Diffusion modeling of protein backbones for the motif-scaffolding problem

Brian Trippe & Jason Yim

COLUMBIA UNIVERSITY

INSTITUTE FOR Protein Design
UNIVERSITY of WASHINGTON

Computational Protein Design Workflow

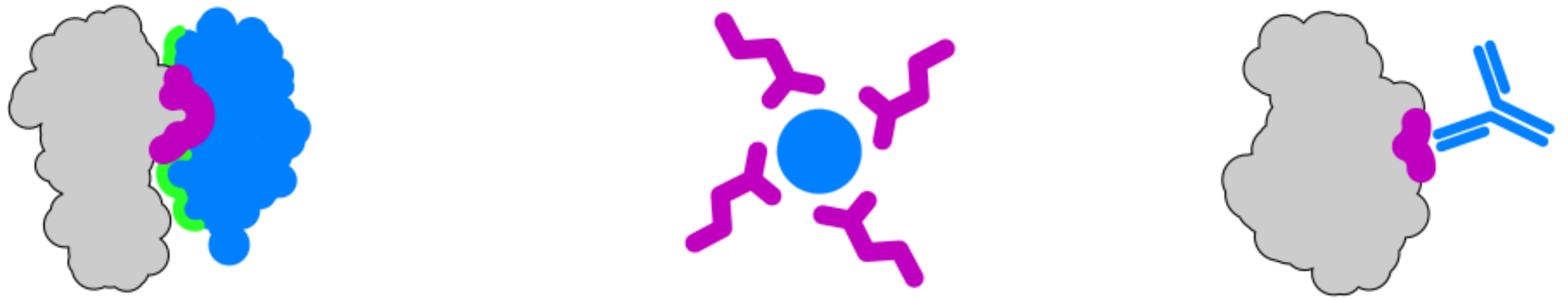
Desired Function
(e.g. binding)
Motif
Identification
Functional "**Motif**"
Motif-Scaffolding
Designed "**Scaffold**"

[Figure credit: Doug
Tischer & David Juergens]

Computational Protein Design Workflow

Desired Function
(e.g. binding)
Motif
Identification
Functional "**Motif**"
Motif-Scaffolding
Designed "**Scaffold**"

Motifs can have various functions and sources



Protein interaction interfaces
(via fragment docking)

Catalytic & metal-binding sites
(quantum chemistry)

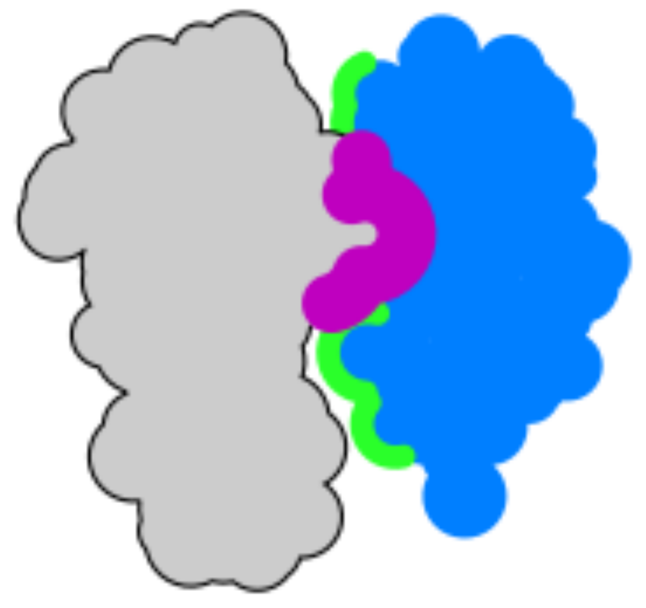
Epitope presentation
(Native interface)

[Figure credit: Doug Tischer & David Juergens]

Computational Protein Design Workflow

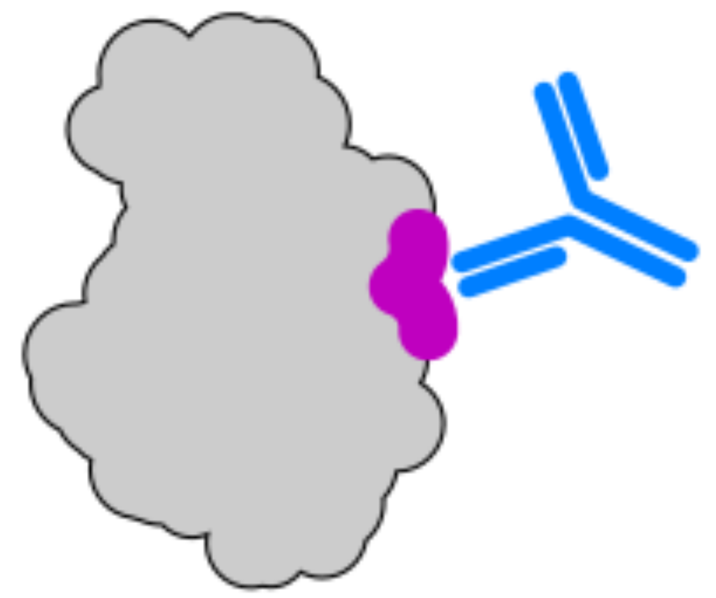
This talk
Desired Function
(e.g. binding)
Motif
Identification
Functional "**Motif**"
Motif-Scaffolding
Designed "**Scaffold**"

Motifs can have various functions and sources



Protein interaction interfaces
(via fragment docking)

Catalytic & metal-binding sites
(quantum chemistry)



Epitope presentation
(Native interface)

[Figure credit: Doug Tischer & David Juergens]

What makes this hard?

What makes this hard?

Post-AlphaFold, protein design is "guess & check"

What makes this hard?

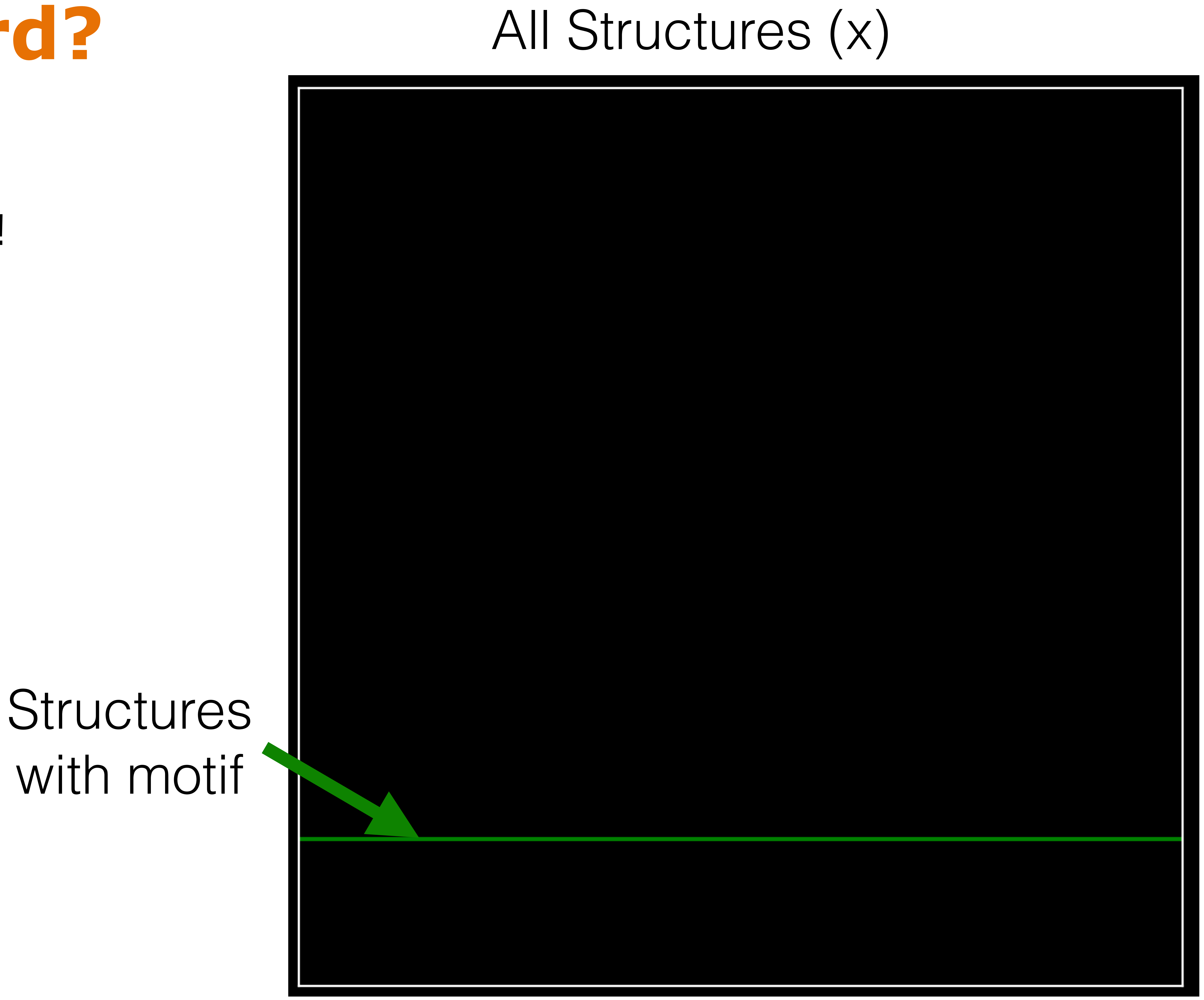
Post-AlphaFold, protein design is "guess & check"

- Naive guessing?

What makes this hard?

Post-AlphaFold, protein design is "guess & check"

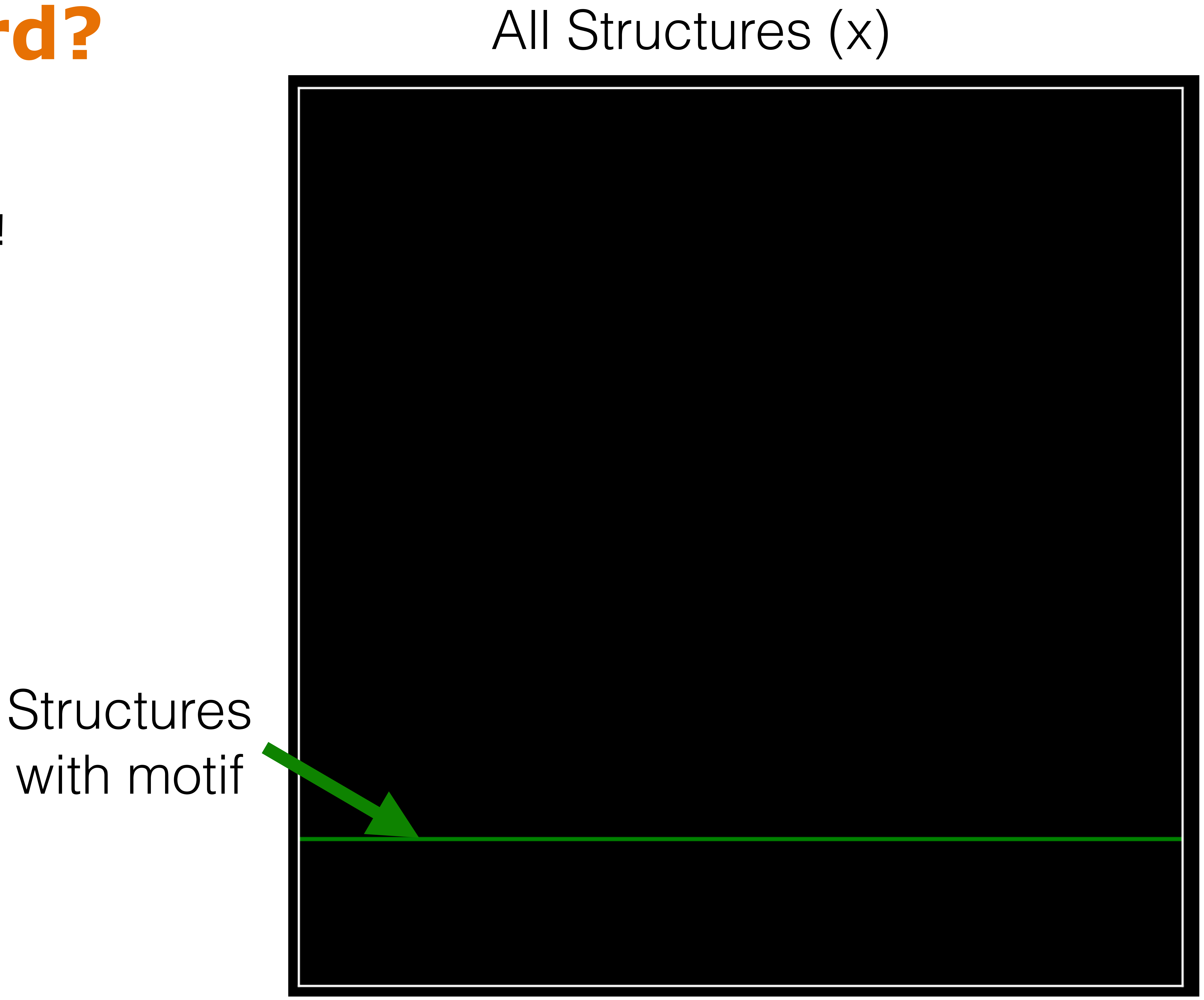
- ~~Naive guessing?~~ ~20^{100} sequences!



What makes this hard?

