Fast protein backbone generation with SE(3) flow matching

Jason Yim[*] [†] [‡]
jyim@csail.mit.edu

Andrew Campbell[*][‡][§]
campbell@stats.ox.ac.uk

Andrew Y. K. Foong[‡][¶]
andrewfoong@microsoft.com

Michael Gastegger[¶]
mgastegger@microsoft.com

José Jiménez-Luna[¶]
jjimenezluna@microsoft.com

Sarah Lewis[¶]
sarahlewis@microsoft.com

Victor Garcia Satorras[¶]
victorgar@microsoft.com

Bastiaan S. Veeling[¶]
basveeling@microsoft.com

Regina Barzilay[†]
regina@csail.mit.edu

Tommi Jaakkola[†]
tommi@csail.mit.edu

Frank Noé[¶]
franknoe@microsoft.com

Abstract

We present *FrameFlow*, a method for fast protein backbone generation using SE(3) flow matching. Specifically, we adapt FrameDiff, a state-of-the-art diffusion model, to the flow-matching generative modeling paradigm. We show how flow matching can be applied on SE(3) and propose modifications during training to effectively learn the vector field. Compared to FrameDiff, FrameFlow requires five times fewer sampling timesteps while achieving two fold better designability. The ability to generate high quality protein samples at a fraction of the cost of previous methods paves the way towards more efficient generative models in *de novo* protein design.

1 Introduction

Generative models have demonstrated the potential to design novel protein structures for bespoke functions. Much of this success is due to advancements in diffusion models, which have been applied to various protein representations, ranging from carbon-alpha only [1], to torsion angles [2] and the SE(3) backbone frame representation [3]. Of these, the frame representation has been shown to achieve state-of-the-art results in *de novo* protein design tasks such as RFdiffusion [4].

However, a major drawback of diffusion models is their inference speed, with $\sim$ 1000 model forward passes often required to produce high-quality samples. This can make large-scale inference prohibitively expensive if the score model is large, as in the case of RFdiffusion. Recently, flow matching methods, which remove stochasticity from the sampling path, have emerged as an alternative

[*]Work done during an internship at Microsoft Research AI4Science.
[†]Massachusetts Institute of Technology
[‡]Corresponding authors
[§]University of Oxford
[¶]Microsoft Research AI4Science

to diffusion models, and have been generalised to Riemannian manifolds [5, 6]. The connection between flow matching and optimal transport is particularly appealing, as the linear interpolating schedule enforces straighter sampling trajectories that can be simulated with fewer integration steps [5]. These benefits have already been demonstrated in the computer vision domain, where flow matching provides results comparable to diffusion-based models at a fraction of their cost [7].

Motivated by these results, we develop flow matching in the context of protein backbone generation. We present *FrameFlow*, an adaptation of the FrameDiff [3] diffusion model to flow matching. In concurrent work, Bose et al. [8] also develop an $\mathrm{SE}(3)$ flow matching method for protein backbone generation, but don't demonstrate a speed-up during sampling compared to diffusion models. They focused on using minibatch optimal transport and stochastic differential equations to achieve higher designability. In contrast, in this work we take advantage of the flow matching framework to focus on improved performance and efficiency.

The paper is organized as follows. Section 2 describes flow matching on the $\mathrm{SE}(3)$ manifold, and introduces FrameFlow for regressing the conditional vector field. Section 3 presents our results when training FrameFlow on the SCOPe dataset [9]. By using flow matching, we obtain 2 fold better designability, comparable diversity and equal novelty scores compared to FrameDiff, while using five times fewer sampling timesteps. Compared to GENIE [10], we achieve a 23 times sampling speedup while maintaining a significantly higher designability score.

2 Method

2.1 $\mathrm{SE}(3)$ flow matching

Flow matching on Riemannian manifolds. Flow matching (FM) [5] is a simulation-free method for learning continuous normalizing flows (CNFs) [11], a class of deep generative models that generates data by integrating an ordinary differential equation (ODE) over a learned vector field. Recently, flow matching has been extended to general Riemannian manifolds [6], which we use to model the space of protein backbones. We first give a general introduction to flow matching on manifolds, before specializing to our application.

