Amortized Sampling with Transferable Normalizing Flows

Charlie B. Tan [*1] **Majdi Hassan** [*2 3] **Leon Klein** [4] **Saifuddin Syed** [1] **Dominique Beaini** [2 3 5]
Michael M. Bronstein [1 6] **Alexander Tong** [†2 3 7] **Kirill Neklyudov** [†2 3]

Abstract

Efficient equilibrium sampling of molecular conformations remains a core challenge in computational chemistry and statistical inference. Classical approaches such as molecular dynamics or Markov chain Monte Carlo inherently lack *amortization*; the computational cost of sampling must be paid in-full for each system of interest. The widespread success of generative models has inspired interest into overcoming this limitation through learning sampling algorithms. Despite performing on par with conventional methods when trained on a single system, learned samplers have so far demonstrated limited ability to transfer across systems. We prove that deep learning enables the design of scalable and transferable samplers by introducing ENSEMBLE, a 280 million parameter all-atom *transferable* normalizing flow trained on a corpus of peptide molecular dynamics trajectories up to 8 residues in length. ENSEMBLE draws zero-shot uncorrelated proposal samples for arbitrary peptide systems, achieving the previously intractable transferability across sequence length, whilst retaining the efficient likelihood evaluation of normalizing flows. Through extensive empirical evaluation we demonstrate the efficacy of ENSEMBLE as a proposal for a variety of sampling algorithms, finding a simple importance sampling-based finetuning procedure to achieve superior performance to established methods such as sequential Monte Carlo. ENSEMBLE opens the door for further research into sampling methods and finetuning objectives.

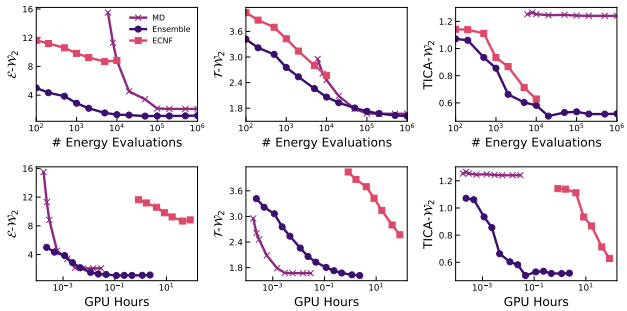

Figure 1: **ENSEMBLE exceeds the quantitative performance of baseline on *unseen* peptide systems.** Wasserstein-2 distances on energy, dihedral angle, and TICA projection ($\mathcal{T}$-$\mathcal{W}_2$), for molecular dynamics, equivariant continuous normalizing flow (with SNIS), and ENSEMBLE (with SNIS), at a range of energy evaluation (above) and GPU walltime budgets (below). Mean of 30 unseen tetrapeptide systems, ECNF evaluated only up to 10^4 energy evaluations due to prohibitive GPU walltime. ENSEMBLE is the most performant method at any energy budget on all metrics. Whilst MD exceeds the performance of ENSEMBLE with GPU hours on E-$\mathcal{W}_2$ and $\mathcal{T}$-$\mathcal{W}_2$, it is significantly inferior on TICA-$\mathcal{W}_2$, indicating a failure to sample all metastable states.

1. Introduction

Accurately sampling molecular configurations from the Boltzmann distribution is a fundamental problem in statistical physics with profound implications for understanding biological and chemical systems. Key applications include protein folding Noé et al. (2009); Lindorff-Larsen et al. (2011), protein–ligand binding (Buch et al., 2011), and crystal structure prediction (Köhler et al., 2023); processes that underpin advances in drug discovery and material science.

Conventional approaches such as Markov Chain Monte Carlo (MCMC) (Liu, 2001) and, in particular, Molecular Dynamics (MD) (Leimkuhler & Matthews, 2015) seek to tackle this problem by proposing a general solution, which, however, has practical limitations due to its Markov nature. To accurately integrate the corresponding Hamiltonian dynamics, MD has to be simulated with a fine time-discretization (on the order of femtoseconds), which produces highly correlated samples and prevents efficient exploration of the modes of the Boltzmann density. Although running multiple chains from different initializations is possible, every chain has to be simulated for a long time to ensure proper mixing, which cannot be efficiently parallelized. Finally, the entire simulation has to be re-started from scratch for a new

[*]Equal contribution [†]Equal advising [1]University of Oxford [2]Mila – Quebec AI Institute [3]Université de Montréal [4]Freie Universität Berlin [5]Valence Labs [6]AITHYRA [7]Current: Duke University. Correspondence to: Charlie B. Tan <charlie.tan@cs.ox.ac.uk>, Majdi Hassan <majdi.hassan@mila.quebec>.

Proceedings of the Workshop on Generative AI for Biology at the 42nd International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

system, which bottlenecks the speed of ab initio studies.

Deep learning-based samplers, although considered in different settings, abandon the Markov Chain approach to generating samples and shift the computational burden to a one-time training phase, enabling fast and inexpensive inference compared to MCMC. In the most challenging scenario, these methods consider having access only to the unnormalized density function (analogous to MC methods) (Vargas et al., 2023; Akhound-Sadegh et al., 2024). Boltzmann Generators (BGs) (Noé et al., 2019) consider a more practical scenario when, in addition to the unnormalized density, a dataset of MD trajectories is available, which does not necessarily match the target density. To eliminate the error introduced by the imperfections of the model and training data, BGs rely on training likelihood-based models and perform self-normalized importance sampling (SNIS) (Liu, 2001) at inference time. The availability of the training data and inference-time IS have enabled BGs to generalize across small peptides of the same sequence length (Klein & Noe, 2024), but they still fall short of generalizing to larger and more diverse systems of interest.

In this work, we introduce ENSEMBLE, a large-scale normalizing flow which demonstrates unprecedented abilities to transfer to previously unseen systems of different amino acids, sizes, and temperatures, outperforming MD for the same computational budget (see Fig. 1). Our approach is strikingly simple and scalable, which elucidates the potential of the deep learning-based samplers for sampling applications. In particular, we achieve this through the following series of contributions:

- We introduce ManyPeptidesMD; a novel dataset of molecular dynamics trajectories for peptide systems between 2 and 8 residues. The training dataset consists of $\approx 16,000$ peptide sequences simulated for 50 ns each, with $\approx 8,000$ sequences for octopeptides alone. This dataset is a superset of that of Klein et al. (2023a), increasing the sequence count 10-fold.
- Building on the recently proposed TarFlow (Zhai et al., 2024), we propose architectural modifications, which allow for better modeling of peptide systems, system-transferable conditioning, and generation of peptide sequences of varying length.
- We study the use of ENSEMBLE as a proposal distribution for different Monte Carlo algorithms, finding the learned proposal to be sufficiently powerful for accurate sampling with standard SNIS, which does not require tuning of parameters. Furthermore, resampled generations can be used for efficient finetuning of ENSEMBLE on previously unseen systems.
- Finally, we empirically demonstrate that ENSEMBLE achieves state-of-the-art performance when sampling from the equilibrium distribution on previously unseen peptide systems of length up to 8 residues surpassing the continuous normalizing flow-based transferable Boltzmann generator (Klein & Noe, 2024) whilst generating proposals $4 \cdot 10^3$ times faster.

2. Background

2.1. Normalizing flows

The fundamental challenge of the probabilistic modeling is the design of the density model that, at the same time, allows for efficient generation of samples from this density. Normalizing flows (Rezende & Mohamed, 2015) approach this challenge by defining a diffeomorphism — differentiable invertible function with a differentiable inverse. Namely, given a simple prior density $q_z(z)$ and a parameterized flow (diffeomorphism) $f_\theta^{-1}(z)$, one can define the push-forward distribution as the map of samples from a simple prior distribution $z \sim q_z(z)$ via the learnable flow $x = f_\theta^{-1}(z) \sim q_\theta(x)$ with the parameters θ. The density of the push-forward distribution can then be found via the change-of-variables formula

$$q_\theta(x) = \int dz\, q_z(z)\delta(x - f_\theta^{-1}(z)) = q_z(f_\theta(x))\left|\frac{\partial f_\theta(x)}{\partial x}\right|, \tag{1}$$

where $|\partial f_\theta(x)/\partial x|$ is the determinant Jacobian of the map f_θ. However, for practical applications, one has to be able to efficiently evaluate $f_\theta(z)^{-1}$ in order to generate samples and $|\partial f_\theta(x)/\partial x|$ for evaluating the density via the change-of-variables formula.

Auto-regressive normalizing flows (Kingma et al., 2016; Papamakarios et al., 2017; Zhai et al., 2024) define a rich family of invertible maps with tractable Jacobian as a sequence of transformations $f_\theta^{-1} = f_1^{-1} \circ \ldots \circ f_\tau^{-1}$ ($\circ$ denotes the composition), where each transformation $z_t = f_t^{-1}(z_{t+1})$ is defined as an auto-regression over the dimensions. That is, the i-th coordinate of the output is

$$z_t[i] = \begin{cases} z_{t+1}[i]\,, & i = 0\,, \\ z_{t+1}[i] \cdot \exp(\alpha_t(z_t[:i])) + \mu_t(z_t[:i])\,, & i \in [1, d]\,, \end{cases} \tag{2}$$

where we adopt slicing notation denoting the i-th dimension as $z[i]$ and all the dimensions up to i-th (exclusive) as $z[:i]$. Notably, the auto-regressive structure allows for efficient evaluation of the Jacobian determinant due to its lower-triangular structure and the inverse function $z_{t+1} = f_t(z_t)$,

i.e.

$$\log\left|\frac{\partial f_t(z_{t+1})}{\partial z_{t+1}}\right| = -\sum_{i=1}^{d}\alpha_t(z_t[:i]), \tag{3}$$

$$z_{t+1}[i] = \begin{cases} z_t[i], & i = 0 \\ \dfrac{z_t[i]-\mu_t(z_t[:i])}{\exp(\alpha_t(z_t[:i]))}, & i \in [1,d] \end{cases}$$

However, clearly, such transformation always leave the leading dimension $z_t[0]$ untouched, that is why they have to be interleaved with permutations over the dimensions $f_\theta = \pi_\tau \circ f_\tau \circ \ldots \circ \pi_1 \circ f_1$, where π_t is the permutation across dimensions. In practice, Zhai et al. (2024) use simple inversions for all π_t across the entire model.

Normalizing flows are versatile probabilistic models, which allow for different training strategies: for learning target unnormalized densities, variational inference (Rezende & Mohamed, 2015), for generative modeling, the maximum likelihood principle (Kingma & Dhariwal, 2018a). Analogously, one can use the estimated densities at the inference time, which we discuss in the next section.

2.2. Boltzmann generators

Despite allowing for different kinds of supervision at the training time (target unnormalized densities or empirical target distributions), the quality of samples generated at the inference-time does not allow for scientific applications requiring high precision, e.g. free energy estimation. *Boltzmann generators* address specifically this challenge by doing self-normalized importance sampling (SNIS) at the inference time. Namely, to evaluate the expectation of a statistic $\varphi(x)$ w.r.t. the target Boltzmann density $p(x)$ one can use the following consistent Monte Carlo estimator

$$\mathbb{E}_{p(x)}\varphi(x) \approx \sum_{i=1}^{n}\frac{w_i}{\sum_{j=1}^{n} w_j}\varphi(x_i)\,,\quad w_i = \frac{p(x_i)}{q(x_i)}\,,\quad x_i \sim q(x)\,, \tag{4}$$

where $q_\theta(x)$ is the density of the learned normalizing flow. The SNIS estimator converges, for $n \to \infty$, to the true value $\mathbb{E}_{p(x)}\varphi(x)$. Furthermore, Tan et al. (2025) demonstrated that one can reduce the variance of the SNIS estimator even further by leveraging the scalability of the recently proposed TarFlow (Zhai et al., 2024) and combining it with the continuous-time Sequential Monte Carlo (Jarzynski, 1997; Albergo & Vanden-Eijnden, 2024).

Transferable Boltzmann Generators (TBG) (Klein & Noe, 2024) made a first attempt of learning a sampler that generalizes across different target densities corresponding to the molecular systems of different type. In details, TBG parameterizes the proposal distribution as a continuous normalizing flow (CNF) (Chen et al., 2018), where the vector field is defined by the equivariant graph neural network (Klein et al., 2023b). The crucial part of the algorithm is the peptide-dependent embedding of N different atoms

$$z \in \mathbb{R}^{N\times(3+d)}\,,\quad z[i,:] = [y_i, A_i, R_i, P_i]\,, \tag{5}$$

where $z[i,:]$ corresponds to i-th atom in the system with the spacial coordinates $y_i \in \mathbb{R}^3$ and the conditional information $[A_i, R_i, P_i] \in \mathbb{R}^d$: atom type A_i, amino acid residue type R_i, and the position of the amino acid in the sequence P_i.

Despite successful generalization to previously unseen dipeptides when trained on 200 dipeptides (Klein et al., 2023a), TBG architecture introduces significant bottlenecks for inference and fine-tuning. Indeed, the learned CNF requires accurate integration of the vector field and computationally expensive evaluation of its divergence for evaluating the learned density model. For instance, to perform importance sampling proportionally to the target Boltzmann density, TBG requires around 4 GPU-days to produce 30k samples for a single dipeptide system. Furthermore, the expensive evaluation of density makes it infeasible to train or finetune TBG via the reverse KL-divergence or create a replay buffer of a substantial size.