Post-AlphaFold, protein design is "guess & check"

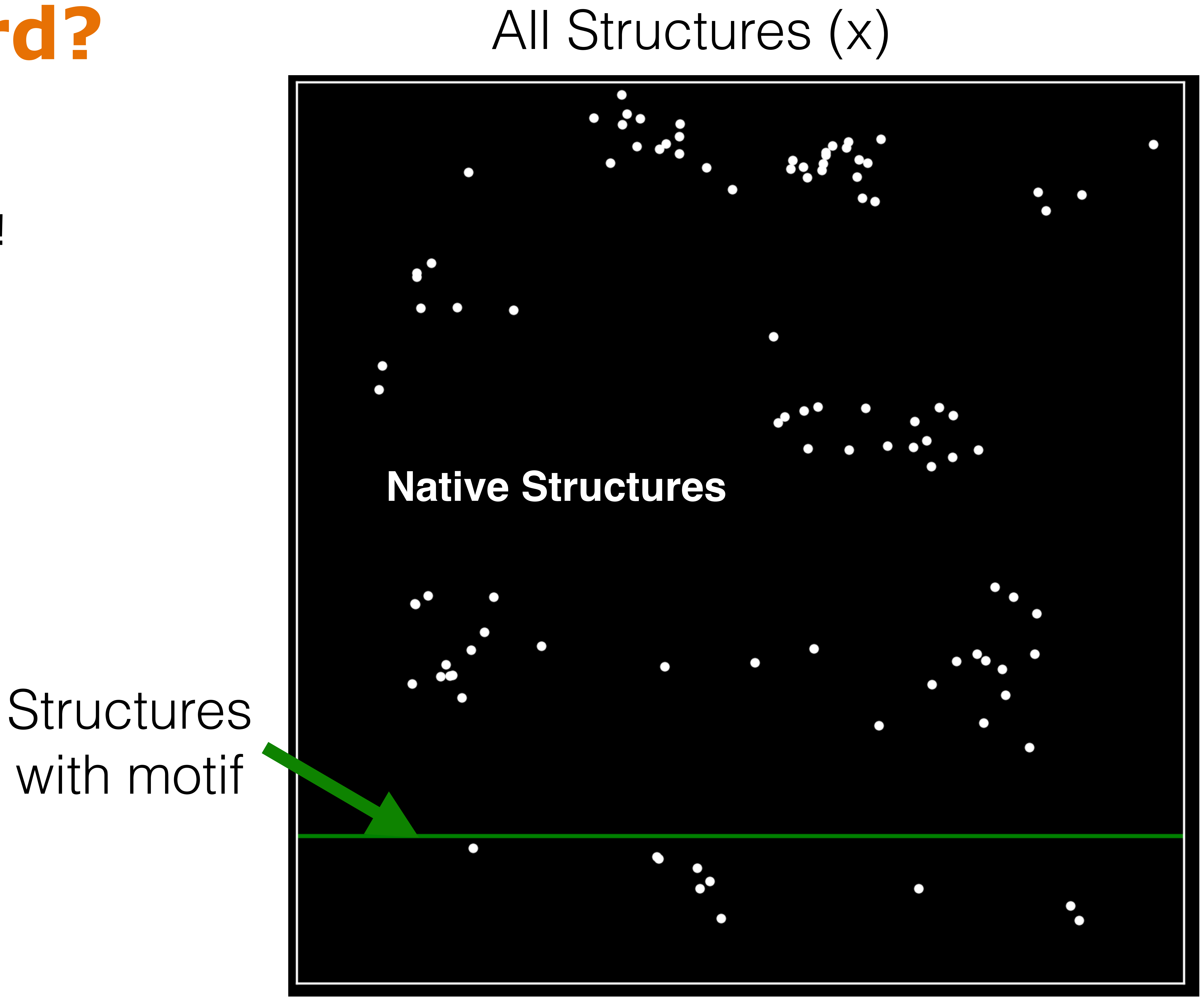
- ~~Naive guessing?~~ ~20^{100} sequences!
- Native structures?



What makes this hard?

Post-AlphaFold, protein design is "guess & check"

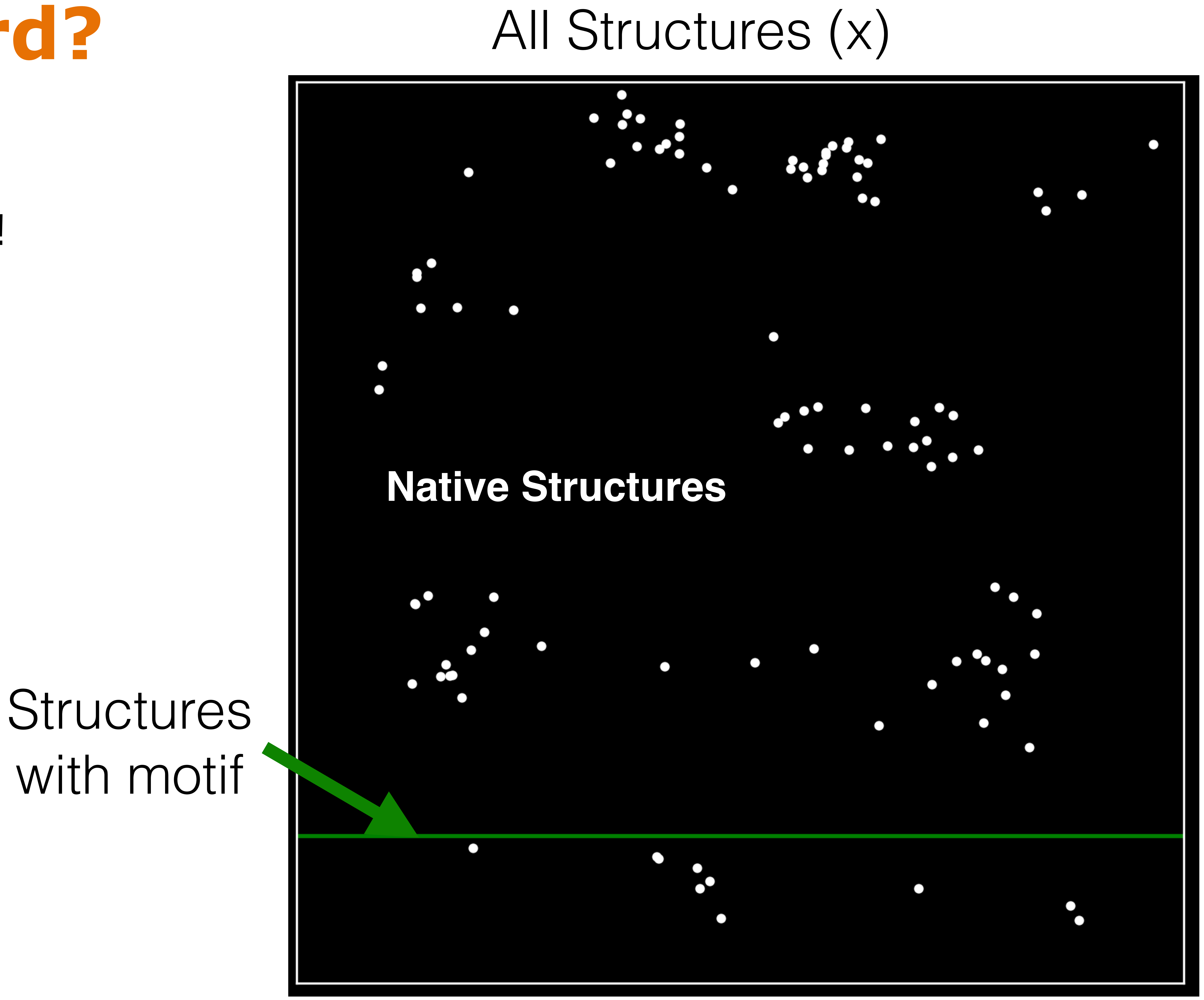
- ~~Naive guessing?~~ ~20^{100} sequences!
- Native structures? Too sparse



What makes this hard?

Post-AlphaFold, protein design is "guess & check"

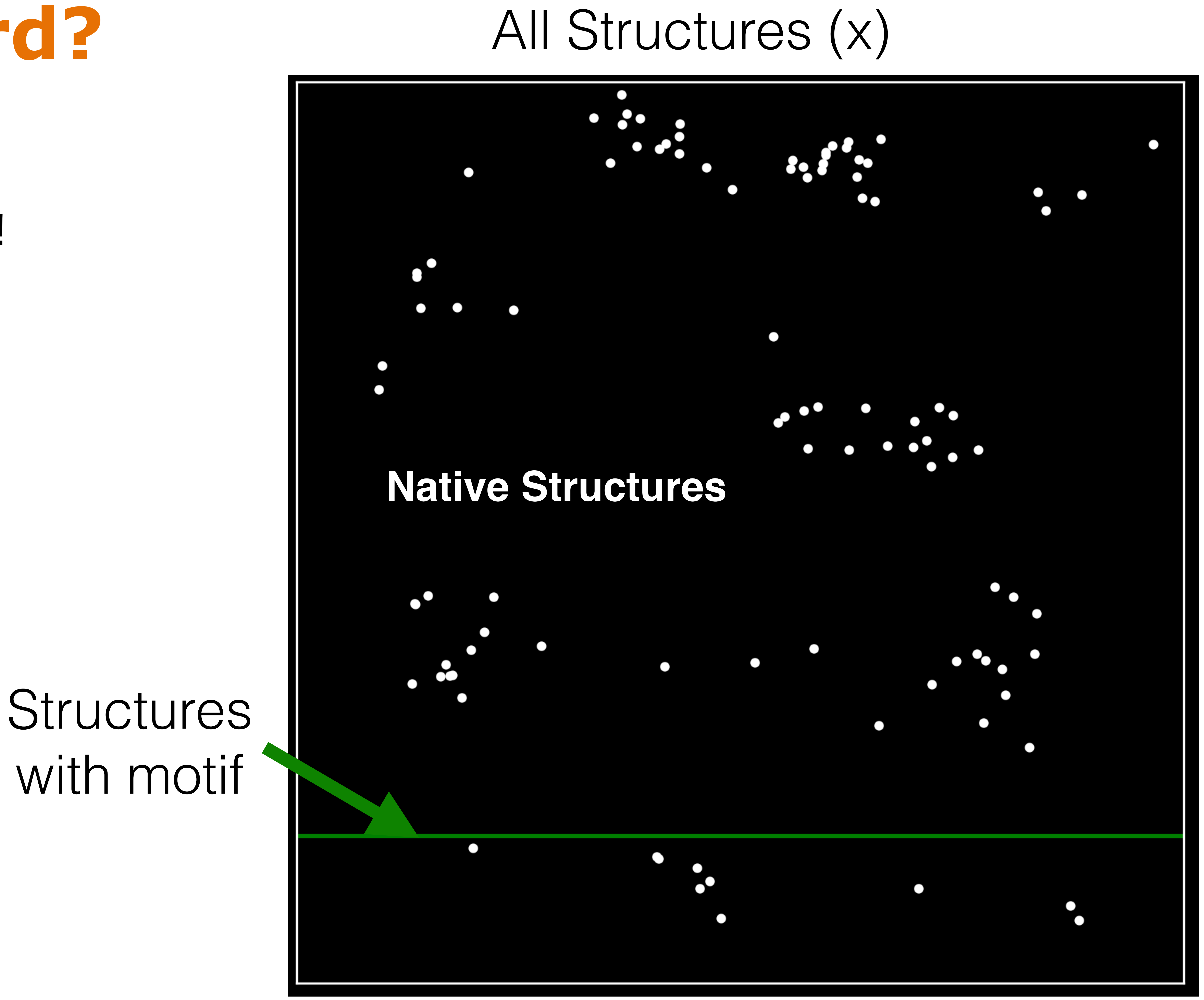
- ~~Naive guessing?~~ ~20^{100} sequences!
- Native structures? Too sparse
- Existing ML tools?



What makes this hard?

Post-AlphaFold, protein design is "guess & check"

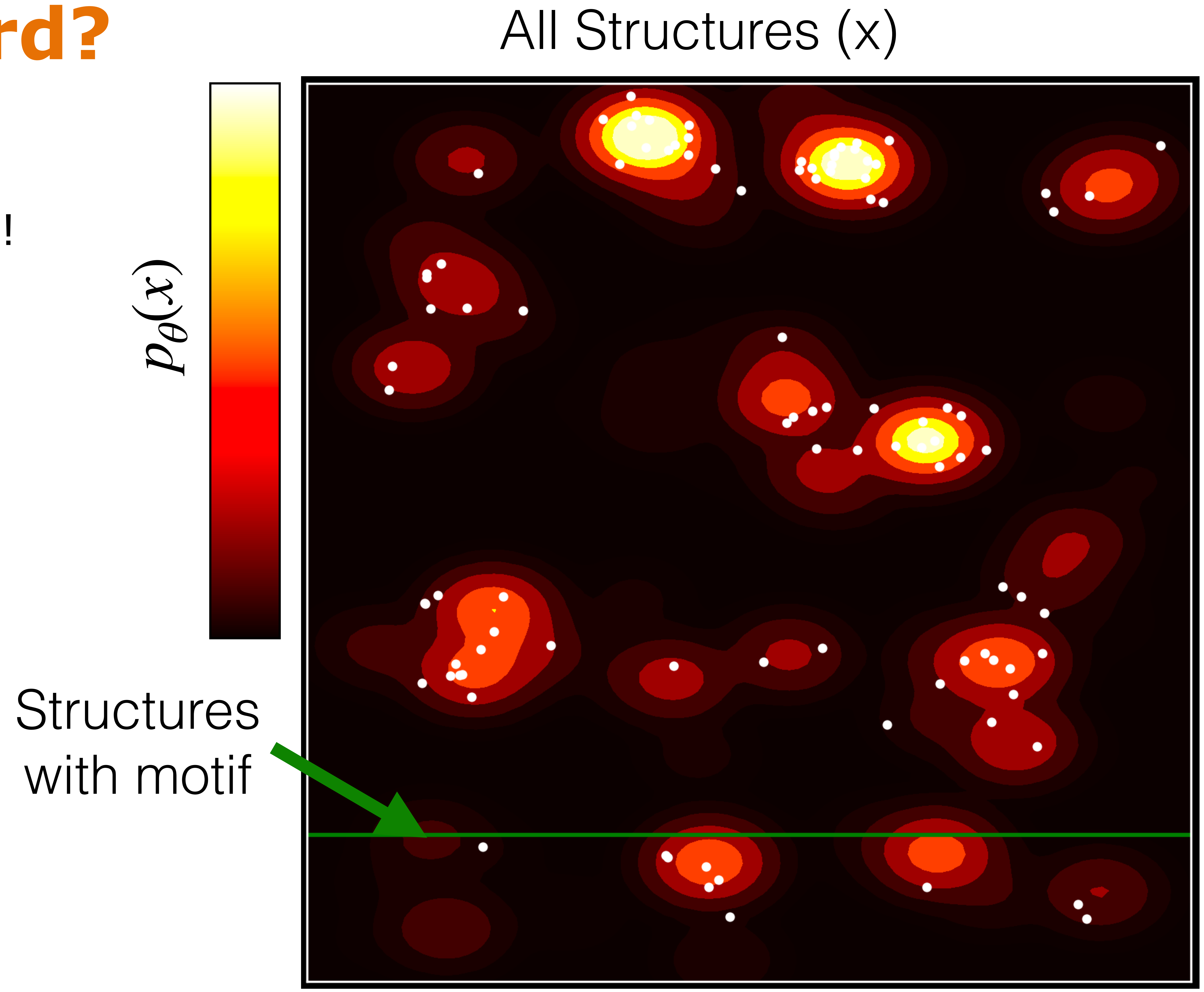
- ~~Naive guessing?~~ ~20^{100} sequences!
- Native structures? Too sparse
- Existing ML tools? Low diversity, high compute cost



What makes this hard?