On a manifold $\mathcal{M}$, the CNF $\phi_t(\cdot) : \mathcal{M} \to \mathcal{M}$ is defined by integrating along a time-dependent vector field $v_t(x) \in \mathcal{T}_x\mathcal{M}$ where $\mathcal{T}_x\mathcal{M}$ is the tangent space of the manifold at $x \in \mathcal{M}$:

$$\frac{d}{dt}\phi_t(x) = v_t(\phi_t(x)), \quad \phi_0(x) = x. \tag{1}$$

Time is parameterized by $t \in [0, 1]$. The flow is used to transform a simple prior density p_0 towards the data distribution p_1 using the push-forward equation $p_t = [\phi_t]_* p_0$, where the density of p_t is

$$p_t(x) = [\phi_t]_* p_0(x) = p_0(\phi_t^{-1}(x))e^{-\int_0^t \mathrm{div}(v_t)(x_s)\,\mathrm{d}s}.$$

We refer to the sequence of probability distributions $\{p_t : t \in [0, 1]\}$ as the *probability path*. The vector field v_t that generates a given p_t is intractable in general but can be learned efficiently by decomposing the target probability path p_t as a mixture of tractable *conditional* probability paths, $p_t(x|x_1)$. Each conditional path satisfies $p_0(x|x_1) = p_0(x)$, and $p_1(x|x_1) \approx \delta(x - x_1)$. The desired unconditional probability path p_t can then be written as an average of the conditional probability paths with respect to the data distribution: $p_t(x) = \int p_t(x|x_1)p_1(x_1)\,\mathrm{d}x_1$.

Let $u_t(x|x_1) \in \mathcal{T}_x\mathcal{M}$ be the *conditional vector field* that generates the conditional probability path $p_t(x|x_1)$. The key insight of FM is that the unconditional vector field v_t can be learned using an objective which targets the conditional vector field $u_t(x|x_1)$:

$$\mathcal{L}_{\text{CFM}} := \mathbb{E}_{t,p_1(x_1),p_t(x|x_1)}\left[\|v_t(x) - u_t(x|x_1)\|_g^2\right],$$

where $t \sim \mathcal{U}([0, 1])$, $x_1 \sim p_1(x_1)$, $x \sim p_t(x|x_1)$ and $\|\cdot\|_g^2$ is the norm induced by the Riemannian metric g. This loss can be reparameterized by defining the conditional flow, $x_t = \psi_t(x_0|x_1)$, where ψ_t is the solution to $\frac{d}{dt}\psi_t(x) = u_t(\psi_t(x_0|x_1)|x_1)$ with initial condition $\psi_0(x_0|x_1) = x_0$. The conditional flow matching loss can then be written as:

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{t,p_1(x_1),p_0(x_0)}\left[\|v_t(x_t) - \dot{x}_t\|_g^2\right]. \tag{2}$$

Once trained, samples can be generated by simulating eq. (1) using the learned vector field v_t.

Flow matching on $\mathrm{SE}(3)$**.** We now describe the application of FM to protein backbone generation. The backbone atom positions of each residue in a protein backbone are parameterized by a rigid transformation $T \in \mathrm{SE}(3)$ (see Jumper et al. [12], Yim et al. [3]). Each frame $T = (r, x)$ consists of a rotation $r \in \mathrm{SO}(3)$ and a translation vector $x \in \mathbb{R}^3$. The protein backbone is made of N residues meaning it can be parameterized by $\mathbf{T} = [T^{(1)}, \ldots, T^{(N)}]$ with $\mathbf{T} \in \mathrm{SE}(3)^N$. Our development focuses on a single frame, but extends to all frames in a backbone since $\mathrm{SE}(3)^N$ is a product space and we choose an additive metric over the frames. For notational simplicity, we use superscripts to refer to residue indices while subscripts refer to time.

Following Yim et al. [3], we define a metric on SE(3) by choosing $\langle (a, y), (a', y') \rangle_{\mathrm{SE}(3)} = \langle a, a' \rangle_{\mathrm{SO}(3)} + \langle y, y' \rangle_{\mathbb{R}^3}$ where $\langle a, a' \rangle_{\mathrm{SO}(3)} = \mathrm{Tr}(a a'^{\mathsf{T}})/2$ and $\langle y, y' \rangle_{\mathbb{R}^3} = \sum_{i=1}^3 y_i y'_i$ are the canonical metrics on $\mathrm{SO}(3)$ and $\mathbb{R}^3$ for tangent vectors $a \in \mathfrak{so}(3)$ and $y \in \mathbb{R}^3$, respectively. This metric enables us to consider $\mathrm{SO}(3)$ and $\mathbb{R}^3$ independently when training and sampling.

