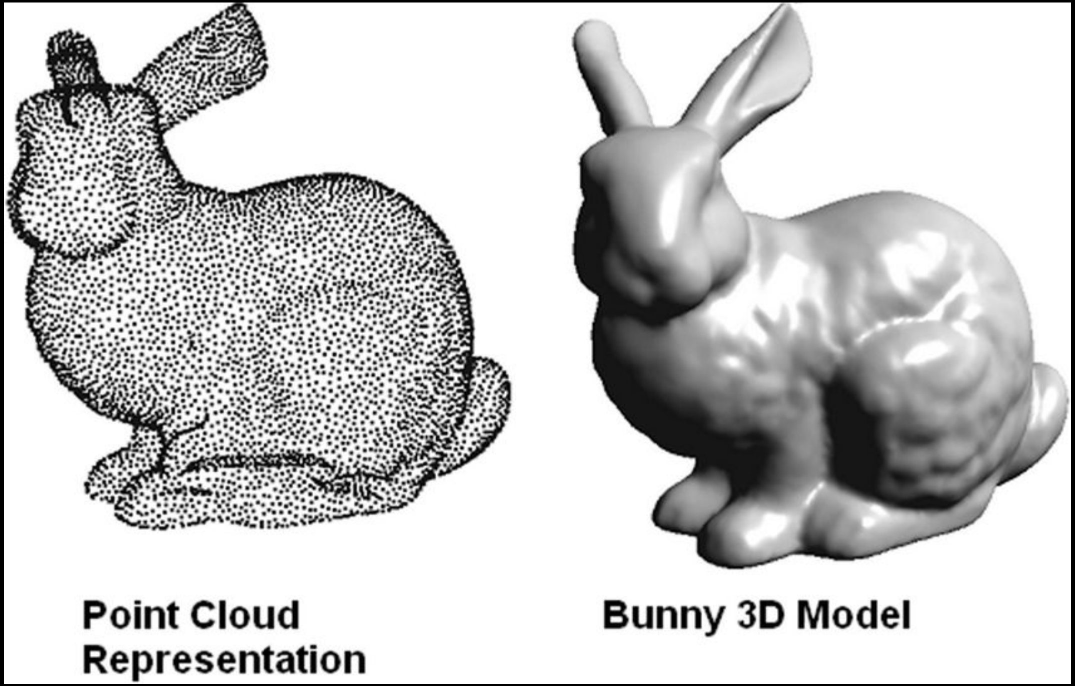
# Illuminating protein space with a programmable generative model

Paper Presentation
COS597N, Fall ‘23
November 16, 2023
Alkin Kaz

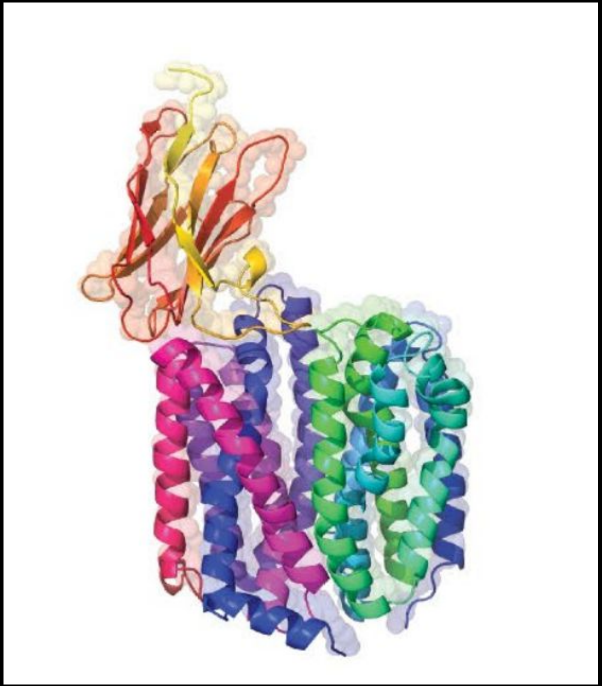
Generate:*Chroma*

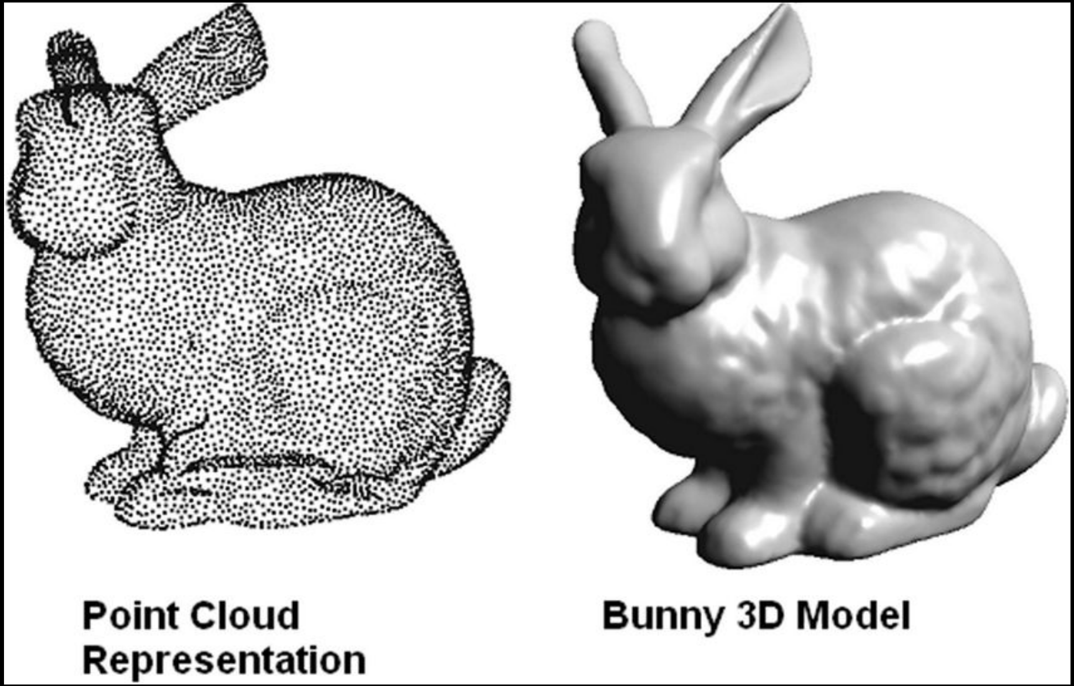
## Prologue — The Stanford Bunny



(Image taken from https://doi.org/10.1109/TIFS.2008.2011081)