Post-AlphaFold, protein design is "guess & check"

- ~~Naive guessing?~~ ~20^{100} sequences!
- Native structures? Too sparse
- Existing ML tools? Low diversity, high compute cost



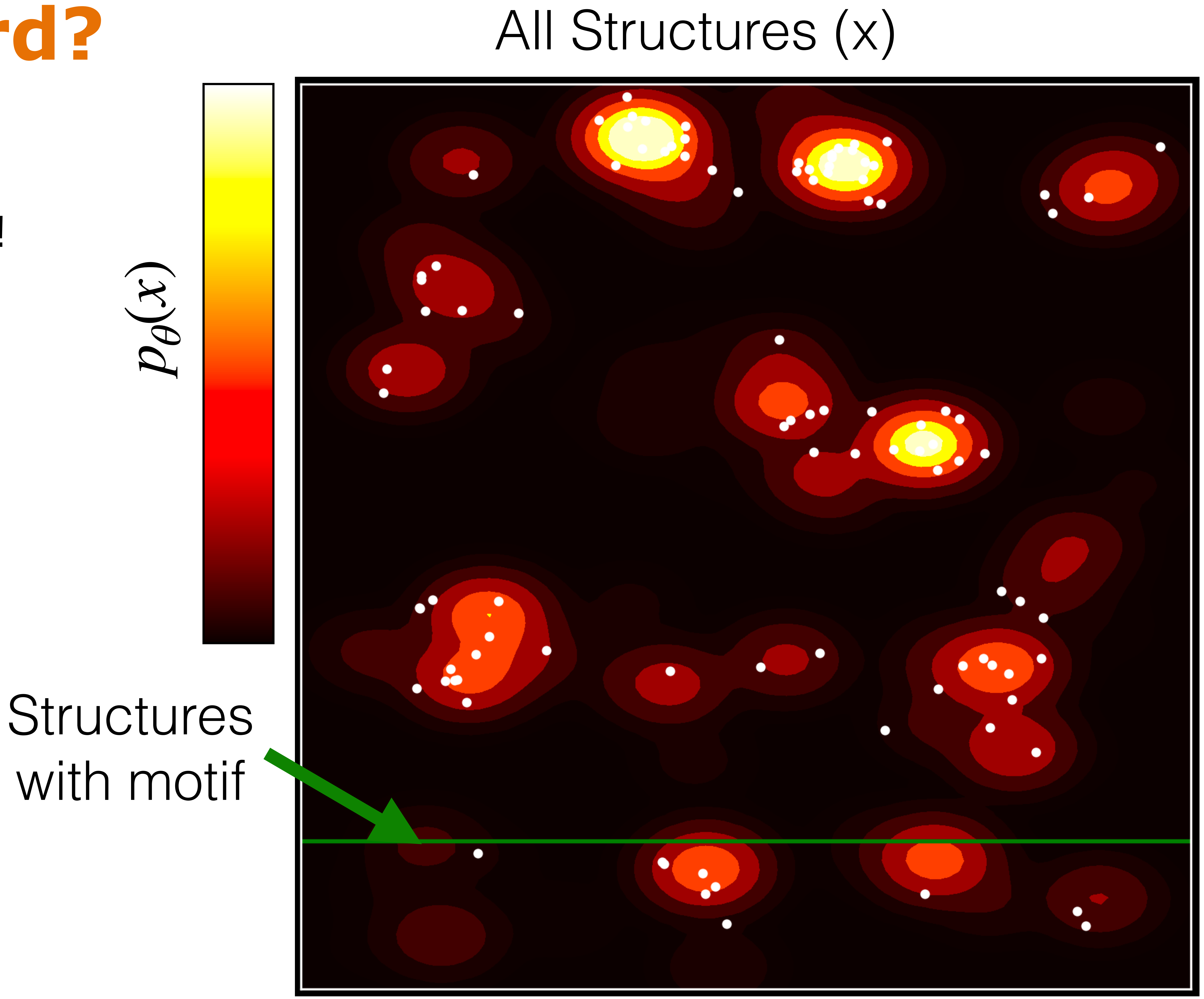
What makes this hard?

Post-AlphaFold, protein design is "guess & check"

- ~~Naive guessing?~~ ~20^{100} sequences!
- Native structures? Too sparse
- Existing ML tools? Low diversity, high compute cost

Our Approach:

1. Learn model of structure, $p_\theta(x)$, from native proteins



What makes this hard?

Post-AlphaFold, protein design is "guess & check"

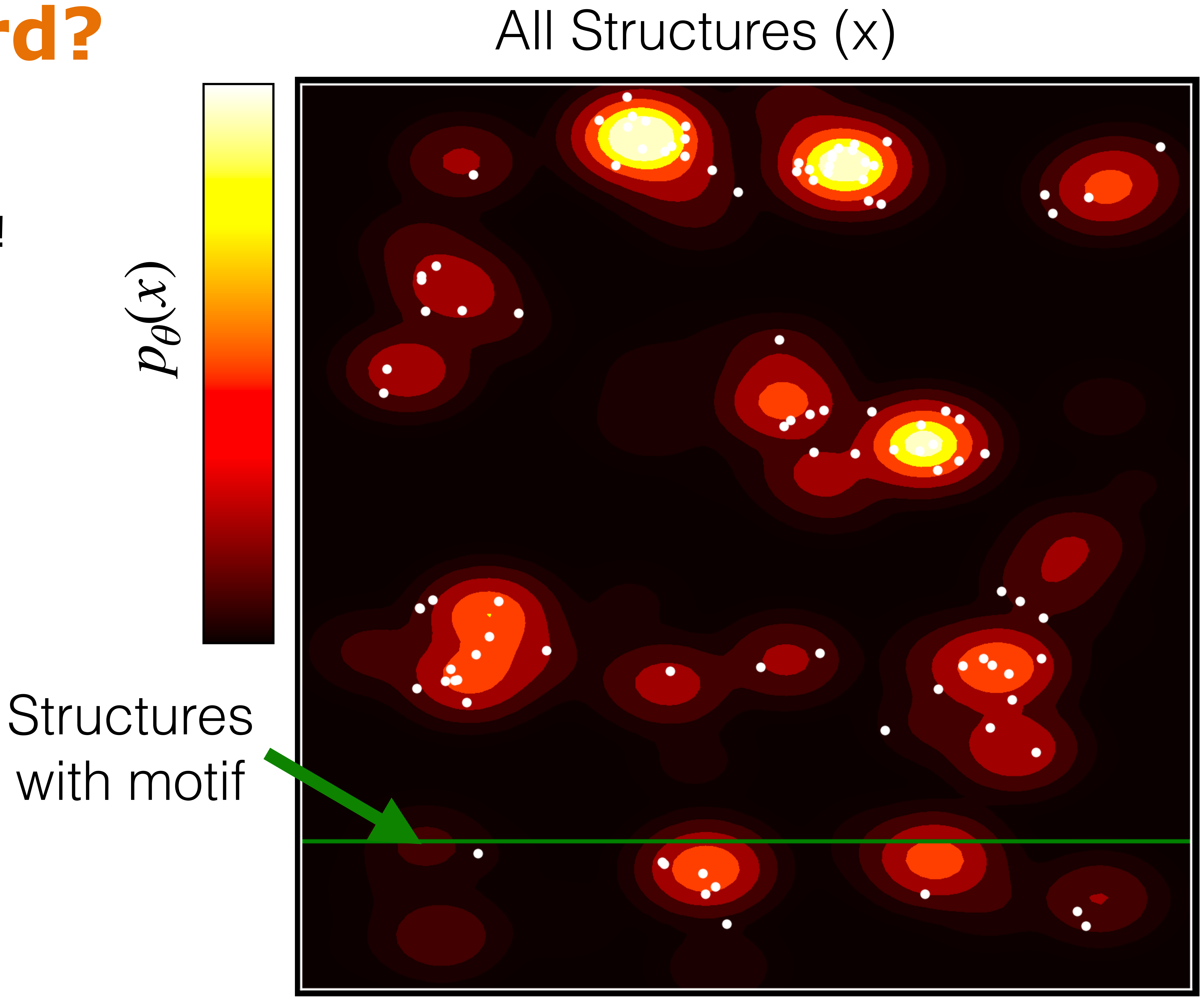
- ~~Naive guessing?~~ ~20^{100} sequences!
- Native structures? Too sparse
- Existing ML tools? Low diversity, high compute cost

Our Approach:

1. Learn model of structure, $p_\theta(x)$, from native proteins

2. Sample scaffolds given motif:

- Partition $x = [x_M, x_S]$
  Motif **S**caffold
- Draw $x_S \sim p_\theta(x_S \mid x_M)$



What makes this hard?

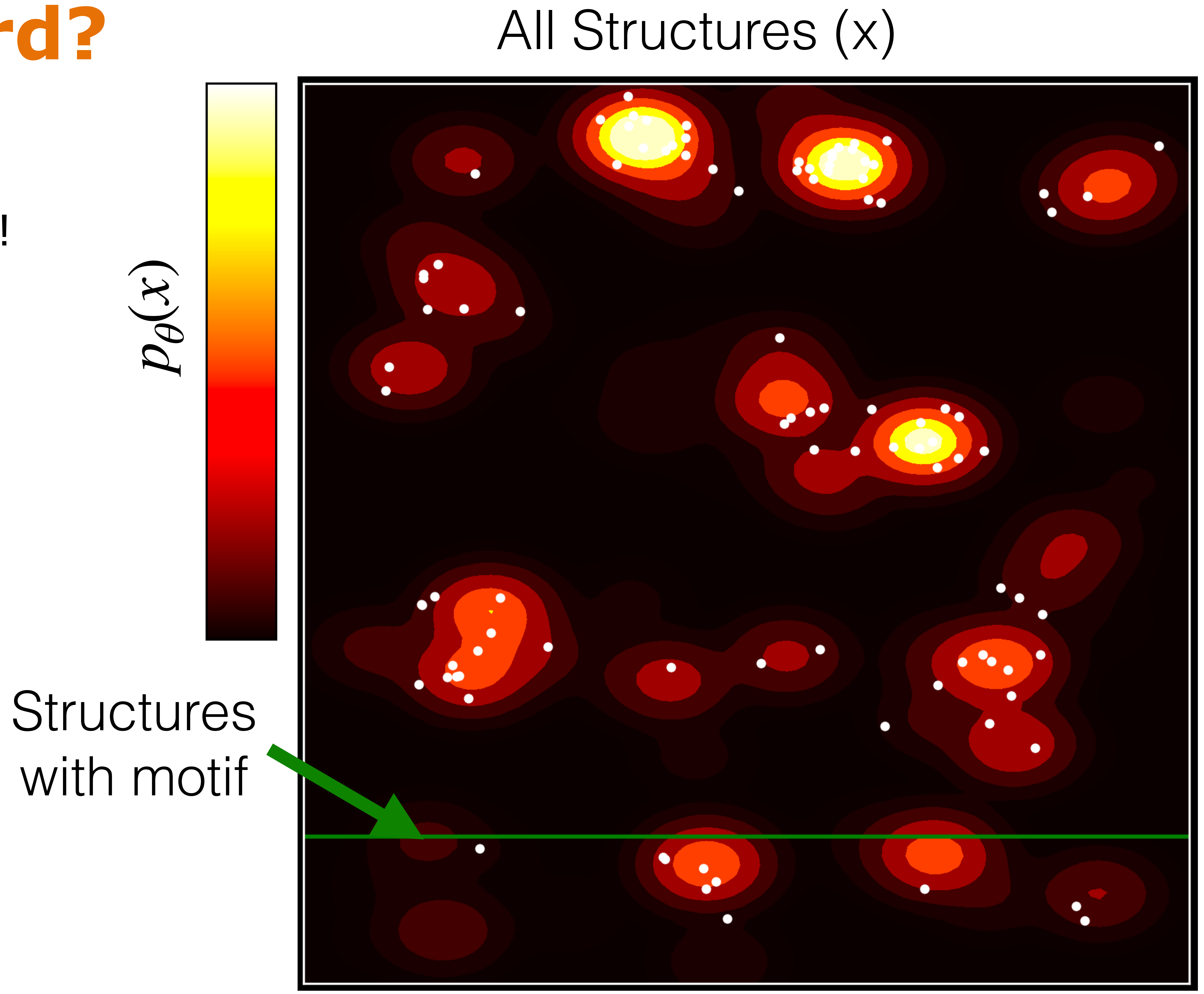
Post-AlphaFold, protein design is "guess & check"

- ~~Naive guessing?~~ ~20^{100} sequences!
- Native structures? Too sparse
- Existing ML tools? Low diversity, high compute cost

Our Approach:

1. Learn model of structure, $p_\theta(x)$, from native proteins

2. Sample scaffolds given motif:
   - Partition $x = [x_M, x_S]$
     Motif **S**caffold
   - Draw $x_S \sim p_\theta(x_S \mid x_M)$



Key Tool: Diffusion generative models & sequential Monte Carlo

What makes this hard?