Our priors are chosen as the uniform distribution on SO(3) and the unit Gaussian on $\mathbb{R}^3$, $p_0(T_0) = \mathcal{U}(SO(3)) \otimes \mathcal{N}(0, I_3)$. Following Chen and Lipman [6], the conditional flow $T_t = \psi_t(T_0 | T_1)$ is defined to be along the geodesic path connecting T_0 and T_1:

$$T_t = \exp_{T_0}\left(t \log_{T_0}(T_1)\right), \tag{3}$$

where $\exp_T$ is the exponential map and $\log_T$ is the logarithmic map at point T. Notably, distance along the geodesic varies linearly with time. With our choice of metric, eq. (3) simplifies to the following:

$$\begin{aligned} \text{Translations } (\mathbb{R}^3) : x_t &= (1-t)x_0 + t x_1 \\ \text{Rotations } (\mathrm{SO}(3)) : r_t &= \exp_{r_0}\left(t \log_{r_0}(r_1)\right). \end{aligned} \tag{4}$$

Both $\mathbb{R}^3$ and $\mathrm{SO}(3)$ are *simple manifolds* where closed form geodesics can be derived. Specifically, $\exp_{r_0}$ can be computed using Rodrigues' formula and $\log_{r_0}$ is similarly easy to compute [3]. With these considerations in mind, our overall objective can be written as:

$$\mathcal{L}_{\mathrm{SE}(3)} = \mathbb{E}_{t, p_1(\mathbf{T}_1), p_0(\mathbf{T}_0)}\left[\sum_{n=1}^{N}\left\{\left\| v_x^{(n)}(\mathbf{T}_t, t) - \dot{x}_t^{(n)} \right\|^2_{\mathbb{R}^3} + \left\| v_r^{(n)}(\mathbf{T}_t, t) - \dot{r}_t^{(n)} \right\|^2_{\mathrm{SO}(3)}\right\}\right],$$

where (n) refers to the nth residue, $t \sim \mathcal{U}([0, 1-\epsilon])$ for $\epsilon = 10^{-3}$. The vectors $\{v_x^{(n)}, v_r^{(n)}\}_{n=1}^N$ approximate the vector field as in eq. (2), and are modeled with an SE(3)-equivariant neural network (Section 2.2). Following our definitions of $x_t^{(n)}$ and $r_t^{(n)}$ we compute their time derivatives and approximate them as:

$$\dot{x}_t^{(n)} = \frac{x_1^{(n)} - x_t^{(n)}}{1-t}, \quad \dot{r}_t^{(n)} = \frac{\log_{r_t^{(n)}}(r_1^{(n)})}{1-t}, \quad v_x^{(n)} := \frac{\hat{x}_1^{(n)} - x_t^{(n)}}{1-t}, \quad v_r^{(n)} := \frac{\log_{r_t^{(n)}}(\hat{r}_1^{(n)})}{1-t}, \tag{5}$$

where $\{(\hat{x}_1^{(n)}, \hat{r}_1^{(n)})\}_{n=1}^N$ are predictions of the clean frames given the corrupted frames $\mathbf{T}_t$ at time t. Following Yim et al. [3], we reparameterize the objective as predicting the clean data:

$$\begin{aligned} \mathcal{L}_{\mathrm{SE}(3)} = \mathbb{E}_{t, p_1(\mathbf{T}_1), p_0(\mathbf{T}_0)}\Bigg[\frac{1}{(1-t)^2}\sum_{n=1}^{N}\Big\{ &\left\| \hat{x}_1^{(n)}(\mathbf{T}_t, t) - x_1^{(n)} \right\|^2_{\mathbb{R}^3} + \\ &\left\| \log_{r_t^{(n)}}\left(\hat{r}_1^{(n)}(\mathbf{T}_t, t)\right) - \log_{r_t^{(n)}}\left(r_1^{(n)}\right) \right\|^2_{\mathrm{SO}(3)} \Big\}\Bigg]. \end{aligned} \tag{6}$$

