DIFFUSION PROBABILISTIC MODELING OF PROTEIN BACKBONES IN 3D FOR THE MOTIF-SCAFFOLDING PROBLEM

Brian L. Trippe*[*] [†] btrippe@mit.edu **Jason Yim***[*][†] jyim@mit.edu **Doug Tischer** [‡] dtischer@uw.edu
David Baker[‡] dabaker@uw.edu **Tamara Broderick**[†] tbroderick@mit.edu **Regina Barzilay**[†]
regina@csail.mit.edu **Tommi Jaakkola**[†] tommi@csail.mit.edu

Introduction

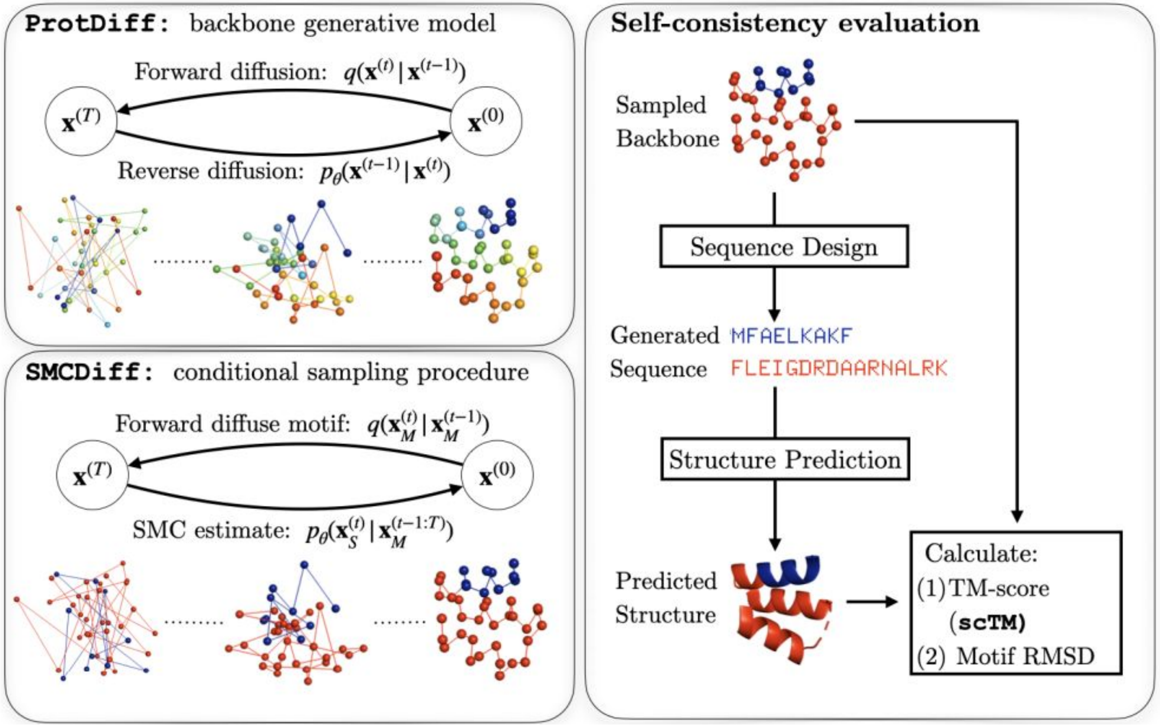
- A pipeline that generates scaffolds given motifs
- Previous work was limited, could only generate small scaffolds (up to length 20) and lacked diversity in generated scaffolds
- This approach can generate longer scaffolds (length 80) and has diversity in generated scaffolds for a given motif.

Contents

1. Motif Scaffolding
2. Backbone generative model ProtDiff
3. Motif sampler SMCDiff
4. Evaluation
5. Miscellaneous

The Motif Scaffolding Problem

- Essentially divides the protein sequence (and structure) into a Motif (M) and Scaffold (S), the motif encodes functionality and is supported on the scaffold
- **A** -> set of 20 amino acids, **s** -> sequence, the structure is a function of the sequence **x(s)**
- **M**U**S** = {1, 2, …., N} for N residues
- Goal: given a motif **M** and its structure x_M, get sequences s such that
  - $$\mathbf{x}(s)_{\mathcal{M}} \approx \mathbf{x}_{\mathcal{M}}$$