Post-AlphaFold, protein design is "guess & check"

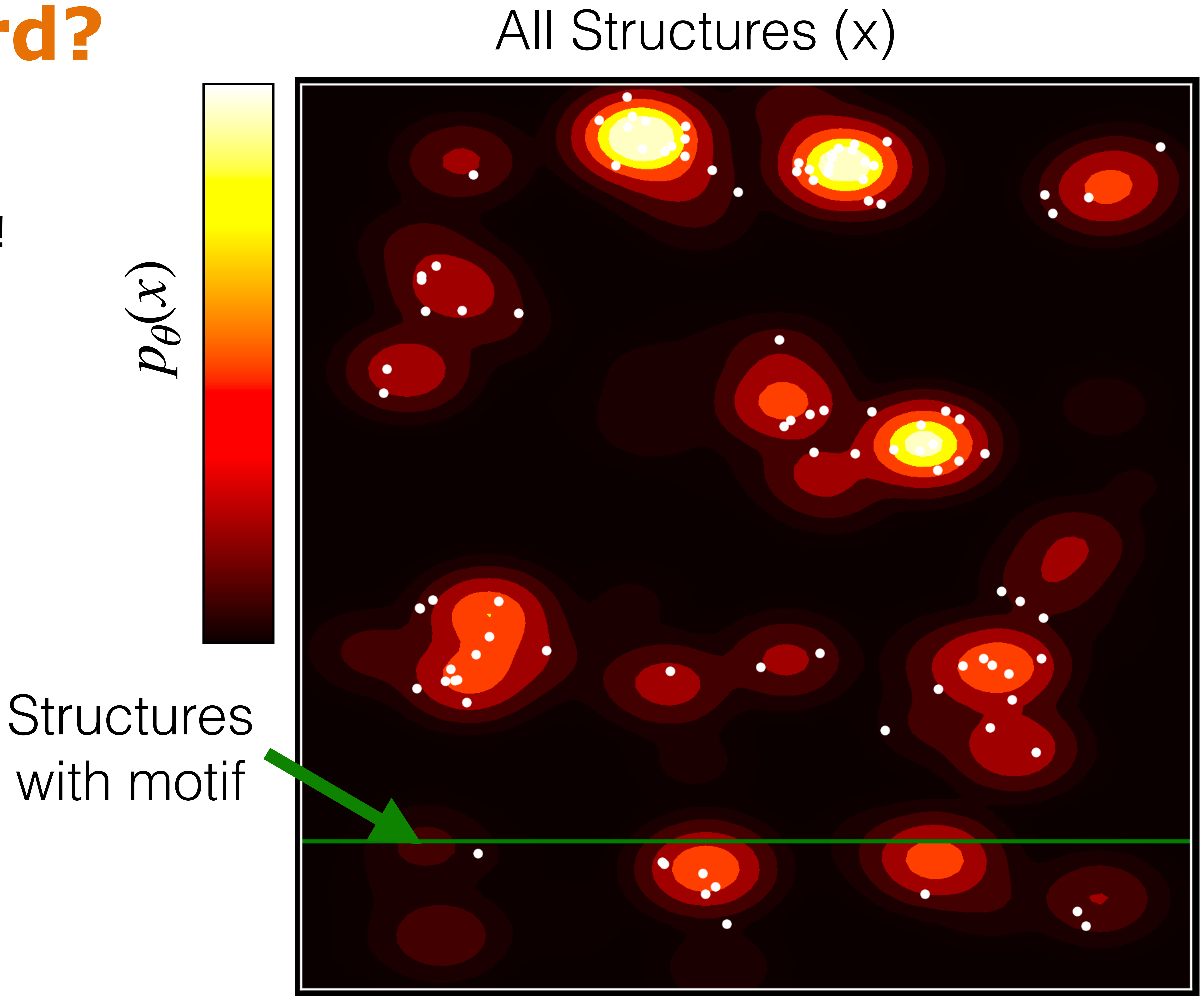
- ~~Naive guessing?~~ ~20^{100} sequences!
- Native structures? Too sparse
- Existing ML tools? Low diversity, high compute cost

Our Approach:

1. Learn model of structure, $p_\theta(x)$, from native proteins

2. Sample scaffolds given motif:

- Partition $x = [x_M, x_S]$ **M**otif **S**caffold
- Draw $x_S \sim p_\theta(x_S \mid x_M)$



Key Tool: Diffusion generative models & sequential Monte Carlo

We show: Methods with potential to build long, diverse scaffolds

Roadmap

Roadmap

Learning $p_\theta(x)$ [ProtDiff]

- Diffusion generative modeling background
- Adapting diffusion for protein backbones
- Model performance and limitations

Roadmap

Learning $p_\theta(x)$ **[ProtDiff]**

- Diffusion generative modeling background
- Adapting diffusion for protein backbones
- Model performance and limitations

Sampling $x_S \sim p_\theta(x_S \mid x_M)$ **[SMCDiff]**

- Why conditional sampling vs. “naive” in-painting?
- Sequential Monte Carlo for exact sampling in the large-compute limit

Roadmap

Learning $p_\theta(x)$ **[ProtDiff]**
- Diffusion generative modeling background
- Adapting diffusion for protein backbones
- Model performance and limitations

Sampling $x_S \sim p_\theta(x_S \mid x_M)$ **[SMCDiff]**
- Why conditional sampling vs. “naive” in-painting?
- Sequential Monte Carlo for exact sampling in the large-compute limit

Limitations, Related Work, and Future directions

Diffusion models on protein backbones

Figures/slides borrowed from:

- CVPR 2022 Tutorial: Denoising Diffusion-based Generative Modeling: Foundations and Applications

State-of-the-art

DALL·E 2

"a propaganda poster depicting a cat dressed as french emperor napoleon holding a piece of cheese"

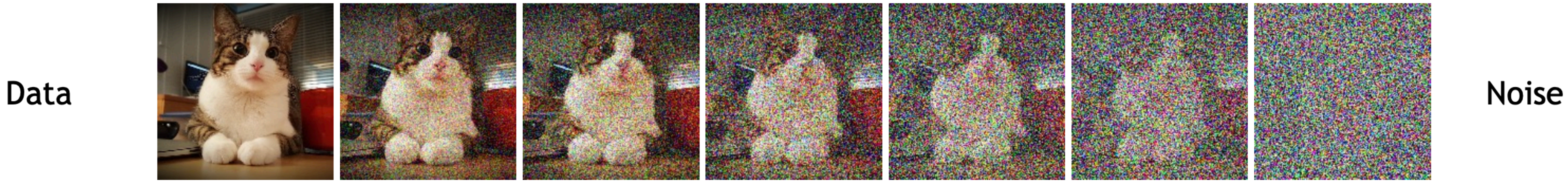
IMAGEN

"A photo of a raccoon wearing an astronaut helmet, looking out of the window at night."

[Figure from CVPR tutorial]

Denoising diffusion probabilistic models

General idea

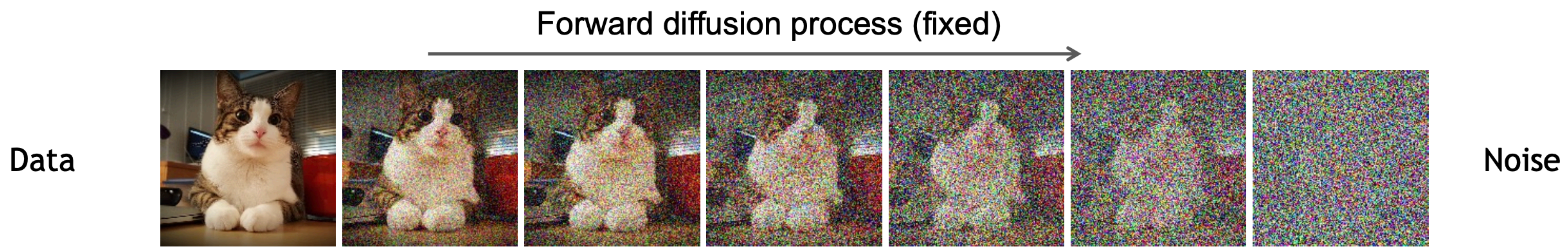


[Figure from CVPR tutorial]

Denoising diffusion probabilistic models

General idea

- Forward *diffusion* process gradually adds noise to input data.

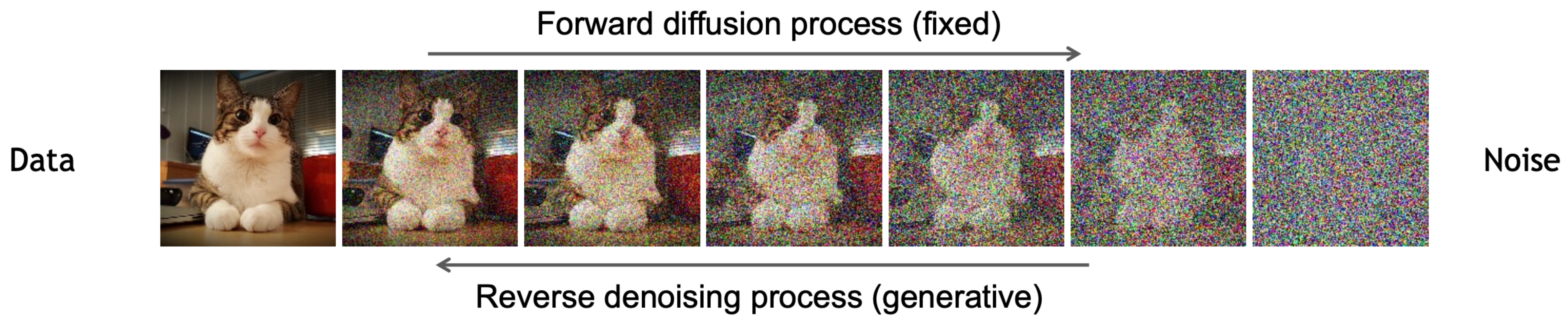


[Figure from CVPR tutorial]

Denoising diffusion probabilistic models

General idea

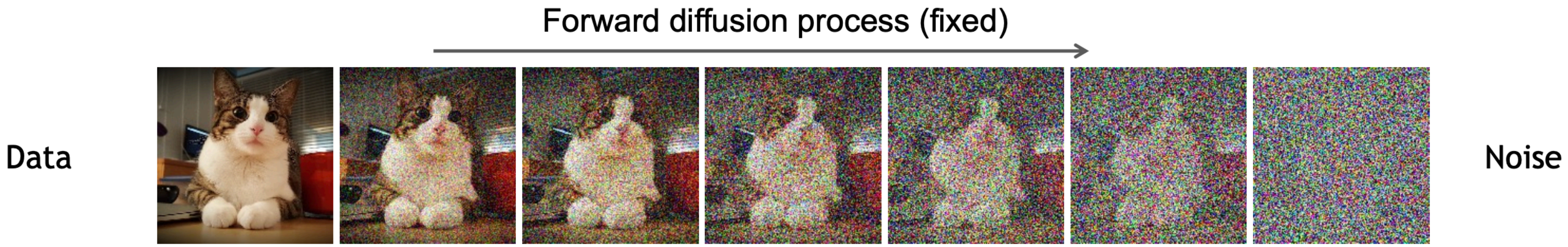
- Forward *diffusion* process gradually adds noise to input data.
- Reverse *denoising* process generates data by removing noise.


Forward diffusion process (fixed)
Data
Noise
Reverse denoising process (generative)

[Figure from CVPR tutorial]

Denoising diffusion probabilistic models

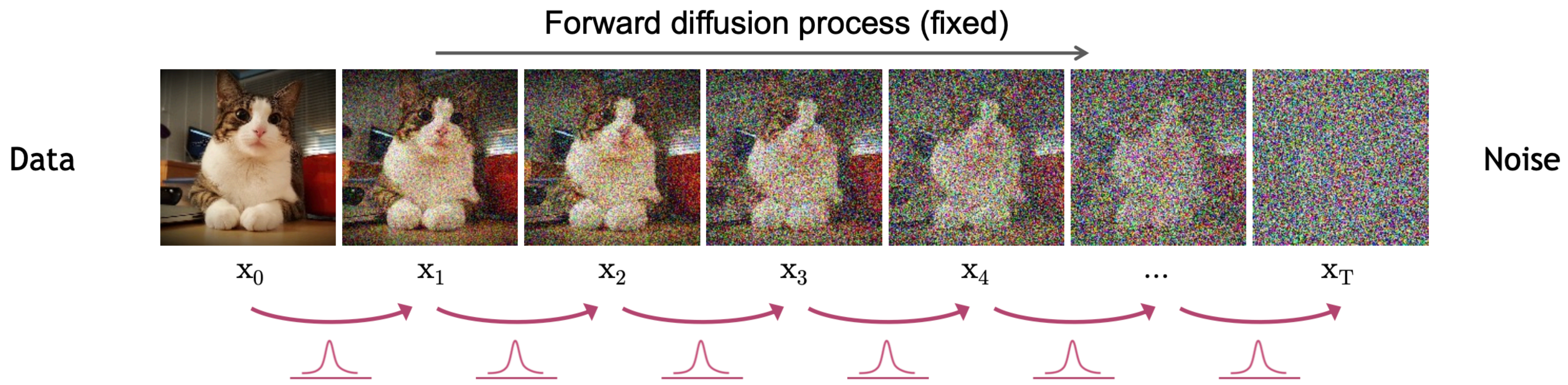
Forward diffusion



[Figure from CVPR tutorial]

