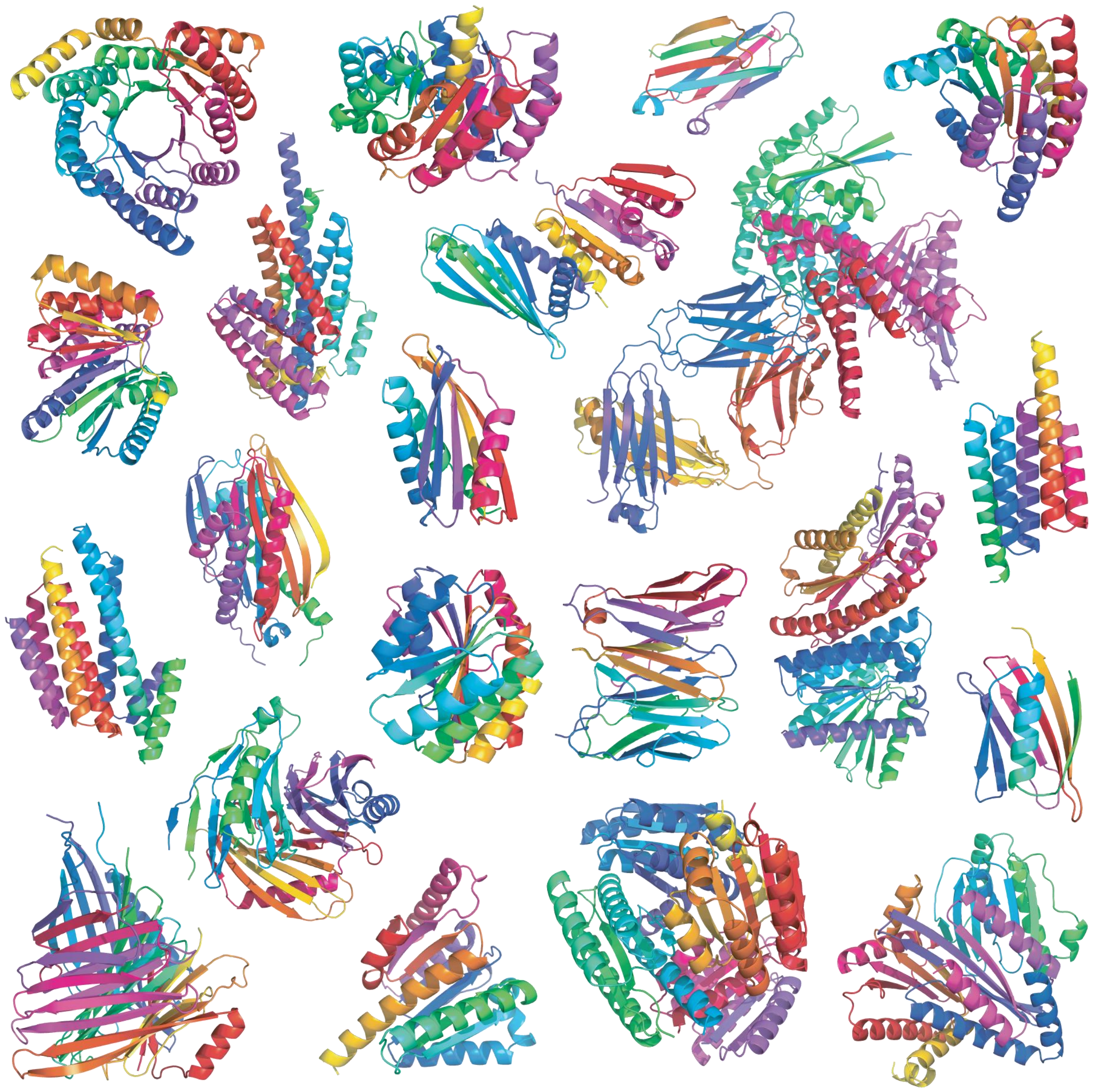


Chroma[1]

Chen, Tianhao

2024/4/23

Illuminating protein space with a programmable generative model

John B. Ingraham, Max Baranov, Zak Costello, Karl W. Barber, Wujie Wang, Ahmed Ismail, Vincent Frappier, Dana M. Lord, Christopher Ng-Thow-Hing, Erik R. Van Vlack, Shan Tie, Vincent Xue, Sarah C. Cowles, Alan Leung, João V. Rodrigues, Claudio L. Morales-Perez, Alex M. Ayoub, Robin Green, Katherine Puentes, Frank Oplinger, Nishant V. Panwar, Fritz Obermeyer, Adam R. Root, Andrew L. Beam, ... Gevorg Grigoryan

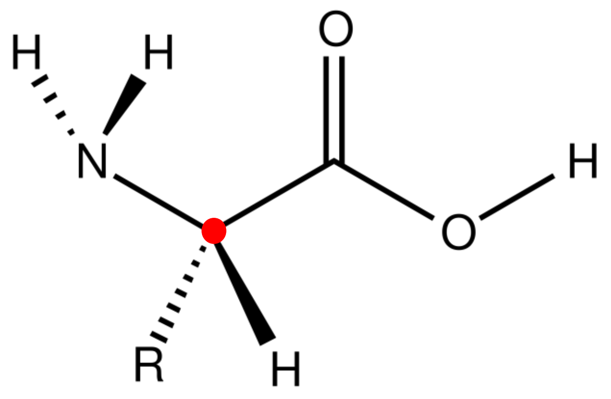
+ Show authors

Nature **623**, 1070–1078 (2023) | Cite this article

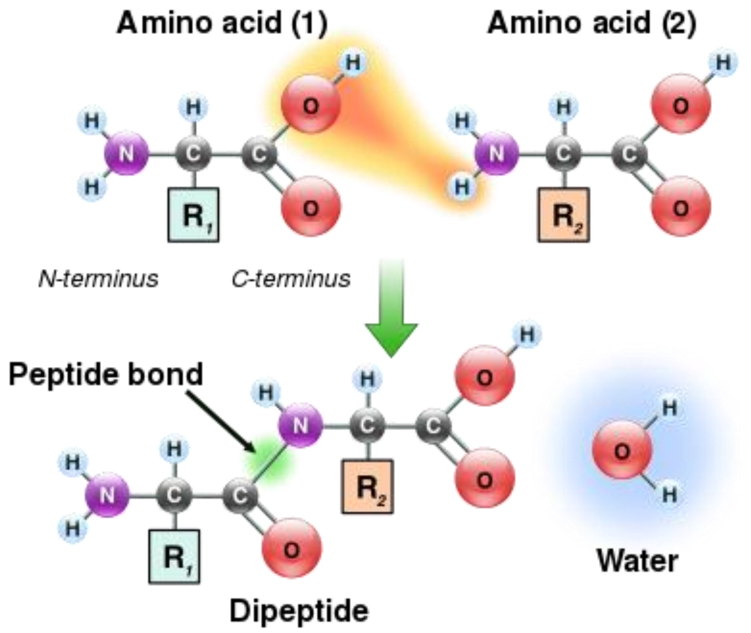
78k Accesses | **18** Citations | **297** Altmetric | Metrics

Contents

Amino acids and proteins



Amino acid:
- Ca: alpha carbon
- NH2: amino group
- R: sidechain
- COOH: carboxyl group



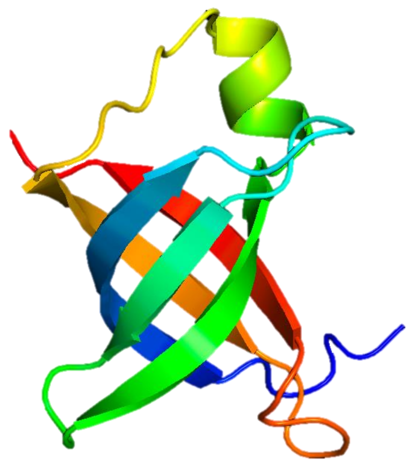
Formation of peptide bond

Peptide bond joins amino acids sequentially to form a protein sequence

Protein Modalities

MAKEDTLEFPGVVKELLPNAT
FRVELDNGHELIAVMAGKMRK
NRIRVLAGDKVQVEMTPYDLS
KGRINYRFK

Amino acid sequence



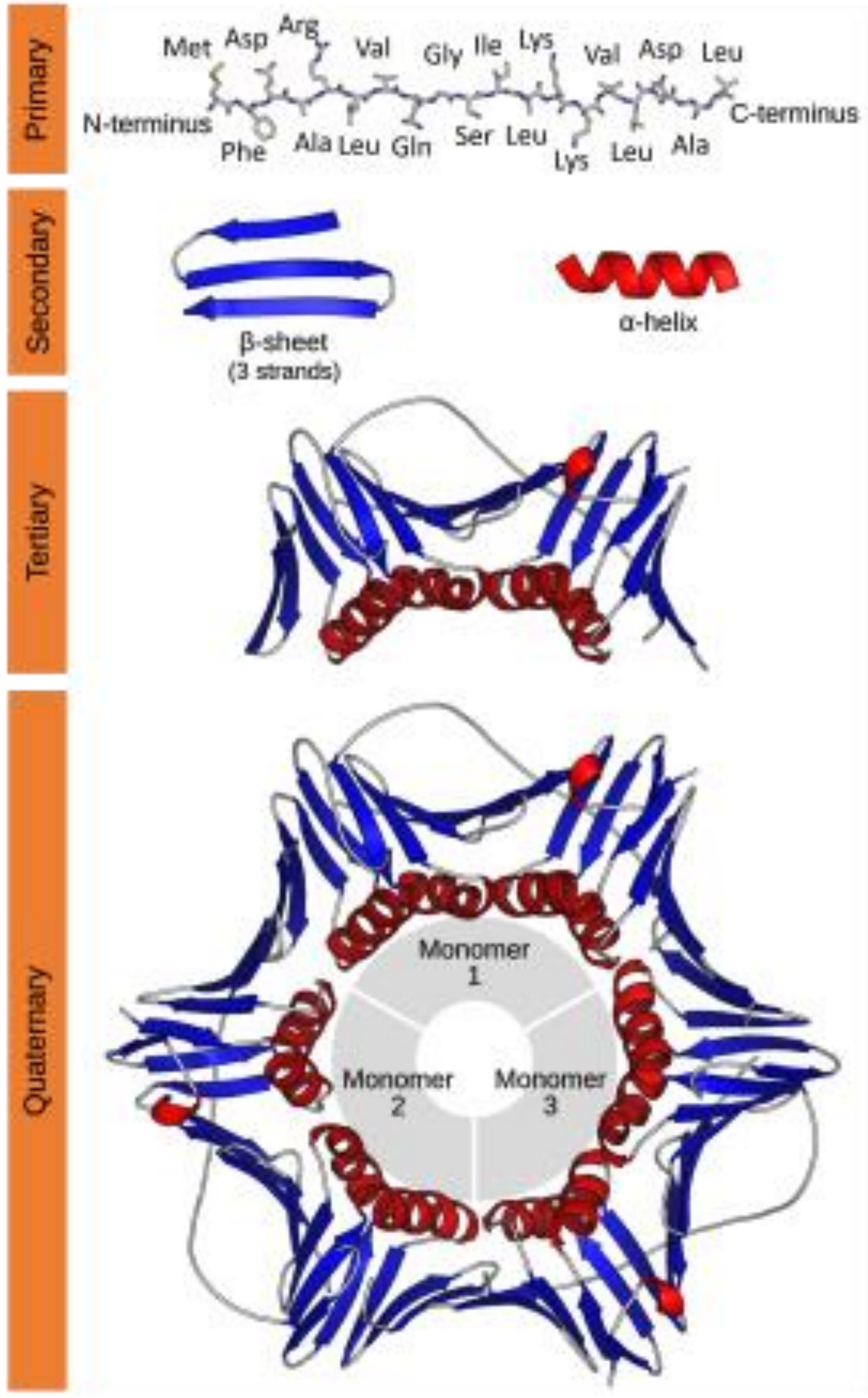
3D structure

Stabilizes the binding of IF-2
and IF-3 on the 30S subunit...
Belongs to the IF-1 family.
Subcellular location: Cytoplasm.

