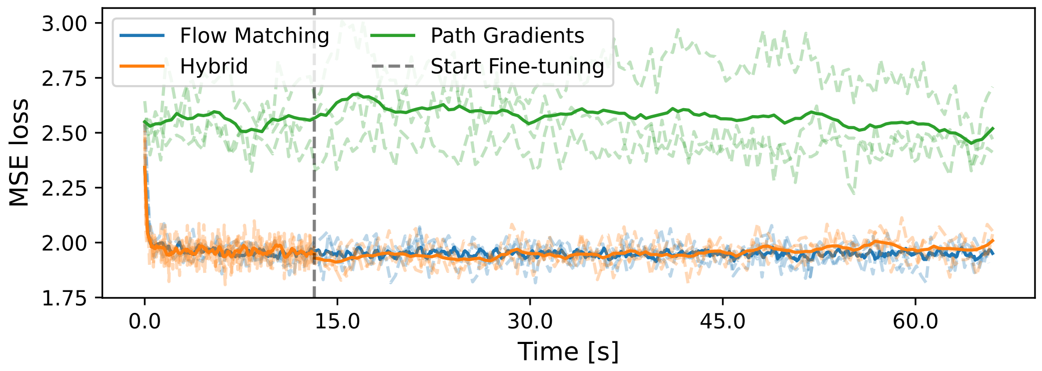

Figure 2: FM loss objective 10 during training on 2D GMM averaged on three runs. Training with Path gradients leaves the MSE largely unchanged.

explicate the theoretical properties of the existing estimators for training via Maximum Likelihood, Path Gradients and Flow Matching.

While the naive calculation of Path Gradients requires significantly more compute and memory than Flow Matching, we take several steps to obtain constant memory and speed ups in the next section. As we show experimentally, fine-tuning with Path Gradient for a few epochs significantly improves the performance for Continuous Normalizing Flows compared to only using Flow Matching.

3 Path Gradients and Flow Matching

To build an intuition for the behavior of Flow Matching (FM) versus Path Gradient (PG) estimators, we start with a simple 2D Gaussian Mixture Model. We compare three training strategies: 1) FM only, 2) PG only, and 3) a hybrid approach, using FM to quickly approximate the target distribution p_1, and subsequently applying PG for fine-tuning. Although slower per training step, PG has strong theoretical guarantees near the optimum.
For Flow Matching, we use the standard formulation proposed in Lipman et al. (2023). The dynamics v_θ is modeled using a four-layer fully connected neural network with 64 units per layer and ELU activations. Given the simplicity of the model and task, we ignore memory usage in this setup. All experiments are run on a CPU and complete in roughly one minute.
As shown in Figure 1, FM training rapidly improves initially but soon reaches a plateau with slow improvements after. PG training in contrast, progresses more slowly but eventually reaches and surpasses the FM plateau. The hybrid strategy, i.e. beginning with FM and switching to PG, results in the best performance, combining fast convergence with improved final accuracy.
Interestingly, as shown in Figure 2, applying PG after FM has little effect on the MSE loss defined in Equation (10), suggesting that PG fine-tunes the variational distribution q_θ without significantly altering the dynamics v_θ. Nonetheless, this subtle adjustment leads to visibly better samples and improved density matching (cf. Figure 1).
In the 2D case, we compute the divergence term in Equation (6) exactly by directly evaluating the trace of the Jacobian. However, this approach scales quadratically with the number of dimensions and quickly becomes computationally prohibitive in higher-dimensional settings. A practical solution is Hutchinson's trace estimator (Hutchinson, 1989), which provides us faster but noisy estimators for the divergence.
For ODE integration, we use a fourth-order Runge–Kutta (RK4) scheme with 15 time steps, resulting in 60 function evaluations per integration trajectory. Under these settings, training on a single batch is empirically about 275 times faster with Flow Matching compared to Path Gradients. This discrepancy is expected, as the cost of function evaluations differs substantially between the two methods.

While these results are promising, the 2D setup allows exact computation of the divergence and may not generalize to higher-dimensional or more complex tasks. Moreover, memory usage becomes a significant bottleneck in these more realistic scenarios.

Augmented Adjoint Method In order to have constant memory irrespective of the number of function evaluations, we adapt the augmented adjoint method (Vaitl et al., 2022b) for Path Gradient estimators for the forward KL divergence Equation (16)

$$\frac{d}{d\theta} KL(p_{0,\theta}|q_0) = \mathbb{E}_{x_1 \sim p_1} \left[\frac{\partial}{\partial x_0} \left(\log \frac{p_{0,\theta}}{q_0}(x_0) \right) \frac{\partial T_\theta^{-1}(x_1)}{\partial \theta} \right] . \tag{17}$$

In order to compute the force of the variational distribution

$$\frac{\partial \log p_{0,\theta}(x_0)}{\partial x_0} \tag{18}$$

we use the augmented adjoint method by solving the ODE

$$\frac{d}{dt} \frac{\partial \log p_{t,\theta}(x_t)}{\partial x_t} = - \frac{\partial \log p_{t,\theta}(x_t)}{\partial x_t}^T \frac{\partial v_\theta(x_t, t)}{\partial x_t} - \frac{\partial}{\partial x_t} \text{Tr} \left(\frac{\partial v_\theta(x_t, t)}{\partial x_t} \right) \tag{19}$$

with initial condition

$$\frac{\partial \log p_{1,\theta}}{\partial x_1} = \frac{\partial \log p_1(x_1)}{\partial x_1} \tag{20}$$

from $t = 1$ to 0 (Vaitl et al., 2022b).

Computing Path Gradients requires computing the gradient of the divergence

$$\text{Tr} \left(\frac{\partial v_\theta(x_t, t)}{\partial x_t} \right) \tag{21}$$

as well as the ODEs 5 and 19 per integration step. Even for training on a single time-step, this is more resource intensive than Flow Matching, which only requires the derivative of the vector field $\frac{\partial v_\theta(x_t,t)}{\partial \theta}$. Thus, although the memory demands do not increase with additional integration steps of the ODE, the memory required for Path Gradients is greater than with Flow Matching. Since the computational load associated with calculating the terms varies across different architectures, we discuss them individually for every experiment.
For RK4 with 15 integration steps, the ODEs 5, 6 and 19 are evaluated in the backward direction ($t: 1 \to 0$) and the ODE 5 and the adjoint method in the forward direction, amounting in 300 function evaluations, compared to two (the forward and backward call) with Flow Matching. Thus we expect Flow Matching to be faster than Path Gradients by a factor of at least 150.

4 Experiments

Motivated by the improvements on the toy model, we now turn to more challenging problems, namely the experiments done in Klein et al. (2023b) and Klein & Noé (2024). To this end, we use the same architecture, a CNF with an EGNN (E(n) Equivariant Graph Neural Network) (Satorras et al., 2021; Garcia Satorras et al., 2021) for the vector field. The architecture in Klein & Noé (2024) is mostly the same as in Klein et al. (2023b), with the main difference of the encoding of the atoms. While the model in Klein et al. (2023b) treats the same atom type as indistinguishable, the model in Klein & Noé (2024) encodes nearly all atoms differently. Only hydrogens bound to the same atom are treated as indistinguishable. For more details on the model architecture see Appendix A.2. While we maintain the architecture and the initial training process with Flow Matching, we change the later training procedure to include Path Gradients. As in Klein et al. (2023b), we investigate optimal transport Flow Matching (OT FM), – and for the second set of experiments on LJ13 – equivariant optimal transport Flow Matching (EQ OT FM), and naive Flow Matching (standard FM).
We first evaluate the models on two many particle system with pair-wise Lennard-Jones interactions with 13 and 55 particles. In addition we investigate Alanine dipeptide (AD2) at $T = 300K$ both with a classical and a semi-empirical force field (XTB). In contrast to (Klein et al., 2023b; Klein & Noé, 2024), we do not bias the training data towards the positive φ state. The bias was originally introduced to make learning problem simpler, as the unlikely state is more prominent (Klein et al., 2023b). This, however, skews the evaluation, since the metrics, namely ESS and Negative log likelihood (NLL), are based on the target distribution, not the biased one. Moreover, it assumes system-specific knowledge of slow-varying degrees of freedom, information that is typically unavailable for unknown systems.