3. Scalable transferable normalizing flows as Boltzmann generators

3.1. Architecture of ENSEMBLE

ENSEMBLE builds on the recent TarFlow architecture (Zhai et al., 2024), which parameterizes a sequence of autoregressive affine transformations via blocks of transformer layers. The expressivity and favorable scalability of the transformer layers enables TarFlow to effectively model high dimensional data, whilst the affine autoregressive flow parameterization ensure fast and accurate energy evaluation. With minimal modifications TarFlow is capable of successfully modeling high-dimensional molecular data (Tan et al., 2025). Here we describe our design choices that make transferability possible.

Transferability across system dimensions We extend TarFlow to support concurrent training on sequences of arbitrary length. Whilst transformers natively support sequences of arbitrary length, special consideration is required within a normalizing flow such as TarFlow that is defined for fixed input and output dimensions. We therefore define appropriate masking to the affine sequence updates and log-determinant aggregation to prevent padding tokens influencing either computation, under arbitrary sequence permutations. We additionally replace the fixed-length learnable position embedding with the more extrapolation-friendly sinusoidal embedding. This design enables ENSEMBLE to efficient train across a distribution of systems s by maximizing the normalized log-likelihood

$$\max_\theta \mathbb{E}_s \frac{1}{d(s)}\mathbb{E}_{x\sim p(x\,|\,s)}\log q_\theta(x) \tag{6}$$

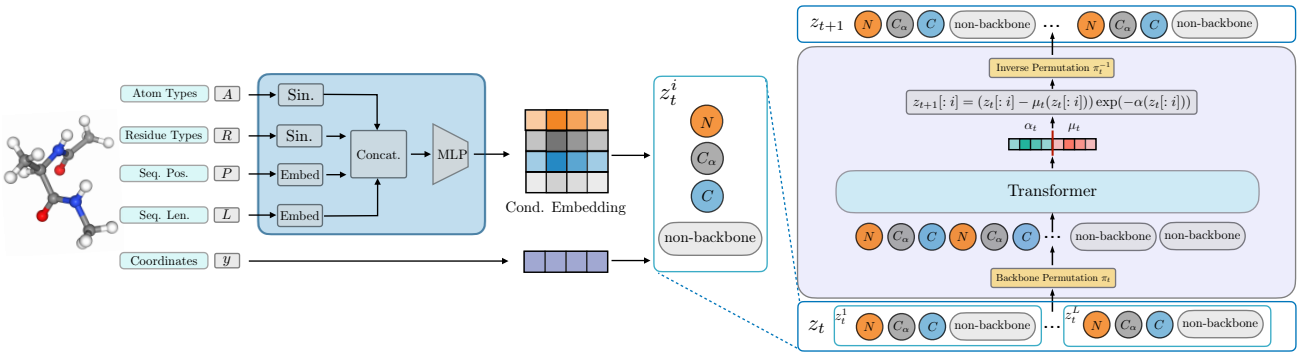

Figure 2: **All-atom block-wise autoregressive normalizing flow based on the TarFlow (Zhai et al., 2024).** Peptide molecules are encoded as the atom types A, residue types R, residue sequence position P, and residue length L. Atom positions in 3D Cartesian coordinates define the system state. An embedding of the peptide molecule is applied as conditioning to the coordinates such that ENSEMBLE achieves transferability between systems. Within each block the sequence z_t is permuted and passed to a transformer, defining an autoregressive affine update. In the backbone permutation the backbone $[N_i, C_{\alpha,i}, C_i]_{i=1}^{L}$ of all residues is updated before any sidechains, providing additional diversity to the causal attention for global structure modeling.

where $d(s)$ is the size of the system s. This extended architecture allows for parallel processing of data dimensions, enabling transferability and scalability across lengths.

Adaptive system conditioning The standard TarFlow employs simple additive conditioning for class-conditional image generation. Whilst we find this to be sufficient to define a system-transferable normalizing flow, we follow many large-scale atomistic transformer architectures in applying conditioning through adaptive layer normalization, adaptive scaling, and SwiGLU-based transition blocks Abramson et al. (2024); Geffner et al. (2024). The system conditioning features are constructed from atom types A, residue types R, sequence positions P, and sequence lengths L. Atom and residue types are both embedded using lookup-table based embedding layers, whilst sinusoidal embeddings are a more suitable selection for the naturally ordered sequence position and sequence length. Residue-level information is repeated appropriately to give atom-level embeddings.

Chemistry-aware sequence permutations In the image setting, TarFlow employ only an identity and flip permutation to the sequence of image tokens (Zhai et al., 2024). When applying TarFlow to peptide systems Tan et al. (2025) similarly employ only an identity and flip permutation on the atom ordering defined by the PDB formatting, in which the sequence is defined per-residue starting with backbone atoms followed by sidechain atoms. Whilst a simple identity and flip may be appropriate for the regular grid of image data, we argue this to be suboptimal for the diversity of pair-wise geometric interactions present in molecular systems. This motivates our introduction of chemistry-aware peptide permutation sequences, defined to promote effective molecular modeling. We define the *backbone permutation*, such that the backbone atoms $[N_i, C_{\alpha,i}, C_i]_{i=1}^{L}$ are updated for the full peptide sequence, before the model returns to update the sidechains. By updating the coordinates of the backbone atoms first, the model refines the global structure of the peptide, defining the dihedral angles as a contiguous sequence. Crucially, when the sidechain positions are subsequently updated via causal attention, they are able to attend to the backbone structure, hence enabling local updates to be influenced by global macrostructure. We further employ a *backbone-flip* permutation to provide additional permutation diversity to the autoregressive modeling.

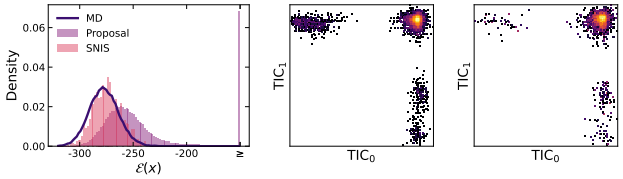

Figure 3: **ENSEMBLE accurately samples from the Boltzmann distributions of unseen octopepitde system.**. Empirical results for sampling from DGVAHALS peptide system, not present in training data. Energy histogram (left) for MD data, ENSEMBLE proposal and ENSEMBLE reweighted using SNIS, demonstrate fine-grained detail accuracy. TICA plots for MD (center) and SNIS-reweighted ENSEMBLE (right) illustrate mode coverage.

3.2. Inference and fine-tuning of ENSEMBLE

Applicability of Boltzmann Generators significantly depends on the inference-time throughput and their transferability to previously unseen systems. Here, we describe how one can perform inference-time importance sampling, fine-tuning using the generated samples, and annealing of the proposal to different target temperatures.

Importance sampling. At the inference time, one can use ENSEMBLE to estimate the expectation of statistics $\varphi(x)$ w.r.t. the target Boltzmann density $p(x)$ via a Self-Normalized Importance Sampling (SNIS) estimator.

Namely, we consider standard SNIS, discrete-time annealed importance sampling (AIS) (Neal, 2001), and continuous-time AIS (Jarzynski, 1997; Albergo & Vanden-Eijnden, 2024). All these estimators are of the form

$$\mathbb{E}_{p(x)}\varphi(x) \approx \sum_{i=1}^{n} \frac{w_i}{\sum_{j=1}^{n} w_j}\varphi(x_i)\,, \quad w_i = \frac{p(x_i)}{q(x_i)}\,, \quad x_i \sim q(x)\,, \tag{7}$$

where the only difference between them is the proposal density $q(x)$. Note that these estimators can be interpreted as the expectation over the empirical distribution, i.e.

$$\sum_{i=1}^{n} \frac{w_i}{\sum_{j=1}^{n} w_j}\varphi(x_i) = \mathbb{E}_{\tilde{p}(x)}\varphi(x)\,, \tag{8}$$

$$\tilde{p}(x) = \sum_{i=1}^{n} \frac{w_i}{\sum_{j=1}^{n} w_j}\delta(x - x_i)\,.$$

In practice, we compare $p(x)$ with $\tilde{p}(x)$ instead of measuring statistics $\varphi(x)$. For completeness, we describe all the considered estimators in Appendix E.

Self-refinement. For a previously unseen system s we demonstrate the ability to fine-tune ENSEMBLE using the Self-Refinement strategy. Namely, we iteratively generate the empirical distribution $\tilde{p}(x \mid s)$ by resampling the samples from pre-trained model $q_\theta(x \mid s)$ proportionally to $p(x \mid s)$ and using these samples for fine-tuning ENSEMBLE. Note that this is different from fine-tuning because the true samples from the target are not available. In particular, we update the parameters by maximizing the likelihood on the resampled proposal, i.e.

$$\max_{\theta} \mathbb{E}_{\tilde{p}(x|s)}[\log q_\theta(x \mid s)], \tag{9}$$

$$\tilde{p}(x \mid s) = \sum_{i=1}^{n} \frac{w_i}{\sum_{j=1}^{n} w_j}\,\delta(x - x_i),$$

$$w_i = \texttt{detach}\left(\frac{p(x_i \mid s)}{q_\theta(x_i \mid s)}\right).$$

Temperature transfer. ENSEMBLE serves as an efficient proposal for Boltzmann densities with different temperatures; hence, allowing for transferability across temperatures. In order to do this, we aim at changing the temperature $T = 1/\beta$ of the learned density model while generating samples, i.e.

$$\beta \log q_\theta(f_\theta^{-1}(z)) \; = \beta \log q_z(z) - \beta \log \left|\frac{\partial f_\theta^{-1}(z)}{\partial z}\right|. \tag{10}$$

Note that, for the measure-preserving flows $\log|\partial f_\theta^{-1}(z)/\partial z| = 0$, one simply has to change the temperature of the prior distribution (i.e. sample $z \sim q_z(z)^\beta$ instead of $z \sim q_z(z)$) to change the temperature of the density model, which is a standard technique in the normalizing flow literature (Kingma & Dhariwal, 2018a; Dibak et al., 2022). Although this assumption does not usually hold in practice, we found that scaling the temperature of the prior results in an efficient proposal for the Boltzmann density with the corresponding temperature.

4. Experiments

To establish the performance of ENSEMBLE, we first introduce a new molecular dynamics dataset for peptides, then test the ability of ENSEMBLE to sample from unseen systems of up to 8 residues.

4.1. Molecular dynamics trajectory dataset

We introduce ManyPeptidesMD; a novel dataset of peptide MD trajectories for sequences ranging from 2 to 8 residues in length. This data is a superset of the trajectories at length 2 and 4 introduced by Klein et al. (2023a). Following Klein et al. (2023a) all simulation is performed using `OpenMM` (Eastman et al., 2017) with the `amber-14` forcefield. For training, a total of 15,420 additional uniform sampled sequences are simulated for $50\,\mathrm{ns}$, greatly increasing over the prior dataset of 1647 sequences. For evaluation, 30 sequences of length 8 are randomly sampled such that all amino acids are represented equally, and simulated for $1\,\mu\mathrm{s}$. The evaluation data at lengths 2 and 4 is unchanged from the trajectories of Klein et al. (2023a). Further details on dataset collection and MD configuration provided in Appendix A.

Table 2: Number of sequences used per peptide length for training and evaluation.

Seq. length	2	3	4	5	6	7	8
Training	200	664	1457	1288	1967	2634	8867
Evaluation	16	—	30	—	—	—	30

4.2. Scale-transferability of ENSEMBLE

We train the first Boltzmann generators transferable across peptide sequence length. We train the ENSEMBLE architecture defined in Section 3.1, an unmodified TarFlow (Zhai et al., 2024) as in SBG (Tan et al., 2025), and the equivariant continuous normalizing flow of Klein & Noe (2024). For the ECNF we use the improved training recipe of Tan et al. (2025), denoted as ECNF++. All models are trained for 5×10^5 iterations with batch size 512. Both ENSEMBLE and TarFlow are suitably scalable to long sequences and are trained on the full dataset detailed in Section 4.1. However, sampling 8 residue sequences with ECNF++ was found prohibitively expensive, hence the training data was limited to sequences up to and including length 4. Comprehensive training details are provided in Appendix B, results for the unweighted proposal distributions are provided in Appendix C.

Table 1: Quantitative results for flows with importance sampling on peptide systems up to 8 residues. All flows evaluated using SNIS with a budget of 10^4 energy evaluations. Best values in **bold**.