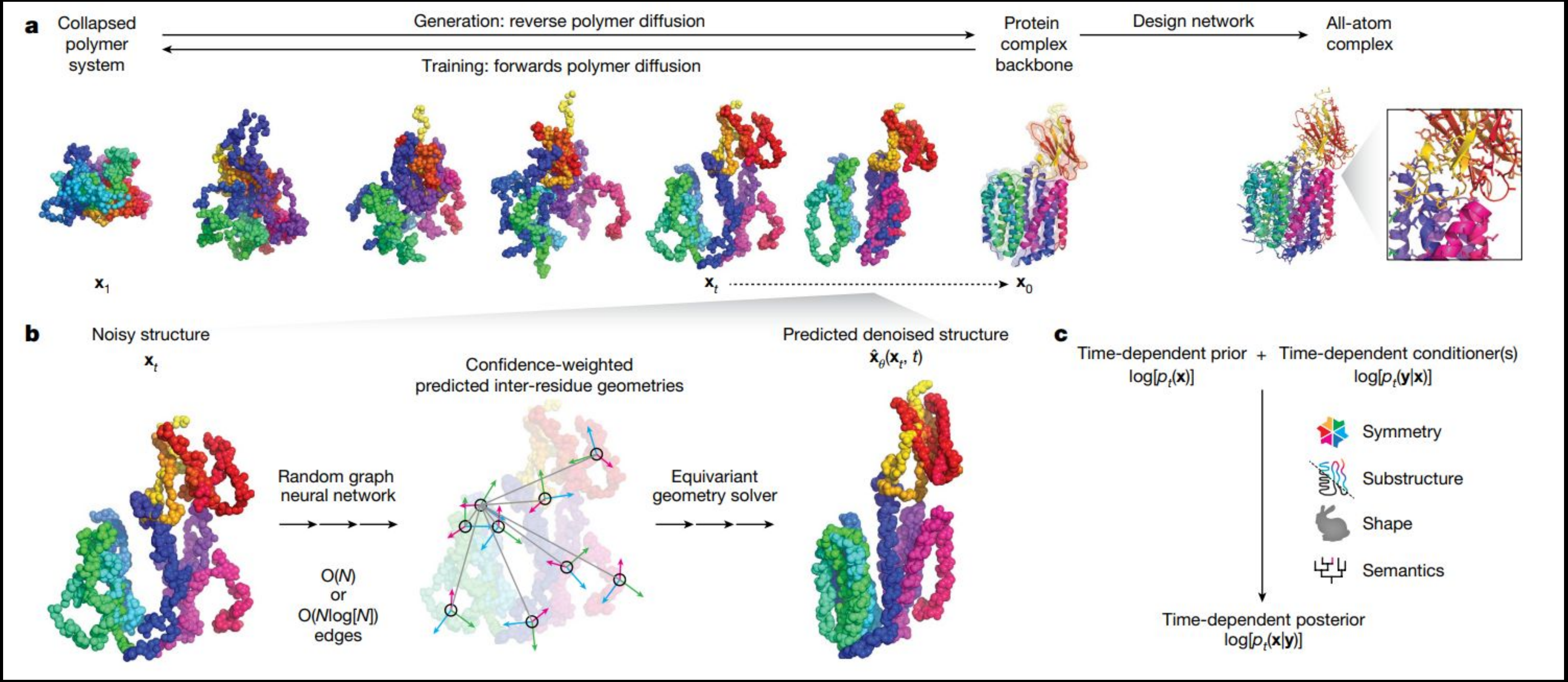
## Prologue — The Stanford Bunny





(Image taken from https://doi.org/10.1109/TIFS.2008.2011081)

# Introducing... Generate:*Chroma*



a
Collapsed polymer system
Generation: reverse polymer diffusion
Training: forwards polymer diffusion
Protein complex backbone
Design network
All-atom complex
$\mathbf{x}_1$
$\mathbf{x}_t$
$\mathbf{x}_0$
b
Noisy structure
$\mathbf{x}_t$
Random graph neural network
O(N) or O(Nlog[N]) edges
Confidence-weighted predicted inter-residue geometries
Equivariant geometry solver
Predicted denoised structure
$\hat{\mathbf{x}}_\theta(\mathbf{x}_t, t)$
c
Time-dependent prior $\log[p_t(\mathbf{x})]$
+
Time-dependent conditioner(s) $\log[p_t(\mathbf{y}|\mathbf{x})]$
Symmetry
Substructure
Shape
Semantics
Time-dependent posterior $\log[p_t(\mathbf{x}|\mathbf{y})]$

## Outline

## Motivation

- ➔ Modeling the joint, all-atom likelihood of sequences and three-dimensional structures of full protein complexes,
- ➔ Achieving this with computation that scales sub-quadratically with the size of the protein system,
- ➔ Enabling conditional sampling under diverse design constraints without retraining.

## Background

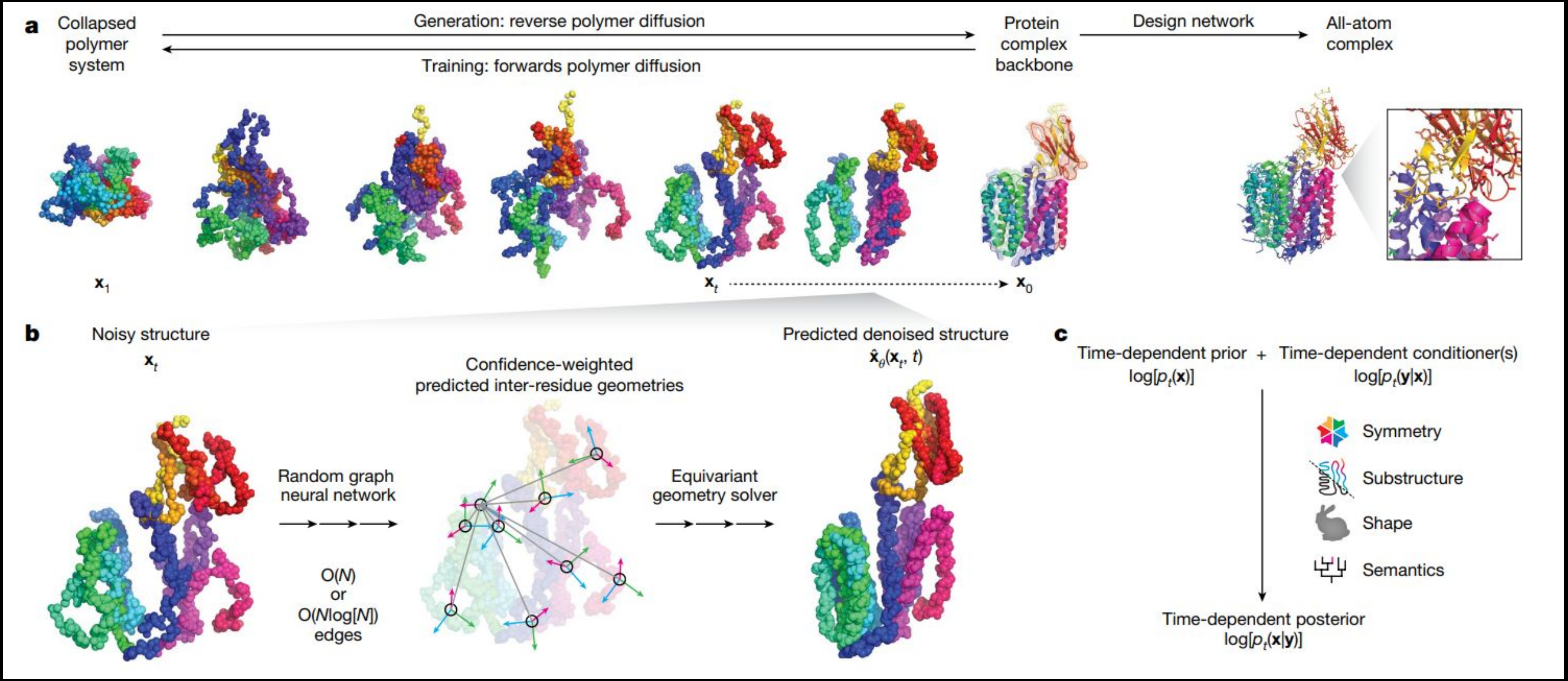
- Inverse Design:
  - From: Backbone Coordinates
  - To: Sequence
- Diffusion Models:
  - Photorealistic images at inference time
  - Can introduce multi-modal constraints

## Research Question

*Can we introduce a diffusive model for protein design that can design large complexes with conditions given at inference time?*