Biological functions and properties

Protein structure

- Primary
- Secondary
- Tertiary
- Quaternary

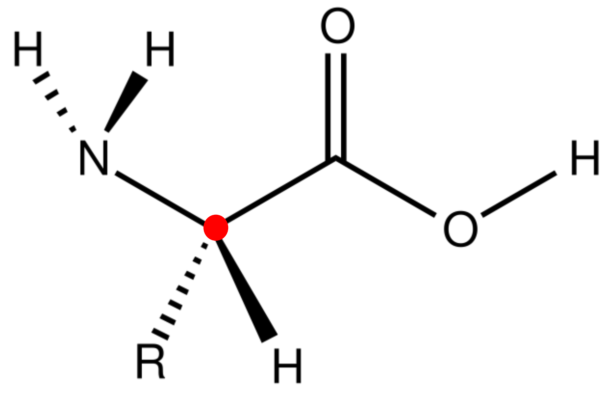


Protein design

- Folding
  - predict the structure that a given sequence might fold into
  - AlphaFold2[2]
- Inverse folding
  - design sequence that folds in to given structure
  - ProteinMPNN[3]
- **Structure / sequence design**
  - **Generate novel proteins** satisfying specific properties or constraints
  - Function, symmetry, substructure…

Chroma: overview

- Backbone generation
  - N-Ca-C=O
  - $\boldsymbol{x} \in \mathbb{R}^{4N\times 3}$
- Sequence and side-chain design
  - Based on generated backbone
  - Sequence: $\mathbf{s} \in [[20]]^N$
    - Amino acid type
  - Side-chain: $\boldsymbol{\chi} \in (-\pi, \pi]^{4N}$
    - Torsion angles

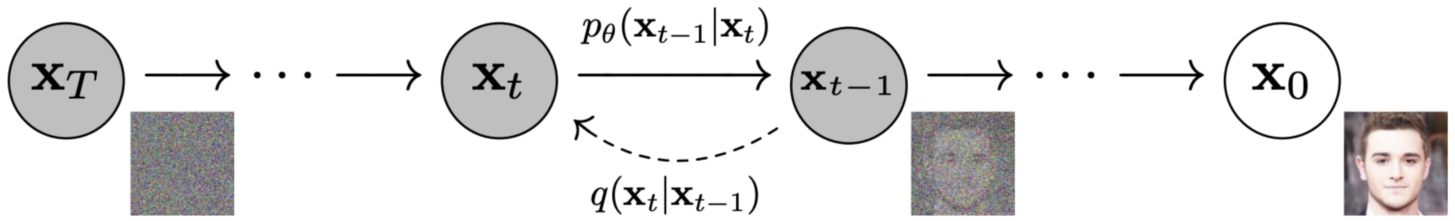


$$\log p_\theta(\mathbf{x},\mathbf{s},\chi) = \underbrace{\log p_\theta(\mathbf{x})}_{\text{backbone likelihood}} + \underbrace{\log p_\theta(\mathbf{s}|\mathbf{x})}_{\text{sequence likelihood}} + \underbrace{\log p_\theta(\chi|\mathbf{x},\mathbf{s})}_{\text{side-chain likelihood}}$$

diffusion

Inverse folding

Diffusion[4]


$\mathbf{x}_T$
$\mathbf{x}_t$
$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$\mathbf{x}_{t-1}$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$
$\mathbf{x}_0$

- Overview
  - Generate novel samples from a learned distribution
  - Learns a distribution by learning to denoise a noised data
  - Denoising process is broken down into many small steps for easier learning
- Forward process: gradually add (gaussian) noise to data

$$x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}z_t \longrightarrow x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}z$$

$$\boxed{x_{t-1}|x_t, x_0 \sim N(\tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I)}$$

- Reverse process: gradually denoise noisy data

$$\boxed{p_\theta(x_{t-1}|x_t) = N(x_{t-1}|\mu_\theta(x_t, t), \Sigma_\theta(x_t, t))}$$

$$p_\theta(x_T) = N(x_T|0, I) \qquad \mu_\theta(x_t, t) = \tilde{\mu}_t(x_t, x_0);$$

- Sampling: start from pure noise, do reverse process

Diffusion: training

- Train on ELBO of $\log p_\theta(x_0)$
  - Regard $x_1, \ldots, x_T$ as latent variable
  - $p_\theta(x_{0:T})$ is the reverse process, parameterized by neural network
  - $q(x_{1:T}|x_0)$ is the forward process, replaces intractable $p_\theta(x_{1:T}|x_0)$

$$\mathbb{E}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \mathbb{E}_q\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right]$$

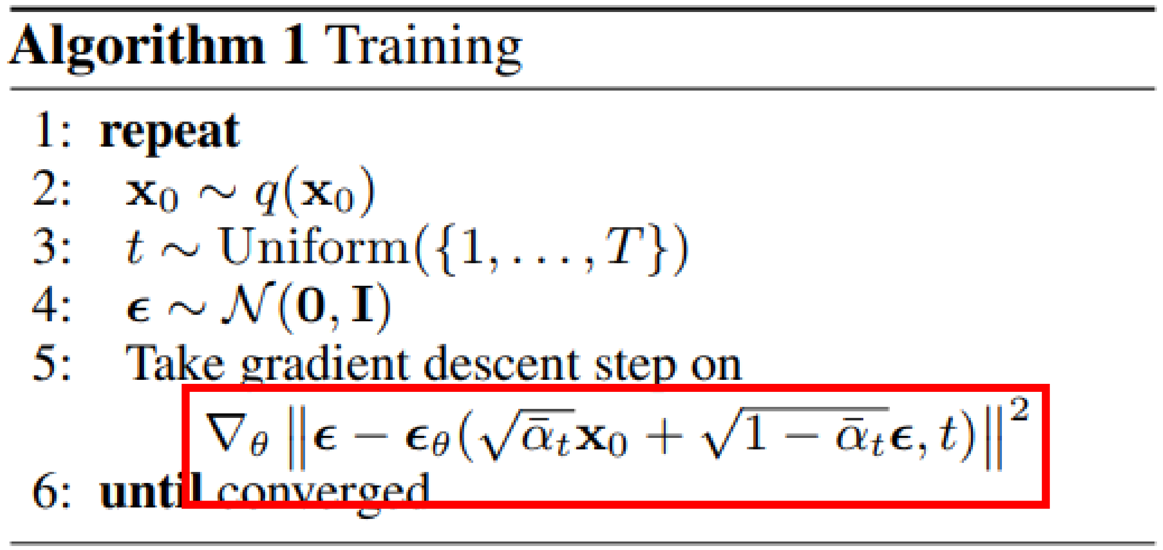
$$\mathbb{E}_q\left[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)\parallel p(\mathbf{x}_T))}_{L_T} + \sum_{t>1}\underbrace{D_{\mathrm{KL}}(\boxed{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}\parallel \boxed{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)})}_{L_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}\right]$$

Forward diffusion $\quad dx_t = -\frac{1}{2}\beta(t)x_t dt + \sqrt{\beta(t)}dW_t$

Reverse diffusion $\quad dy_t = \frac{1}{2}\beta(T-t)y_t dt + \beta(T-t)\underbrace{\nabla_{y_t}\ln\rho_{T-t}(y_t)}_{\text{score function}}dt + \sqrt{\beta(T-t)}dW_t$

Diffusion: training

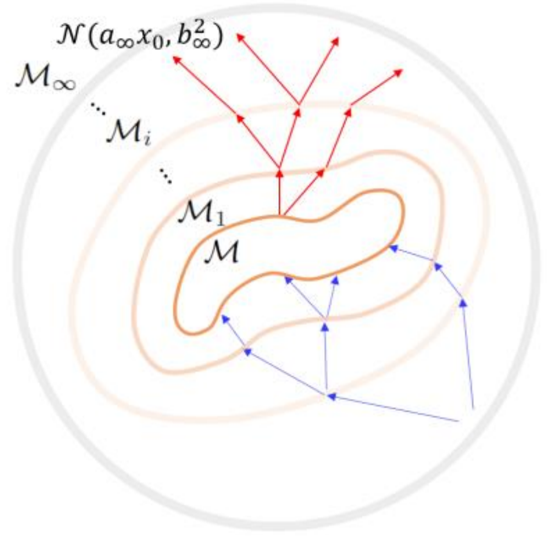
- Train on ELBO of $\log p_\theta(x_0)$
  - Regard $x_1, \dots, x_T$ as latent variable
  - $p_\theta(x_{0:T})$ is the reverse process, parameterized by neural network
  - $q(x_{1:T}|x_0)$ is the forward process, replaces intractable $p_\theta(x_{1:T}|x_0)$
- Essentially train on **denoising loss** / reconstruction loss



Algorithm 1 Training

```
1: repeat
2:   $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3:   $t \sim \mathrm{Uniform}(\{1, \dots, T\})$
4:   $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5:   Take gradient descent step on
         $\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$
6: until converged
```

Diffusion: why add noise



- **Intuition:** to learn a mapping from noise to data
- **Technically:** For better score estimation[15][16]
  - score: $\nabla_x \log p(x)$
  - If $x \sim N(x_0, \mathrm{I})$, $\nabla_x \log p(x) \propto x_0 - x \rightarrow$ point towards high density region
- Diffusion is a score-based generative model
  - The score of noised $p_\theta(x_{t-1}|x_t)$ "point towards" x_{t-1}

$$p_\theta(x_{t-1}|x_t) = N(x_{t-1}|\mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

  - Sampling of noiseless x_0 is achieved through a series of score guidance, $x_T \rightarrow x_{T-1} \rightarrow \cdots \rightarrow x_0$
  - Tweedie's formula: $\mathbb{E}(\mu|x) = x + \nabla_x \log p(x)$
    - $\mu \sim p(\mu)$ unobserved, $p(\mu)$ unknown
    - $x = \mu + \epsilon,\ \epsilon \sim N(0, I)$, observed

$\nabla_x \log p(x) = \mathbb{E}(\mu|x) - x$
point towards μ from x

Diffusion: why add noise

- **Score-based generative modeling**
- Score: $\nabla_x \log p(x)$
- How to sample with score: Langevin dynamics

$$\tilde{\mathbf{x}}_t = \tilde{\mathbf{x}}_{t-1} + \underbrace{\boxed{\frac{\epsilon}{2}\nabla_{\mathbf{x}} \log p(\tilde{\mathbf{x}}_{t-1})}}_{\text{towards high likelihood}} + \boxed{\sqrt{\epsilon}\,\mathbf{z}_t} \quad \text{avoid collapse at maxima}$$

- How to estimate score: score matching

$$\frac{1}{2}\mathbb{E}_{p_{\text{data}}}[\|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})\|_2^2]$$

Diffusion: why add noise

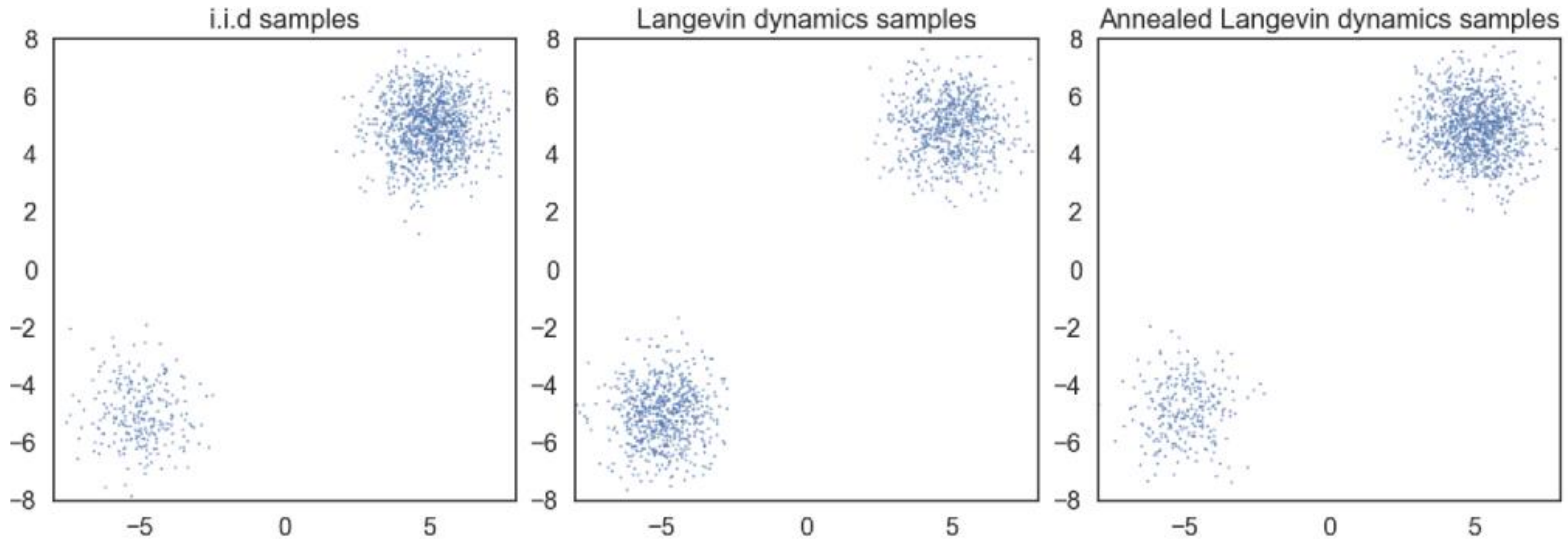
- **Challenges of noiseless score matching**
- Score undefined for most of the space
  - High dimensional data lie on low dimensional manifolds
  - Results difficulty in sampling
  - Noise perturbed score defined for entire space
- Score hard to learn for low density regions
  - Score matching accuracy depends on training data
  - Noise perturbed data is abundant

$$\frac{1}{2}\mathbb{E}_{\boxed{p_{\text{data}}}}\left[\|\mathbf{s}_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})\|_2^2\right]$$

Low density region
→ little training data
→ poor estimation

Diffusion: why add noise

- **Challenges of noiseless score matching**
- Langevin sampling do not recover relative weights
  - Even with ground truth score
  - Annealed Langevin dynamics recovers relative weights



Diffusion: why add noise

- **Denoising score matching**[17]
  - Estimation of $\nabla_{\tilde{x}} \log q_\sigma(\tilde{x})$
    - $q_\sigma(\tilde{x}, x) = q_\sigma(\tilde{x}|x)q(x)$, $q_\sigma(\tilde{x}) = \int q_\sigma(\tilde{x}, x)dx$
    - Learn slightly perturbed score
  - Explicit score matching ~ denoising score matching
    - Learn tractable score instead of intractable score
  - Diffusion: DSM with multiple noise scales

$$J_{ESMq_\sigma}(\theta) = \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}})}\left[\frac{1}{2}\left\|\psi(\tilde{\mathbf{x}};\theta) - \frac{\partial \log q_\sigma(\tilde{\mathbf{x}})}{\partial \tilde{\mathbf{x}}}\right\|^2\right]$$

ψ: parameterized score function

$$J_{DSMq_\sigma}(\theta) = \mathbb{E}_{q_\sigma(\mathbf{x},\tilde{\mathbf{x}})}\left[\frac{1}{2}\left\|\psi(\tilde{\mathbf{x}};\theta) - \frac{\partial \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}{\partial \tilde{\mathbf{x}}}\right\|^2\right]$$

derivation: exchange integral and partial

Diffusion: how to apply

- Choose variable / space to do diffusion on
  - Diffusion on raw data / latent features (latent diffusion[5])
  - Diffusion on entire space / manifolds[6] (e.g. on sphere)
  - Noise type: Gaussian, IGSO(3), Uniform…
- Forward schedule
  - Rate of noising data as time step increases
  - Linear schedule, Cosine schedule[7], …
  - Essentially weighs different SNR during training