Sequence length →	2AA (16 systems)			4AA (30 systems)				8AA (30 systems)			
Model ↓	ESS ↑	E-$\mathcal{W}_2$ ↓	$\mathcal{T}$-$\mathcal{W}_2$ ↓	ESS ↑	E-$\mathcal{W}_2$ ↓	$\mathcal{T}$-$\mathcal{W}_2$ ↓	TICA-$\mathcal{W}_2$ ↓	ESS ↑	E-$\mathcal{W}_2$ ↓	$\mathcal{T}$-$\mathcal{W}_2$ ↓	TICA-$\mathcal{W}_2$ ↓
ECNF	**0.173**	**0.500**	**0.285**	—	—	—	—	—	—	—	—
ENCF++	0.021	3.139	0.592	0.008	8.745	2.579	0.678	—	—	—	—
TarFlow	0.076	0.600	0.445	0.034	2.064	2.187	0.547	0.006	15.352	5.520	1.058
ENSEMBLE	0.119	0.523	0.379	**0.058**	**1.565**	**2.051**	**0.522**	**0.009**	**12.201**	**5.354**	**1.047**

To establish the performance of ENSEMBLE as a sampler proposal distribution, we first evaluate the trained flows in the Boltzmann generator setting. Here we generate a set of proposal particles $\{x_i\}_{i=1}^N$, evaluate model likelihoods $q_\theta(x_i)$ and reweight using SNIS as in Eq. (7). For all flows we permit a sampling budget of 10^4 energy evaluations. The primary evaluation metrics are the Wasserstein-2 distance on: (i) the energy distribution E-$\mathcal{W}_2$, (ii) the dihedral angle torus distribution $\mathcal{T}$-$\mathcal{W}_2$, (iii) the first 2 TICA component projections TICA-$\mathcal{W}_2$. The energy distribution is highly sensitive to perturbation in bond length and angle, hence E-$\mathcal{W}_2$ measures accuracy on fine-grained details. The dihedral angle tori and TICA projection describe macrostructure, hence $\mathcal{T}$-$\mathcal{W}_2$ and TICA-$\mathcal{W}_2$ measure accuracy in terms of metastable state coverage. We additionally report effective sample size (ESS); the variance of the importance weights. For metric definitions and further details on sampling evaluation procedure please refer to Appendix D.

We present metrics for ENSEMBLE and baseline methods in Table 1, where ECNF is the model trained by Klein & Noe (2024). Evidently on dipeptide sequences ECNF is the most performant proposal, achieving the highest ESS, lowest E-$\mathcal{W}_2$, and lowest $\mathcal{T}$-$\mathcal{W}_2$. However, ENSEMBLE has negligible reduction in $\mathcal{T}$-$\mathcal{W}_2$ and TICA-$\mathcal{W}_2$ despite being trained to model a much wider range of systems, and being orders of magnitude faster to sample. At scales of tetra and octopeptides ENSEMBLE achieves the strongest performance, confirming its efficacy as a proposal for peptide systems of varying sequence length. ECNF++ performs very poorly on both dipeptides and tetrapeptides, seemingly struggling to concurrently model systems of varying length. Fig. 1 confirms the success of ENSEMBLE with SNIS as an amortized sampler, achieving the best performance per-energy evaluation and per-hour on the critical TICA-$\mathcal{W}_2$ describing metastable state coverage, out performing both an MD baseline and ECNF++. We additionally present results on the unseen octopeptide `DGVAHALS` in Fig. 3, demonstrating the unprecedented scalability of ENSEMBLE; further results are provided in C.

4.3. Architecture ablation study

We proceed to ablate the TarFlow architectural variations applied in ENSEMBLE, as described in Section 3.1: (i) the adapt + transition blocks in the transformer layers, (ii) the backbone permutations interleaved into our permutation sequence. To reduce computational cost, we ablate by training on sequences of only 4 residues and less, all other training details are unchanged from Section 4.2. We present quantitative results for these modifications in Table 3. We see a significant improvement in effective samples size on dipeptides, and across all metrics on tetrapeptides. These results confirm the efficacy of these modifications for atomistic modeling, particularly the backbone permutations which introduce no additional runtime complexity over the standard TarFlow.

Table 3: Ablation results for ENSEMBLE architecture components on peptide systems up to 4 residues. SNIS is performed with a fixed budget of 2.5×10^4 energy evaluations. Improvement over standard TarFlow bolded.

Sequence length →	2AA (16 systems)			4AA 30 systems)			
Model ↓	ESS ↑	E-$\mathcal{W}_2$ ↓	$\mathcal{T}$-$\mathcal{W}_2$ ↓	ESS ↑	E-$\mathcal{W}_2$ ↓	$\mathcal{T}$-$\mathcal{W}_2$ ↓	TICA-$\mathcal{W}_2$ ↓
TarFlow base	0.115	0.287	0.261	0.041	1.272	1.670	0.614
Adapt + Transition	**0.147**	**0.283**	0.273	**0.053**	**0.932**	**1.657**	0.625
Backbone permutation	**0.146**	0.299	0.264	**0.056**	**1.097**	**1.652**	**0.541**

4.4. Sampling algorithms

Having established the unmatched scalable performance of ENSEMBLE in the standard Boltzmann generator framework, we now consider alternative sampling algorithms made tractable by its efficient likelihood computation. We evaluate SNIS, SMC in continuous time, SMC in discrete time, and the simple instantiation of self-refinement defined in Section 3.2. All methods are permitted a sampling budget of 10^6 energy evaluation, further details on the method configurations are provided in Appendix D. Quantitative results are presented in Table 4. These results reveal the surprising result that, given a suitably strong proposal distribution, SNIS is the strongest sampling algorithm, outperforming both SMC variants. The performance of SNIS with self-refinement at further improving the E-$\mathcal{W}_2$ provides strong evidence in favor of proposal refinement within sampling methods, as an alternative to resource allocation solely on annealing-based methods. We provide qualitative results for the unseen `SAEL` system in Fig. 4, illustrating the improved enhanced mode coverage of ENSEMBLE with SNIS over a MD baseline given an allocation of 10^6 energy evaluations. These results provide further confirmation of ENSEMBLE successful amortized sampling with ENSEMBLE.

Table 4: Results for sampling algorithms using ENSEMBLE as a proposal on peptide systems up to 4 residues. All algorithms provided with fixed budget of 10^6 energy evaluations. Best values bolded.

Sequence length →	2AA (16 systems)			4AA (30 systems)			
Algorithm ↓	ESS ↑	E-$\mathcal{W}_2$ ↓	$\mathcal{T}$-$\mathcal{W}_2$ ↓	ESS ↑	E-$\mathcal{W}_2$ ↓	$\mathcal{T}$-$\mathcal{W}_2$ ↓	TICA-$\mathcal{W}_2$ ↓
SNIS	0.121	0.369	**0.264**	0.058	1.137	**1.613**	**0.503**
SMC Continuous	—	2.673	0.508	—	12.307	2.313	0.687
SMC Discrete	—	0.371	0.445	—	1.012	1.972	0.569
SNIS + self-refinement	0.072	**0.327**	0.283	0.033	**0.829**	1.720	0.543

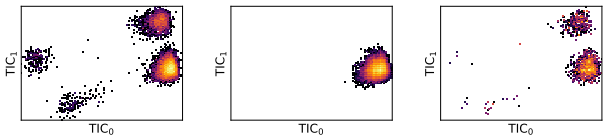

Figure 4: **By drawing *uncorrelated* proposal samples, ENSEMBLE achieves greater metastable state coverage than molecular dynamics for the same number of energy evaluations.** TICA projection plots for unseen tetrapeptide system (SAEL). After 10^9 energy evaluations molecular dynamics (left) has traversed four distinct metastable states, taken to be ground truth. However, with an energy evaluation budget of 10^6 molecular dynamics explores only a single metastable state, highlighting the limitations of simulation-based sampling methods for mode exploration. ENSEMBLE with SNIS wright) samples all 4 states given the same budget of energy evaluations, indicating successful amortization of the mode exploration problem.

Inference-time temperature transfer. We evaluate the scaled prior (SP) technique for inference-time temperature transfer introduced in Section 3.2. We collect additional 1 µs MD trajectories for the SAEL unseen tetrapeptide at temperatures defined by geometric series between the base model temperature of 310 K, and 800 K. We then perform SNIS using 2×10^5 energy evaluations from ENSEMBLE, both naively and with the scaled prior inference method. Results are presented in Fig. 5, with scaled prior universally outperforming naïve SNIS. We emphasize that scaled prior *does not* require any finetuning and introduces negligible increase in complexity at inference. These results thus demonstrate that ENSEMBLE is transferable not only in system, but also in temperature, opening a variety of avenues of further exploration.

5. Related Work

Normalizing Flows and Boltzmann Generators Normalizing flows (Rezende & Mohamed, 2015; Dinh et al., 2016; Durkan et al., 2019; Kingma & Dhariwal, 2018b; Kolesnikov et al., 2024; Zhai et al., 2024) fell out of favor as generative models as GANs (Goodfellow et al., 2014) and subsequently diffusion models (Song et al., 2021; Ho et al., 2020) showed better empirical generative quality. However, they have still found uses in scientific applications where efficient likelihood calculations are necessary. Boltzmann generators (Noé et al., 2019) leverage normalizing flows (both discrete and continuous (Chen et al., 2018)) for SNIS

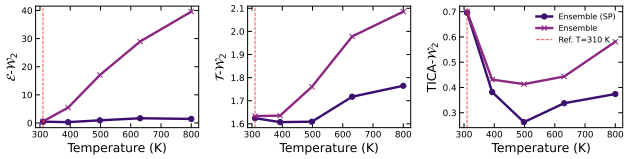

Figure 5: **Scaled prior greatly the ability of ENSEMBLE to accurately reweight to arbitrary temperatures.** Metrics for ENSEMBLE on SAEL unseen tetrapeptide, targeting the temperatures up to 800 K. Naïvely applying SNIS to the target temperature leads to a rapid degradation in energy distribution, and to a lesser extent the dihedral angle distribution. Applying prior scaling as defined in Section 3.2 leads to a significant improvement in energy distribution at high temperatures and moderate improvement in dihedral angles. Notably, the TICA distribution *improves* at higher temperatures irrespective of scaled prior usage, although scaled prior remains more effective.

given a target density for access to consistent independent samples. However, their scalability has been limited to relatively small problems until this work with transferability only demonstrated up to dipeptides prior to this work (Klein & Noe, 2024) with the use of continuous normalizing flows and flow matching (Lipman et al., 2023; Albergo et al., 2023; Liu, 2022), which is extremely expensive to sample from with likelihoods due to the need for divergence calculation (Grathwohl et al., 2019).

ML Accelerated MD sampling. Machine learning methods are a promising direction for accelerated sampling of molecular conformations. One line of work uses ML to predict longer-time transitions directly (e.g., TimeWarp (Klein et al., 2023a)), while another focuses on generative modeling to approximate the Boltzmann distribution from precollected data (Boltzmann emulators) (Wayment-Steele et al., 2024; Lewis et al., 2025; Jing et al., 2024). However, these generative models often lack the ability to compute exact likelihoods efficiently, making proper reweighting or free energy difference calculation difficult or impossible.

6. Conclusion

We demonstrate that deep learning-based sampling methods can efficiently transfer to previously unseen systems at the unprecedented biomolecular scale. Furthermore, they outperform conventional sampling algorithms such as MD when compared for the same computational budget and runtime. This fact re-evaluates the generalization abilities of learned samplers and establishes new avenues for their development.

Notably, ENSEMBLE demonstrates the state-of-the-art performance whilst keeping the simplicity of many design choices; thus, leaving a lot of room for improvement. Indeed, the introduced architectural changes do not restrict the architecture and can be adapted for other domains. Furthermore, the fact that standard SNIS achieves state-of-the-art performance indicates that the learned proposal is very close

to the target density. Clearly, this can be further improved by applying and carefully tuning advanced Monte Carlo methods, e.g. SMC. Finally, our self-refinement strategy simply trains on the samples collected via importance sampling and can be further improved by using more advanced Monte Carlo methods. In future work it would be interesting to train ENSEMBLE on larger, more diverse, and more realistic datasets such as the recent OMol25 (Levine et al., 2025) dataset.

Limitations Whilst conventional Monte Carlo algorithms make no assumption on the target density function, the transferability of learned samplers including ENSEMBLE depends on the assumption that the system is a member of some structured space of energy functions, most commonly the chemical space of molecules. The same applies for the transferability across temperatures. Despite demonstrating surprising abilities to sample from the higher temperature densities, we assume that precise transfer to, more challenging, lower temperatures would require additional conditioning, which ENSEMBLE currently lacks.

Acknowledgments

This research is partially supported by the EP- SRC Turing AI World-Leading Research Fellowship No. EP/X040062/1 and EPSRC AI Hub No. EP/Y028872/1. The authors acknowledge funding from UNIQUE, CIFAR, NSERC, Intel, and Samsung. The research was enabled in part by computational resources provided by the Digital Research Alliance of Canada (`https://alliancecan.ca`), Mila (`https://mila.quebec`), and NVIDIA. The authors acknowledge Hugging Face for hosting our dataset. KN was supported by IVADO and Institut Courtois.

