Full-Atom Peptide Design based on Multi-modal Flow Matching

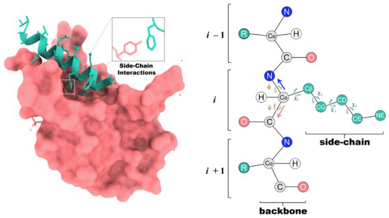
Jiahan Li*, Chaoran Cheng*, Zuofan Wu, Ruihan Guo, Shitong Luo, Zhizhou Ren, Jian Peng, Jianzhu Ma

ICML 2024

Outline

- Task: Peptide Design
- PepFlow
  - Spherical flow for orientation (Rotation)
  - Euclidean flow for position (Translation)
  - Toric flow for torsion angles
  - Simplex flow for residue type
- Framework and Algorithm
  - Training process
  - Sampling process
- Experiments
- Conclusion

Peptide Design



- Peptides are single-chains proteins comprising 3-20 residues.
- A residue can be parameterized as:
  - Type: $a_i \in \{1...20\}$
  - Frame:
    - Position vector(C_α): $x^i \in \mathbb{R}^3$
    - Rotation matrix: $R^i \in SO(3)$
  - Torsion angles: $\chi_i \in [0, 2\pi)^4$
- A protein with n residues can be parameterized as: $\{(a^i, R^i, x^i, \chi^i)\}_{i=1}^n$
- Given a n-residue peptide $C^{\text{pep}} = \{(a^j, R^j, x^j, \chi^j)\}_{j=1}^n$ and its m-residue target protein(receptor) $C^{\text{rec}} = \{(a^i, R^i, x^i, \chi^i)\}_{i=1}^m$, we aim to model the conditional joint distribution $p(C^{\text{pep}}|C^{\text{rec}})$.

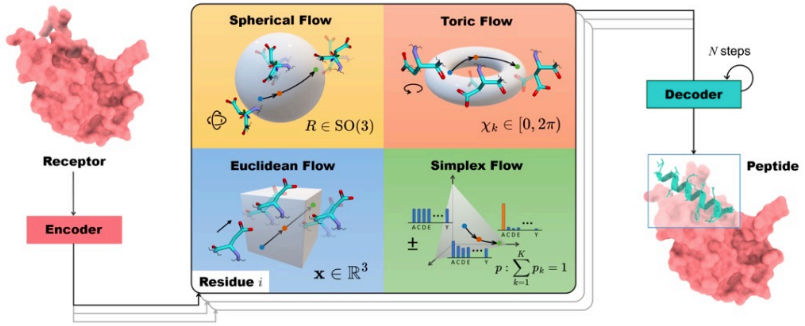
Peptide Design

- We empirically decompose the joint probability $p(C^{\text{pep}}|C^{\text{rec}})$ into the product of probabilities of four basic elements:

$$p(C^{\text{rep}}|C^{\text{rec}}) \propto p(\{\chi^j\}_{j=1}^n|C^{\text{rec}}) \cdot p(\{x^j\}_{j=1}^n|C^{\text{rec}}) \cdot p(\{R^j\}_{j=1}^n|C^{\text{rec}}) \cdot p(\{a^j\}_{j=1}^n|C^{\text{rec}}). \quad (1)$$

- We use different probability flows :
  - Orientation $R^j \longrightarrow$ Spherical Flow
  - Position $x^j \longrightarrow$ Euclidean flow
  - Torsion angles $\chi^j \longrightarrow$ Toric flow
  - Type $a^j \longrightarrow$ Simplex flow

PepFlow Architecture



- ► The multi-modal flow matching decoder finally recovers the full-atom peptide structure and sequence iteratively using the Euler method.