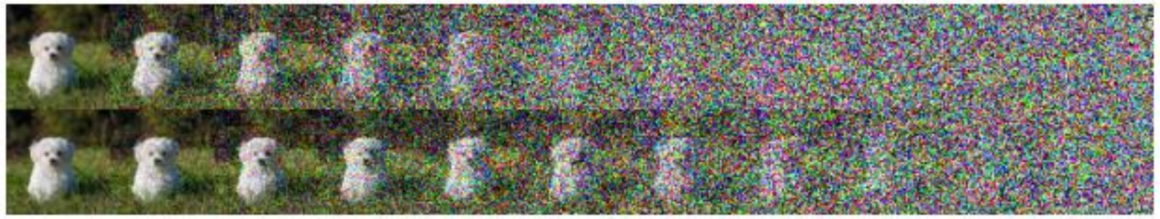
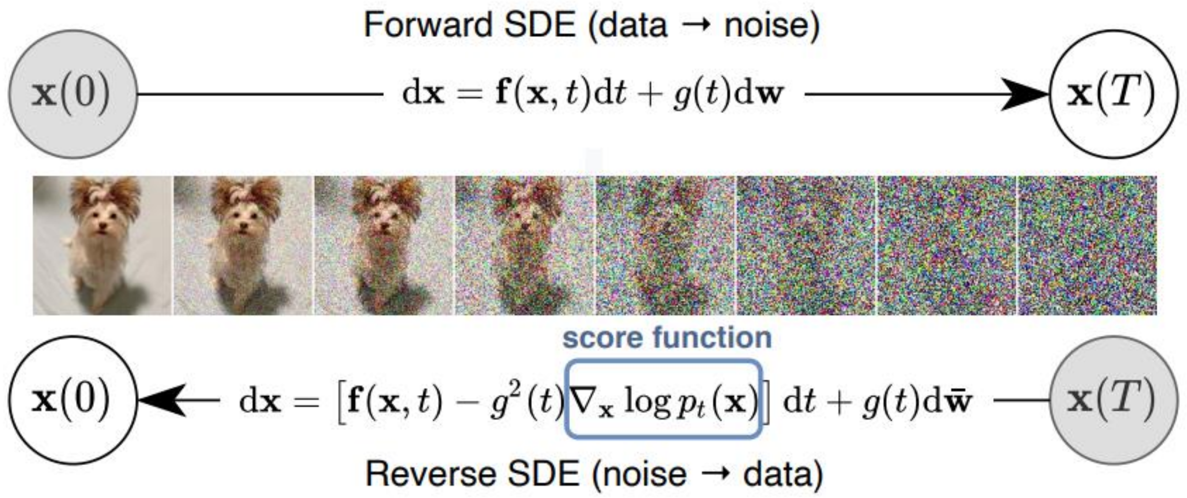


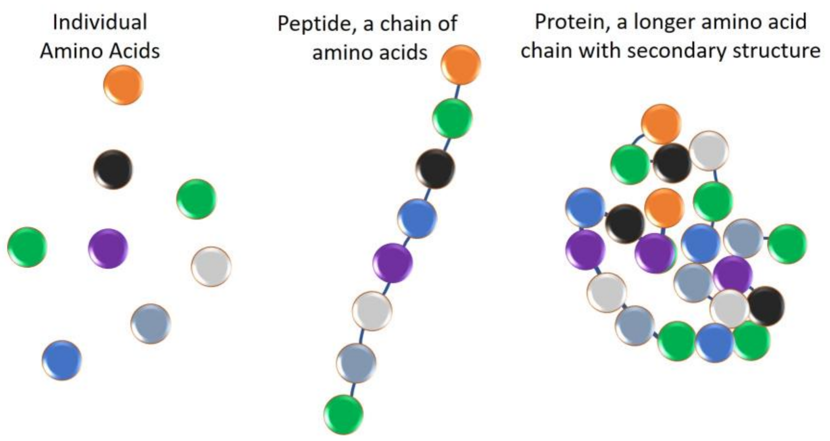
Figure 3. Latent samples from linear (top) and cosine (bottom)

Diffusion: how to apply

- Noise prediction network
  - A suitable neural network for diffusion variable
  - Input noisy data, output clean data / added noise
  - U-Net, DiT, GNN, ⋯
- Sampling procedure
  - Various SDE/ODE solvers based on learned **score function**[8]

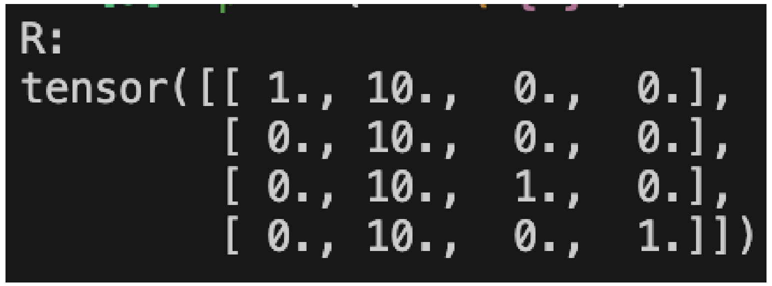

Forward SDE (data → noise)
$\mathbf{x}(0)$
$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$
$\mathbf{x}(T)$
score function
$\mathbf{x}(0)$
$\mathrm{d}\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g^2(t)\boxed{\nabla_\mathbf{x} \log p_t(\mathbf{x})}]\,\mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}}$
$\mathbf{x}(T)$
Reverse SDE (noise → data)

Diffusion: Chroma

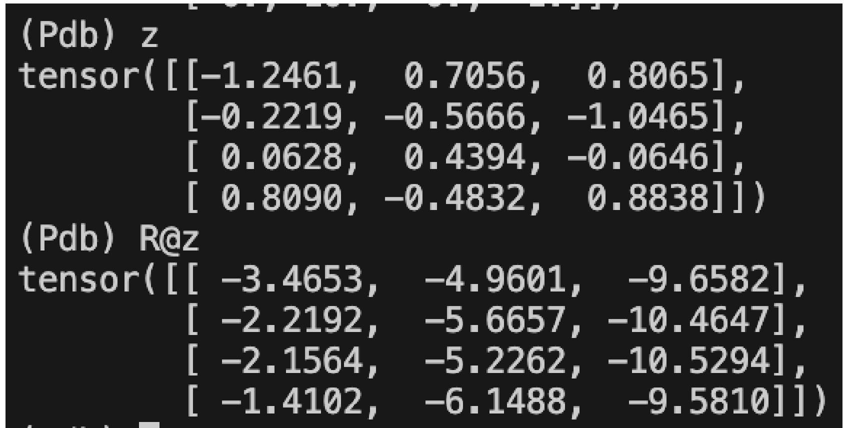

Individual Amino Acids
Peptide, a chain of amino acids
Protein, a longer amino acid chain with secondary structure

- Diffusion on backbone coordinates
  - N-Ca-C=O
  - $\boldsymbol{x} \in \mathbb{R}^{4N\times 3}$
- Diffusion with correlated noise $p(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}; \alpha_t \mathbf{x}_0, \sigma_t^2 \mathbf{R}\mathbf{R}^\intercal)$
  - Intuition: removes simple correlation in data, and diffusion learns the residual
  - Correlation: backbone atoms of the same residue have similar coords
  - Analogy: RGB channels of an image are usually correlated

Amino acid diameter: 4~10 Å
Bond lengths: 1~2 Å



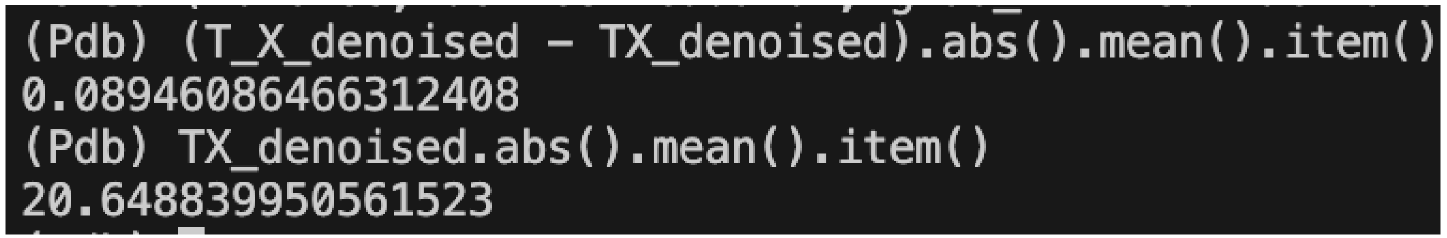
```
R:
tensor([[ 1., 10.,  0.,  0.],
        [ 0., 10.,  0.,  0.],
        [ 0., 10.,  1.,  0.],
        [ 0., 10.,  0.,  1.]])
```



```
(Pdb) z
tensor([[-1.2461,  0.7056,  0.8065],
        [-0.2219, -0.5666, -1.0465],
        [ 0.0628,  0.4394, -0.0646],
        [ 0.8090, -0.4832,  0.8838]])
(Pdb) R@z
tensor([[ -3.4653,  -4.9601,  -9.6582],
        [ -2.2192,  -5.6657, -10.4647],
        [ -2.1564,  -5.2262, -10.5294],
        [ -1.4102,  -6.1488,  -9.5810]])
```

Diffusion: Chroma

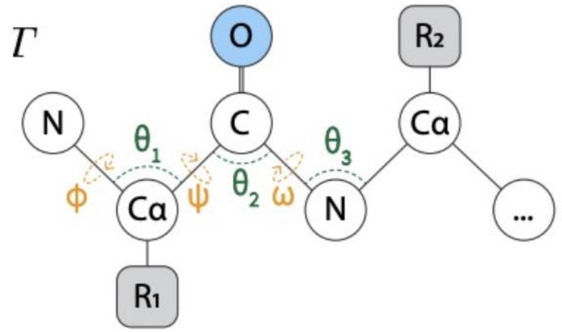
- Noise prediction network
  - Should be equivariant to **rotation and translation** (SE(3))
  - Rotations and translations should not affect denoise performance
- Equivariance
  - $f \circ g(x) = g \circ f(x)$, $\forall g \in \mathcal{G}$, then f is equivariant to $\mathcal{G}$
  - Invariance: $f \circ g(x) = f(x)$
- **SE(3)-equivariant GNN** as noise prediction network
  - `denoise(T @ X) = T @ denoise(X)` $\leftarrow x = a\, x_0 + b\, \epsilon$
  - `T` is an SE(3) transformation: $T(x) = Rx + t$

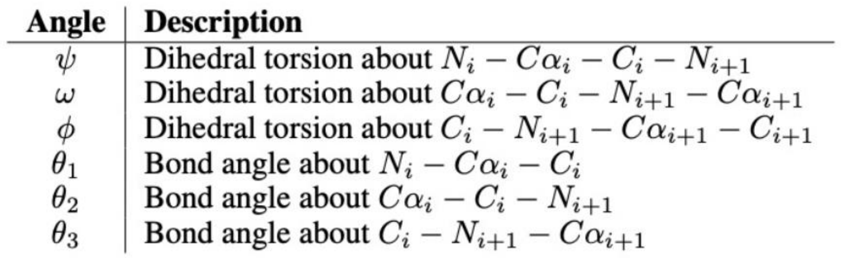


```
(Pdb) (T_X_denoised - TX_denoised).abs().mean().item()
0.08946086466312408
(Pdb) TX_denoised.abs().mean().item()
20.648839950561523
```

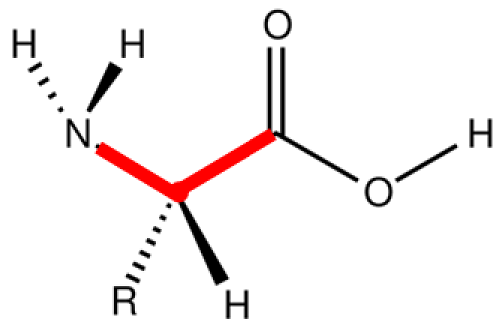
Equivariant GNN: How?

- Equivariant to SE(3) transformation on coordinates
- Method: **invariant features + equivariant operations**
- Initialize invariant node and edge features
  - Relative distances, internal angles[9], bond lengths, protein diameters
  - Use MPNN to update initial features (concat neighbors, mlp, attn···)
- Update coords with equivariant operations
  - Extract, compose, average