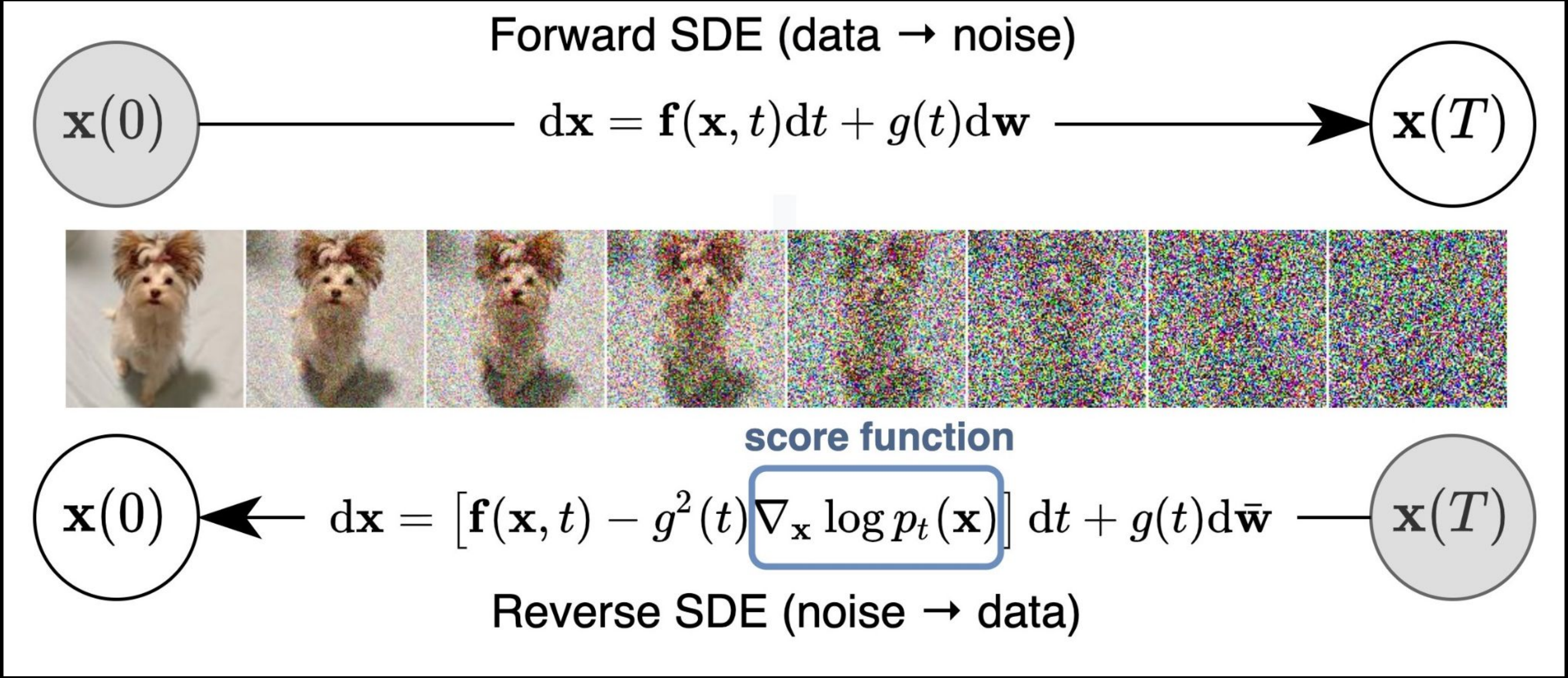
**YES!**

# Method: Overview


a
Collapsed polymer system
Generation: reverse polymer diffusion
Training: forwards polymer diffusion
Protein complex backbone
Design network
All-atom complex
$\mathbf{x}_1$
$\mathbf{x}_t$
$\mathbf{x}_0$
b
Noisy structure
$\mathbf{x}_t$
Random graph neural network
O($N$) or O($N$log[$N$]) edges
Confidence-weighted predicted inter-residue geometries
Equivariant geometry solver
Predicted denoised structure
$\hat{\mathbf{x}}_\theta(\mathbf{x}_t, t)$
c
Time-dependent prior
$\log[p_t(\mathbf{x})]$
+
Time-dependent conditioner(s)
$\log[p_t(\mathbf{y}|\mathbf{x})]$
Symmetry
Substructure
Shape
Semantics
Time-dependent posterior
$\log[p_t(\mathbf{x}|\mathbf{y})]$

(Nature paper, Fig. 1)

## Method: Diffusion Primer


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

(https://yang-song.net/blog/2021/score/)

# Method: Correlated Diffusion

Forward-Time SDE

$$\mathrm{d}\mathbf{x} = -\frac{1}{2}\beta_t \mathbf{x}\,\mathrm{d}t + \sqrt{\beta_t}\mathbf{R}\,\mathrm{d}\mathbf{w}.$$

Reverse-Time SDE

$$\begin{aligned}\mathrm{d}\mathbf{x} &= \left(-\frac{1}{2}\,\mathbf{x} - \mathbf{R}\mathbf{R}^\intercal \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right)\beta_t\ \mathrm{d}t + \sqrt{\beta_t}\ \mathbf{R}\ \mathrm{d}\bar{\mathbf{w}}.\\ &= \left(\frac{\alpha_t + 1}{2(1-\alpha_t^2)}\mathbf{x} - \frac{\alpha_t}{1-\alpha_t^2}\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{x}, t)\right)\beta_t\ \mathrm{d}t + \sqrt{\beta_t}\ \mathbf{R}\ \mathrm{d}\bar{\mathbf{w}}.\end{aligned}$$

Reverse-Time SDE (Conditional on Auxiliary Constraints)

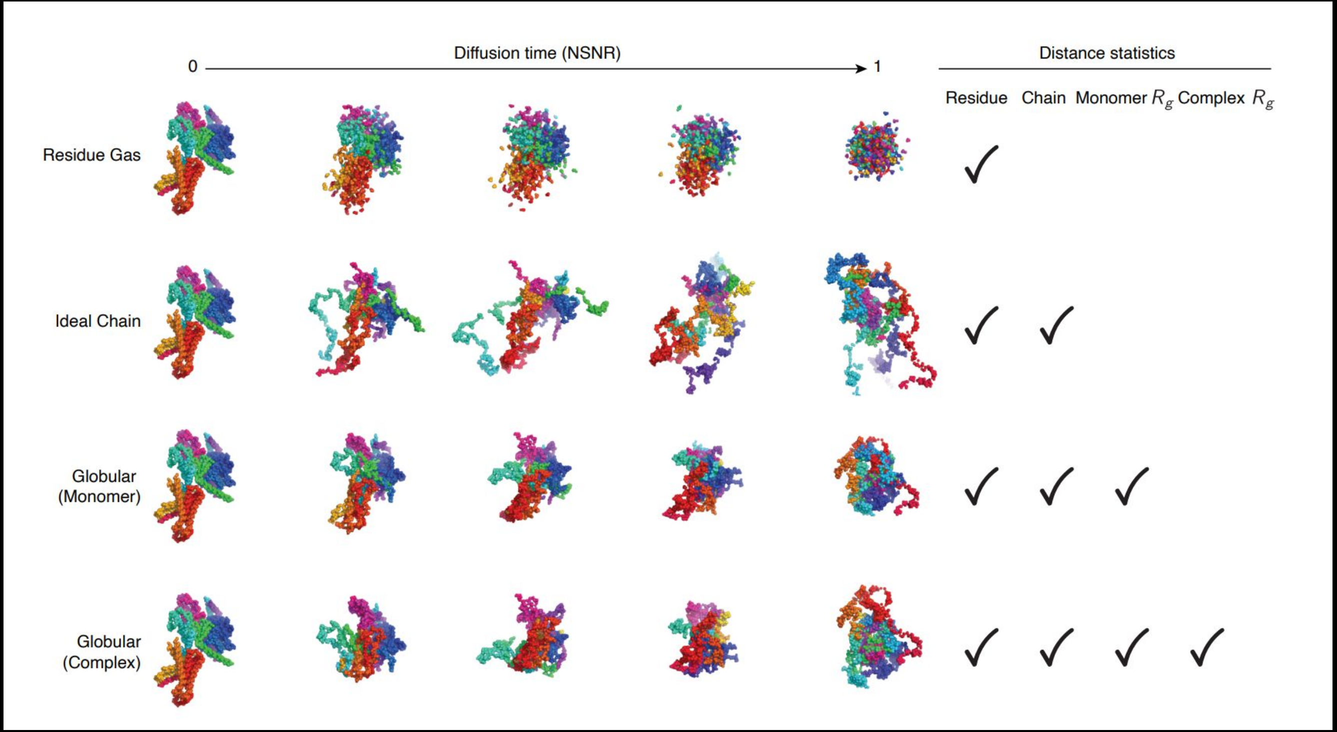
$$\begin{aligned}\mathrm{d}\mathbf{x} &= \left(-\frac{1}{2}\,\mathbf{x} - \mathbf{R}\mathbf{R}^\intercal \left(\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) + \nabla_{\mathbf{x}} \log p_t(\mathbf{y}|\mathbf{x})\right)\right)\beta_t\ \mathrm{d}t + \sqrt{\beta_t}\ \mathbf{R}\ \mathrm{d}\bar{\mathbf{w}}\\ &= \left(\frac{\alpha_t + 1}{2(1-\alpha_t^2)}\mathbf{x} - \frac{\alpha_t}{1-\alpha_t^2}\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{x}, t) - \mathbf{R}\mathbf{R}^\intercal \nabla_{\mathbf{x}} \log p_t(\mathbf{y}|\mathbf{x})\right)\beta_t\ \mathrm{d}t + \sqrt{\beta_t}\ \mathbf{R}\ \mathrm{d}\bar{\mathbf{w}}.\end{aligned}$$