Conditional Flow Matching (CFM) objective

- The conditional flow matching framework provides a simple way to learn a probability flow ψ that pushes the source distribution p_0 to the target distribution p_1 of the data points $\boldsymbol{x} \in \mathbb{R}^3$.
  - The time-dependent flow: $\psi_t : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$
  - The associated vector field: $u_t : (0,1) \times \mathbb{R}^d \to \mathbb{R}^d$
  - The ODE: $\frac{d}{dt}\boldsymbol{x}_t = u_t(\boldsymbol{x}_t)$, where $\boldsymbol{x}_t = \psi_t(\boldsymbol{x}_0)$.
- When conditioning the time-dependent vector field and probability flow on the specific sample $\boldsymbol{x}_1 \sim p_1(\boldsymbol{x}_1)$ and the prior sample $\boldsymbol{x}_0 \sim p_0(\boldsymbol{x}_0)$. We can model $\boldsymbol{x}_t = \psi_t(\boldsymbol{x}_0|\boldsymbol{x}_1)$ and $u_t(\boldsymbol{x}_t|\boldsymbol{x}_0, \boldsymbol{x}_1) = \frac{d}{dt}X_t$ with Conditional Flow Matching(CFM objective):

$$\mathcal{L}(\theta) = \mathbb{E}_{t,p_1(x_1),p_0(x_0)} \left\| v_\theta(\boldsymbol{x}_t, t) - u_t(\boldsymbol{x}_t|\boldsymbol{x}_1, \boldsymbol{x}_0) \right\|^2 \quad (1) \tag{2}$$

where $t \sim \mathcal{U}(0,1)$.

Euclidean Flow for Position

- The conditional flow ψ^{pos} and vector field u_t^{pos} are established by the linear interpolation between $\boldsymbol{x}_0^j \sim \mathcal{N}(0, I_3)$ and $\boldsymbol{x}_1^j \in p(\boldsymbol{x}^j \mid C^{\text{rec}})$:

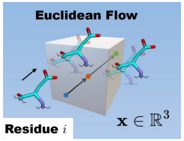
$$\psi_t^{pos}(\boldsymbol{x}_0^j|\boldsymbol{x}_1^j) = t\boldsymbol{x}_1^j + (1-t)\boldsymbol{x}_0^j \tag{3}$$

- Then we get the vector field by taking the derivative of the flow with respect to t.

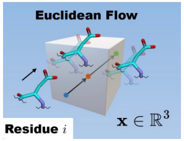
$$u_t^{pos}(\boldsymbol{x}_t^j|\boldsymbol{x}_1^j, \boldsymbol{x}_0^j) = \boldsymbol{x}_t^j - \boldsymbol{x}_0^j = \frac{\boldsymbol{x}_1^j - \boldsymbol{x}_t^j}{1-t} \tag{4}$$

- A translation-invariant neural network v_{pos} is applied to predict the vector field. The CFM objective of the j th residue is formulated as:

$$\mathcal{L}_j^{pos} = \mathbb{E}_{t, p(\boldsymbol{x}_1^j), p(\boldsymbol{x}_0^j)} \left\| v^{pos}(\boldsymbol{x}_t^j, t, C^{rec}) - (\boldsymbol{x}_1^j - \boldsymbol{x}_0^j) \right\|^2 \tag{5}$$



Euclidean Flow for Position



- The conditional flow ψ^{pos} and vector field u_t^{pos} are established by the linear interpolation between $x_0^j \sim \mathcal{N}(0, I_3)$ and $\boldsymbol{x}_1^j \in p(\boldsymbol{x}^j \mid C^{\text{rec}})$:

$$\psi_t^{pos}(\boldsymbol{x}_0^j | \boldsymbol{x}_1^j) = t\boldsymbol{x}_1^j + (1-t)\boldsymbol{x}_0^j \tag{6}$$

- Then we get the vector field by taking the derivative of the flow with respect to t.