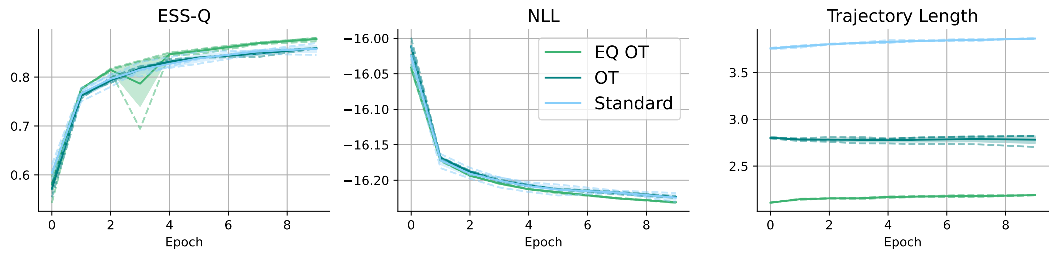

Figure 3: Reverse ESS, NLL and trajectory length for Flows trained with standard FM, Optimal Transport and Equivariant Transport during fine-tuning with Path Gradients on LJ13. We can observe that fine-tuning largely leaves the trajectory length unchanged, while substantially improving performance. Mean $\pm$ sterr over three runs.

Finally, we also investigate the 2AA dataset consisting of dipeptides simulated at $T = 310K$ with a classical force field (Klein et al., 2023a), to evaluate the influence of PG in a transferability. For more details on the datasets see Appendix A.3. We published code for replicating our experiments [2]. In total, we investigate three scenarios:

- First, fine-tuning with Path Gradients compared to fine-tuning with OT Flow Matching given the same limited computational resources in Section 4.1.
- Second, we apply Path Gradients on the FM-trained models using unlimited resources to maximize performance after training with Standard FM, OT FM and EQ OT FM as done in Klein et al. (2023b), see Section 4.2. In this second case, we additionally examine the ODE integration length to understand how PG fine-tuning influences the model.
- Finally, we test if the path gradient fine-tuning improves results for transferability in the setting by investigating transferable Boltzmann Generators (Klein & Noé, 2024) trained and evaluated on dipeptides, see Section 4.3.

4.1 Fine-tuning using the same limited resources

We enforce comparable memory and wall-time constraints to those used for Flow Matching alone by adjusting batch size and employing gradient accumulation. Consequently, our fine-tuning runs with Path Gradients complete in less wall-time and occupy a similar memory footprint. For full training statistics and implementation details, see Appendix A.4 and Appendix A.5.
Replicating the experiments in Klein et al. (2023b), we evaluate how path-gradient fine-tuning impacts model performance. Table 1 presents a comparison between models trained with pure OT Flow Matching and those optimized using the hybrid approach of OT FM pre-training and path-gradient fine-tuning.
We observe that fine-tuning with Path Gradients improves the ESS metrics, i.e. ESS_p and ESS_q, across most datasets and models, despite their high variance. The hybrid approach reliably improves the NLL for all models and datasets, but one. The only exception is the standard Flow (Klein et al., 2023b) on AD2 with XTB, where fine-tuning with FM still performs better on average. Examining the ESS, we find that both approaches fail to adequately fit the target density. This supports the hypothesis that Path Gradients are only effective for fine-tuning when the flow already provides a reasonably good approximation of the target. Notably, on LJ55, our hybrid approach nearly triples the ESS, and on AD2 with XTB, it doubles it – *in the same training time.*

4.2 Effect of Path Gradient fine-tuning on flow trajectory length

We use the provided saved models from (Klein et al., 2023b) for LJ13 and investigate the performance and flowed trajectory length during fine-tuning with Path Gradients. We used a batch-size of 256 and trained with Path Gradients for 10 epochs. In Figure 3, we can see that the different pre-trained models have similar NLLs and ESS, but differ in trajectory lengths. Fine-tuning reliably improves the

[2]github.com/lenz3000/path-grads-after-fm

Table 1: Comparison between fine-tuning with Optimal Transport Flow Matching and Path Gradients. First all models were pre-trained with Optimal Transport Flow Matching like in (Klein et al., 2023b). We compare fine-tuning with Flow Matching and Path Gradients. For all experiments, we limited the VRAM and runtime to be roughly equivalent. Mean ± sterr on three runs. **bold**: sterrs do not overlap.

Model	**Training type**	**NLL** (↓)	**ESS$_q$** %(↑)	**ESS$_p$** %(↑)
		LJ13		
(Klein et al., 2023b)	Only FM	-16.09 ± 0.03	54.36 ± 5.43	58.18 ± 0.71
	Hybrid (ours)	$\mathbf{-16.21 \pm 0.00}$	$\mathbf{82.97 \pm 0.40}$	$\mathbf{82.87 \pm 0.35}$
		LJ55		
(Klein et al., 2023b)	Only FM	-88.45 ± 0.04	3.74 ± 1.06	2.97 ± 0.08
	Hybrid (ours)	$\mathbf{-89.19 \pm 0.05}$	$\mathbf{11.04 \pm 3.98}$	$\mathbf{13.71 \pm 3.15}$
		Alanine dipeptide - XTB		
(Klein et al., 2023b)	Only FM	-107.89 ± 0.07	0.74 ± 0.33	0.01 ± 0.01
	Hybrid (ours)	-107.77 ± 0.18	0.25 ± 0.37	0.01 ± 0.01
(Klein & Noé, 2024)	Only FM	-125.75 ± 0.01	3.38 ± 0.50	2.65 ± 0.50
	Hybrid (ours)	$\mathbf{-125.82 \pm 0.02}$	$\mathbf{7.30 \pm 1.28}$	6.57 ± 2.98
		Alanine dipeptide - Classical		
(Klein et al., 2023b)	Only FM	-110.14 ± 0.01	4.88 ± 0.42	0.26 ± 0.19
	Hybrid (ours)	$\mathbf{-110.30 \pm 0.13}$	2.86 ± 3.05	0.04 ± 0.06
(Klein & Noé, 2024)	Only FM	-128.01 ± 0.01	14.42 ± 2.44	12.21 ± 1.06
	Hybrid (ours)	$\mathbf{-128.26 \pm 0.02}$	$\mathbf{24.39 \pm 6.86}$	13.89 ± 4.91

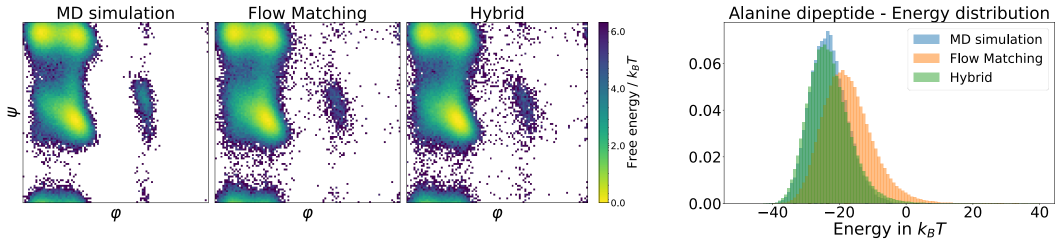

Figure 4: Alanine dipeptide results for the TBG model and the classical force field with and without Path Gradient finetuning. **Left:** Ramachandran plots for the dihedral angel distribution of a reference MD simulation and non reweighted samples from the different TBG models. **Right:** Corresponding energy distributions of generated samples.

performance of the flows with relative little change to the trajectory lengths. For the full statistics see Appendix A.7. We observe a similar pattern in the other systems investigated. For alanine dipeptide, for instance, generated samples show significantly improved bond-length and energy distributions, yet the global conformational landscape, captured by the φ and ψ dihedral-angle distributions in the Ramachandran plot, remains mostly unchanged (see Figure 4 and Figure 6).

These experiments show that our proposed hybrid approach is perfectly suited for maximizing performance while keeping the properties of the model.

4.3 Transferability on dipeptides

Finally, we also investigate whether fine-tuning with Path Gradients improves performance in the transferable setting. To this end we fine-tune Transferable Boltzmann Generators (TBG) (Klein & Noé, 2024), which were trained on 200 different dipeptides and evaluate on 16 unseen ones, like in Klein & Noé (2024). The experiments show that fine-tuning with path gradients improves the NLL and energies for all evaluated unseen test dipeptides. On average, we observe a relative improvement of around 23% in the ESS$_q$, reaching an efficiency of 9.79%. For more details see Appendix A.10.

5 Discussion

We have presented a hybrid training strategy for Boltzmann Generators that uses Flow Matching and Path Gradients. We have shown how to efficiently compute Path Gradients with constant memory and without the need for additional samples. While substantially slower per training step, Path Gradients are a powerful tool for fine-tuning, leveraging first-order information from the energy function – an underexplored avenue in the scientific machine learning community. Our results demonstrate that Path Gradients can significantly improve sample quality on the same computational budget as Flow Matching, when the model is already reasonably close to the target distribution. Our experiments show that Path Gradients only apply minor changes to the flow trajectory and to the variational distribution q_θ while still substantially increasing the sampling efficiency and performance.

