EPFL

Graph Generative Models

Alireza Gargoori Motlagh

Graph Representations for Biology and Medicine - EE626

Background

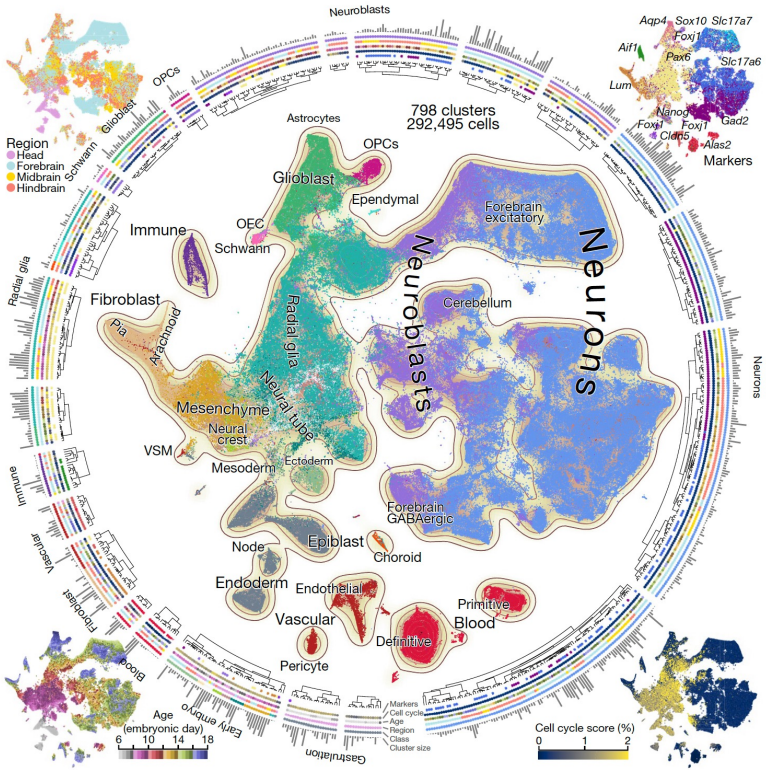
BSc:
Electrical Engineering @ Sharif University of Technology

1st Year PhD Student:
Computational and Quantitative Biology (EDCB) @ EPFL

la manno lab
Laboratory of Brain Development and Biological Data Science

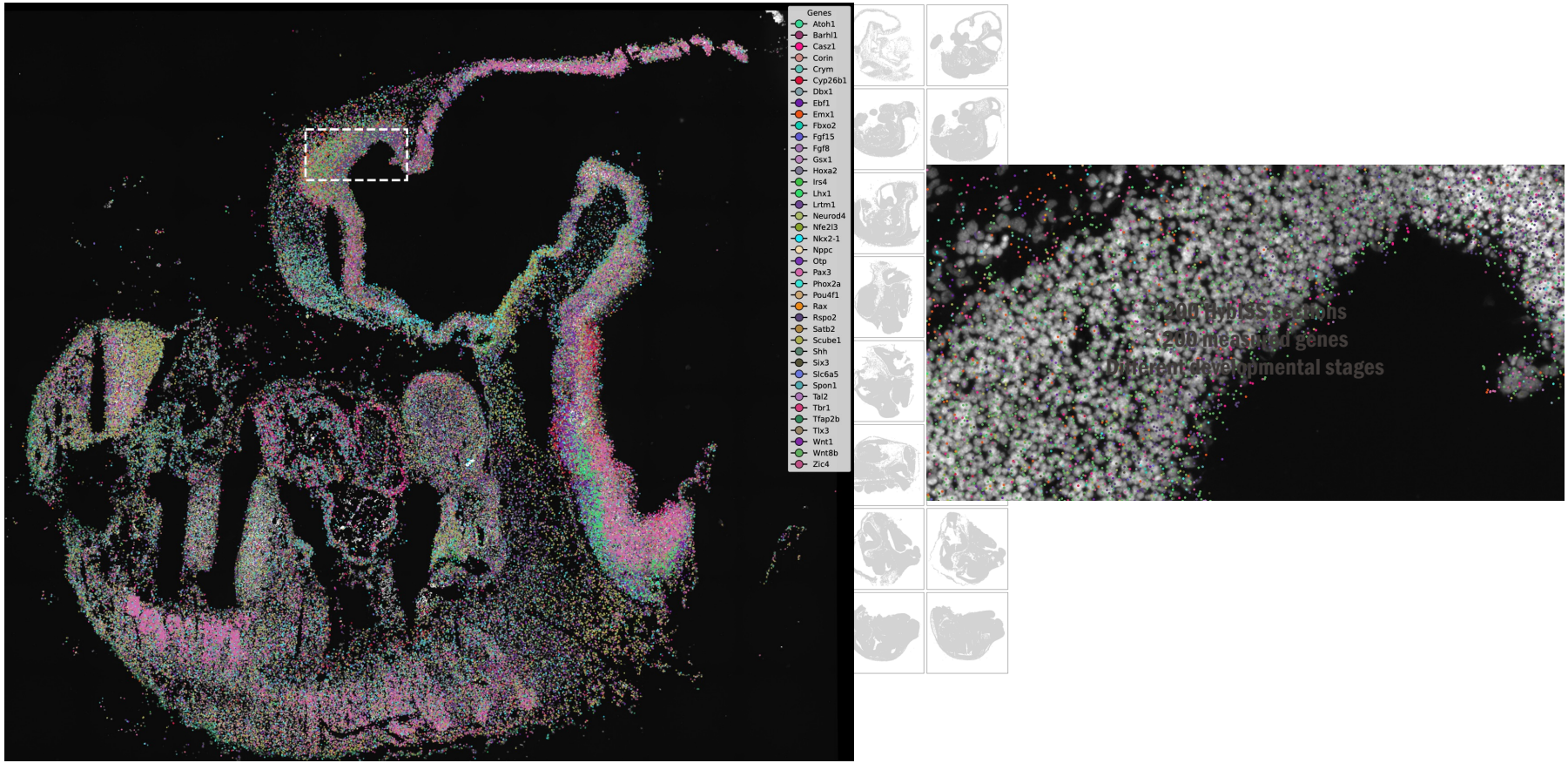
- ML-based method development for Single-Cell and Spatial Omics data

From Single-cell Atlas to Spatio-temporal Mapping

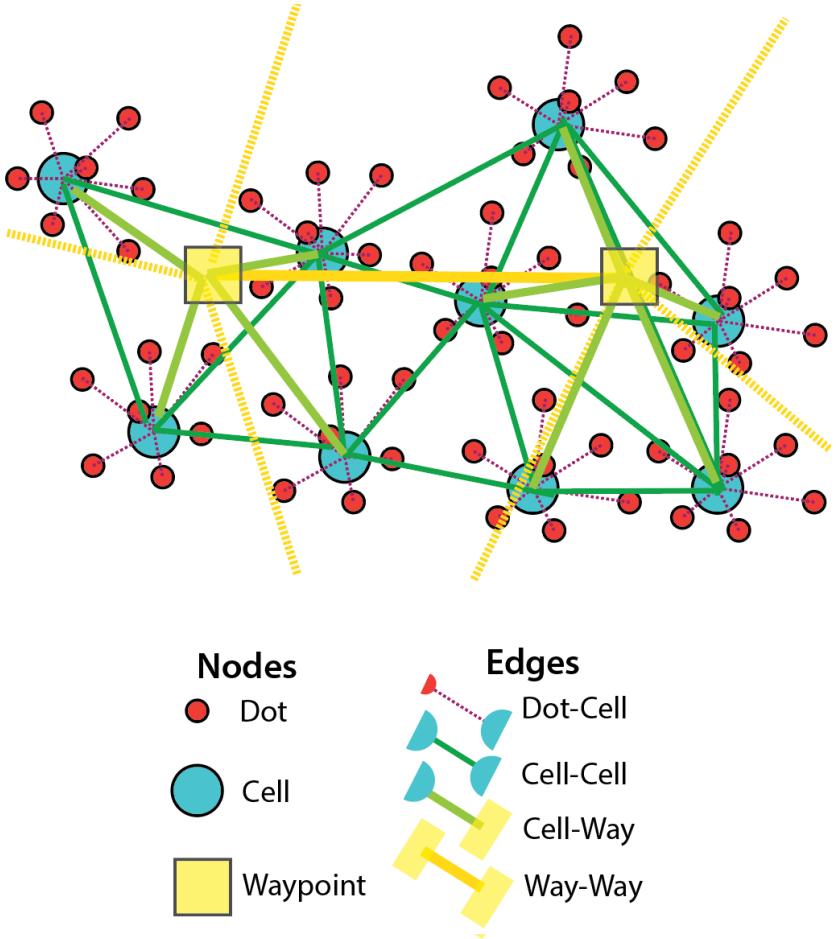


La Manno et al. Nature 2021

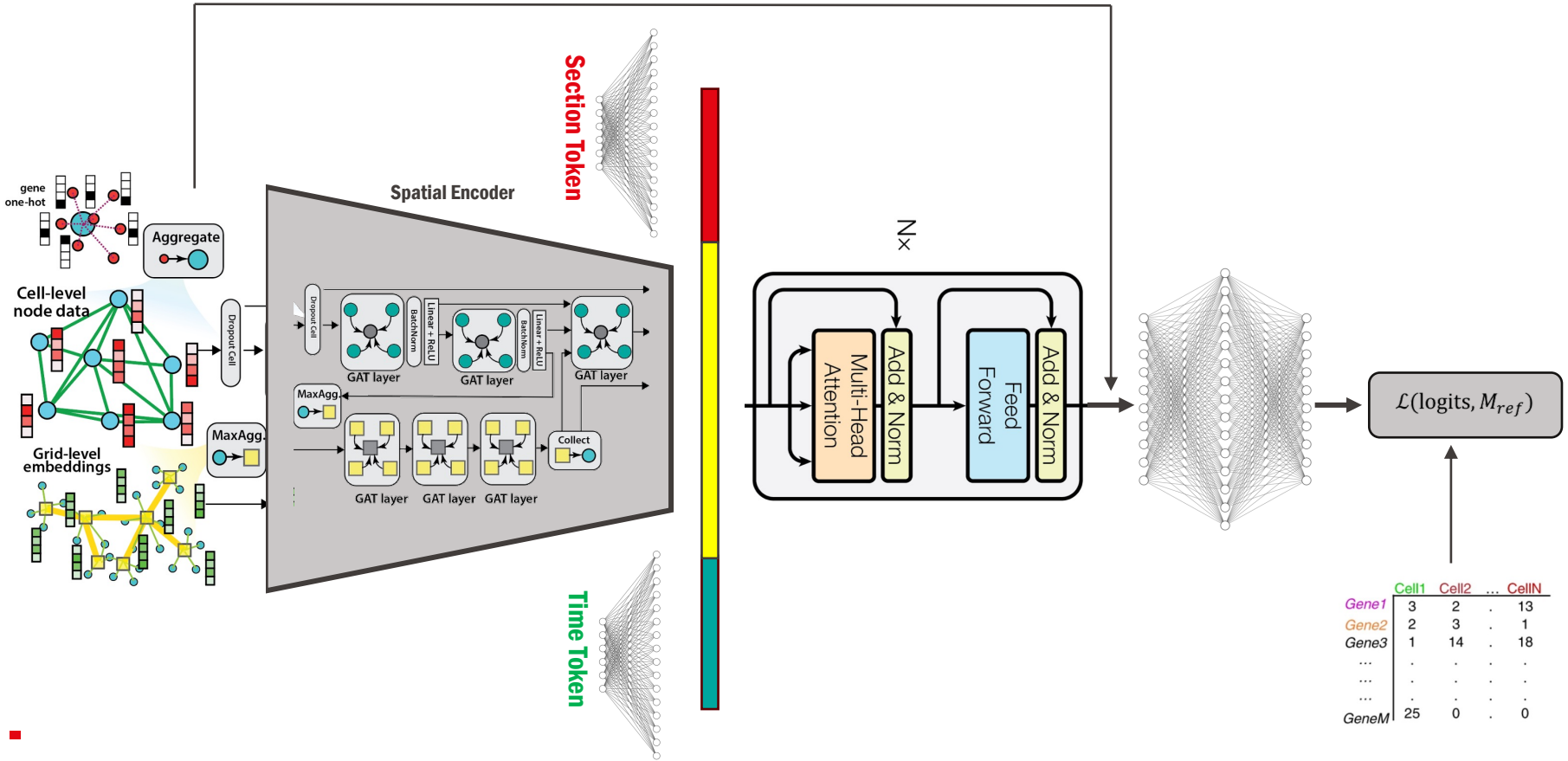
HybISS as the starting point for cell type identification

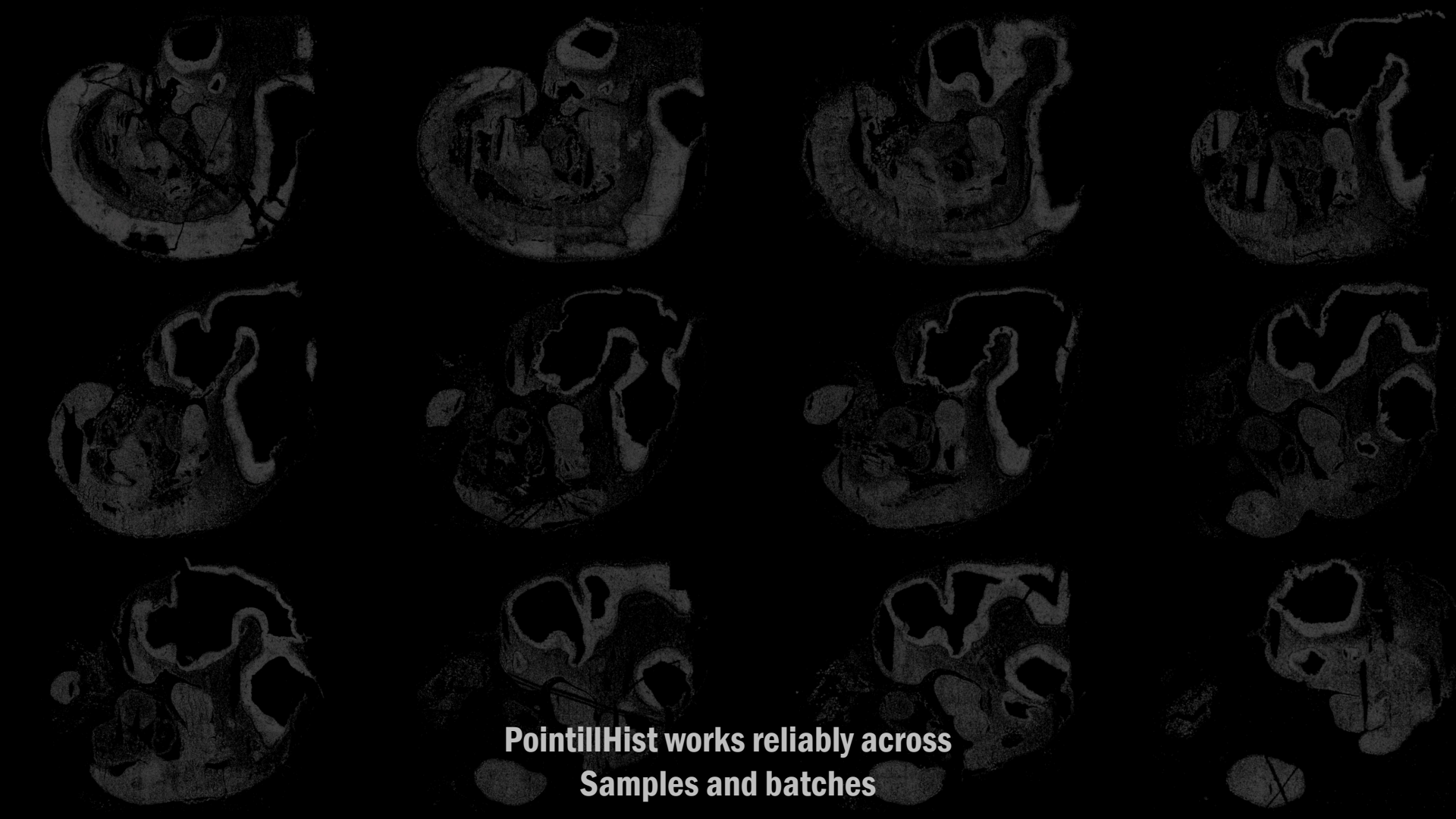


Reasoning on the data structure



PointillHist Architecture



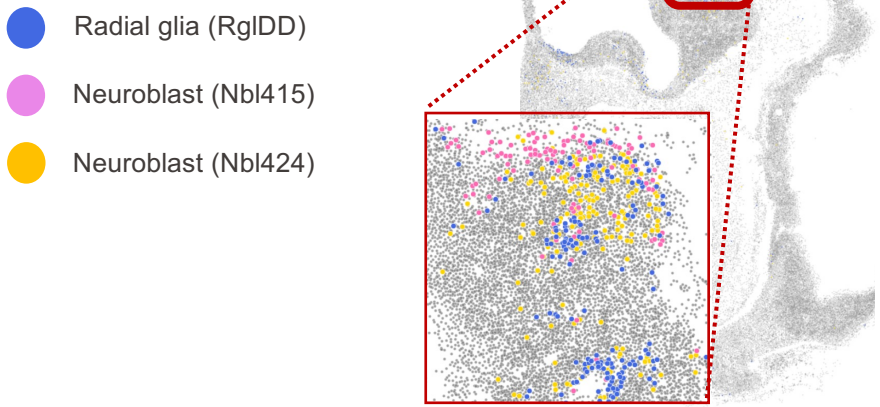


PointillHist works reliably across
Samples and batches

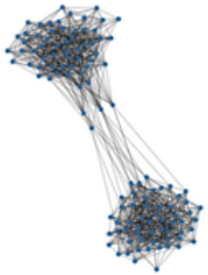
Connecting specific progenitors to progeny

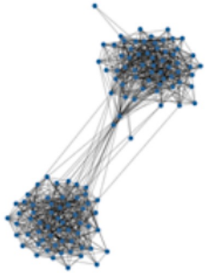
E11.5

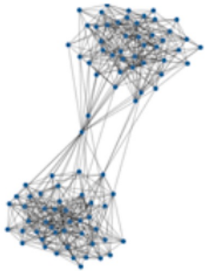
E12.5



Graph Generative Models





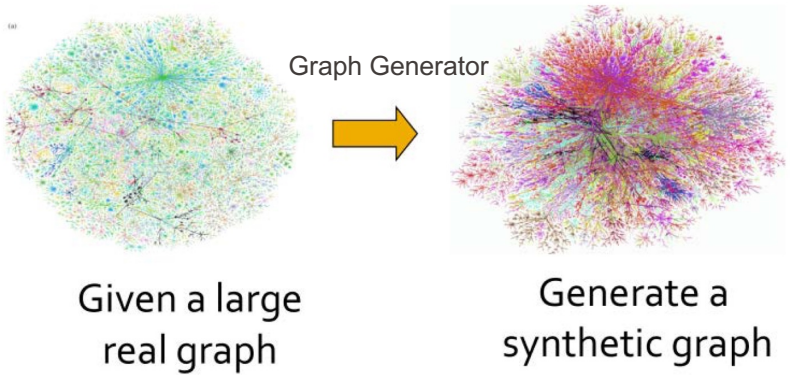


A Survey on Deep Graph Generation: Methods and Application, LoG, PMLR (https://arxiv.org/abs/2203.06714)

Graph Generation Problem

- **Problem:**

Using a set of graphs, we want to generate graphs that resemble similarity to real graphs.