References

Abramson, J., Adler, J., Dunger, J., Evans, R., Green, T., Pritzel, A., Ronneberger, O., Willmore, L., Ballard, A. J., Bambrick, J., Bodenstein, S. W., Evans, D. A., Hung, C.-C., O'Neill, M., Reiman, D., Tunyasuvunakool, K., Wu, Z., Žemgulytė, A., Arvaniti, E., Beattie, C., Bertolli, O., Bridgland, A., Cherepanov, A., Congreve, M., Cowen-Rivers, A. I., Cowie, A., Figurnov, M., Fuchs, F. B., Gladman, H., Jain, R., Khan, Y. A., Low, C. M. R., Perlin, K., Potapenko, A., Savy, P., Singh, S., Stecula, A., Thillaisundaram, A., Tong, C., Yakneen, S., Zhong, E. D., Zielinski, M., Žídek, A., Bapst, V., Kohli, P., Jaderberg, M., Hassabis, D., and Jumper, J. M. 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.

Akhound-Sadegh, T., Rector-Brooks, J., Bose, A. J., Mittal, S., Lemos, P., Liu, C.-H., Sendera, M., Ravanbakhsh, S., Gidel, G., Bengio, Y., Malkin, N., and Tong, A. Iterated Denoising Energy Matching for Sampling from Boltzmann Densities, February 2024. URL `http://arxiv.org/abs/2402.06121`. arXiv:2402.06121 [cs, stat].

Albergo, M. S. and Vanden-Eijnden, E. Nets: A non-equilibrium transport sampler. *arXiv preprint arXiv:2410.02711*, 2024.

Albergo, M. S., Boffi, N. M., and Vanden-Eijnden, E. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint 2303.08797*, 2023.

Buch, I., Giorgino, T., and De Fabritiis, G. Complete reconstruction of an enzyme-inhibitor binding process by molecular dynamics simulations. *Proceedings of the National Academy of Sciences*, 108 (25):10184–10189, June 2011. doi: 10.1073/pnas.1103547108. URL `https://www.pnas.org/doi/abs/10.1073/pnas.1103547108`. Company: National Academy of Sciences Distributor: National Academy of Sciences Institution: National Academy of Sciences Label: National Academy of Sciences Publisher: Proceedings of the National Academy of Sciences.

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

Del Moral, P. Mean field simulation for monte carlo integration. *Monographs on Statistics and Applied Probability*, 126(26):6, 2013.

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.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. Neural spline flows. *Advances in neural information processing systems*, 32, 2019.

Eastman, P., Swails, J., Chodera, J. D., McGibbon, R. T., Zhao, Y., Beauchamp, K. A., Wang, L.-P., Simmonett, A. C., Harrigan, M. P., Stern, C. D., Wiewiora, R. P., Brooks, B. R., and Pande, V. S. OpenMM 7: Rapid development of high performance algorithms for molecular dynamics. *PLoS computational biology*, 13(7):e1005659, July 2017. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1005659.

Flamary, R., Courty, N., Gramfort, A., Alaya, M. Z., Boisbunon, A., Chambon, S., Chapel, L., Corenflos, A., Fatras, K., Fournier, N., Gautheron, L., Gayraud, N. T. H., Janati, H., Rakotomamonjy, A., Redko, I., Rolet, A., Schutz, A., Seguy, V., Sutherland, D. J., Tavenard, R., Tong, A., and Vayer, T. POT: Python Optimal Transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021. ISSN 1533-7928. URL http://jmlr.org/papers/v22/20-451.html.

Geffner, T., Didi, K., Zhang, Z., Reidenbach, D., Cao, Z., Yim, J., Geiger, M., Dallago, C., Kucukbenli, E., Vahdat, A., and Kreis, K. Proteina: Scaling Flow-based Protein Structure Generative Models. October 2024. URL https://openreview.net/forum?id=TVQLu34bdw&noteId=ypeoraSUA0.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks. In *NeurIPS*, 2014.

Grathwohl, W., Chen, R. T. Q., Bettencourt, J., Sutskever, I., and Duvenaud, D. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *International Conference on Learning Representations (ICLR)*, 2019.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

Jarzynski, C. Nonequilibrium equality for free energy differences. *Physical Review Letters*, 78(14):2690, 1997.

Jing, B., Stärk, H., Jaakkola, T., and Berger, B. Generative Modeling of Molecular Dynamics Trajectories. *Advances in Neural Information Processing Systems*, 37:40534–40564, December 2024.

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

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions, 2018b. URL https://arxiv.org/abs/1807.03039.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems*, 29, 2016.

Klein, L. and Noe, F. Transferable Boltzmann Generators. *Advances in Neural Information Processing Systems*, 37:45281–45314, December 2024.

Klein, L., Foong, A., Fjelde, T., Mlodozeniec, B., Brockschmidt, M., Nowozin, S., Noé, F., and Tomioka, R. Timewarp: Transferable acceleration of molecular dynamics by learning time-coarsened dynamics. *Advances in Neural Information Processing Systems*, 36:52863–52883, 2023a.

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

Kolesnikov, A., Pinto, A. S., and Tschannen, M. Jet: A modern transformer-based normalizing flow. *arXiv preprint arXiv:2412.15129*, 2024.

Köhler, J., Invernizzi, M., Haan, P. D., and Noe, F. 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.

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

Levine, D. S., Shuaibi, M., Spotte-Smith, E. W. C., Taylor, M. G., Hasyim, M. R., Michel, K., Batatia, I., Csányi, G., Dzamba, M., Eastman, P., Frey, N. C., Fu, X., Gharakhanyan, V., Krishnapriyan, A. S., Rackers, J. A., Raja, S., Rizvi, A., Rosen, A. S., Ulissi, Z., Vargas, S., Zitnick, C. L., Blau, S. M., and Wood, B. M. The open molecules 2025 (omol25) dataset, evaluations, and models, 2025. URL https://arxiv.org/abs/2505.08762.

Lewis, S., Hempel, T., Jiménez-Luna, J., Gastegger, M., Xie, Y., Foong, A. Y. K., Satorras, V. G., Abdin, O., Veeling, B. S., Zaporozhets, I., Chen, Y., Yang, S., Schneuing, A., Nigam, J., Barbero, F., Stimper, V., Campbell, A., Yim, J., Lienen, M., Shi, Y., Zheng,

S., Schulz, H., Munir, U., Tomioka, R., Clementi, C., and Noé, F. Scalable emulation of protein equilibrium ensembles with generative deep learning. *bioRxiv*, 2025. doi: 10.1101/2024.12.05.626885. URL https://www.biorxiv.org/content/early/2025/02/25/2024.12.05.626885.

Lindorff-Larsen, K., Piana, S., Dror, R. O., and Shaw, D. E. How Fast-Folding Proteins Fold. *Science*, 334(6055):517–520, October 2011. doi: 10.1126/science.1208351. URL https://www.science.org/doi/10.1126/science.1208351. Publisher: American Association for the Advancement of Science.

Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling. *International Conference on Learning Representations (ICLR)*, 2023.

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

Liu, Q. Rectified flow: A marginal preserving approach to optimal transport. *arXiv preprint arXiv:2209.14577*, 2022.

Loshchilov, I. and Hutter, F. Decoupled Weight Decay Regularization. September 2018. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Midgley, L., Stimper, V., Antorán, J., Mathieu, E., Schölkopf, B., and Hernández-Lobato, J. M. SE(3) Equivariant Augmented Coupling Flows. *Advances in Neural Information Processing Systems*, 36:79200–79225, December 2023.

Neal, R. M. Annealed importance sampling. *Statistics and computing*, 11:125–139, 2001.

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.

Noé, F., Schütte, C., Vanden-Eijnden, E., Reich, L., and Weikl, T. R. Constructing the equilibrium ensemble of folding pathways from short off-equilibrium simulations. *Proceedings of the National Academy of Sciences*, 106 (45):19011–19016, November 2009. doi: 10.1073/pnas.0905466106. URL https://www.pnas.org/doi/abs/10.1073/pnas.0905466106. Company: National Academy of Sciences Distributor: National Academy of Sciences Institution: National Academy of Sciences Label: National Academy of Sciences Publisher: Proceedings of the National Academy of Sciences.

Papamakarios, G., Pavlakou, T., and Murray, I. Masked autoregressive flow for density estimation. *Advances in neural information processing systems*, 30, 2017.

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

Roberts, G. O. and Tweedie, R. L. Exponential convergence of langevin distributions and their discrete approximations. 1996.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations, 2021. URL https://arxiv.org/abs/2011.13456.

Tan, C. B., Bose, A. J., Lin, C., Klein, L., Bronstein, M. M., and Tong, A. Scalable Equilibrium Sampling with Sequential Boltzmann Generators, February 2025. URL http://arxiv.org/abs/2502.18462. arXiv:2502.18462 [cs].

Vargas, F., Grathwohl, W., and Doucet, A. Denoising diffusion samplers. *arXiv preprint arXiv:2302.13834*, 2023.

Wayment-Steele, H. K., Ojoawo, A., Otten, R., et al. Predicting multiple conformations via sequence clustering and alphafold2. *Nature*, 625:832–839, 2024. doi: 10.1038/s41586-023-06832-9. URL https://doi.org/10.1038/s41586-023-06832-9.

Zhai, S., Zhang, R., Nakkiran, P., Berthelot, D., Gu, J., Zheng, H., Chen, T., Bautista, M. A., Jaitly, N., and Susskind, J. Normalizing Flows are Capable Generative Models, December 2024. URL http://arxiv.org/abs/2412.06329. arXiv:2412.06329 [cs].

A. Dataset

Sequence sampling Training sequences are collected for peptide lengths 3, 5, 6, 7, and 8 by uniformly sampling the 20 standard amino acids. For the 8-residue test data, a sequence of length $30 \cdot 8 = 240$ is constructed by concatenating 12 of each amino acid. This sequence is then randomly permuted and split into peptides of length 8, ensuring that each amino acid is represented uniformly. In both training and test sets, the N- and C-terminal residues are protonated to form the zwitterionic state of the peptides. Initial structure files (PDB format) are generated using AmberTools' `tleap`.

Molecular dynamics simulation Local energy minimization is performed with the Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BGFS) algorithm. Energy minimization is followed by burn-in simulation of length $50\,\mathrm{ps}$, after which samples are collected every $5\,\mathrm{ps}$ (train) or $10\,\mathrm{ps}$ (test) until the simulation budget is exhausted. Full MD simulation parameters are provided in Table 5.

Table 5: `OpenMM` simulation parameters.

Force field	`amber-14`
Integration time step	1 fs
Friction coefficient	$0.3\,\mathrm{ps}^{-1}$
Temperature	$310\,\mathrm{K}$
Nonbonded method	`CutoffNonPeriodic`
Nonbonded cutoff	$2\,\mathrm{nm}$
Integrator	`LangevinMiddleIntegrator`

Table 6: Training and evaluation dataset parameters.

	Train	Test
Burn-in period	$50\,\mathrm{ps}$	$50\,\mathrm{ps}$
Sampling interval	$5\,\mathrm{ps}$	$10\,\mathrm{ps}$
Simulation time	$50\,\mathrm{ns}$	$1\,\mu\mathrm{s}$

B. Training details

All models are trained for $5 \cdot 10^5$ iterations using a batch size of 512 with the AdamW optimizer (Loshchilov & Hutter, 2018). We employ a cosine learning rate schedule in which the initial and final learning rates are a reduction of the maximal value by factor of 500, as well as exponential moving average with decay of 0.999. No overfitting was observed hence no early stopping was required. Samples are normalized using a factor computed as the mean standard deviation across all training samples, noting that a single value must be shared across systems of different dimensionality. An overview of all training configurations is provided in Table 7.

Continuous Normalizing Flows We use the ECNF++ training recipe defined by Tan et al. (2025); this entails a learning rate of $5 \cdot 10^{-4}$ and weight decay of $1 \cdot 10^{-2}$, with default AdamW hyperparameters of AdamW β_1, β_2 of $(0.9, 0.999)$. In contrast the ECNF of Klein & Noe (2024) was trained without weight decay or exponential weight averaging. The channel width and layer depth of both models is defined in Table 8.

TarFlows Following Zhai et al. (2024) and Tan et al. (2025) we use a learning rate of $1 \cdot 10^{-4}$, weight decay of $4 \cdot 10^{-4}$, and AdamW β_1, β_2 of $(0.9, 0.95)$. Data augmentation is applied as random rotations and Gaussian center of mass augmentation, in which every the entire system conformation is translated by a vector $c \sim \mathcal{N}(0, \sigma^2 I_3)$. The σ^2 value is chosen to match that of the prior, which has a center of mass $\sigma^2 = \frac{1}{N}$ where N is the number of atoms. Given N is in our case variable for a single model trained on multiple systems, we use the mean value across training samples. The architecture width and depth is provided in Table 8.

B.1. Computational requirements

All training experiments are run on a heterogeneous cluster of NVIDIA H100 and L40S GPUs using distributed data parallelism. The training throughput for each model is presented in Table 9.

Table 7: Overview of training configurations.

	ECNF	ECNF++	TarFlow / ENSEMBLE
Learning Rate	$5 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$1 \cdot 10^{-4}$
Weight Decay	0.0	$1 \cdot 10^{-2}$	$4 \cdot 10^{-4}$
β_1, β_2	0.9, 0.999	0.9, 0.999	0.9, 0.95
EMA Decay	0.0	0.999	0.999

Table 8: Overview of model scaling parameters. For TarFlow variants depth corresponds to number of parameterized transformations, for ECNF variants this is simply the number of graph neural network layers.

