Group Meeting - February 12, 2021

Paper review & Research progress

Truong Son Hy *

*Department of Computer Science
The University of Chicago

Ryerson Physical Lab

Content

- Research update
- Literature review:
  1. **Auto-Encoding Molecular Conformations** (NeurIPS 2020 workshop), `https://arxiv.org/pdf/2101.01618.pdf`
  2. **Stochastic Normalizing Flows** (NeurIPS 2020), `https://arxiv.org/pdf/2002.06707.pdf`

Research update

1. Simple implementation (no learnable parameter, single layer) Cormorant (N-Body) network in TensorFlow 2 (with Keras) using the C++ core of CG. Testing with the task of learning the total sum of Coulomnb forces between every pair of atoms. This is just a template API for building more complicated networks and then embed them into 3D conformation generation.
2. We need a better C++/CUDA core. The current speed is 6ms for a forward pass of a system of 15 atoms.
3. TorchMD: A deep learning framework for molecular simulations.
   Paper: `https://arxiv.org/pdf/2012.12106.pdf`
   Code: `https://github.com/torchmd`

Paper 1

Auto-Encoding Molecular Conformations (NeurIPS 2020 workshop)
Robin Winter, Frank Noé, Djork-Arné Clevert
`https://arxiv.org/pdf/2101.01618.pdf`

Note: This is still a workshop paper. I think they will resubmit to a mainstream conference soon.

Internal coordinate representation

Another name **Z-matrix**. A molecule spatial arrangement (conformation) Ξ is defined by:

- The set of distances $\mathcal{D} = \{d_1, .., d_{N_\mathcal{D}}\}$ between bonded atoms (bond length).
- The angles $\Phi = \{\phi_1, .., \phi_{N_\Phi}\}$ of three connected atoms (bond angles).
- The torsion angles (dihedral angles) $\Psi = \{\psi_1, .., \psi_{N_\Psi}\}$ of three consecutive bonds.

This representation is invariant to rotations and rigid translations and can always be transformed to and from Cartesian coordinates.