## Method: Correlated Diffusion

(Hybrid Langevin) Reverse-Time SDE

$$\mathrm{d}\mathbf{x} = h_t\,\mathbf{x} - g_t^2\left(\lambda_t + \frac{\lambda_0\psi}{2}\right)\mathbf{R}\mathbf{R}^\intercal\nabla_{\mathbf{x}_t}\log p_t(\mathbf{x}_t)\,\mathrm{d}t + g_t\sqrt{(1+\psi)}\,\mathbf{R}\,\mathrm{d}\bar{\mathbf{w}}.$$

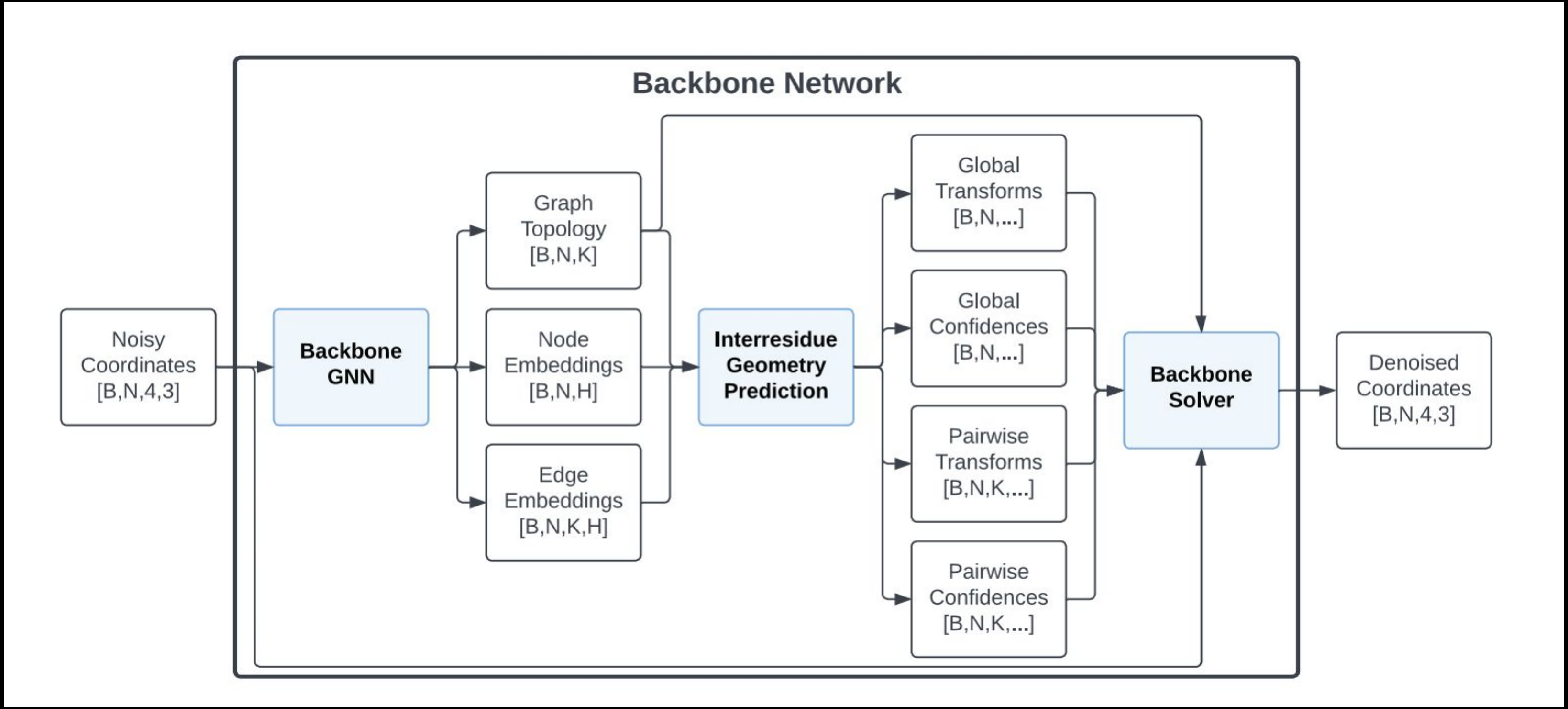
## Method: Polymer-Structured Diffusion



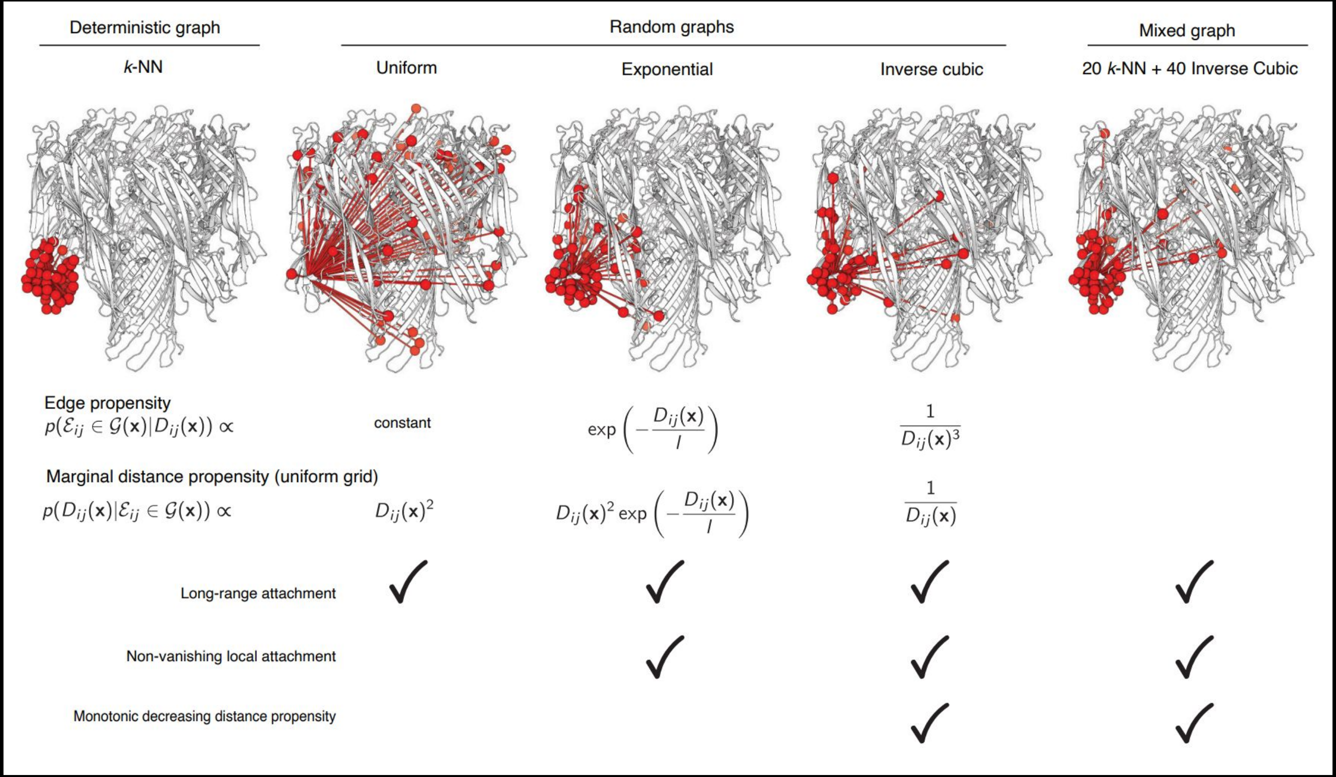
## Method: Chroma Overview

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

# Method: ChromaBackbone

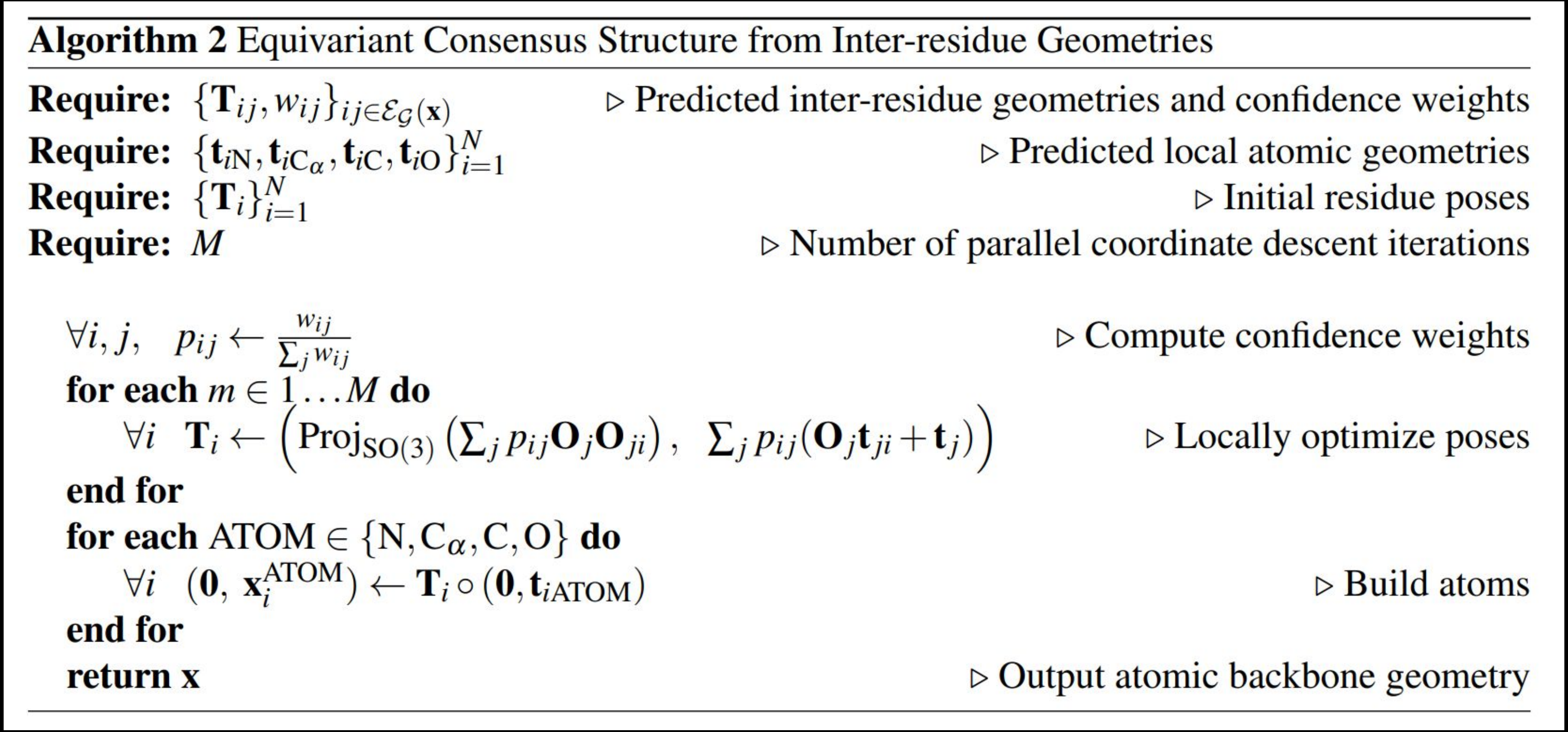

Backbone Network
Noisy Coordinates [B,N,4,3]
Backbone GNN
Graph Topology [B,N,K]
Node Embeddings [B,N,H]
Edge Embeddings [B,N,K,H]
Interresidue Geometry Prediction
Global Transforms [B,N,...]
Global Confidences [B,N,...]
Pairwise Transforms [B,N,K,...]
Pairwise Confidences [B,N,K,...]
Backbone Solver
Denoised Coordinates [B,N,4,3]

# Method: Random Graph Neural Networks



| | Deterministic graph | Random graphs | | | Mixed graph |
|---|---|---|---|---|---|
| | $k$-NN | Uniform | Exponential | Inverse cubic | 20 $k$-NN + 40 Inverse Cubic |
| Edge propensity $p(\mathcal{E}_{ij} \in \mathcal{G}(\mathbf{x}) \mid D_{ij}(\mathbf{x})) \propto$ | | constant | $\exp\left(-\frac{D_{ij}(\mathbf{x})}{l}\right)$ | $\frac{1}{D_{ij}(\mathbf{x})^3}$ | |
| Marginal distance propensity (uniform grid) $p(D_{ij}(\mathbf{x}) \mid \mathcal{E}_{ij} \in \mathcal{G}(\mathbf{x})) \propto$ | | $D_{ij}(\mathbf{x})^2$ | $D_{ij}(\mathbf{x})^2 \exp\left(-\frac{D_{ij}(\mathbf{x})}{l}\right)$ | $\frac{1}{D_{ij}(\mathbf{x})}$ | |
| Long-range attachment | | ✓ | ✓ | ✓ | ✓ |
| Non-vanishing local attachment | | | ✓ | ✓ | ✓ |
| Monotonic decreasing distance propensity | | | | ✓ | ✓ |