	ECNF	ECNF++	TarFlow	ENSEMBLE
Channels	128	256	384	384
Depth	9	9	8	8
Layers per block	N/A	N/A	8	8
Parameters (M)	1	4	115	285

Table 9: Training throughput for models presented in Table 1. We highlight ECNF++ to be trained only on sequences up to length 4, whereas TarFlow and ENSEMBLE are trained on sequences up to length 8.

	ECNF++	TarFlow	ENSEMBLE
Training iterations / H100 hour	960	1132	260

C. Additional results

C.1. Proposal Performance

We compare the performance of ECNF++, TarFlow, and ENSEMBLE before and after SNIS (with 10^4 samples) in Table 10. Evidently, ECNF++ demonstrates stronger performance with the initial proposals across a majority of metrics. In contrast, TarFlow and ENSEMBLE perform poorly on the E-$\mathcal{W}_2$ without reweighting, but achieve comparable results to ECNF++ on the $\mathcal{T}$-$\mathcal{W}_2$ and TICA-$\mathcal{W}_2$. Notably, the macrostructure $\mathcal{T}$-$\mathcal{W}_2$ and TICA-$\mathcal{W}_2$ metrics are in most cases *worse* after resampling irrespective of the proposal flow used, a phenomenon we attribute to the small size of the particle set.

Table 10: Quantitative results for flows comparing the proposal performance and performance after importance sampling on peptide systems up to 4 residues. SNIS performed with a budget of 10^4 energy evaluations.

Sequence length →		2AA (16 systems)		4AA (30 systems)		
Model ↓		E-$\mathcal{W}_2$ ↓	$\mathcal{T}$-$\mathcal{W}_2$ ↓	E-$\mathcal{W}_2$ ↓	$\mathcal{T}$-$\mathcal{W}_2$ ↓	TICA-$\mathcal{W}_2$ ↓
ENCF++	Proposal	303.423	0.273	$3.13 \cdot 10^8$	1.560	0.506
	SNIS	3.139	0.592	8.745	2.579	0.678
TarFlow	Proposal	$1.390 \cdot 10^{11}$	0.301	$5.931 \cdot 10^{14}$	1.577	0.479
	SNIS	0.600	0.445	2.064	2.187	0.547
ENSEMBLE	Proposal	$1.765 \cdot 10^{15}$	0.294	$7.444 \cdot 10^{15}$	1.539	0.719
	SNIS	0.523	0.379	1.565	2.051	0.522

C.2. Sequence-wise performance

Here we present sequence-wise performance for ECNF++, standard TarFlow, and ENSEMBLE using SNIS with 10^4 samples for the unseen peptides at sequence lengths 2, 4 and 8.

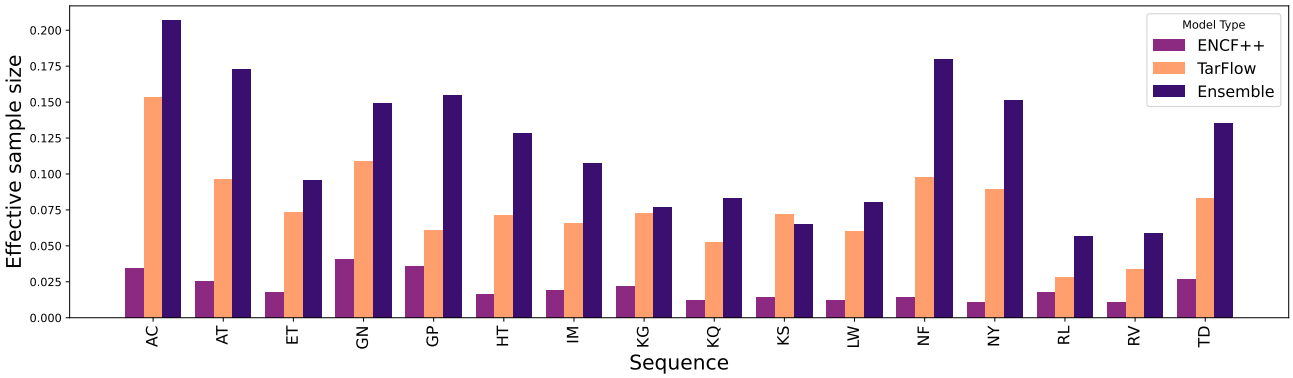

Figure 6: Dipeptide effective sample size. SNIS with 10^4 samples.

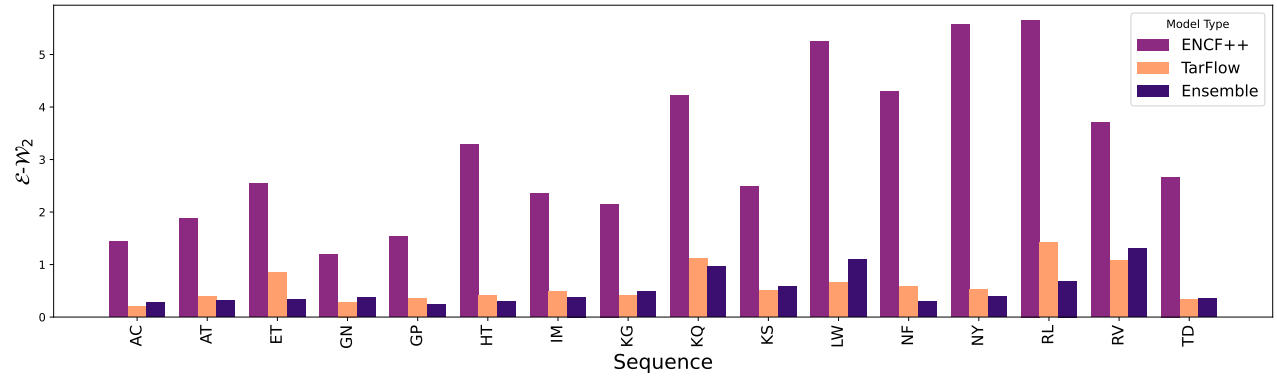

Figure 7: Dipeptide energy Wasserstein-2 distance. SNIS with 10^4 samples.

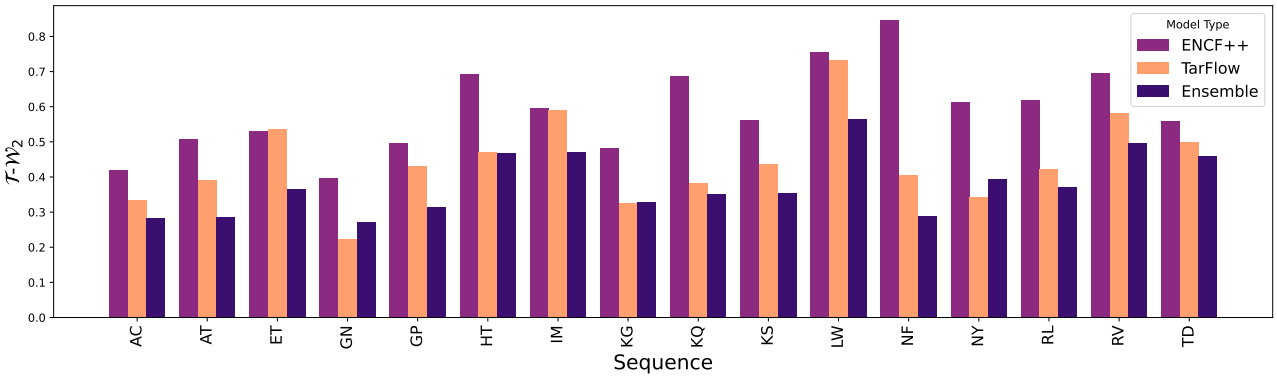

Figure 8: Dipeptide dihedral torus Wasserstein-2 distance. SNIS with 10^4 samples.

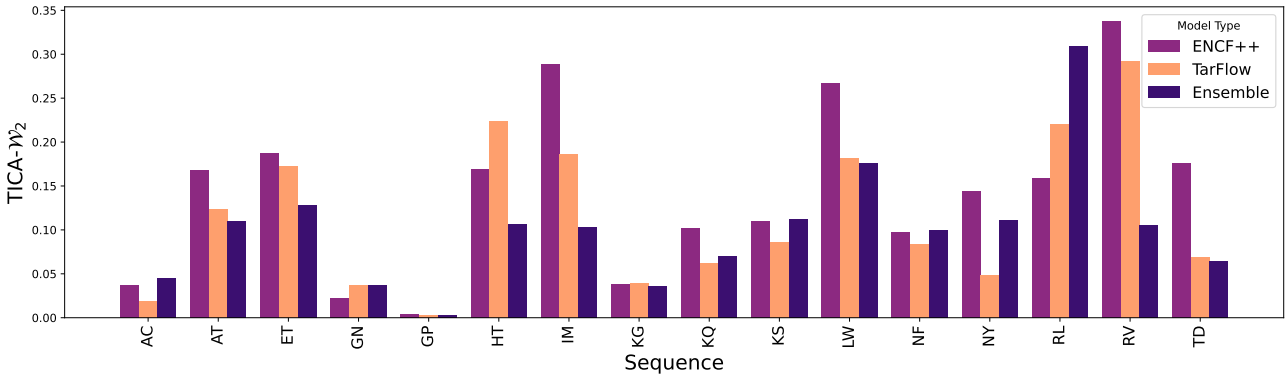

Figure 9: Dipeptide TICA Wasserstein-2 distance. SNIS with 10^4 samples.

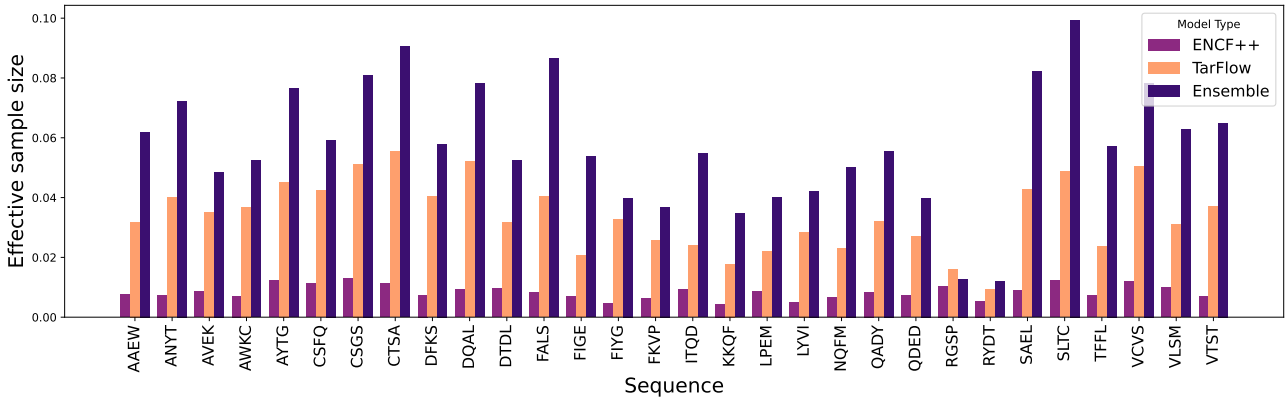

Figure 10: Tetrapeptide effective sample size. SNIS with 10^4 samples.

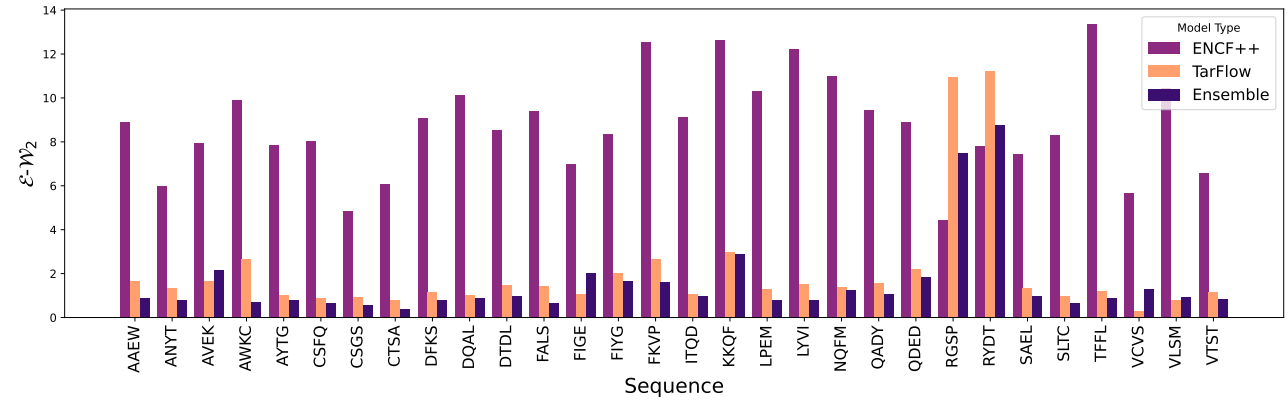

Figure 11: Tetrapeptide energy Wasserstein-2 distance. SNIS with 10^4 samples.