Denoising diffusion probabilistic models

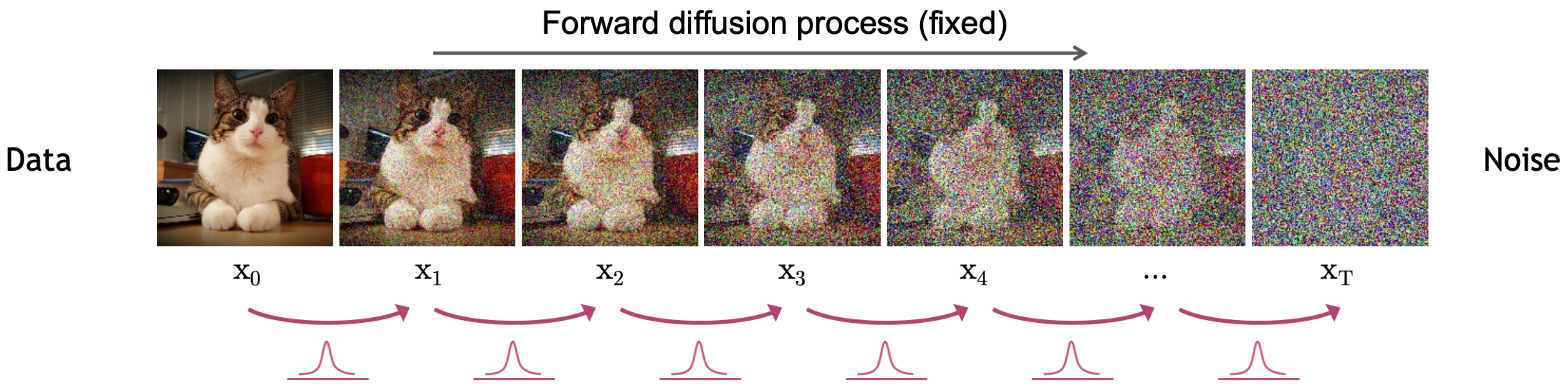
Forward diffusion



[Figure from CVPR tutorial]

Denoising diffusion probabilistic models

Forward diffusion



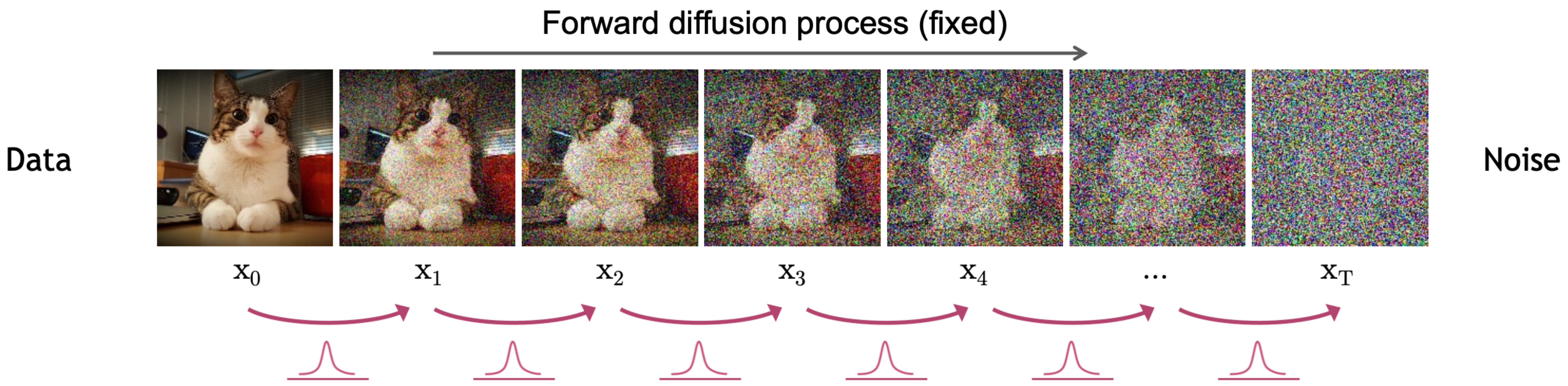
$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x_t}; \sqrt{1-\beta_t}\mathbf{x_{t-1}}, \beta_t\mathbf{I})$$

- β_t how much noise is added on each step
- [Ho et al]: $\beta_0 = 0.0001$, $\beta_T = 0.2$, $\beta_{t-1} < \beta_t$
- $(1-\beta_t)$ is how much signal is kept.

[Figure from CVPR tutorial]

Denoising diffusion probabilistic models

Forward diffusion



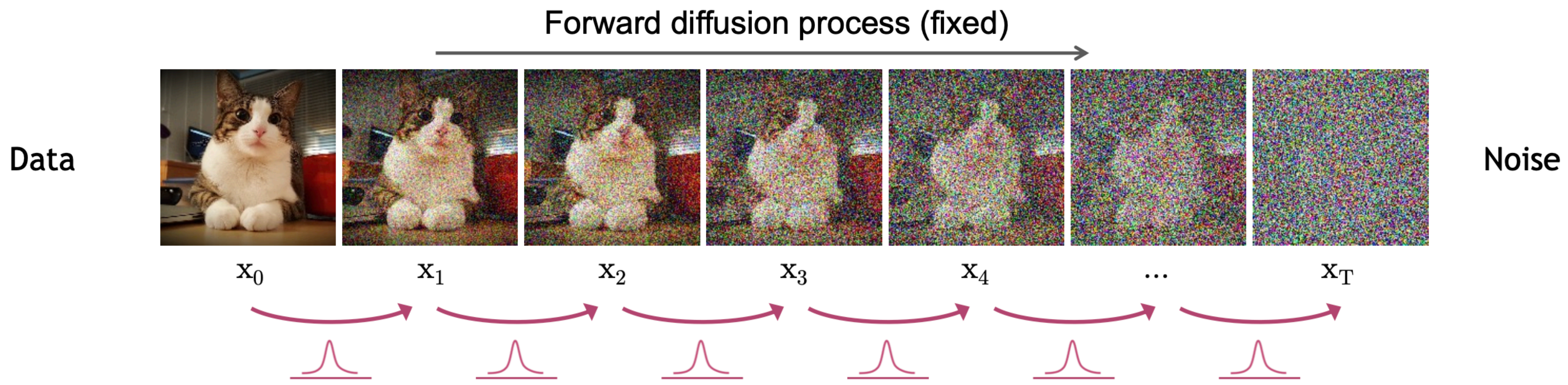
$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x_t}; \sqrt{1-\beta_t}\mathbf{x_{t-1}}, \beta_t\mathbf{I}) \quad \Rightarrow \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \qquad \text{(joint)}$$

- β_t how much noise is added on each step
- [Ho et al]: $\beta_0 = 0.0001$, $\beta_T = 0.2$, $\beta_{t-1} < \beta_t$
- $(1 - \beta_t)$ is how much signal is kept.

[Figure from CVPR tutorial]

Denoising diffusion probabilistic models

Forward diffusion



$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x_t}; \sqrt{1-\beta_t}\mathbf{x_{t-1}}, \beta_t\mathbf{I}) \Rightarrow q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$ (joint)

- β_t how much noise is added on each step
- [Ho et al]: $\beta_0 = 0.0001$, $\beta_T = 0.2$, $\beta_{t-1} < \beta_t$
- $(1-\beta_t)$ is how much signal is kept.

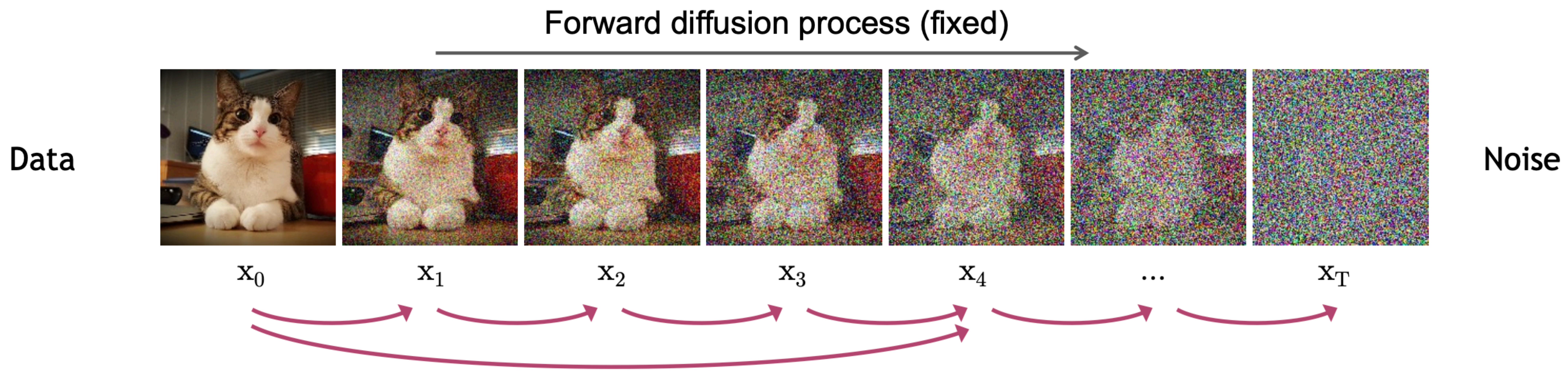
- Training requires sampling every x_t
- But this is expensive for large t : $q(x_t | x_{t-1}) \cdot q(x_{t-1} | x_{t-2}) \ldots q(x_1 | x_0)$

[Figure from CVPR tutorial]

Denoising diffusion probabilistic models

Marginal forward distribution

- Desirable during training to sample $q(x_t)$ for any t in $O(1)$ instead of $O(T)$

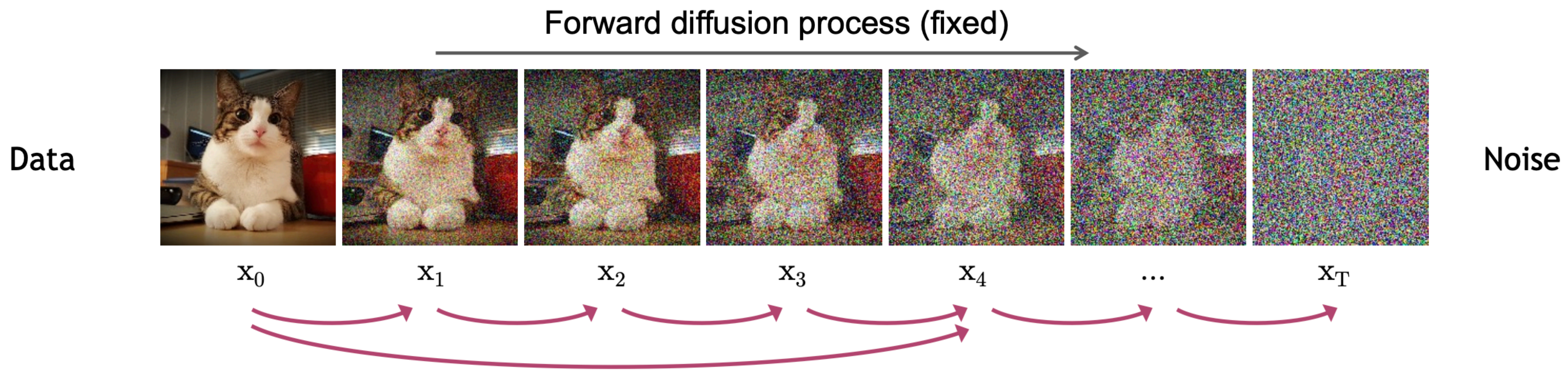


[Figure from CVPR tutorial]

Denoising diffusion probabilistic models

Marginal forward distribution

- Desirable during training to sample $q(x_t)$ for any t in $O(1)$ instead of $O(T)$



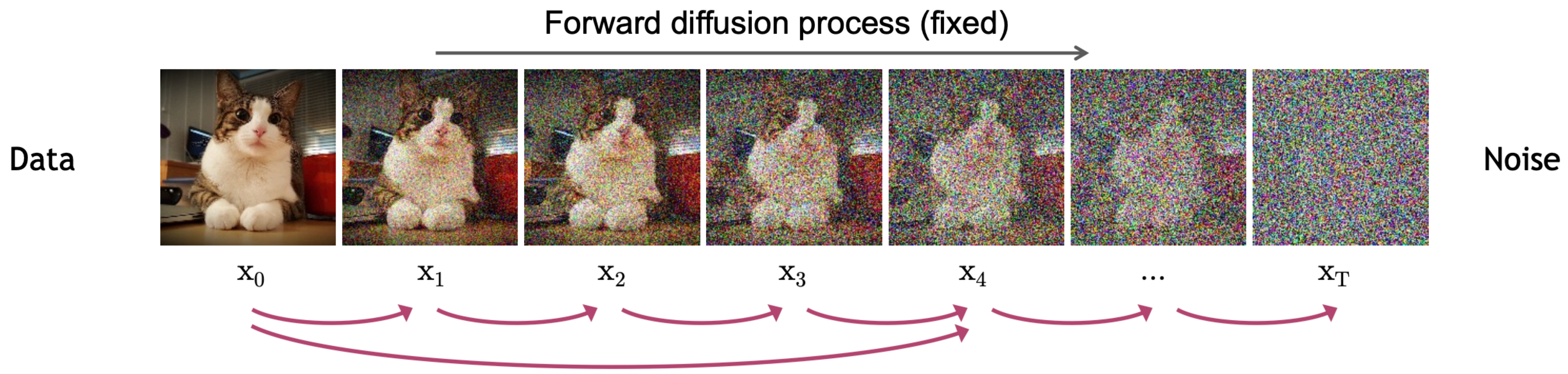
Define $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$

[Figure from CVPR tutorial]

Denoising diffusion probabilistic models

Marginal forward distribution

- Desirable during training to sample $q(x_t)$ for any t in $O(1)$ instead of $O(T)$