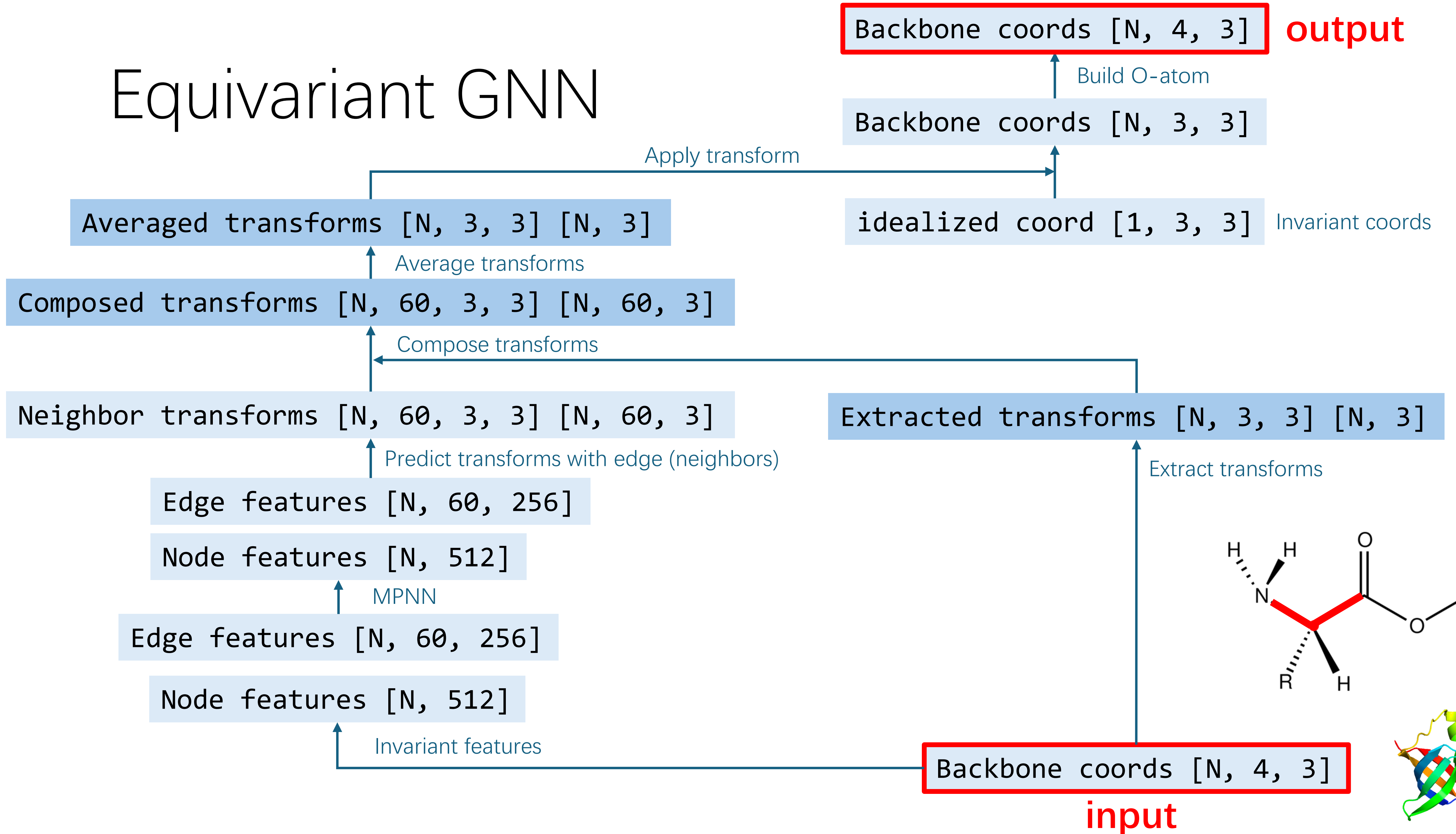


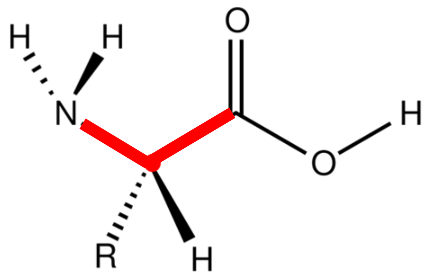


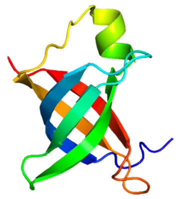
Angle	Description
ψ	Dihedral torsion about $N_i - C\alpha_i - C_i - N_{i+1}$
ω	Dihedral torsion about $C\alpha_i - C_i - N_{i+1} - C\alpha_{i+1}$
ϕ	Dihedral torsion about $C_i - N_{i+1} - C\alpha_{i+1} - C_{i+1}$
θ_1	Bond angle about $N_i - C\alpha_i - C_i$
θ_2	Bond angle about $C\alpha_i - C_i - N_{i+1}$
θ_3	Bond angle about $C_i - N_{i+1} - C\alpha_{i+1}$



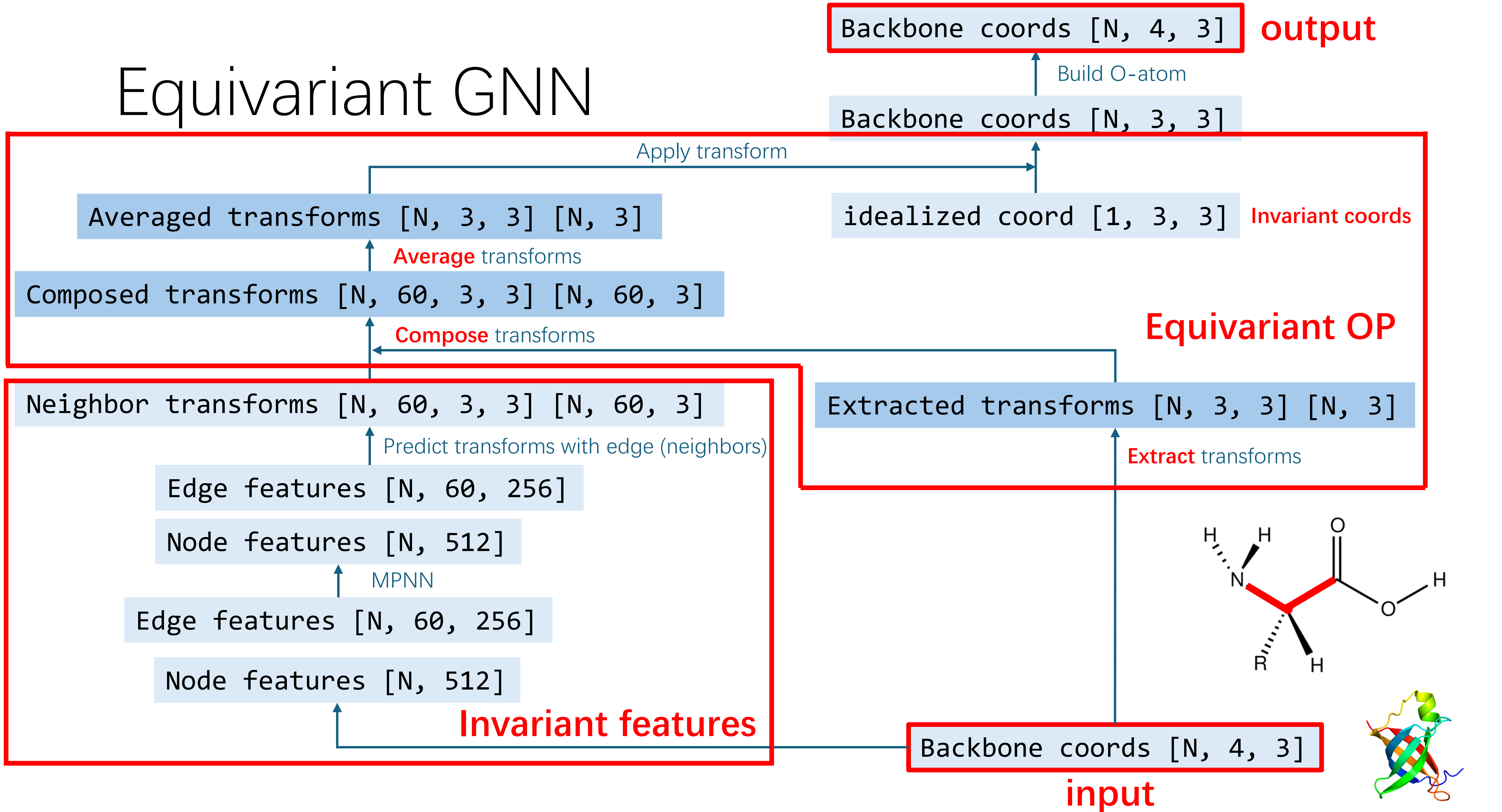
Equivariant GNN


Backbone coords [N, 4, 3]
output
Build O-atom
Backbone coords [N, 3, 3]
Apply transform
Averaged transforms [N, 3, 3] [N, 3]
idealized coord [1, 3, 3]
Invariant coords
Average transforms
Composed transforms [N, 60, 3, 3] [N, 60, 3]
Compose transforms
Neighbor transforms [N, 60, 3, 3] [N, 60, 3]
Extracted transforms [N, 3, 3] [N, 3]
Predict transforms with edge (neighbors)
Extract transforms
Edge features [N, 60, 256]
Node features [N, 512]
MPNN
Edge features [N, 60, 256]
Node features [N, 512]
Invariant features
Backbone coords [N, 4, 3]
input

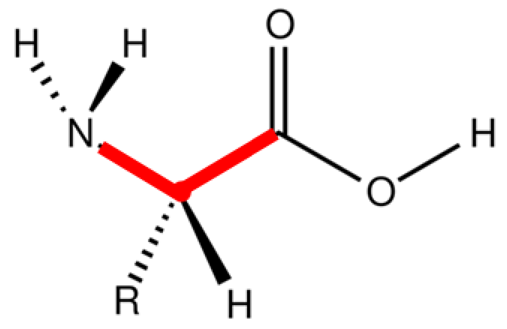




Equivariant GNN


Backbone coords [N, 4, 3]
output
Build O-atom
Backbone coords [N, 3, 3]
Apply transform
Averaged transforms [N, 3, 3] [N, 3]
idealized coord [1, 3, 3]
Invariant coords
Average transforms
Composed transforms [N, 60, 3, 3] [N, 60, 3]
Compose transforms
Equivariant OP
Neighbor transforms [N, 60, 3, 3] [N, 60, 3]
Extracted transforms [N, 3, 3] [N, 3]
Predict transforms with edge (neighbors)
Extract transforms
Edge features [N, 60, 256]
Node features [N, 512]
MPNN
Edge features [N, 60, 256]
Node features [N, 512]
Invariant features
Backbone coords [N, 4, 3]
input
H
H
N
O
H
O
R
H

GNN: Extract transforms



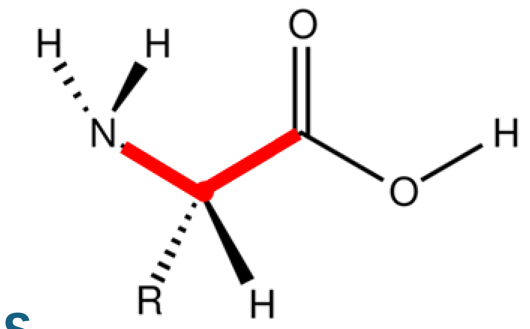
- **From backbone coords to rotations and translations**
  - $\text{extract}(x) = (R, t) = T_{\text{extract}}$
  - `[N, 4, 3]` → `[N, 3, 3]` + `[N, 3]`
  - `[N, 3, 3]`: 3x3 rotation matrix
    - Ca → N direction
    - Cross(Ca → N, Ca → C)
    - Cross product of the above 2 directions
  - `[N, 3]`: Ca coords

Global rotation of N-Ca-C

Global translation of N-Ca-C

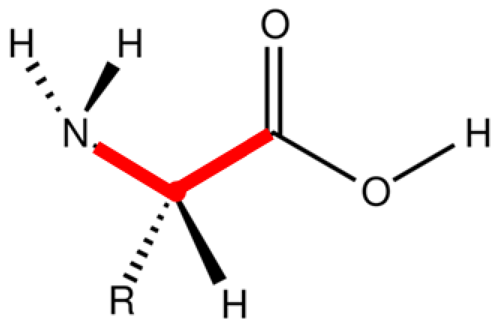
- For reconstruction, use canonical bond length and bond angles (x^*)
  - Diffusion noising process may produce variant bond lengths
  - Only extract **direction** from noised coords, distance is ignored
  - O atom coords are extrapolated from N-Ca-C coords
  - $x \sim T_{\text{extract}}(x^*), \quad T(x) \sim T \circ T_{\text{extract}}(x^*)$
- Equivariant: $\text{extract}\big(T(x)\big) = T(\text{extract}(x))$

GNN: Compose transforms



- **Compose extracted transform and neighbor transforms**
- Extracted transform: (R_i, t_i)
  - $i = 1, \dots, \mathrm{num\_residue}$
  - Rotation and position of each individual amino acid
  - Origin → residue
- Neighbor transforms: (R_{ji}, t_{ji})
  - $i = 1, \dots, \mathrm{num\_residue}, j = 1, \dots, \mathrm{num\_neighbor}$
  - Relative orientation and position of each amino acid from its neighbor's perspective
  - Neighbor → residue
- Composed transforms: $\left(R_j R_{ji}, t_j + R_j t_{ji}\right) = \left(R_j, t_j\right) \circ (R_{ji}, t_{ji})$
  - Absolute orientation and position of each amino acid from its neighbor's perspective
  - Origin → neighbor → residue (compose in **reverse** order)

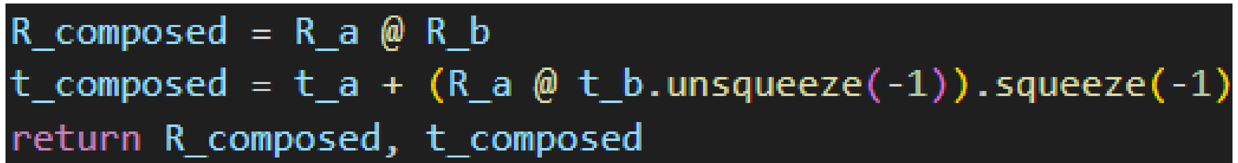
GNN: Compose transforms



- **Compose extracted transform and neighbor transforms**
- Intuition: from neighbors' relative view to global view
  - $T_{\text{neighbor}} \to T_{\text{extract}} \circ T_{\text{neighbor}} \quad (T_{ji} \to T_j \circ T_{ji})$
  - Note: T_{neighbor} is predicted from invariant edge features of x
- Previous slide: $\text{extract}\big(T(x)\big) = T\big(\text{extract}(x)\big)$
- Compose in reverse order (for equivariance)
  - $T_{\text{extract}} \circ T_{\text{neighbor}}$, for x. $\quad (T_j \circ T_{ji})$
  - $T \circ T_{\text{extract}} \circ T_{\text{neighbor}}$, for $T(x)$. $\quad (T \circ T_j \circ T_{ji})$

Global transform of residue i from neighbor j's opinion

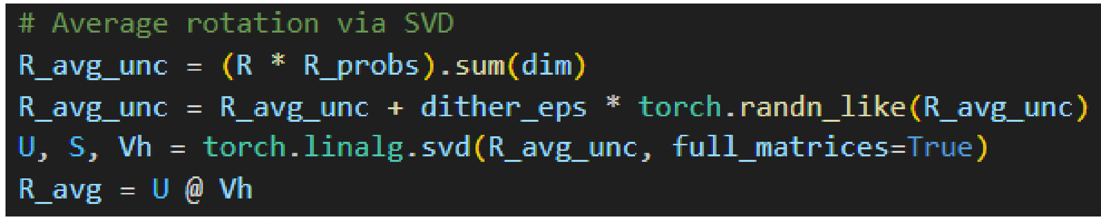
- Conclusion: Compose is equivariant



```
R_composed = R_a @ R_b
t_composed = t_a + (R_a @ t_b.unsqueeze(-1)).squeeze(-1)
return R_composed, t_composed
```