5.1 Limitations

Path Gradients come with several limitations. First, they require access to a well-defined and differentiable energy function, which restricts their use to domains like molecular modeling and excludes standard tasks such as natural image generation. Second, they rely on unbiased training samples. Finally, the method is computationally more expensive and tends to improve performance only when the model is already close to the target distribution. While Path Gradients do not directly speed up the expensive CNF sampling process, they increase the ESS, thereby reducing the total number of samples needed and, hence, at least partially alleviating the high sampling cost.

5.2 Future Work

Given the similarities between Flow Matching and Diffusion models, extending Path Gradients to diffusion-based frameworks presents an exciting and promising direction. The dynamics used in our experiments have also been used for molecular conformation generation using Diffusion models (Hoogeboom et al., 2022). Adoption to their applications would be an interesting and suitable candidate for future work.
Furthermore, in model distillation (e.g. (Salimans & Ho, 2022)) the gradient information of the larger model is available, even though there might be no first order information of the original data. Here our approach could help to improve the distillation process.

5.3 Broader Impact

This foundational research has no immediate societal impact, but if scalable, it could be used to accelerate drug and materials discovery by replacing MD simulations. Potential risks include misuse for biothreat development and the lack of convergence guarantees, which may lead to incomplete sampling and misleading conclusions.

Acknowledgements

LV wants to thank Pan Kessel and Shinichi Nakajima for the helpful discussions. LK gratefully acknowledges support by BIFOLD - Berlin Institute for the Foundations of Learning and Data.

References

Abbott, R., Albergo, M., Botev, A., Boyda, D., Cranmer, K., Hackett, D., Kanwar, G., Matthews, A., Racaniere, S., Razavi, A., et al. Normalizing flows for lattice gauge theory in arbitrary space-time dimension. *ArXiv preprint*, abs/2305.12345, 2023.

Abdin, O. and Kim, P. M. Pepflow: direct conformational sampling from peptide energy landscapes through hypernetwork-conditioned diffusion. *bioRxiv*, pp. 2023–06, 2023.

Abramson, J., Adler, J., Dunger, J., Evans, R., Green, T., Pritzel, A., Ronneberger, O., Willmore, L., Ballard, A. J., Bambrick, J., et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, pp. 1–3, 2024.

Agrawal, A. and Domke, J. Disentangling impact of capacity, objective, batchsize, estimators, and step-size on flow vi. *arXiv preprint arXiv:2412.08824*, 2024.

Agrawal, A., Sheldon, D. R., and Domke, J. Advances in Black-Box VI: Normalizing Flows, Importance Weighting, and Optimization. In *Advances in Neural Information Processing Systems 33*, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/c91e3483cf4f90057d02aa492d2b25b1-Abstract.html`.

Ahmad, R. and Cai, W. Free energy calculation of crystalline solids using normalizing flows. *Modelling and Simulation in Materials Science and Engineering*, 30(6):065007, sep 2022. ISSN 0965-0393, 1361-651X. doi: 10.1088/1361-651X/ac7f4b. URL `https://iopscience.iop.org/article/10.1088/1361-651X/ac7f4b`.

Albergo, M. S., Boffi, N. M., and Vanden-Eijnden, E. Stochastic interpolants: A unifying framework for flows and diffusions. *ArXiv preprint*, abs/2303.08797, 2023. URL `https://arxiv.org/abs/2303.08797`.

Andrade, D. LOFT-Stable Training of Normalizing Flows for Variational Inference. In *6th Workshop on Tractable Probabilistic Modeling*, 2023.

Bacchio, S., Kessel, P., Schaefer, S., and Vaitl, L. Learning trivializing gradient flows for lattice gauge theories. *Physical Review D*, 107:L051504, 2023. doi: 10.1103/PhysRevD.107.L051504. URL `https://link.aps.org/doi/10.1103/PhysRevD.107.L051504`.

Bannwarth, C., Ehlert, S., and Grimme, S. Gfn2-xtb—an accurate and broadly parametrized self-consistent tight-binding quantum chemical method with multipole electrostatics and density-dependent dispersion contributions. *Journal of Chemical Theory and Computation*, 15(3):1652–1671, 2019. doi: 10.1021/acs.jctc.8b01176. URL `https://doi.org/10.1021/acs.jctc.8b01176`. PMID: 30741547.

Bauer, M. and Mnih, A. Generalized Doubly Reparameterized Gradient Estimators. In *Proceedings of the 38th International Conference on Machine Learning*, 2021. URL `http://proceedings.mlr.press/v139/bauer21a.html`.

Boyda, D., Kanwar, G., Racanière, S., Rezende, D. J., Albergo, M. S., Cranmer, K., Hackett, D. C., and Shanahan, P. E. Sampling using SU (N) gauge equivariant flows. *Physical Review D*, (7), 2021.

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems 31*, 2018.

Dibak, M., Klein, L., Krämer, A., and Noé, F. Temperature steerable flows and Boltzmann generators. *Phys. Rev. Res.*, 4:L042005, Oct 2022. doi: 10.1103/PhysRevResearch.4.L042005.

Ding, X. and Zhang, B. Computing absolute free energy with deep generative models. *Biophysical Journal*, 120(3):195a, 2021a.

Ding, X. and Zhang, B. Deepbar: A fast and exact method for binding free energy computation. *Journal of Physical Chemistry Letters*, 12:2509–2515, 3 2021b. ISSN 19487185. doi: 10.1021/acs.jpclett.1c00189.

Draxler, F., Sorrenson, P., Zimmermann, L., Rousselot, A., and Köthe, U. Free-form flows: Make any architecture a normalizing flow. In Dasgupta, S., Mandt, S., and Li, Y. (eds.), *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pp. 2197–2205. PMLR, 02–04 May 2024. URL `https://proceedings.mlr.press/v238/draxler24a.html`.

Esser, P., Kulal, S., Blattmann, A., Entezari, R., Müller, J., Saini, H., Levi, Y., Lorenz, D., Sauer, A., Boesel, F., et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

Garcia Satorras, V., Hoogeboom, E., Fuchs, F., Posner, I., and Welling, M. E(n) equivariant normalizing flows. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 4181–4192. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper/2021/file/21b5680d80f75a616096f2e791affac6-Paper.pdf`.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*, 2014.

Grathwohl, W., Chen, R. T. Q., Bettencourt, J., Sutskever, I., and Duvenaud, D. FFJORD: Free-Form Continuous Dynamics for Scalable Reversible Generative Models. In *Proceedings of the 7th International Conference on Learning Representations*, 2019.

Ho, J., Jain, A., and Abbeel, P. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems 33*, 2020.

Hoogeboom, E., Satorras, V. G., Vignac, C., and Welling, M. Equivariant diffusion for molecule generation in 3d. In *International conference on machine learning*, pp. 8867–8887. PMLR, 2022.

Hutchinson, M. F. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

Invernizzi, M., Krämer, A., Clementi, C., and Noé, F. Skipping the replica exchange ladder with normalizing flows. *The Journal of Physical Chemistry Letters*, 13:11643–11649, 2022.

Jin, Y., Sun, Z., Li, N., Xu, K., Jiang, H., Zhuang, N., Huang, Q., Song, Y., Mu, Y., and Lin, Z. Pyramidal flow matching for efficient video generative modeling. *arXiv preprint arXiv:2410.05954*, 2024.

Jing, B., Corso, G., Chang, J., Barzilay, R., and Jaakkola, T. Torsional diffusion for molecular conformer generation. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 24240–24253. Curran Associates, Inc., 2022. URL `https://proceedings.neurips.cc/paper/2022/file/994545b2308bbbbc97e3e687ea9e464f-Paper-Conference.pdf`.

Jing, B., Berger, B., and Jaakkola, T. Alphafold meets flow matching for generating protein ensembles. *arXiv preprint arXiv:2402.04845*, 2024.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

Kanwar, G. Flow-based sampling for lattice field theories. *arXiv preprint arXiv:2401.01297*, 2024.

Kim, J. C., Bloore, D., Kapoor, K., Feng, J., Hao, M.-H., and Wang, M. Scalable normalizing flows enable boltzmann generators for macromolecules. In *International Conference on Learning Representations (ICLR)*, 2024.

Kingma, D. P. and Welling, M. Auto-Encoding Variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations*, 2014. URL `http://arxiv.org/abs/1312.6114`.

