FALCON: Few-step Accurate Likelihoods for Continuous Flows

Anonymous authors
Paper under double-blind review

Abstract

Scalable sampling of molecular states in thermodynamic equilibrium is a long-standing challenge in statistical physics. Boltzmann Generators tackle this problem by pairing a generative model, capable of exact likelihood computation, with importance sampling to obtain consistent samples under the target distribution. Current Boltzmann Generators primarily use continuous normalizing flows (CNFs) trained with flow matching for efficient training of powerful models. However, likelihood calculation for these models is extremely costly, requiring thousands of function evaluations per sample, severely limiting their adoption. In this work, we propose Few-step Accurate Likelihoods for Continuous Flows (FALCON), a method which allows for few-step sampling with a likelihood accurate enough for importance sampling applications by introducing a hybrid training objective that encourages invertibility. We show FALCON outperforms state-of-the-art normalizing flow models for molecular Boltzmann sampling and is *two orders of magnitude faster* than the equivalently performing CNF model.

1 Introduction

Sampling molecular configurations from the Boltzmann distribution $p(x) \propto \exp(-\mathcal{E}(x))$ where $\mathcal{E}(x)$ is the potential energy of a configuration x, is a foundational and long-standing challenge in statistical physics. The ability to generate samples according to this distribution is the foundation for determining many other observables, such as free energies and heat capacities, which govern real-world behaviour. Consequently, efficient Boltzmann sampling is essential for progress in a large range of areas, from characterizing the function of biomolecules, to accelerating drug design, and discovering novel materials (Frenkel & Smit, 2023; Liu, 2001; Ohno et al., 2018; Stoltz et al., 2010).

The difficulty of this task arises from the structure of the energy for molecules of interest. The energy landscape is high dimensional and non-smooth with many local energy minima. These rugged energies severely challenge classical simulation-based methods like Molecular Dynamics (MD) (Leimkuhler & Matthews, 2015) and Monte Carlo Markov Chains (MCMC) as they become easily trapped in local minima, requiring a computationally inaccessible number of steps to mix between modes. These samplers generate many correlated samples, creating large inefficiencies, where an ideal sampler would generate i.i.d. samples from the underlying data distribution, $p(x)$.

Boltzmann Generators (BGs) have emerged as a way to address this inefficiency by amortizing the cost through the training of a generative model to learn to sample from $p_\theta(x)$ close to $p(x)$. These samples can then be corrected to $p(x)$ using self-normalized importance sampling (SNIS) (Noé et al., 2019). SNIS requires efficient access to $p_\theta(x)$ and $\mathcal{E}(x)$ for every sample drawn $x \sim p_\theta(x)$ for the correction step, but guarantees statistical consistency of the corrected samples, as illustrated in Fig. 1 (Noé et al., 2019; Liu, 2001).

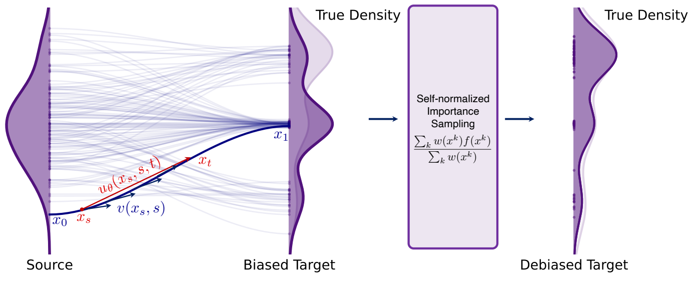

Figure 1: Flow map learns from biased data, with SNIS re-weighting generated samples consistent with the Boltzmann distribution, approaching equality with infinite samples under mild regularity conditions.

The main design choice in BGs is which type of generator to use. Modern BGs (Klein et al., 2023a; Klein & Noe, 2024) primarily make use of generators based on continuous normalizing flows (CNFs) (Chen et al., 2018; Grathwohl et al., 2019) due to their expressive power, ease of training, and flexibility of parameterization (Köhler et al., 2020) (see Table 1). However, while it is possible to access the $p_\theta(x)$ of a CNF, it is extremely computationally costly to approximate $p_\theta(x)$ with sufficient accuracy. Two primary reasons contribute to this cost: (**1**) Approximate estimators are not sufficiently accurate, making full Jacobian calculations necessary for each step along the flow; and (**2**) Many steps are necessary to control discretization error for sufficient performance (Fig. 2).

Recently, there have been significant advancements in few-step generation using flow models (Song et al., 2023; Boffi et al., 2025a; Frans et al., 2025; Guo et al., 2025; Sabour et al., 2025; Geng et al., 2025). These models are extremely powerful few-step generators with flexible architectures and efficient simulation-free training; however, these few-step samplers do not natively admit efficient estimators of the likelihood, making them unsuitable for the high-precision demands of importance sampling and scientific applications such as Boltzmann Generation (Rehman et al., 2025).

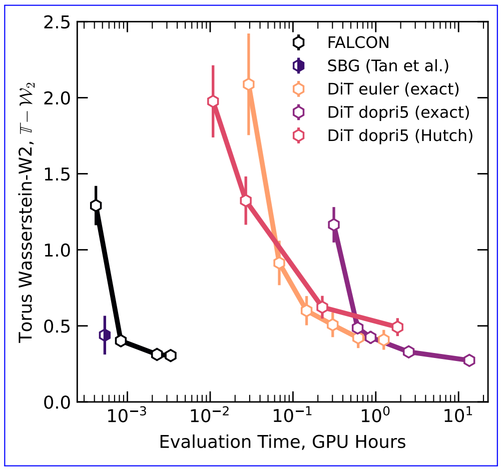

Figure 2: Performance-inference time comparison between NFs and CNFs for 10^4 dipeptide samples.

In this work, we investigate how to design a generative model that combines the best of both worlds: the training efficiency and architectural freedom of simulation-free flow models with the fast sampling and likelihood evaluation of discrete-time invertible models. We propose FEW-STEP ACCURATE LIKELIHOODS FOR CONTINUOUS FLOWS (FALCON), a flow-based model that enables few-step sampling while providing a likelihood estimate that is both fast to compute and accurate enough for importance sampling applications. FALCON leverages a hybrid training objective that combines a regression loss for stable and efficient few-step generation with a cycle-consistency term to encourage invertibility prior to convergence. Our main contributions are:

- We introduce FEW-STEP ACCURATE LIKELIHOODS FOR CONTINUOUS FLOWS (FALCON): a new continuous flow-based generative model for Boltzmann sampling that is invertible, trainable with a regression loss, and supports free-form architectures, while enabling both few-step generation and efficient likelihood evaluation.
- Orthogonally, we introduce a simple and scalable, softly equivariant continuous flow architecture that significantly improves over the current state-of-the-art equivariant flow model architecture.
- We show that FALCON is **two orders of magnitude faster** than CNF-based Boltzmann Generators for equivalent performance (Fig. 2, drastically reducing the computational cost of CNFs, and taking significant strides towards real-world large-scale molecular sampling applications.
- We show that FALCON outperforms the current state-of-the-art normalizing flow-based Boltzmann Generator across all metrics, even when FALCON is given **250× fewer samples** (Tan et al., 2025a).

2 BACKGROUND AND PRELIMINARIES

We are interested in drawing statistically independent samples from a target Boltzmann distribution p_{target} with partition function $\mathcal{Z}$, defined over $\mathbb{R}^d$:

$$p_{\text{target}}(x) \propto \exp\left(-\mathcal{E}(x)\right), \quad \mathcal{Z} = \int_{\mathbb{R}^d} \exp\left(-\mathcal{E}(x)\right) dx \tag{1}$$

where $\mathcal{E} : \mathbb{R}^d \to \mathbb{R}$ is the energy of the system, which we can efficiently compute for any x. In this work we do not require the energy to be differentiable. Unlike in the pure sampling setting (Akhound-Sadegh et al., 2024; Havens et al., 2025; Akhound-Sadegh et al., 2025; Midgley et al., 2023; Zhang & Chen, 2022; Vargas et al., 2023), we also assume access to a small biased dataset $\mathcal{D} = \{x^i\}_{i=1}^N$ of N samples (Noé et al., 2019). This makes it possible to perform an initial learning phase that fits a generative model with parameters θ, producing a proposal distribution, $p^\theta(x)$ (Noé et al., 2019).

Boltzmann Generators. (BGs) (Noé et al., 2019) combine deep generative models capable of exact likelihoods, with a target energy function and a self-normalized importance sampling (SNIS) step to re-weight generated samples to the target Boltzmann distribution. The generative model is first trained on a possibly biased dataset $\mathcal{D}$ as close as possible to p_{target}. BGs then draw K independent samples $x^i \sim p_1^\theta, i \in [K]$ and compute the corresponding unnormalized importance weights for each sample such that $w(x^i) \triangleq \exp(-\mathcal{E}(x^i))/p_1^\theta(x^i)$. Given these importance weights, we can then compute a consistent Monte–Carlo estimate of any observable $o(x)$ of interest under p_{target} using self-normalized importance sampling (Liu, 2001) as:

$$\mathbb{E}_{p_{\text{target}}}[o(x)] = \mathbb{E}_{p_1^\theta}[o(x)\bar{w}(x)] \approx \frac{\sum_{i=1}^K w(x^i)o(x^i)}{\sum_{i=1}^K w(x^i)}. \tag{2}$$

This allows for inference-time scaling as the Monte–Carlo estimate of any observable converges in probability to the correct value as the number of samples grows.

Flow Matching Models. Flow matching models (Lipman et al., 2023; Albergo & Vanden-Eijnden, 2023; Liu, 2022; Peluchetti, 2021) are probabilistic generative models that learn a continuous interpolation between an easy-to-sample distribution $p_0 = p_{\text{noise}}$ and the data distribution $p_1 = p_{\text{data}}$ in $\mathbb{R}^d$. Let $x_s = sx_1 + (1-s)x_0$ be a point at time $s \in [0,1]$ between two points $x_0 \sim p_0$ and $x_1 \sim p_1$. The flow matching objective is then $\mathbb{E}_{x_0 \sim p_0, x_1 \sim p_1, s \sim \text{Unif}(0,1)} w(s) \|v_\theta(x_s, s) - (x_1 - x_0)\|_2^2$ for some parameterized vector field v_θ, and weighting function $w : [0,1] \to \mathbb{R}^+$. We can then sample using an ordinary differential equation (ODE) $x_s^\theta = \int_0^s v_\theta(x_\tau, \tau) d\tau$ with the initial condition $x_0 \sim p_0$ and are guaranteed (under some mild assumptions) that $p_1^\theta(\hat{x}_1) \approx p_1(\hat{x}_1)$.

Furthermore, the density p_s^θ can be computed using the instantaneous change of variables formula (Chen et al., 2018) $\frac{\partial \log p(x_s)}{\partial s} = -\text{tr}\left(\frac{\partial v_\theta}{\partial x_s}\right)$ using the integral across time by solving a single $d+1$ dimensional ODE:

$$\begin{bmatrix} x_t \\ \log p_s^\theta(x_s) \end{bmatrix} = \int_0^s \begin{bmatrix} v_\theta(x_\tau, \tau) \\ -\text{tr}\left(\frac{\partial v_\theta}{\partial x_\tau}\right) \end{bmatrix} d\tau, \text{ with initial condition } \begin{bmatrix} x_0 \\ \log p_0(x_0) \end{bmatrix} \tag{3}$$

where the integral is discretized into T steps and the trace can either be computed exactly in $O(dT)$ function evaluations or approximated using Hutchinson's trace estimator $\text{tr}(J) = \mathbb{E}_\epsilon[\epsilon^T J \epsilon]$ for some noise vector $\epsilon \in \mathbb{R}^d$ in $O(T)$ function evaluations (Hutchinson, 1990). In practice, this is a major bottleneck because a large number of steps is needed to control discretization error (see Fig. 2).