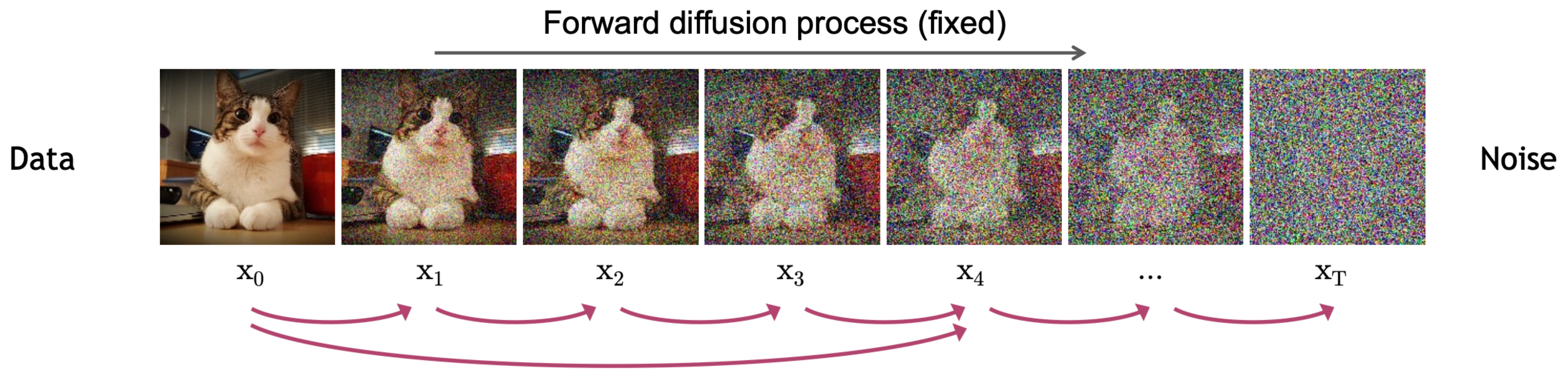
Define $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$ ➡ $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \underbrace{\sqrt{\bar{\alpha}_t}\mathbf{x}_0}_{\text{Remaining signal}}, \underbrace{(1-\bar{\alpha}_t)\mathbf{I}}_{\text{Injected noise}}),$ (Diffusion Kernel)

[Figure from CVPR tutorial]

Denoising diffusion probabilistic models

Marginal forward distribution

- Desirable during training to sample $q(x_t)$ for any t in $O(1)$ instead of $O(T)$



Define $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$ ➡ $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \underbrace{\sqrt{\bar{\alpha}_t}\mathbf{x}_0}_{\text{Remaining signal}}, \underbrace{(1-\bar{\alpha}_t)\mathbf{I}}_{\text{Injected noise}}),$ (Diffusion Kernel)

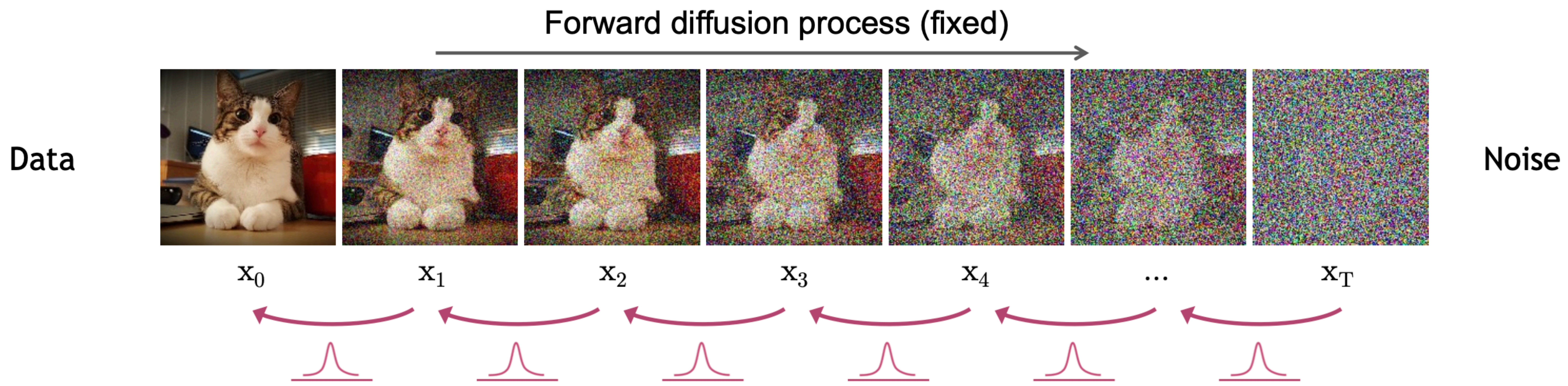
$$\underbrace{q(\mathbf{x}_t)}_{\text{Diffused data dist.}} = \int \underbrace{q(\mathbf{x}_0, \mathbf{x}_t)}_{\text{Joint dist.}}\, d\mathbf{x}_0 = \int \underbrace{q(\mathbf{x}_0)}_{\text{Input data dist.}}\, \underbrace{q(\mathbf{x}_t|\mathbf{x}_0)}_{\text{Diffusion kernel}}\, d\mathbf{x}_0$$

For sampling: $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\,\epsilon$ where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

[Figure from CVPR tutorial]

Denoising diffusion probabilistic models

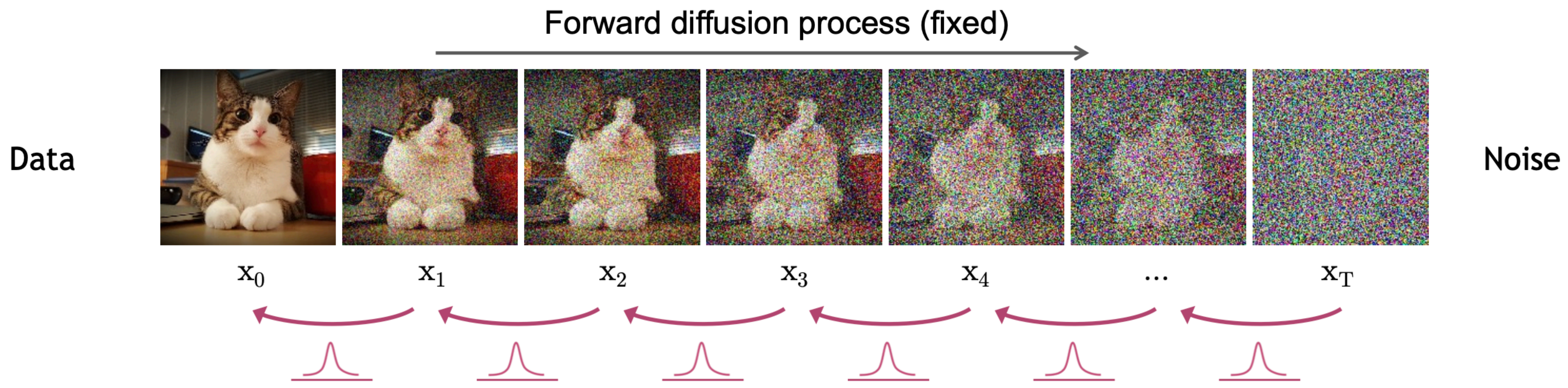
Reverse denoising process



[Figure from CVPR tutorial]

Denoising diffusion probabilistic models

Reverse denoising process

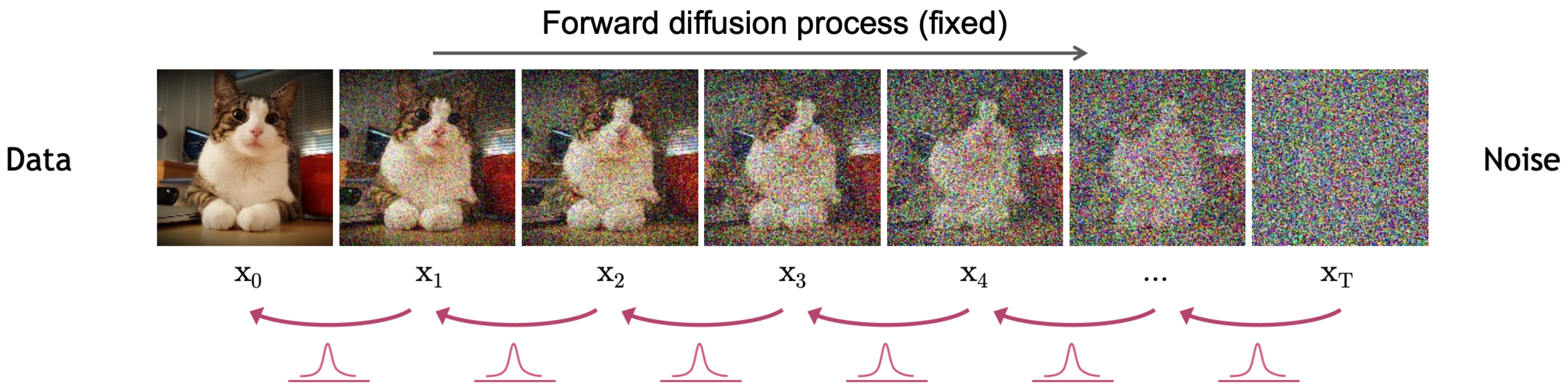


$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$

[Figure from CVPR tutorial]

Denoising diffusion probabilistic models

Reverse denoising process



$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$

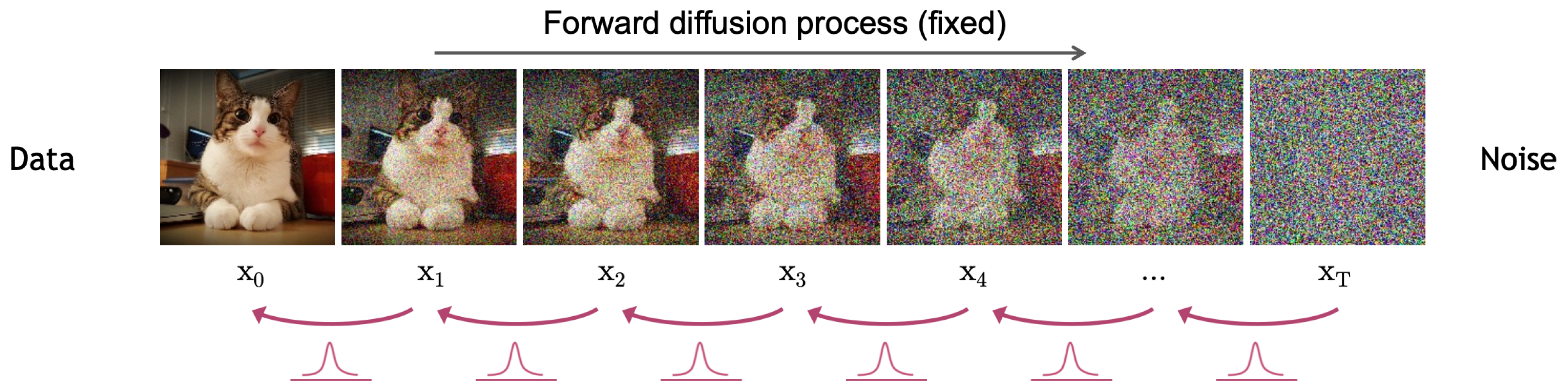
Goal: $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \approx q(\mathbf{x}_t|\mathbf{x}_{t-1})$

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \underbrace{\mu_\theta(\mathbf{x}_t, t)}_{\text{Trainable network}}, \sigma_t^2\mathbf{I})$$

[Figure from CVPR tutorial]

Denoising diffusion probabilistic models

Reverse denoising process



$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$

Goal: $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \approx q(\mathbf{x}_t|\mathbf{x}_{t-1})$

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \underbrace{\mu_\theta(\mathbf{x}_t, t)}_{\text{Trainable network}}, \sigma_t^2\mathbf{I})$$

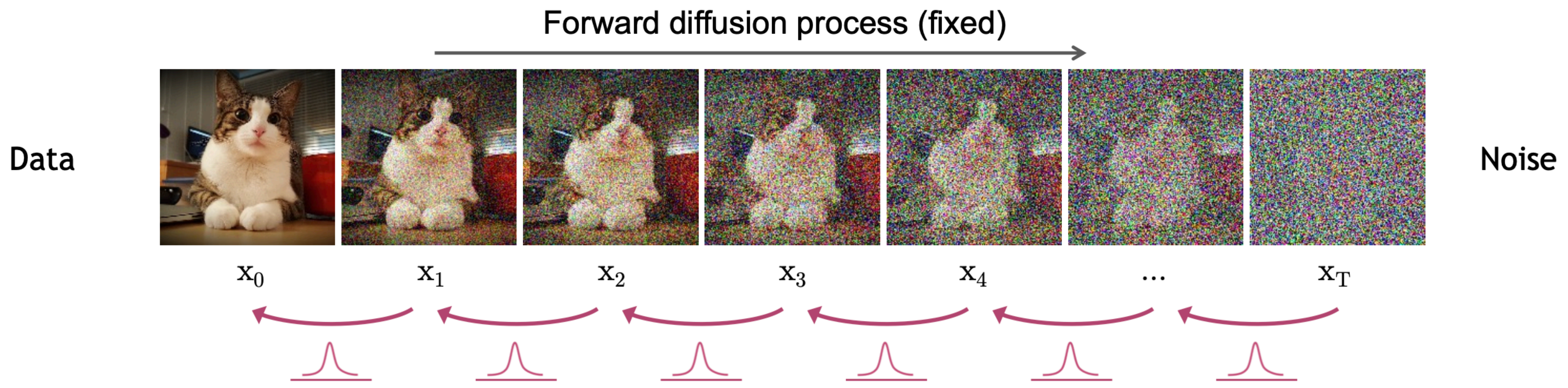
$\Rightarrow$ $$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

First sample from x_T and go backwards to x_0

[Figure from CVPR tutorial]

Denoising diffusion probabilistic models

Reverse denoising process



$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$

Goal: $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \approx q(\mathbf{x}_t|\mathbf{x}_{t-1})$

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \underbrace{\mu_\theta(\mathbf{x}_t, t)}_{\text{Trainable network}}, \sigma_t^2 \mathbf{I})$$

➡ $$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

First sample from x_T and go backwards to x_0

Sampling is $O(T)$

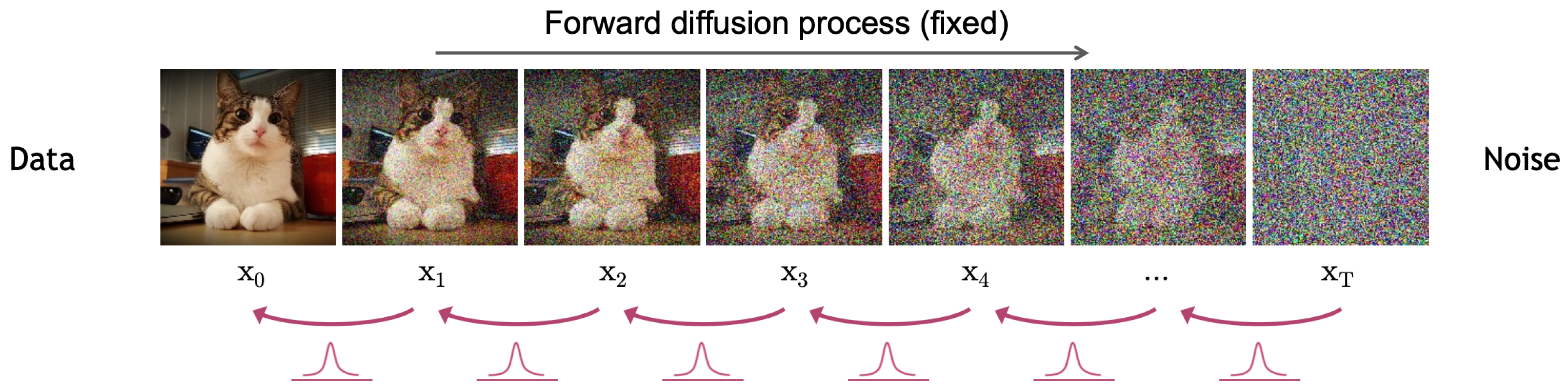
$T \in [100, 1000]$ depending on problem.