Klein, L. and Noé, F. Transferable boltzmann generators. *arXiv preprint arXiv:2406.14426*, 2024.

Klein, L., Foong, A. Y. K., Fjelde, T. E., Mlodozeniec, B. K., Brockschmidt, M., Nowozin, S., Noe, F., and Tomioka, R. Timewarp: Transferable acceleration of molecular dynamics by learning time-coarsened dynamics. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023a. URL https://openreview.net/forum?id=EjMLpTgvKH.

Klein, L., Krämer, A., and Noé, F. Equivariant flow matching. *Advances in Neural Information Processing Systems*, 36:59886–59910, 2023b.

Köhler, J., Klein, L., and Noé, F. Equivariant Flows: Exact Likelihood Generative Learning for Symmetric Densities. In *Proceedings of the 37th International Conference on Machine Learning*, 2020. URL http://proceedings.mlr.press/v119/kohler20a.html.

Köhler, J., Krämer, A., and Noé, F. Smooth Normalizing Flows. In *Advances in Neural Information Processing Systems 34*, 2021.

Larsen, A. H., Mortensen, J. J., Blomqvist, J., Castelli, I. E., Christensen, R., Dułak, M., Friis, J., Groves, M. N., Hammer, B., Hargus, C., Hermes, E. D., Jennings, P. C., Jensen, P. B., Kermode, J., Kitchin, J. R., Kolsbjerg, E. L., Kubal, J., Kaasbjerg, K., Lysgaard, S., Maronsson, J. B., Maxson, T., Olsen, T., Pastewka, L., Peterson, A., Rostgaard, C., Schiøtz, J., Schütt, O., Strange, M., Thygesen, K. S., Vegge, T., Vilhelmsen, L., Walter, M., Zeng, Z., and Jacobsen, K. W. The atomic simulation environment—a python library for working with atoms. *Journal of Physics: Condensed Matter*, 29(27):273002, 2017. URL http://stacks.iop.org/0953-8984/29/i=27/a=273002.

Lewis, S., Hempel, T., Jiménez-Luna, J., Gastegger, M., Xie, Y., Foong, A. Y., Satorras, V. G., Abdin, O., Veeling, B. S., Zaporozhets, I., et al. Scalable emulation of protein equilibrium ensembles with generative deep learning. *bioRxiv*, pp. 2024–12, 2024.

Lipman, Y., Chen, R. T., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling. In *Proceedings of the 11th International Conference on Learning Representations*, 2023.

Liu, X., Gong, C., and Liu, Q. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Midgley, L. I., Stimper, V., Simm, G. N., Schölkopf, B., and Hernández-Lobato, J. M. Flow Annealed Importance Sampling Bootstrap. In *Proceedings of the 11th International Conference on Learning Representations*, 2023.

Mohamed, S., Rosca, M., Figurnov, M., and Mnih, A. Monte Carlo Gradient Estimation in Machine Learning. *Journal of Machine Learning Research*, 21:5183–5244, 2020. URL http://jmlr.org/papers/v21/19-346.html.

Nicoli, K. A., Nakajima, S., Strodthoff, N., Samek, W., Müller, K.-R., and Kessel, P. Asymptotically unbiased estimation of physical observables with neural samplers. *Physical Review E*, (2), 2020.

Nicoli, K. A., Anders, C. J., Funcke, L., Hartung, T., Jansen, K., Kessel, P., Nakajima, S., and Stornati, P. Estimation of Thermodynamic Observables in Lattice Field Theories with Deep Generative Models. *Physical Review Letters*, 126(3):032001, 2021.

Nicoli, K. A., Anders, C. J., Hartung, T., Jansen, K., Kessel, P., and Nakajima, S. Detecting and Mitigating Mode-Collapse for Flow-based Sampling of Lattice Field Theories. *Physical Review D*, 108:114501, 2023. doi: 10.1103/PhysRevD.108.114501. URL https://link.aps.org/doi/10.1103/PhysRevD.108.114501.

Noé, F., Olsson, S., Köhler, J., and Wu, H. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365(6457):eaaw1147, 2019.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. Normalizing flows for probabilistic modeling and inference. *CoRR*, 2019. URL http://arxiv.org/abs/1912.02762v1.

Papamakarios, G., Nalisnick, E. T., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. Normalizing Flows for Probabilistic Modeling and Inference. *Journal of Machine Learning Research*, 22:57:1–57:64, 2021. URL http://jmlr.org/papers/v22/19-1028.html.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Poli, M., Massaroli, S., Yamashita, A., Asama, H., Park, J., and Ermon, S. Torchdyn: Implicit models and neural numerical methods in pytorch. In *Neural Information Processing Systems, Workshop on Physical Reasoning and Inductive Biases for the Real World*, volume 2, 2021.

Pontryagin, L. S. *Mathematical theory of optimal processes*. Routledge, 1987.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *International conference on machine learning*, pp. 1530–1538. PMLR, 2015.

Rizzi, A., Carloni, P., and Parrinello, M. Multimap targeted free energy estimation, 2023.

Roeder, G., Wu, Y., and Duvenaud, D. Sticking the Landing: Simple, Lower-Variance Gradient Estimators for Variational Inference. In *Advances in Neural Information Processing Systems 30*, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/e91068fff3d7fa1594dfdf3b4308433a-Abstract.html.

Salimans, T. and Ho, J. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Satorras, V. G., Hoogeboom, E., and Welling, M. E (n) equivariant graph neural networks. In *International conference on machine learning*, pp. 9323–9332. PMLR, 2021.

Schebek, M., Invernizzi, M., Noé, F., and Rogal, J. Efficient mapping of phase diagrams with conditional boltzmann generators. *Machine Learning: Science and Technology*, 2024.

Schönle, C., Gabrié, M., Lelièvre, T., and Stoltz, G. Sampling metastable systems using collective variables and jarzynski-crooks paths. *arXiv preprint arXiv:2405.18160*, 2024.

Schreiner, M., Winther, O., and Olsson, S. Implicit transfer operator learning: Multiple time-resolution models for molecular dynamics. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=1kZx7JiuA2.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-Based Generative Modeling through Stochastic Differential Equations. In *Proceedings of the 9th International Conference on Learning Representations*, 2021.

Song, Y., Gong, J., Xu, M., Cao, Z., Lan, Y., Ermon, S., Zhou, H., and Ma, W.-Y. Equivariant flow matching with hybrid probability transport for 3d molecule generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=hHUZ5V9XFu.

Stark, H., Jing, B., Barzilay, R., and Jaakkola, T. Harmonic self-conditioned flow matching for joint multi-ligand docking and binding site design. In *Forty-first International Conference on Machine Learning*, 2024.

Tamagnone, S., Laio, A., and Gabrié, M. Coarse-grained molecular dynamics with normalizing flows. *Journal of Chemical Theory and Computation*, 20(18):7796–7805, 2024.

Tan, C. B., Bose, A. J., Lin, C., Klein, L., Bronstein, M. M., and Tong, A. Scalable equilibrium sampling with sequential boltzmann generators. *arXiv preprint arXiv:2502.18462*, 2025.

Tong, A., Malkin, N., Huguet, G., Zhang, Y., Rector-Brooks, J., Fatras, K., Wolf, G., and Bengio, Y. Conditional flow matching: Simulation-free dynamic optimal transport. *arXiv preprint arXiv:2302.00482*, 2(3), 2023.

Tucker, G., Lawson, D., Gu, S., and Maddison, C. J. Doubly Reparameterized Gradient Estimators for Monte Carlo Objectives. In *Proceedings of the 7th International Conference on Learning Representations*, 2019.

Vaitl, L. *Path Gradient Estimators for Normalizing Flows*. PhD thesis, Technische Universität Berlin, 2024.

Vaitl, L., Nicoli, K. A., Nakajima, S., and Kessel, P. Gradients should stay on path: better estimators of the reverse-and forward KL divergence for normalizing flows. *Machine Learning: Science and Technology*, 3(4):045006, 2022a.

Vaitl, L., Nicoli, K. A., Nakajima, S., and Kessel, P. Path-Gradient Estimators for Continuous Normalizing Flows. In *Proceedings of the 39th International Conference on Machine Learning*, 2022b. URL `https://proceedings.mlr.press/v162/vaitl22a.html`.

Vaitl, L., Winkler, L., Richter, L., and Kessel, P. Fast and unified path gradient estimators for normalizing flows. In *to be presented in 12th International Conference on Learning Representations*, 2024.