ProtDiff: backbone generative model
Forward diffusion: $q(\mathbf{x}^{(t)} | \mathbf{x}^{(t-1)})$
$\mathbf{x}^{(T)}$
$\mathbf{x}^{(0)}$
Reverse diffusion: $p_\theta(\mathbf{x}^{(t-1)} | \mathbf{x}^{(t)})$
SMCDiff: conditional sampling procedure
Forward diffuse motif: $q(\mathbf{x}_M^{(t)} | \mathbf{x}_M^{(t-1)})$
$\mathbf{x}^{(T)}$
$\mathbf{x}^{(0)}$
SMC estimate: $p_\theta(\mathbf{x}_S^{(t)} | \mathbf{x}_M^{(t-1:T)})$
Self-consistency evaluation
Sampled Backbone
Sequence Design
Generated Sequence MFAELKAKF FLEIGDRDAARNALRK
Structure Prediction
Predicted Structure
Calculate:
(1) TM-score (scTM)
(2) Motif RMSD

- **ProtDiff:** Backbone generative model, it can sample out reasonable backbone structures, uses EGNN for training.
- **SMCDiff:** Conditional sampling that uses ProtDiff to generate scaffolds (red) conditioned on the motif (blue)
- **Evaluation:** The generated structures are passed through ProteinMPNN and AlphaFold2 to generate the sequence and structure. Evaluation metrics are TMscore (for the full structure) and RMSD for the motif

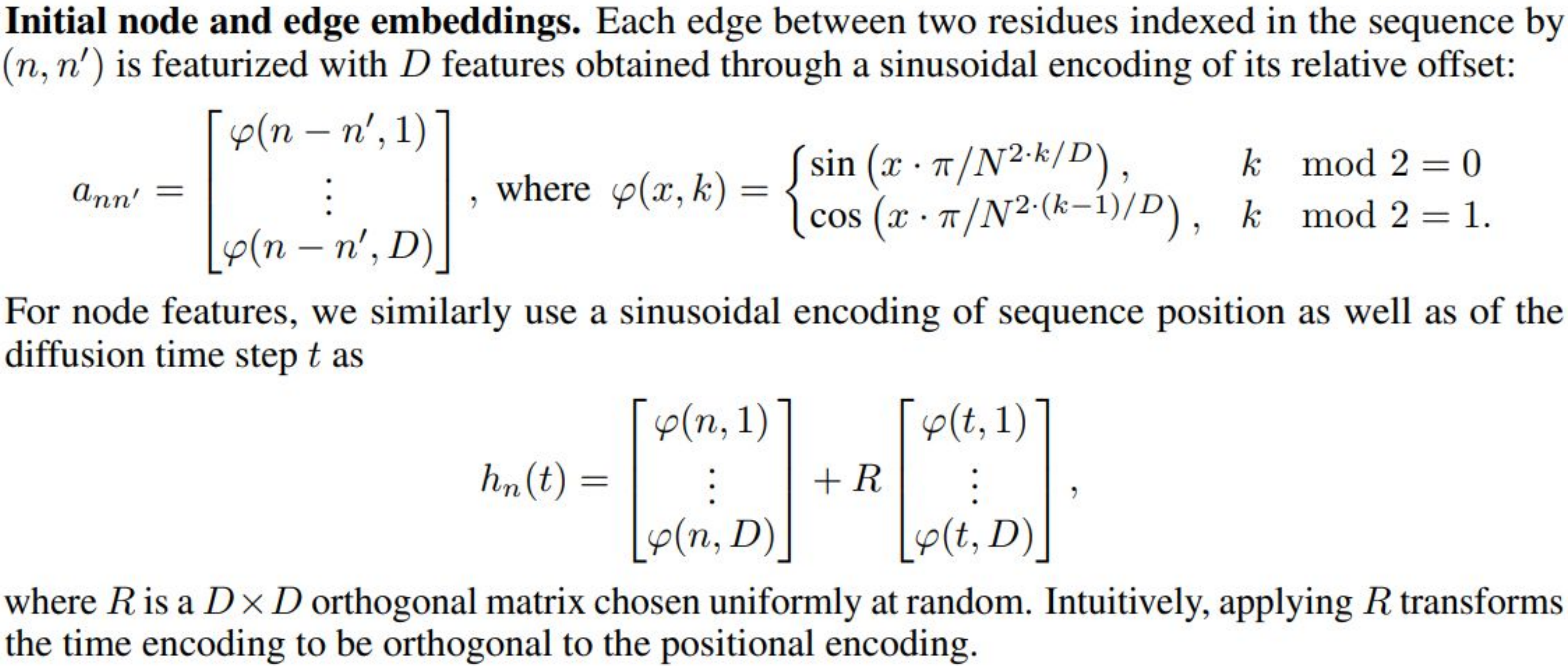
ProtDiff: structure prediction pipeline

- Uses a fully connected residue graph alongside EGNN to predict the diffused coordinates.