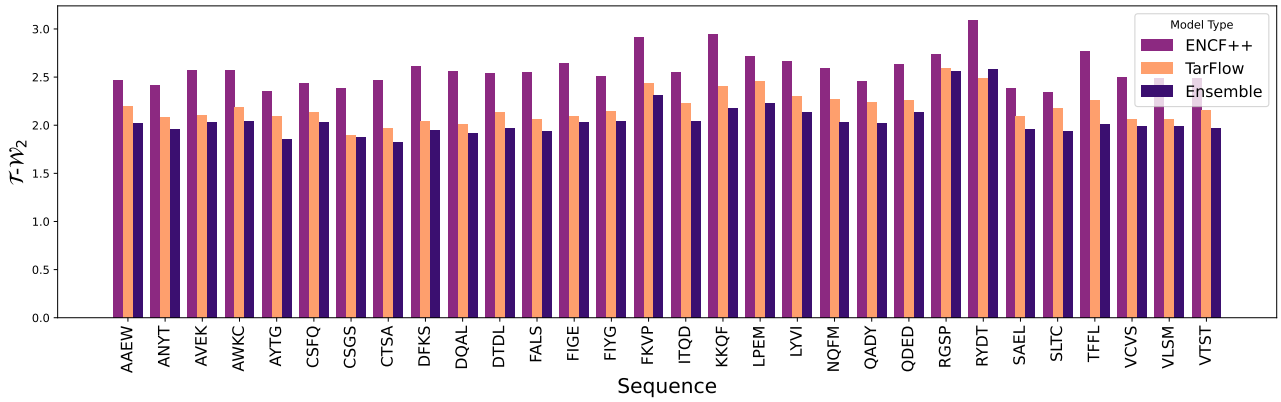

Figure 12: Tetrapeptide dihedral torus Wasserstein-2 distance. SNIS with 10^4 samples.

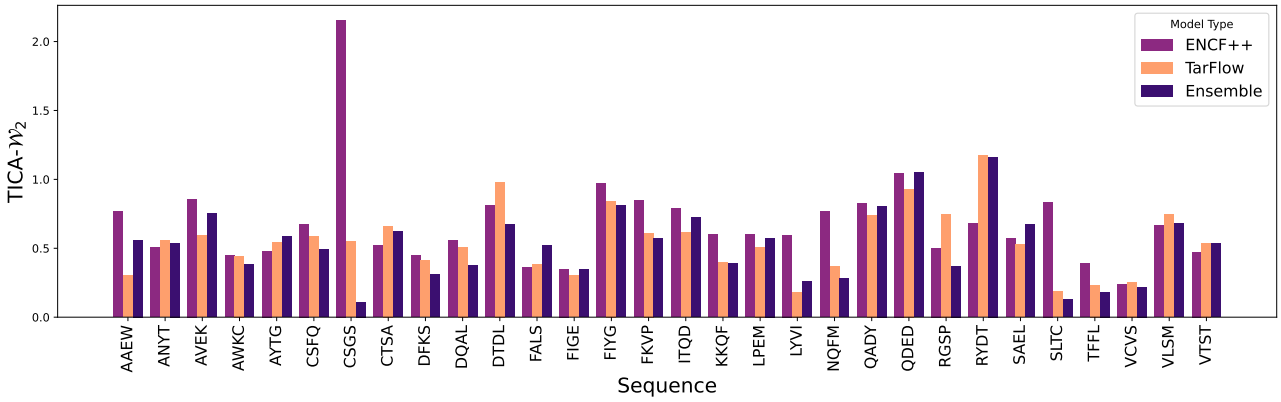

Figure 13: Tetrapeptide TICA Wasserstein-2 distance. SNIS with 10^4 samples.

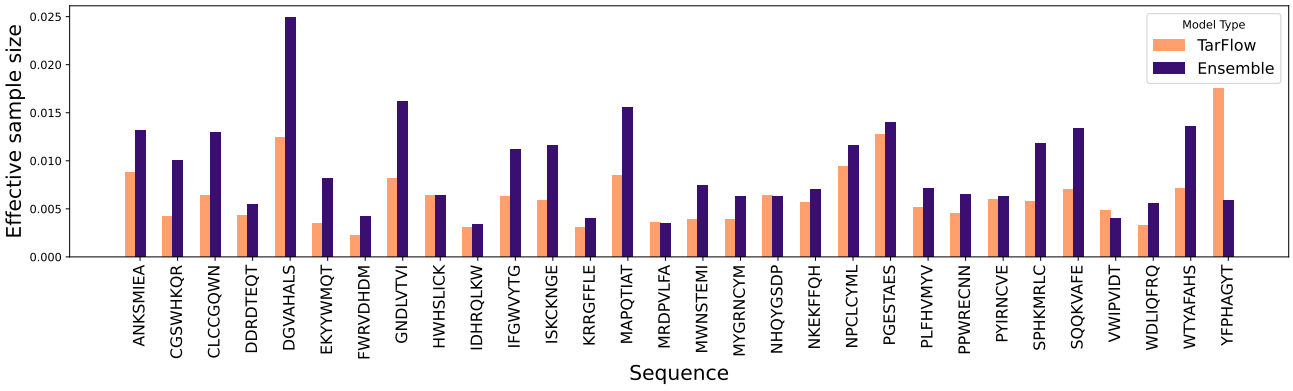

Figure 14: Octopeptide effective sample size. SNIS with 10^4 samples.

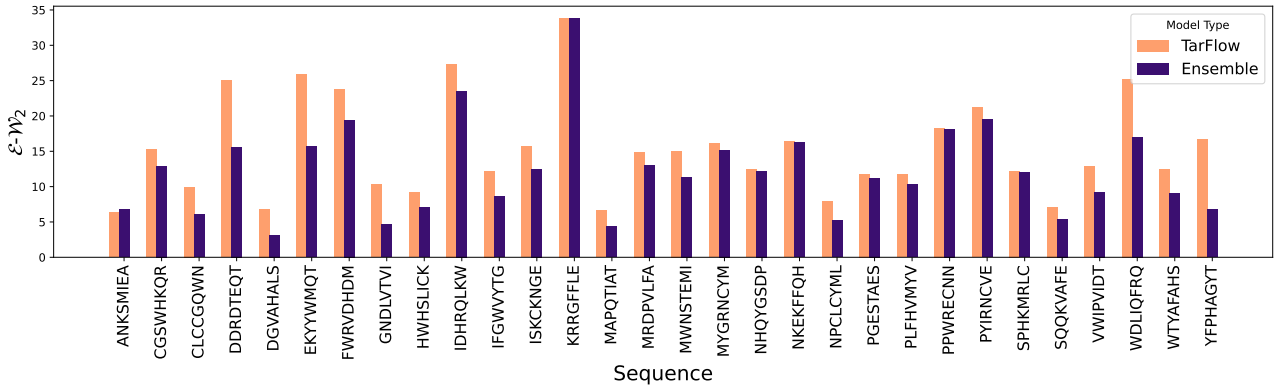

Figure 15: Octopeptide energy Wasserstein-2 distance. SNIS with 10^4 samples.

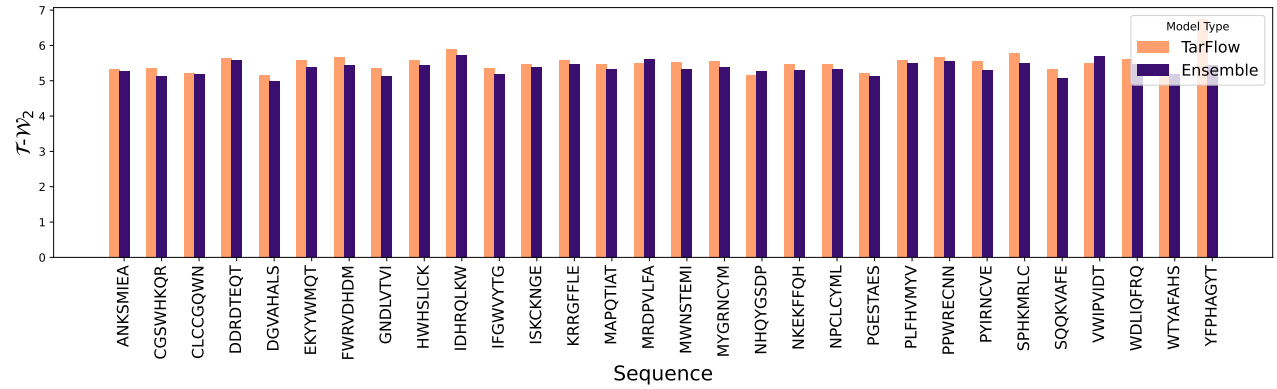

Figure 16: Octopeptide dihedral torus Wasserstein-2 distance. SNIS with 10^4 samples.

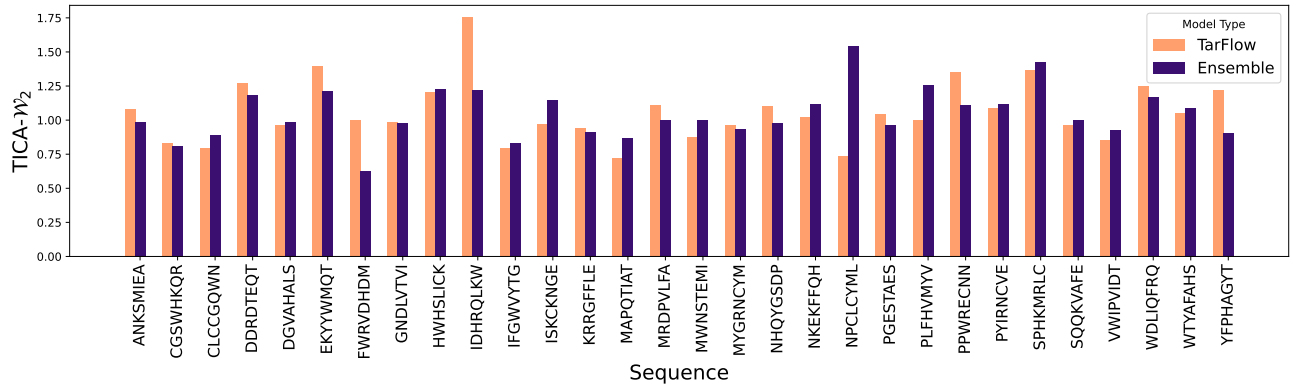

Figure 17: Octopeptide TICA Wasserstein-2 distance. SNIS with 10^4 samples.

C.3. Octopeptide Ramachandran plots

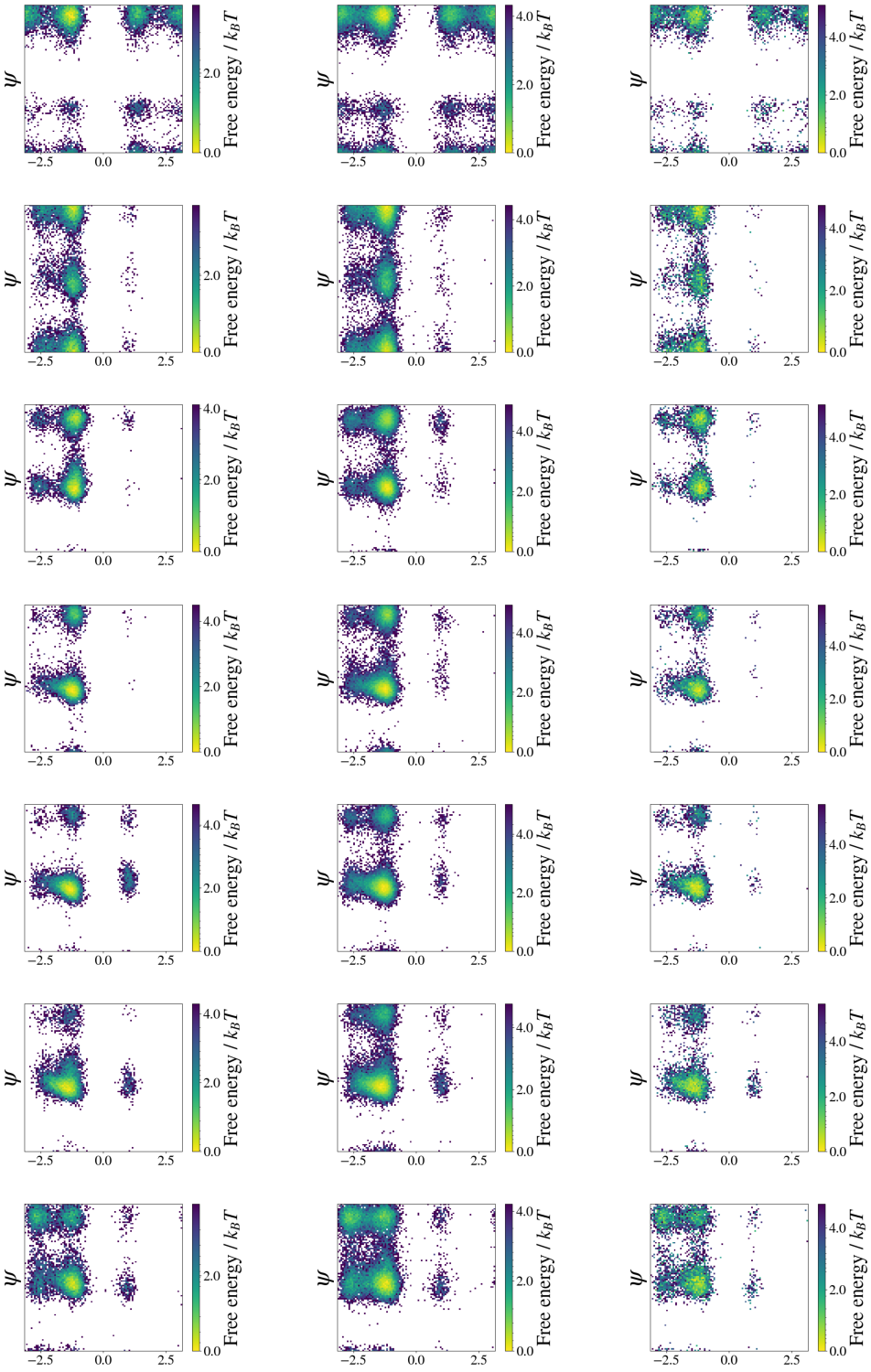

Figure 18: Ramachandran plots for DGVAHALS unseen octopeptide system. Ground truth (left column), ENSEMBLE proposal (center column), ENSEMBLE SNIS with 10^5 samples (right column).