Wirnsberger, P., Ballard, A. J., Papamakarios, G., Abercrombie, S., Racanière, S., Pritzel, A., Jimenez Rezende, D., and Blundell, C. Targeted free energy estimation via learned mappings. *The Journal of Chemical Physics*, 153(14):144112, 2020.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: In Section 3 we present how to efficiently do fine-tuning in constant memory and Section 4, we show both, improved performance and minimal changes to the trajectory length.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: In Section 5.1 we discuss the limitations of the presented approach.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: The cross-references for the formulae for Path Gradients on samples in Section 2.4, as well as the adapted augmented adjoint state method Equation (19) are cross-referenced and numbered. The same holds true for equations on EQ-OT-, OT-, and standard FM in Section 2.3

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We provide all the necessary information in Sections 4,A.5,A.2, A.4, A.7. Further we recreate previously published experiments. Additionally we also provide code and checkpoints for the trained models.

   Guidelines:

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

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: We provide code and trained models in the supplementary material. We further describe where to download the datasets in Appendix A.3.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Appendix A.3 and Appendix A.7 specify data splits, hyperparameters, the finding of hyperparameters and optimizer choice

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: We present standard errors on three repetitions for all our experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: In Section 4 and the Appendix we specify the main CPU or GPU, the compute time as well as the VRAM usage, where it is relevant.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: Yes

   Justification: We conform to the Code of Ethics and have communicated the impact of the work in Section 5.3

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: We do this in Section 5.3.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [NA]

    Justification: Does not apply

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We mention the licenses for the data, code, and models in Appendix A.3 and Appendix A.13.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: Code is documented.

    Guidelines:

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

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

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

    Answer: [NA]

    Justification: Does not apply.

    Guidelines:

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

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

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

    Answer: [NA]

    Justification: Does not apply

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [NA]

    Justification: Only used for writing/editing

    Guidelines:

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

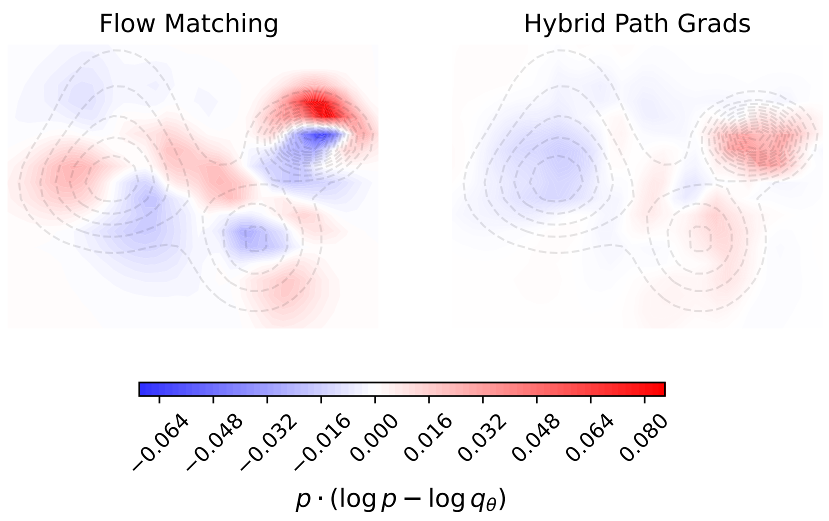

Figure 5: Loss in space after training as done in Figure 1

A Appendix

A.1 2D GMM experiment

The 2D GMM is a Gaussian Mixture model with 4 equally weighted Gaussians

$$\mathcal{N}\left(\begin{bmatrix} a_i \\ b_i \end{bmatrix}, \begin{bmatrix} c_i + 0.01 & 0 \\ 0 & d_i + 0.01 \end{bmatrix}\right), \tag{22}$$

with $a_i, b_i, c_i, d_i \sim \mathcal{N}(0, 1)$, $i \in \{1, 2, 3, 4\}$. The training set consists of 2000 samples, the $KL(p|q_\theta)$ and MSE loss are evaluated on 2048 samples. We used Adam with default parameters and lr=1e-2 for pure FM/PG and pre-training with FM and lr=5e-3 for finetuning with PG. All experiments are run on an Intel i7-1165G7 CPU and ignoring the validation, training completes in roughly 45 seconds.

Figure 5 shows the weighted difference

$$p(x) \cdot \log \frac{p(x)}{q_\theta(x)} \tag{23}$$

after training.

A.2 Architecture

We use the same model architecture as introduced in Klein et al. (2023b); Klein & Noé (2024). We here summarize it briefly, closely following the presentation in Klein & Noé (2024).

The underlying normalizing flow model for the Boltzmann Generator is a CNF. The corresponding vector field $v_\theta(t, x)$ is parametrized by an $O(D)$- and $S(N)$-equivariant graph neural network (EGNN) Garcia Satorras et al. (2021); Satorras et al. (2021). The vector field $v_\theta(x, t)$ consists of L consecutive EGNN layers. The position of the i-th particle x_i is updated according to the following set of equations:

$$h_i^0 = (t, a_i), \quad m_{ij}^l = \phi_e\left(h_i^l, h_j^l, d_{ij}^2\right), \tag{24}$$

$$x_i^{l+1} = x_i^l + \sum_{j \neq i} \frac{(x_i^l - x_j^l)}{d_{ij} + 1} \phi_d(m_{ij}^l), \tag{25}$$

$$h_i^{l+1} = \phi_h\left(h_i^l, m_i^l\right), \quad m_i^l = \sum_{j \neq i} \phi_m(m_{ij}^l) m_{ij}^l, \tag{26}$$

$$v_\theta(x^0, t)_i = x_i^L - x_i^0 - \frac{1}{N} \sum_j^N (x_j^L - x_j^0), \tag{27}$$

where the ϕ_α represent different neural networks, d_{ij} is the Euclidean distance between particle i and j, t is the time, a_i is an embedding for each particle.

The proposed model architecture in Klein et al. (2023b) uses for alanine dipeptide distinct encodings a_i for all backbone atoms and the atom types for all other atoms. In contrast, the model architecture in Klein & Noé (2024) uses distinct encodings for all atoms, except for Hydrogens bond to the same Carbon atom. We refer to this model as *TBG*, which stands for transferable Boltzmann Generator, even though we do not deploy it in a transferable way in this work.

For more details see Klein et al. (2023b); Klein & Noé (2024).

A.3 Datasets

Here we provide more details on the investigated datasets.

We use the same training and test splits as defined in Klein et al. (2023b); Klein & Noé (2024).

Lennard-Jones systems The energy $U(x)$ for the Lennard-Jones systems is given by

$$U(x) = \frac{1}{2}\left[\sum_{i,j}\left(\left(\frac{1}{d_{ij}}\right)^{12} - 2\left(\frac{1}{d_{ij}}\right)^{6}\right)\right], \tag{28}$$

where d_{ij} is the distance between particle i and j. The authors of Klein et al. (2023b) (CC BY 4.0) made the datasets available here: `https://osf.io/srqg7/?view_only=28deeba0845546fb96d1b2f355db0da5`.
For computing the metrics we use the following number of test samples:
LJ13: ESS_q: 5×10^5; ESS_p, NLL: 5×10^5.
LJ55: ESS_q: 1×10^5; ESS_p, NLL: 1×10^5.

Alanine dipeptide The classical alanine dipeptide dataset was generated with an MD simulation, using the classical *Amber ff99SBildn* force-field at 300K for implicit solvent for a duration of 1 ms Köhler et al. (2021) with the `openMM` library. The datasets is available as part of the public `bgmol` (MIT licence) repository here: `https://github.com/noegroup/bgmol`.

The alanine dipeptide dataset with the semi empirical force-field, was generated by relaxing 10^5 randomly selected states from the classical MD simulation. The relaxation was performed with the semi-empirical *GFN2-xTB* force-field for 100 fs each, using (Bannwarth et al., 2019) and the ASE library (Larsen et al., 2017) with a friction constant of 0.5 a.u. The test set was created it in the same way. The authors of Klein et al. (2023b) made the relaxed alanine dipeptide with the *semi empirical* force field available here (CC BY 4.0): `https://osf.io/srqg7/?view_only=28deeba0845546fb96d1b2f355db0da5`.

For AD2-XTB, we precompute the target forces of the samples, to re-use them during PG training.
For computing the metrics we use the following number of test samples:
ESS_q: 2×10^5; ESS_p, NLL: 1×10^5.

Dipetide dataset (2AA) The dipeptide dataset was introduced by Klein et al. (2023a) and is available at `https://huggingface.co/datasets/microsoft/timewarp`. The dataset consists of classical MD trajectories of dipeptides at 310K. There are 200 trajectories in the train set, simulated for 50ns each and 100 each in the validation and test set, simulated for 1μs each.