Table 1: Related method overview

Method	Invertible	Regression-loss	Few Step	FreeForm Arch.
BioEmu	✗	✓	✗	✓
FlowMaps	✗	✓	✓	✓
TBG	✓	✓	✗	✓
RegFlow	✓	✓	✓	✗
Prose	✓	✗	✓	✗
FFFlows	✓	✗	✓	✓
FALCON (Ours)	✓	✓	✓	✓

Few-step Flow Models. Flow matching models can require hundreds of steps for accurate approximation of p_1. To speed up sampling, few-step flow models such as consistency models (CMs) (Song et al., 2023), optimal transport-based methods (Pooladian et al., 2023; Tong et al., 2024a;b; Shi et al., 2023), and flow map models (Boffi et al., 2025a; Sabour et al., 2025; Geng et al., 2025; Frans et al., 2025; Guo et al., 2025) attempt to train a model that generates high quality samples in many fewer steps. Recently, efficient models that take not only the current sample time, but also the target sample time have shown particular flexibility and effectiveness in the one- to few-step regimes. In these models, u_θ is augmented with an additional input t, which denotes the target time to capture the *average velocity* (Geng et al., 2025) u as:

$$u(x_s, s, t) = \frac{1}{t-s} \int_s^t v(x_\tau, \tau) d\tau \tag{4}$$

to minimize the average velocity objective:

$$\mathbb{E}_{s,t,x_s}\left[w(s,t)\left\|u_\theta(x_s, s, t) - \frac{1}{t-s}\int_s^t v(x_\tau, \tau) d\tau\right\|^2\right] \tag{5}$$

where the average velocity u_θ is parameterized by a neural network as depicted in Fig. 1 and v is the vector field of the probability flow ODE that transports samples from the noise distribution p_0 to the data distribution p_1. Once the average velocity is learned, then samples can be drawn using any discretization of the time interval $[0, 1]$, such that $t_0 = 0, t_1, \ldots t_T = 1$ as $x_{t_i} = x_{t_{i-1}} + (t_i - t_{i-1})u_\theta(x_{t_{i-1}}, t_{i-1}, t_i)$ for $i \in 1 \ldots T$. However, thus far, few-step flow models have only been applied for fast generation, and, as we show, do not natively guarantee efficient access to likelihoods in realistic settings as the learned average velocity map u_θ is not guaranteed to be invertible before the training objective is perfectly minimized, making the standard change-of-variables formula inapplicable. These models and their relationship to FALCON are summarized in Table 1.

3 Few-step Accurate Likelihoods for Continuous Flows

We now introduce Few-step Accurate Likelihoods for Continuous Flows (FALCON), a novel flow-based generative model designed to address the inherent efficiency limitations of using continuous flow models with Boltzmann Generators. Our method departs from traditional continuous normalizing flow (CNFs) by training a flow map that operates in a few discrete steps, while simultaneously achieving invertibility to ensure fast and accurate likelihood computation for Boltzmann Generation. This is achieved through a hybrid training objective, which, by enabling stable few-step generation, dramatically reduces the inference cost. This efficiency allows us to use much larger and more expressive architectures (Vaswani et al., 2017; Peebles & Xie, 2023; Ma et al., 2024) that were previously computationally infeasible to scale in the BG setting.

Flow Maps are Flawed Boltzmann Generators. The core of FALCON is a generative process that learns an invertible map from a simple base distribution p_0 to the target molecular distribution p_1 in a small number of steps. We first examine the suitability of the existing few-step flows for importance sampling applications, concluding that, on their own, they are not sufficient. We first define the continuous map with respect to a vector field v as:

$$X_v(x_s, s, t) = \int_s^t v(x_\tau, \tau)d\tau + x_s, \tag{6}$$

and note that under mild regularity conditions on v, this map is always invertible up to discretization error. For any invertible map, we can compute the change in density with respect to the input using the change of variables formula, which requires computing the Jacobian, at the approximate cost of d function evaluations, and the determinant, which, while an $O(d^3)$ operation, is in practice negligible compared to the function evaluation cost.

This invertibility property also holds for flow map models at the optima, which we address in the following proposition.

Proposition 1. *Let $u_\theta^\star$ be a minimizer of Eq. 5 with respect to some v. Also, define the Jacobian of X as $\mathbf{J}_X = \frac{\partial X}{\partial x_s}$, and the discrete flow map:*

$$X_u(x_s, s, t) = x_s + (t - s)u_\theta^\star(x_s, s, t) \tag{7}$$

Then, for sufficiently smooth $u_\theta^\star$ and v and for any $(s, t) \in [0, 1]^2$,

1. *$X_u(\cdot, s, t)$ is an invertible map everywhere,*
2. *$\log p_t^{u^\star}(x_t) = \log p_s^{u^\star}(x_s) - \log |\det \mathbf{J}_{X_u}(x_s)|$ almost everywhere.*

We provide a more precise statement and proofs for all propositions in §A. This means that *optimal* flow maps are, in some ways, ideal Boltzmann Generators in that they have relatively efficient access to both samples and likelihood; however, this property only holds at the optima, and in the case that $X_u(\cdot, s, t) = X_v(\cdot, s, t)$ for all s, t, which in practice is extremely challenging to satisfy.

In practice, for standard flow map models, $X_u(\cdot, s, t) \neq X_v(\cdot, s, t)$ and we have no guarantee that X_u will be invertible, making efficient likelihood calculation all but impossible. However, we note that this condition is actually much stronger than we need for FALCON. For our uses, while we would like $X_u(\cdot, s, t)$ to be close to $X_v(\cdot, s, t)$, for accurate and efficient likelihood computation, we only require that X_u is invertible, not that it matches the particular invertible map defined by X_v. This leads us to define an additional invertibility loss:

$$\mathcal{L}_{\text{inv}}(\theta) = \mathbb{E}_{s,t,x_s} \|x_s - X_u(X_u(x_s, s, t), t, s)\|^2, \tag{8}$$

to be used in conjunction with the average velocity objective and flow matching objectives, $\mathcal{L}_{\text{cfm}}$, for a final loss comprised of three components:

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{cfm}}(\theta) + \lambda_{\text{avg}}\mathcal{L}_{\text{avg}}(\theta) + \lambda_r\mathcal{L}_{\text{inv}}(\theta), \tag{9}$$

with variants $\mathcal{L}_{\text{avg}}$ proposed below. Minimizing this loss has a less strict requirement for the correctness of the Boltzmann Generator specifically:

Proposition 2. *Let $u_\theta^\star$ be a minimizer of $\mathcal{L}_{inv}$ (Eq. (8)) with respect to some v. Then, for sufficiently smooth $u_\theta^\star$ and v and for any $(s,t) \in [0,1]^2$, $X_u(\cdot, s, t)$ is an invertible map everywhere, and $\log p_t^{u^\star}(x_t) = \log p_s^{u^\star}(x_s) - \log|\det \mathbf{J}_{X_u}(x_s)|$ almost everywhere.*

Thus, minimizing the invertibility loss is sufficient for valid Boltzmann Generation, even without exactly reproducing the continuous-time flow. Note that the proposition provides a constructive guarantee of invertibility, and, in practice, we only require the existence of an inverse, not its explicit form. This condition ensures that FALCON acts as a consistent generator of the target energy distribution, $\mathcal{E}(x)$, while benefiting from fast inference-time scalability.

FALCON Enables Scalable Architectures. Previous Boltzmann Generators based on continuous normalizing flows for molecular applications utilize small equivariant architectures (Klein et al., 2023a; Klein & Noe, 2024; Tan et al., 2025a; Aggarwal et al., 2025) *up to 2.3 million parameters*. These models are limited in their scale due to the high cost of inference with multi-step adaptive step size samplers, which are needed to control the error in the likelihood calculation. FALCON, by enabling relatively cheap few-step sampling, can greatly improve performance. Specifically, we use a standard diffusion transformer (DiT) network (Peebles & Xie, 2023) with an additional time embedding head. We also use a combination of data augmentation to enforce soft SO(3) (rotation) equivariance and subtraction of the mean to enforce translation invariance following (Tan et al., 2025a;b).

Formulations for $\mathcal{L}_{\text{avg}}$ in the context of Boltzmann Generators. Many forms of $\mathcal{L}_{\text{avg}}$ have been explored in the context of fast generation (Geng et al., 2025; Guo et al., 2025; Boffi et al., 2025a; Sabour et al., 2025); We discuss these losses in §B. In this work, we consider the following loss, which is equivalent to the MeanFlow loss of Geng et al. (2025), as well as the ESD objective in Boffi et al. (2025a). This choice is based on the superior performance of this loss in image experiments (Sabour et al., 2025), as well as its potential for efficient implementation, which we discuss below.

$$\mathcal{L}_{\text{avg}} \triangleq \mathbb{E}_{s,t,x_s} \left\| u_\theta(x_s, s, t) - \text{sg}\Big(v(x_s, s) - (t-s)\big(v(x_s,s)\partial_{x_s} u_\theta + \partial_s u_\theta\big)\Big)\right\|^2 \tag{10}$$

Note that since $x_s = sx_1 + (1-s)x_0$, we can directly use $v(x_s, s) = x_1 - x_0$.

Efficient Implementation. As noted in multiple previous works, $\mathcal{L}_{\text{avg}}$ can be efficiently implemented using a single Jacobian vector product (JVP) call using forward automatic differentiation. Specifically, we have:

$$u_\theta(x_s, s, t), \frac{du_\theta}{ds} = \text{jvp}(u_\theta, (x_s, s, t), (v_s, 1, 0))$$

where the jvp function takes a callable function, inputs, and a vector which is the vector part of the JVP.

Additionally, for this loss specifically, we can combine $\mathcal{L}_{\text{cfm}}$ and $\mathcal{L}_{\text{avg}}$ if we implement $v(x_s, s) = u_\theta(x_s, s, s)$, i.e. passing the same time to u, representing the instantaneous velocity. Specifically, we can implement the sum of the two losses by changing the distribution of s, t in Eq. 10 to include some percentage of the time when $s = t$ to correspond with some fraction of $\mathcal{L}_{\text{cfm}}$ loss.

Algorithm 1: Training FALCON

```
Input: Sampleable $p_0$ and $p_1$, regularization
       weight $\lambda_r$, network $u_\theta$
Output: The trained network $u_\theta$
while training do
    $(x_0, x_1) \sim p_0 \times p_1$;
    $x_s \leftarrow sx_1 + (1-s)x_0$;
    $v_s \leftarrow x_1 - x_0$;
    $u_\theta, \frac{\partial u_\theta}{\partial s} \leftarrow \text{jvp}(u_\theta, (x_s, s, t), (v_s, 1, 0))$;
    $u_{\text{tgt}} \leftarrow v_s - (t-s)\frac{\partial u_\theta}{\partial s}$;
    $\hat{x}_t \leftarrow x_s + (t-s)u_\theta$;
    $\hat{x}_s \leftarrow \hat{x}_t + (s-t)u_\theta(\hat{x}_t, t, s)$;
    $\mathcal{L}(\theta) \leftarrow$
      $\|u - \text{stopgrad}(u_{\text{tgt}})\|^2 + \lambda_r \|x_s - \hat{x}_s\|^2$;
    $\theta \leftarrow \text{update}(\theta, \nabla_\theta \mathcal{L}(\theta))$;
return $u_\theta$;
```

Our method is the first to require flow maps both in the forward and backward directions. We therefore need to consider the parameterization $u_\theta(x_s, t, s)$, i.e. the backwards direction, specifically at the discontinuity when $s = t$. When $t \to s^+$, then $u_\theta(x_s, s, t) = v(x_s, s)$, but when $t \to s^-$, then $u_\theta(x_s, s, t) = -v(x_s, s)$. To address this, we parameterize our flow map u_θ such that $u_\theta(x_s, s, t) = \text{sign}(t-s)h_\theta(x_s, s, t)$ (Sabour et al., 2025).