$$u_t^{pos}(\boldsymbol{x}_t^j | \boldsymbol{x}_1^j, \boldsymbol{x}_0^j) = \boldsymbol{x}_t^j - \boldsymbol{x}_0^j = \frac{\boldsymbol{x}_1^j - \boldsymbol{x}_t^j}{1-t} \tag{7}$$

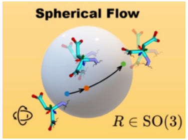
- During generation, we first sample from the prior $x_0^j \sim \mathcal{N}(0, I_3)$ and solve the probability flow with the learned predictor v^{pos} using the N-step forward Euler method to get the position of residue j with $t = \{0, \ldots, \frac{N-1}{N}\}$:

$$\boldsymbol{x}_{t+\frac{1}{N}}^j = \boldsymbol{x}_t^j + \frac{1}{N} v^{pos}(\boldsymbol{x}_t^j, t, C^{rec}) \tag{8}$$

Spherical Flow for Orientation

- The conditional flow ψ^{ori} and vector field u_t^{ori} are established by the geodesic interpolation between $R_0^j \sim U(SO(3))$ and $R_1^j \in p(R^j \mid C^{\text{rec}})$ with the geodesic distance decreasing linearly by time:

$$\psi_t^{\text{ori}}(R_0^j \mid R_1^j) = \exp_{R_0^j}(t \log_{R_0^j}(R_1^j)) \tag{9}$$



$$u_t^{\text{ori}}(R_t^j \mid R_0^j, R_1^j) = \frac{\log_{R_t^j} R_1^j}{1-t} \tag{10}$$

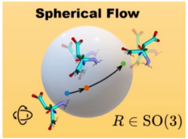
- A rotation-equivariant neural network v^{ori} is applied to predict the vector field. The CFM objective on $SO(3)$ is formulated as:

$$\mathcal{L}_{\text{ori}}^j = \mathbb{E}_{t, p\left(R_1^j\right), p\left(R_0^j\right)} \left\| v^{\text{ori}}\left(R_t^j, t, C_{\text{rec}}\right) - \frac{\log_{R_t^j} R_1^j}{1-t} \right\|_{\mathrm{SO}(3)}^2 \tag{11}$$

Spherical Flow for Orientation

- The conditional flow ψ^{ori} and vector field u_t^{ori} are established by the geodesic interpolation between $R_0^j \sim U(SO(3))$ and $R_1^j \in p(R^j \mid C^{rec})$ with the geodesic distance decreasing linearly by time:

$$\psi_t^{\mathrm{ori}}(R_0^j \mid R_1^j) = \exp_{R_0^j}(t \log_{R_0^j}(R_1^j)) \tag{12}$$



$$u_t^{\mathrm{ori}}(R_t^j \mid R_0^j, R_1^j) = \frac{\log_{R_t^j} R_1^j}{1-t} \tag{13}$$

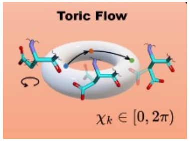
- During inference, we initiate the process from $R_0^j \sim U(SO(3))$ and proceed by taking small steps along the geodesic path in $SO(3)$ over the time step t by Euler steps:

$$R^j_{t+\frac{1}{N}} = \exp_{R_t^j}(\frac{1}{N} v^{ori}(R_t^j, t, C^{rec})) \tag{14}$$

Toric Flow for Angles

- The conditional flow ψ^{ang} and vector field u_t^{ang} are constructed by $\chi_0^j \sim U([0, 2\pi]^5)$ and $\chi_1^j \in p(\chi^j \mid C^{\text{rec}})$ along the geodesic:

$$\psi_t^{ang}(\chi_0^j|\chi_1^j) = (\chi_0^j + t(\chi_1^j - \chi_0^j))\%(2\pi) \tag{15}$$



$$u_t^{ang}(\chi_t^j|\chi_0^j, \chi_1^j) = wrap\left(\frac{\chi_1^j - \chi_t^j}{1-t}\right) \tag{16}$$