Active area of research to speed up sampling.

[Figure from CVPR tutorial]

Denoising diffusion probabilistic models

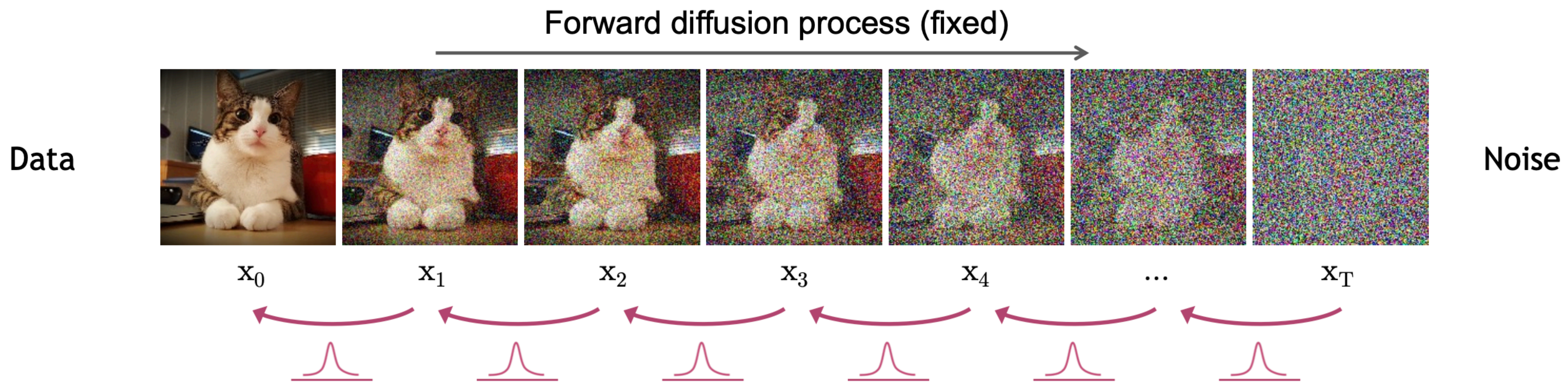
Optimization



[Figure from CVPR tutorial]

Denoising diffusion probabilistic models

Optimization

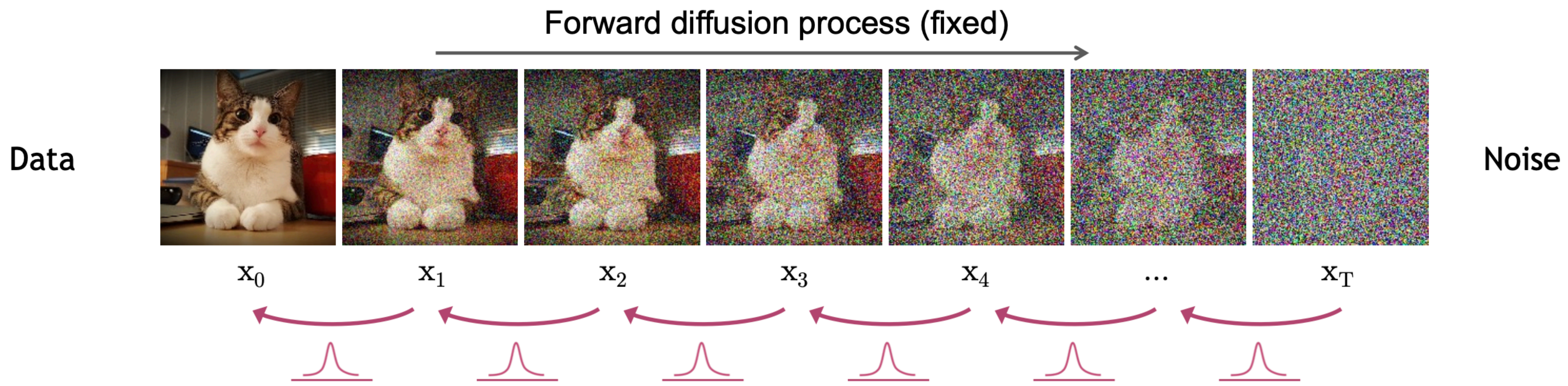


Learning reverse transition: $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I})$

[Figure from CVPR tutorial]

Denoising diffusion probabilistic models

Optimization



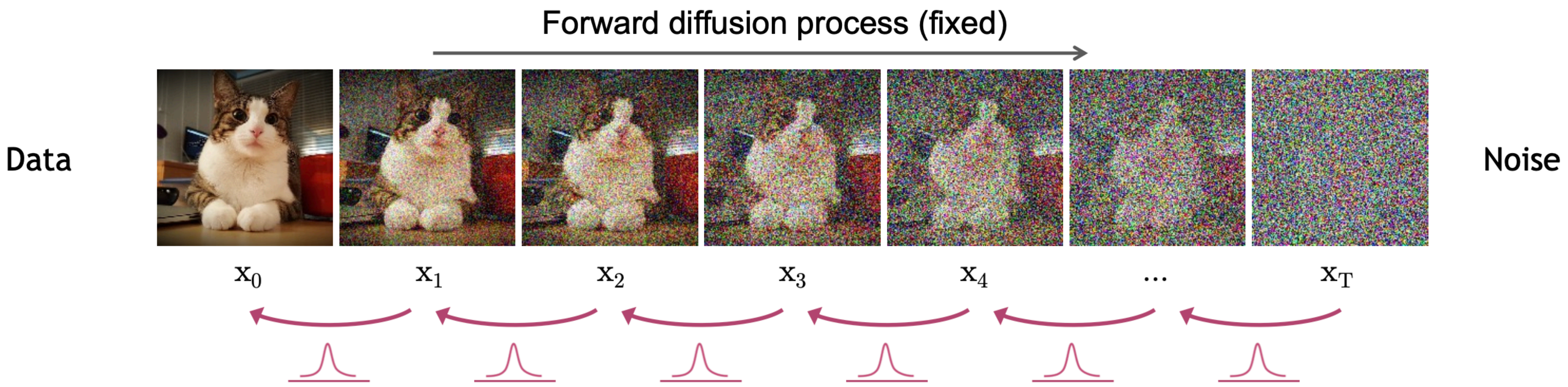
Learning reverse transition: $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I})$

Recall $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\,\epsilon$

[Figure from CVPR tutorial]

Denoising diffusion probabilistic models

Optimization



Learning reverse transition: $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I})$

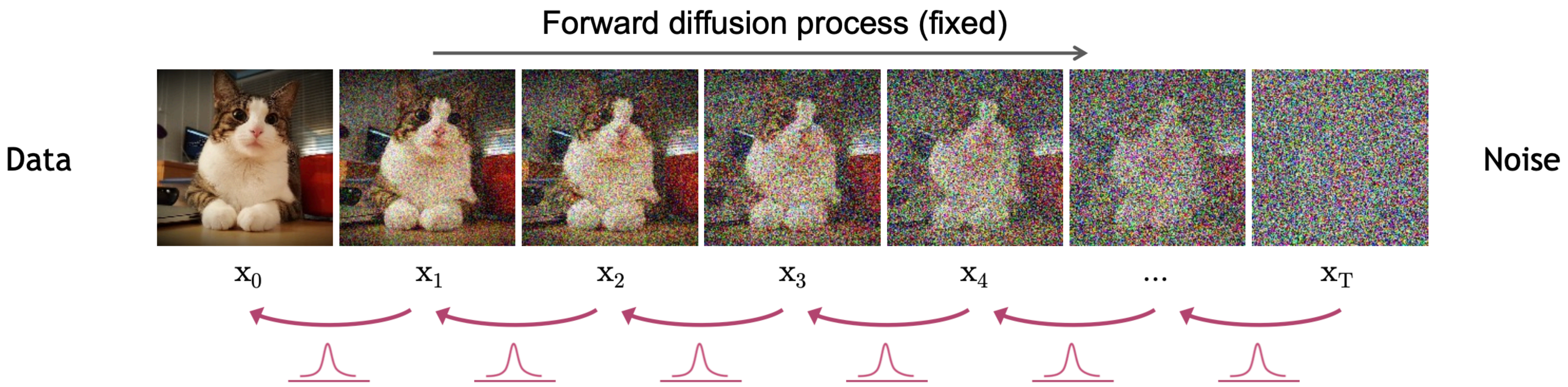
Recall $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\,\epsilon$

Related to x_0: $\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon_\theta(\mathbf{x}_t, t)\right)$

[Figure from CVPR tutorial]

Denoising diffusion probabilistic models

Optimization



Learning reverse transition: $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I})$

Recall $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\,\epsilon$

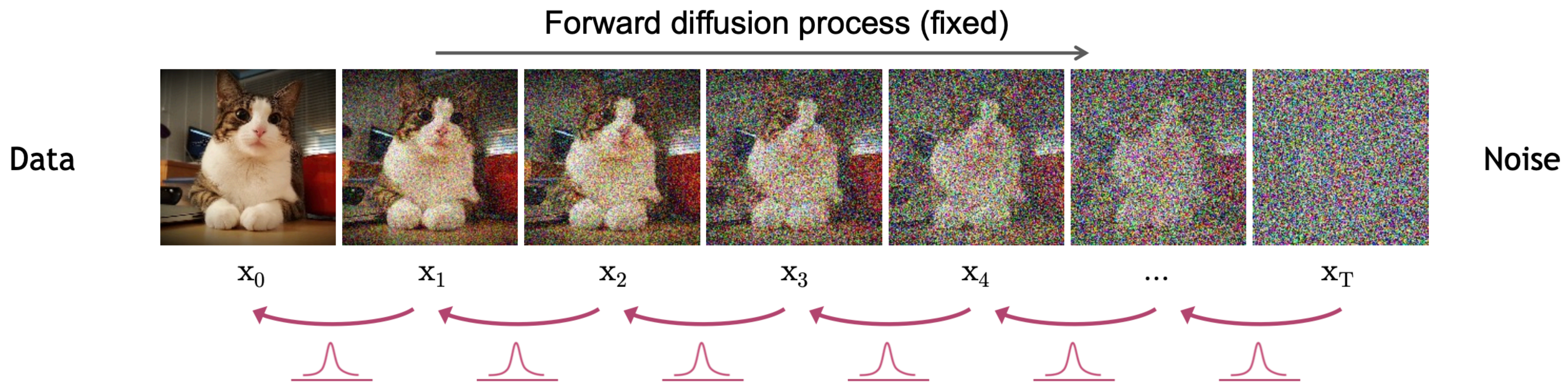
Related to x_0: $\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right)$

Reparameterization from [Ho et al]:
$L = \mathbb{E}[||\epsilon - \epsilon_\theta||^2]\,, \quad \epsilon \sim \mathcal{N}(0,1)$

[Figure from CVPR tutorial]

Denoising diffusion probabilistic models

Optimization



Learning reverse transition: $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I})$

Recall $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\,\epsilon$

Related to x_0: $\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right)$

Reparameterization from [Ho et al]:
$L = \mathbb{E}[||\epsilon - \epsilon_\theta||^2]\,, \quad \epsilon \sim \mathcal{N}(0,1)$

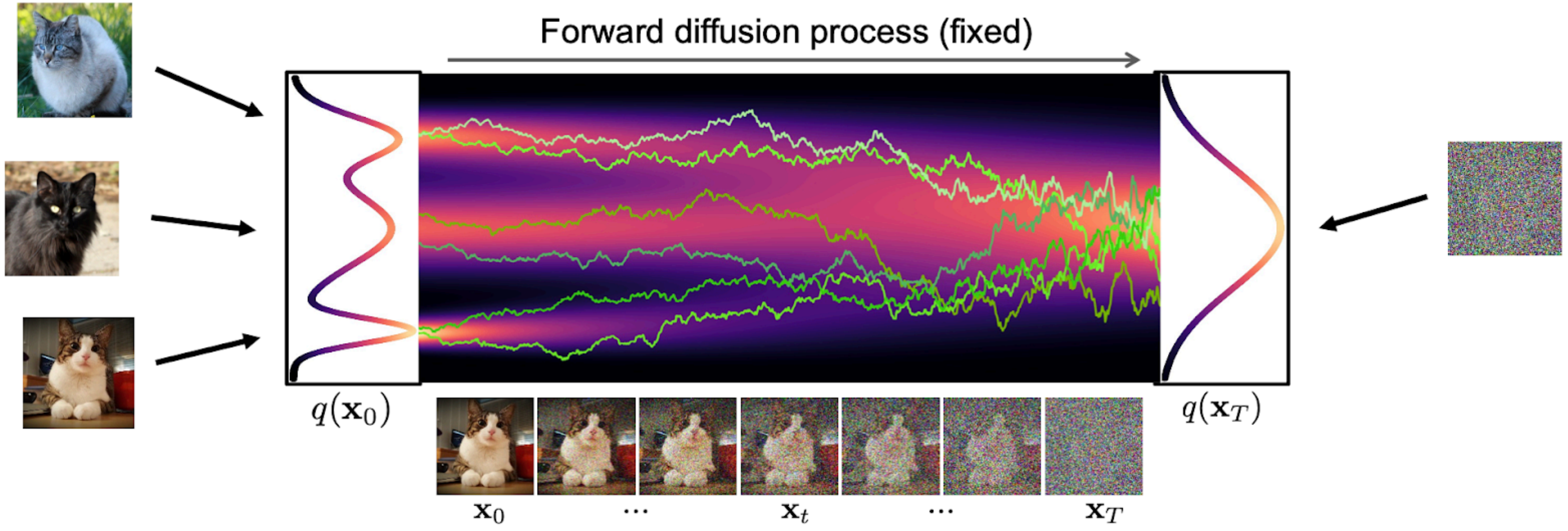
Noise prediction model: $\epsilon_\theta(x_t, t)$

[Figure from CVPR tutorial]

Denoising diffusion probabilistic models

Intuition

- No encoder (replaced with forward diffusion) only need to train decoder (compared to VAEs).
- Decoder is easy to train (compared to GANs) with simple loss that are easy to learn/predict.