4 Experiments

In this section, we first demonstrate that FALCON achieves more scalable performance over state-of-the-art continuous flows across both global and local metrics on tri-alanine, alanine tetrapeptide, and hexa-alanine (Table 3). Next, we empirically demonstrate that FALCON flows exceed the performance of state-of-the-art discrete NFs, even when they are given vastly larger sampling budgets (Fig. 4). Then, we elucidate the importance of regularization in achieving invertibility and aiding generative performance (Fig. 6). Finally, we ablate inference schedules and show their impact on performance across metrics as a function of sampler choice (Fig. 7).

4.1 Experimental Setup

Datasets. We evaluate the performance of FALCON on equilibrium conformation sampling tasks, focusing on alanine dipeptide (ALDP), tri-alanine (AL3), alanine tetrapeptide (AL4), and hexa-alanine (AL6). Datasets are obtained from implicit solvent molecular dynamics (MD) simulations with the `amber-14` force field, as detailed in §D.3. We train on biased data and test on a held-out unbiased dataset, using self-normalized importance sampling (SNIS) and force-field energies to compute log-likelihoods and re-sample molecules from the target Boltzmann density.

Baselines. We benchmark FALCON against both discrete and continuous normalizing flows. We include four discrete normalizing flow baselines: (**1**) SE(3)-EACF (Midgley et al., 2023); (**2**) RegFlow (Rehman et al., 2025); (**3**) SBG (Tan et al., 2025a) with standard SNIS (SBG IS); and (**4**) SBG with SMC sampling (SBG SMC), as well as three continuous flows: (**1**) ECNF (Klein et al., 2023b); (**2**) ECNF++ (Tan et al., 2025a); and (**3**) BoltzNCE (Aggarwal et al., 2025), a recent SE(3)-equivariant architecture leveraging geometric vector perceptrons (GVPs) (Jing et al., 2020) on alanine dipeptide. For all continuous flows, samples and likelihoods are generated by integrating over the vector field using the Dormand–Prince 4(5) integrator with atol $=$ rtol $= 10^{-5}$ (Dormand & Prince, 1986) to ensure a fair comparison between methods. More details on architectures and parameters are covered in §D.1.

Metrics. We report Effective Sample Size (ESS), and the 2-Wasserstein distance on both the energy distribution ($\mathcal{E}$-$\mathcal{W}_2$), and dihedral angles ($\mathbb{T}$-$\mathcal{W}_2$). The full definitions of the metrics are included in §E. The energy captures local details, as minor atomic displacements yield large variations in the energy distribution, while $\mathbb{T}$-$\mathcal{W}_2$ captures global structure via mode coverage across metastable states. We include energy histograms in the main text, with Ramachandran plots relegated to §F.4. For robustness, all quantitative experiments are performed on three seeds of the model and reported as mean $\pm$ standard deviation in the tables and figures. For all benchmarks, in cases where dashes are present, data was unavailable, except for SBG SMC (Tan et al., 2025a), where ESS is not a valid metric.

4.2 FALCON Outperforms State-of-the-Art Methods

Superior Scalability Over Continuous Flows. Computing likelihoods in CNFs is computationally prohibitive, limiting their scalability in the Boltzmann Generator setting. Although the current state-of-the-art, ECNF++, performs exceptionally well on ESS and $\mathbb{T}$-$\mathcal{W}_2$ for alanine dipeptide (see Table 2) (Tan et al., 2025a), it fails to scale to larger molecules, as seen in Table 3. In contrast, for larger systems—tri-alanine, alanine tetrapeptide, and hexa-alanine—FALCON substantially outperforms ECNF++ across all metrics, demonstrating superior scalability to larger molecular systems. The true MD energy distributions, learned proposals, and re-sampled energies for alanine dipeptide, tri-alanine, alanine tetrapeptide, and hexa-alanine are all shown in Fig. 3.

Table 2: Results on alanine dipeptide.

	Alanine dipeptide (ALDP)		
Algorithm ↓	ESS ↑	$\mathcal{E}$-$\mathcal{W}_2$ ↓	$\mathbb{T}$-$\mathcal{W}_2$ ↓
BoltzNCE	—	0.27 ± 0.02	0.57 ± 0.00
SE(3)-EACF	$< 10^{-3}$	108.202	2.867
RegFlow	0.036	0.519	0.958
ECNF	0.119	0.419	0.311
ECNF++	**0.275** $\pm$ **0.010**	0.914 ± 0.122	**0.189** $\pm$ **0.019**
SBG IS	0.030 ± 0.012	0.873 ± 0.338	0.439 ± 0.129
SBG SMC	—	0.741 ± 0.189	0.431 ± 0.141
FALCON (Ours)	0.067 ± 0.013	**0.225** $\pm$ **0.104**	0.402 ± 0.021

Improved Sample Quality Over Discrete Flows. Discrete NFs have recently been shown to be highly performant Boltzmann Generators (Rehman et al., 2025; Tan et al., 2025a;b). SBG (Tan et al., 2025a), based on the TARFlow architecture (Zhai et al., 2024), outperforms all previously reported methods across both global and local metrics (see Table 3). Here, we demonstrate that FALCON, even in a few steps, can outperform SBG across all metrics. We make two main claims to assert FALCON's competitive advantage, in comparison with discrete NFs: (**1**) Discrete NFs, despite being fast one-step generators, consistently underperform compared to FALCON across all global and local

Table 3: Quantitative results on tri-alanine (AL3), alanine tetrapeptide (AL4), and hexa-alanine (AL6). Baseline methods presented with SNIS, unless stated otherwise. Evaluations are conducted over 10^4 samples.

	Tri-alanine (AL3)			Tetrapeptide (AL4)			Hexa-alanine (AL6)		
Algorithm ↓	ESS ↑	$\mathcal{E}$-$\mathcal{W}_2$ ↓	$\mathbb{T}$-$\mathcal{W}_2$ ↓	ESS ↑	$\mathcal{E}$-$\mathcal{W}_2$ ↓	$\mathbb{T}$-$\mathcal{W}_2$ ↓	ESS ↑	$\mathcal{E}$-$\mathcal{W}_2$ ↓	$\mathbb{T}$-$\mathcal{W}_2$ ↓
ECNF++	0.003 ± 0.002	2.206 ± 0.813	0.962 ± 0.253	0.016 ± 0.001	5.638 ± 0.483	1.002 ± 0.061	0.006 ± 0.001	10.668 ± 0.285	1.902 ± 0.055
RegFlow	0.029	1.051	1.612	0.010	6.277	3.476	—	—	—
SBG IS	0.052 ± 0.013	0.758 ± 0.506	0.502 ± 0.016	0.046 ± 0.014	1.068 ± 0.495	0.969 ± 0.067	0.034 ± 0.015	1.021 ± 0.239	1.431 ± 0.085
SBG SMC	—	0.598 ± 0.084	0.503 ± 0.029	—	1.007 ± 0.382	1.039 ± 0.069	—	1.189 ± 0.357	1.444 ± 0.140
FALCON (Ours)	**0.077 ± 0.004**	**0.544 ± 0.013**	**0.452 ± 0.011**	**0.055 ± 0.003**	**0.686 ± 0.047**	**0.858 ± 0.077**	**0.060 ± 0.017**	**0.892 ± 0.311**	**1.256 ± 0.132**

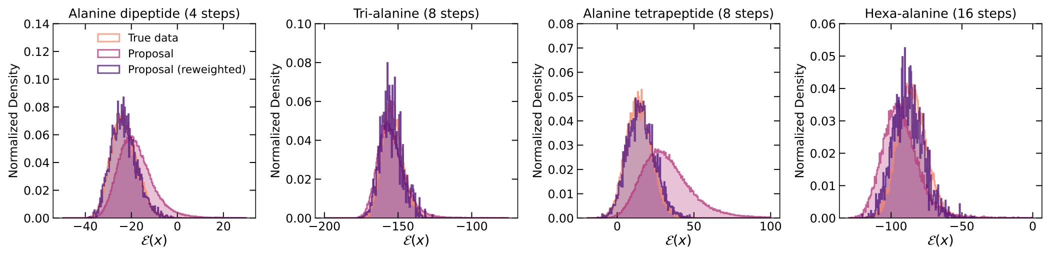

Figure 3: True MD energy distribution with best FALCON unweighted and re-sampled proposals for alanine dipeptide (**left**), tri-alanine (**center left**), and alanine tetrapeptide (**center right**), and hexa-alanine (**right**).

metrics (Table 3); and (**2**) Increasing the number of samples can partially close this gap; however, even with 5×10^6 samples—250× more than those used to evaluate FALCON—SBG's performance on $\mathcal{E}$-$\mathcal{W}_2$ remains significantly worse than that of a 4-step FALCON Flow (as demonstrated in Fig. 4).

4.3 Analysis of Computational Efficiency

Training Efficiency Compared to Discrete NFs. Discrete NFs benefit from fast inference, but are slow and unstable to train due to the maximum likelihood objective (Xu & Campbell, 2023; Andrade, 2024). By contrast, CNFs trained with a flow matching objective trade more stable and faster training for slower inference. When considering both training and evaluation time together (Table 4), we see that FALCON—despite being marginally slower at inference than the discrete NFs for the same number of samples—achieves faster cumulative training + inference times for superior performance due to the expedited training objective.

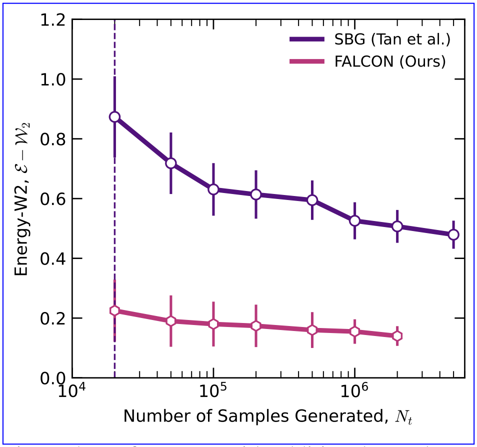

Figure 4: Performance with additional samples.

Inference Efficiency Compared to CNFs. A primary contribution of FALCON is the dramatic reduction in the computational cost required to achieve high-quality samples with accurate likelihoods. Fig. 2 directly illustrates this advantage. To reach a comparable level of performance on the $\mathbb{T}$-$\mathcal{W}_2$ metric for alanine dipeptide, a traditional CNF requires inference times that are two orders of magnitude longer than FALCON. This efficiency gain is what enables the use of larger, more expressive architectures and makes large-scale molecular sampling practical. For additional discussion, see §F.3.

Table 4: Cumulative training + inference time across flows. 10^4 samples evaluated with (atol = rtol = 10^{-5} for our CNF and 4-step FALCON). All experiments were conducted on one NVIDIA L40S with batch size 1024. Note: For hexa-alanine, obtaining 10^4 samples from the Dopri5-integrated CNF was computationally infeasible.