$$wrap(u) = (u + \pi)\%(2\pi) - \pi \tag{17}$$

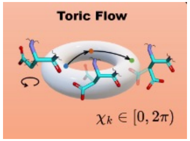
- A neural network v_{ang} is applied to predict the vector field. The CFM objective on $SO(3)$ is formulated as:

$$\mathcal{L}_{ang}^j = \mathbb{E}_{t,p(\chi_1^j),p(\chi_0^j)} \left\| wrap\left(v^{ang}(\chi_t^j, t, C^{rec}) - \frac{\chi_1^j - \chi_0^j}{1-t}\right)\right\|^2 \tag{18}$$

Toric Flow for Angles

- The conditional flow ψ^{ang} and vector field u_t^{ang} are constructed by $\chi_0^j \sim U([0, 2\pi]^5)$ and $\chi_1^j \in p(\chi^j \mid C^{\text{rec}})$ along the geodesic:

$$\psi_t^{ang}(\chi_0^j|\chi_1^j) = (\chi_0^j + t(\chi_1^j - \chi_0^j))\%(2\pi) \tag{19}$$



$$u_t^{ang}(\chi_t^j|\chi_0^j, \chi_1^j) = wrap\left(\frac{\chi_1^j - \chi_t^j}{1 - t}\right) \tag{20}$$

$$wrap(u) = (u + \pi)\%(2\pi) - \pi \tag{21}$$

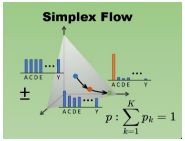
- During inference, we initiate the process from $\chi_0^j \sim U([0, 2\pi]^5)$ and proceed by taking small steps along the geodesic path in $\mathbb{T}^5$ over the time step t:

$$\chi_{t+\frac{1}{N}}^j = \left(\chi_t^j + \frac{1}{N}v^{ang}(\chi_t^j, t, C^{rec})\right)\%2\pi \tag{22}$$

Simplex Flow for Type

- We choose our prior distribution of $\chi_0^j \sim \mathcal{N}(0, K^2 I)$ such that the prior distribution on simplex becomes the logistic-normal distribution by construct.
- The conditional flow ψ^{type} and vector field u_t^{type} are defined as:

$$\psi_t^{\text{type}}(s_0^j|s_1^j) = ts_1^j + (1-t)s_0^j \tag{23}$$



$$u_t^{\text{type}}(s_t^j|s_1^j, s_0^j) = s_t^j - s_0^j = \frac{s_1^j - s_t^j}{1-t} \tag{24}$$

where $\text{logit}(a^j) = s^j \in \mathbb{R}^{20}$, and $s^j[i] = \begin{cases} K, & \text{if } i = a^j \\ -K, & \text{otherwise} \end{cases}$

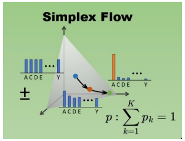
- A neural network v_{type} is applied to predict the vector field. The CFM objective of the j th residue is formulated as:

$$\mathcal{L}_j^{\text{type}} = \mathbb{E}_{t,p(s_1^j),p(s_0^j)} \left\| v^{\text{type}}(s_t^j, t, C^{\text{rec}}) - (s_1^j - s_0^j) \right\|^2 \tag{25}$$

Simplex Flow for Type

- We choose our prior distribution of $\chi_0^j \sim \mathcal{N}(0, K^2 I)$ such that the prior distribution on simplex becomes the logistic-normal distribution by construct.
- The conditional flow ψ^{type} and vector field u_t^{type} are defined as:

$$\psi_t^{\text{type}}(s_0^j | s_1^j) = t s_1^j + (1-t) s_0^j \tag{26}$$



$$u_t^{\text{type}}(s_t^j | s_1^j, s_0^j) = s_t^j - s_0^j = \frac{s_1^j - s_t^j}{1-t} \tag{27}$$