Symmetries. We perform all modelling within the zero center of mass (CoM) subspace of $\mathbb{R}^{N \times 3}$ as in [3]. This entails simply subtracting the CoM from the prior sample and all datapoints. As x_t is a linear interpolation between the noise sample and data, x_t will have 0 CoM also. This guarantees that the distribution of sampled frames that the model generates is SE(3)-invariant. To see this, note that the prior distribution is SE(3)-invariant and the vector field $\{v_x^{(n)}, v_r^{(n)}\}_{n=1}^N$ is equivariant because we use an SE(3)-equivariant architecture to predict $\{\hat{x}_1^{(n)}, \hat{r}_1^{(n)}\}_{n=1}^N$. Hence by Köhler et al. [13], the push-forward of the prior under the flow is invariant.

2.2 FrameFlow

Section 2.1 relies on learning an equivariant vector field using an equivariant neural network. In this section, we discuss the choice of network architecture and additional modifications to improve performance.

Network Architecture. To learn $\hat{T}_1^{(n)} = (\hat{r}_1^{(n)}, \hat{x}_1^{(n)})$ for every residue n, we utilize the FramePred architecture introduced in FrameDiff [3] which incorporates Invariant Point Attention (IPA) updates introduced in Jumper et al. [12] to encode spatial features and ensure its outputs are equivariant with respect to the input. Between IPA layers are transformer layers [14] used to encode sequence-level features. Unlike FrameDiff, we do not predict the psi angle to recover the oxygen atom but use the planar geometry of the backbone to impute the oxygen atoms, as done in RFdiffusion. All other hyperparameters, *e.g.* hidden dimensions, and the use of self-conditioning [15], follow FrameDiff.

Loss modifications. We weight the rotation loss terms in eq. (6) by 0.5 to be on a similar scale as the translation loss. We notice the loss explodes for $t \approx 1$ due to the $1/(1-t)^2$ term; we found it beneficial to clip this scaling to $1/(1-\min\{t, 0.9\})^2$.

SO(3) inference scheduler. Our development of $\mathrm{SO}(3)$ FM (Section 2.1) follows Chen and Lipman [6] in using a linear scheduler $\kappa(t) = 1 - t$. However, we found this schedule to perform poorly for $\mathrm{SO}(3)$ in the context of $\mathrm{SE}(3)$ FM. Instead, we utilize a exponential scheduler $\kappa(t) = e^{-ct}$ for some constant c. For high c, the rotations accelerates towards the data faster than the translations which still follow the linear schedule. The $\mathrm{SO}(3)$ flow in eq. (4) and vector field in eq. (5) become the following when re-derived,

$$\begin{aligned} r_t &= \exp_{r_0}\left(\left(1 - e^{-ct}\right)\log_{r_0}(r_1)\right) \\ v_r^{(n)} &= c\log_{r_t^{(n)}}\left(\hat{r}_1^{(n)}\right). \end{aligned} \tag{7}$$

We find $c = 10$ or 5 to work well and use $c = 10$ in our experiments. Interestingly, we found the best performance when $\kappa(t) = 1 - t$ was used for $\mathrm{SO}(3)$ during training while $\kappa(t) = e^{-ct}$ is used during inference. We found using $\kappa(t) = e^{-ct}$ during training made training too easy with little learning happening. The vector field in eq. (7) matches the vector field in FoldFlow when inference annealing is performed. However, their choice of scaling was attributed to normalizing the predicted vector field rather than the schedule.

Alternative SO(3) prior. Rather than using the $\mathcal{U}(\mathrm{SO}(3))$ prior during training, we find using the $\mathrm{IGSO3}(\sigma = 1.5)$ prior [16] used in FrameDiff to result in improved performance. The choice of $\sigma = 1.5$ will shift the r_0 samples away from π where near degenerate solutions can arise in the geodesic. During sampling, we still use the $\mathcal{U}(\mathrm{SO}(3))$ prior.

Pre-alignment. Following Klein et al. [17] and Shaul et al. [18], we pre-align samples from the prior and the data by using the Kabsch algorithm to align the noise with the data to remove any global rotation that results in a increased kinetic energy of the ODE. Specifically, for translation noise $X_0 \sim \mathcal{N}(0, I_3)^N$ and data $X_1 \sim p_1$ where $X_0, X_1 \in \mathbb{R}^{N\times 3}$ we solve $r^* = \arg\min_{r\in\mathrm{SO}(3)} \|rX_0 - X_1\|_2^2$ and use the *aligned* noise $r^* X_0$ during training. We found pre-aligment to aid in training efficiency.