Proposal

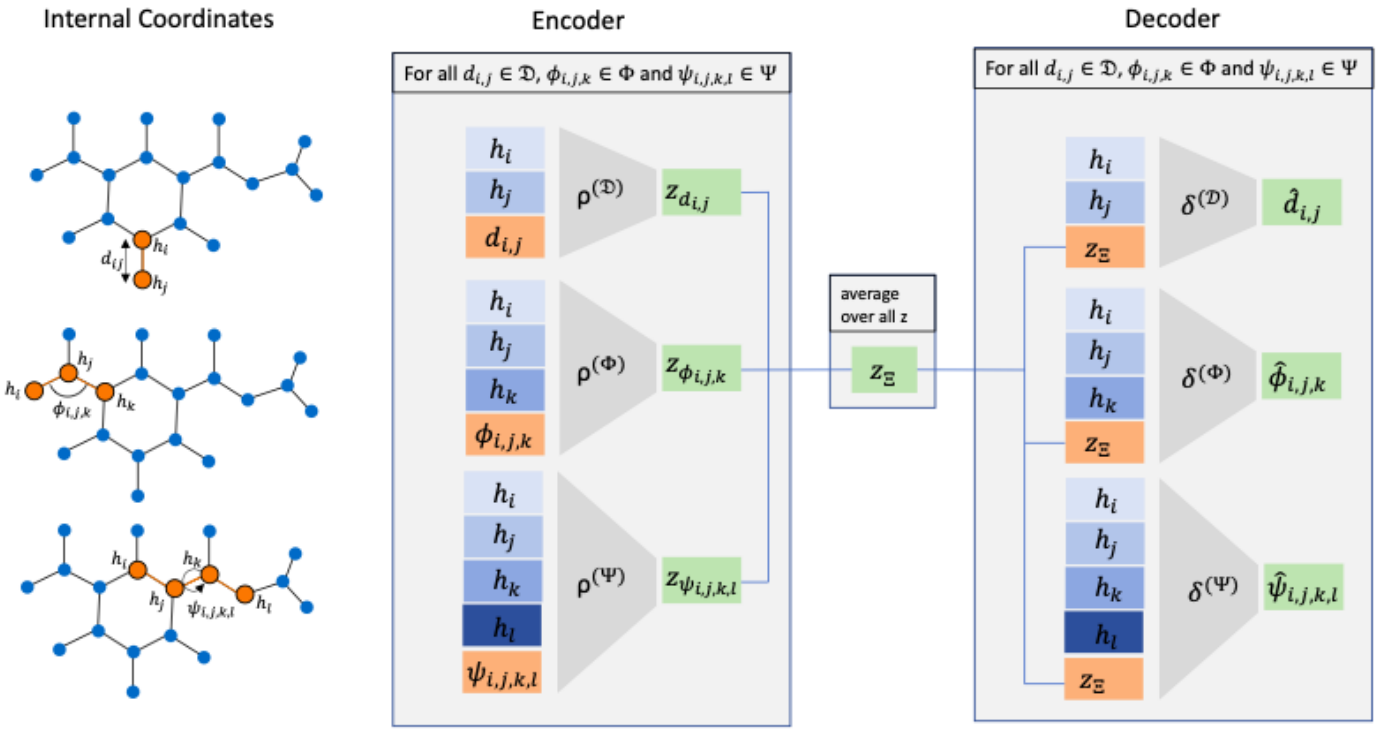

Figure 1: Model architecture of the conformation encoder (middle) and decoder (right). The encoding functions $\rho^{(\mathcal{D})}$, $\rho^{(\Phi)}$ and $\rho^{(\Psi)}$ encode their respective internal coordinates into a latent representation. The decoding functions $\delta^{(\mathcal{D})}$, $\delta^{(\Phi)}$ and $\delta^{(\Psi)}$ are conditioned on the averaged latent representations (conformer embedding) to reconstruct their respective internal coordinates, given a set of node embeddings $h_i \in \mathcal{H}$. On the left hand side, the definition of the internal coordinates, bond length $d_{i,j} \in \mathcal{D}$, bond angle $\phi_{i,j,k} \in \Phi$ and dihedral angle $\psi_{i,j,k,l} \in \Psi$, is visualized.

Conformation autoencoder (1)

Goal

Find functions f_Θ (encoder) and g_Θ (decoder) that map a conformation $\Xi_\mathcal{G}$ of a molecule $\mathcal{G}$ to and from a fixed-sized latent representation $z_\Xi \in \mathbb{R}^{F_z}$.

Minimizing the reconstruction error of the internal coordinates Ξ for a given molecule:

$$C_\Xi = \frac{1}{N_\mathcal{D}} \sum_{d \in \mathcal{D}} ||d - \hat{d}||_2^2 + \frac{1}{N_\Phi} \sum_{\phi \in \Phi} ||\phi - \hat{\phi}||_2^2$$

$$+ \frac{1}{N_\Psi} \sum_{\psi \in \Psi} \min\{||\psi - \hat{\psi}||_2^2, 2\pi - ||\psi - \hat{\psi}||_2^2\}$$

Conformation autoencoder (2)