- During inference, we perform Euler steps to solve the probability flow on the logit space, residue types can be sampled from the corresponding probability vector on the simplex:

$$s_{t+\frac{1}{N}}^j = s_t^j + \frac{1}{N} v^{\text{type}}(s_t^j, t, C^{\text{rec}}) \tag{28}$$

$$a_{t+1/N}^j \sim \text{softmax}(s_{t+\frac{1}{N}}^j) \tag{29}$$

Main loss function

- Combining all modalities, we obtain the final multi-modal conditional flow matching objective for residue j as the weighted sum of different conditional flow matching objectives:

$$\mathcal{L}^j_{\text{cfm}} = \mathbb{E}_t\left(\lambda^j_{\text{pos}}\mathcal{L}^j_{\text{pos}} + \lambda^j_{\text{ori}}\mathcal{L}^j_{\text{ori}} + \lambda^j_{\text{ang}}\mathcal{L}^j_{\text{ang}} + \lambda^j_{\text{type}}\mathcal{L}^j_{\text{type}}\right) \tag{30}$$

- Considering auxiliary reconstruction losses, The overall training objective is:

$$\mathcal{L} = \mathbb{E}_t\left[\frac{1}{n}\sum_j\left(\mathcal{L}^j_{cfm} + \lambda^{aux}\left(\mathcal{L}^j_{bb} + \mathcal{L}^j_{tor}\right)\right)\right] \tag{31}$$

where the reconstruction losses about coordinates and torsion angle are: [1] [2]

$$\mathcal{L}^j_{bb} = \sum_{k_{bb}}\left\|(\hat{R}^j_1\boldsymbol{x}_{0,k_{bb}} + \hat{\boldsymbol{x}}^j_1) - \boldsymbol{x}_{1,k_{bb}}\right\|^2 \tag{32}$$

$$\mathcal{L}^j_{tor} = \sum_k\left\|(\hat{\chi}^j_{k,1} - \chi^j_{k,1})\%(2\pi)\right\|^2 \tag{33}$$

[1] $\hat{R}^j_1$ and $\hat{\boldsymbol{x}}^j_1$ are predictions from the model and $\boldsymbol{x}_{0,k_{bb}}$ and $\boldsymbol{x}_{1,k_{bb}}$ are the initial backbone atoms coordinates and the ground truth backbone atom coordinates.

[2] where $\hat{\chi}^j_{k,1}$ is the prediction from the model and we $\chi^j_{k,1}$ is the ground truth torsion angle

Algorithm

Training process:

Algorithm 1 Training Multi-Modal PepFlow

```
1: while not converged do
2:     Sample protein-peptide pair $C^{\text{rec}}, C^{\text{pep}}$ from dataset
3:     Encode target $\mathbf{h}, \mathbf{z} = \text{Enc}(C^{\text{rec}})$
4:     Sample prior state $C_0^{\text{pep}} = \{(a_0^j, R_0^j, \mathbf{x}_0^j, \chi_0^j)\}_{j=1}^n$
5:     Sample $t \sim U(0,1)$
6:     Decode predicted peptide $\overline{C}^{\text{pep}} = \text{Dec}(C_t^{\text{pep}}, t, \mathbf{h}, \mathbf{z})$
7:     Calculate the vector fields and the loss according to Eq.(23)
8:     Update the parameters of Enc and Dec
9: end while
```

Inference process:

Algorithm 2 Sampling with Multi-Modal PepFlow

```
1: Encode target $\mathbf{h}, \mathbf{z} = \text{Enc}(C^{\text{rec}})$
2: Sample prior state $C_0^{\text{pep}} = \{(a_0^j, R_0^j, \mathbf{x}_0^j, \chi_0^j)\}_{j=1}^n$
3: for $t \leftarrow 1$ to $N$ do
4:     Decode predicted peptide $\overline{C}_{\frac{t}{N}}^{\text{pep}} = \text{Dec}(C_{\frac{t-1}{N}}^{\text{pep}}, t, \mathbf{h}, \mathbf{z})$
5:     Calculate vector fields and update the peptide $C_{\frac{t}{N}}^{\text{pep}} = \text{EulerStep}(\overline{C}_{\frac{t}{N}}^{\text{pep}}, C_{\frac{t-1}{N}}^{\text{pep}}, \frac{1}{N})$
6: end for
7: return $\overline{C}_1^{\text{pep}}$
```