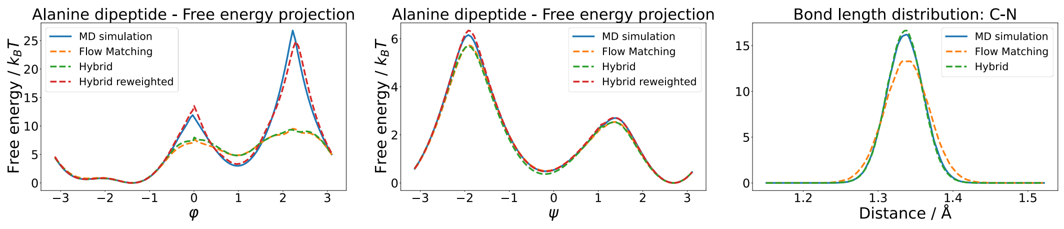

Figure 6: Alanine dipeptide results for the TBG model and the classical force field with and without Path Gradient finetuning. **Left and middle:** Free energy projection for the φ and ψ dihedral angles, respectively. **Right:** Bond-length distribution for a Carbon - Nitrogen bond.

A.4 Finetuning experiments

We used Adam with a learning rate of 1e-4 for PG and only the training set provided. For each of the training points, we require access to the force of the target. We use Hutchinson's estimator for estimating the trace of the Jacobian. Importantly, in order to keep the memory constant, we avoid saving checkpoints. For our experiments we used A100 GPUs.

For LJ13, we used a batch-size of 64 instead of 256, which uses about 90% of the original memory (FM $1.52GB$, PG $1.38GB$). For fine-tuning we used 2 epochs for Path Gradients, which took 117 min, instead of 135 min for 1000 epochs with Flow Matching. For LJ55, we use the same batch-sizes resulting in $13.24GB$ for Flow Matching and $14.76GB$ for Path Gradients. We fine-tuned with Path Gradients for 1 epoch, taking 394 minutes instead of 440 min for 400 epochs with Flow Matching.

Because we are measuring the ESS and neg-loglikelihood on the target density, we removed the reweighting from the Alanine Dipeptide data. If we changed the target density to its reweighted version and computed the gradients of the weighting w.r.t. the samples, we could also straightforwardly apply Path Gradients to the reweighted distribution. Further, fine-tuning on AD2 with Path Gradients showed some instabilities during training. To fight these, we employed gradient clipping to a norm of 1 and gradient accumulation to a batch-size of around 1000 in batches of 50. The resulting fine-tuning with Path Gradients uses $2.44GB$ VRAM and 131 min, compared to 200 minutes with $2.95GB$ with Flow Matching and batch-size 256.

We used the same number of samples like (Klein et al., 2023b) for the ESS, but note that a higher number might have been beneficial for more reliable estimates.

For the experiments, we did not run exhaustive hyperparameter tuning. The Batch-sizes were set to have a similar memory footprint. The hybrid PG learning rate was set to the middle between the initial and FM-fine-tuning one after some preliminary experiments.

A.5 Additional results for alanine dipeptide

Here we present additional alanine dipeptide results for the TBG model in Figure 6. We observe that the φ and ψ dihedral-angle projections change little after path-gradient fine-tuning. Moreover, reweighting to the target distribution via Equation (2) succeeds in both cases, though it is more efficient post–fine-tuning, as reflected by a higher ESS (see Table 1). In contrast, all bond-length distributions align much more closely with the target after fine-tuning, exemplified here by a Carbon - Nitrogen bond in Figure 6.

A.6 Additional results for alanine dipeptide - classical with more memory

For classical AD2, we further investigated the performance of fine-tuning with PGs when allowed a larger memory footprint. We fine-tuned with batch-size 1024 for two epochs, which resulted in $22GB$ VRAM usage and a runtime of 92 minutes. In Table 2 we can see that this yielded a further improvement over Table 1.

Table 2: Additional experiment for fine-tuning TBG (Klein & Noé, 2024) with Path Gradients on classical Alanine Dipeptide. First all models were pre-trained with Optimal Transport Flow Matching like in (Klein et al., 2023b). We compare fine-tuning with Flow Matching and Path Gradients. Here we did not limit the memory usage and trained PG with batch-size 1024 for 2 epochs. mean $\pm$ sterr over 3 runs.

Training type	**NLL** ($\downarrow$)	**ESS$_q$** %($\uparrow$)	**ESS$_p$** %($\uparrow$)
Only FM	-128.01 ± 0.01	14.42 ± 2.44	12.21 ± 1.06
Hybrid (ours) unlimited	$\mathbf{-128.33 \pm 0.04}$	$\mathbf{29.47 \pm 2.24}$	$\mathbf{19.57 \pm 4.79}$

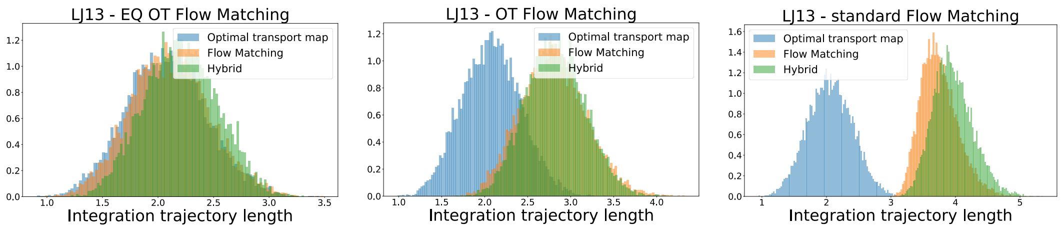

Figure 7: Integration trajectory lengths for the Lennard-Jones system with 13 particles. Compared are different Flow Matching models as well as the same models after fine-tuning with Path Gradients (Hybrid).

A.7 Additional results for experiments on flow trajectory length

Fine-tuning on the 13-particle Lennard–Jones system used an A100 for 10 epochs (152 min/run) starting from the Klein et al. (2023b) checkpoints, with training and evaluation on 5×10^5 samples. As shown in Table 3, path-gradient fine-tuning boosts NLL and ESS across the board while leaving flow trajectory lengths nearly unchanged (see Figure 7). One of the nine runs diverged during training, was discarded and repeated.

Table 3: Performance metrics before and after fine-tuning for different methods for the LJ13 system. mean $\pm$ sterr over 3 runs.

Method	**Fine-tuning**	**Trajectory Length**	**NLL**	**ESS$_q$ (%)**	**ESS$_p$ (%)**
Standard	before	3.76 ± 0.00	-16.02 ± 0.01	60.45 ± 0.89	40.23 ± 20.11
	after	3.86 ± 0.00	-16.22 ± 0.00	85.78 ± 0.69	85.93 ± 0.57
Optimal Transport	before	2.80 ± 0.01	-16.01 ± 0.01	57.16 ± 0.88	57.07 ± 0.54
	after	2.78 ± 0.04	-16.22 ± 0.00	85.91 ± 0.03	85.78 ± 0.02
Equivariant Optimal Transport	before	2.11 ± 0.00	-16.04 ± 0.00	58.11 ± 1.93	19.77 ± 12.36
	after	2.19 ± 0.00	-16.23 ± 0.00	87.77 ± 0.24	58.40 ± 29.20

A.8 Comparison with maximum likelihood training

We here present comparisons with maximum likelihood training as a baseline on LJ13. To this end we compare to Garcia Satorras et al. (2021), which is nearly the same architecture as in Klein et al. (2023b), but the model is trained via maximum likelihood training instead of flow matching. The results are shown in Table 4.

A.9 Additional energy histograms

We show additional energy histograms of the systems investigated in Section 4 in Figure 8. The average energy for every investigated system is smaller and closer to the target energy distribution after PG finetuning.

Method	NLL	ESS$_q$ in %
ML training (Garcia Satorras et al., 2021)	-15.83 ± 0.07	39.78 ± 6.19
Only FM (Klein et al., 2023b)	-16.09 ± 0.03	54.36 ± 5.43
Hybrid approach (Ours)	-16.21 ± 0.00	82.97 ± 0.40

Table 4: Results for the Lennard-Jones system with 13 particles (LJ13).