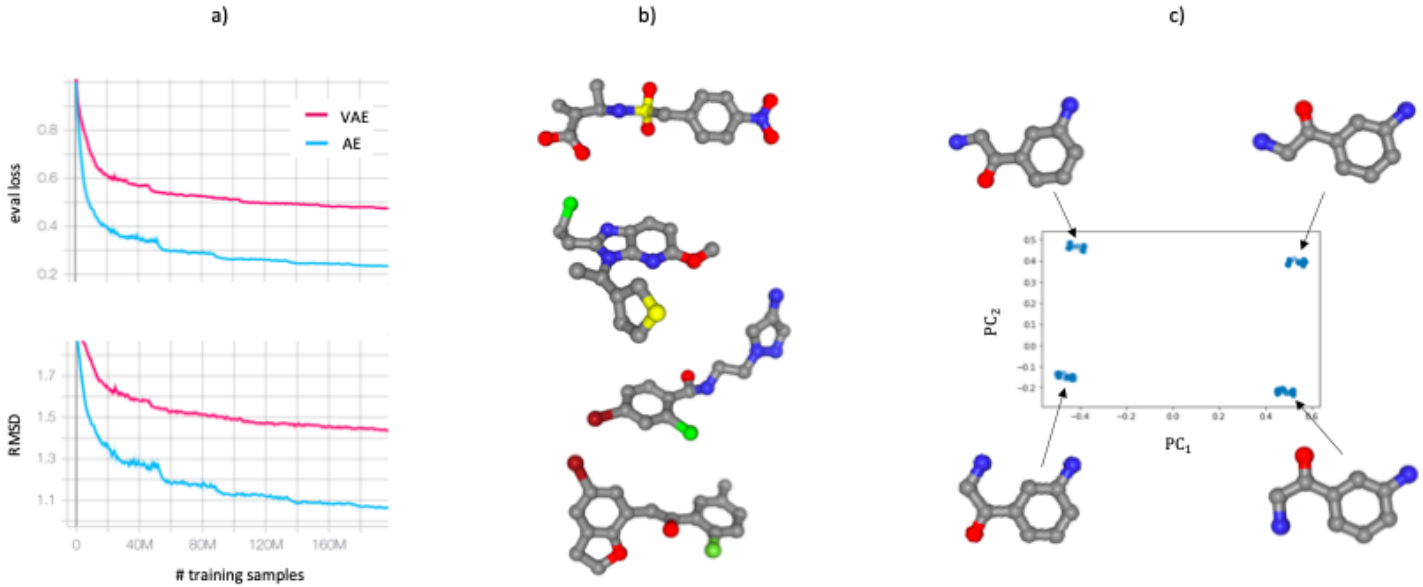

Figure 2: a) Learning curves for the autoencoder (AE) and variational autoencoder (VAE) during training, with evaluation loss as defined in (1) and root means squared deviation (RMSD) between predicted and input conformations on a holdout set. b) Four example conformations generated by our proposed model. c) First two principle components of the latent representation (conformation embedding) of 200 conformations with a corresponding representative conformation for each cluster.

Conformation autoencoder (3)

Encoder:

$$z_{\Xi} = f_{\Theta}(\mathcal{H}, \Xi) = \sigma\left(\sum_{\xi \in \Xi} \rho(\mathcal{H}, \xi)\right) = \frac{1}{N_{\Xi}} \sum_{\xi \in \Xi} \rho_{\Theta}(\mathcal{H}, \xi)$$

$$= \frac{1}{N_{\mathcal{D}} + N_{\Phi} + N_{\Psi}} \left(\sum_{d \in \mathcal{D}} \rho_{\Theta}^{(\mathcal{D})}(\mathcal{H}, d) + \sum_{\phi \in \Phi} \rho_{\Theta}^{(\Phi)}(\mathcal{H}, \phi) + \sum_{\psi \in \Psi} \rho_{\Theta}^{(\Psi)}(\mathcal{H}, \psi) \right).$$

where:

- $\mathcal{H} = \{h_1, .., h_N\}$ are the node embeddings produced by the molecular graph encoder (with attention model).
- $\rho_{\Theta}^{(\mathcal{D})}$, $\rho_{\Theta}^{(\Phi)}$, $\rho_{\Theta}^{(\Psi)}$ are feed-forward neural nets that take bond lengths, bond angles, and dihedral angles along with the graph context $\mathcal{H}$.

Decoder: additional neural nets $\delta_{\Theta}^{(\mathcal{D})}$, $\delta_{\Theta}^{(\Phi)}$, and $\delta_{\Theta}^{(\Psi)}$.

Experiment

Summary:

- **PubChem3D** dataset: organic molecules, up to 50 heavy atoms.
- Multiple conformations generated by the forcefield software **OMEGA**.
- Metrics:
  1. RMSD
  2. Internal energy with the **MMFF94 forcefield** (implemented in the Python package **RDKit**)
- Result:
  1. Median energetic difference: 80 kcal/mol.
  2. RMSD got worse by 0.07 $\AA$ comparing to ETKDG (distance geometry).
  3. The result is not mature, missing baselines.

Paper 2

Stochastic Normalizing Flows (NeurIPS 2020)
Hao Wu, Jonas Köhler, Frank Noé
https://arxiv.org/pdf/2002.06707.pdf
https://github.com/noegroup/stochastic_normalizing_flows

Note:

- I think the idea is similar to the stochastic (hierarchical) VAEs where we stack multiple VAEs on top of each other.
- The result with stochastic is better, but the paper misses the analysis and intuition of why it is better.

Background (1)

Boltzmann-type distribution

Given a known energy function $u(x)$:

$$p(x) \propto \exp(-u(x))$$

Sampling is usually done by:

- Molecular dynamics (MD)
- Markov-Chain Monte-Carlo (MCMC)

Normalizing Flows (NF)

Learn an invertible function $f_\theta : \mathbb{R}^n \to \mathbb{R}^n$ that transforms sample $z \sim p(z)$ of a simple prior density (e.g. Gaussian) into $x = f_\theta(z)$ of a complex density p_{f_θ}:

$$p_{f_\theta}(x) = p(f_\theta^{-1}(x)) \det \frac{\partial f_\theta^{-1}(x)}{\partial x}$$

Background (2)

Boltzmann-Generating Flows

Use NF to generate samples x_k from a density p_{f_θ} that approximates the Boltzmann density.