	ECNF++	SBG	DiT CNF (Ours)	FALCON (Ours)
Alanine dipeptide	12.52	16.83	9.56	7.65
Tri-alanine	19.59	24.67	17.54	11.45
Alanine tetrapeptide	32.17	41.67	24.10	18.84
Hexa-alanine	137.4	57.50	82.10	25.76

Table 5: Quantitative results on alanine dipeptide, tri-alanine, alanine tetrapeptide, and hexa-alanine compared to our Dopri5-integrated CNFs. Evaluations were conducted over 10^4 points across methods. Note: For hexa-alanine, obtaining 10^4 samples from the Dopri5-integrated CNF was computationally infeasible.

System	Algorithm ↓	ESS ↑	$\mathcal{E}$-$\mathcal{W}_2$ ↓	$\mathbb{T}$-$\mathcal{W}_2$ ↓	NFE
Alanine dipeptide	FALCON-Dopri5	0.264 ± 0.058	0.442 ± 0.048	0.218 ± 0.023	257
	FALCON	0.067 ± 0.013	0.225 ± 0.104	0.402 ± 0.021	4
Tri-alanine	FALCON-Dopri5	0.125 ± 0.034	0.382 ± 0.053	0.370 ± 0.093	265
	FALCON	0.077 ± 0.004	0.544 ± 0.013	0.452 ± 0.011	8
Alanine tetrapeptide	FALCON-Dopri5	0.129 ± 0.015	0.665 ± 0.047	0.640 ± 0.093	200
	FALCON	0.055 ± 0.003	0.686 ± 0.047	0.858 ± 0.077	8
Hexa-alanine	FALCON-Dopri5	0.128 ± 0.031	1.013 ± 0.115	1.320 ± 0.201	207
	FALCON	0.060 ± 0.017	0.892 ± 0.311	1.256 ± 0.132	16

Inference vs. Accuracy Trade-off. In FALCON, by using a flow map formulation, we can trade off performance for faster evaluation by adjusting the number of inference steps post-hoc. In Table 5, we show that a high-NFE, adaptive step solver achieves superior performance to the state-of-the-art continuous time ECNF++, as well as a few-step FALCON Flow; however, we also demonstrate that in this few-step regime, FALCON still outperforms every method considered (see Table 3) using two orders of magnitude fewer function evaluations (only 4-16 steps, depending on the dataset). Depending on the compute budget and goal, we demonstrate in Fig. 5 how FALCON is able to interpolate between slow and accurate sampling with fast but less accurate sampling.

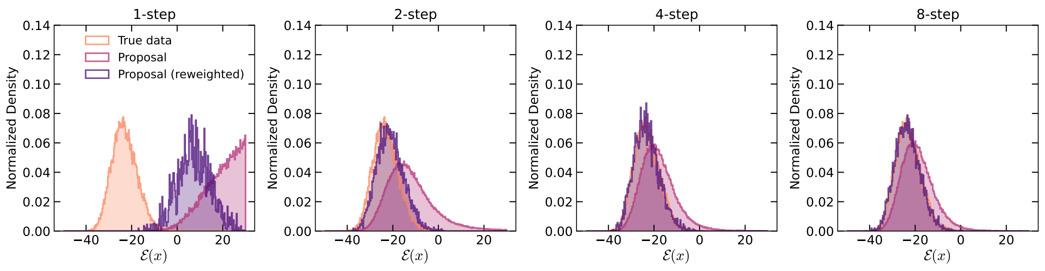

Figure 5: Improved proposal and re-weighted sample energies with increased steps for alanine dipeptide.

4.4 Ablation Studies and Design Choices

Verifying FALCON's Invertibility. As CNFs are only invertible at convergence, we introduce a regularization term in the loss to promote numerical invertibility in the few-step regime. In Fig. 6, we demonstrate the trade-off from this term on both the ESS and 2-Wasserstein distance on dihedral angles: weak regularization leads to poor invertibility and degraded performance, whereas strong regularization enforces flow invertibility, albeit at the cost of reduced sample quality. We fix the regularization constant to 10.0 for all experiments performed, unless stated otherwise to balance performance and numerical invertibility.

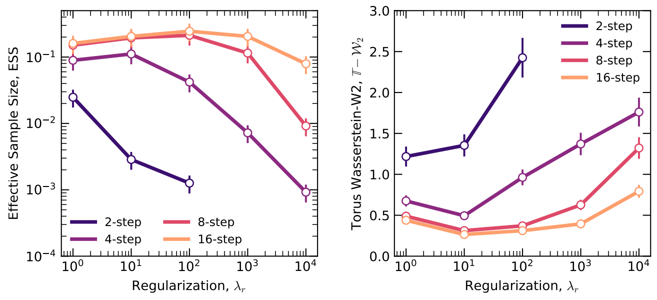

Figure 6: Performance trade-off with increasing regularization.

We also directly prove that an inverse exists for our trained flow, by training an auxiliary network to invert a frozen FALCON Flow for the forward direction. We find that FALCON achieves invertibility errors on the order of 10^{-4}, which is the same order of magnitude as the invertibility of discrete and continuous NFs. Additional details can be found in §F.2 and Fig. 12.

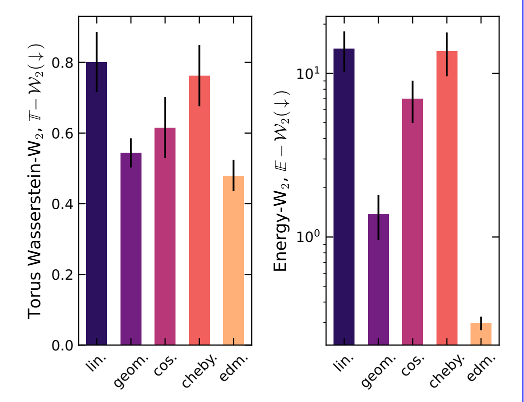

Figure 7: Performance vs. choice of inference schedule.

Impact of Inference Schedules. In the few step regime, performance can be significantly impacted by the choice of inference schedule. We run ablations on various schedules for alanine dipeptide with 8-steps, summarizing the results in Fig. 7. We note that the EDM scheduler substantially outperforms all other schedulers, in agreement with observations from the diffusion literature (Karras et al., 2022); sampling more points near the data distribution proves beneficial in aiding generative performance as the variance of the flow field is higher closer to the target distribution. For all reported results, we use the EDM scheduler. In §D.2, we provide additional details regarding scheduler definitions, inference setups, and parameter selection for EDM.

5 Related Work

Boltzmann Generators. Boltzmann Generators (BGs) (Noé et al., 2019) are used to sample molecular conformations (Klein et al., 2023a) and enable consistent estimates of thermodynamic observables (Wirnsberger et al., 2020; Rizzi et al., 2023; Schebek et al., 2024). While traditionally BGs are based on discrete normalizing flows, more recent work in machine learning makes use of more powerful continuous normalizing flow architectures for invariance (Köhler et al., 2020; Köhler et al., 2023) and expressive power (Klein et al., 2023a; Klein & Noe, 2024). A few other works have explored the usage of approximate likelihoods (Draxler et al., 2024; Sorrenson et al., 2024; Aggarwal et al., 2025), but have until now been unable to scale. Rehman et al. (2025) also proposes a more efficient BG using a new-regression-based objective to train discrete normalizing flow architectures, but requires an invertible architecture limiting scalability and performance.

Few-step Flows. Diffusion and flow matching methods are now central to domains from vision (Song et al., 2021a;b; Lipman et al., 2023) to scientific applications in material and drug discovery (Abramson et al., 2024; Noé et al., 2019). Scalable regression-based losses make these models fast to train, yet inference remains costly due to the need for numerous vector field integrations, motivating efforts to reduce computational expense. One- and few-step methods, like consistency models (Song et al., 2023; Song & Dhariwal, 2023; Geng et al., 2024), shortcut models (Frans et al., 2025), and flow maps (Boffi et al., 2025a;b; Sabour et al., 2025; Guo et al., 2025) have gained recent attention. Although there have been numerous efforts in improving generative performance in image settings, developing invertible few-step flows for scientific applications has seen far less interest. Our work, as far as we know, is the first demonstration of an invertible few-step flow with fast likelihoods.

6 Conclusion

In this work, we introduced Few-step Accurate Likelihoods for Continuous Flows, a novel few-step flow-based generative model designed to address the long-standing challenge of scalable and efficient Boltzmann distribution sampling. Our approach successfully combines the expressiveness and training efficiency of modern flow-based models with a few-step sampling capability and for fast, accurate likelihood estimation. By leveraging a hybrid training objective, FALCON provides a practical solution for the computationally expensive likelihood evaluations that have historically limited the widespread adoption of Boltzmann Generators.

Our empirical results demonstrate that FALCON not only outperforms the existing state-of-the-art discrete normalizing flow models, but also provides a significant leap in computational efficiency over previous continuous flow models. We showed that our model is *two orders of magnitude faster* than an equivalently performing CNF-based Boltzmann Generator, making real-world, molecular sampling applications significantly more feasible. This represents a critical step toward unlocking the potential of Boltzmann Generators in fields ranging from drug discovery to materials science.

Limitations. Despite its advancements, FALCON has several key limitations that are crucial to acknowledge. First, while our results demonstrate that the computed likelihoods are *empirically good enough* for practical applications, we cannot efficiently guarantee their theoretical correctness. This represents a trade-off between computational efficiency and absolute theoretical certainty.

Additionally, while theoretically possible, achieving true one-step generation remains out of reach for our current models, and we believe further architectural improvements and training methodologies are necessary to fully realize this potential.

Finally, our current research has primarily focused on the application of FALCON to Boltzmann Generation in molecular conformation sampling. Future work will explore applying our approach to Bayesian inference, robotics, and other domains where rapid and accurate likelihood estimation is critical. We also see potential in models with structured Jacobians (Rezende & Mohamed, 2015; Dinh et al., 2017; Zhai et al., 2024; Kolesnikov et al., 2024) to facilitate even faster sampling.

Ethics Statement

Our work is primarily focused on theoretical algorithmic development for faster and more accurate generative models for sampling from Boltzmann densities, with reduced focus on experimental implementation. However, we recommend that future users of our work exercise appropriate caution when applying it to domains that may involve sensitive considerations.

Reproducibility Statement

We undertake multiple measures to ensure the reproducibility of our work. A dedicated section in §F.1 outlines the setup required to generate each of our reported figures. Further, we provide comprehensive information on the MD datasets used to train our models, including simulation parameters as well as the training, validation, and test splits used. We also include a separate section detailing model configurations, learning rate schedules, optimizer settings, hyperparameter choices, and other relevant aspects to facilitate reproduction in §D.1. We will also release all developed code publicly upon acceptance.

References

Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J. Ballard, Joshua Bambrick, Sebastian W. Bodenstein, David A. Evans, Chia-Chun Hung, Michael O'Neill, David Reiman, Kathryn Tunyasuvunakool, Zachary Wu, Akvilė Žemgulytė, Eirini Arvaniti, Charles Beattie, Ottavia Bertolli, Alex Bridgland, Alexey Cherepanov, Miles Congreve, Alexander I. Cowen-Rivers, Andrew Cowie, Michael Figurnov, Fabian B. Fuchs, Hannah Gladman, Rishub Jain, Yousuf A. Khan, Caroline M. R. Low, Kuba Perlin, Anna Potapenko, Pascal Savy, Sukhdeep Singh, Adrian Stecula, Ashok Thillaisundaram, Catherine Tong, Sergei Yakneen, Ellen D. Zhong, Michal Zielinski, Augustin Žídek, Victor Bapst, Pushmeet Kohli, Max Jaderberg, Demis Hassabis, and John M. Jumper. Accurate structure prediction of biomolecular interactions with AlphaFold 3. *Nature*, 630(8016):493–500, June 2024. ISSN 1476-4687. doi: 10.1038/s41586-024-07487-w. URL https://www.nature.com/articles/s41586-024-07487-w. Publisher: Nature Publishing Group.