$$\epsilon_\theta(\mathbf{x}^{(t)}, t) = \hat{\mathbf{x}} - \mathbf{x}^{(t)}, \quad \hat{\mathbf{x}} = \mathrm{EGNN}[\mathbf{x}^{(t)}, h(t)]$$

- Alongside node and edge features, they have 3 sets of associated embeddings.
  - Sinusoidal embeddings of the relative offset between residues as edge features
  - Sinusoidal embeddings of sequence position as node feature
  - Sinusoidal embeddings of diffusion timestep as node feature (orthogonal to position embeddings)

Appendix C



Initial node and edge embeddings. Each edge between two residues indexed in the sequence by (n, n') is featurized with D features obtained through a sinusoidal encoding of its relative offset:

$$a_{nn'} = \begin{bmatrix} \varphi(n - n', 1) \\ \vdots \\ \varphi(n - n', D) \end{bmatrix}, \text{ where } \varphi(x, k) = \begin{cases} \sin\left(x \cdot \pi / N^{2 \cdot k / D}\right), & k \mod 2 = 0 \\ \cos\left(x \cdot \pi / N^{2 \cdot (k-1)/D}\right), & k \mod 2 = 1. \end{cases}$$

For node features, we similarly use a sinusoidal encoding of sequence position as well as of the diffusion time step t as

$$h_n(t) = \begin{bmatrix} \varphi(n, 1) \\ \vdots \\ \varphi(n, D) \end{bmatrix} + R \begin{bmatrix} \varphi(t, 1) \\ \vdots \\ \varphi(t, D) \end{bmatrix},$$

where R is a $D \times D$ orthogonal matrix chosen uniformly at random. Intuitively, applying R transforms the time encoding to be orthogonal to the positional encoding.

SMCDiff: Conditional sampling pipeline

- Goal: approximate the conditional distribution given by $p_\theta(\mathbf{x}_{\mathcal{S}}^{(0)} \mid \mathbf{x}_{\mathcal{M}}^{(0)})$
- This guarantees ability to sample scaffolds given the motif.
- Ideally we want to be able to get the conditional distribution over the scaffold space knowing the marginals for the motif and the full structure, but we cannot analytically solve this. This leads to poor performance.

$$p_\theta(\mathbf{x}_{\mathcal{S}}^{(0)} \mid \mathbf{x}_{M}^{(0)}) \propto p_\theta(\mathbf{x}_{\mathcal{S}}^{(0)}, \mathbf{x}_{M}^{(0)}) = p_\theta(\mathbf{x}^{(0)}) = \int p_\theta(\mathbf{x}^{(T)}) \prod_{t=0}^{T-1} p_\theta(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t+1)}) d\mathbf{x}^{(1:T)}$$

Solution

- Noise the conditioning variable slightly, and use it to generate samples for the next time step, essentially approximating the intractable integral.

$$\mathbf{x}_{\mathcal{M}}^{(1:T)} \sim q(\mathbf{x}_{\mathcal{M}}^{(1:T)} \mid \mathbf{x}_{\mathcal{M}}^{(0)}).$$

$$\mathbf{x}_{\mathcal{S}}^{(t)} \sim p_\theta(\mathbf{x}_{\mathcal{S}}^{(t)} \mid \mathbf{x}_{\mathcal{M}}^{(t+1)}, \mathbf{x}_{\mathcal{S}}^{(t+1)})$$

- This is called the **replacement method.** This is a good approximation, but it leads to an irreducible approximation error that is only dependent on the forward diffusion process (Proof and details in Appendix D)
- This error **cannot** be reduced by optimizing the reverse process.

Address replacement error: Particle filtering/Sequential Monte Carlo

- Particle filtering is used to approximate a function by spreading out particles and assigning them some predefined weight, the using some form of observed likelihood to resample for better placed particles.

```
Algorithm ParticleFilter

Input:
  U_t: Control input at time t (if applicable)
  Z_t: Observation at time t
  X_{t-1}: Set of particles at time t-1

Output:
  X_t: Updated set of particles at time t

Algorithm:
  1.  X_t = []
  2.  For each particle x^{[i]}_{t-1} in X_{t-1} do:
  3.       x'^{[i]}_t = Sample_Motion_Model(U_t, x^{[i]}_{t-1})
           // Propagate x^{[i]}_{t-1} according to the motion model
  4.       w'^{[i]}_t = Measurement_Model(Z_t, x'^{[i]}_t)
           // Weight x'^{[i]}_t according to the observation likelihood
  5.       Add x'^{[i]}_t to X_t with weight w'^{[i]}_t
  6.  End For
  7.  X_t = Resample(X_t)
      // Resample the set of particles according to the weights
  8.  Return X_t
```

Particle Filtering to look at different trajectories

- A limitation of the replacement method is it only looks at one possible noised version of the scaffold without looking beyond into the trajectory
- Uses particle filtering to look at different scaffold denoising trajectories and take the ones with higher likelihood

$$p_\theta(\mathbf{x}_{\mathcal{M}}^{(t-1)} \mid \mathbf{x}^{(t)})$$

- Takes K number of samples, diffuses their trajectories, and looks at how well the motif at time (t-1) fits in with the K different trajectories.
- Uses particle filtering to get the trajectory with highest likelihood, and uses that to sample the scaffold.