1. Training by Energy:

$$\mathcal{L}_{\mathrm{KL}} = \mathbb{E}_{z\sim p}\left[u(f_\theta(z)) - \log\left|\det\frac{\partial f_\theta(z)}{\partial z}\right|\right]$$

2. Training by Examples:

$$\mathcal{L}_{\mathrm{ML}} = \mathbb{E}_{x\sim p_{\mathrm{data}}}\left[-\log p(f_\theta^{-1}(x)) - \log\left|\det\frac{\partial f_\theta^{-1}(x)}{\partial x}\right|\right]$$

3. Final loss:

$$\mathcal{L} = (1-\lambda)\mathcal{L}_{\mathrm{KL}} + \lambda\mathcal{L}_{\mathrm{ML}}$$

Stochastic Normalizing Flows (1)

A SNF is a sequence of T stochastic and deterministic transformations. We sample $z = y_0$ from the prior μ_Z, and generate a forward path $(y_1, .., y_T)$ resulting in a proposal y_T:

- Forward path probabilities:

$$p_f(z = y_0 \to y_T = x) = \prod_{t=0}^{T-1} q_t(y_t \to y_{t+1})$$

$$y_{t+1}|y_t \sim q_t(y_t \to y_{t+1})$$

- Backward path probabilities:

$$p_b(x = y_T \to y_0 = z) = \prod_{t=0}^{T-1} \tilde{q}_t(y_{t+1} \to y_t)$$

$$y_t|y_{t+1} \sim \tilde{q}_t(y_{t+1} \to y_t)$$

Stochastic Normalizing Flows (2)

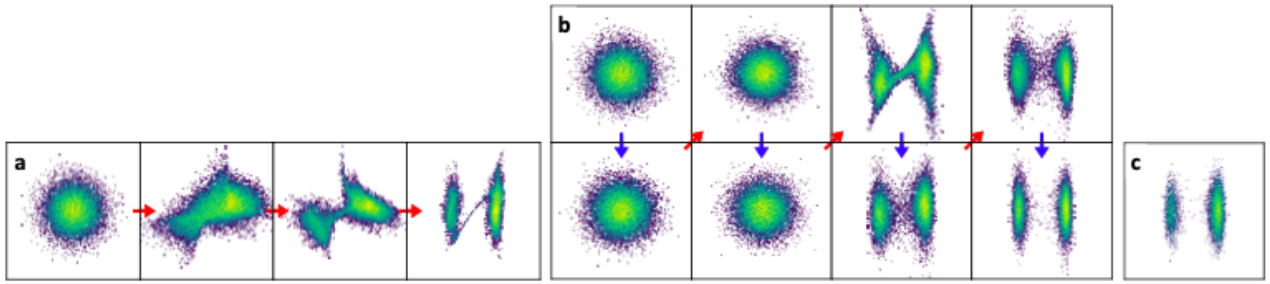

Figure 1: **Deterministic versus stochastic normalizing flow for the double well.** Red arrows indicate deterministic transformations, blue arrows indicate stochastic dynamics. **a**) 3 RealNVP blocks (2 layers each). **b**) Same with 20 BD steps before or after RealNVP blocks. **c**) Unbiased sample from true distribution.