Experiment: Sequence-Structure Co-design

- This task requires the generation of both the sequence and bound-state structure of the peptide based on its target protein.
- We generate 64 peptides for each target protein for every evaluated model.[3] [4]

Table 1. Evaluation of methods in the sequence-structure co-design task.

	Geometry				Energy		Design	
	AAR % ↑	RMSD Å ↓	SSR % ↑	BSR % ↑	Stability % ↑	Affinity % ↑	Designability % ↑	Diversity ↑
RFdiffusion	40.14	4.17	63.86	26.71	**26.82**	16.53	**78.52**	0.38
ProteinGenerator	45.82	4.35	29.15	24.62	23.48	13.47	71.82	0.54
Diffusion	47.04	3.28	74.89	49.83	15.34	17.13	48.54	0.57
PepFlow w/Bb	50.46	2.30	82.17	82.17	14.04	18.10	50.03	**0.64**
PepFlow w/Bb+Seq	**53.25**	2.21	**85.22**	85.19	19.20	19.39	56.04	0.50
PepFlow w/Bb+Seq+Ang	51.25	**2.07**	83.46	**86.89**	18.15	**21.37**	65.22	0.42

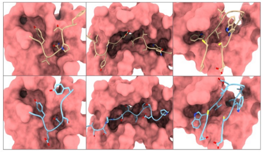


Figure 4. Three examples of the generated peptides. **Top**: native peptides; **Bottom**: generated peptides. PDB: 3MXY, 6OX4, 5DJY.

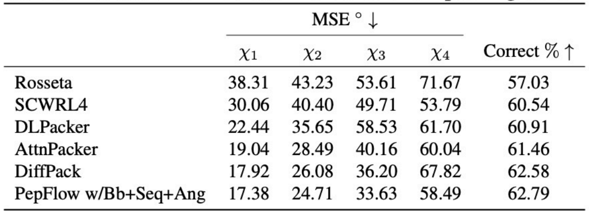
[3] The secondary-structure similarity ratio (SSR) calculates the proportion of shared secondary structures

[4] The binding site ratio (BSR) is the overlapping ratio between the bind- ing site of the generated peptide and the native binding site on the target protein.

Experiment: Side-chain Packing

- This task predicts the side-chain angles of the peptide.
- We generate 64 side-chain conformations for each peptide by each model to recover side-chain angles.[5]

Table 3. Evaluation of methods in the side-chain packing task.



	MSE ° ↓				
	χ_1	χ_2	χ_3	χ_4	Correct % ↑
Rosseta	38.31	43.23	53.61	71.67	57.03
SCWRL4	30.06	40.40	49.71	53.79	60.54
DLPacker	22.44	35.65	58.53	61.70	60.91
AttnPacker	19.04	28.49	40.16	60.04	61.46
DiffPack	17.92	26.08	36.20	67.82	62.58
PepFlow w/Bb+Seq+Ang	17.38	24.71	33.63	58.49	62.79

[5] We also include the proportion of the Correct presdictions that deviate within 20◦ around the ground truth.

Conclusion

- We introduced PepFlow, a novel flow-based generative model tailored for target-specific full-atom peptide design.
- PepFlow characterizes each modality of peptide residues into the corresponding manifold and constructs flows and vector fields for each modality.
- PepFlow has demonstrated promising performance by modeling full-atom joint distributions and exhibits potential applications in protein design beyond peptides, such as antibody and enzyme design