- **Applications:**

Drug design, Material design, Program Synthesis, Social Network Modeling, etc.

Approaches

- **Traditional Methods**
  - Generating graphs with similar statistical and hand-crafted features derived from real data.
  - Oversimplifying assumption!

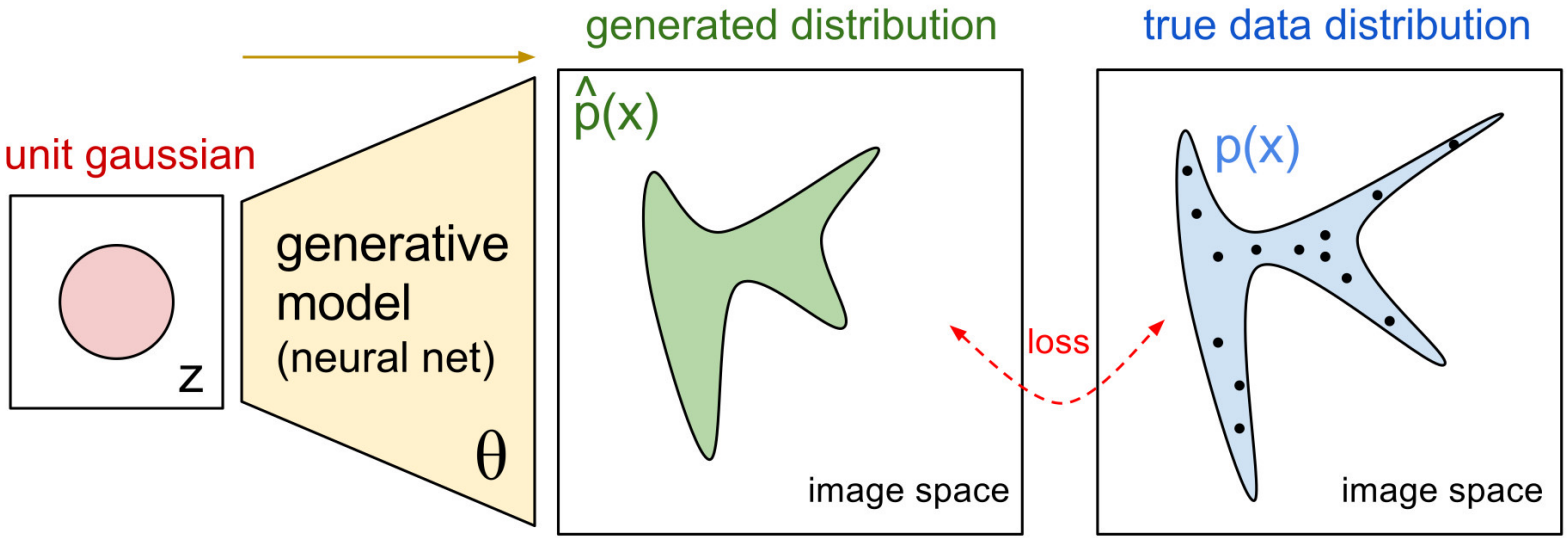
Approaches

- **Traditional Methods**
  - Generating graphs with similar statistical and hand-crafted features derived from real data.
  - Oversimplifying assumption!

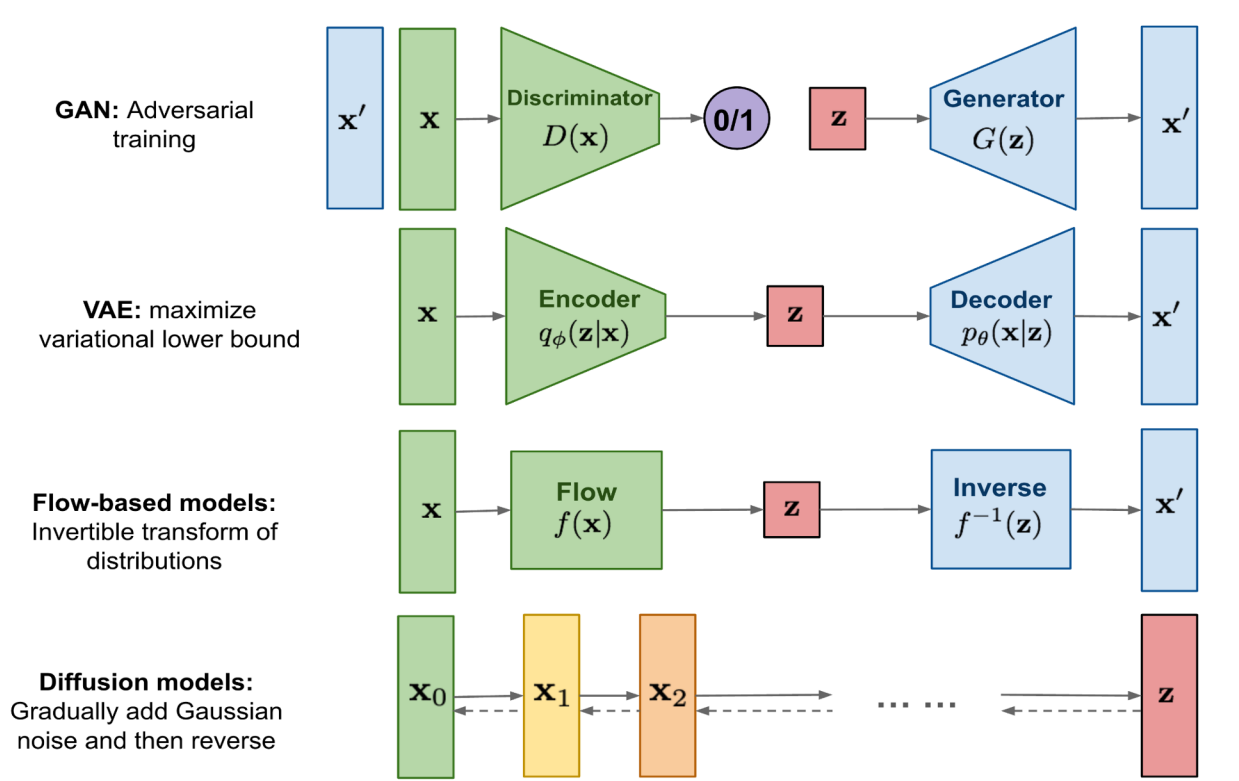
- **Deep Graph Generative Models**
  - Learn the graph formation process from the data itself.
  - Can capture complex relationships!

Overview of Generative Models

- Discriminative models focus on classifying or distinguishing between existing categories of data.
- Generative models, on the other hand, learn to replicate the underlying distribution of data and can generate new samples.

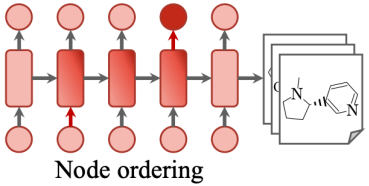


Common Practices in Generative Models

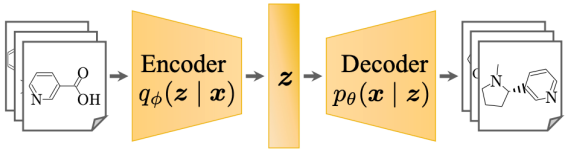


Alireza Gargoori Motlagh

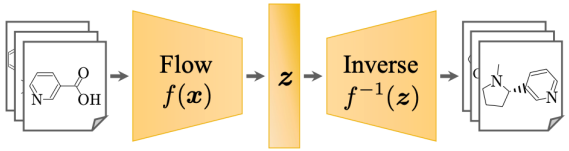
Overview of Deep Graph Generative Models



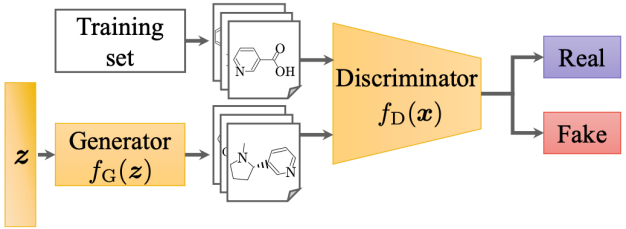
(1) Auto-regressive models



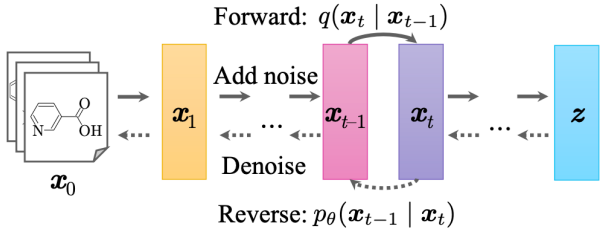
(2) Variational autoencoders



(3) Normalizing flows



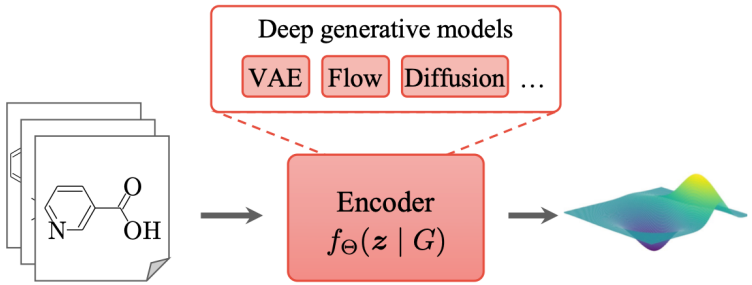
(4) Generative adversarial networks



(5) Diffusion models

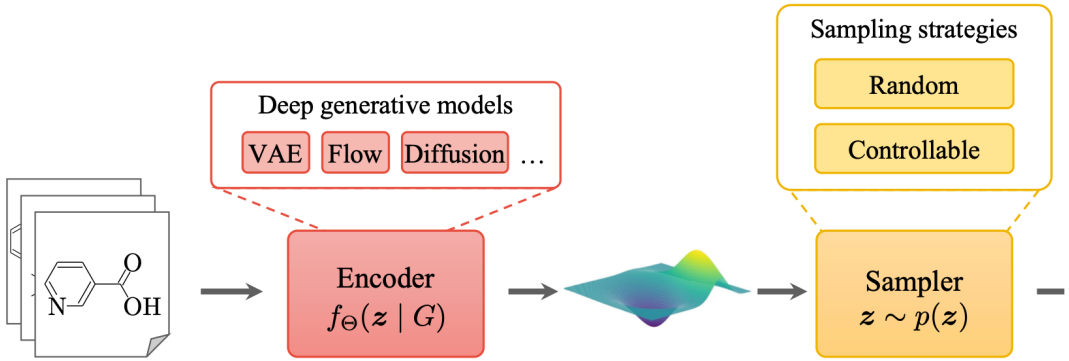
Deep Graph Generative Models

Alireza Gargoori Motlagh



Encoder maps observed graphs into a stochastic distribution.

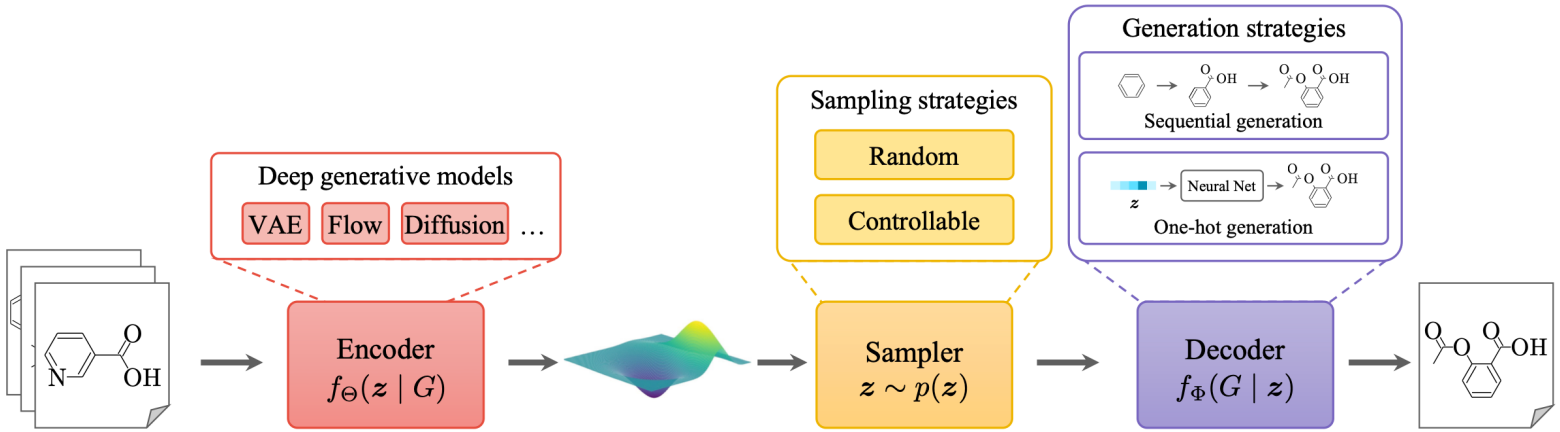
Deep Graph Generative Models



Sampler draws latent representations from that distribution.

Alireza Gargoori Motlagh

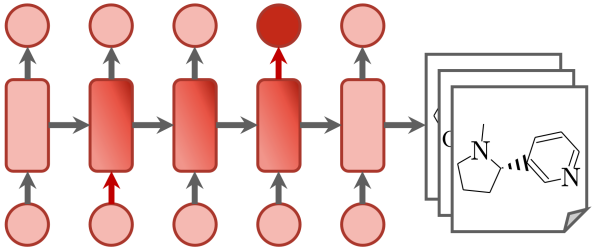
Deep Graph Generative Models



Decoder receives latent codes and produces graphs.

Alireza Gargoori Motlagh

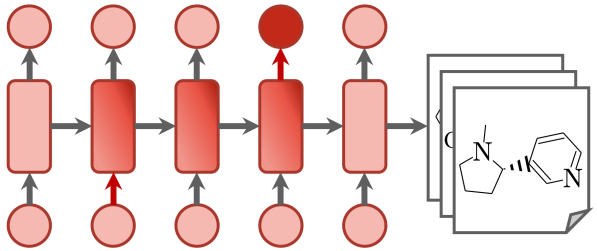
Auto-Regressive Models



- AR models rely on the Chain Rule of Probability:

$$p(G^{\pi}) = \prod_{i=1}^{N} p(G_i^{\pi} \mid G_1^{\pi}, G_2^{\pi}, \cdots, G_{i-1}^{\pi}) = \prod_{i=1}^{N} p(G_i^{\pi} \mid G_{<i}^{\pi})$$

Auto-Regressive Models



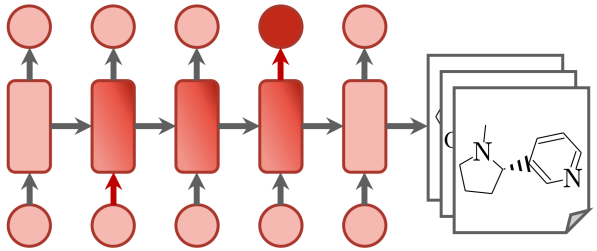
- AR models rely on the Chain Rule of Probability:

$$p(G^{\pi}) = \prod_{i=1}^{N} p(G_i^{\pi} \mid G_1^{\pi}, G_2^{\pi}, \cdots, G_{i-1}^{\pi}) = \prod_{i=1}^{N} p(G_i^{\pi} \mid G_{<i}^{\pi})$$

- The factorized distribution is learnt with Maximum Likelihood Estimation (MLE):

$$\theta^* = \arg\max_{\theta} E_{x \sim p_{data}} \log p_{model}(x \mid \theta)$$

Auto-Regressive Models



- AR models rely on the Chain Rule of Probability:

$$p(G^{\pi}) = \prod_{i=1}^{N} p(G_i^{\pi} \mid G_1^{\pi}, G_2^{\pi}, \cdots, G_{i-1}^{\pi}) = \prod_{i=1}^{N} p(G_i^{\pi} \mid G_{<i}^{\pi})$$

- The factorized distribution is learnt with Maximum Likelihood Estimation (MLE):

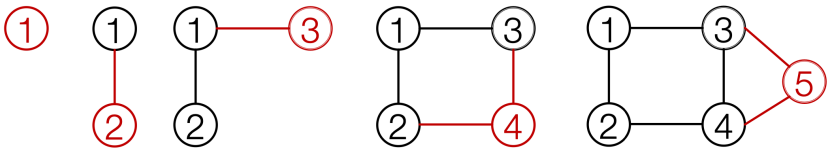
$$\theta^* = \arg\max_{\theta} E_{x \sim p_{data}} \log p_{model}(x \mid \theta)$$

- Since AR works like sequential generation, applying AR models requires a pre-specified ordering π of nodes in the graph.

AR: GraphRNN

The sequence S^π has **two levels** (S is a sequence of sequences):

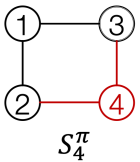
- **Node-level:** add nodes, one at a time
- **Edge-level:** add edges between existing nodes



$S^\pi = (S_1^\pi , S_2^\pi , S_3^\pi , \cdots , S_4^\pi , S_5^\pi)$

"Add node 1" "Add node 5"

- Each Node-level step is an edge-level sequence



S_4^π

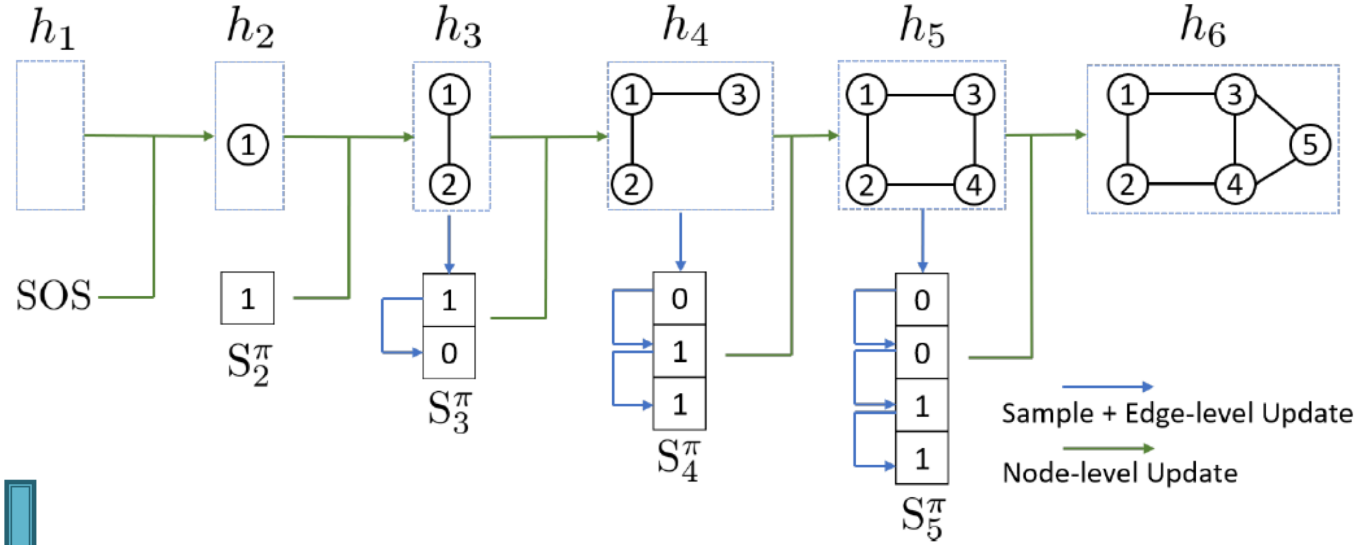
$S_4^\pi = (S_{4,1}^\pi , S_{4,2}^\pi , S_{4,3}^\pi)$