## Method: Structure from Inter-Residue Geometry



**Algorithm 2** Equivariant Consensus Structure from Inter-residue Geometries

```
Require: $\{\mathbf{T}_{ij}, w_{ij}\}_{ij \in \mathcal{E}_{\mathcal{G}}(\mathbf{x})}$    ▷ Predicted inter-residue geometries and confidence weights
Require: $\{\mathbf{t}_{i\mathrm{N}}, \mathbf{t}_{i\mathrm{C}_\alpha}, \mathbf{t}_{i\mathrm{C}}, \mathbf{t}_{i\mathrm{O}}\}_{i=1}^{N}$    ▷ Predicted local atomic geometries
Require: $\{\mathbf{T}_i\}_{i=1}^{N}$    ▷ Initial residue poses
Require: $M$    ▷ Number of parallel coordinate descent iterations

  $\forall i, j, \quad p_{ij} \leftarrow \frac{w_{ij}}{\sum_j w_{ij}}$    ▷ Compute confidence weights
  for each $m \in 1 \dots M$ do
      $\forall i \quad \mathbf{T}_i \leftarrow \left(\mathrm{Proj}_{\mathrm{SO}(3)}\left(\sum_j p_{ij}\mathbf{O}_j\mathbf{O}_{ji}\right), \ \sum_j p_{ij}(\mathbf{O}_j\mathbf{t}_{ji} + \mathbf{t}_j)\right)$    ▷ Locally optimize poses
  end for
  for each ATOM $\in \{\mathrm{N}, \mathrm{C}_\alpha, \mathrm{C}, \mathrm{O}\}$ do
      $\forall i \quad (\mathbf{0}, \mathbf{x}_i^{\mathrm{ATOM}}) \leftarrow \mathbf{T}_i \circ (\mathbf{0}, \mathbf{t}_{i\mathrm{ATOM}})$    ▷ Build atoms
  end for
  return $\mathbf{x}$    ▷ Output atomic backbone geometry
```

## Method: ChromaDesign

Potts Decoder (ChromaDesign Potts)

$$p_\theta(\mathbf{s}|\mathbf{x}_t,t) = \frac{1}{Z(\mathbf{x}_t,t,\boldsymbol{\theta})}\exp\left(-\sum_i \mathbf{h}_i(s_i;\mathbf{x}_t,t) - \sum_{i<j}\mathbf{J}_{ij}(s_i,s_j;\mathbf{x}_t,t)\right).$$

Autoregressive Decoder (ChromaDesign Multi)

$$p_\theta(\mathbf{s}|\mathbf{x}_t,t) = \prod_i p_\theta(s_{\pi_i}|s_{\pi_{i-1}},\ldots,s_{\pi_1},\mathbf{x}_t,t).$$