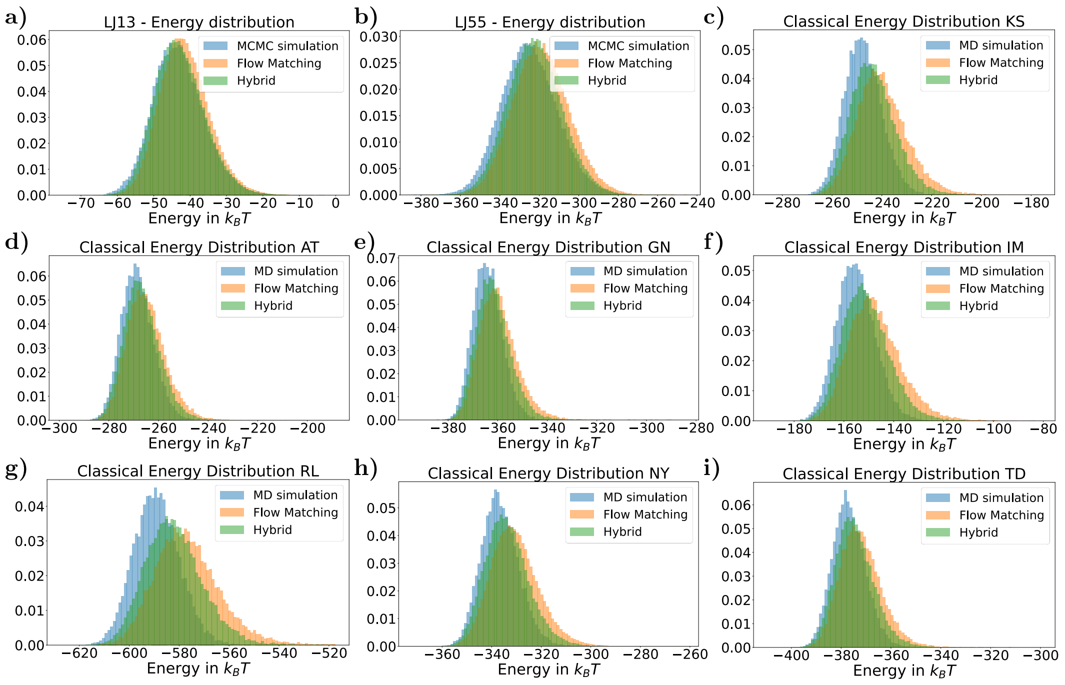

Figure 8: Energy histograms for the **a)** LJ13 systems, **b)** the LJ55 system, and **c-i)** Different didpeptides from the testset for the transferable Boltzmann Generator (TBG).

A.10 Results for transferable Boltzmann Generators (TBG) evaluated on dipeptides

We applied the hybrid approach to transferable Boltzmann Generators (TBG) on dipeptides as introduced in Klein & Noé (2024). We again fine-tune the pretrained Boltzmann Generator from Klein & Noé (2024) with path gradients. Training took 6 days on an A100 with learning rate 0.00001. This transferable Boltzmann Generator is trained on a subset of all possible dipeptides and evaluated on unseen ones (for more details see Appendix A.3). Due to the expensive evaluation, we evaluated the model on 16 test dipeptides and chose the same subset as in Klein & Noé (2024). Our experiments show that fine-tuning with path gradients improves the NLL and energies for all evaluated unseen test dipeptides with an average improvement to -100.93 from -100.72 NLL and also shows average improvements for the ESS$_q$ of around 23% to an efficiency of 9.79% as shown in Table 5. An example energy histogram is shown in Figure 8c.

A.11 About the different estimators

In the following, we first discuss the Maximum Likelihood (ML) and Path Gradient (PG) estimators for the forward KL and show benefits of PG over FM for a toy example.

A.11.1 The Maximum Likelihood and Path Gradient Estimators

In short: both the ML and PG estimators are unbiased and consistent, but they differ in variance. For PG, we can give guarantees about their variance, once q_θ equals to p, see e.g. (Vaitl, 2024; Roeder et al., 2017; Tucker et al., 2019; Vaitl et al., 2022b).

Dipeptide	**NLL (Before)**	**NLL (After PG)**	**ESS$_q$ (Before) in $\%$**	**ESS$_q$ (After PG) in $\%$**
KS	-100.82	-100.99	6.0	4.6
AT	-75.49	-75.61	20.8	21.5
GN	-65.45	-65.55	19.2	25.6
LW	-148.91	-149.22	0.4	3.6
NY	-118.24	-118.47	9.5	10.5
IM	-106.80	-107.01	3.1	4.6
TD	-81.09	-81.19	4.1	11.6
HT	-103.06	-103.27	0.2	5.5
KG	-88.60	-88.79	5.0	7.4
NF	-116.45	-116.67	3.7	12.2
RL	-135.71	-136.12	1.3	1.5
ET	-89.94	-90.08	6.3	4.1
AC	-63.90	-63.91	31.8	33.3
GP	-71.63	-71.77	10.5	8.0
KQ	-119.36	-119.68	4.3	2.0
RV	-126.28	-126.62	1.3	0.6
Average	-100.73	-100.93	7.96	9.79

Table 5: Comparison of NLL and ESS before and after PG for different dipeptides from the testset.

Both Maximum Likelihood and the Path Gradient estimators optimize the Forward KL Equation 8. Let's look at both estimators in detail and compare them. A full derivation can be found in Appendix B.3.2 of Vaitl et al. (2024) & Chapter 4 of Vaitl (2024).

To obtain the estimator $\mathcal{G}_{ML}$, we first observe that the first term in the KL divergence is constant w.r.t. θ.

$$KL(p|q_\theta) = \mathbb{E}_{p(x_1)}\left[\log p(x_1) - \log q_\theta(x_1)\right] \,, \tag{29}$$

which means that it does not enter in the gradient if we directly estimate the gradients via an MC estimator.

$$\begin{aligned} \frac{dKL(p|q_\theta)}{d\theta} &= -\mathbb{E}_p(x_1)\left[\frac{d}{d\theta}\log q_\theta(x_1)\right] \\ \approx \mathcal{G}_{ML} &= -\frac{1}{N}\sum_{i=1}^{N}\frac{d}{d\theta}\log q_\theta(x_1^{(i)})\,, x_1^{(i)} \sim p\,. \end{aligned} \tag{30}$$

For the exact calculation of the estimator, we decompose it fully with

$$q_\theta(x_1) = q_0(T_\theta^{-1}(x_1))\left|\det\frac{\partial T_\theta^{-1}(x_1)}{\partial x_1}\right| \,. \tag{31}$$

The actual calculation for the ML estimator then is

$$\mathcal{G}_{ML} = -\frac{1}{N}\sum_{i=1}^{N}\frac{d}{d\theta}\left(\log q_0(T_\theta^{-1}(x_1)) + \log\left|\frac{\partial T_\theta^{-1}(x_1)}{\partial x_1}\right|\right) \,. \tag{32}$$

For PG, we use Eq. 17 to directly obtain the MC estimator $\mathcal{G}_{PG}$

$$\begin{aligned} \frac{d}{d\theta}KL(p_{0,\theta}|q_0) &= E_{x_1\sim p_1}\left[\frac{\partial}{\partial x_0}\left(\log\frac{p_{0,\theta}}{q_0}(x_0)\right)\frac{\partial T_\theta^{-1}(x_1)}{\partial\theta}\right] \\ \approx \mathcal{G}_{PG} &= \frac{1}{N}\sum_{i=1}^{N}\frac{\partial}{\partial x_0^{(i)}}\left(\log p_{0,\theta}(x_0^{(i)}) - \log q_0(x_0^{(i)})\right)\frac{\partial T_\theta^{-1}(x_1^{(i)})}{\partial\theta}\,, x_1^{(i)} \sim p\,. \end{aligned} \tag{33}$$

Here $x_0^{(i)}$ is a shorthand for $T_\theta^{-1}(x_1^{(i)})$. If we again decompose the term we expose the full calculation

$$\mathcal{G}_{PG} = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial}{\partial x_0^{(i)}}\left(\log p(T_\theta(x_0^{(i)})) + \log\left|\frac{\partial T_\theta(x_0)}{\partial x_0}\right| - \log q_0(x_0^{(i)})\right)\frac{\partial T_\theta^{-1}(x_1^{(i)})}{\partial\theta} \tag{34}$$

Comparing (32) and (34), we see that the path gradient estimator incorporates $\mathcal{G}_{PG}$ the gradient information of the target p – $\frac{\partial \log p(T_\theta(x_0^{(i)})}{\partial x_0^{(i)}}$ – while $\mathcal{G}_{ML}$ – and by construction also $\mathcal{G}_{FM}$ – does not.

A.11.2 Variance of the estimators

Opposed to $\mathcal{G}_{ML}$ and $\mathcal{G}_{FM}$, we have nice guarantees about the variance of the PG estimator $\mathcal{G}_{PG}$ at the optimum and close to it.