"Not connect 4, 1" "Connect 4, 2" "Connect 4, 3"

0 1 1

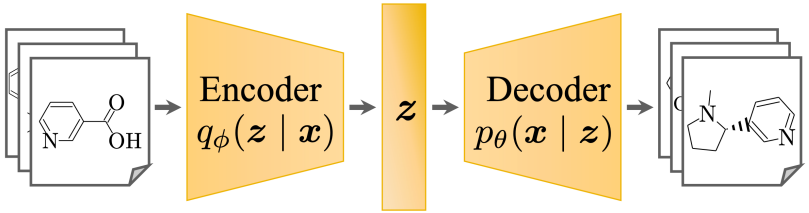
GraphRNN Training

Node-level RNN generates the initial state for edge-level RNN



Edge-level RNN sequentially predict if the new node will connect to each of the previous node

Variational Autoencoders

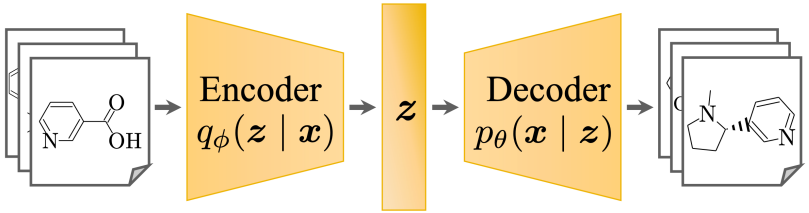


- VAEs estimate the $p(G)$ by maximizing the Evidence Lower Bound (ELBO) as follows:

$$\mathcal{L}_{\text{VAE}} = \mathbb{E}_{z \sim q_\phi(\boldsymbol{z}|G)} \log(p_\theta(G \mid \boldsymbol{z})) - D_{\text{KL}}(q_\phi(\boldsymbol{z} \mid G) \parallel p_\theta(\boldsymbol{z}))$$

Alireza Gargoori Motlagh

Variational Autoencoders



- VAEs estimate the $p(G)$ by maximizing the Evidence Lower Bound (ELBO) as follows:

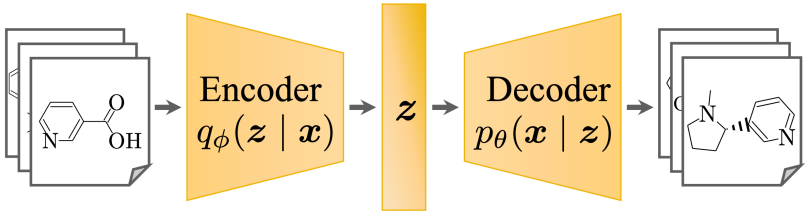
$$\mathcal{L}_{\text{VAE}} = \mathbb{E}_{z \sim q_\phi(\boldsymbol{z}|G)} \log(p_\theta(G \mid \boldsymbol{z})) - D_{\text{KL}}(q_\phi(\boldsymbol{z} \mid G) \parallel p_\theta(\boldsymbol{z}))$$

- The first term is the reconstruction objective. It represents the log-likelihood of the true data G under the reconstructed distribution $p_\theta(G|z)$.

- The second term acts as a regularizer, encouraging the distribution of latent variables $q_\phi(z|G)$ to approximate a prior distribution $p(z)$, often standard normal distribution.

Alireza Gargoori Motlagh

Variational Autoencoders



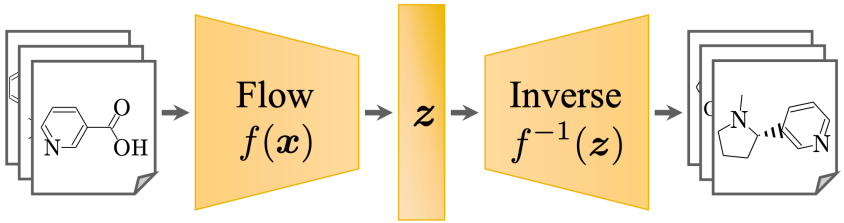
- VAEs estimate the $p(G)$ by maximizing the Evidence Lower Bound (ELBO) as follows:

$$\mathcal{L}_{\text{VAE}} = \mathbb{E}_{z \sim q_\phi(\boldsymbol{z}|G)} \log(p_\theta(G \mid \boldsymbol{z})) - D_{\text{KL}}(q_\phi(\boldsymbol{z} \mid G) \parallel p_\theta(\boldsymbol{z}))$$

- The first term is the reconstruction objective. It represents the log-likelihood of the true data G under the reconstructed distribution $p_\theta(G|z)$.
- The second term acts as a regularizer, encouraging the distribution of latent variables $q_\phi(z|G)$ to approximate a prior distribution $p(z)$, often standard normal distribution.
- Encoder and Decoders are usually parameterized by GNNs (e.g. GCN or GAT layers).

Alireza Gargoori Motlagh

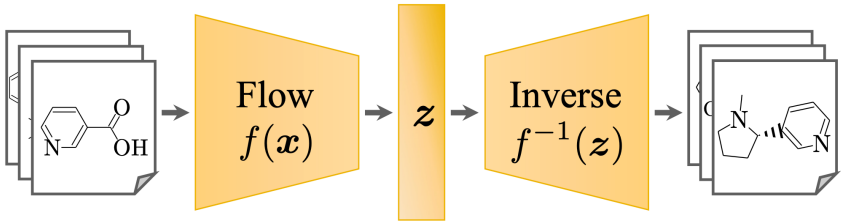
Normalizing Flows



- Normalizing flow estimates the density of graphs $p(G)$ directly with an invertible and deterministic mapping between the latent variables and the graphs via the change of variable theorem.

$$p(G) = p(\boldsymbol{z}) \left| \det \left(\frac{\partial f^{-1}(G)}{\partial G} \right) \right|$$

Normalizing Flows



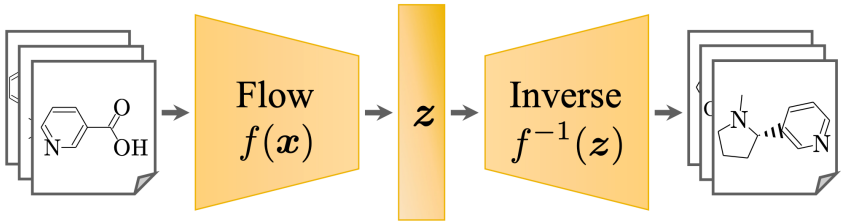
- Normalizing flow estimates the density of graphs $p(G)$ directly with an invertible and deterministic mapping between the latent variables and the graphs via the change of variable theorem.

$$p(G) = p(\boldsymbol{z}) \left| \det \left(\frac{\partial f^{-1}(G)}{\partial G} \right) \right|$$

- Since the encoder $f(G)$ needs to be invertible, the decoder is essentially $f^{-1}(z)$.

Alireza Gargoori Motlagh

Normalizing Flows



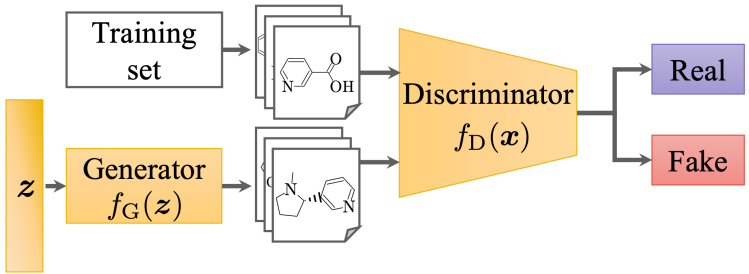
- Normalizing flow estimates the density of graphs $p(G)$ directly with an invertible and deterministic mapping between the latent variables and the graphs via the change of variable theorem.

$$p(G) = p(\boldsymbol{z}) \left| \det \left(\frac{\partial f^{-1}(G)}{\partial G} \right) \right|$$

- Since the encoder $f(G)$ needs to be invertible, the decoder is essentially $f^{-1}(z)$.

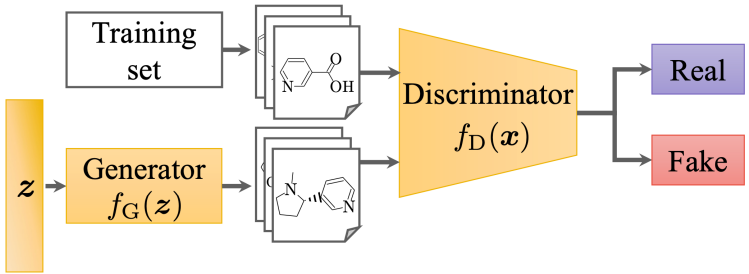
- Normalizing-flow-based models are usually trained by maximizing the log-likelihood over the training data G.

Generative Adversarial Networks



- GANs consists of two main components:
  - Generator f_G for generating realistic graphs
  - Discriminator f_D for distinguishing between synthetic and real graphs.

Generative Adversarial Networks

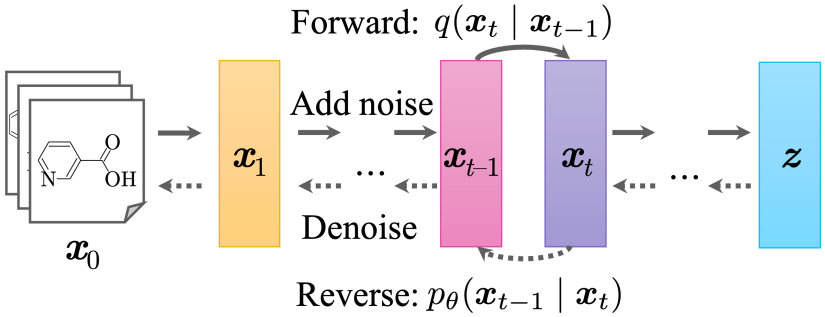


- GANs consists of two main components:
  - Generator f_G for generating realistic graphs
  - Discriminator f_D for distinguishing between synthetic and real graphs.
- The training objective is the following min-max game:

$$\min_{f_G} \max_{f_D} \mathcal{L}_{\text{GAN}}(f_G, f_D) = \mathbb{E}_{G \sim p(G)}[\log f_D(G)] + \mathbb{E}_{z \sim p(z)}[\log(1 - f_D(f_G(z)))]$$

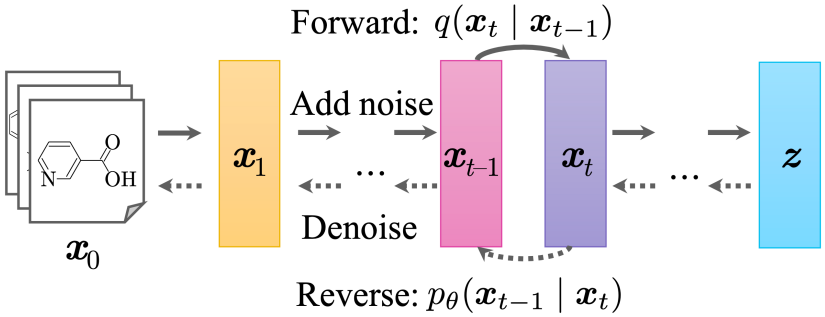
- This competition drives both networks to improve continuously, allowing GANs to generate increasingly realistic data.

Diffusion Models



- The forward diffusion process constantly adds noise to the data sample x_0, while the reverse diffusion process recreates the true data sample from a Gaussian noise input $x_T \sim \mathcal{N}(0, I)$.

Diffusion Models



- The forward diffusion process constantly adds noise to the data sample x_0, while the reverse diffusion process recreates the true data sample from a Gaussian noise input $x_T \sim \mathcal{N}(0, I)$.

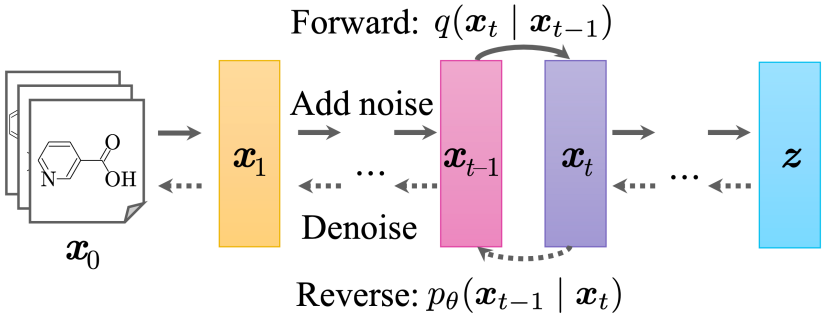
- Forward diffusion process:

$$q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t;\ \sqrt{1-\beta_t}\boldsymbol{x}_{t-1},\ \beta_t \boldsymbol{I}),$$

$$q(\boldsymbol{x}_{1:T} \mid \boldsymbol{x}_0) = \prod_{t=1}^{T} q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}),$$

β_t is a noise scheduler that controls the step size.

Diffusion Models



- The forward diffusion process constantly adds noise to the data sample x_0, while the reverse diffusion process recreates the true data sample from a Gaussian noise input $x_T \sim \mathcal{N}(0, I)$.

- Forward diffusion process:

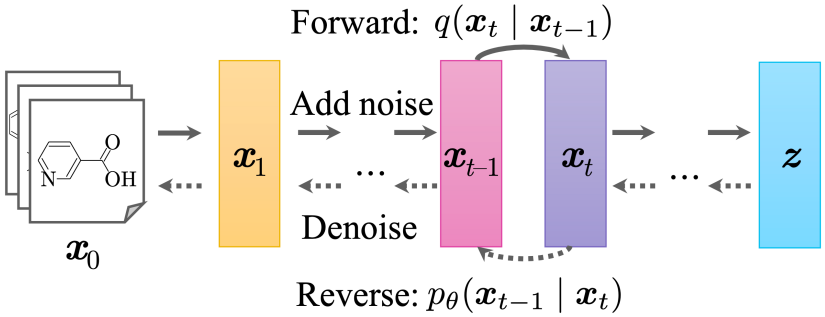
$$q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t;\ \sqrt{1-\beta_t}\boldsymbol{x}_{t-1},\ \beta_t \boldsymbol{I}),$$

$$q(\boldsymbol{x}_{1:T} \mid \boldsymbol{x}_0) = \prod_{t=1}^{T} q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}),$$

β_t is a noise scheduler that controls the step size.

- ***Statistical Physics:*** Reverse diffusion process would also be gaussian if β_t is small enough!

Diffusion Models

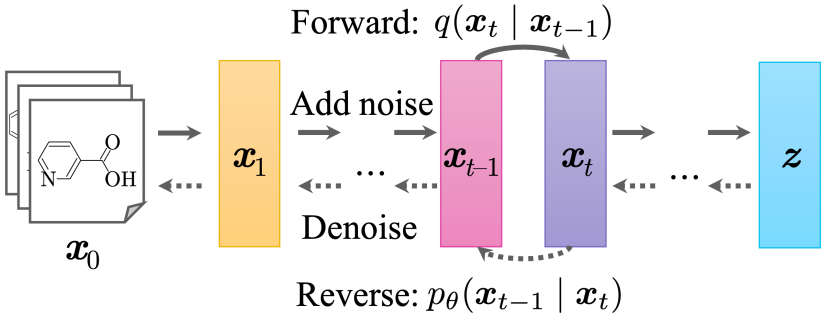


- Since $q(x_{t-1}|x_t)$ is intractable, we learn a model to approximate these conditional probabilities.

$$p_\theta(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t), \boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t, t))$$

Alireza Gargoori Motlagh

Diffusion Models



- Since $q(x_{t-1}|x_t)$ is intractable, we learn a model to approximate these conditional probabilities.

$$p_\theta(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t), \boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t, t))$$

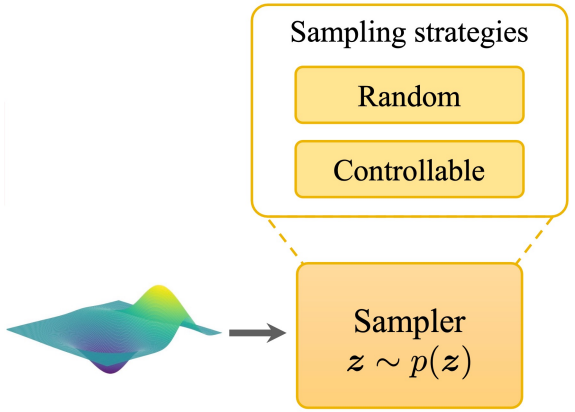
- We use variational lower bound to maximize log likelihood:

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

$$\mathcal{L}_{\text{VLB}} = \mathbb{E}_{q(\boldsymbol{x}_{0:T})} \left[\log \frac{q(\boldsymbol{x}_{1:T} \mid \boldsymbol{x}_0)}{p_\theta(\boldsymbol{x}_{0:T})} \right] \geq -\mathbb{E}_{q(\boldsymbol{x}_0)} \log p_\theta(\boldsymbol{x}_0)$$

Sampling Strategies

Random Sampling: draws latent samples from the prior distribution, in which the model learns to approximate the distribution of the observed graphs.



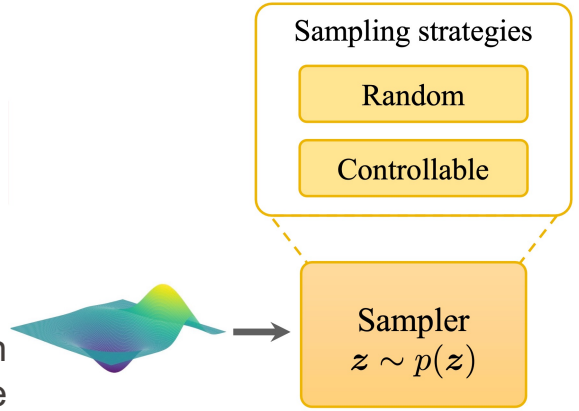
Sampling Strategies

Random Sampling: draws latent samples from the prior distribution, in which the model learns to approximate the distribution of the observed graphs.

Controllable Sampling: samples new graphs with controls (i.e. desired properties)

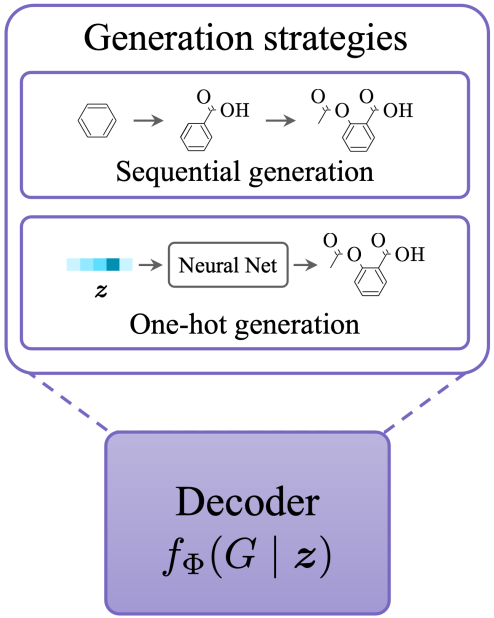
- ***Disentangled sampling:*** factorizes the latent vector $\boldsymbol{z}$ with each dimension z_n focusing on one property p_n, following the disentanglement regularization that encourages the learnt latent variables to be disentangled from each other.

- ***Conditional sampling:*** introduces a conditional code $\boldsymbol{c}$ that explicitly controls the property of generated graphs. In this case, the final representation $\hat{\boldsymbol{z}}$ is usually the concatenation of $\boldsymbol{z}$ and $\boldsymbol{c}$.