Algorithm 1 `SMCDiff`: Particle filtering for conditionally sampling from unconditional diffusion models

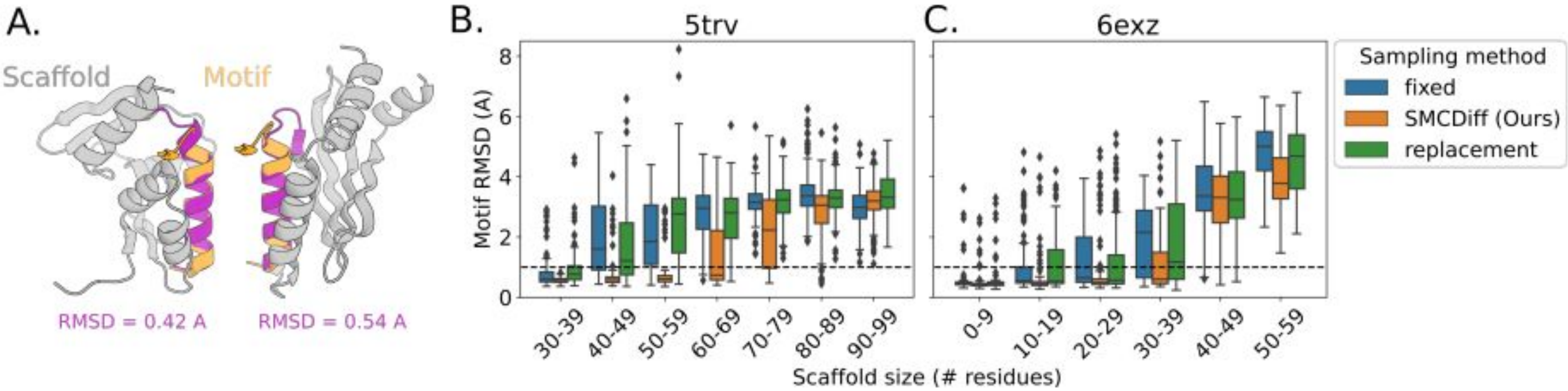
```
**Input:** $\mathbf{x}_{\mathcal{M}}^{(0)}$ (motif), $K$ (# particles)
// Forward diffuse motif
$\breve{\mathbf{x}}_{\mathcal{M}}^{(1:T)} \sim q(\mathbf{x}_{\mathcal{M}}^{(1:T)} \mid \mathbf{x}_{\mathcal{M}}^{(0)})$

// Reverse diffuse particles
$\forall k,\ \mathbf{x}_k^{(T)} \overset{i.i.d.}{\sim} p_\theta(\mathbf{x}^{(T)})$
**for** $t = T, \ldots, 1$ **do**
    // *Replace* motif
    $\forall k,\ \mathbf{x}_k^{(t)} \leftarrow [\breve{\mathbf{x}}_{\mathcal{M}}^{(t)}, \mathbf{x}_{\mathcal{S},k}^{(t)}]$

    // Re-weight based on $\breve{\mathbf{x}}_{\mathcal{M}}^{(t-1)}$
    $\forall k,\ w_k^{(t)} \leftarrow p_\theta(\breve{\mathbf{x}}_{\mathcal{M}}^{(t-1)} \mid \mathbf{x}_k^{(t)})$
    $\forall k,\ \tilde{w}_k^{(t)} \leftarrow w_k^{(t)} / \sum_{k'=1}^{K} w_{k'}^{(t)}$
    $\tilde{\mathbf{x}}_{1:K}^{(t)} \sim \mathtt{Resample}(\tilde{w}_{1:K}^{(t)}, \mathbf{x}_{1:K}^{(t)})$

    // Propose next step
    $\forall k,\ \mathbf{x}_k^{(t-1)} \overset{indep.}{\sim} p_\theta(\mathbf{x}^{(t-1)} \mid \tilde{\mathbf{x}}_k^{(t)})$
**end for**
Return $\mathbf{x}_{\mathcal{S},1:K}^{(0)}$
```

Figure 2 from the paper



Thank you! Questions?