Rishal Aggarwal, Jacky Chen, Nicholas M Boffi, and David Ryan Koes. BoltzNCE: Learning likelihoods for boltzmann generation with stochastic interpolants and noise contrastive estimation. *arXiv preprint arXiv:2507.00846*, 2025.

Tara Akhound-Sadegh, Jarrid Rector-Brooks, Avishek Joey Bose, Sarthak Mittal, Pablo Lemos, Cheng-Hao Liu, Marcin Sendera, Siamak Ravanbakhsh, Gauthier Gidel, Yoshua Bengio, Nikolay Malkin, and Alexander Tong. Iterated Denoising Energy Matching for Sampling from Boltzmann Densities, February 2024. URL http://arxiv.org/abs/2402.06121. arXiv:2402.06121 [cs, stat].

Tara Akhound-Sadegh, Jungyoon Lee, Avishek Joey Bose, Valentin De Bortoli, Arnaud Doucet, Michael M. Bronstein, Dominique Beaini, Siamak Ravanbakhsh, Kirill Neklyudov, and Alexander Tong. Progressive inference-time annealing of diffusion models for sampling from boltzmann densities. In *Advances in Neural Information Processing Systems*, 2025.

Michael S. Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *International Conference on Learning Representations (ICLR)*, 2023.

Daniel Andrade. Stable training of normalizing flows for high-dimensional variational inference. *arXiv preprint arXiv:2402.16408*, 2024.

Nicholas M. Boffi, Michael S. Albergo, and Eric Vanden-Eijnden. How to build a consistency model: Learning flow maps via self-distillation, 2025a. URL https://arxiv.org/abs/2505.18825.

Nicholas Matthew Boffi, Michael Samuel Albergo, and Eric Vanden-Eijnden. Flow map matching with stochastic interpolants: A mathematical framework for consistency models. *Transactions on Machine Learning Research*, 2025b.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Neural Information Processing Systems (NIPS)*, 2018.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. *International Conference on Learning Representations (ICLR)*, 2017.

John R Dormand and Peter J Prince. Runge-kutta triples. *Computers & Mathematics with Applications*, 12(9):1007–1017, 1986.

Felix Draxler, Peter Sorrenson, Lea Zimmermann, Armand Rousselot, and Ullrich Köthe. Free-form flows: Make any architecture a normalizing flow. In *International Conference on Artificial Intelligence and Statistics*, pp. 2197–2205. PMLR, 2024.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models, 2025. URL https://arxiv.org/abs/2410.12557.

Daan Frenkel and Berend Smit. *Understanding Molecular Simulation: from Algorithms to Applications*. Elsevier, 2023.

Zhengyang Geng, Ashwini Pokle, William Luo, Justin Lin, and J Zico Kolter. Consistency models made easy. *arXiv preprint arXiv:2406.14548*, 2024.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J. Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling, 2025. URL https://arxiv.org/abs/2505.13447.

Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. FFJORD: free-form continuous dynamics for scalable reversible generative models. In *International Conference on Representation Learning (ICLR)*, 2019.

Yi Guo, Wei Wang, Zhihang Yuan, Rong Cao, Kuan Chen, Zhengyang Chen, Yuanyuan Huo, Yang Zhang, Yuping Wang, Shouda Liu, and Yuxuan Wang. Splitmeanflow: Interval splitting consistency in few-step generative modeling, 2025. URL https://arxiv.org/abs/2507.16884.

Aaron Havens, Benjamin Kurt Miller, Bing Yan, Carles Domingo-Enrich, Anuroop Sriram, Brandon Wood, Daniel Levine, Bin Hu, Brandon Amos, Brian Karrer, Xiang Fu, Guan-Horng Liu, and Ricky T. Q. Chen. Adjoint sampling: Highly scalable diffusion samplers via adjoint matching, 2025. URL https://arxiv.org/abs/2504.11713.

M.F. Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics - Simulation and Computation*, 19(2):433–450, 1990. doi: 10.1080/03610919008812866.

Bowen Jing, Stephan Eismann, Patricia Suriana, Raphael JL Townshend, and Ron Dror. Learning from protein structure with geometric vector perceptrons. *arXiv preprint arXiv:2009.01411*, 2020.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

Leslie Kish. Confidence intervals for clustered samples. *American Sociological Review*, 22(2): 154–165, 1957.

Leon Klein and Frank Noe. Transferable Boltzmann Generators. *Advances in Neural Information Processing Systems*, 37:45281–45314, December 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5035a409f5798e188079e236f437e522-Abstract-Conference.html.

Leon Klein, Andreas Krämer, and Frank Noe. Equivariant flow matching. *Advances in Neural Information Processing Systems*, 36:59886–59910, December 2023a. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/bc827452450356f9f558f4e4568d553b-Abstract-Conference.html.

Leon Klein, Andreas Krämer, and Frank Noé. Equivariant flow matching. *Neural Information Processing Systems (NeurIPS)*, 2023b.

Jonas Köhler, Leon Klein, and Frank Noé. Equivariant flows: exact likelihood generative learning for symmetric densities. *International Conference on Machine Learning (ICML)*, 2020.

Alexander Kolesnikov, André Susano Pinto, and Michael Tschannen. Jet: A modern transformer-based normalizing flow. *arXiv preprint arXiv:2412.15129*, 2024.

Jonas Köhler, Michele Invernizzi, Pim De Haan, and Frank Noe. Rigid Body Flows for Sampling Molecular Crystal Structures. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 17301–17326. PMLR, July 2023. URL https://proceedings.mlr.press/v202/kohler23a.html. ISSN: 2640-3498.

Benedict Leimkuhler and Charles Matthews. *Molecular Dynamics: With Deterministic and Stochastic Numerical Methods*. Springer, 2015.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *International Conference on Learning Representations (ICLR)*, 2023.

Jun S Liu. *Monte Carlo Strategies in Scientific Computing*. Springer, 2001.

Qiang Liu. Rectified flow: A marginal preserving approach to optimal transport. *arXiv*, 2022.

Nanye Ma, Mark Goldstein, Michael S. Albergo, Nicholas M. Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *Computer Vision – ECCV 2024*, 2024.

Laurence Illing Midgley, Vincent Stimper, Gregor NC Simm, Bernhard Schölkopf, and José Miguel Hernández-Lobato. Flow annealed importance sampling bootstrap. *International Conference on Learning Representations (ICLR)*, 2023.

Frank Noé, Simon Olsson, Jonas Köhler, and Hao Wu. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365(6457):eaaw1147, 2019.

Kaoru Ohno, Keivan Esfarjani, and Yoshiyuki Kawazoe. *Computational Materials Science: From Ab Initio to Monte Carlo Methods*. Springer, 2018.

William Peebles and Saining Xie. Scalable Diffusion Models with Transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182, Paris, France, October 2023. IEEE. ISBN 9798350307184. doi: 10.1109/ICCV51070.2023.00387. URL https://ieeexplore.ieee.org/document/10377858/.

Stefano Peluchetti. Non-denoising forward-time diffusions, 2021.

Aram-Alexandre Pooladian, Heli Ben-Hamu, Carles Domingo-Enrich, Brandon Amos, Yaron Lipman, and Ricky T. Q. Chen. Multisample flow matching: Straightening flows with minibatch couplings. In *International Conference on Machine Learning*, 2023.

Danyal Rehman, Oscar Davis, Jiarui Lu, Jian Tang, Michael Bronstein, Yoshua Bengio, Alexander Tong, and Avishek Joey Bose. Efficient regression-based training of normalizing flows for boltzmann generators, 2025. URL https://arxiv.org/abs/2506.01158.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. *International Conference on Machine Learning (ICML)*, 2015.

Andrea Rizzi, Paolo Carloni, and Michele Parrinello. Free energies at qm accuracy from force fields via multimap targeted estimation. *Proceedings of the National Academy of Sciences*, 2023.

Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. Align your flow: Scaling continuous-time flow map distillation, 2025. URL https://arxiv.org/abs/2506.14603.

Maximilian Schebek, Michele Invernizzi, Frank Noé, and Jutta Rogal. Efficient mapping of phase diagrams with conditional boltzmann generators. *Machine Learning: Science and Technology*, 2024.

Yuyang Shi, Valentin De Bortoli, Andrew Campbell, and Arnaud Doucet. Diffusion schrödinger bridge matching. In *Advances in Neural Information Processing Systems*, 2023.

Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. *arXiv preprint arXiv:2310.14189*, 2023.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021a.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations, 2021b. URL https://arxiv.org/abs/2011.13456.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, 2023.

Peter Sorrenson, Felix Draxler, Armand Rousselot, Sander Hummerich, Lea Zimmermann, and Ullrich Köthe. Lifting architectural constraints of injective flows. In *International Conference on Learning Representations*, 2024.

Gabriel Stoltz, Mathias Rousset, and Tony Lelièvre. *Free Energy Computations: A Mathematical Perspective*. World Scientific, 2010.

Charlie B. Tan, Avishek Joey Bose, Chen Lin, Leon Klein, Michael M. Bronstein, and Alexander Tong. Scalable Equilibrium Sampling with Sequential Boltzmann Generators, February 2025a. URL http://arxiv.org/abs/2502.18462. arXiv:2502.18462 [cs].

Charlie B. Tan, Majdi Hassan, Leon Klein, Saifuddin Syed, Dominique Beaini, Michael M. Bronstein, Alexander Tong, and Kirill Neklyudov. Amortized sampling with transferable normalizing flows. In *Advances in Neural Information Processing Systems*, 2025b.

Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, 2024a.

Alexander Tong, Nikolay Malkin, Kilian Fatras, Lazar Atanackovic, Yanlei Zhang, Guillaume Huguet, Guy Wolf, and Yoshua Bengio. Simulation-free schrödinger bridges via score and flow matching. In *AISTATS*, 2024b.

Francisco Vargas, Will Grathwohl, and Arnaud Doucet. Denoising diffusion samplers. *International Conference on Learning Representations (ICLR)*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

Peter Wirnsberger, Andrew J Ballard, George Papamakarios, Stuart Abercrombie, Sébastien Racanière, Alexander Pritzel, Danilo Jimenez Rezende, and Charles Blundell. Targeted free energy estimation via learned mappings. *J. Chem. Phys.*, 2020.

Zuheng Xu and Trevor Campbell. Embracing the chaos: analysis and diagnosis of numerical instability in variational flows. *Advances in Neural Information Processing Systems*, 36:32360–32386, 2023.

Shuangfei Zhai, Ruixiang Zhang, Preetum Nakkiran, David Berthelot, Jiatao Gu, Huangjie Zheng, Tianrong Chen, Miguel Angel Bautista, Navdeep Jaitly, and Josh Susskind. Normalizing Flows are Capable Generative Models, December 2024. URL http://arxiv.org/abs/2412.06329. arXiv:2412.06329 [cs].

Qinsheng Zhang and Yongxin Chen. Path integral sampler: a stochastic control approach for sampling. *International Conference on Learning Representations (ICLR)*, 2022.