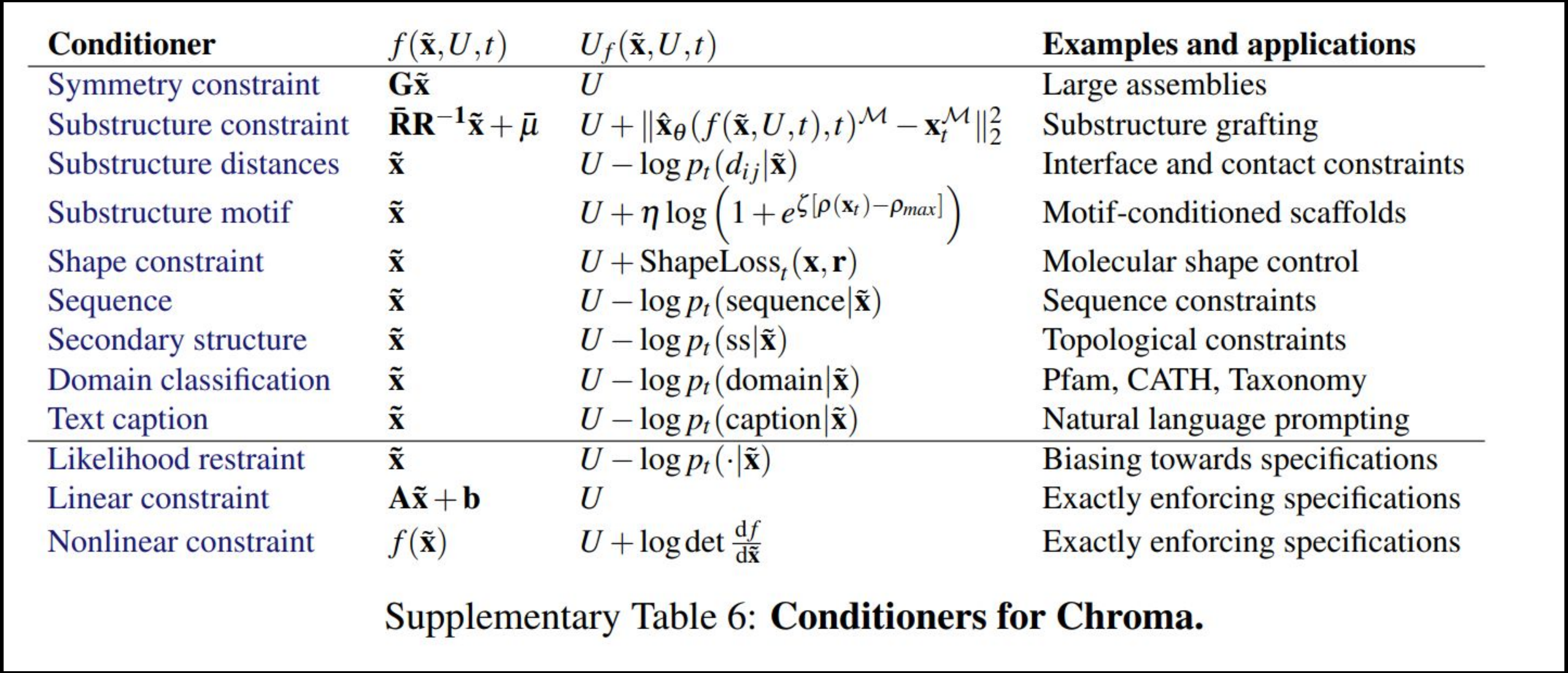
## Method: Conditioners

$$\mathrm{d}\tilde{\mathbf{x}} = -\frac{\beta_t \psi}{2} \lambda_t \mathbf{R}\mathbf{R}^\top \nabla_{\tilde{\mathbf{x}}} U(f(\tilde{\mathbf{x}}_t); \mathbf{y}, t)\, \mathrm{d}t + \sqrt{\beta_t \psi}\, \mathbf{R}\, \mathrm{d}\bar{\mathbf{w}}.$$

**RECALL:**
Reverse-Time SDE (Non-Langevin, Conditional on Auxiliary Constraints)

$$\begin{aligned}
\mathrm{d}\mathbf{x} &= \left(-\frac{1}{2}\,\mathbf{x} - \mathbf{R}\mathbf{R}^\top \left(\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) + \nabla_{\mathbf{x}} \log p_t(\mathbf{y}|\mathbf{x})\right)\right) \beta_t\, \mathrm{d}t + \sqrt{\beta_t}\, \mathbf{R}\, \mathrm{d}\bar{\mathbf{w}} \\
&= \left(\frac{\alpha_t + 1}{2(1-\alpha_t^2)}\mathbf{x} - \frac{\alpha_t}{1-\alpha_t^2}\hat{\mathbf{x}}_\theta(\mathbf{x}, t) - \mathbf{R}\mathbf{R}^\top \nabla_{\mathbf{x}} \log p_t(\mathbf{y}|\mathbf{x})\right) \beta_t\, \mathrm{d}t + \sqrt{\beta_t}\, \mathbf{R}\, \mathrm{d}\bar{\mathbf{w}}.
\end{aligned}$$

# Method: Conditioners



| Conditioner | $f(\tilde{\mathbf{x}}, U, t)$ | $U_f(\tilde{\mathbf{x}}, U, t)$ | Examples and applications |
|---|---|---|---|
| Symmetry constraint | $\mathbf{G}\tilde{\mathbf{x}}$ | $U$ | Large assemblies |
| Substructure constraint | $\bar{\mathbf{R}}\mathbf{R}^{-1}\tilde{\mathbf{x}} + \bar{\mu}$ | $U + \lVert \hat{\mathbf{x}}_\theta(f(\tilde{\mathbf{x}}, U, t), t)^{\mathcal{M}} - \mathbf{x}_t^{\mathcal{M}} \rVert_2^2$ | Substructure grafting |
| Substructure distances | $\tilde{\mathbf{x}}$ | $U - \log p_t(d_{ij} \mid \tilde{\mathbf{x}})$ | Interface and contact constraints |
| Substructure motif | $\tilde{\mathbf{x}}$ | $U + \eta \log\left(1 + e^{\zeta[\rho(\mathbf{x}_t) - \rho_{max}]}\right)$ | Motif-conditioned scaffolds |
| Shape constraint | $\tilde{\mathbf{x}}$ | $U + \text{ShapeLoss}_t(\mathbf{x}, \mathbf{r})$ | Molecular shape control |
| Sequence | $\tilde{\mathbf{x}}$ | $U - \log p_t(\text{sequence} \mid \tilde{\mathbf{x}})$ | Sequence constraints |
| Secondary structure | $\tilde{\mathbf{x}}$ | $U - \log p_t(\text{ss} \mid \tilde{\mathbf{x}})$ | Topological constraints |
| Domain classification | $\tilde{\mathbf{x}}$ | $U - \log p_t(\text{domain} \mid \tilde{\mathbf{x}})$ | Pfam, CATH, Taxonomy |
| Text caption | $\tilde{\mathbf{x}}$ | $U - \log p_t(\text{caption} \mid \tilde{\mathbf{x}})$ | Natural language prompting |
| Likelihood restraint | $\tilde{\mathbf{x}}$ | $U - \log p_t(\cdot \mid \tilde{\mathbf{x}})$ | Biasing towards specifications |
| Linear constraint | $\mathbf{A}\tilde{\mathbf{x}} + \mathbf{b}$ | $U$ | Exactly enforcing specifications |
| Nonlinear constraint | $f(\tilde{\mathbf{x}})$ | $U + \log\det \frac{\mathrm{d}f}{\mathrm{d}\tilde{\mathbf{x}}}$ | Exactly enforcing specifications |