Generation Strategies

The decoder transforms latent codes into graph structures, tackling challenges like the discrete and high-dimensional nature of graph data. To address non-differentiability, two decoding strategies are commonly used:



Generation Strategies

One-Shot Generation:

Generates the entire graph (adjacency matrix and optional node/edge features) in a single step.

- ***Advantages:*** Independent of node ordering, efficient for small graphs.
- **Limitations:** Requires predefined maximum nodes and scales poorly $(O(N^2))$ with graph size.

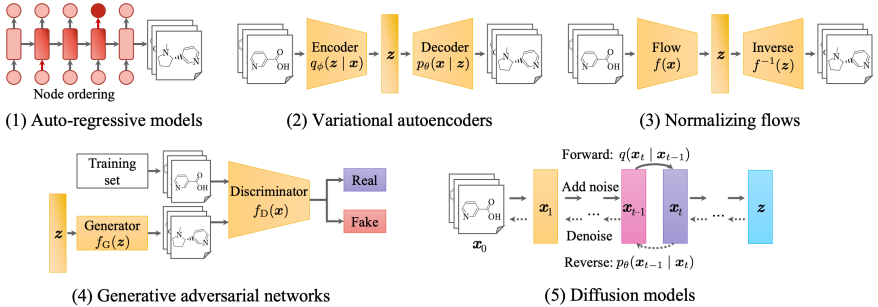
Sequential Generation:

Generates graphs step-by-step by sampling probabilistic node/edge matrices and relies on predefined node ordering (BFS).

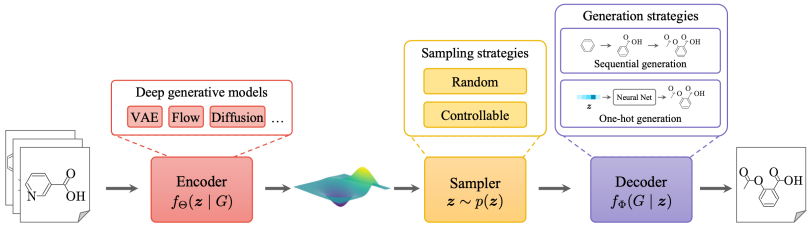
- ***Advantages:*** Flexible for unknown graph sizes and allows constraint checking during generation.
- **Limitations:** Accumulates errors over long sequences, causing discrepancies in large graphs.

Wrap-up

- We studied different approaches for deep graph generative models.



- We evaluated the possible strategies for encoding, sampling, and decoding for latent variable models.