APPENDIX

A THEORY

Proposition 1. *Let $u_\theta^\star$ be a minimizer of Eq. 5 with respect to some v. Also, define the Jacobian of X as $\mathbf{J}_X = \frac{\partial X}{\partial x_s}$, and the discrete flow map:*

$$X_u(x_s, s, t) = x_s + (t - s)u_\theta^\star(x_s, s, t) \tag{7}$$

Then, for sufficiently smooth $u_\theta^\star$ and v and for any $(s, t) \in [0, 1]^2$,

1. *$X_u(\cdot, s, t)$ is an invertible map everywhere,*
2. *$\log p_t^{u^\star}(x_t) = \log p_s^{u^\star}(x_s) - \log |\det \mathbf{J}_{X_u}(x_s)|$ almost everywhere.*

We first define what we mean by sufficiently smooth in both propositions.

Assumption 1. *$u_\theta^\star$ is C^1 almost everywhere.*

This assumption on $u_\theta^\star$ allows the application of the change of variables formula almost everywhere, which is necessary to compute the log likelihood under u. We note that this is satisfied by most sufficiently expressive modern architectures including those using ReLU type activations, which are not C^1 everywhere, but are almost everywhere.

Assumption 2. *v is uniformly Lipschitz continuous in x and continuous in t.*

This assumption on v is necessary in order to satisfy the Picard-Lindelöf Theorem. We note that if the Lipschitz condition does not hold then the Peano existence theorem implies that the initial value problem (IVP) of the ODE may not be invertible. If, however, we have both continuity in x and t as well as a Lipschitz condition for v on x, then by the Picard-Lindelöf Theorem, we have existence and uniqueness of the IVP. We first recall Picard-Lindelöf.

Theorem 1 (Picard-Lindelöf). *Let $D \subset \mathbb{R}^d \times \mathbb{R}$ be a closed rectangle with $(x_0, t_0) \in$ int D, the interior of D. Let $v : D \to \mathbb{R}^d$ be a function that is continuous in t and Lipschitz continuous in x (with Lipschitz constant independent from t). Then there exists some $\epsilon > 0$ such that the initial value problem is:*

$$\frac{dx}{dt} = v(x_t, t), \quad x(t_0) = x_0 \tag{11}$$

has a unique solution $x(t)$ on the interval $[t_0 - \epsilon, t_0 + \epsilon]$.

Under these conditions on v we are now able to prove the proposition.

Proof. Recall that if $u_\theta^\star$ minimizes Eq. 5:

$$\mathbb{E}_{s,t,x_s}\left[w(s,t)\|u_\theta(x_s, s, t) - \frac{1}{t-s}\int_s^t v(x_\tau, \tau)d\tau\|^2\right]$$

then we have that $u_\theta^\star(x_s, s, t) = \frac{1}{t-s}\int_s^t v(x_\tau, \tau)d\tau$ for all $s, t \in [0, 1]^2$. Furthermore we have

$$\begin{aligned} X_u(x_s, s, t) &= x_s + (t - s)u_\theta^\star(x_s, s, t) && (12)\\ &= x_s + (t - s)\frac{1}{t-s}\int_s^t v(x_\tau, \tau)d\tau && (13)\\ &= x_s + \int_s^t v(x_\tau, \tau)d\tau && (14) \end{aligned}$$

which by application of the Picard-Lindelöf theorem is invertible for all s, t, proving part 1 of the proposition.

To prove part 2, we note that Eq. 14 is differentiable with respect to x_s and X_s is invertible, therefore by the change-of-variables formula and Assumption 1, we arrive at part 2 of the proposition. □

Proposition 2. *Let $u_\theta^\star$ be a minimizer of $\mathcal{L}_{inv}$ (Eq. (8)) with respect to some v. Then, for sufficiently smooth $u_\theta^\star$ and v and for any $(s, t) \in [0, 1]^2$, $X_u(\cdot, s, t)$ is an invertible map everywhere, and $\log p_t^{u^\star}(x_t) = \log p_s^{u^\star}(x_s) - \log |\det \mathbf{J}_{X_u}(x_s)|$ almost everywhere.*

Proof. First we recall Eq. 8:

$$\mathcal{L}_{\text{inv}} = \mathbb{E}_{s,t,x_s}\|x_s - X_u(X_u(x_s, s, t), t, s)\|^2 \tag{15}$$

In this case, the minimizer of $\mathcal{L}_{\text{inv}}$ that is also in C^1 is pointwise invertible by definition as $X_u(\cdot, t, s)$ is the inverse of $X_u(\cdot, s, t)$ as $\mathcal{L}_{\text{inv}} \to 0$ everywhere.

For the change of variables to apply, we also need Assumption 1 to apply almost everywhere. $\square$

We note that this loss ensures *pointwise* invertibility. Additional restrictions are needed to ensure that the log likelihood can be calculated efficiently.

B OTHER FORMULATIONS FOR $\mathcal{L}_{\text{AVG}}$

There are various formulations of $\mathcal{L}_{\text{avg}}$ that have been explored in the recent literature. We note that FALCON directly benefits from any new advancements in this rapidly-evolving research direction. Specifically, the following three different $\mathcal{L}_{\text{avg}}$ losses can be used for training flow maps.

1. $\mathcal{L}_1 \triangleq \mathbb{E}_{s,t,x_s}\left\| u_\theta(s,t,x_s) - \text{sg}\left(v(x_s,s) - (t-s)\big(v(x_s,s)\partial_{x_s}u_\theta + \partial_s u_\theta\big)\right)\right\|^2$, which is equivalent to the MeanFlow loss of Geng et al. (2025), as well as the ESD objective in Boffi et al. (2025a). Note that since $x_s = sx_1 + (1-s)x_0$, we can directly use $v(x_s,s) = x_1 - x_0$.
2. $\mathcal{L}_2 \triangleq \mathbb{E}_{s,t,x_s}\left\| u_\theta(s,t,x_s) - \text{sg}\Big(\lambda u_\theta(x_s,s,r) + (1-\lambda)u_\theta(X_u(x_s,s,r),r,t)\Big)\right\|^2$, for $r = \lambda s + (1-\lambda)t$, is equivalent to the SplitMeanFlow loss of Guo et al. (2025), as well as the scaled PSD objective in Boffi et al. (2025a). In Boffi et al. (2025a), two ways of sampling the intermediate time r are explored. Deterministic mid-point sampling, which sets $\lambda = 0.5$, as well as uniform sampling of λ. Guo et al. (2025) only explores uniform sampling. This is also a generalization of Frans et al. (2025), which additionally samples s and t in a binary-tree fashion.
3. $\mathcal{L}_3 \triangleq \mathbb{E}_{s,t,x_s}\left\| \partial_t X_u(x_s,s,t) - \text{sg}\Big(u_\theta\big(X_u(x_s,s,t),t,t\big)\Big)\right\|^2$ is the Lagrangian objective presented in Boffi et al. (2025a).

As discussed in §3, we choose $\mathcal{L}_{\text{avg}} \triangleq \mathcal{L}_1$ in this paper. This choice is based on the superior performance of this loss compared to $\mathcal{L}_3$ in the image experiments of Sabour et al. (2025) and the fact that using jvp, it can be trained more efficiently compared to $\mathcal{L}_2$.

C EXTENDED RELATED WORK

C.1 RELATION TO FREE-FORM FLOWS

Free-form Flows (FFF) enable arbitrary architectures to function as normalizing flows by jointly training an auxiliary network to approximate the inverse of the forward model (Draxler et al., 2024). The forward model maps data to latents, while the auxiliary network regularizes this mapping by learning a tractable estimator for the Jacobian term in the change of variables. As correctly highlighted by Draxler et al. (2024), although the loss encourages invertibility, this guarantee only holds when the reconstruction error is sufficiently small—a condition that can be difficult to meet in practice. In our setting, FALCON is trained with an entirely different loss function—albeit with a cycle-consistency term to promote invertibility—similar to that observed by Draxler et al. (2024). In addition, we explicitly validate invertibility for FALCON, in line with their observations through a dedicated experiment.

C.2 RELATION TO REGFLOWS

RegFlows (Rehman et al., 2025) enable a more efficient training process for standard discrete normalizing flows by avoiding the maximum-likelihood (MLE) objective, whose unstable training dynamics often make optimization a challenge; instead, RegFlows distill the knowledge of a predefined invertible coupling—either using a pre-trained continuous normalizing flow or a pre-computed optimal transport map—into a discrete normalizing flow via a regression objective. While this sidesteps issues associated with MLE and yields an efficient training pipeline, it also imposes a strict requirement on the existence of an invertible reference map, which must be provided in advance.

Further, RegFlows are constrained to inherently invertible architectures, which limits their expressivity and ultimately their scalability. As we show in Table 2 and Table 3, even when comparing against the strongest RegFlow variant (NSF), FALCON consistently outperforms it across all metrics, with pronounced gains on larger peptide systems. In contrast, FALCON is the first few-step Boltzmann Generator to employ a fully free-form architecture, surpassing state-of-the-art inherently invertible discrete flows (and even continuous normalizing flows) while avoiding the strict architectural constraints imposed by invertibility. Note, although FALCON is only guaranteed to be invertible either at convergence or via an additional regression objective, this relaxation enables substantially higher expressivity and delivers significant empirical performance gains. Lastly, we demonstrate the practical invertibility of FALCON through experiments performed in §F.2.

D Experimental Details

All training experiments are run on a heterogeneous cluster of NVIDIA H100 and L40S GPUs using distributed data parallelism (DDP). All models were trained with three random seeds, and reported values are averages across runs. Additional training and inference details are included in the following. For benchmarks with existing methods, in cases where dashes are present, data was unavailable, except for the case with SBG SMC (Tan et al., 2025a), where ESS is not a valid metric.

D.1 Training Details

Architecture. We adopt a Diffusion Transformer (DiT) backbone for FALCON, with the same model size used for all peptides. The details of the backbone configuration are in Table 6 below.

Table 6: Overview of FALCON configurations across datasets.

Dataset	Hidden Size	Blocks	Heads	Cond. Dim.	Parameters (M)
Alanine dipeptide	192	6	6	64	3.2
Tri-alanine	192	6	6	64	3.2
Alanine tetrapeptide	192	6	6	64	3.2
Hexa-alanine	192	6	6	64	3.2

Training Configuration All models were trained with an exponential moving average (EMA) on the weights using a decay rate of 0.999. Logit values were clipped at 0.002, with compositional energy regularization disabled. For evaluation, we generated 10^4 proposal samples, and used the same number for re-sampling and computing all metrics. Lastly, center-of-mass augmentation was applied with a standard deviation of $1/\sqrt{n}$, where n is the number of particles.

Hyperparameters Optimization was performed using AdamW with learning rate $\text{lr} = 5 \times 10^{-4}$, $\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$, and weight decay 10^{-4}. A cosine annealing learning rate schedule with a warm-up phase covering 5% of the training iterations was also used.

D.2 Inference Schedulers

We evaluated five inference schedules to assess their impact on generative performance. The linear baseline distributes steps uniformly across the trajectory, while the geometric, cosine, Chebyshev, and EDM schedules bias step allocation to emphasize regions where the data distribution is more sensitive. The mathematical definitions and parameter settings for each schedule are provided below.