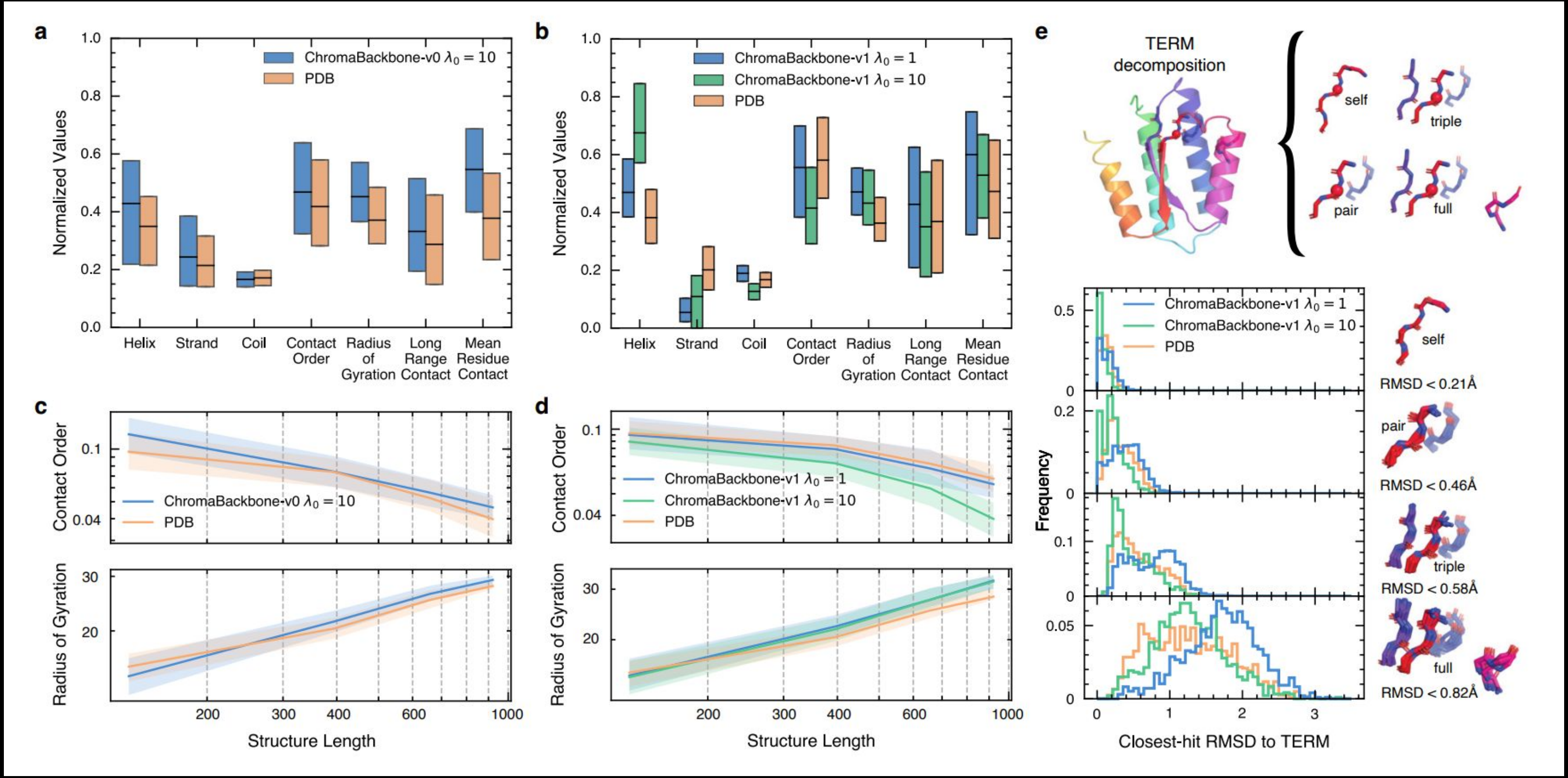
Supplementary Table 6: **Conditioners for Chroma.**

## Method: Conditioners

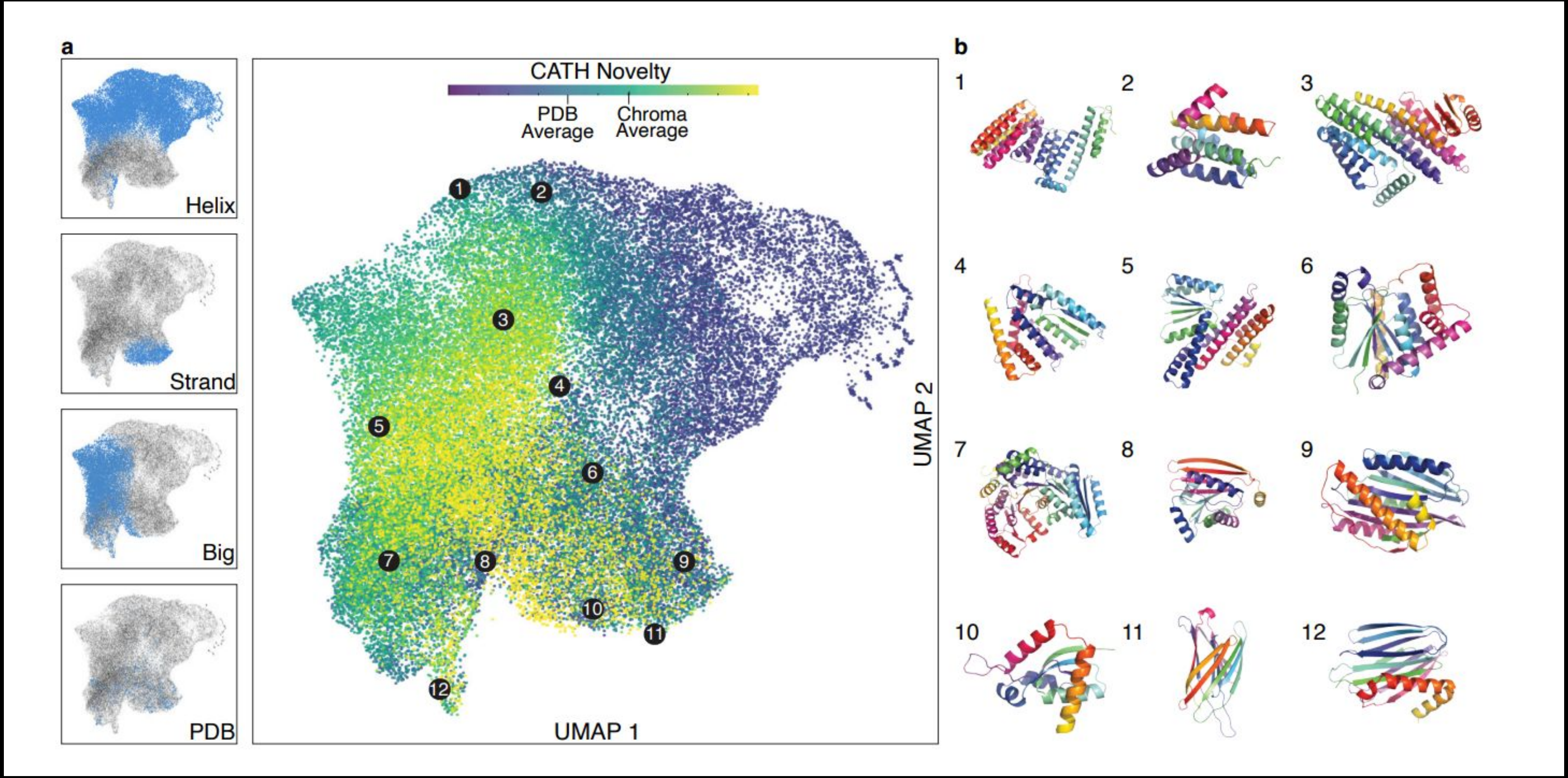
Further networks were developed:

- ProClass:
  - Trained to label/classify with CATH, PFAM labels.
- ProCap:
  - GPT-Neo 125M, trained on Pile (articles from arXiv and PubMed).
  - Fine-tuned with annotations on PDB.

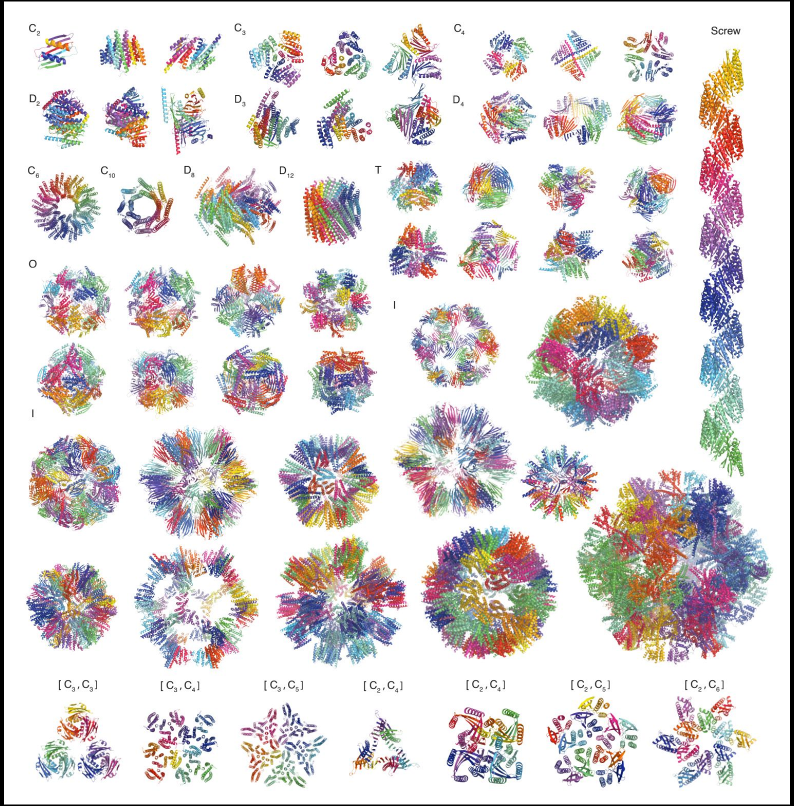
## Results: Reproducing Natural Statistics