Graph Generative Models: Application for Biological Science

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

Ali Goktug Attar

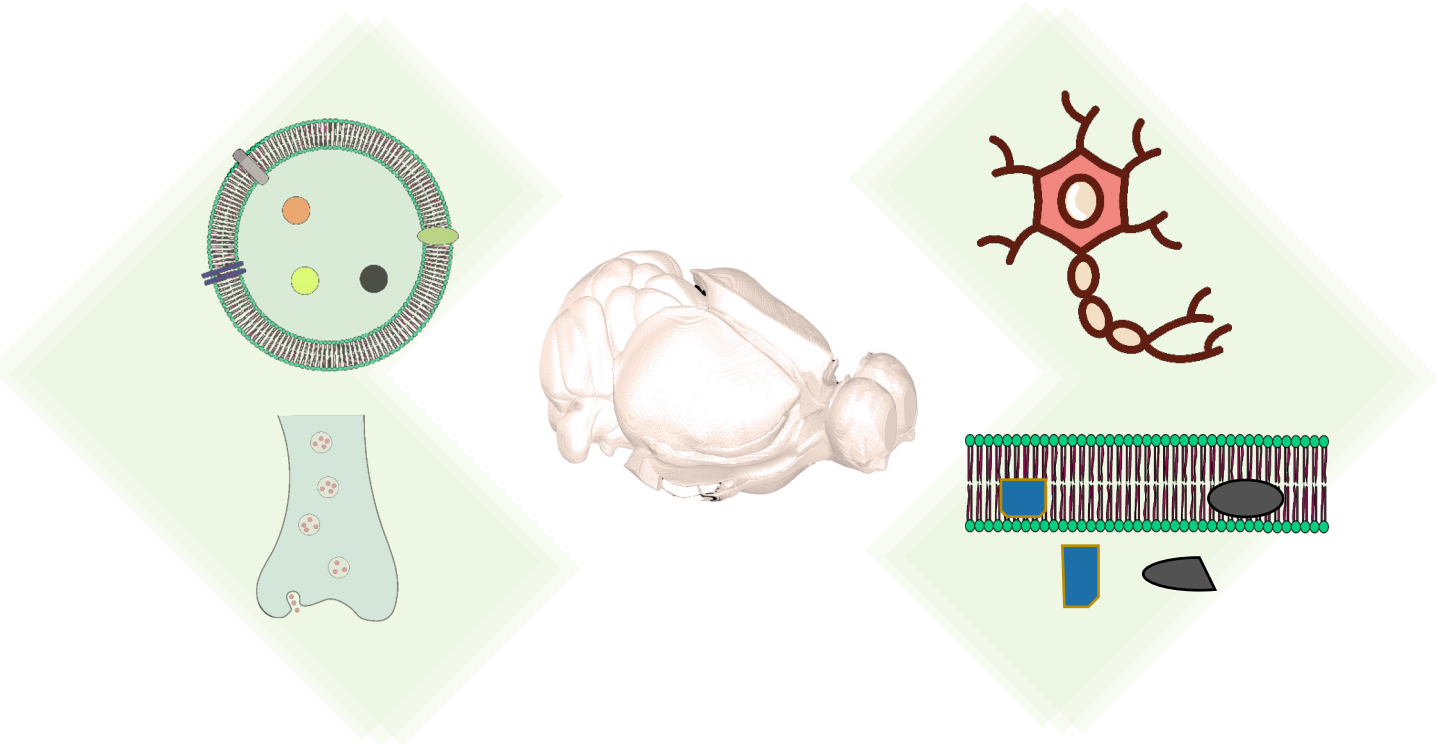
PhD Student @ EDCB

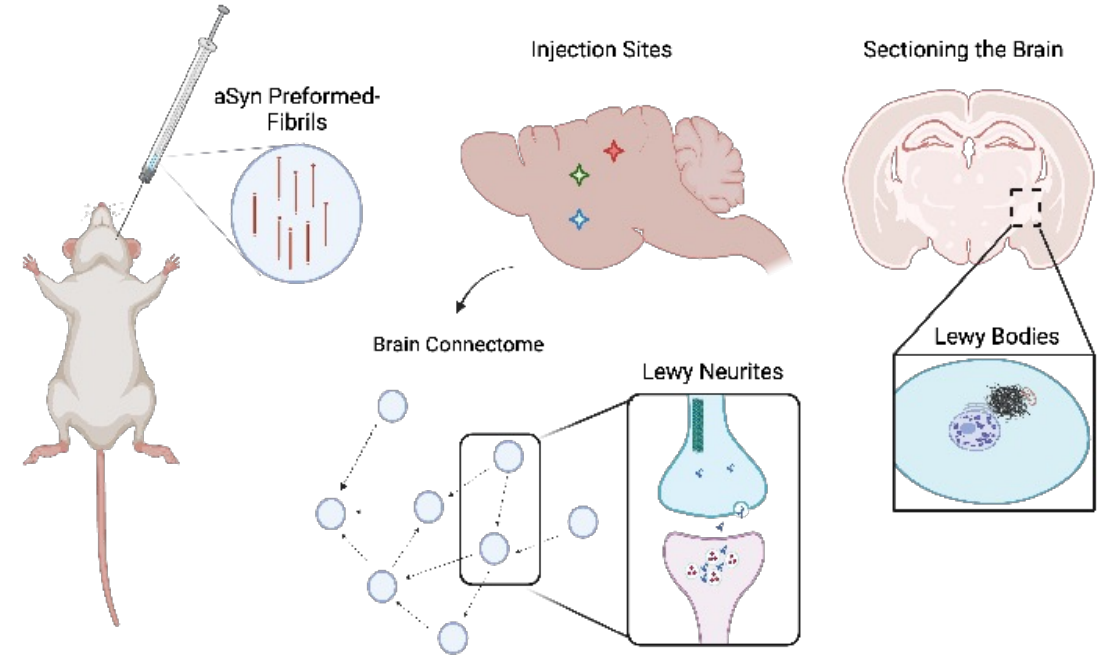
La Manno Lab

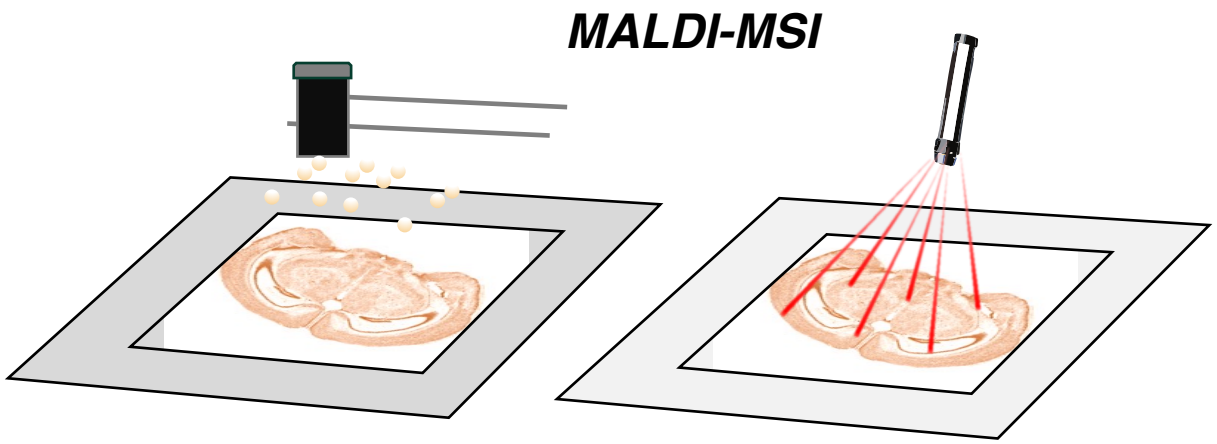
EE-626

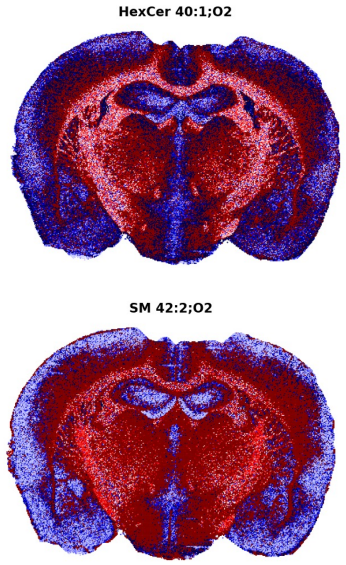
27.11.2024

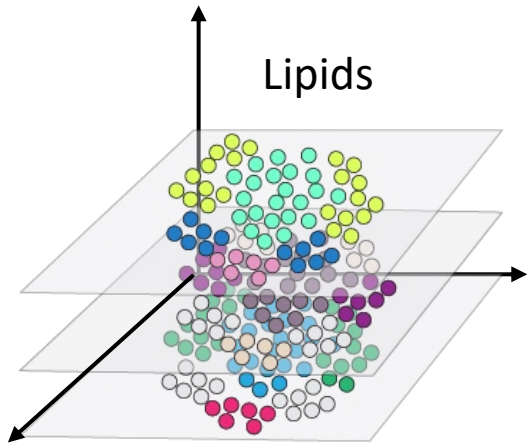
Research Background



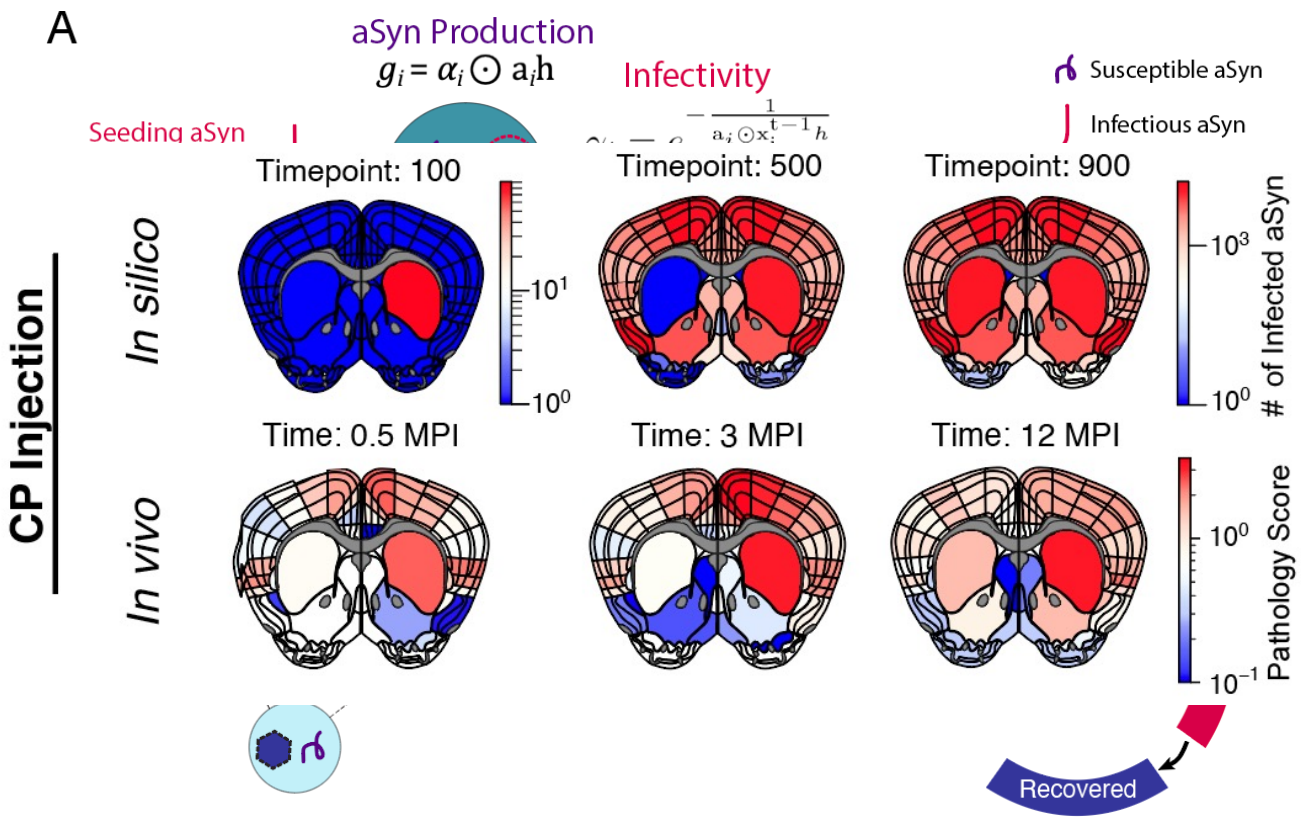


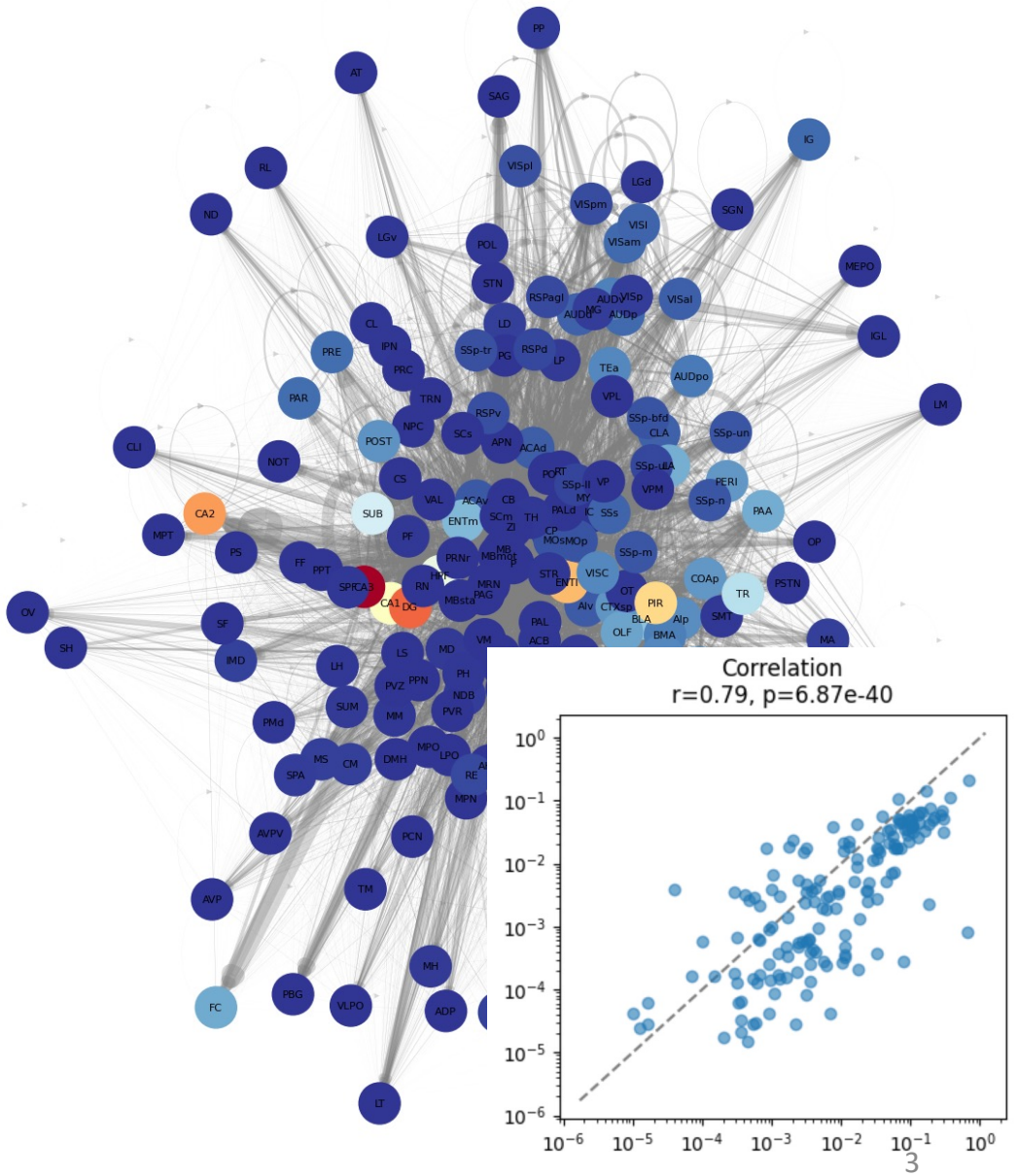




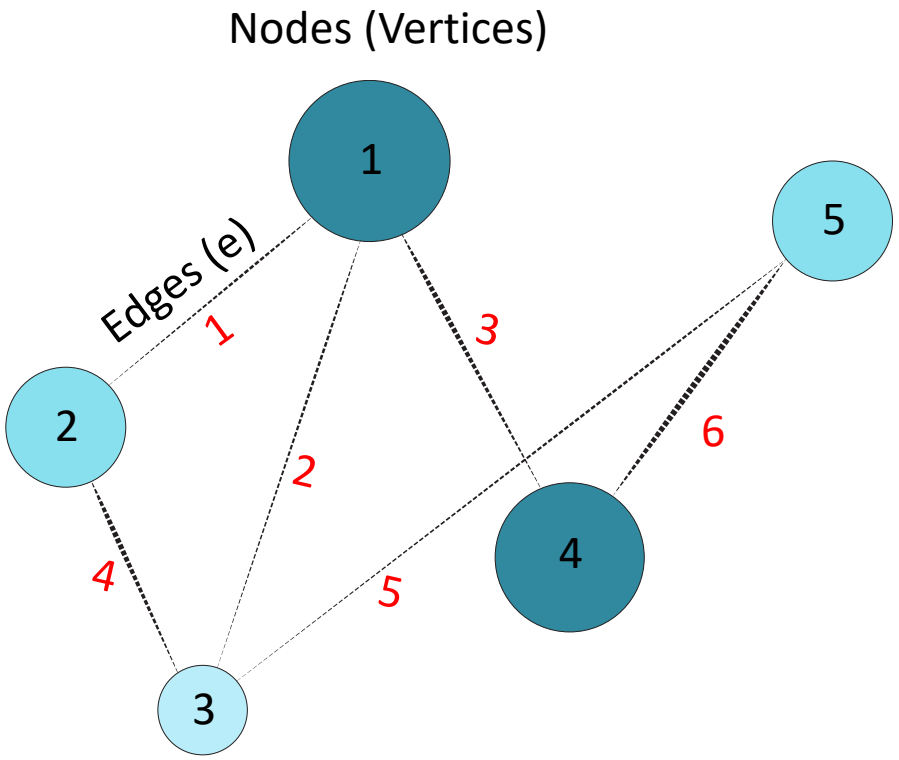


Modelling Approaches





Introduction to Graphs



$$G = \{V, E\}$$

$$V = \{v1, \ldots, v_N\}$$

$$E = \{e1, \ldots, e_M\}$$

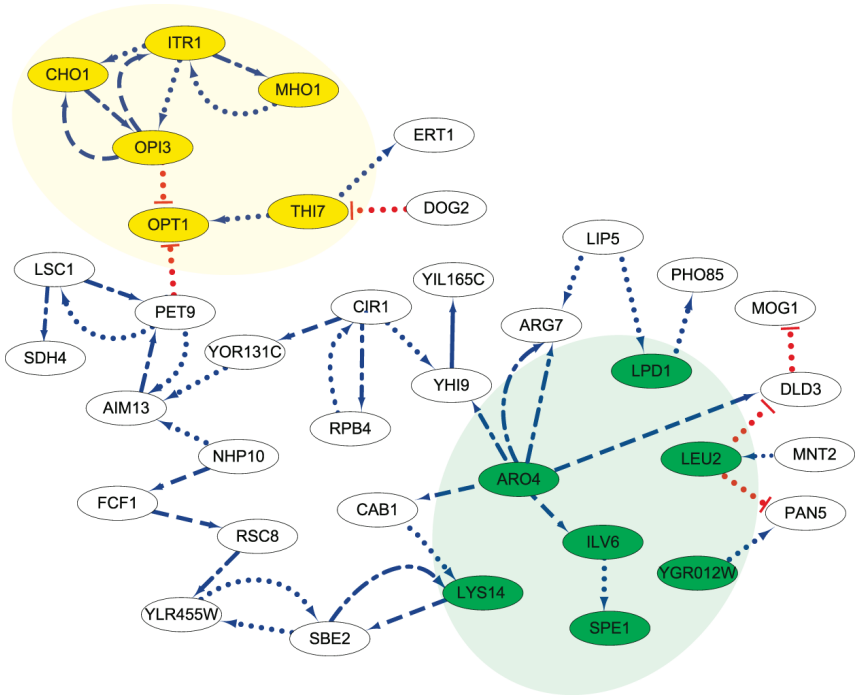
$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

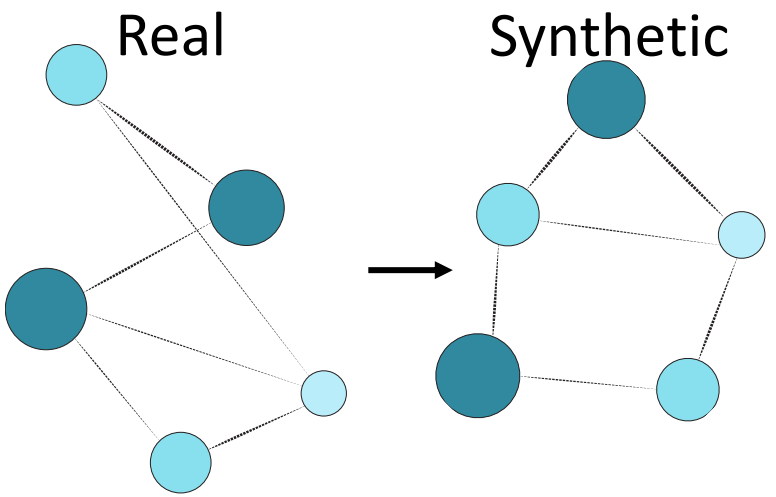
- Graphs can represent various structures
- Can reveal hidden patterns and relationships

Graph Generative Models

- **Definition**: Models designed to generate graphs that mimic real-world structures.
- Capturing the patterns & relationship
- Synthetic graphs that maintain the statistical properties

Chen et al, 2019





Why Study Generative Models for Graphs?

Limited Data Availability

Generate synthetic graphs when real data is scarce or restricted

Drug Discovery

Design novel molecular structures with desired properties

Predictions / Simulations

Insights / Discoveries

Understanding Complex Systems

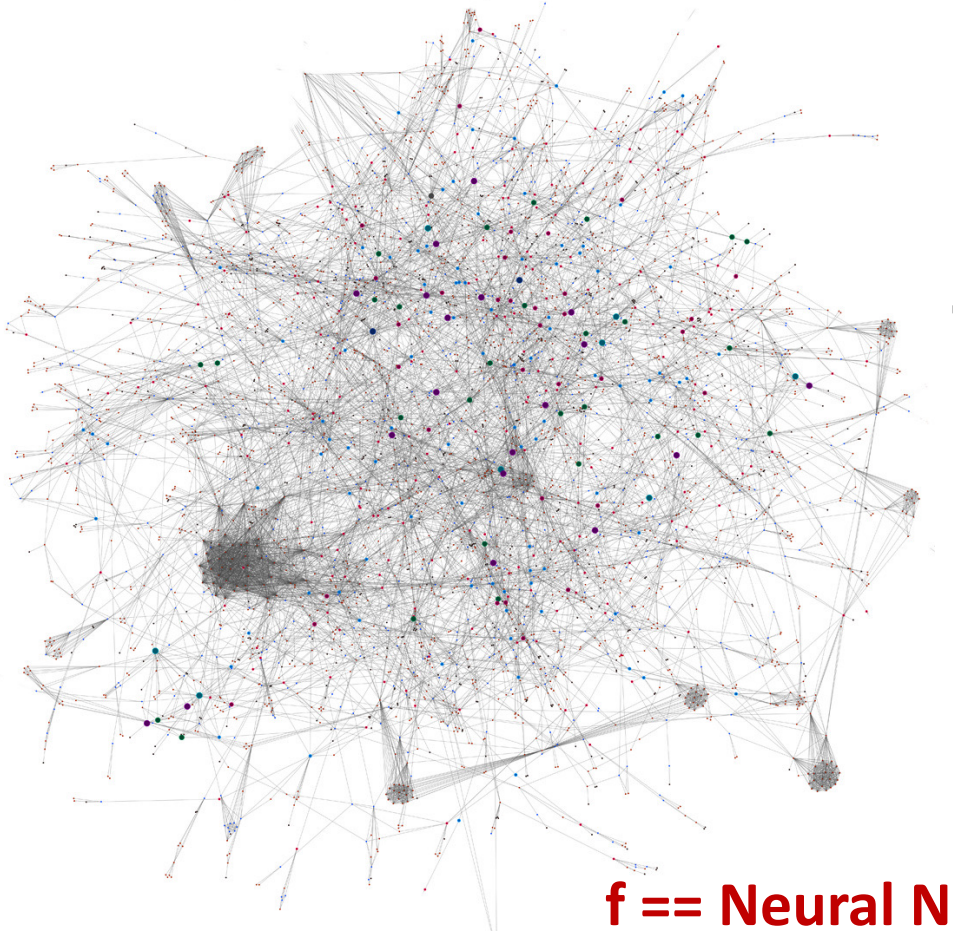
Model how networks form and evolve over time

Scientific Discovery

Explore possible configurations of complex networks

Basics of Graph Generative Models

$P_{data}(G)$



Learn the Distribution

$x_i \sim P_{data}(x)$

Model

Maximum Likelihood

$$\theta^* = \arg\max_{\theta} \mathbb{E}_{x \sim p_{\text{data}}} \log p_{\text{model}}(\boldsymbol{x} \mid \boldsymbol{\theta})$$

$\sum_i \log p_{model}(x_i;\theta^*)$

Model that generated the observed sample X

f == Neural Network To be Trained!

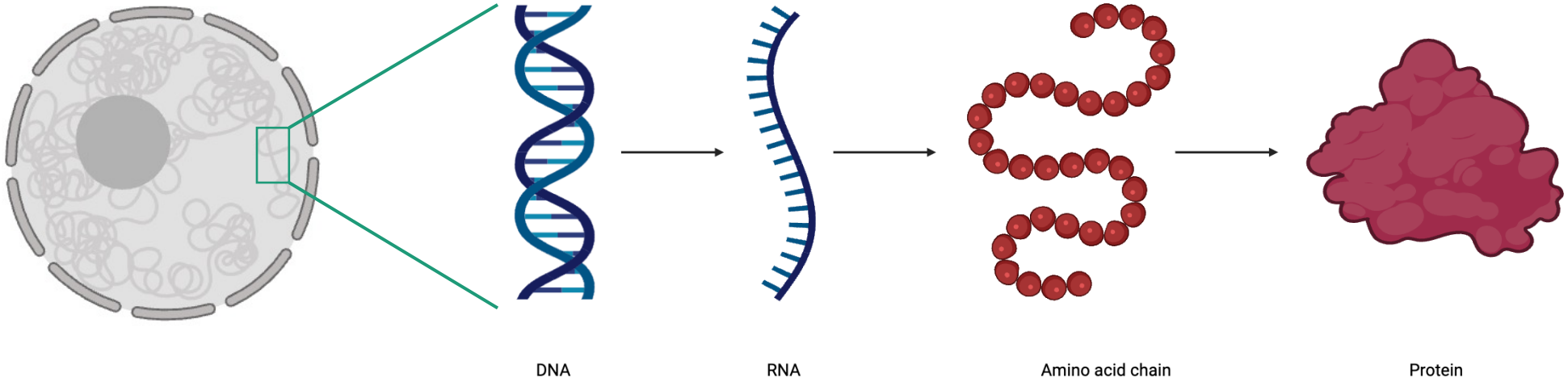
$z_i \sim N(0,1)$

$x_i = f(z_i; \theta)$

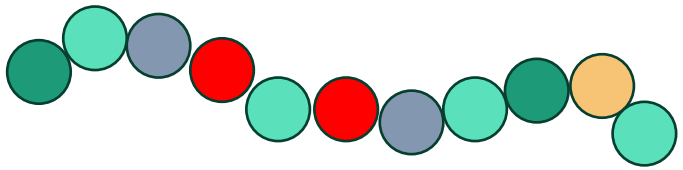
Illuminating protein space with a programmable generative model

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

Proteins: Element of Life

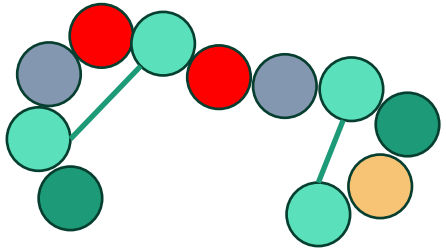


Primary protein structure



Polymer Representation

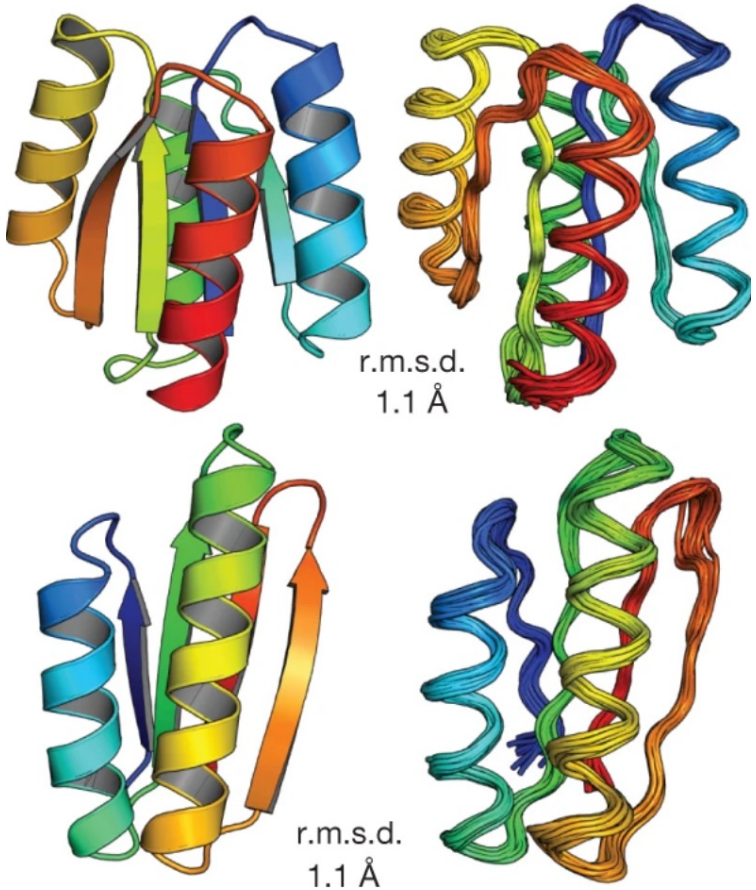
Tertiary protein structure



CA^{2+} BOUND CALCYCLIN

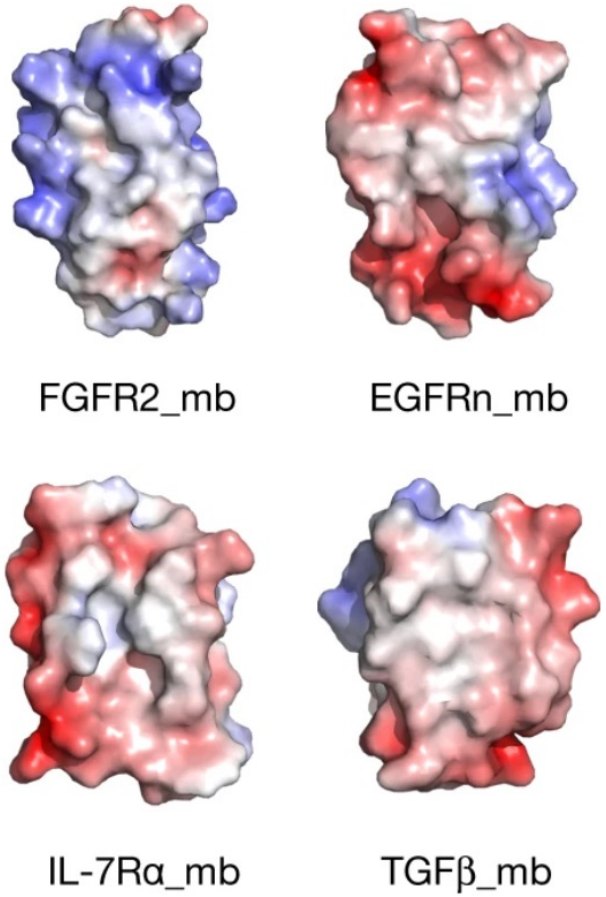
Computational Protein Design

In silico design *Wild-type*



Koge et al., 2012

In silico design of ”miniprotein binders”



Cao et al., 2022

Designing de novo proteins for biological problems

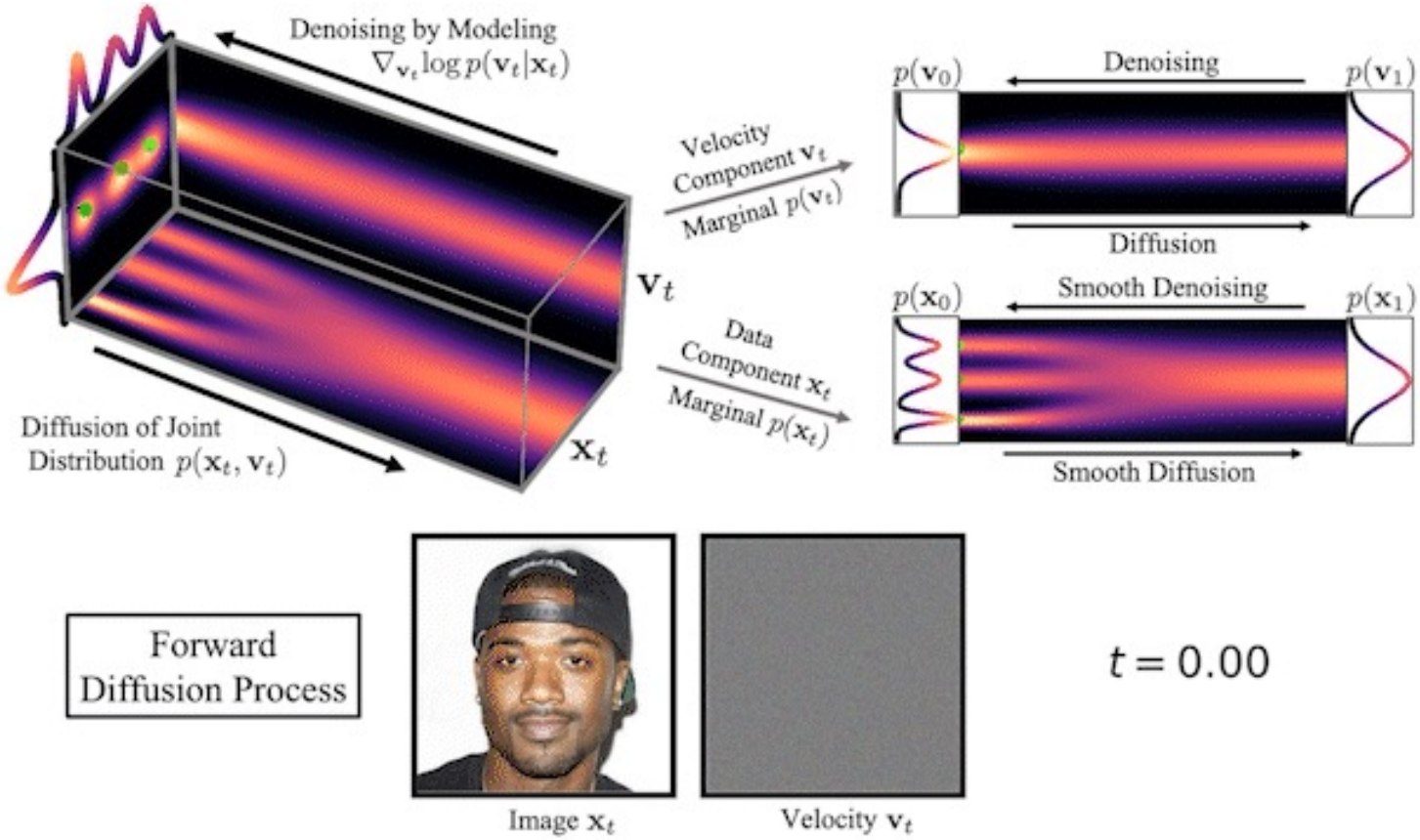
Designing more robust enzymes

Understand complex physical interactions

Diffusion Models

Midjourney
DALL-E 2

Generating images and other types of data by gradually adding random noise to data and then learning to reverse this