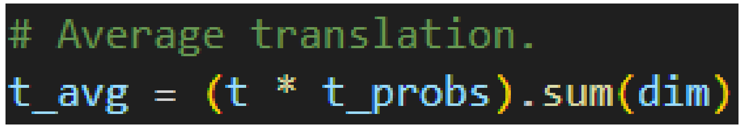
GNN: Average transforms

- **Average composed neighbor transforms (consensus structure)**
- $\mathrm{avg}_R(\{R_i\})$ for $i = 1, \dots, \mathrm{num\_neighbor}$
  - Weight $\{w_i\}$ from edge features
  - Take SVD: $\sum w_i R_i = UDV^T$
  - $\mathrm{avg}_R(\{R_i\}) \coloneqq UV^T \times \det(UV^T)$, (to ensure det=+1)
  - observation: $\mathrm{avg}_R(\{RR_i\}) = \mathrm{R}\,\mathrm{avg}_R(\{R_i\})$



```
# Average rotation via SVD
R_avg_unc = (R * R_probs).sum(dim)
R_avg_unc = R_avg_unc + dither_eps * torch.randn_like(R_avg_unc)
U, S, Vh = torch.linalg.svd(R_avg_unc, full_matrices=True)
R_avg = U @ Vh
```

- $\mathrm{avg}_t(\{t_i\})$
  - Weight $\{w_i\}$ from edge features
  - $\mathrm{avg}_t(\{t_i\}) \coloneqq \sum w_i t_i$
  - observation: $\mathrm{avg}_t(\{Rt_i + t\}) = R\,\mathrm{avg}_t(\{t_i\}) + t$



```
# Average translation.
t_avg = (t * t_probs).sum(dim)
```

- Conclusion: Average is equivariant

GNN: Average transforms

- **Average composed neighbor transforms (consensus structure)**
- Why SVD: consensus structure optimization
- $d(T, T_i) = \|R - R_i\|_F^2 + \|t - t_i\|^2$
- Find T minimize $\sum w_i d(T, T_i)$
- Average rotations:
    - $d(R, R_i) = \|R - R_i\|_F^2 = \mathrm{tr}((R - R_i)^T (R - R_i))$
    - Minimize $\sum w_i d(R, R_i)$
    - **$\Leftrightarrow$ Maximize $\mathbf{tr}(\boldsymbol{R}^T \overline{\boldsymbol{R}})$, $\overline{\boldsymbol{R}} = \sum \boldsymbol{w}_i \boldsymbol{R}_i$**

GNN: Average transforms

- **Average composed neighbor transforms (consensus structure)**
- Why SVD: consensus structure optimization
- **Maximize $\mathrm{tr}(R^T\bar{R})$, $\bar{R} = \sum w_i R_i$**
- $\bar{R} = UDV^T = (UV^T)(VDV^T) = \widetilde{U}P$
    - Polar decomposition: $\widetilde{U}$ orthogonal, $P \succcurlyeq 0$
- $R^T\bar{R} = (R^T\widetilde{U})P = OP$, O orthogonal, $P \succcurlyeq 0$
- **Lemma**: $\mathrm{tr}(OP) \leq \mathrm{tr}(P)$, equal when $O|_P = I \Leftrightarrow R = \widetilde{U} = UV^T$
    - Proof: using $P = VDV^T$, we have $\mathrm{tr}(OP) = \sum d_i < Ov_i, v_i >$

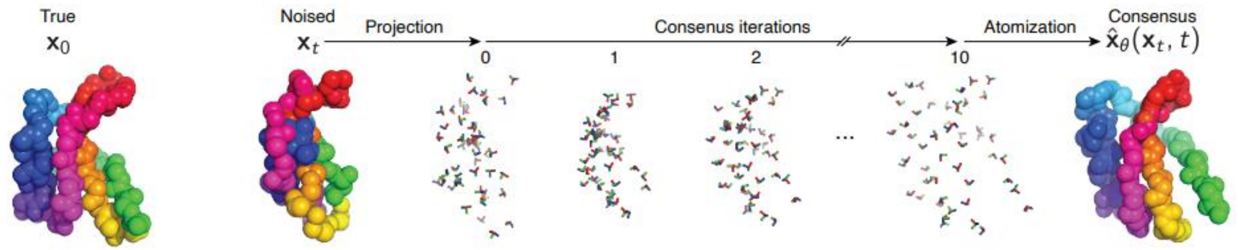
GNN: Average transforms

- **Average composed neighbor transforms (consensus structure)**
- Why SVD: consensus structure optimization

$$\mathbf{T}_i^\star \leftarrow \arg\min_{\mathbf{T}_i} U\left(\{\mathbf{T}_i\};\{w_{ij},\mathbf{T}_{ij}\}\right).$$

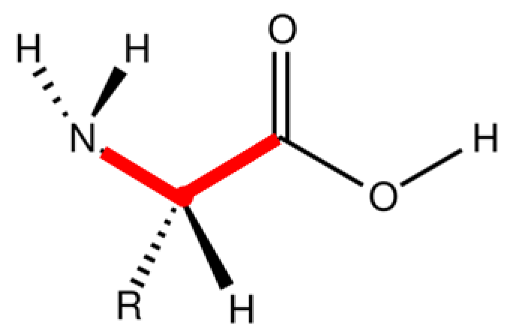
$$\begin{aligned} U\left(\{\mathbf{T}_i\};\{w_{ij},\mathbf{T}_{ij}\}\right) &= \sum_{i,j} w_{ij}\left\|\mathbf{T}_i - \mathbf{T}_j \circ \mathbf{T}_{ji}\right\|^2 \\ &= \sum_{i,j} w_{ij}\left\|\mathbf{O}_i - \mathbf{O}_j\mathbf{O}_{ji}\right\|^2 + w_{ij}\left\|\mathbf{t}_i - (\mathbf{O}_j\mathbf{t}_{ji}+\mathbf{t}_j)\right\|^2. \end{aligned}$$

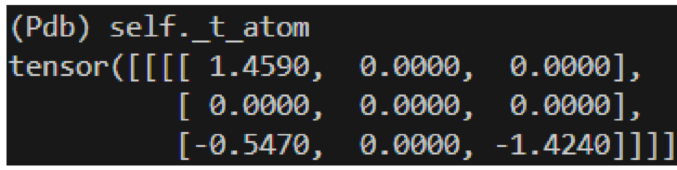
$$\mathbf{T}_i^\star = \left(\mathrm{Proj}_{\mathrm{SO}(3)}\left(\sum_j p_{ij}\mathbf{O}_j\mathbf{O}_{ji}\right), \sum_j p_{ij}(\mathbf{O}_j\mathbf{t}_{ji}+\mathbf{t}_j)\right), \quad \text{where } p_{ij} = \frac{w_{ij}}{\sum_j w_{ij}}$$



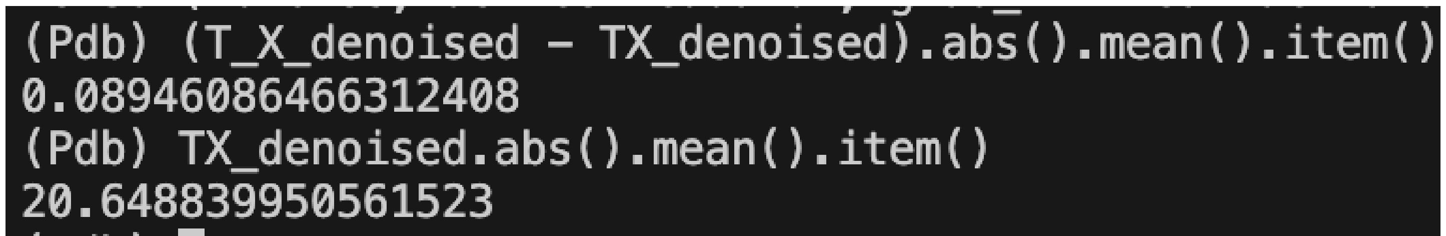
GNN: all-together

- Input: x (noisy coords)
- Output: $T_{\mathrm{GNN}}(x^*)$ (denoised coords)
  - x^* is idealized coords with Ca at origin, invariant to x
- $T_{\mathrm{GNN}} = \underbrace{\mathrm{average}}_{\text{equivariant}}(\underbrace{T_{\mathrm{extract}}}_{\text{equivariant}} \underbrace{\circ}_{\text{equivariant}} \underbrace{T_{\mathrm{neighbor}}}_{\text{invariant}})$
  - Average, extract, compose are equivariant (equivariant OP)
  - T_{neighbor} is invariant (from invariant features)
- Conclusion: T_{GNN} is equivariant




```
(Pdb) self._t_atom
tensor([[[[ 1.4590,  0.0000,  0.0000],
          [ 0.0000,  0.0000,  0.0000],
          [-0.5470,  0.0000, -1.4240]]]]
```

x^*

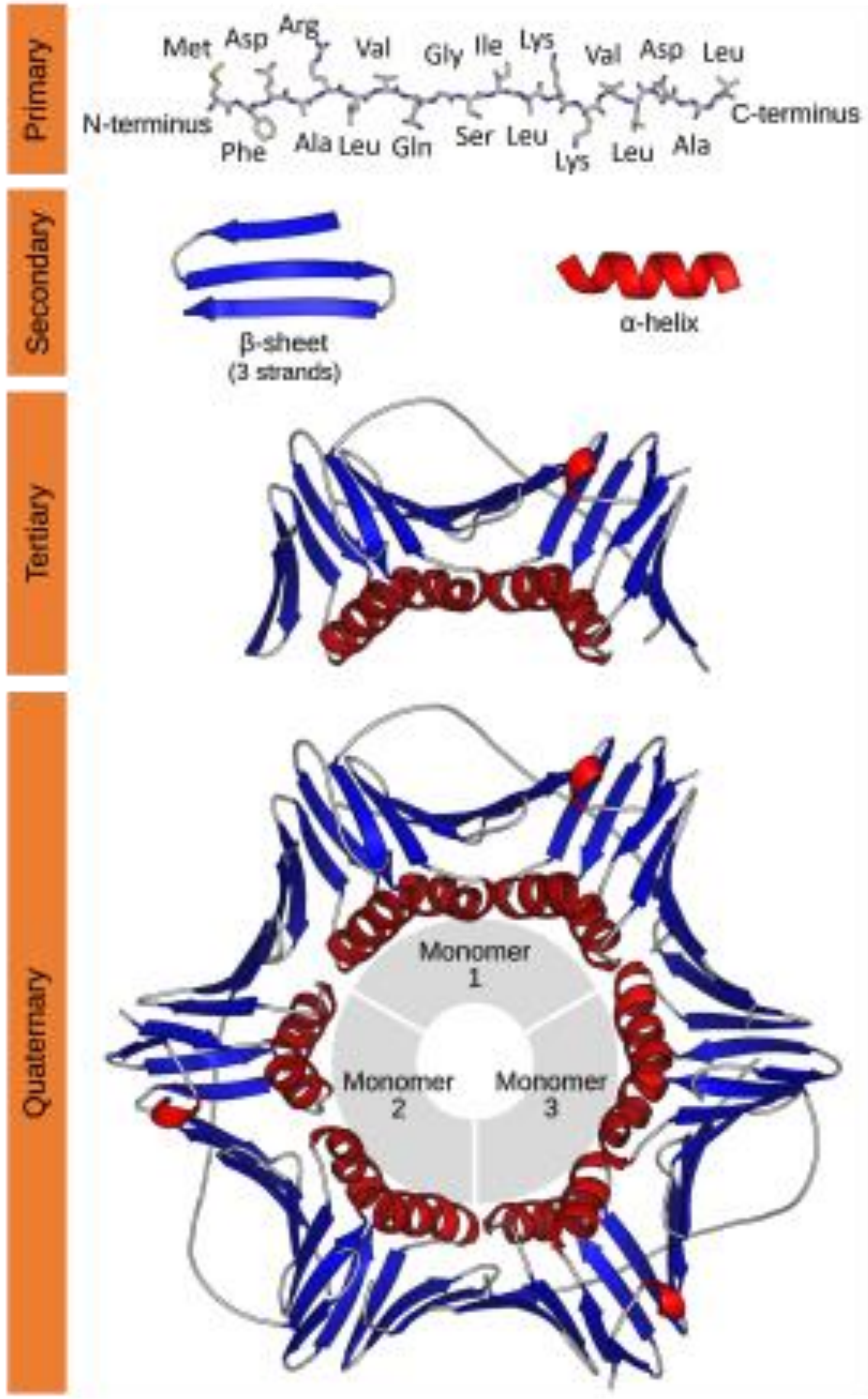

```
(Pdb) (T_X_denoised - TX_denoised).abs().mean().item()
0.0894608646312408
(Pdb) TX_denoised.abs().mean().item()
20.648839950561523
```

Diffusion: Chroma training

- ELBO (backbone atom-wise MSE)
- Substructure MSE
  - Essentially weighted MSE emphasizing local substructure
  - Why: ELBO mainly focus on low-freq statistics
    - Analogy: SD generate fingers poorly
  - How: substructure regions of training data can be easily identified
- Relative distance matrix MSE