Linear. The linear schedule spaces is distributed uniformly between 1 and 0, with $N \in \mathbb{N}$ denoting the total number of inference steps such that:

$$t_i = 1 - \frac{i}{N}, \quad i \in \{0, \ldots, N\}. \tag{16}$$

Geometric. The geometric schedule exponentially allocates more resolution to early steps. We let $\alpha \in \mathbb{R}$ and $\alpha > 1$, denote the geometric base. For all experiments conducted, we set $\alpha = 2$:

$$t_i = \frac{\alpha^{N-i} - 1}{\alpha^N - 1}, \quad i \in \{0, \ldots, N\}. \tag{17}$$

Cosine. The cosine schedule follows a squared-cosine law, concentrating more steps near $t = 0$. This has been used in diffusion models for improved stability. With $N \in \mathbb{N}$:

$$t_i = \cos^2\left(\frac{\pi}{2} \cdot \frac{i}{N}\right), \quad i \in \{0, \ldots, N\}, \quad t_N = 0. \tag{18}$$

Chebyshev. The Chebyshev schedule, derived from the nodes of Chebyshev polynomials of the first kind, minimizes polynomial interpolation error, and distributes steps more densely near boundaries:

$$t_i = \tfrac{1}{2}\Big(\cos\Big(\tfrac{\pi(i+0.5)}{N+1}\Big) + 1\Big), \quad i \in \{0, \ldots, N\}, \quad t_N = 0. \tag{19}$$

EDM. The EDM schedule (Karras et al., 2022) parameterizes the noise level σ using a power-law with exponent $\rho \in \mathbb{R}^+$. The interpolation is performed in ρ-space to allocate more steps where the generative process is most sensitive. For all experiments performed, we use the same parameters proposed by Karras et al. (2022), with $\rho = 7$, $\sigma_{\min} = 10^{-3}$, and $\sigma_{\max} = 1$, and $\sigma_{\min}, \sigma_{\max} \in \mathbb{R}^+$:

$$\sigma_i = \Big(\sigma_{\max}^{1/\rho} + \tfrac{i}{N}\left(\sigma_{\min}^{1/\rho} - \sigma_{\max}^{1/\rho}\right)\Big)^{\rho}, \quad i \in \{0, \ldots, N\}, \tag{20}$$

$$t_i = \frac{\sigma_i - \sigma_{\min}}{\sigma_{\max} - \sigma_{\min}}, \quad t_N = 0. \tag{21}$$

This formulation ensures a smooth transition between $\sigma_{\max}$ (high noise) and $\sigma_{\min}$ (low noise, near-deterministic refinement).

D.3 Dataset Details

For the datasets used, we follow the same training, validation, and test split used by (Tan et al., 2025a), where a single MCMC chain is decomposed into 10^5, 2×10^4, and 10^4 samples for training, validation, and test. The training and validation data are each taken as contiguous regions of the chain to simulate the realistic scenario where you have generated an MCMC trajectory and would like to use a Boltzmann Generator to continue generating samples. The data is split so that (after warmup) the first 10^5 samples are for train, the next 2×10^4 are for validation, and the test samples are uniformly strided subsamples from the remaining MCMC chain. Earlier parts of the trajectory undersample certain modes enabling a biased training set that we attempt to debias through access to the energy function and SNIS. MD simulations were all run for 1 µs, with a timestep of 1 fs, at temperatures of 300K, 310K, and 300K for alanine dipeptide, tri-alanine, and alanine tetrapeptide, respectively, in line with those conducted by Klein & Noe (2024). The force fields used for all three molecules, in the same order, were the `Amber ff99SBildn`, `Amber 14`, and `Amber ff99SBildn`, in line with prior work by Tan et al. (2025a).

Alanine Dipeptide. The Ramachandran plots for the training and test split are provided in Fig. 8. Following Klein et al. (2023a), we up sample the lowest sampled mode (middle right) to make the problem easier for BGs. This also has the added benefit of testing what happens to FALCON when the training dataset is biased relative to the true distribution.

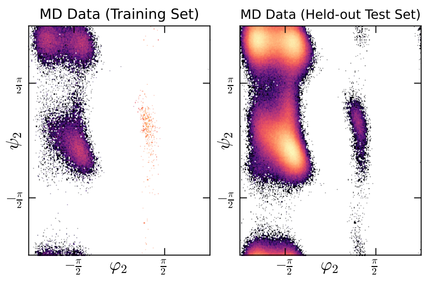

Figure 8: **Left:** Training data for alanine dipeptide; **Right:** Test data for alanine dipeptide.

Tri-alanine. Similarly to alanine dipeptide, the Ramachandran plots for the training and test split are in Fig. 9. As there are two pairs of torsion angles that parameterize the system, there are two sets of Ramachandran plots included for each training and test. In tri-alanine, we can see that the training set actually misses a mode entirely ($\psi_1 \approx \pi/3$), and undersamples this mode for ψ_2 relative to the test set. This is a great test of finding a new mode (in ψ_1) and correctly weighting a mode (in ψ_2).

Alanine Tetrapeptide. For the tetrapeptide, there are three sets of Ramachandran plots each for training and test given the three pairs of torsions angles that parameterize the molecule, in Fig. 10.

Hexa-alanine. For hexa-alanine, we also include Ramachandran plots in Fig. 11. The first row shows training data, while the second row shows a held-out test set used to evaluate performance.

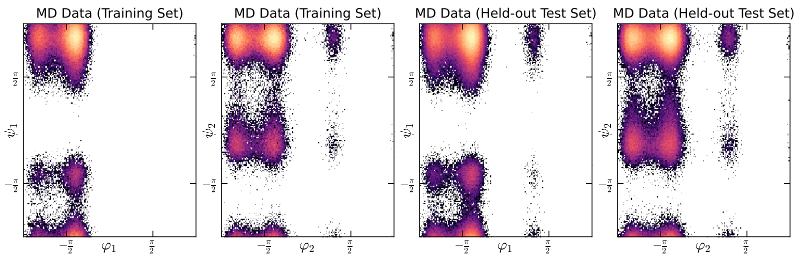

Figure 9: **Left and left center:** Training data for tri-alanine; **Right center and right:** Test data for tri-alanine.

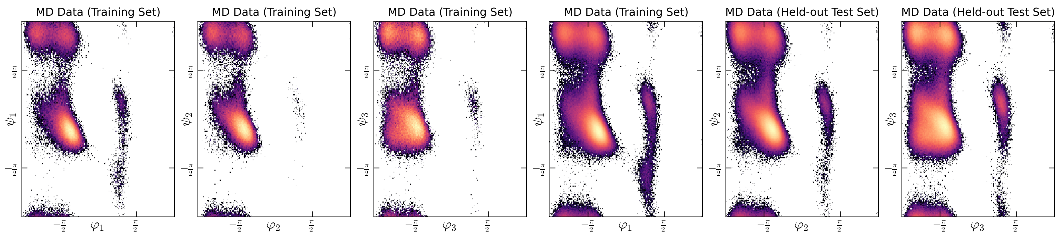

Figure 10: **First three:** Training data for alanine tetrapeptide; **Last three:** Test data for alanine tetrapeptide.

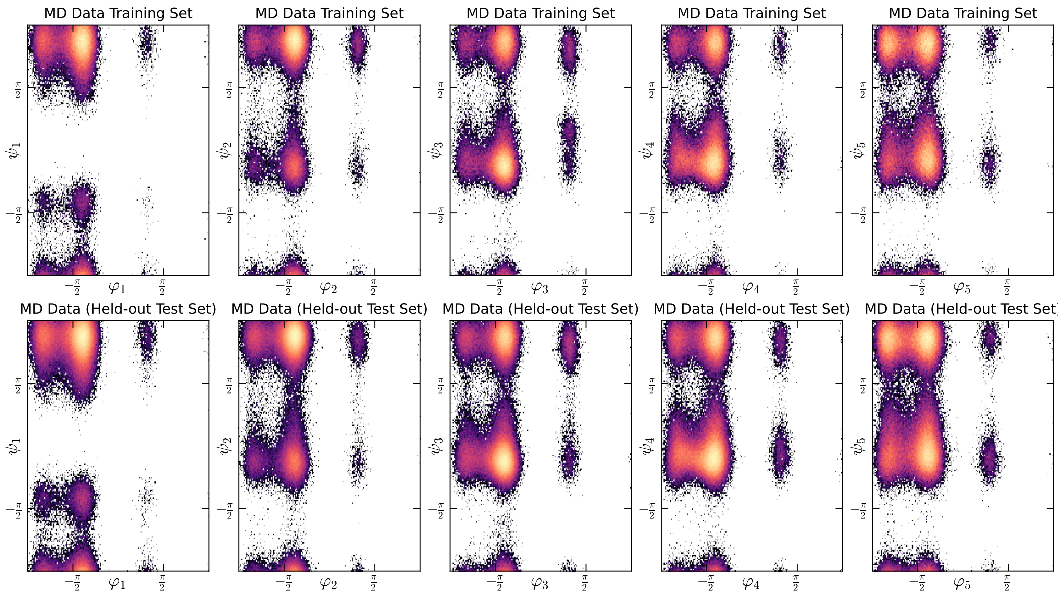

Figure 11: **First five:** Training data for hexa-alanine; **Last three:** Test data for hexa-alanine.

E METRICS

Effective Sample Size (ESS). To quantify sampling efficiency, we compute the effective sample size (ESS) following Kish's definition (Kish, 1957). Given $N \in \mathbb{N}$ generated particles with unnormalized importance weights $\{w_i\}_{i=1}^N \subset \mathbb{R}^+$, the ESS is normalized by N as:

$$\text{ESS}(\{w_i\}_{i=1}^N) \triangleq \frac{1}{N} \frac{1}{\sum_{i=1}^N w_i^2} \left(\sum_{i=1}^N w_i \right)^2 . \tag{22}$$

The ESS reflects how many independent, equally-weighted samples would provide equivalent statistical power to the weighted sample. Higher ESS is desirable for more performant models.

2-Wasserstein Energy Distance ($\mathcal{E}$-$\mathcal{W}_2$). To compare energy distributions, we compute the 2-Wasserstein distance between generated and reference samples from our MD dataset. For distributions

$p, q \in \mathcal{P}(\mathbb{R})$ over energy values, and $\Pi(p, q)$ the set of admissible couplings, we define:

$$\mathcal{E}\text{-}\mathcal{W}_2(p,q)^2 \triangleq \min_{\pi \in \Pi(p,q)} \int_{\mathbb{R}\times\mathbb{R}} |x-y|^2 \, d\pi(x,y). \tag{23}$$

The $\mathcal{E}$-$\mathcal{W}_2$ measures how closely the generated energy histogram aligns with that of the reference, with high sensitivity to structural accuracy due to bond-length–dependent energies. Since small perturbations in local structure induce large fluctuations in the energy distribution, this metric captures this variance, with lower values of $\mathcal{E}$-$\mathcal{W}_2$ being more favourable for generative models.

Torus 2-Wasserstein Distance ($\mathbb{T}$-$\mathcal{W}_2$). To assess structural similarity in torsional space, we compute a 2-Wasserstein distance over dihedral angles. For a molecule with $L \in \mathbb{N}$ residues, define the dihedral vector as:

$$\text{Dihedrals}(x) = (\phi_1, \psi_1, \ldots, \phi_{L-1}, \psi_{L-1}) \in [0, 2\pi)^{2(L-1)}. \tag{24}$$

The cost on the torus accounts for periodicity of angles:

$$c_{\mathcal{T}}(x,y)^2 = \sum_{i=1}^{2(L-1)} \left[(\text{Dihedrals}(x)_i - \text{Dihedrals}(y)_i + \pi) \bmod 2\pi - \pi \right]^2. \tag{25}$$

The torus 2-Wasserstein distance between two distributions $p, q \in \mathcal{P}([0, 2\pi)^{2(L-1)})$ is then:

$$\mathbb{T}\text{-}\mathcal{W}_2(p,q)^2 \triangleq \min_{\pi \in \Pi(p,q)} \int c_{\mathcal{T}}(x,y)^2 \, d\pi(x,y). \tag{26}$$

This captures macroscopic conformational differences while respecting angular periodicity. The $\mathbb{T}$-$\mathcal{W}_2$ provides a more global assessment of performance, for instance revealing missed conformational modes that would not be captured by the $\mathcal{E}$-$\mathcal{W}_2$.