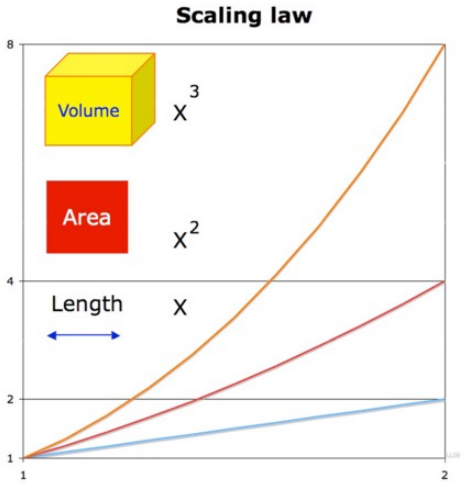


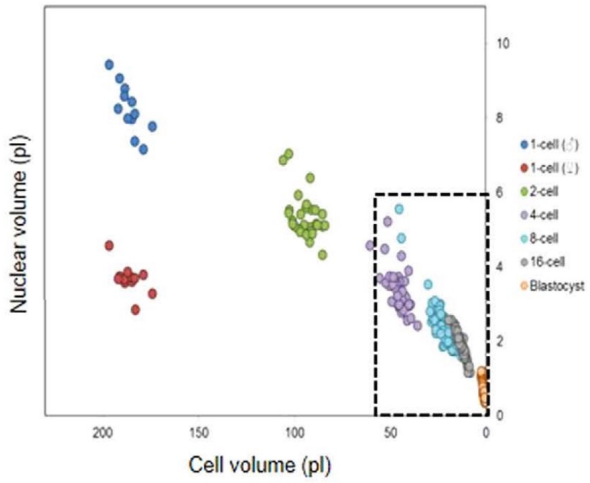
Nvidia Developer Web Site

Chroma: a generative model for proteins and protein complexes

Generative Model:

- Consider Joint Sequences
- Full 3D Structures of Proteins
- Diverse design constraints without retraining

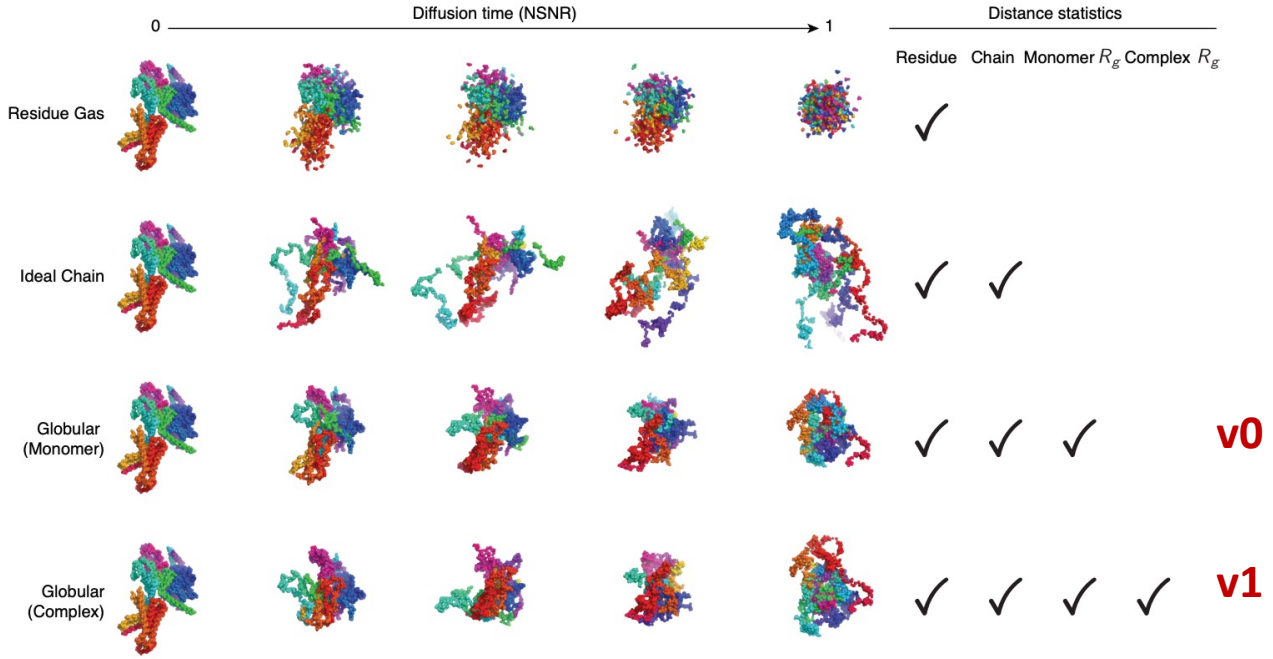




Tsichlaki and FitzHarris, 2016

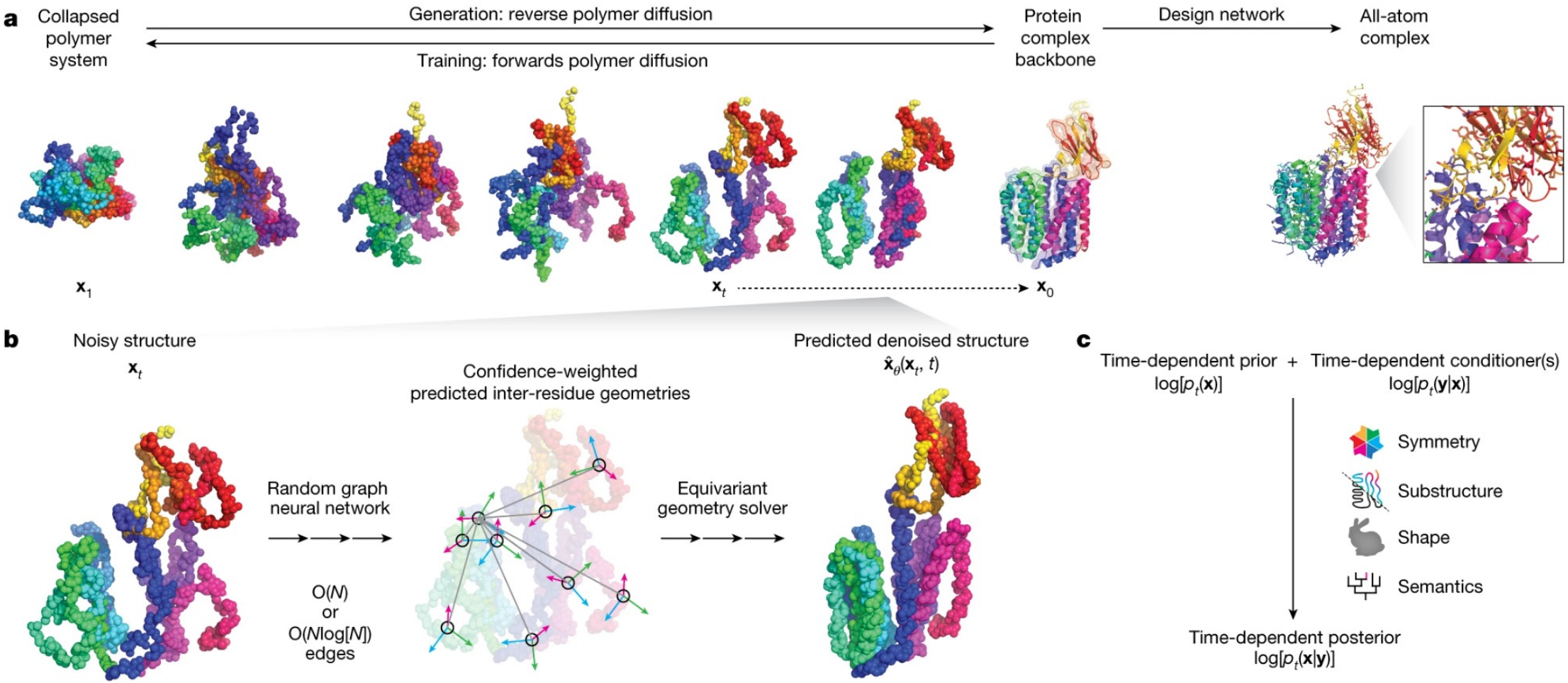
- ***Multivariate Gaussian Distributions***
  - ***Enforce protein chain and R_g statistics***

(Correlated Noise)

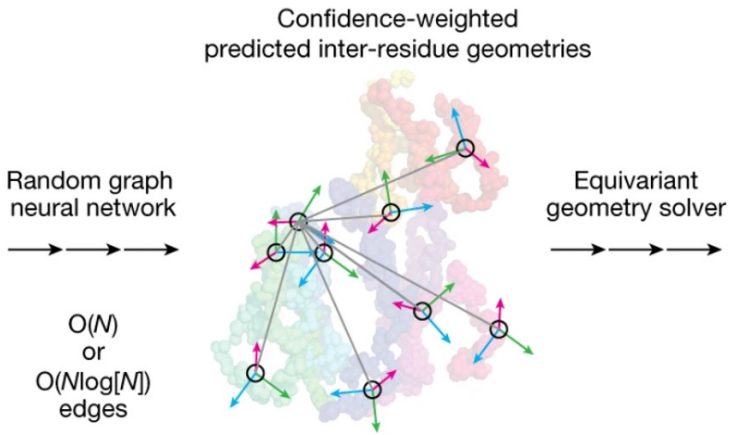


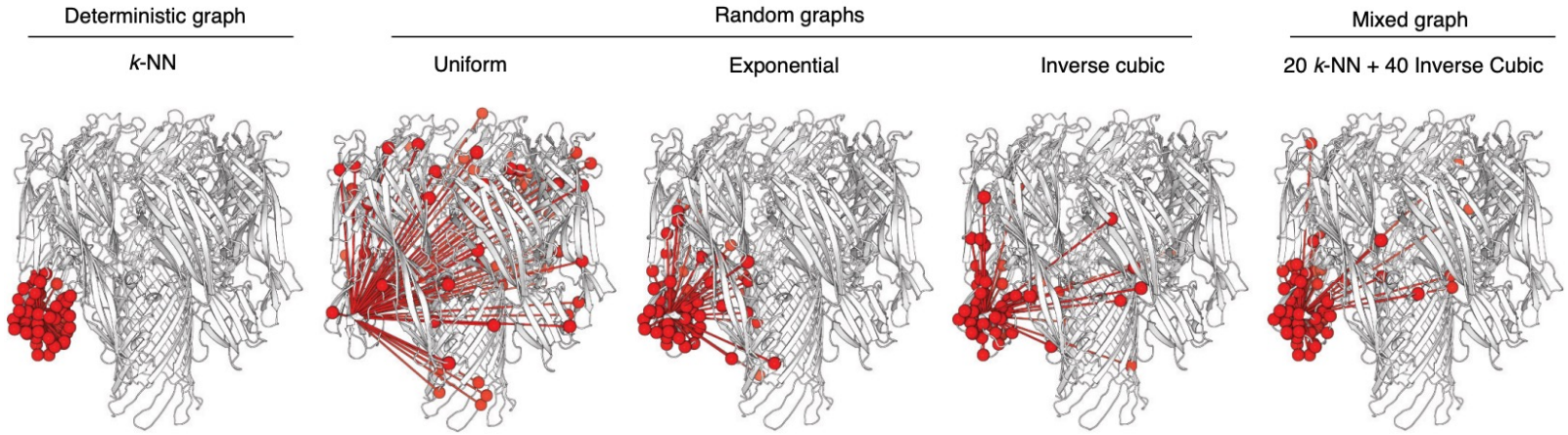
$$R_g \approx 2. \times N^{0.4}$$

Fig. 1: Chroma is a generative model for proteins and protein complexes that combines structured diffusion for protein backbones with scalable molecular neural networks for backbone synthesis and all-atom design.



Supplementary Figure 4: Random graphs with distance-weighted attachment efficiently capture long-range context

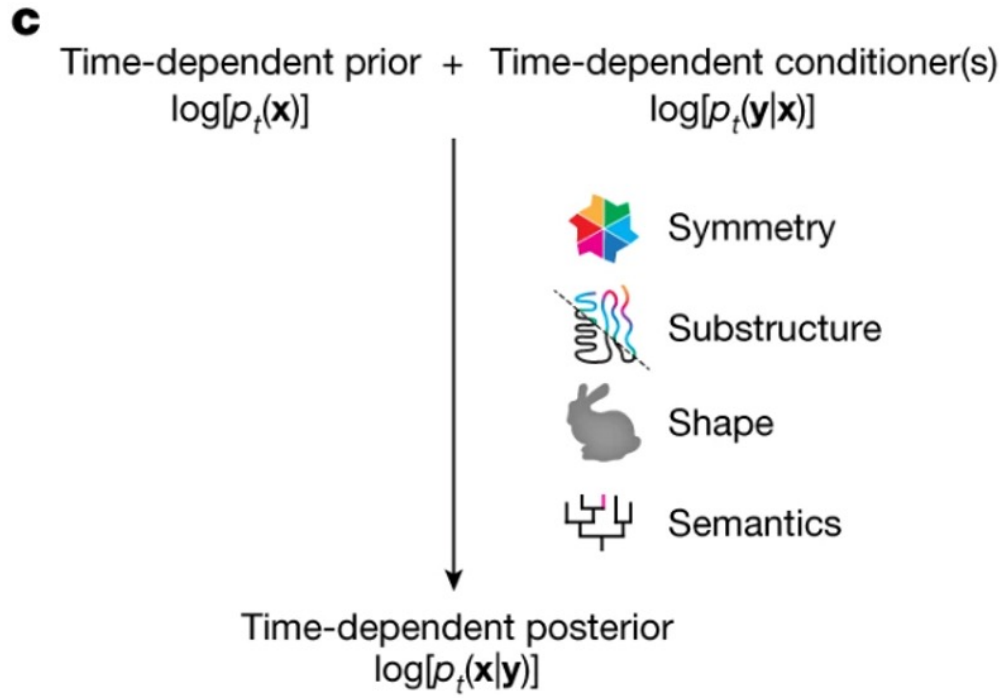




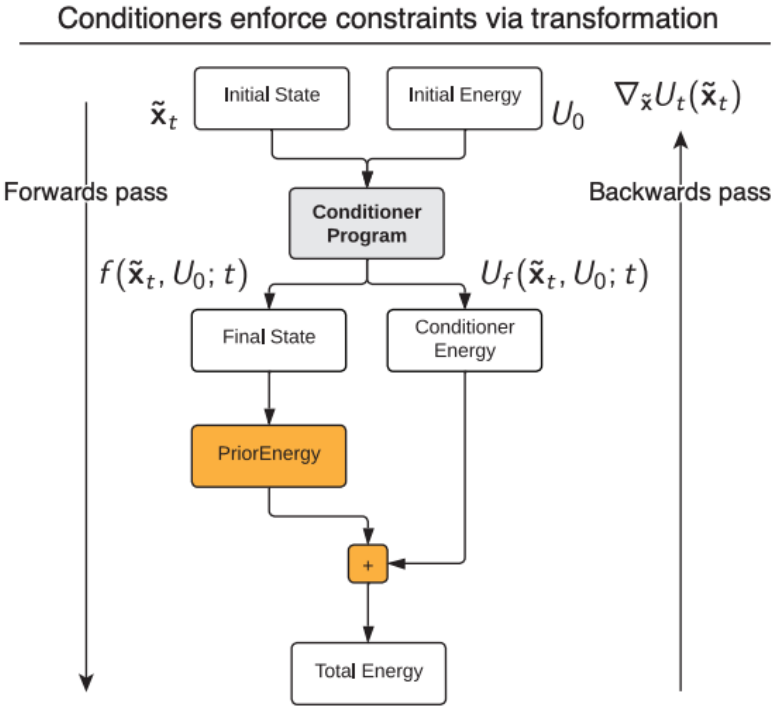
	Deterministic graph	Random graphs			Mixed graph
	k-NN	Uniform	Exponential	Inverse cubic	20 *k*-NN + 40 Inverse Cubic
Edge propensity $p(\mathcal{E}_{ij} \in \mathcal{G}(\mathbf{x}) \mid D_{ij}(\mathbf{x})) \propto$		constant	$\exp\left(-\frac{D_{ij}(\mathbf{x})}{l}\right)$	$\frac{1}{D_{ij}(\mathbf{x})^3}$	
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				✓	✓

Fig. 1: Chroma is a generative model for proteins and protein complexes that combines structured diffusion for protein backbones with scalable molecular neural networks for backbone synthesis and all-atom design.

Conditioning Framework: Allows control over generated properties



Bayes' theorem for score functions



Supplementary Figure 2: Low-temperature sampling drives towards high-likelihood states with increased secondary structure content.

*Less random noise in each denoising step, more **conservative***

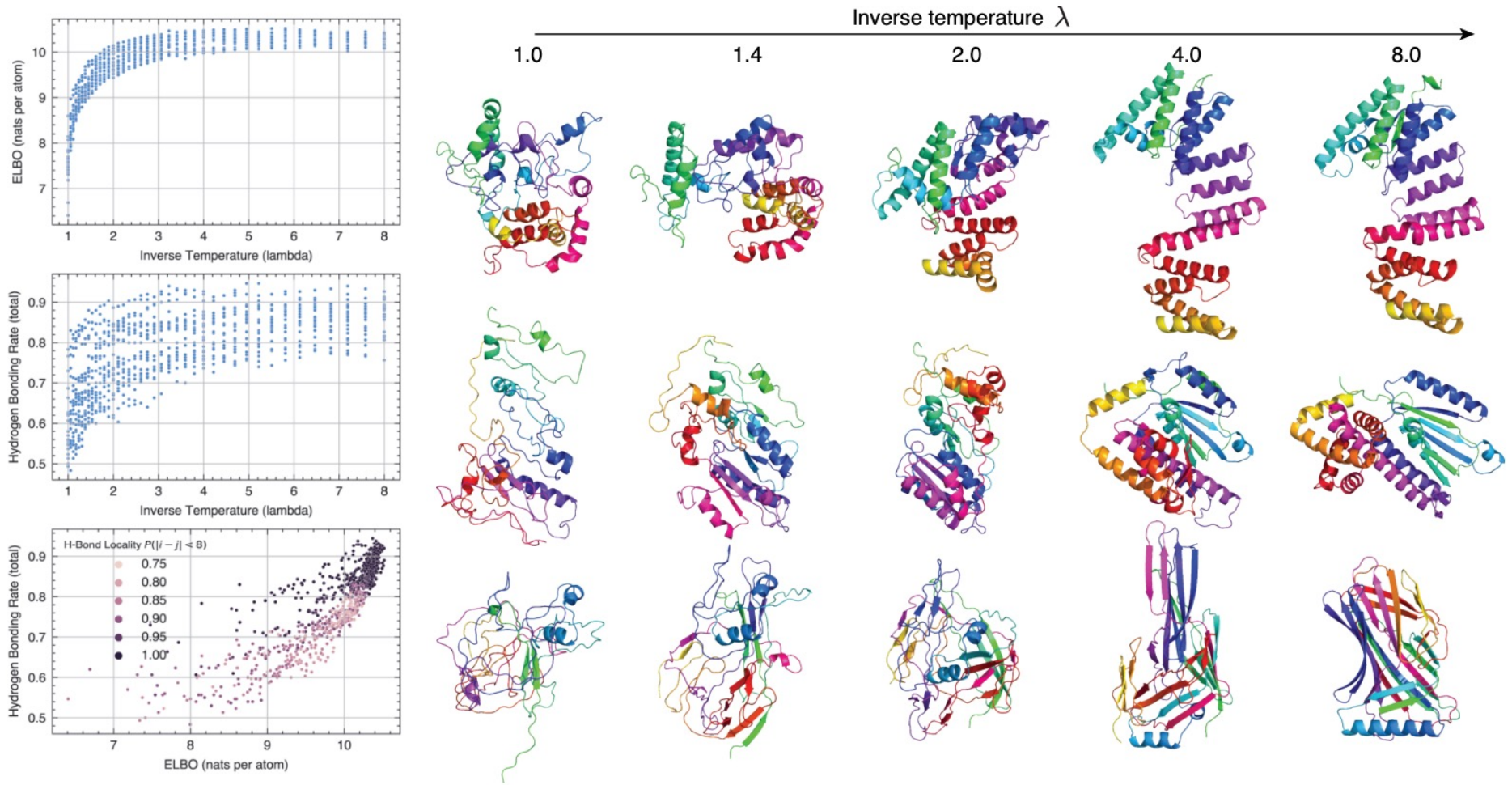
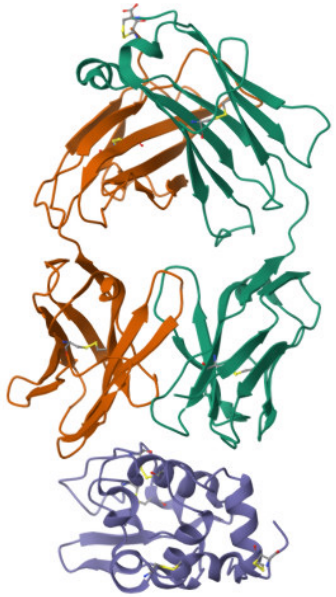


Fig. 2: Analysis of unconditional samples reveals diverse geometries that exhibit new higher-order structures and refold in silico.