We first recapitulate results from previous work and then, by using a simple example, show that $\mathcal{G}_{FM}$ does not necessarily exhibit zero variance at the optimum.

In general, the variance of the Path Gradient estimators $\mathcal{G}_{PG}$ is bounded by the squared Lipschitz constant of the term

$$(\log p_{0,\theta}(x_0) - \log q_0(x_0)) = (\log p(x_1) - \log q_\theta(x_1)) \ , \tag{35}$$

see Mohamed et al. (2020) Section 5.3.2.
Thus, if the target density p is not well approximated by q_θ, the variance of the gradient estimator can be large and training with PG might not be beneficial. If $\log p$ and $\log q_\theta$ are close in the sense that the Lipschitz constant of their difference is small, we can assume path gradient estimators to be helpful.

Gradient estimators at the optimum In the case of perfect approximation, i.e. $q_\theta(x) = p(x) \forall x \in \mathcal{X}$, the following statements about the ML estimator and the Path Gradient estimators are known. The gradients for path gradient estimators are deterministically 0, i.e. $\mathbb{E}[\mathcal{G}_{PG}] = 0, \mathrm{Var}[\mathcal{G}_{PG}] = 0$, while for ML the variance is generically nonzero $\mathbb{E}[\mathcal{G}_{ML}] = 0, \mathrm{Var}[\mathcal{G}_{ML}] = \frac{1}{N}\mathcal{I}_\theta$. Where $\mathcal{I}_\theta$ is the Fisher Information Matrix of $p_{0,\theta}$ (Vaitl et al., 2022a).

Why is that? Already the review by Papamakarios et al. (2021) notes in Section 2.3.3 that the duality of the KL divergence means that fitting the model q_θ to the target p using ML is equivalent to fitting $p_{0,\theta}$ to the base q_0 under the reverse KL. This means that the results from Vaitl (2024) directly hold. We only adapted the notation.

A.11.3 Variance of $\mathcal{G}_{FM}$ for a toy example

We do not have a term for the variance of $\mathcal{G}_{FM}$ for general settings, but we can show that it does not deterministically vanish like for $\mathcal{G}_{PG}$ via a simple example. This means better behavior for PG than for FM at the optimum, which we also verified empirically. Our assumptions aim to simplify the example as much as possible.

- First, assume the standard loss for Flow Matching.
- Further, assume the two densities to be the same D-dimensional Normal distribution $q_0 = p = \mathcal{N}(0, I)$.
- Finally, assume the CNF is the identity parametrized by a single parameter θ, i.e. $v_\theta = \theta = 0$.

In this example $q_\theta(x_1) = q_0(x_1) \cdot I$ approximates the target density p perfectly and the optimal gradient estimator is 0. Yet, in this setting the variance of the estimator $\mathcal{G}_{FM}$ is non-zero:

$$\mathrm{Var}[\mathcal{G}_{FM}] = \frac{8}{ND} \, , \tag{36}$$

preventing the model from staying at the optimum during training.

Proof: The gradient estimator for the FM loss Eq (10) is

$$\mathcal{G}_{FM} = \frac{1}{N} \sum_i \frac{\partial}{\partial \theta} \left(||v_\theta - (x_1^{(i)} - \tilde{x}_0^{(i)})||^2 \right) \, , \tag{37}$$

where $\tilde{x}_0^{(i)}, x_1^{(i)} \sim \mathcal{N}(0, I)$ [3].

[3]Note that here $\tilde{x}_0^{(i)}$ and $x_1^{(i)}$ are independently sampled. Before $x_0^{(i)}$ was the transformed sample

First, we break down the terms

$$\mathcal{G}_{FM} = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial}{\partial\theta}\frac{1}{D}\sum_{d=1}^{D}\left(v_{\theta,d} + \tilde{x}_{0,d}^{(i)} - x_{1,d}^{(i)}\right)^2$$
$$= \frac{1}{ND}\sum_{i=1}^{N}\sum_{d=1}^{D} 2\left(v_{\theta,d} + \tilde{x}_{0,d}^{(i)} - x_{1,d}^{(i)}\right)\frac{\partial v_{\theta,d}}{\partial\theta}\,. \tag{38}$$

Because v_θ is parametrized by θ, we set in $\frac{\partial v_{\theta,d}}{\partial\theta} = 1$ and $v_{\theta,d} = 0$ and separate the terms

$$\mathcal{G}_{FM} = \frac{1}{ND}\sum_i\sum_d 2(\tilde{x}_{0,d}^{(i)} - x_{1,d}^{(i)}) = \frac{2}{ND}\left(\sum_d\sum_i \tilde{x}_{0,d}^{(i)} - \sum_d\sum_i x_{1,d}^{(i)}\right). \tag{39}$$

We can compute the distribution of $\mathcal{G}_{FM}$ by using the property

$$\sum_{i=1}^{N} a_i \sim \mathcal{N}(0, N\sigma_a^2) \text{ for } a \sim \mathcal{N}(0,\sigma_a^2)\,. \tag{40}$$

So the sum over dimensions and samples follows the normal distribution with variance DN:

$$\sum_d\sum_i \tilde{x}_{0,d}^{(i)} \sim N(0, DN) \tag{41}$$

and the difference has twice its variance

$$\sum_d\sum_i \tilde{x}_{0,d}^{(i)} - \sum_d\sum_i x_{1,d}^{(i)} \sim \mathcal{N}(0, 2DN)\,. \tag{42}$$

The expectation of the estimator $\mathcal{G}_{FM}$ then is 0 and the variance is simply

$$\mathrm{Var}[\mathcal{G}_{FM}] = \frac{4}{N^2D^2}\mathrm{Var}[(\sum_d\sum_i \tilde{x}_{0,d}^{(i)} - \sum_d\sum_i x_{1,d}^{(i)})] = \frac{4}{N^2D^2}2DN = \frac{8}{ND}\,. \tag{43}$$

□

Interestingly, the derivative $\mathcal{G}_{FM}$ is invariant to re-ordering because the sums of x_1 and $\tilde{x}_0$ are invariant to permutation. The result thus also holds for OT-FM. This property is due to the simple assumption, the vector field v_θ is independent of x_t.

A.12 Pseudocode for forward KL path gradients via augmented adjoint

Algorithm 1 Augmented Adjoint Dynamics

```
1: function FORWARD($t, x_t, \nabla \log q$)
2:     $\dot{x}, \dot{\mathrm{div}} \leftarrow$ black_box_dynamics($t, x_t$)
3:     $\nabla \dot{\log} q \leftarrow \mathrm{gradient_x}\left[-\nabla \log q \cdot \dot{x} - \dot{\mathrm{div}}\right]$
4:     return $\dot{x}, \nabla \dot{\log} q, -\dot{\mathrm{div}}$
5: end function
```

Algorithm 2 Pathwise Gradient Estimator

```
1: function PATHWISEGRADIENTESTIMATE($x_1$, prior, target, flow)
2:     $\log p_1 \leftarrow -$target.energy($x_1$)
3:     $\nabla \log p_1 \leftarrow \mathrm{gradient}_{x_1}(\log p_1)$        ▷ Integrate using Augmented Adjoint state method
4:     $x_0, \nabla \log p_{0,\theta}, \log|\det J| \leftarrow$ flow.integrateAugAdjoint($x_1, \nabla \log p_1$), inverse=True)
5:     $\log q_0 \leftarrow -$prior.energy($x_0$)
6:     $\nabla \log q_0 \leftarrow \mathrm{gradient}_{x_o}(\log q_0)$        ▷ Compute gradient of loss w.r.t. sample $x_0$
7:     $\nabla_{x_0}\mathcal{L} \leftarrow \frac{1}{N}(\nabla \log p_{0,\theta} - \nabla \log q_0)$   ▷ Backpropagate using standard Adjoint state method
8:     path gradients $\leftarrow \mathrm{gradient}_\theta\,(x_0 \cdot$ detach($\nabla_{x_0}\mathcal{L}$)])
9: end function
```

A.13 Code libraries

We primarily use the following code libraries: *PyTorch* (BSD-3) (Paszke et al., 2019), *bgflow* (MIT license) (Noé et al., 2019; Köhler et al., 2020), *torchdyn* (Apache License 2.0) (Poli et al., 2021). Additionally, we use the code from (Garcia Satorras et al., 2021) (MIT license) for EGNNs, as well as the code from (Klein et al., 2023b) (MIT license) and (Klein & Noé, 2024) (MIT license) for models and dataset evaluations.