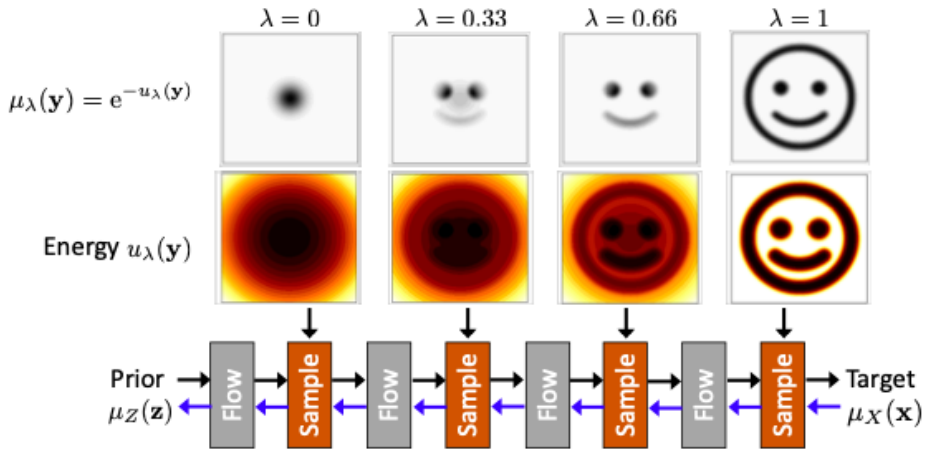

Figure 2: **Schematic for Stochastic Normalizing Flow (SNF).** An SNF transforms a tractable prior $\mu_Z(\mathbf{z}) \propto \exp(-u_0(\mathbf{z}))$ to a complicated target distribution $\mu_X(\mathbf{x}) \propto \exp(-u_1(\mathbf{x}))$ by a sequence of deterministic invertible transformations (flows, grey boxes) and stochastic dynamics (sample, ochre) that sample with respect to a guiding potential $u_\lambda(\mathbf{x})$. SNFs can be trained and run in forward mode (black) and reverse mode (blue).

Stochastic Normalizing Flows (3)

In contrast to NFs, the probability that an SNF generates a sample x:

$$p_X(x) = \int \mu_Z(y_0) p_f(y_0 \to y_T) dy_0 .. dy_{T-1}$$

is generally intractable, that involves an integral over all paths that end in x. Unnormalized importance weight proportional to the acceptance ratio to each sample path from $z = y_0$ to $x = y_T$:

$$w(z \to x) = \exp\left(- u_X(x) + u_Z(z) + \sum_t \Delta S_t(y_t) \right) \propto \frac{\mu_X(x) p_b(x \to z)}{\mu_Z(z) p_f(z \to x)}$$

where

$$\Delta S_t = \log \frac{\tilde{q}_t(y_{t+1} \to y_t)}{q_t(y_t \to y_{t+1})}$$

Stochastic Normalizing Flows (4)

The parameters of a SNF can be optimized by minimizing the Kullback - Leibler divergence between the forward and backward path probabilities, or alternatively maximizing forward and backward path weights:

$$J_{\mathrm{KL}} = \mathbb{E}_{\mu_Z(z)p_f(z\to x)}[-\log w(z \to x)]$$

$$= \mathcal{D}_{\mathrm{KL}}(\mu_Z(z)p_f(z \to x)||\mu_X(x)p_b(x \to z)) + \text{const}$$

Varitional bound:

$$\mathcal{D}_{\mathrm{KL}}(p_X(x)||\mu_X(x)) \leq \mathcal{D}_{\mathrm{KL}}(\mu_Z(z)p_f(z \to x)||\mu_X(x)p_b(x \to z))$$

Stochastic Normalizing Flows (5)