Chroma generates 100,000 single-chain proteins and 20,000 protein complexes

ANTIBODY-ANTIGEN COMPLEX
(3HFM, PDB)



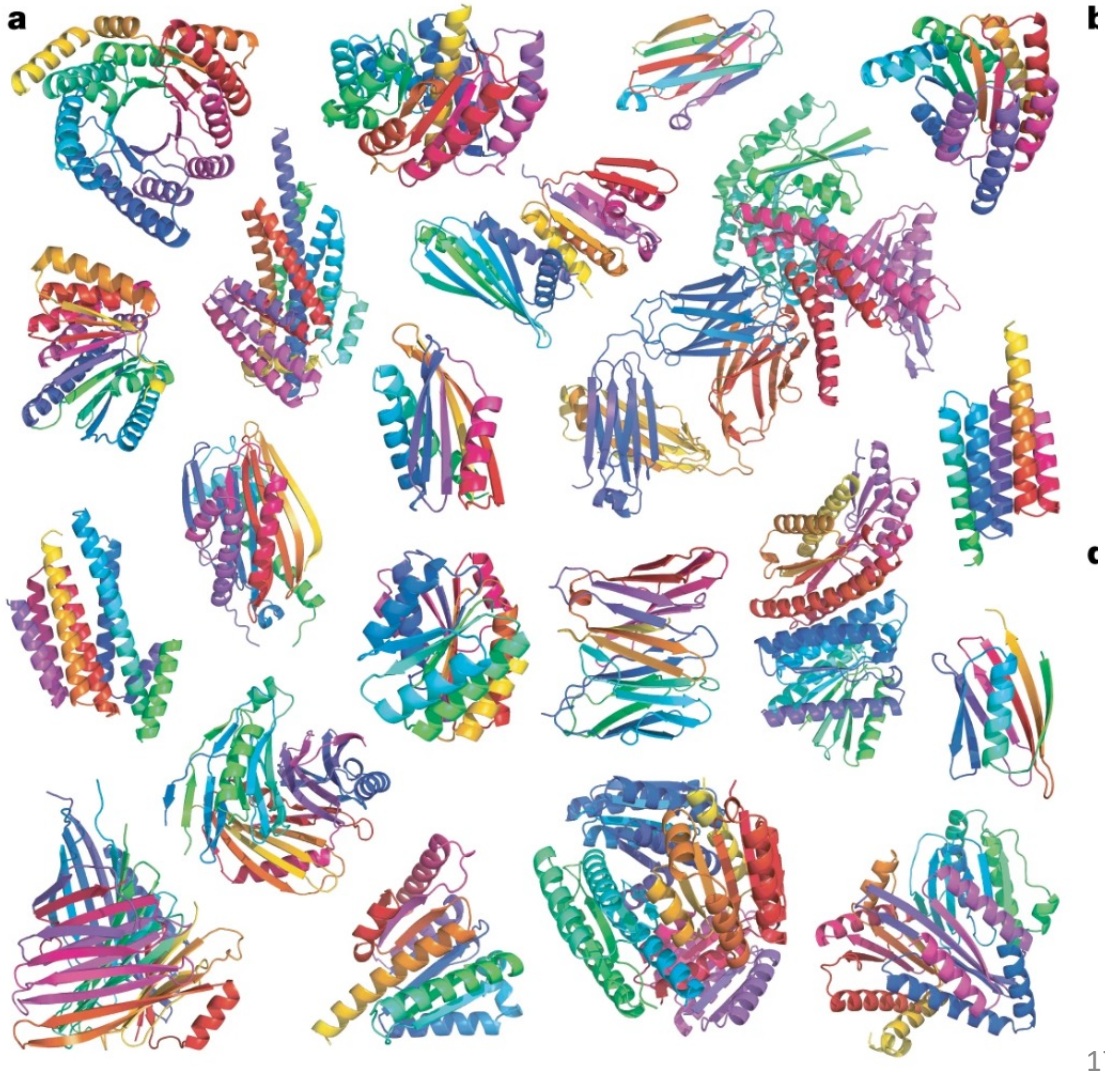
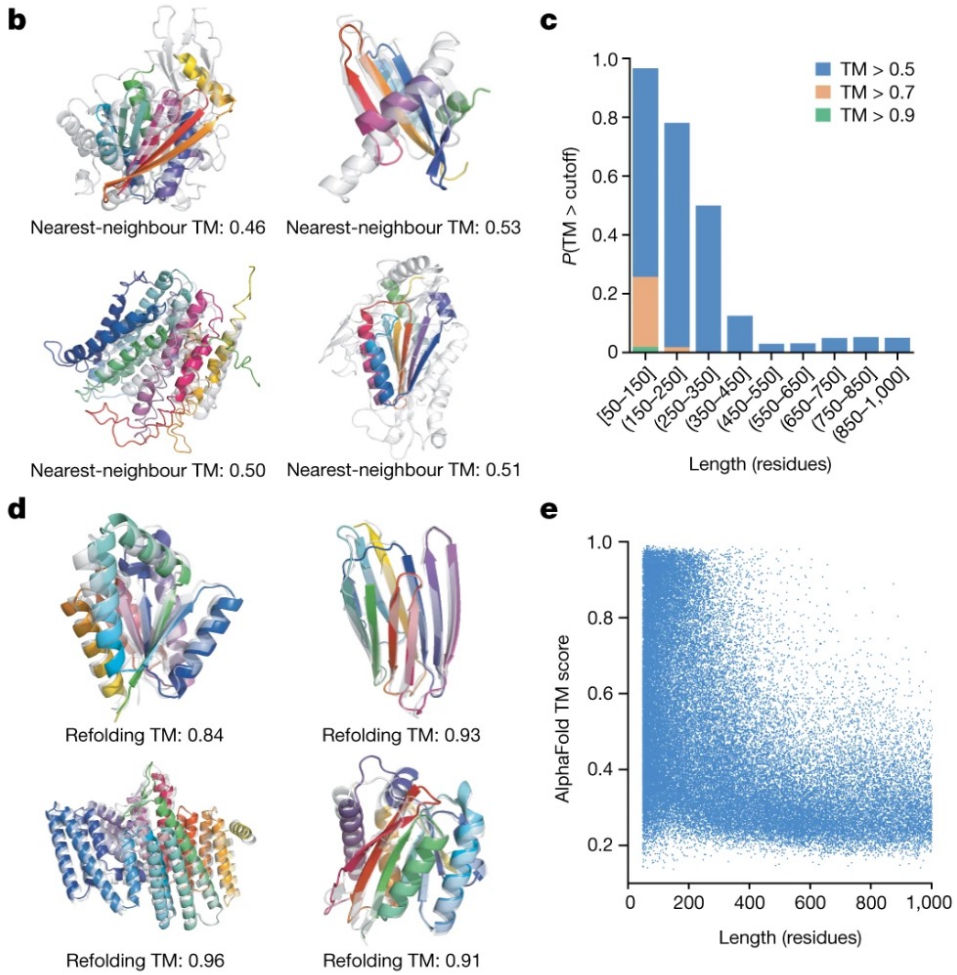
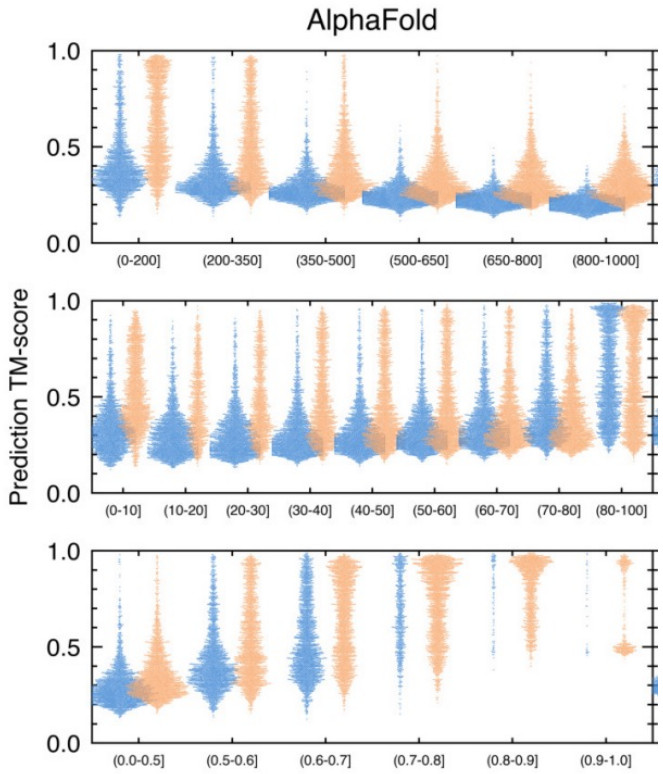


Fig. 2: Analysis of unconditional samples reveals diverse geometries that exhibit new higher-order structures and refold in silico.



Comparison with AlphaFold Predictions



Unseen structures at a frequency that increases sharply with length

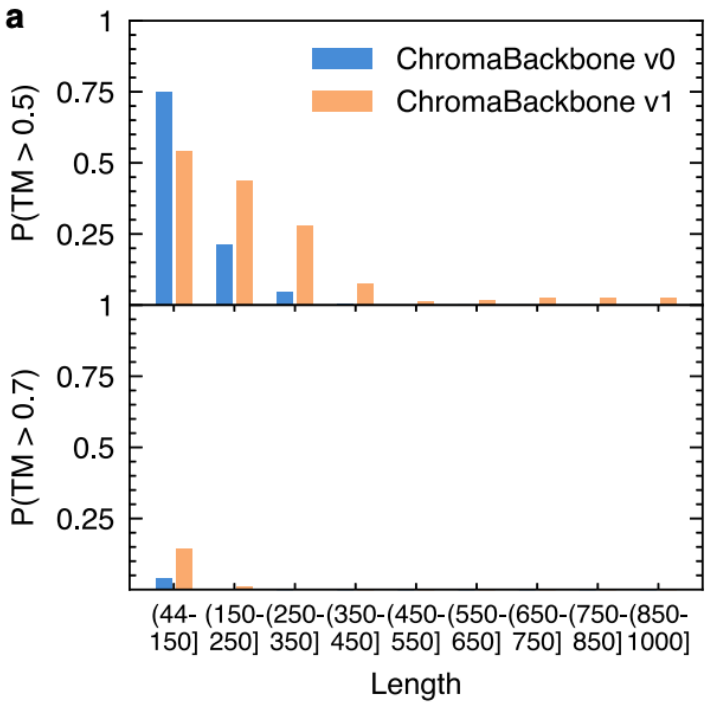


Fig. 3: Symmetry, substructure and shape conditioning enable geometric molecular programming.

Conditioners parameterize the protein design

Quasilinear computation time

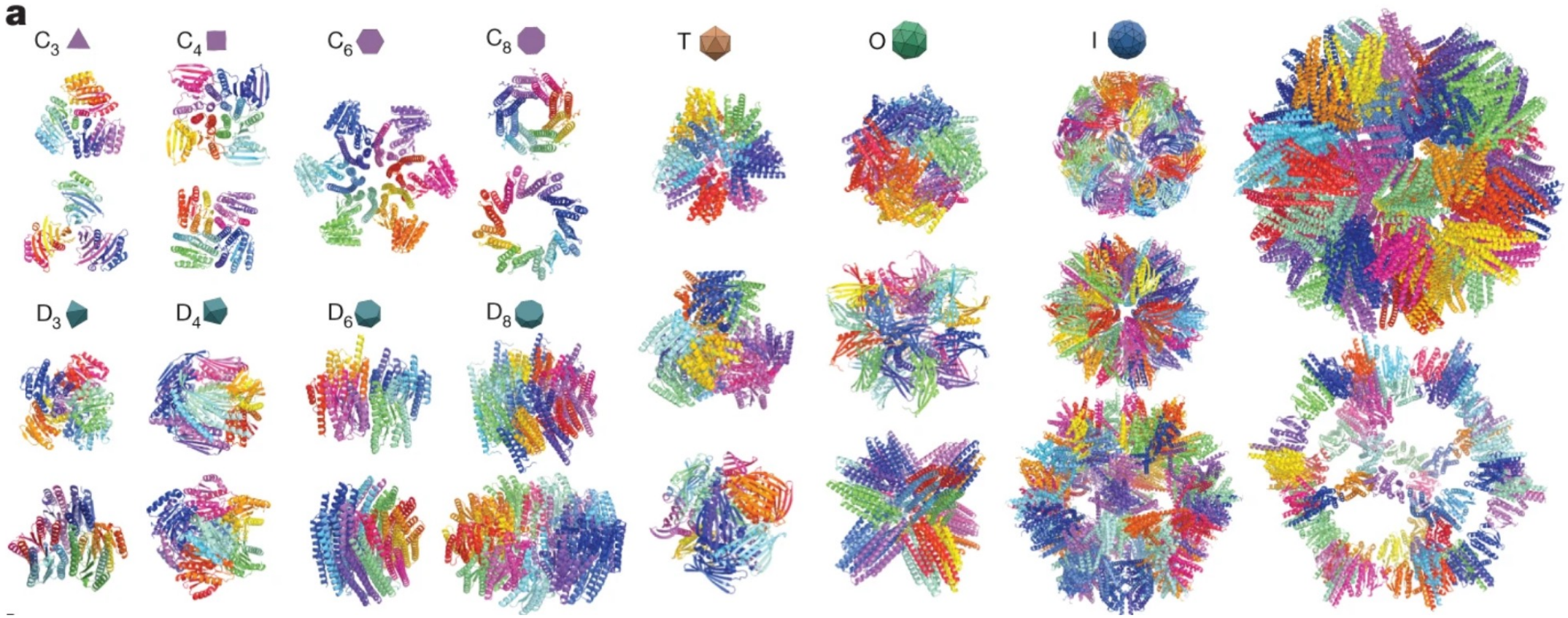


Fig. 3: Symmetry, substructure, and shape conditioning enable geometric molecular programming.

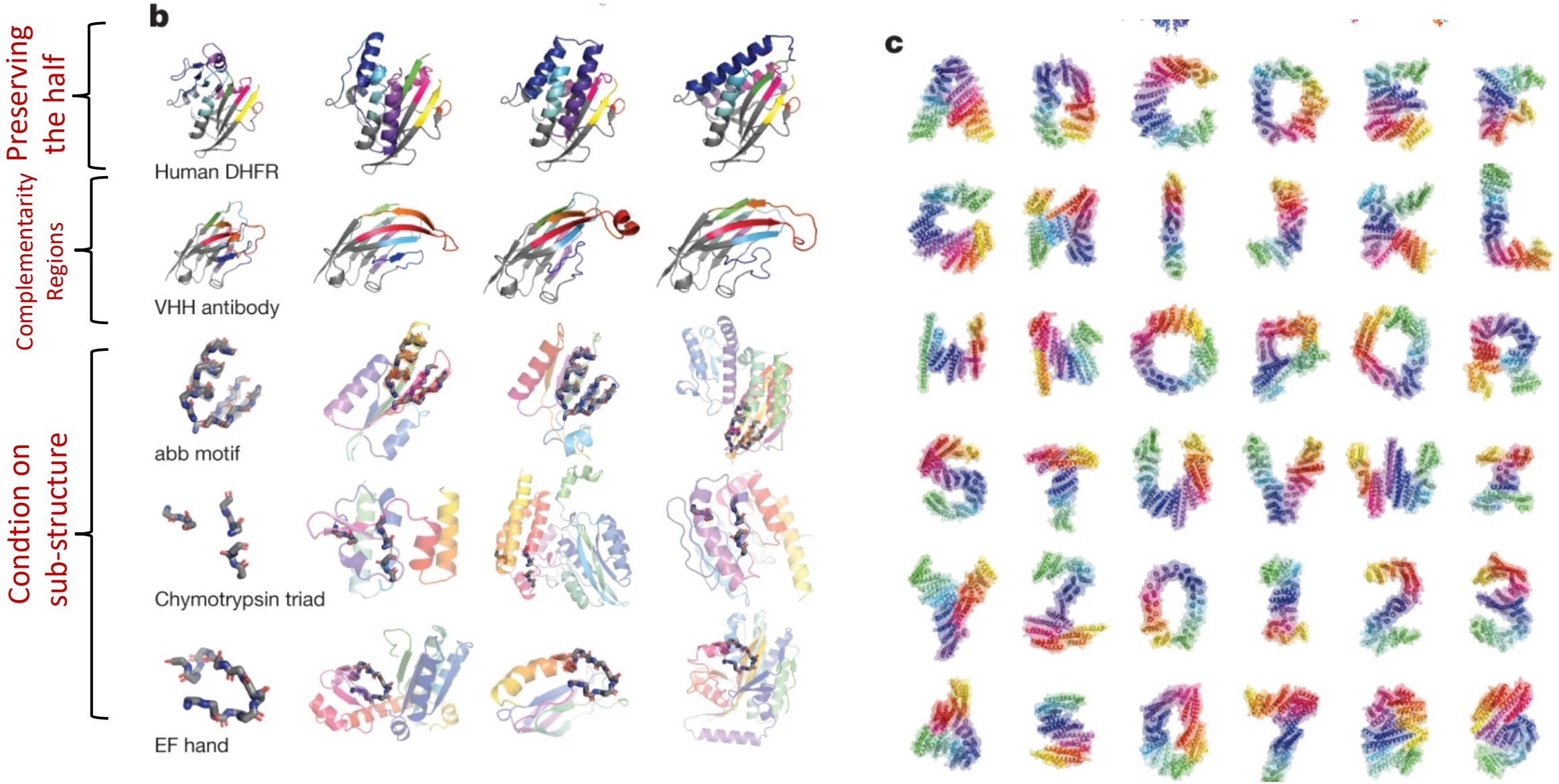


Fig. 4: Protein structure classifiers and caption models can bias the sampling process towards user-specified properties.