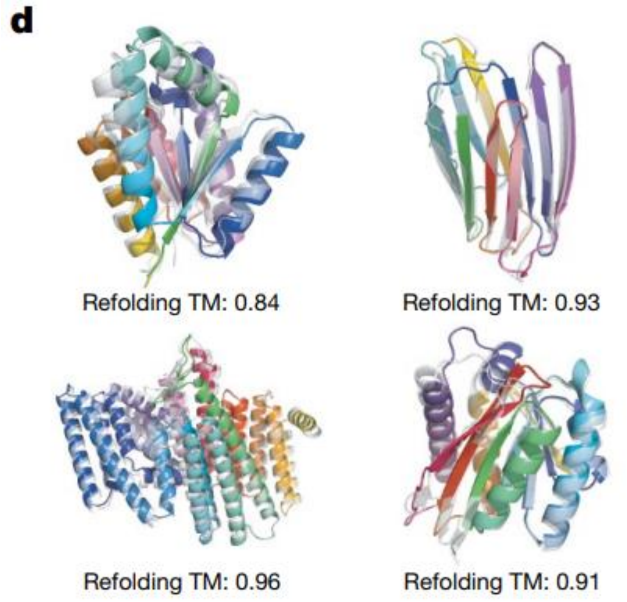
$$\mathcal{D}_{\text{distance}}(\mathbf{x}, \mathbf{x}') = \sum_{ij} (D_{ij}^{\text{CA}}(\mathbf{x}) - D_{ij}^{\text{CA}}(\mathbf{x}'))^2$$

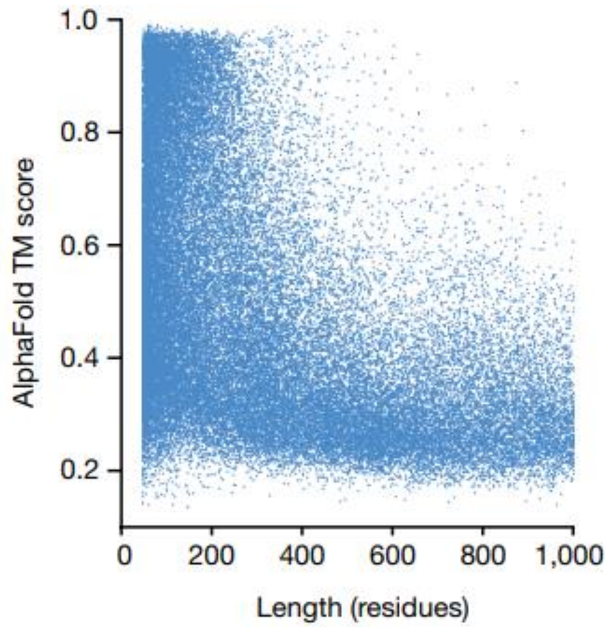
(D: Internal distance)



Evaluation of unconditional samples

- **Refolding TM-score**
- Run inverse folding on generated backbone
- Predict structure of designed sequence with AlphaFold2
- Compare refolded structure and generated structure





Conditional generation

- General method
- Chroma’s conditioners
  - Symmetry constraint
  - Substructure constraint
  - Protein class
  - Shape guidance
  - Text guidance

Conditional generation

- General method: **Conditional reverse process**
- Unconditional generation: sample from $p_\theta(x_{t-1}|x_t)$
- Conditional generation: sample from $p_\theta(x_{t-1}|x_t, y)$
  - y is the condition
- In reverse sampling, use score of sample distribution

$$
\begin{aligned}
\underbrace{\nabla_{\mathbf{x}} \log p_t(\mathbf{x}|\mathbf{y})}_{\text{Conditional score}} &= \nabla_{\mathbf{x}} \log \frac{p_t(\mathbf{x})p_t(\mathbf{y}|\mathbf{x})}{p_t(\mathbf{y})} \\
&= \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) + \nabla_{\mathbf{x}} \log p_t(\mathbf{y}|\mathbf{x}) - \cancel{\nabla_{\mathbf{x}} \log p_t(\mathbf{y})} \\
&= \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) + \boxed{\nabla_{\mathbf{x}} \log p_t(\mathbf{y}|\mathbf{x}).}
\end{aligned}
$$

Unconditional score + conditional gradient

Conditional generation

- General method: **Conditional reverse process**
- Unconditional generation: sample from $p_\theta(x_{t-1}|x_t)$
- Conditional generation: sample from $p_\theta(x_{t-1}|x_t, y)$
  - y is the condition
- In reverse sampling, use score of sample distribution
  - Combination of conditions

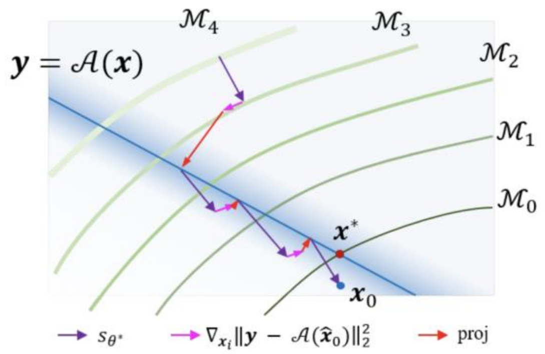
$$\nabla_{\mathbf{x}} \log p_t(\mathbf{x}|\mathbf{y}_1, \ldots, \mathbf{y}_M) = \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) + \sum_{i=1}^{M} \nabla_{\mathbf{x}} \log p_t(\mathbf{y}_i|\mathbf{x}).$$

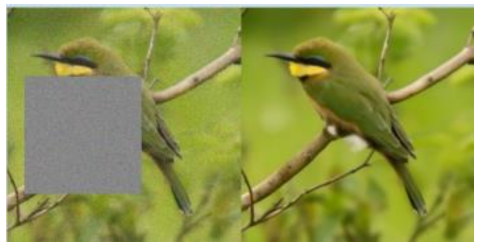
Unconditional score + conditional gradients

Conditional generation

- General method: **Conditional reverse process**
- Unconditional generation: sample from $p_\theta(x_{t-1}|x_t)$
- Conditional generation: sample from $p_\theta(x_{t-1}|x_t, y)$
  - y is the condition
- Or use projection instead of $\nabla_x \log p_t(y|x)$
- Example[10]: inpainting $\mathrm{mask}(x_0^*)$

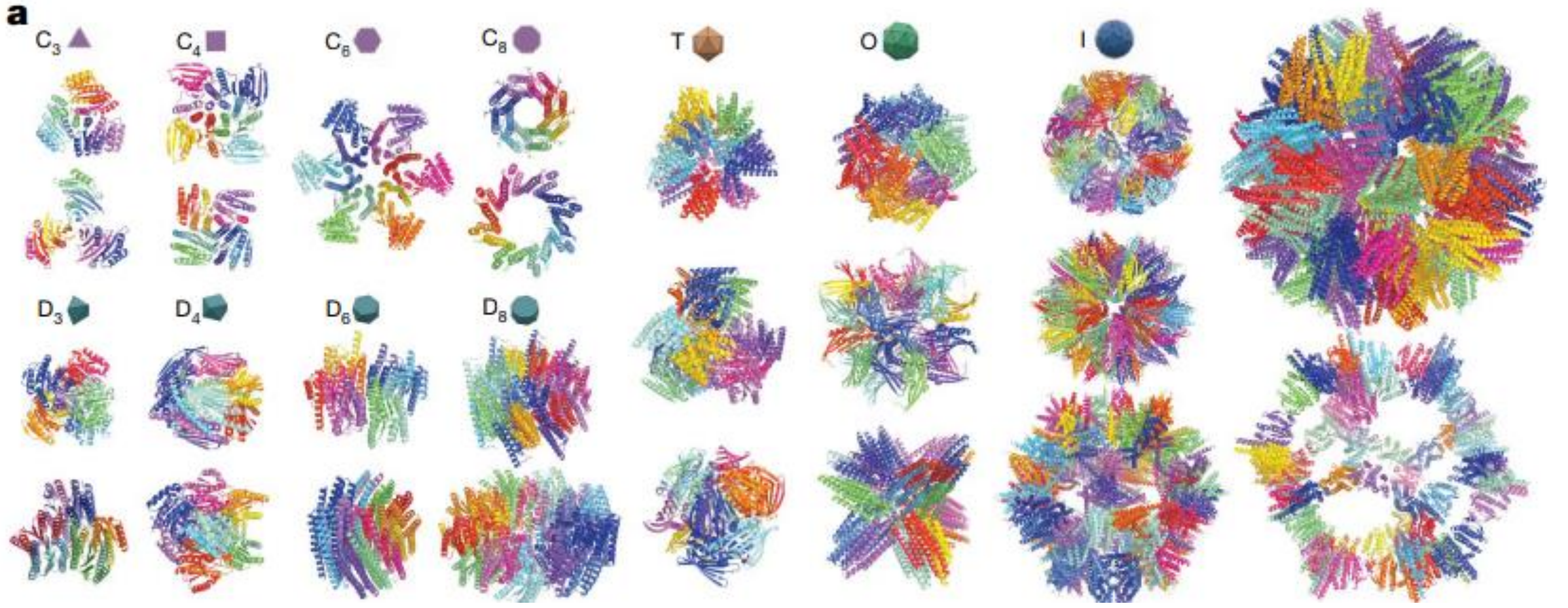
Noisy x_t → denoised x_0 ← Unconditional update
$\rightarrow \mathrm{x}_0 = \mathrm{mask}(x_0^*) + (1 - \mathrm{mask})(x_0)$ ← Projection step
$\rightarrow \mathrm{x}_{t-1} \sim q(x_{t-1}|x_t, x_0)$ → denoised x_0
$\rightarrow \cdots$

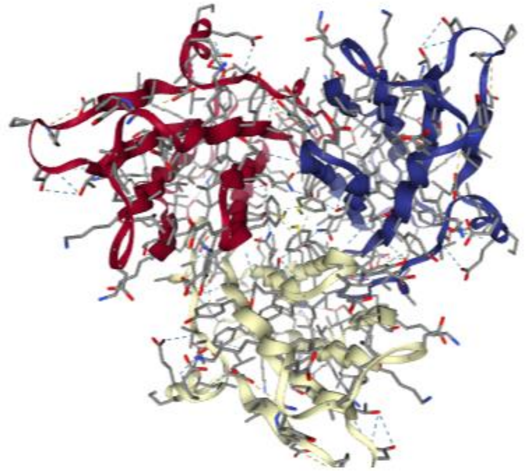


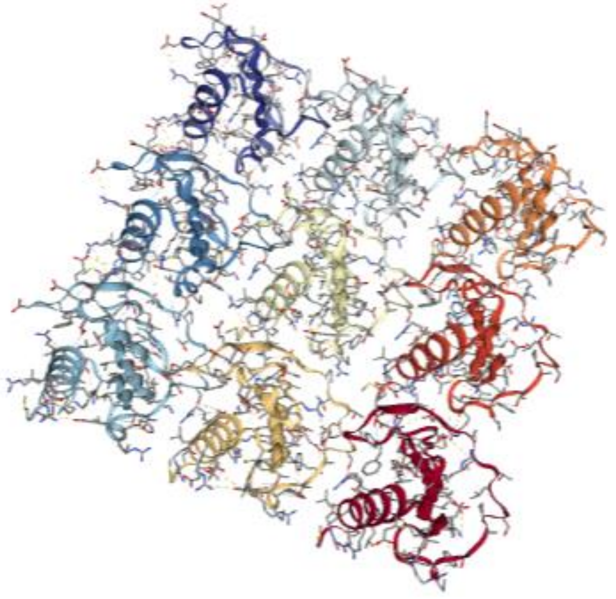

$\mathrm{mask}(x_0^*)$ x_0^*

Conditional generation: Chroma

- **Symmetry constraint**
- Unconditional generation → projection
- Condition as rotation matrices