C.4. Temperature Plots

We present TICA plots in Fig. 19 and energy distributions in Fig. 20 across a range of temperatures (310K, 393K, 498K, 631K, 800K). At each temperature, we generate $2 \cdot 10^5$ samples by scaling the prior with the inverse temperature β, sampling from $\mathcal{N}(0, 1/\beta^2)$. For SNIS, we use the energy at the corresponding temperature to reweight the samples.

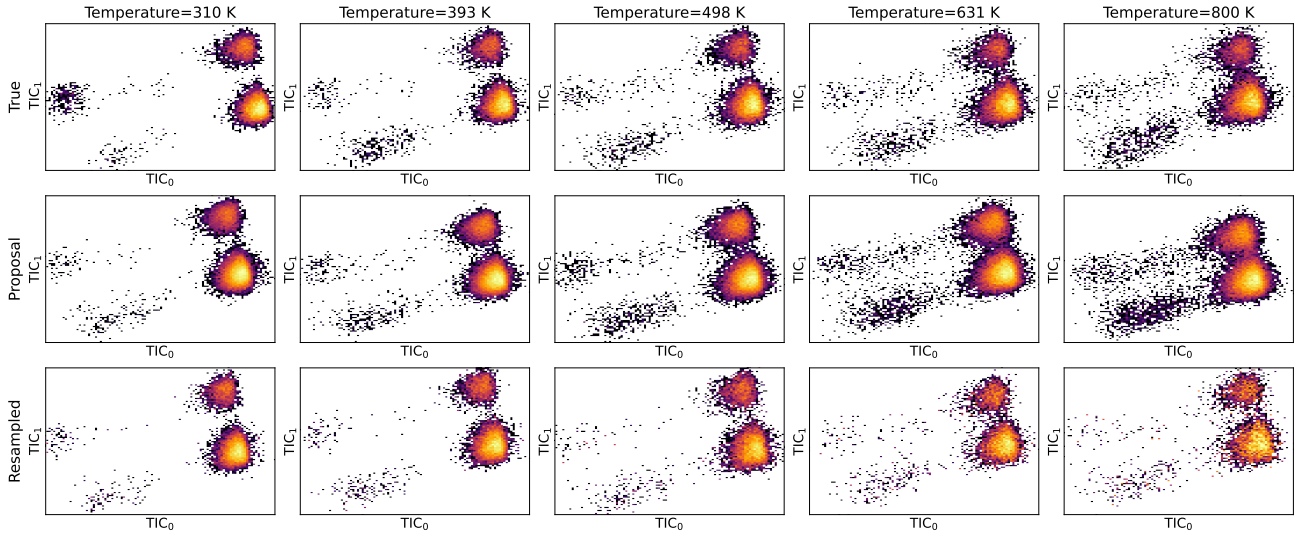

Figure 19: TICA plots for SAEL for different temperatures. Ground Truth MD (top row), ENSEMBLE proposal (middle row), ENSEMBLE SNIS (bottom row) with $2 \cdot 10^5$ samples.

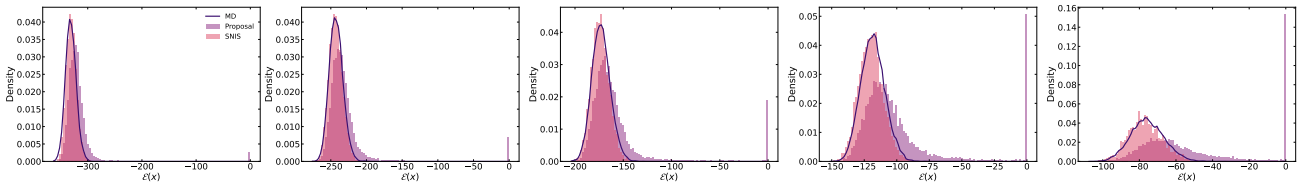

Figure 20: Energy histogram plots for SAEL for different temperatures.

D. Evaluation details

D.1. Proposal sampling and likelihood evaluation

Equivariant continuous normalizing flows Sampling from a continuous normalizing flow (CNF) involves solving the ODE defined by the parameterized vector field $u_t : [0, 1] \times \mathbb{R}^{n\times 3} \to \mathbb{R}^{n\times 3}$

$$\frac{dx_t}{dt} = u_t(x_t), \quad x_0 \sim p_0 \tag{11}$$

The corresponding likelihoods can be obtained using the instantaneous change of variables formula

$$\log p_1(x_1) = \log p_0(x_0) - \int_0^1 \nabla \cdot u_t(x_t) dt, \tag{12}$$

where $\nabla\cdot$ is the divergence operator. In practice both Eq. (11) and Eq. (12) can be integrated simultaneously with an ordinary differential equation (ODE) solver. We use the Dormand–Prince-5 (dopri5) adaptive solver in all ECNF experiments. Given the $E(3)$ equivariance of the ECNF, samples are generated in both possible global chiralities. Following Klein & Noe (2024) we check for incorrect global sample chirality and flip samples appropriately to match the L-amino acids present in the evaluation data. Unlike Klein & Noe (2024) we do not omit any samples with unresolvable chirality. We additionally apply logit clipping, removing the samples with the 0.2% highest importance weights before resampling Midgley et al. (2023).

TarFlow variants As discussed in Section 2.1, samples are generated from a normalizing flow simply by applying f_θ to prior samples $z \sim \mathcal{N}(0, I_{N\times D})$. Model likelihoods are obtained using the change of variables formula (Eq. (1)). Given the lack of translation equivariance in the TarFlow, and the data augmentation applied during training, samples are generated with an approximate scaled χ_3 distribution over centroid norm $||c|| = ||\frac{1}{N}\sum_{i=1}^{N} x_{i,:}|| \sim \sigma\chi_3$. This leads to adverse behavior when resampling with finite samples, hence we apply the center of mass adjustment of (Tan et al., 2025), in which the χ_3 probability density function is divided out of the proposal likelihoods

$$\log p_\theta^c(x) = \log p_\theta(x) - \left[\log\left(\frac{\|c\|^2}{\sigma^3}\right) + \frac{\|c\|^2}{2\sigma^2} - \log\left(\sqrt{2}\Gamma\left(\frac{3}{2}\right)\right)\right] \tag{13}$$

where Γ is the gamma function. This adjustment seeks to account for the radial component introduced by translation non-equivariance. We additionally apply the same weight clipping threshold as in the ECNF when performing SNIS, or before SMC.

D.2. Metrics

We report both effective sample size and a variety of Wasserstein-2 distances as evaluation metrics. For the Wasserstein distances a subsample of 10^4 samples are randomly sampled from the evaluation trajectory as ground truth. Similarly, at most 10^4 generated samples are employed; if a method has generated more samples a random subset is drawn without replacement.

Effective sample size We compute the effective sample size (ESS) using Kish's formula, normalized by the number of samples generated

$$\mathrm{ESS}\big(\{w_i\}_{i=1}^N\big) = \frac{\left(\sum_{i=1}^N w_i\right)^2}{N\sum_{i=1}^N w_i^2}. \tag{14}$$

Empirical Wasserstein distance We compare generated samples to ground truth data, collected as defined in Appendix A, using empirical Wasserstein-2 distances. Given empirical distributions $\mu = \frac{1}{n}\sum_{i=1}^n \delta_{x_i}$ and $\nu = \frac{1}{m}\sum_{j=1}^m \delta_{y_j}$, the empirical Wasserstein-2 distance is defined as

$$W_2(\mu,\nu) = \min_{\pi\in\Pi(\mu,\nu)} \sqrt{\sum_{i=1}^n\sum_{j=1}^m \pi_{ij}\, c(x_i,y_j)^2} \tag{15}$$

where $\Pi(\mu,\nu)$ denotes the set of couplings with marginals μ and ν, and $c(x,y)^2$ is a defined cost function. Different choices of $c(x,y)^2$ define different measures of dissimilarity. We use the `POT` (Flamary et al., 2021) linear optimal transport solver to compute the optimal couplings.

Energy cost The energy of a sample $E(x)$ is sensitive to both bonded forces and non-bonded forces. For the energy Wasserstein-2 distance $E\text{-}\mathcal{W}_2$ the cost function is simply

$$c_E(x,y)^2 = |E(x) - E(y)|^2 \tag{16}$$

Dihedral torus cost The ϕ and ψ backbone dihedral angles of a peptide conformation encode essential information regarding secondary and tertiary structure. We compare generated and ground truth samples in angle space by defining the dihedral angle vector

$$\mathrm{Dihedrals}(x) = (\phi_1, \psi_1, \phi_2, \psi_2, \ldots, \phi_{L-1}, \psi_{L-1}) \tag{17}$$

where L is the number of residues. Given the torus geometry implied by angle periodicity $\phi_i \in (-\pi, \pi]$, a natural cost function is the minimal signed angle difference

$$c_{\mathcal{T}}(x,y)^2 = \sum_{i=1}^{2L}[(\mathrm{Dihedrals}(x)_i - \mathrm{Dihedrals}(y)_i + \pi) \bmod 2\pi - \pi]^2. \tag{18}$$

This metric captures the geometric dissimilarity in dihedral angle space, respecting periodicity.

Time-lagged independent component analysis cost The time-lagged independent component analysis (TICA) projection of time-series data captures directions along which the data exhibits maximal autocorrelation. Within molecular dynamics, TICA is commonly used to detect distinct metastable states. Given mean-free time series data $\tilde{x}_t$, the instantaneous (zero-lag) empirical covariance and time-lagged empirical covariance matrix (at lag time τ) are computed as

$$\hat{C}_{00} = \frac{1}{T-\tau}\sum_{t=1}^{T-\tau} \tilde{x}_t \tilde{x}_t^\top, \quad \hat{C}_{0\tau} = \frac{1}{T-\tau}\sum_{t=1}^{T-\tau} \tilde{x}_t \tilde{x}_{t+\tau}^\top. \tag{19}$$

TICA seeks linear projection vectors $w \in \mathbb{R}^n$ that maximize autocorrelation at lag τ

$$\max_w \frac{w^\top \hat{C}_{0\tau} w}{w^\top \hat{C}_{00} w}. \tag{20}$$

The solution to which is obtained by solving the generalized eigenvalue problem

$$C_{0\tau}\lambda = \lambda C_{00} w, \tag{21}$$

where the eigenvalue λ measures the autocorrelation of the projected component at lag τ, and the eigenvector w defines the corresponding slow mode. To define the TICA Wasserstein-2 distance $\mathrm{TICA}\text{-}\mathcal{W}_2$ we take the full evaluation trajectory *without subsampling* and solve Eq. (21) to obtain the first two TICA projection vectors w_1, w_2. We may then define the following cost function

$$c_{\mathrm{TICA}}(x,y)^2 = \sum_{j=1}^{2} \left[w_j^\top x - w_j^\top y\right]^2 \tag{22}$$

defining similarity in TICA projection space. We emphasize that the TICA projection vectors are only computed on the (full) evaluation trajectory, but that the samples y are restricted to the 10^4 sample subset. In practice we compute the TICA projection for the heavy atom subspace only.

D.3. Sampling algorithm configurations

In this section we define configurations for the sampling algorithms presented in Table 4, as well as the molecular dynamics baseline used in Fig. 1 and Fig. 4. In particular the allocation of the 10^6 energy evaluations within the method is defined.

Molecular dynamics baseline We follow the same procedure used for collecting the main datasets defined in Appendix A, where the parameters defined in Table 5 are unchanged. However, for the baseline the burn-in period is reduced to $5\,\mathrm{ps}$ and sampling interval is reduced to $10\,\mathrm{fs}$. The energy evaluations used by L-BFGS minimization are not counted towards the budget of 10^6 but the burn-in iterations are counted, such that the collected trajectory represents $0.995\,\mathrm{ns}$ of simulation.