3 Experiments

Training. Following GENIE [10], we evaluate FrameFlow by training it on SCOPe with proteins below length 128 for a total of 3938 examples and evaluating on the protein monomer generation task. Our model is trained for 1 day using two NVIDIA A100-48GB GPUs using the batching strategy from FrameDiff of combining proteins with the same length into the same batch to remove extraneous padding. We use the Adam [19] optimizer with learning rate 0.0001, $\beta_1 = 0.9, \beta_2 = 0.999$.

Metrics. To evaluate the model, we sample 10 backbones for every length between 60 and 128 then use ProteinMPNN [20] to design 8 sequences for each backbone. We then compute three metrics used in GENIE and FrameDiff: designability, diversity, and novelty. *Designability* is the main metric where the structure of each of the 8 sequences are predicted using ESMFold [21]. Then we compute

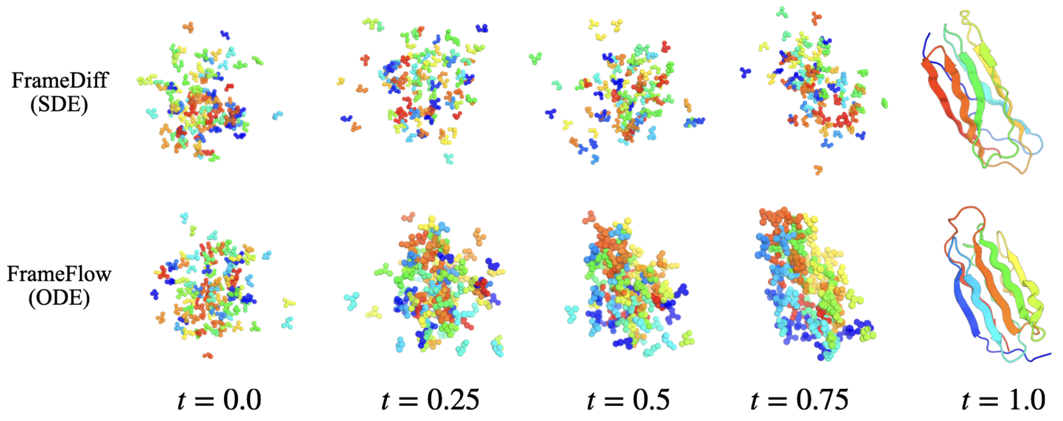

Figure 1: Sampling trajectories for FrameFlow (ODE) and FrameDiff (SDE). FrameFlow leads to much straighter integration paths, which leads to structure appearing sooner in the sampling process and allows for fewer timesteps to be used during sampling.

the minimum RMSD, referred to as `scRMSD`, between all the ESMFold predictions and the sampled backbone. A sample is deemed designable if `scRMSD` < 2.0 Å. Designability is reported as the fraction of designable samples. *Diversity* is computed by computing the number of structural clusters using MaxCluster [22] over all samples with then dividing by the total number of designable samples. We also report the total number of clusters. *Novelty* is performed by considering designable samples and using FoldSeek [23] to search for similar structures and computing the highest average TM-score [24] of samples to any chain in PDB, referred to as `pdbTM`. We report novelty as the average `pdbTM` across all samples.

Baselines. We compare our results to GENIE and FrameDiff, two diffusion models for protein backbones that do not rely on using a pre-trained folding network (unlike RFdiffusion). We use the GENIE GitHub weights trained on the same training set[6] while FrameDiff is re-trained using its default recommended settings. We expect FrameFlow to underperform RFdiffusion which we were unable to re-train on the smaller dataset. Our baselines are intended to demonstrate tradeoffs in speed and performance.

3.1 Results

We use the Euler-Maruyama integrator for SDE sampling and the Euler integrator for ODE sampling. We demonstrate the effect of different numbers of integration timesteps for all methods. Our results are shown in Table 1. We use SDE sampling for GENIE and FrameDiff since these were the methods

Table 1: Protein backbone generation results.