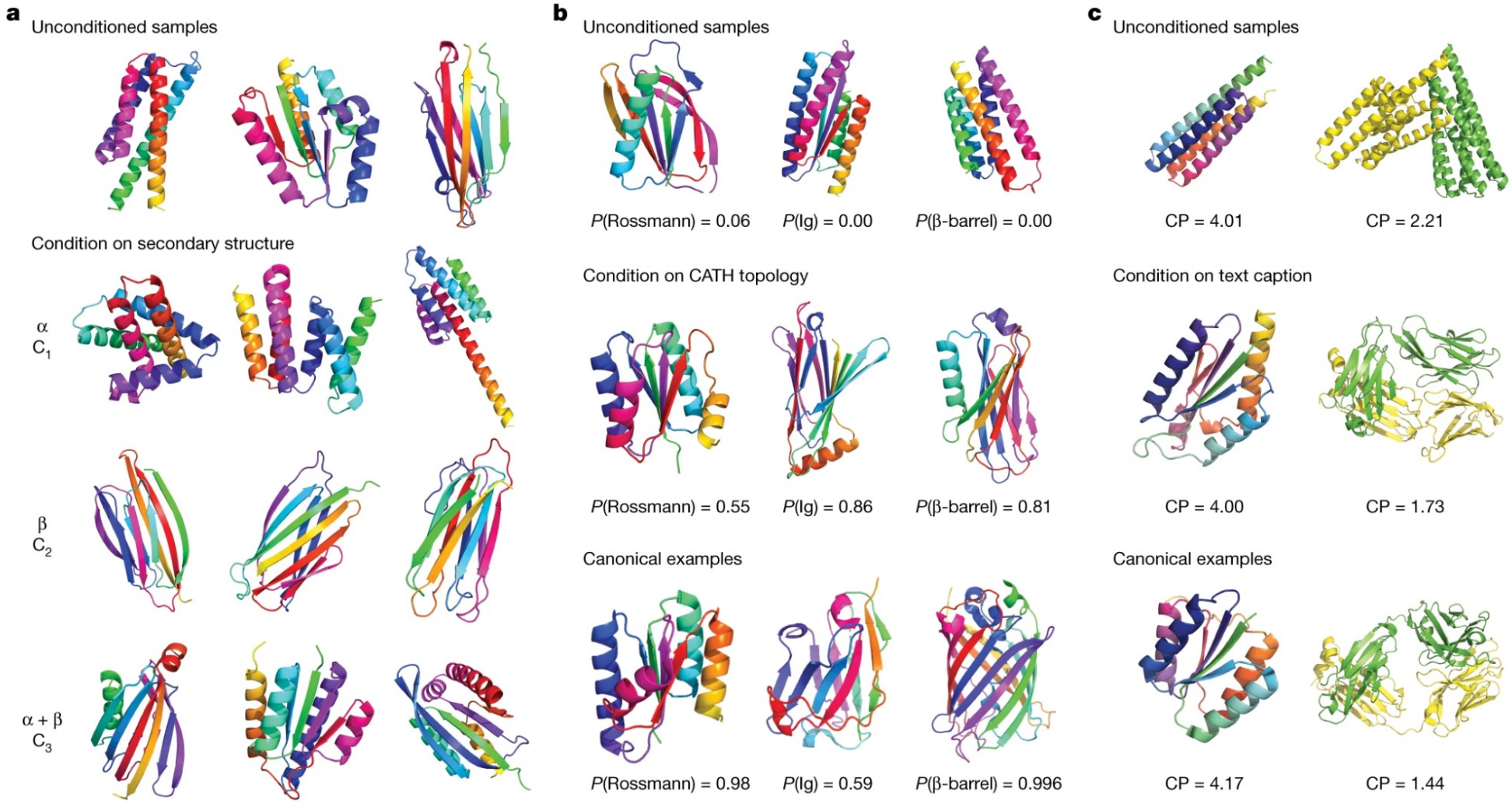
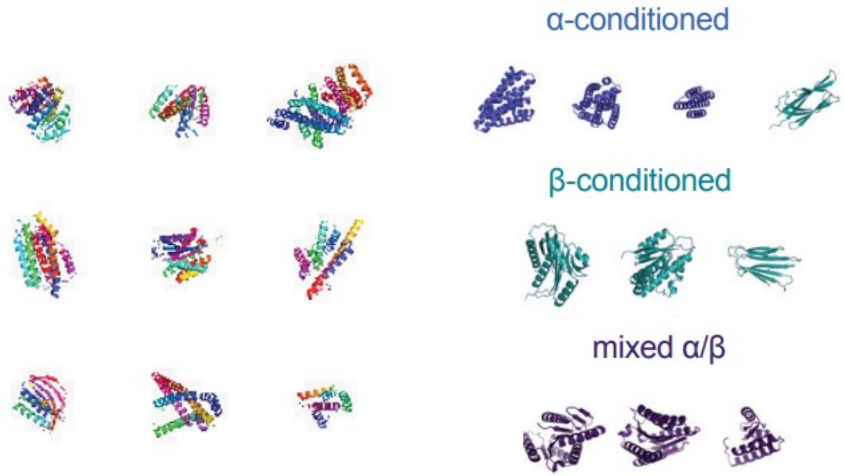
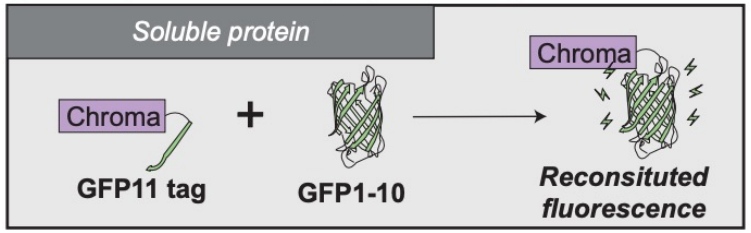


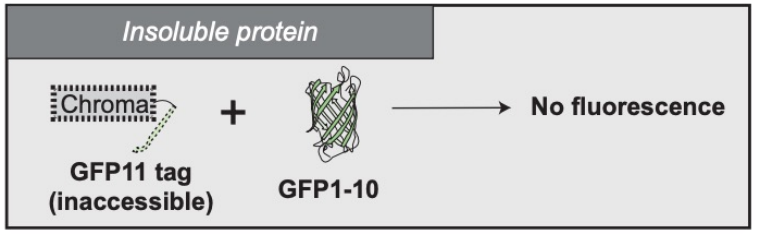
Fig. 5: Experimental validation of Chroma-designed proteins.

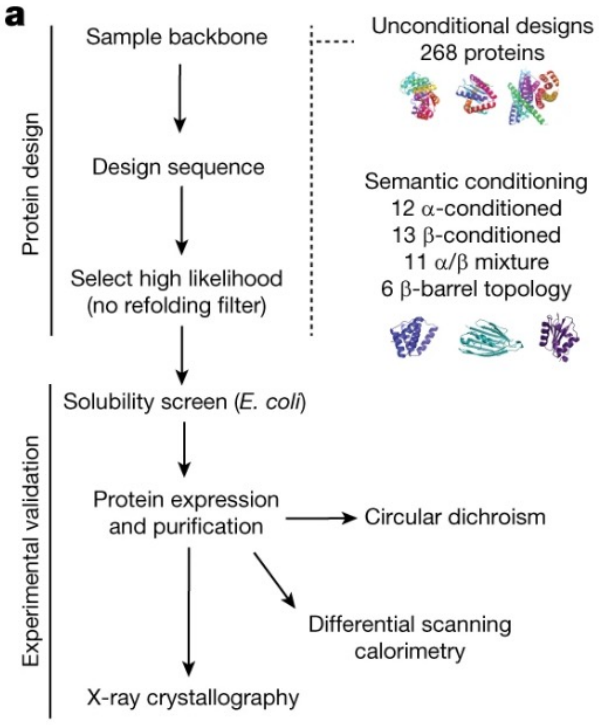
Secondary structure conditional designs.

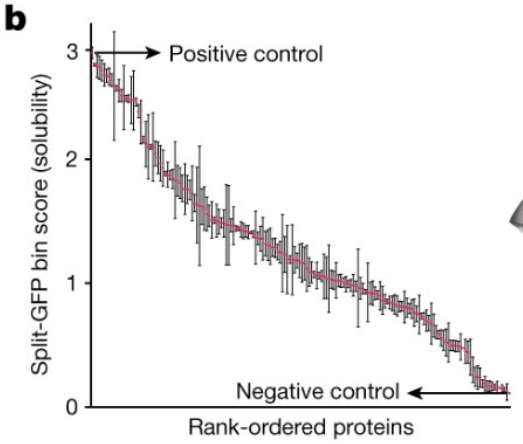


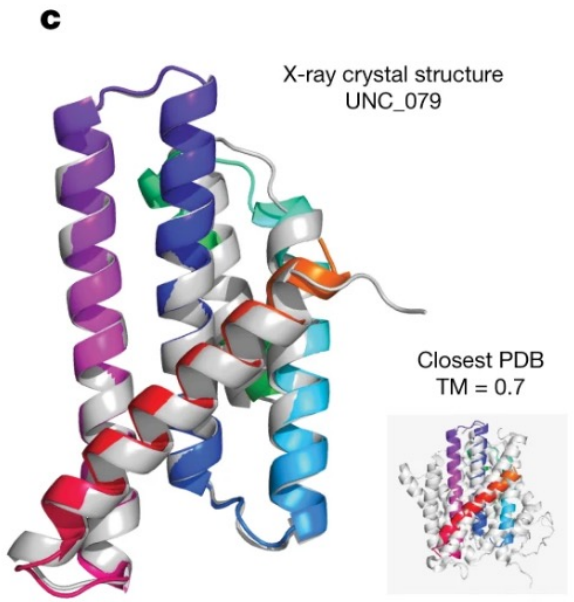
Split-GFP protein solubility reporter assay:











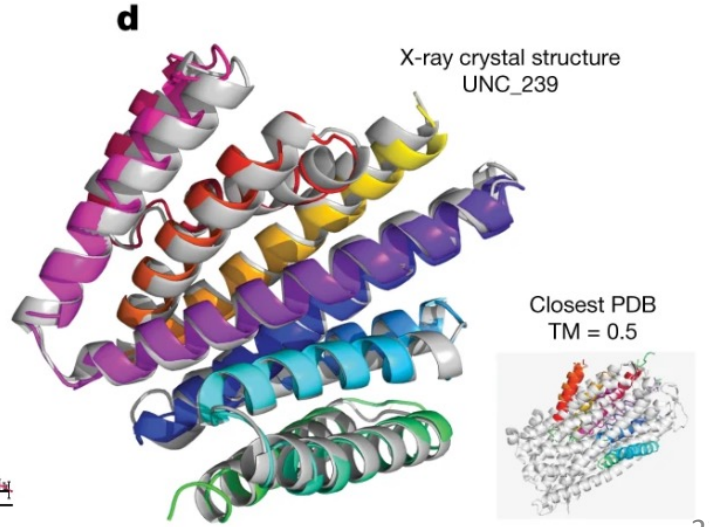
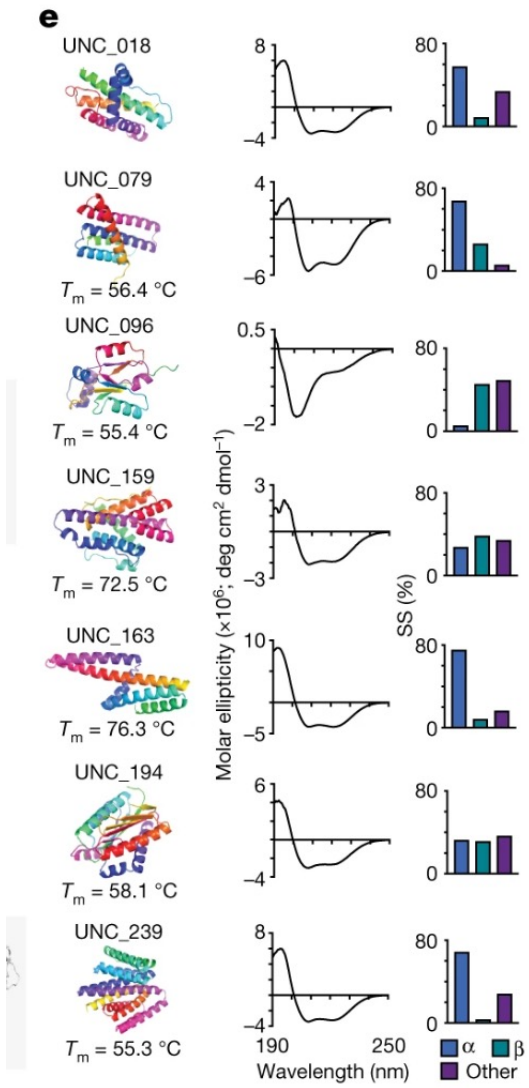
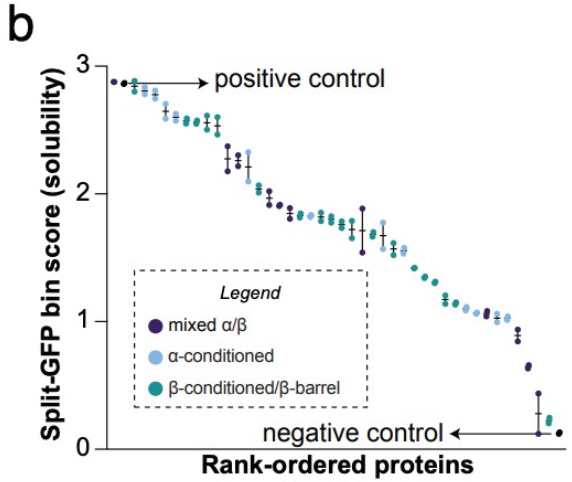
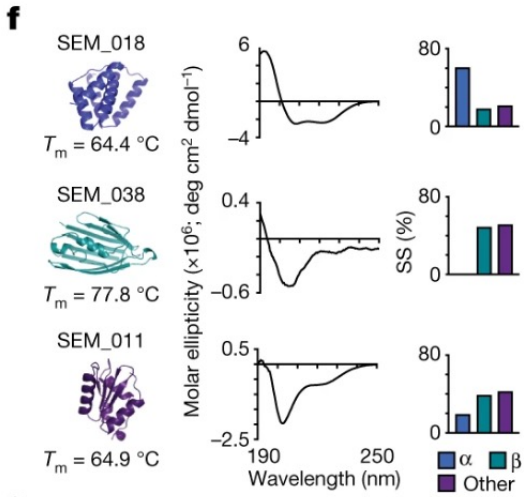


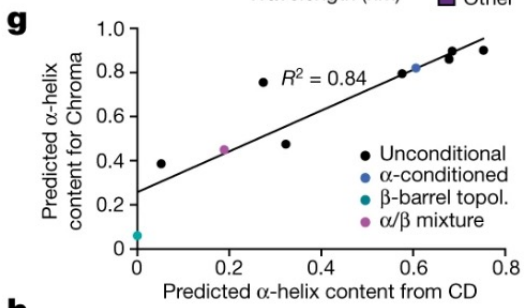
Fig. 5: Experimental validation of Chroma-designed proteins.

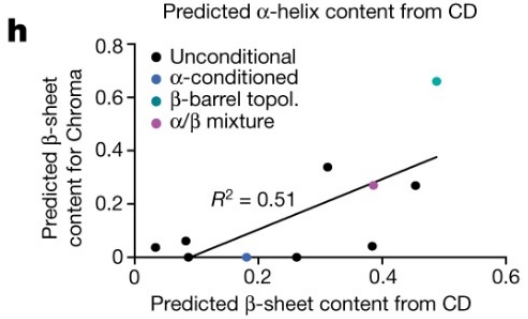


Secondary structure conditional designs.





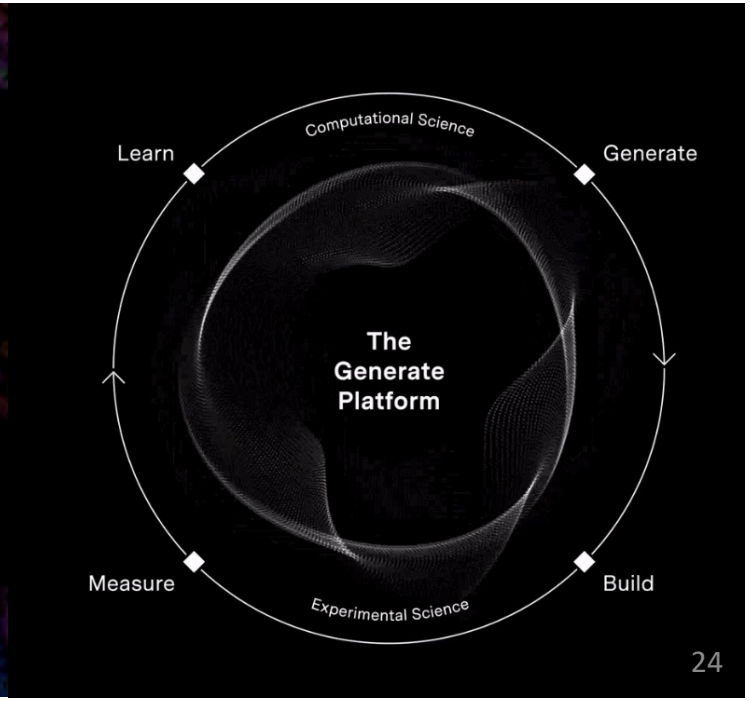




Conclusion Remarks

Chroma can generate protein sequences:

- *With arbitrary shapes and conditioned structures*
- *That is structurally stable and soluble*
- *While being computationally inexpensive*



Thank you!

ChromaBackbone Hyperparameters

Category	**Hyperparameter**	**Value in `ChromaBackbone v0`**	**Value in `ChromaBackbone v1`**
Diffusion Process	Covariance Model	Globular Monomer	Globular Complex
	Noise Schedule	Log-linear SNR (-7,13.5) [62]	Log-linear SNR (-7,13.5)
Graph Features	Node Features	Internal Coordinates	Internal Coordinates
	Edge Features	Atom distances, Atom directions, Chain distances, Transforms	Atom distances, Atom directions, Chain distances, Transforms
	Edges per Node, k	60	60
	Number of Nearest Neighbor Edges	20	20
	Number of Random Edges	40	40
	Random Edge Type	Inverse Cubic	Inverse Cubic
Graph Neural Network	Number of GNN layers	12	12
	Node Embedding Dimension	512	512
	Edge Embedding Dimension	256	256
	Node MLP Dimension	512	512
	Edge MLP Dimension	128	128
	Dropout p	0.1	0.1
Denoising Solver	Inter-residue Parameterization	Direct T_{ij} prediction	Update from $T_{ij}\,(\mathbf{x}_t)$
	Uncertainty Model	Isotropic (1-parameter)	Decoupled (2-parameter)
	Number of Iterations	3	10
	Post-Process Scaling	A	B
Loss Function	Likelihood Loss	ELBO	ELBO
	Auxilliary Losses	ELBO-weighted MSE	$\mathcal{D}_{\text{global}}$, $\mathcal{D}_{\text{fragment}}$, D_{ij} SE, $\hat{T}_{ij}$ SE
Total Number of Parameters		18.6M	18.6M
Total Number of Training Steps		1.6M	1.8M

The Loss Functions

- **ELBO** This is a pure likelihood loss, which is a weighted average squared error loss in whitened space together with additional additive terms to account for normalization and change of variables. It is measured in Nats per atom in Cartesian space and is comparable across different diffusion models.
- **+AuxLoss1** To the base ELBO loss, we add the ELBO-weighted unwhitened loss (Equation 1) that measures mean squared error in Cartesian space.
- **+AuxLoss2** To the base ELBO loss, we add the SSNR-weighted global MSE loss, the SSNR-weighted 7mer fragment MSE loss, the Distance MSE loss, and the Inter-residue Transform MSE loss.