Annealed Importance Sampling We allocate the budget of 10^6 energy evaluations by generating a proposal set of 10^4 samples and performing 100 steps of annealing with resampling at every step. For the continuous variant, Langevin dynamics is used with a step size σ_t of 10^{-7}. Details of the formulation are found in Appendix E.2. For the discrete variant, we apply Langevin dynamics with a step size of 10^{-5}, followed by a Metropolis-Hastings step to accept or reject proposals. We assume sufficient smoothness in the intermediate densities to adaptively update the step size. Specifically, the step size is dynamically adjusted to maintain a Metropolis-Hastings acceptance rate of approximately 60%. Further details of discrete-time AIS are found in Appendix E.3.

Self-refinement We perform 4 rounds of self-refinement. In each round, we spend a portion of the budget to generate $2 \cdot 10^5$ samples and reweight using SNIS. The resulting reweighted samples are then used to finetune the model for 250 gradient steps with a batch size of 256. Notably, the finetuning does not spend the allocated budget as it does not involve energy evaluations. After the final round, the remaining computational budget is allocated to generate a final set of $2 \cdot 10^5$ samples, which are again SNIS reweighted to yield the empirical distribution $\tilde{p}(x|s)$ for a given system s.

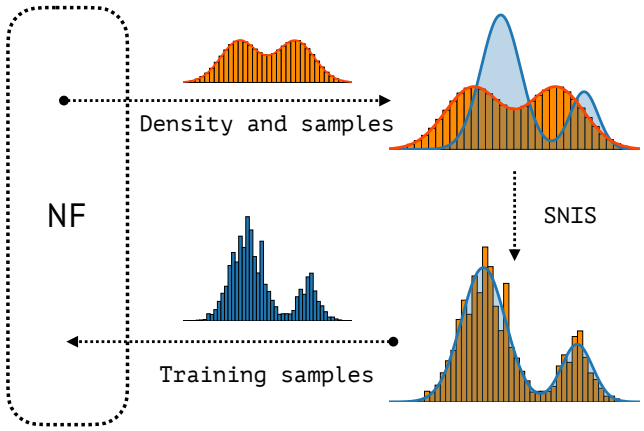

Figure 21: **Self-Refinement procedure**. A pre-trained ENSEMBLE is finetuned at inference-time by iteratively generating samples, reweighting them using SNIS, and training on the reweighted samples.

D.4. Computational requirements

All evaluation experiments are run on a heterogeneous cluster of NVIDIA L40S and RTX8000 GPUs. ECNF++ sampling is parallelized across multiple nodes with unique seeds to reduce sequential runtime. All evaluation timings are recorded using NVIDIA L40S GPUs. The sampling time required for 10^4 samples for each model is presented in Fig. 22.

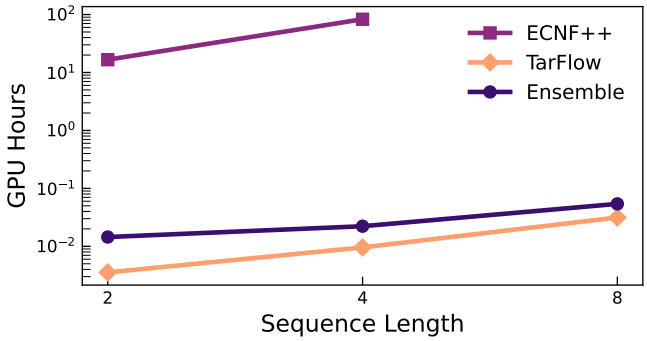

Figure 22: Sampling time for 10^4 samples on NVIDIA L40S GPU for models presented in Table 1.

E. Importance Sampling

E.1. Self-Normalized Importance Sampling (SNIS)

SNIS corresponds to the following estimator

$$\mathbb{E}_{p(x)}\varphi(x) \approx \sum_{i=1}^{n} \frac{w_i}{\sum_{j=1}^{n} w_j}\varphi(x_i)\,, \quad w_i = \frac{p(x_i)}{q(x_i)}\,, \quad x_i \sim q(x)\,, \tag{23}$$

where one uses the learned density model of ENSEMBLE $q_\theta(x)$ as $q(x)$.

E.2. Continuous-time Annealed Importance Sampling (AIS)

Below, we repeat the derivations from (Jarzynski, 1997; Albergo & Vanden-Eijnden, 2024). Namely, we consider the continuous family of marginal densities

$$q_t(x) = \frac{1}{Z_t}\exp(-U_t(x))\,, \quad Z_t = \int dx\ \exp(-U_t(x))\,. \tag{24}$$

The PDE describing the time-evolution of this density is

$$\frac{\partial q_t(x)}{\partial t} = q_t(x)\left[-\frac{\partial U_t(x)}{\partial t} + \mathbb{E}_{q_t(x)}\frac{\partial U_t(x)}{\partial t}\right] \tag{25}$$

$$= \pm\left\langle \nabla, q_t(x)\frac{\sigma_t^2}{2}\nabla \log q_t(x)\right\rangle + q_t(x)\left[-\frac{\partial U_t(x)}{\partial t} + \mathbb{E}_{q_t(x)}\frac{\partial U_t(x)}{\partial t}\right] \tag{26}$$

$$= -\left\langle \nabla, q_t(x)\frac{\sigma_t^2}{2}\nabla \log q_t(x)\right\rangle + \frac{\sigma_t^2}{2}\Delta q_t(x) + q_t(x)\left[-\frac{\partial U_t(x)}{\partial t} + \mathbb{E}_{q_t(x)}\frac{\partial U_t(x)}{\partial t}\right]. \tag{27}$$

This is a Feynman-Kac PDE which can be simulated (Del Moral, 2013) as the following SDE on the extended space of states x_t and weights w_t

$$\begin{aligned} dx_t &= -\frac{\sigma_t^2}{2}\nabla U_t(x)dt + \sigma_t dW_t\,, \quad x_{t=0} \sim q_0(x) \\ d\log w_t &= -\frac{\partial U_t(x)}{\partial t}dt\,, \quad w_{t=0} = 1\,. \end{aligned} \tag{28}$$

The expectation of the statistics $\varphi(x)$ w.r.t. the density $q_T(x)$ then can be estimated using SNIS as follows

$$\mathbb{E}_{q_T(x)}\varphi(x) \approx \sum_{i=1}^{n}\frac{w_T^i}{\sum_{j=1}^n w_T^j}\varphi(x_T^i)\,, \tag{29}$$

where (x_T^i, w_T^i) are the solutions of the SDE Eq. (28).

For the inference time of ENSEMBLE, we define the continuous family of marginals as

$$q_t(x) \propto \exp\big(\underbrace{(1-t)\log q_\theta(x) + t\log p(x)}_{-U_t(x)}\big)\,, \quad t \in [0,1]\,, \tag{30}$$

where $q_\theta(x)$ is the learned density of ENSEMBLE. Thus, Eq. (28) becomes

$$\begin{aligned} dx_t &= \frac{\sigma_t^2}{2}((1-t)\nabla\log q_\theta(x) + t\nabla\log p(x))dt + \sigma_t dW_t\,, \quad x_{t=0}\sim q_0(x) \\ d\log w_t &= (\log p(x) - \log q_\theta(x))dt\,, \quad w_{t=0} = 1\,. \end{aligned} \tag{31}$$

E.3. Discrete-time Annealed Importance Sampling (AIS)

Consider a sequence of marginal densities

$$q_0(x) \propto \exp(-U_0(x)), \ldots, q_K(x) \propto \exp(-U_K(x))\,. \tag{32}$$

Let's denote by $k_t(x_t \,|\, x_{t-1})$ the kernel that satisfies the detailed balance w.r.t. $q_t(x_t)$, i.e.

$$q_t(x_{t-1})k_t(x_t \,|\, x_{t-1}) = q_t(x_t)k_t(x_{t-1} \,|\, x_t)\,. \tag{33}$$

Then, one can write importance sampling estimator for the final marginal as

$$\int dx_K\; q_K(x_K)\varphi(x_K) = \int dx_{K-1}dx_K\; k_K(x_{K-1} \,|\, x_K)q_K(x_K)\varphi(x_K) \tag{34}$$

$$= \int dx_{K-1}dx_K\; k_K(x_K \,|\, x_{K-1})q_K(x_{K-1})\varphi(x_K) \tag{35}$$

$$= \mathbb{E}_{q_{K-1}(x_{K-1})k_K(x_K \,|\, x_{K-1})}\frac{q_K(x_{K-1})}{q_{K-1}(x_{K-1})}\varphi(x_K)\,. \tag{36}$$

Clearly, we can repeat the trick but now for $q_{K-1}(x_{K-1})$. Thus, applying this trick recursively to different marginals, we have

$$\int dx_K\; q_K(x_K)\varphi(x_K) = \mathbb{E}_{q_0(x_0)}\mathbb{E}_{x_1,\ldots,x_K}\prod_{t=0}^{K-1}\frac{q_{t+1}(x_t)}{q_t(x_t)}\varphi(x_K)\,, \tag{37}$$

$$\text{where } x_1, \ldots, x_K \sim k_1(x_1 \,|\, x_0)\ldots k_{K-1}(x_{K-1} \,|\, x_{K-2})k_K(x_K \,|\, x_{K-1})\,. \tag{38}$$

Thus, we have the following SNIS estimator

$$\int dx\; q_K(x)\varphi(x) \approx \sum_i \frac{w_K^i}{\sum_j w_K^j}\varphi(x_K^i)\,, \tag{39}$$

$$\text{where}\;\; x_t^i \sim k_t(x_t \,|\, x_{t-1})\,,\;\; t = 1, \dots, K\,,\;\; x_0 \sim q_0(x_0) \tag{40}$$

$$\log w_t^i = -U_t(x_{t-1}) + U_{t-1}(x_{t-1}) + \log w_{t-1}^i\,. \tag{41}$$

Note that there is a lot of flexibility for the choice of $k_t(x_t \,|\, x_{t-1})$ because we do not use the densities of the transition kernel for the weights. In particular, the Metropolis-Hastings algorithm with any proposal yields a reversible kernel (satisfies the detailed balance), which result in a consistent final estimator. Furthermore, compared to the continuous-time AIS, discrete-time AIS does not introduce the time-discretization error.

At the inference step of ENSEMBLE, we choose

$$q_t(x) \propto \exp\big(\underbrace{(1-t)\log q_\theta(x) + t \log p(x)}_{-U_t(x)}\big)\,,\;\; t = 0, \frac{1}{K}, \frac{2}{K}, \dots, 1\,, \tag{42}$$

and Metropolis-Adjusted Langevin Dynamics as the transition kernel $k_t(x_t \,|\, x_{t-1})$ (Roberts & Tweedie, 1996).

F. Additional Architecture Details

Adaptive Layer Norm and Transition ENSEMBLE integrates adaptive layer normalization and transition modules from (Geffner et al., 2024) into the transformer blocks of the TarFlow architecture (Zhai et al., 2024). The positions of the latent vector z_t are encoded using a sinusoidal positional encoding, which are added directly to z_t. The conditional embedding is used in the adaptive layer normalization and adaptive scale components. See Fig. 23 for details.

Since ENSEMBLE has more parameters from the adaptive layer normalization and transition components, we scale the standard TarFlow model in two ways for a fair comparison: by increasing the width and by increasing the depth (i.e., the number of transformer layers per block) to match the parameter count of ENSEMBLE. However, we observe neither of these scaled variants matches the performance of ENSEMBLE.

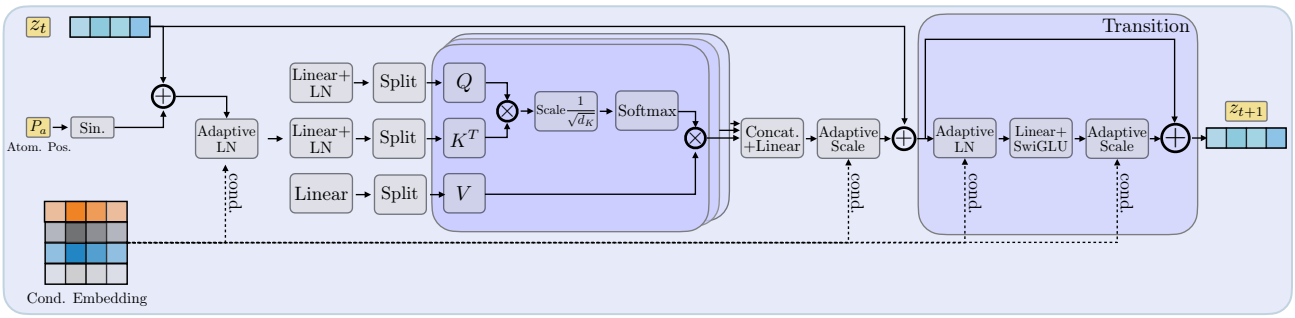

Figure 23: **Adaptive Layer Norm and Transition.** The transformer block is modified to incorporate conditional information using adaptive layer normalization and a transition block. Figure adapted from (Geffner et al., 2024).

.