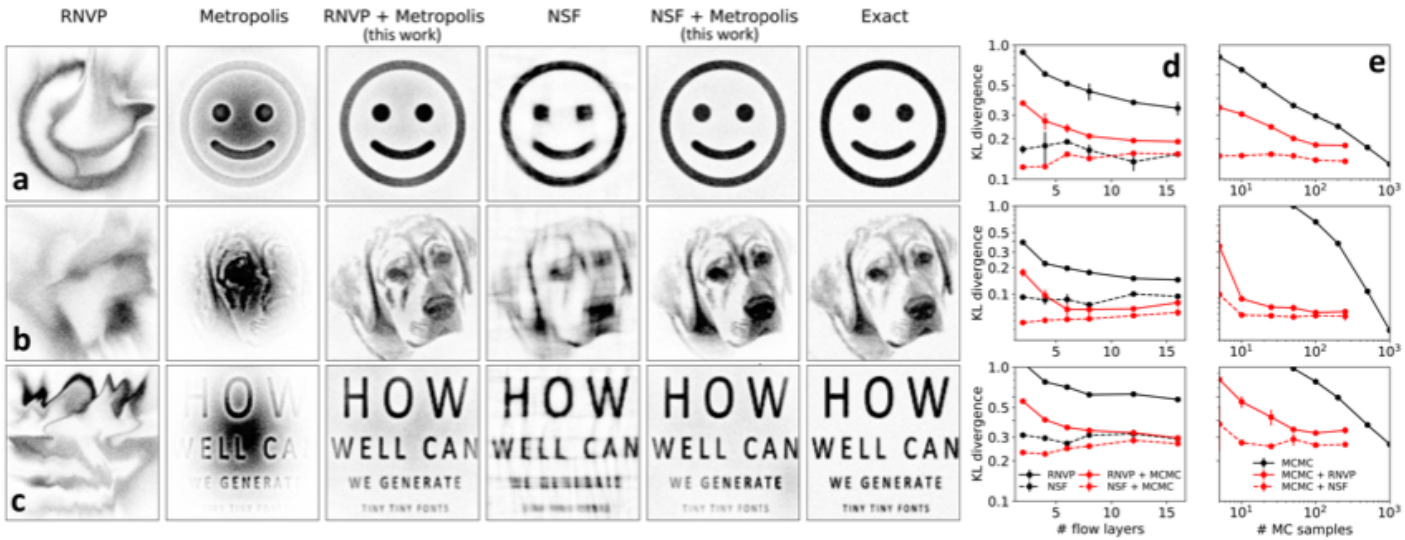

Figure 3: **Sampling of two-dimensional densities**. **a-c)** Sampling of smiley, dog and text densities with different methods. Columns: (1) Normalizing Flow with RealNVP layers, (2) Metropolis MC sampling, (3) Stochastic Normalizing Flow combining (1+2), (4) neural spline flow (NSF), (5) Stochastic Normalizing Flow combining (1+4), (6) Unbiased sample from exact density. **d-e)** Compare representative power and statistical efficiency of different flow methods by showing KL divergence (mean and standard deviation over 3 training runs) between flow samples and true density for the three images from Fig. 3. **d**) Comparison of deterministic flows (black) and SNF (red) as a function of the number of RealNVP or Neural Spline Flow transformations. Total number of MC steps in SNF is fixed to 50. **e**) Comparison of pure Metropolis MC (black) and SNF (red, solid line RealNVP, dashed line Neural spline flow) as a function of the number of MC steps. Total number of RealNVP or NSF transformations in SNF is fixed to 10.

Alanine dipeptide (1)

Evaluation of SNFs on density estimation and sampling molecular structures from a simulation of the **alanine dipeptide** molecule in vacuum:

- The molecule has 66 dimensions in $\boldsymbol{x}$ augmented with 66 auxiliary dimensions in $\boldsymbol{v}$ (velocities).
- Target density:

$$\mu_X(\boldsymbol{x}, \boldsymbol{v}) = \exp\left(- u(\boldsymbol{x}) - \frac{1}{2}||\boldsymbol{v}||^2 \right)$$

  where $u(\boldsymbol{x})$ is the **potential energy** of the molecule and $\frac{1}{2}||\boldsymbol{v}||^2$ is the **kinetic energy** term.
- Prior distribution of the latent μ_Z is an isotropic Gaussian normal distribution in all dimensions.

Alanine dipeptide (2)

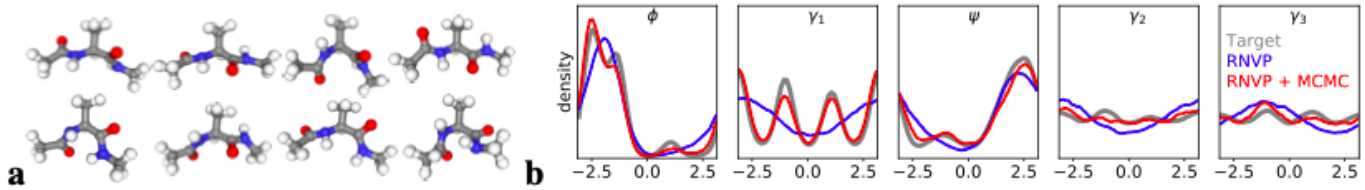

Figure 5: **Alanine dipeptide** sampled with deterministic normalizing flows and stochastic normalizing flows. **a**) One-shot SNF samples of alanine dipeptide structures. **b**) Energy (negative logarithm) of marginal densities in 5 unimodal torsion angles (top) and all 5 multimodal torsion angles (bottom).

Table 2: **Alanine dipeptide**: KL-divergences of RNVP flow and SNF (RNVP+MCMC) between generated and target distributions for all multimodal torsion angles. Mean and standard deviation from 3 independent runs.

KL-div.	ϕ	γ_1	ψ	γ_2	γ_3
RNVP	1.69±0.03	3.82±0.01	0.98±0.03	0.79±0.03	0.79±0.09
SNF	**0.36 ± 0.05**	**0.21** ±0.01	**0.27** ±0.03	**0.12** ±0.02	**0.15** ±0.04

5 multimodal torsion angles:

- Backbone angles ϕ and ψ.
- Methyl rotation angles γ_1, γ_2, and γ_3.