Model		Timesteps	Designability (↑)	Diversity (clusters) (↑)	Novelty (↓)
GENIE	SDE	1000	0.22	0.76 (131)	0.54
		750	0.11	0.79 (60)	0.51
		500	0.0	-	-
FrameDiff	SDE	500	0.42	0.36 (104)	0.66
		100	0.39	0.32 (86)	0.66
FrameDiff	ODE	100	0.26	0.43 (76)	0.65
		10	0.16	0.53 (59)	0.66
FrameFlow	ODE	500	0.81	0.22 (123)	0.69
		100	0.77	0.28 (147)	0.67
		10	0.33	0.54 (124)	0.63

[6]`https://github.com/aqlaboratory/genie/tree/main/weights/scope_l_128`

used in their respective papers. In GENIE, we find designability is low while diversity and novelty are favorable compared to FrameDiff and FrameFlow when using 1000 timesteps. The designability of GENIE even at 1000 timesteps[7] is significantly lower than that of FrameFlow and FrameDiff at 100 timesteps. We note that low designability can skew the diversity and novelty metrics since they are defined conditioned on samples being designable. However, performance in GENIE rapidly deteriorates when we reduce the number of timesteps, and is unusable at 500 timesteps, being unable to produce designable samples.

Importantly, FrameFlow when sampled with only 100 timesteps outperforms the performance of FrameDiff on designability. Using the probability ODE sampling procedure for FrameDiff also does not result in improved performance. FrameFlow's performance deterriorates rapidly with 10 timesteps which other ODE integrators could improve upon. We note that FrameFlow and FrameDiff use exactly the same architecture. This demonstrates the ability of flow matching to significantly reduce inference costs in protein backbone generation.

Diversity appears lower for FrameFlow. However, this is due to diversity *being inversely proportional to the number of designable samples*. We follow the diversity definition used in prior works, but note this metric can be artificially high by methods with low designability. Table 1 includes in parantheses the number of clusters which demonstrates FrameFlow discovering more modes than FrameDiff in the data distribution. GENIE has a high number of clusters and the best novelty indicating its high coverage despite low designability.

While GENIE has less parameters (4.1M) than FrameDiff/FrameFlow (17.4M), it uses expensive triangle updates [12] that requires high memory cost and greater compute for each forward call. Sampling a length 100 protein with 1000 timesteps on an NVIDIA V100 GPU takes GENIE 128 seconds while for FrameDiff/FrameFlow sampling with 100 timesteps takes 5.7 seconds.

Lastly, we visualize the sampling trajectory of both FrameDiff (SDE) and FrameFlow (ODE) for a length 100 protein in Figure 1. Our observations mirror the original motivations behind FM for achieving straighter and faster trajectories.

4 Discussion

In this work, we presented $\mathrm{SE}(3)$ flow matching using the previously developed theory from Chen and Lipman [6] and Yim et al. [3]. We adapted FrameDiff, an $\mathrm{SE}(3)$ diffusion model, into a flow matching model called FrameFlow and demonstrated FrameFlow's superior performance over FrameDiff and GENIE. Our experiments are preliminary demonstrations of the potential of flow matching to aid in scaling generative models for protein design as neural networks increase in size and complexity. Concurrent work, FoldFlow, did not exploit the improved speed and efficiency of flow matching, but instead utilized minibatch optimal transport [25, 7] for improved designability. We believe there is much to explore in the space of flow matching techniques to improve performance in real-world applications with protein design.

Author contributions

JY, AYKF, and FN conceived the study. JY designed and implemented FrameFlow. JY, AYKF, and AC ran experiments. JY, AYKF, and AC wrote the manuscript. MG, JJL, SL, VGS, and BSV contributed to the codebase used for experimentation and are ordered alphabetically. RB, TJ, and FN offered supervision. FN advised and oversaw the study.

References

[1] Brian L Trippe, Jason Yim, Doug Tischer, David Baker, Tamara Broderick, Regina Barzilay, and Tommi Jaakkola. Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem. *arXiv preprint arXiv:2206.04119*, 2022.

[2] Kevin Eric Wu, Kevin K Yang, Rianne van den Berg, James Zou, Alex Xijie Lu, and Ava P Amini. Protein structure generation via folding diffusion. 2022.

[7]GENIE reports 0.85 designability using the scTM > 0.5 criterion. We are able to replicate this finding, but designability in terms of scRMSD is significantly lower.