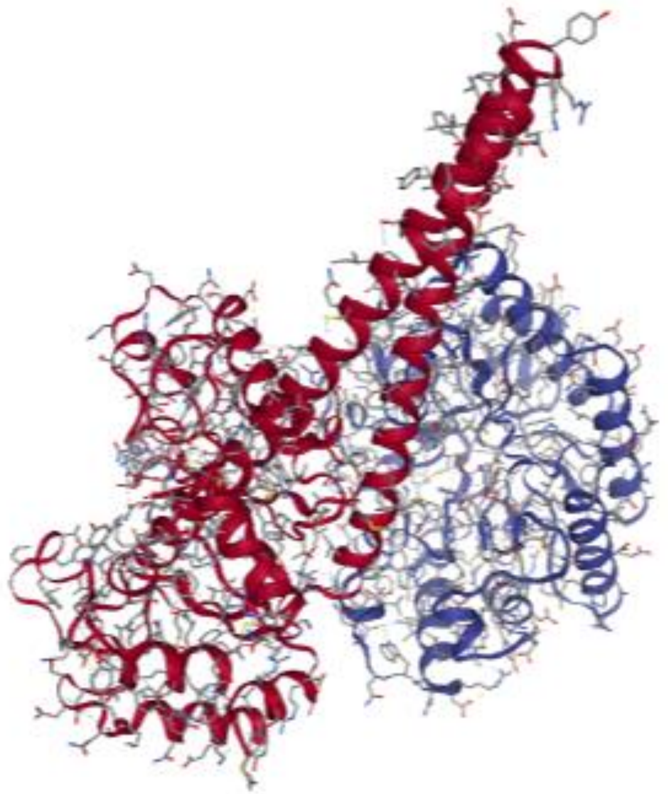


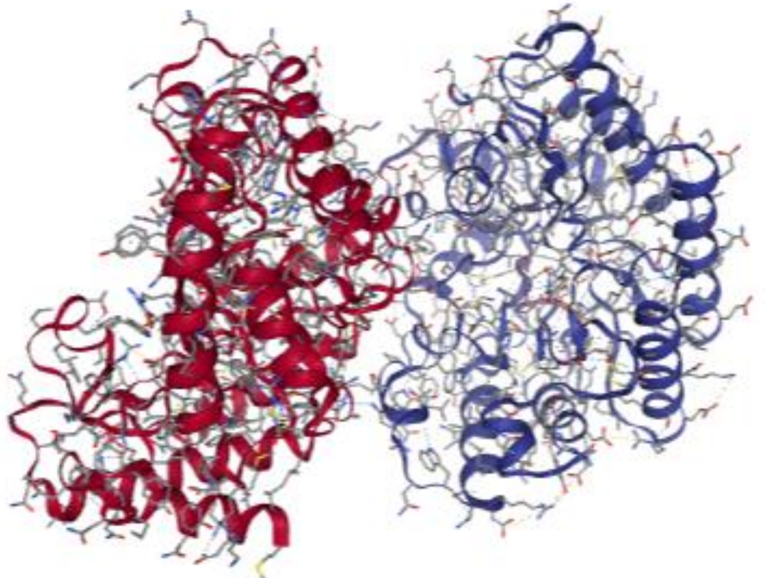


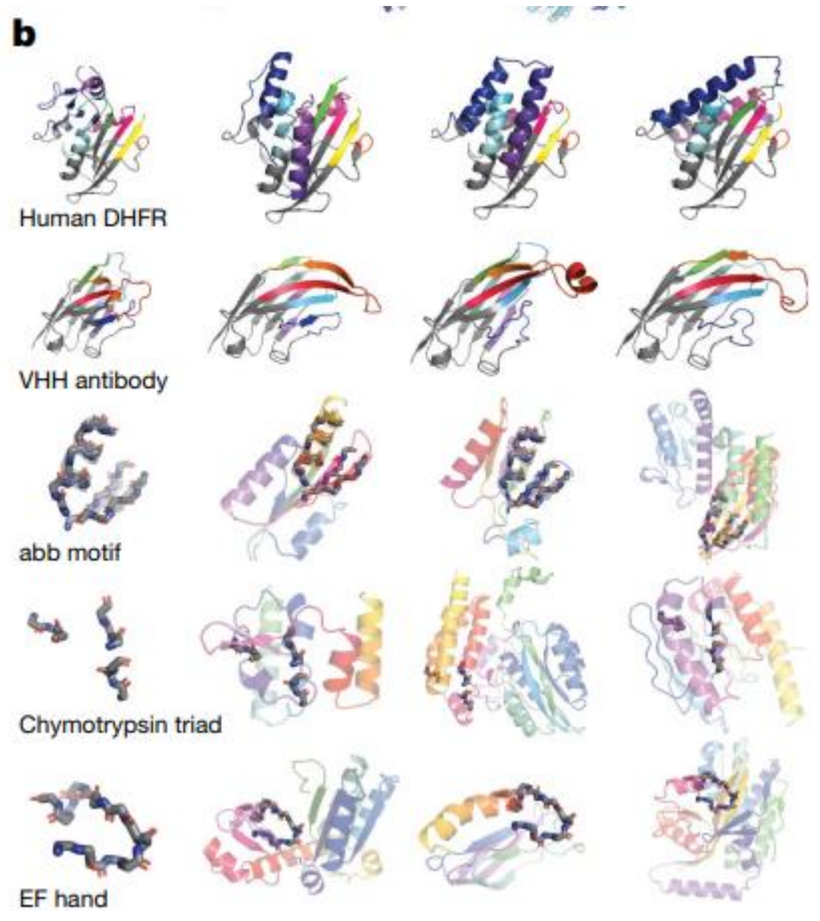


Conditional generation: Chroma

- **Substructure constraint** (inpainting)
- Unconditional generation → projection

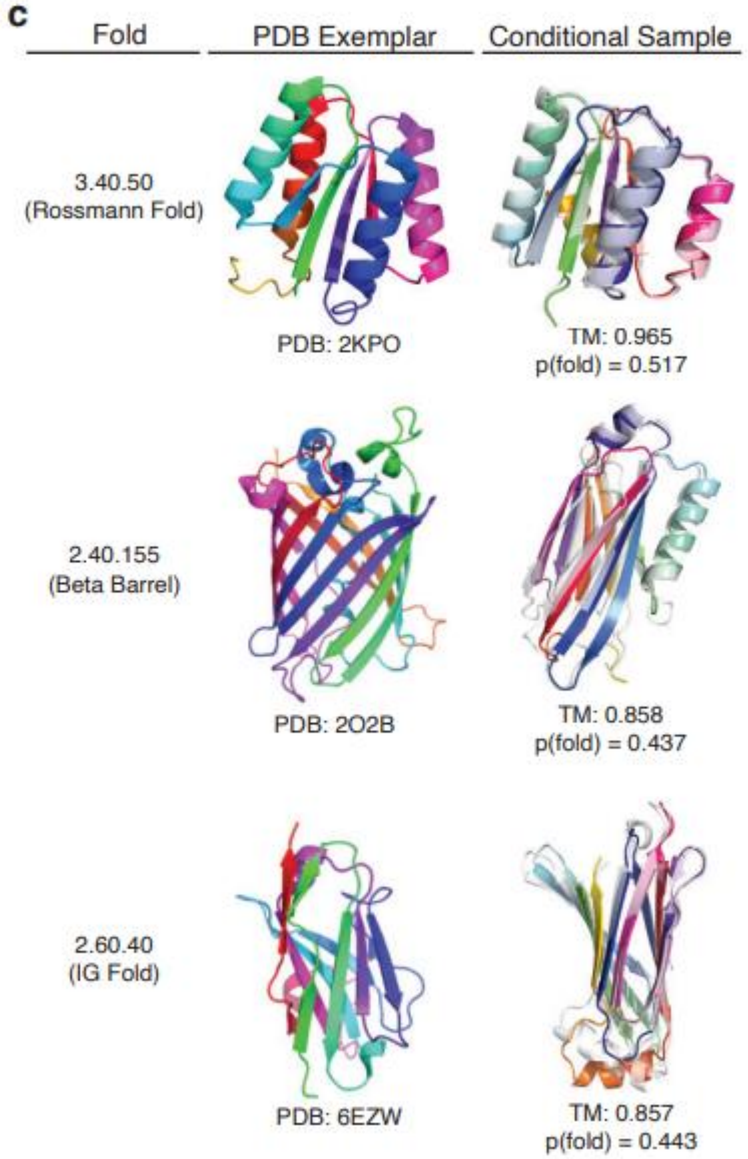






Conditional generation: Chroma

- **Protein class condition** (classifier guidance[11])
- $p(y|x_t)$: protein structure classifier
  - y: class label
  - x_t: noised backbone coords in the diffusion process
- Implementation of $p(y|x_t)$
  - GNN with classifier head

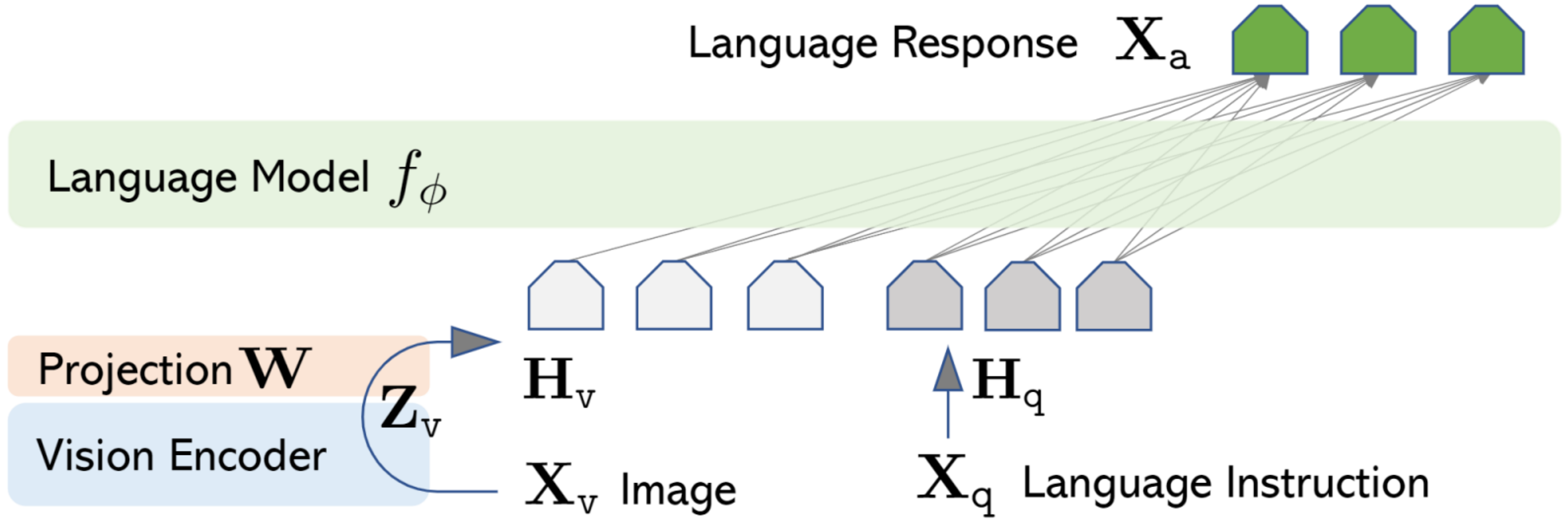


Conditional generation: Chroma

- **Text guidance**
- $p(y|x_t)$: backbone caption model
  - y: text description
  - x_t: noised backbone coords in the diffusion process
- Implementation of $p(y|x_t)$
  - A **multimodal language model**[12]
  - Structure encoder: a GNN pretrained on classification tasks
  - Language model: GPT-Neo[13], pretrained on arXiv, PubMed etc.
- Training data of $p(y|x_t)$: UniProt, PDB (protein caption data)
- Guidance mechanism
  - Use gradient $\nabla_{x_t} \log p(y|x_t)$ to guide denoising of x_t

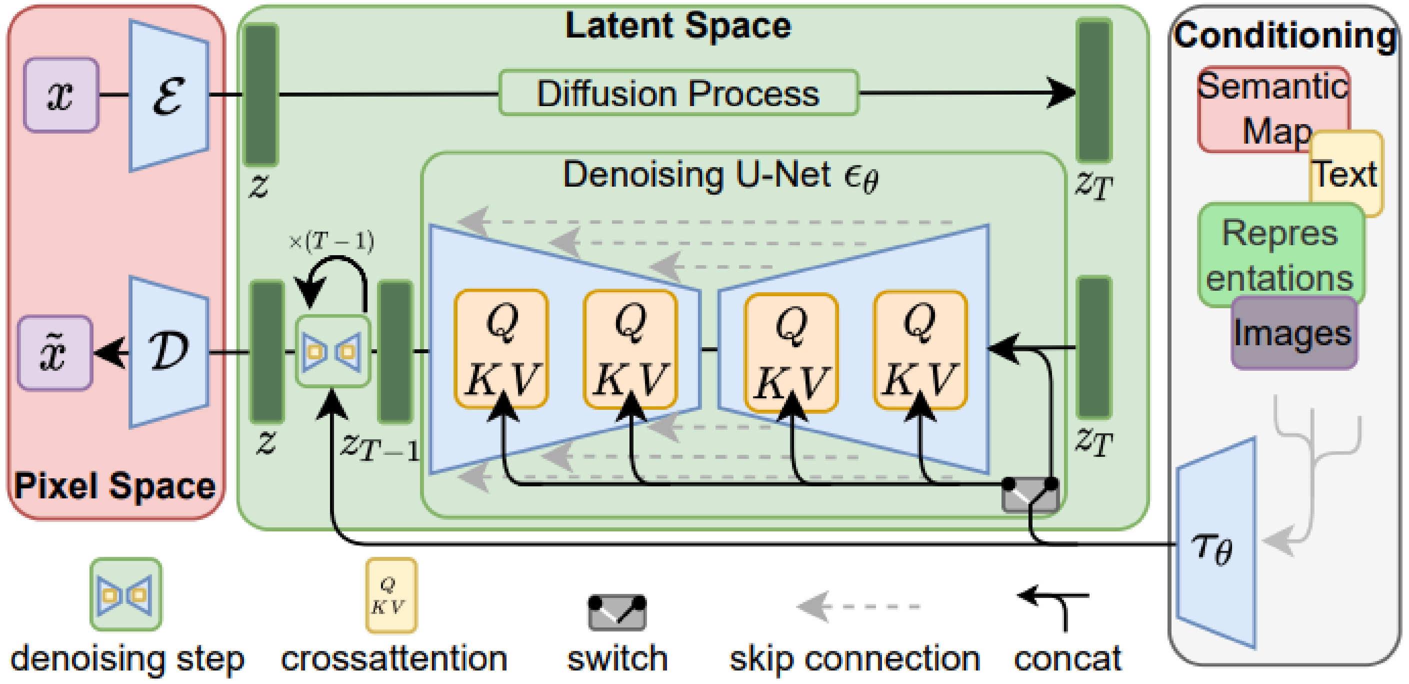
Conditional generation: Chroma

- **Text guidance**
- $p(y|x_t)$: backbone caption model
  - y: text description
  - x_t: noised backbone coords in the diffusion process
- Example of multimodal LLM: LLaVA[12]



Conditional generation: Chroma

- **Text guidance** (different from SD)
- Stable diffusion text guidance
  - Uses cross-attention guidance, instead of external gradient