F Additional Experiments

F.1 Details for Figure Generation

In Fig. 2, we compare performance on the $\mathbb{T}$-$\mathcal{W}_2$ metric for alanine dipeptide between FALCON and our continuous-time DiT CNF. For FALCON, accuracy is controlled by the number of inference steps (1–8). For the DiT CNF, we consider three settings: (**1**) adaptive step Dormand–Prince 4(5) with exact Jacobian trace evaluation, varying atol/rtol from 10^{-1} to 10^{-5}; (**2**) the same tolerances but instead using the Hutchinson trace estimator, trading performance for faster likelihoods with higher variance; and (**3**) fixed-step Euler integration with step sizes from 4 to 256 with exact Jacobian traces, where the upper bound is chosen to roughly match the number of function evaluations needed for Dopri5. The models and training configurations used are presented in §D.1.

In Fig. 3, we demonstrate the energy distributions for unweighted and re-weighted samples for our most performant FALCON Flows. For alanine dipeptide, tri-alanine, alanine tetrapeptide, and hexa-alanine, 4, 8, 8, and 16 steps were used for figure generation. Energy distributions reveal microscopic detail, as marginal changes in local atomic position can have significant impacts on total energy. The best models for each molecular system were used, with pertinent details on model size, training configurations, and hyperparameters detailed in §D.1. Similarly, in Fig. 5, we show the proposal and re-weighted sample estimates for alanine dipeptide, and demonstrate how an increasing number of steps, improves the energy distribution.

In Fig. 4, we compare the performance between SBG's discrete NFs and FALCON, illustrating that despite more samples increasing performance across metrics, the approach is still unable to reach FALCON's performance. For SBG, we specifically take their best model weights for the TarFlow architecture from: `https://github.com/transferable-samplers/transferable-samplers`, and draw $N_s \in \{10^4, 2\times10^4, 5\times10^4, 10^5, 2\times10^5, 5\times10^5, 10^6, 2\times10^6, 5\times10^6\}$ samples three different times (the error bars are representative of the three draws) and evaluate $\mathcal{E}$-$\mathcal{W}_2$ for each set.

In Fig. 6, we investigate how the strength of regularization impacts performance on ESS and $\mathcal{E}$-$\mathcal{W}_2$. Specifically, we demonstrate that small amounts of regularization enable generative performance but impede invertibility, while too much regularization detrimentally impacts sample quality. This trade-off leads to an optimum on both metrics. We investigate $\lambda_r \in \{10^0, 10^1, 10^2, 10^3, 10^4, 10^5\}$, and conclude upon $\lambda_r = 10^1$. To ascertain the optimal λ_r, we ran these experiments on alanine dipeptide, using the same model details and configurations highlighted in §D.1.

Fig. 7 demonstrates the sensitivity the inference schedule plays on generative performance. For this figure, we took our best models—trained with the details presented in §D.1—and ran inference using all three trained seeds to generate uncertainties for each inference schedule. We used our 8-step system, as inference scheduler choice is less important the fewer steps are used.

F.2 Proof of Invertibility

In our loss, we use a cycle-consistency term that regularizes training to promote numerical invertibility. We see in Fig. 6, the introduction of this modified loss aids generative performance; however, the forward-backward reconstruction yields errors on the order of 10^{-2}, indicating an approximately, but not entirely invertible model. To prove that the flow is invertible (but the inverse is challenging to discover during training), we train an auxiliary FALCON Flow in the reverse direction. We use the same network to parameterize the reverse flow as described in §D.1, with the same training configuration and hyperparameter set. Next, we freeze the forward model (which goes from latents $\rightarrow$ data), generate synthetic data (2×10^5 prior-target sample pairs) and train the auxiliary model on these reflow targets to learn the mapping from data back to latents. In Fig. 12, we illustrate the loss curve of the trained auxiliary FALCON. To evaluate invertibility, we draw i.i.d. samples from our prior distribution, pass them through the frozen forward model, and test the auxiliary reverse model on these unseen latents. We evaluate an ℓ_2 error on the recovered latents to find reconstruction within 10^{-4} after 1000 epochs matching the reconstruction accuracy of discrete NFs that are invertible by design. These results support the claim that the learned flow is indeed invertible.

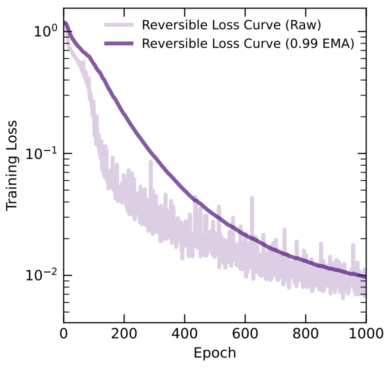

Figure 12: Auxiliary model loss from reflow target training on forward flow.

F.3 Performance Against Number of Function Evaluations

Fig. 13 highlights the efficiency of FALCON in terms of number of function evaluations. We see that FALCON achieves the same torus 2-Wasserstein performance as our DiT CNF with Dopri5, while requiring over two orders of magnitude fewer function evaluations. Whereas Fig. 2 quantified efficiency in terms of inference time, here we measure the number of function evaluations against both fixed-step solvers (Euler with 4–256 steps) and adaptive solvers (Dopri5 run until reaching target tolerances atol $=$ rtol $\in \{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$ for Hutchinson and $\in \{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$ for exact).

Although Hutchinson's trace estimator yields a speedup relative to exact Jacobian computation for CNFs, as seen in Fig. 2, the number of function evaluations needed is higher than the exact Jacobian computation. In either setting, however, FALCON still remains substantially faster. Even with a 4-step solver, FALCON matches the accuracy of DiT CNFs using Hutchinson at atol $=$ rtol $= 10^{-5}$, with an 8-step solver nearly matching the performance of a DiT CNF with Dopri5 set to an atol $=$ rtol $= 10^{-6}$.

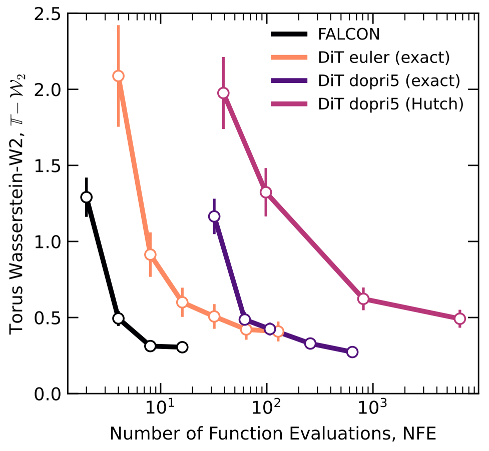

Figure 13: Performance of FALCON vs. our DiT CNFs as a function of NFEs.

F.4 Ramachandran Plots

Alanine Dipeptide. We demonstrate FALCON's capacity to learning global features through the Ramachandran plots in Fig. 14 for alanine dipeptide. We include both the held-out test set and the learned model's map, showcasing its ability to debias the data and capture the true MD distribution. Specifically, the undersampled ϕ mode in the training data is correctly upweighted in the learned model's predictions, indicating accurate likelihood estimates from the learned flow.

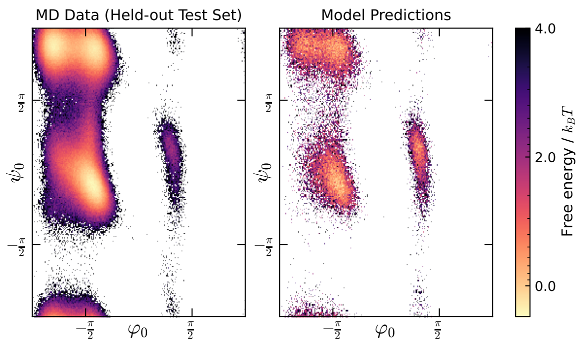

Figure 14: **Left:** Test data for alanine dipeptide; **Right:** FALCON's angular predictions for alanine dipeptide.

Tri-alanine. Similarly, we show the Ramachandran plots for tri-alanine in Fig. 15 exhibiting similar behaviour. Most conformations are correctly captured, with some modes being underweighted.

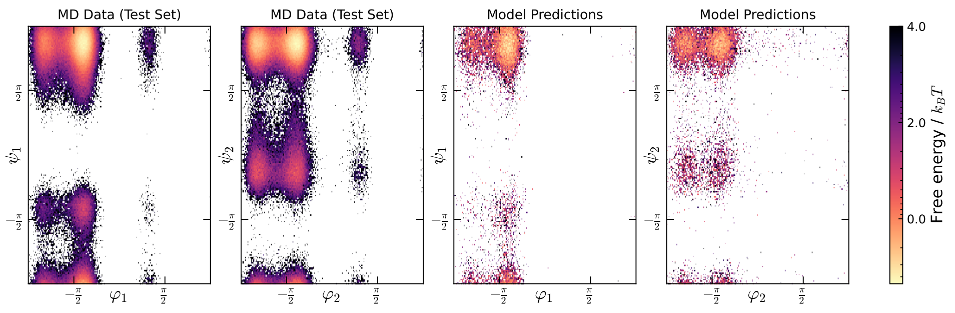

Figure 15: **Left and left center:** Test data for tri-alanine; **Right and right center:** FALCON's angular predictions for tri-alanine.

Alanine Tetrapeptide. Next, we show the Ramachandran plots for alanine tetrapeptide in Fig. 16.

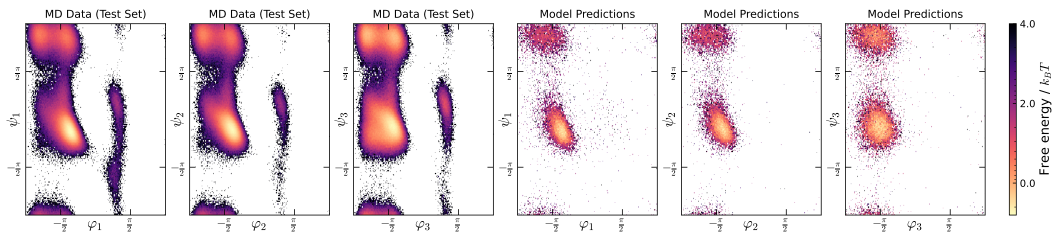

Figure 16: **First three:** Test data for alanine tetrapeptide; **Last three:** FALCON's angular predictions for alanine tetrapeptide.

Hexa-alanine. Finally, we show the Ramachandran plots for hexa-alanine in Fig. 17.

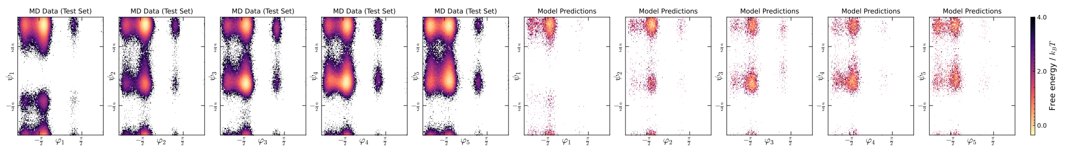

Figure 17: **First five:** Test data for hexa-alanine; **Last five:** FALCON's angular predictions for hexa-alanine.