[3] Jason Yim, Brian L Trippe, Valentin De Bortoli, Emile Mathieu, Arnaud Doucet, Regina Barzilay, and Tommi Jaakkola. Se (3) diffusion model with application to protein backbone generation. *arXiv preprint arXiv:2302.02277*, 2023.

[4] Joseph L Watson, David Juergens, Nathaniel R Bennett, Brian L Trippe, Jason Yim, Helen E Eisenach, Woody Ahern, Andrew J Borst, Robert J Ragotte, Lukas F Milles, et al. De novo design of protein structure and function with rfdiffusion. *Nature*, pages 1–3, 2023.

[5] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *International Conference on Learning Representations*, 2023.

[6] Ricky TQ Chen and Yaron Lipman. Riemannian flow matching on general geometries. *arXiv preprint arXiv:2302.03660*, 2023.

[7] Aram-Alexandre Pooladian, Heli Ben-Hamu, Carles Domingo-Enrich, Brandon Amos, Yaron Lipman, and Ricky Chen. Multisample flow matching: Straightening flows with minibatch couplings. *arXiv preprint arXiv:2304.14772*, 2023.

[8] Avishek Joey Bose, Tara Akhound-Sadegh, Kilian Fatras, Guillaume Huguet, Jarrid Rector-Brooks, Cheng-Hao Liu, Andrei Cristian Nica, Maksym Korablyov, Michael Bronstein, and Alexander Tong. Se(3)-stochastic flow matching for protein backbone generation, 2023.

[9] John-Marc Chandonia, Lindsey Guan, Shiangyi Lin, Changhua Yu, Naomi K Fox, and Steven E Brenner. Scope: improvements to the structural classification of proteins–extended database to facilitate variant interpretation and machine learning. *Nucleic acids research*, 50(D1): D553–D559, 2022.

[10] Yeqing Lin and Mohammed AlQuraishi. Generating novel, designable, and diverse protein structures by equivariantly diffusing oriented residue clouds. *arXiv preprint arXiv:2301.12485*, 2023.

[11] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[12] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 2021.

[13] Jonas Köhler, Leon Klein, and Frank Noé. Equivariant flows: exact likelihood generative learning for symmetric densities. In *International conference on machine learning*, pages 5361–5370. PMLR, 2020.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[15] Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. *International Conference on Learning Representations*, 2023.

[16] Dmitry I Nikolayev and Tatjana I Savyolov. Normal distribution on the rotation group SO(3). *Textures and Microstructures*, 29, 1970.

[17] Leon Klein, Andreas Krämer, and Frank Noé. Equivariant flow matching. *arXiv preprint arXiv:2306.15030*, 2023.

[18] Neta Shaul, Ricky TQ Chen, Maximilian Nickel, Matthew Le, and Yaron Lipman. On kinetic optimal probability paths for generative models. In *International Conference on Machine Learning*, pages 30883–30907. PMLR, 2023.

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[20] J. Dauparas, I. Anishchenko, N. Bennett, H. Bai, R. J. Ragotte, L. F. Milles, B. I. M. Wicky, A. Courbet, R. J. de Haas, N. Bethel, P. J. Y. Leung, T. F. Huddy, S. Pellock, D. Tischer, F. Chan, B. Koepnick, H. Nguyen, A. Kang, B. Sankaran, A. K. Bera, N. P. King, and D. Baker. Robust deep learning-based protein sequence design using ProteinMPNN. *Science*, 378(6615):49–56, 2022.

[21] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023. doi: 10.1126/science.ade2574. URL `https://www.science.org/doi/abs/10.1126/science.ade2574`.

[22] Alex Herbert and MJE Sternberg. MaxCluster: a tool for protein structure comparison and clustering. 2008.

[23] Michel van Kempen, Stephanie Kim, Charlotte Tumescheit, Milot Mirdita, Johannes Söding, and Martin Steinegger. Foldseek: fast and accurate protein structure search. *bioRxiv*, 2022.

[24] Yang Zhang and Jeffrey Skolnick. Tm-align: a protein structure alignment algorithm based on the tm-score. *Nucleic acids research*, 33(7):2302–2309, 2005.

[25] Alexander Tong, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Kilian Fatras, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. In *ICML Workshop on New Frontiers in Learning, Control, and Dynamical Systems*, 2023.