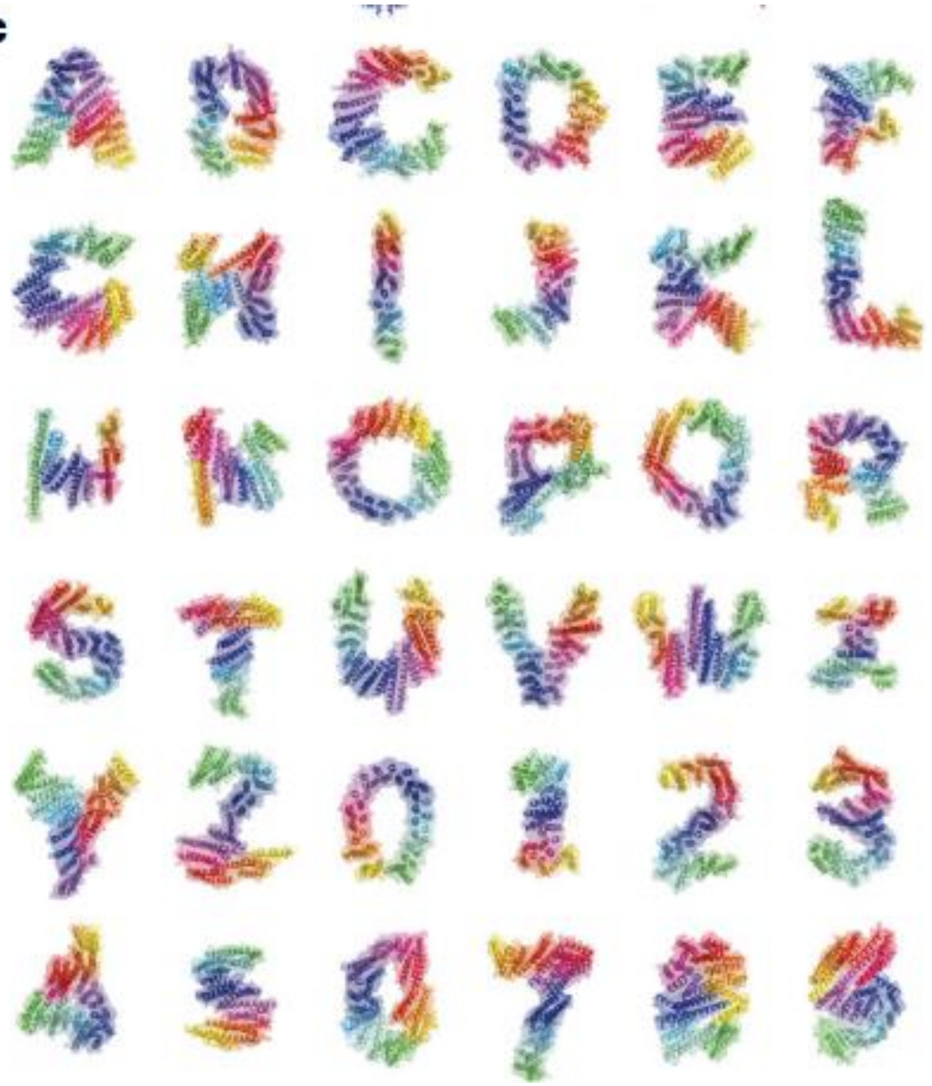
Conditional generation: Chroma

- **Shape guidance**
- Shape condition as point cloud
- $p(y|x_t)$: differentiable shape loss
  - N residues x_t, M points $y = r$
  - Wasserstein distance $K^W \in \mathbb{R}^{N \times M}$
    - Compare x_t and r inter-atomic distance
  - Gromov-Wasserstein distance $K^{GW} \in \mathbb{R}^{N \times M}$
    - Compare respective internal distances of x_t and r
    - More suitable for unaligned distribution in different space

$$\text{ShapeLoss}(\mathbf{x}, \mathbf{r}) = \sum_{i,j} \left(K_{ij}^{\text{GW}} + K_{ij}^{\text{W}}(\mathbf{x}, \mathbf{r}) \right) \|\mathbf{x}_i - \mathbf{r}_j\|,$$

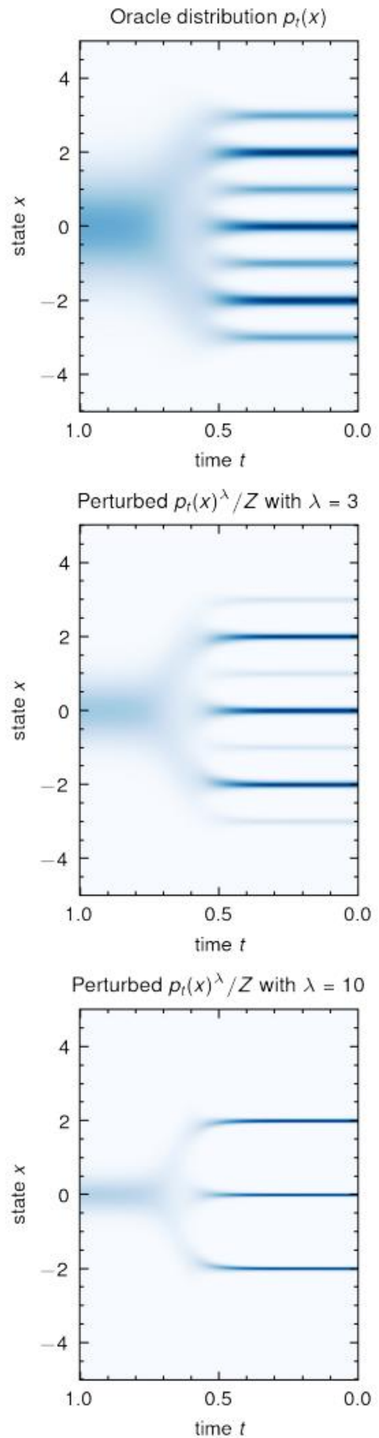
inter-atomic distance

c



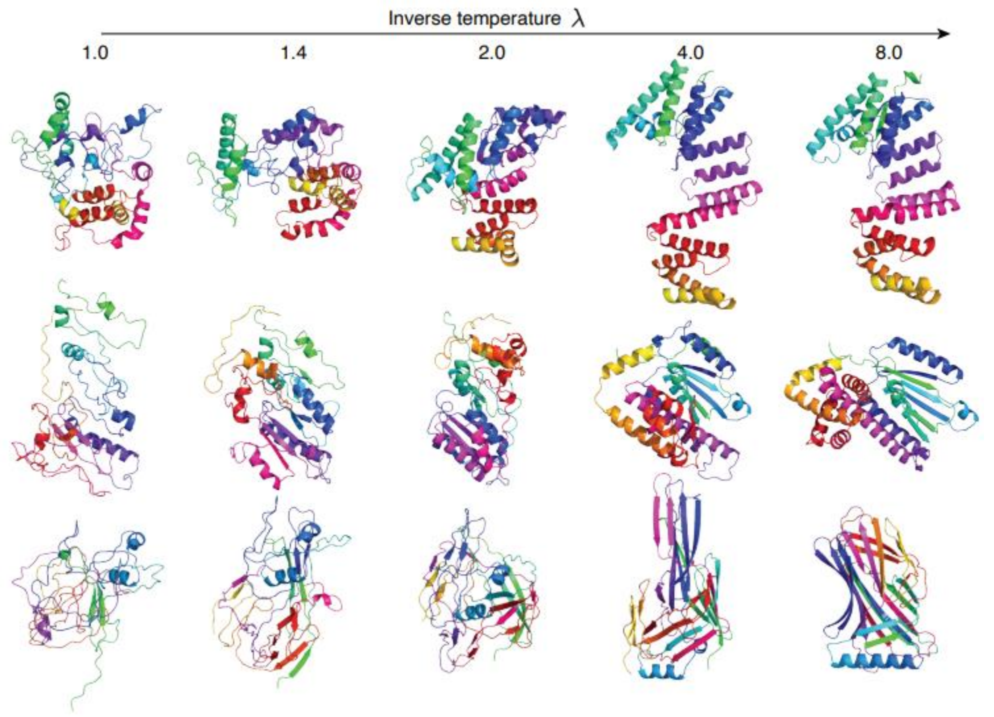
Low temperature sampling

- Goal: generate proteins with higher likelihood
- Motivation: higher likelihood usually means higher quality
- Temperature: trade-off between **quality and diversity**
  - higher temp → higher variance, more diversified output
  - lower temp → more concentrated
  - Analogy: greedy decoding, top-p sampling of LLM
- How
  - Scaling reverse process distribution $p_\theta(x_{t-1}|x_t)$
  - Classifier-free guidance[14]

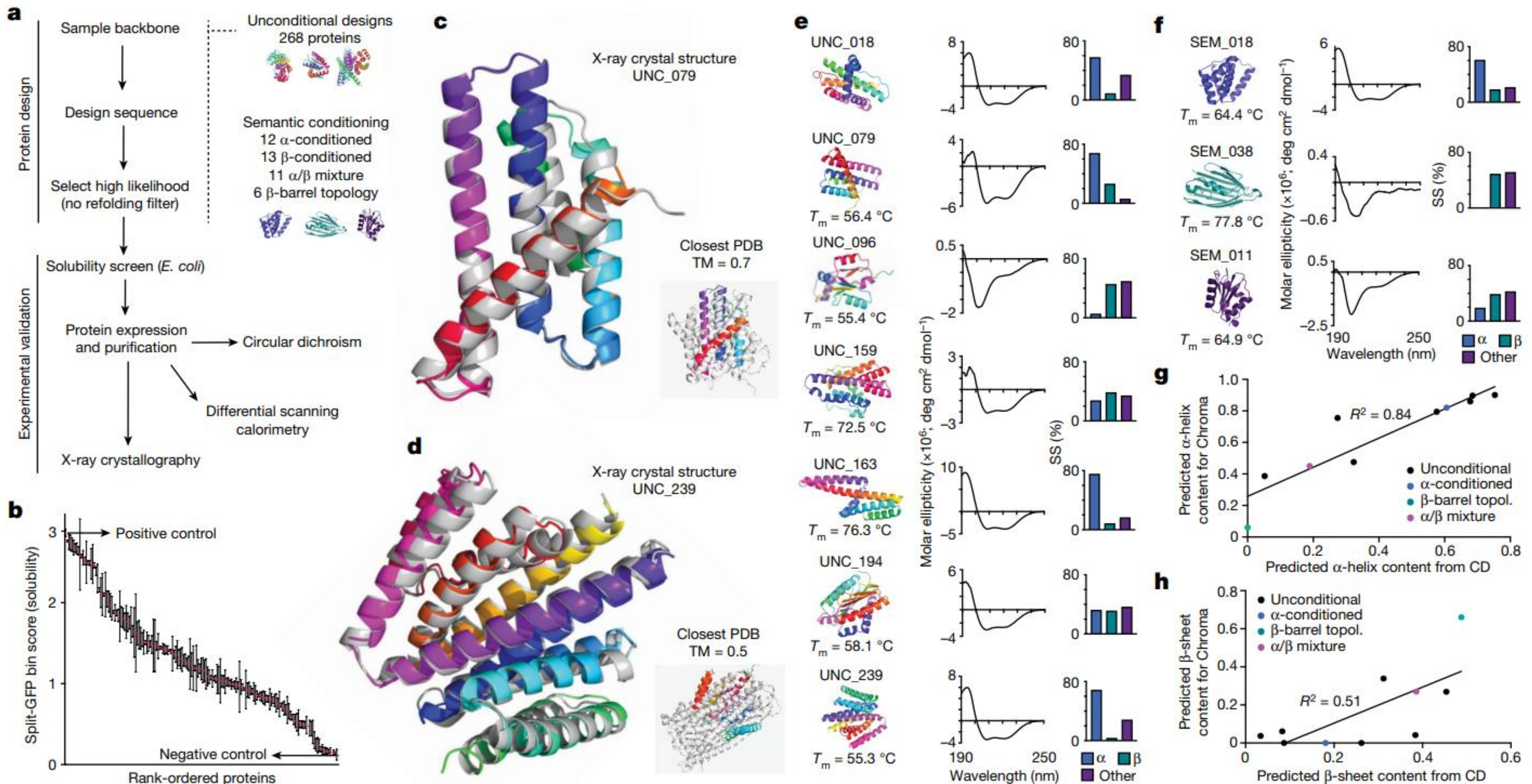


Low temperature sampling

- Scaling $p_\theta(x_{t-1}|x_t) \rightarrow \frac{1}{Z(\lambda)} p_\theta(x_{t-1}|x_t)^\lambda$
  - λ: inverse temperature
- Score is linearly scaled $\nabla_x \log p_\theta(x_{t-1}|x_t) \rightarrow \lambda\, \nabla_x \log p_\theta(x_{t-1}|x_t)$
- Scaling $p_\theta(x_0)$ is more accurate



Wet lab experiments



References

[1] Ingraham, J.B., Baranov, M., Costello, Z. *et al.* Illuminating protein space with a programmable generative model. *Nature* **623**, 1070–1078 (2023).

[2] Jumper, J., Evans, R., Pritzel, A. *et al.* Highly accurate protein structure prediction with AlphaFold. *Nature* **596**, 583–589 (2021).

[3] J. Dauparas *et al.*, Robust deep learning–based protein sequence design using ProteinMPNN. *Science* **378**,49-56(2022).

[4] Ho, J., Jain, A., Abbeel, P., Denoising Diffusion Probabilistic Models, *arXiv preprint,* 2006.11239

[5] Robin, R., Andreas, B. Dominik, L. *et al*. High-Resolution Image Synthesis with Latent Diffusion Models, *arXiv preprint,* 2112.10752

[6] Chin-Wei H.,Milad A., Avishek, B. *et al*. Riemannian Diffusion Models, *arXiv preprint,* 2208.07949

[7] Alexander Quinn Nichol, Prafulla Dhariwal, Improved Denoising Diffusion Probabilistic Models, Proceedings of the 38th International Conference on Machine Learning, PMLR 139:8162-8171, 2021

[8] Yang, S., Jascha S., Diederik K., Score-Based Generative Modeling through Stochastic Differential Equations}, International Conference on Learning Representations, 2021}

[9] Wu, K.E., Yang, K.K., van den Berg, R. *et al.* Protein structure generation via folding diffusion. *Nat Commun* **15**, 1059 (2024).

[10] Hyungjin, C., Jeongsol, K., Michael, T.M. *et al*. Diffusion Posterior Sampling for General Noisy Inverse Problems}, The Eleventh International Conference on Learning Representations, 2023

[11] Prafulla Dhariwal and Alex Nichol. Diffusion models beat GANs on image synthesis. *arXiv preprint,* 2105.05233

[12] Liu, H., Li, C., Wu, Q. and Lee, Y., Visual Instruction Tuning, NeurIPS, 2023

[13] Black, S., Leo, G., Wang, P., Leahy, C., & Biderman, S. (2021). GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow (1.0). Zenodo.

[14] Ho, J., Tim Salimans Classifier-Free Diffusion Guidance, *arXiv preprint,* 2207.12598

[15] Yang Song, and Stefano Ermon. Generative Modeling by Estimating Gradients of the Data Distribution. NeurIPS 2019

[16] P. Vincent. A connection between score matching and denoising autoencoders. Neural computation, 23(7):1661–1674, 2011.