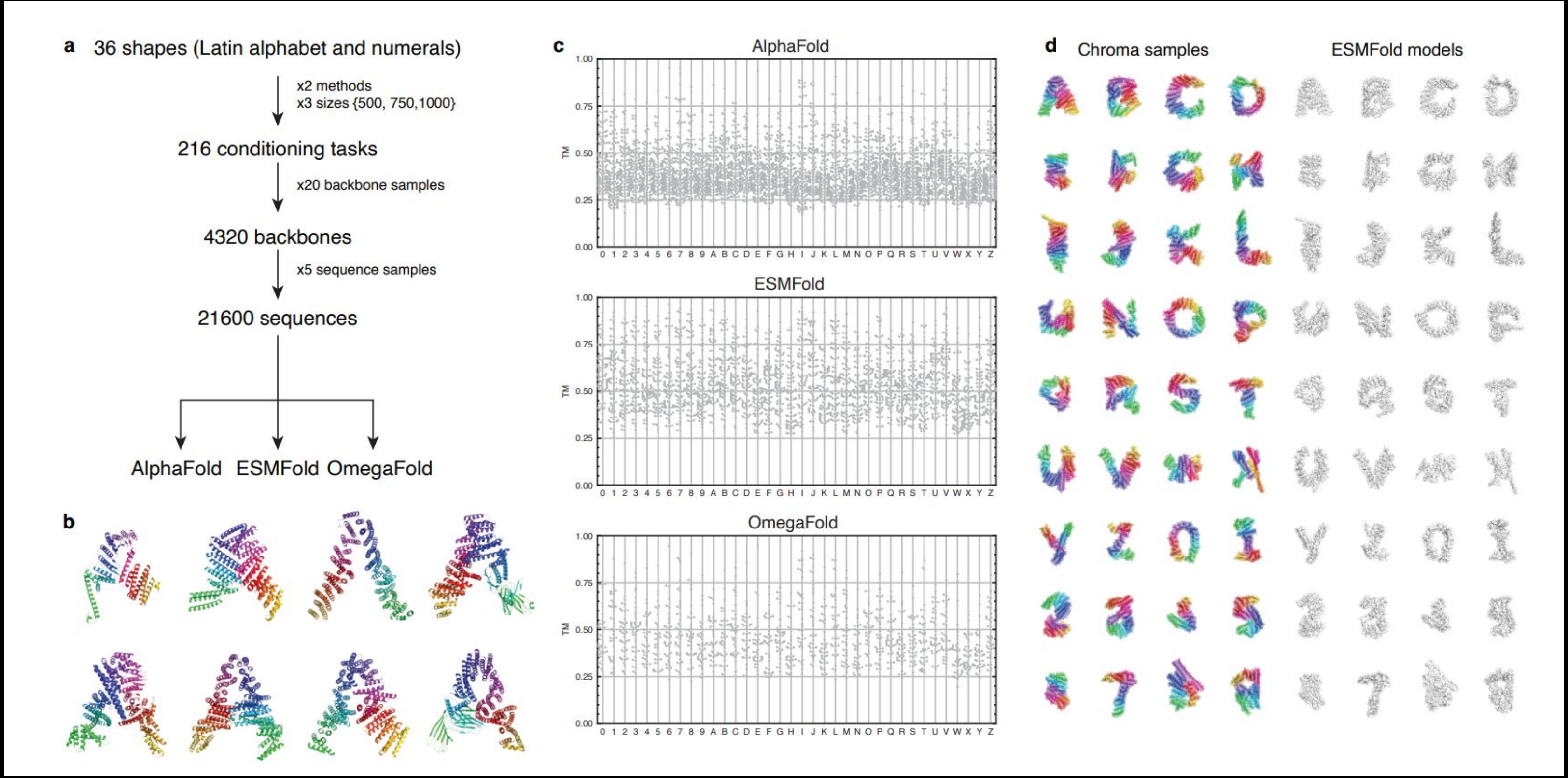
# Results: Novelty



# Results: Conditional Generation



# Results: Conditional Generation and Re-Folding



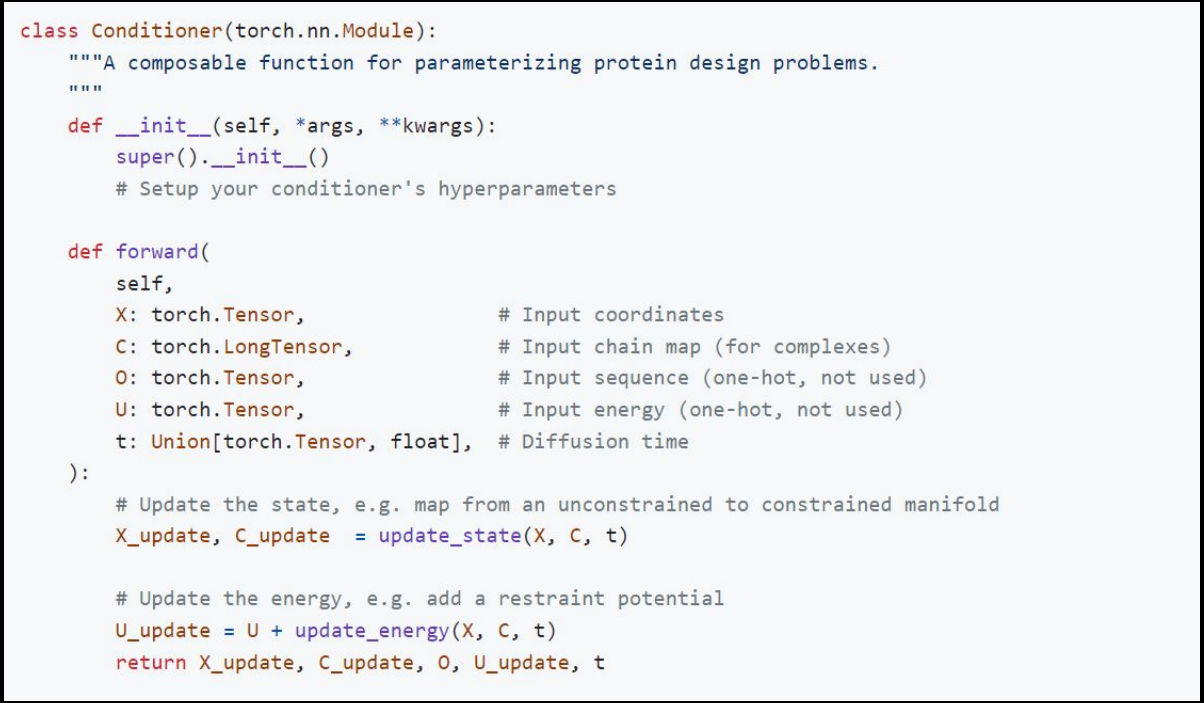
## Conclusions

The hypothesized model is possible. The model can…

- ➔ Generate >3,000 residues in a few minutes on a GPU!
- ➔ Condition on a variety of modalities!
- ➔ Explore novel but feasible designs!

## Future Directions: Try It!

https://github.com/generatebio/chroma



```
class Conditioner(torch.nn.Module):
    """A composable function for parameterizing protein design problems.
    """
    def __init__(self, *args, **kwargs):
        super().__init__()
        # Setup your conditioner's hyperparameters

    def forward(
        self,
        X: torch.Tensor,                   # Input coordinates
        C: torch.LongTensor,               # Input chain map (for complexes)
        O: torch.Tensor,                   # Input sequence (one-hot, not used)
        U: torch.Tensor,                   # Input energy (one-hot, not used)
        t: Union[torch.Tensor, float],     # Diffusion time
    ):
        # Update the state, e.g. map from an unconstrained to constrained manifold
        X_update, C_update  = update_state(X, C, t)

        # Update the energy, e.g. add a restraint potential
        U_update = U + update_energy(X, C, t)
        return X_update, C_update, O, U_update, t
```

## Discussion Questions

How can Chroma be used?

What useful constraints do you see being designed?