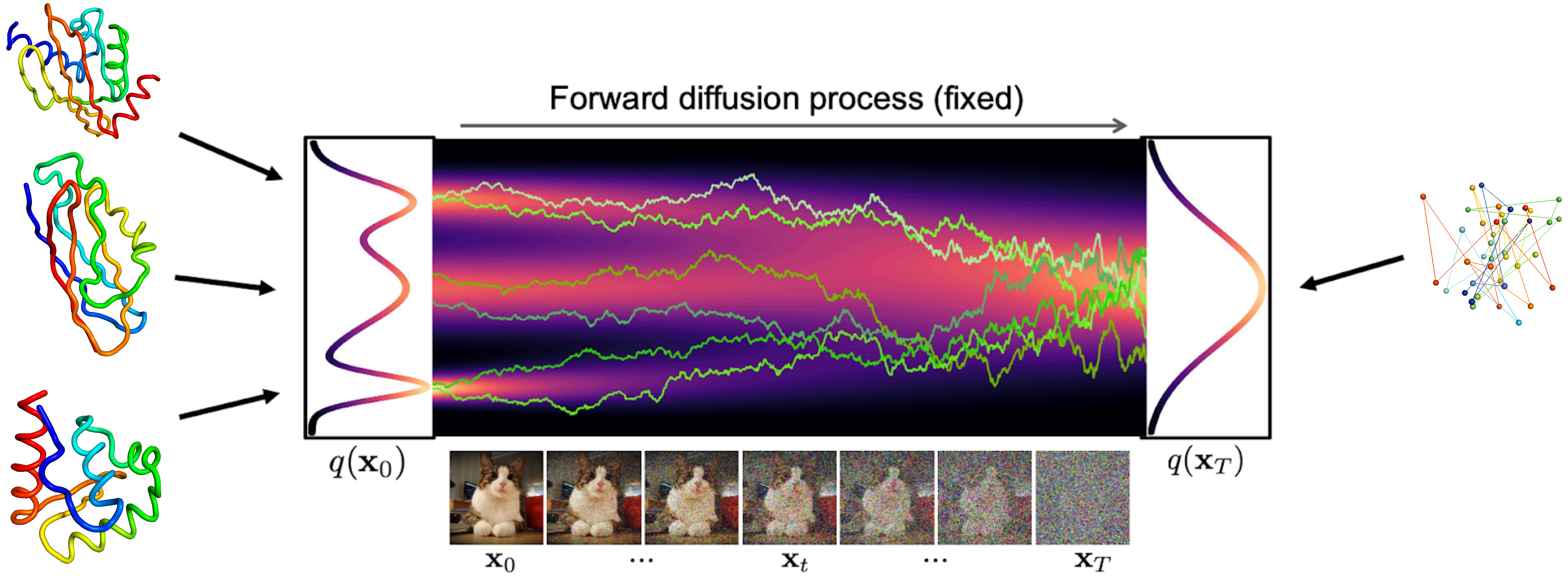


[Figure from CVPR tutorial]

ProtDiff: diffusion on protein backbones

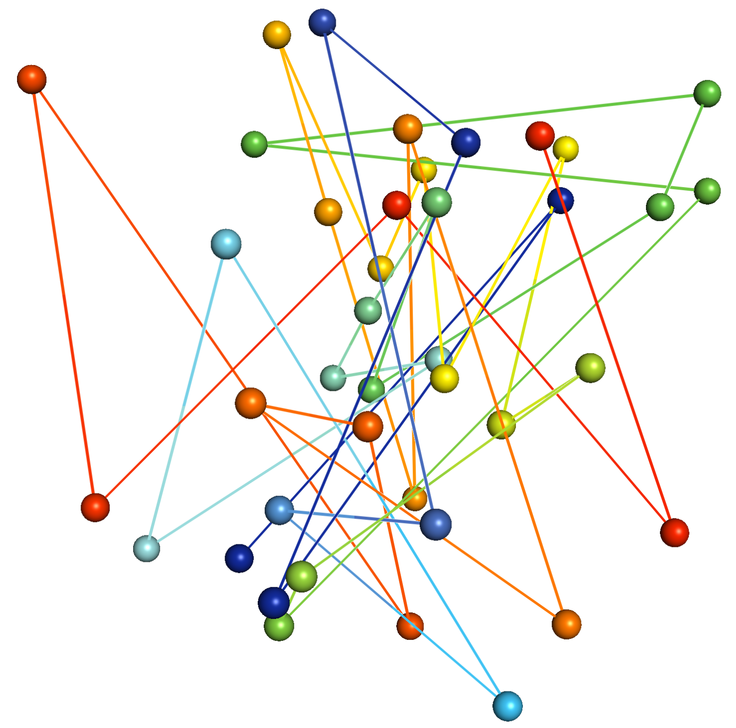
Benefits of diffusion on proteins

- Model directly in 3D space instead of distograms.
- Shown state-of-the-art success in generating small molecules [Hoogeboom et al.] and point clouds [Luo et al.]



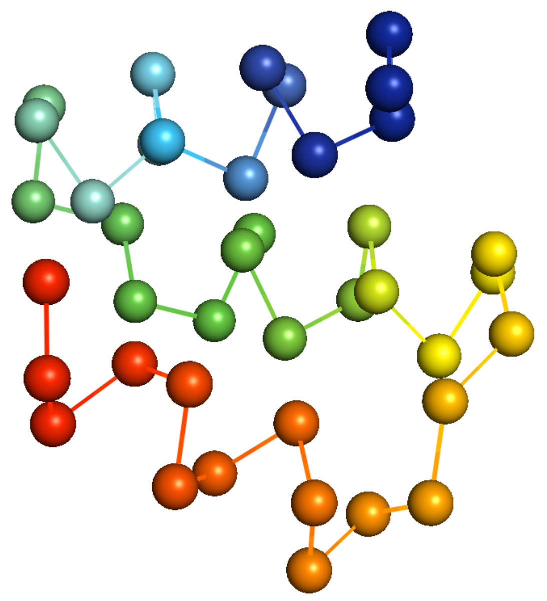
[Figure from CVPR tutorial]

ProtDiff: diffusion on protein backbones

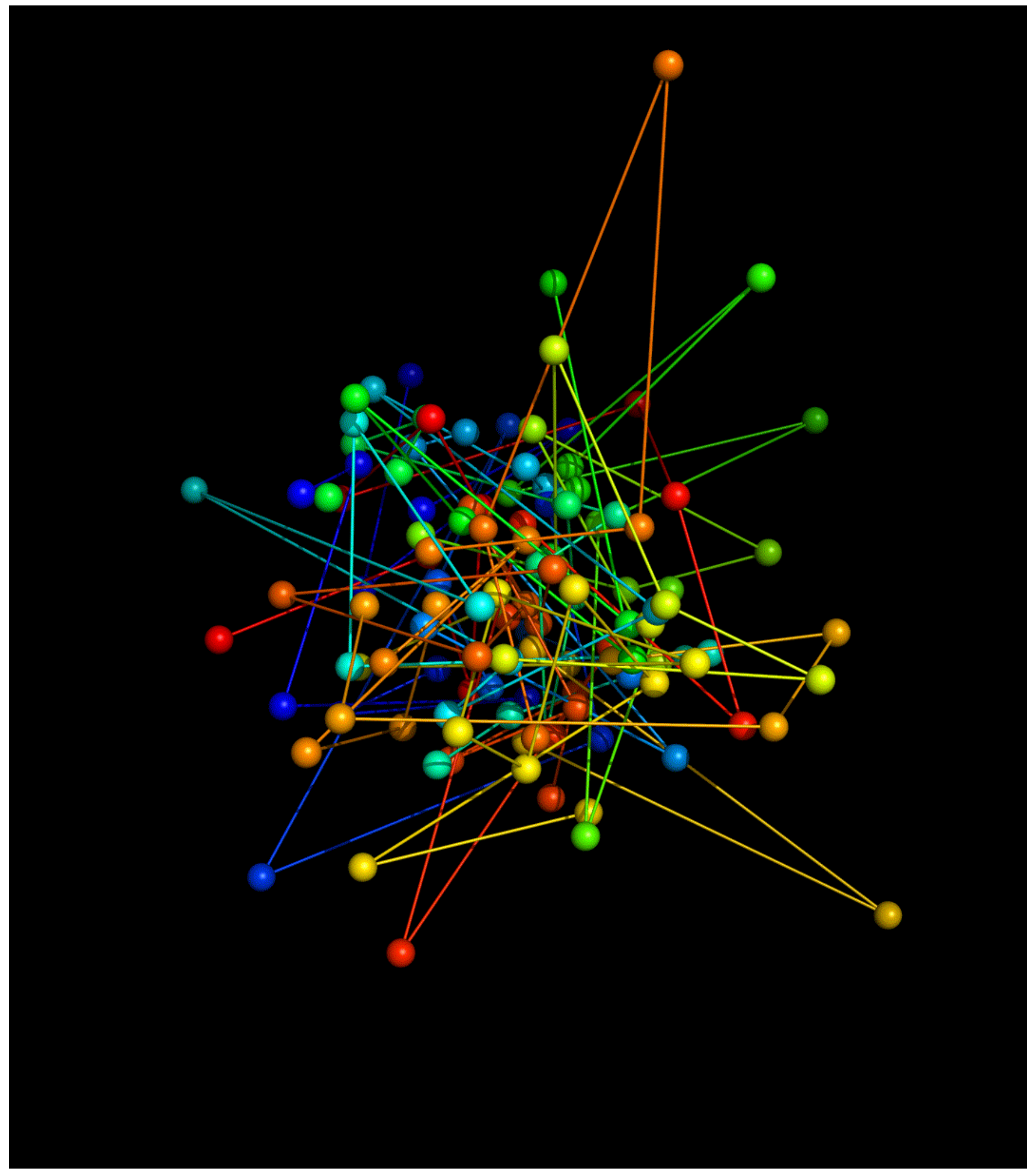


random Gaussian point cloud

$$= \quad x_T \quad \cdots \quad x_t \quad \xrightarrow[\longrightarrow]{\longleftarrow} \quad x_{t-1} \quad \cdots \quad x_0 \quad =$$



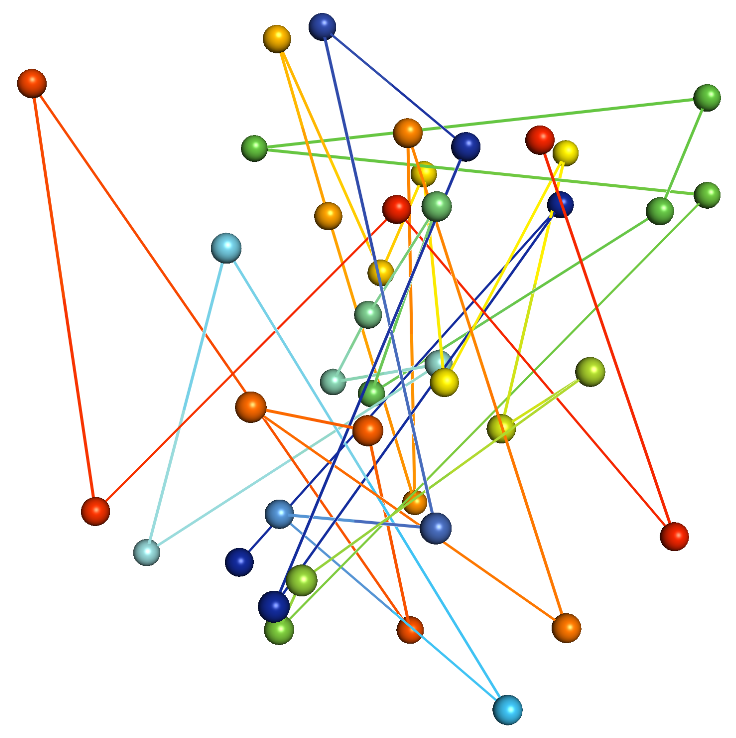
3D Cα coordinates of a protein



Future works:

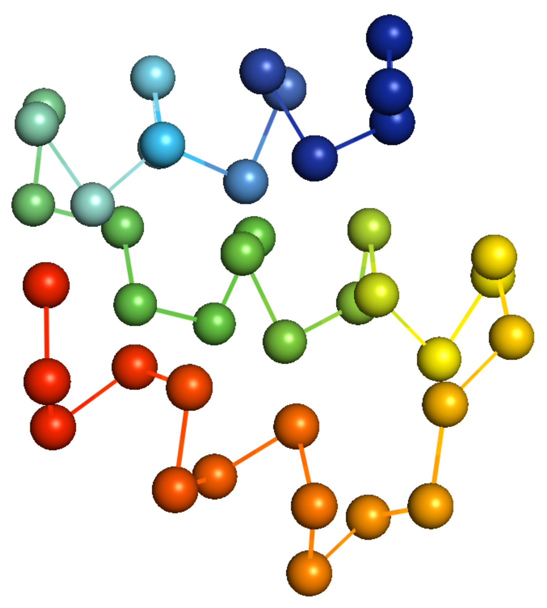
- Incorporate SE(3) rigid bodies.
- Incorporate sequence.

ProtDiff: diffusion on protein backbones

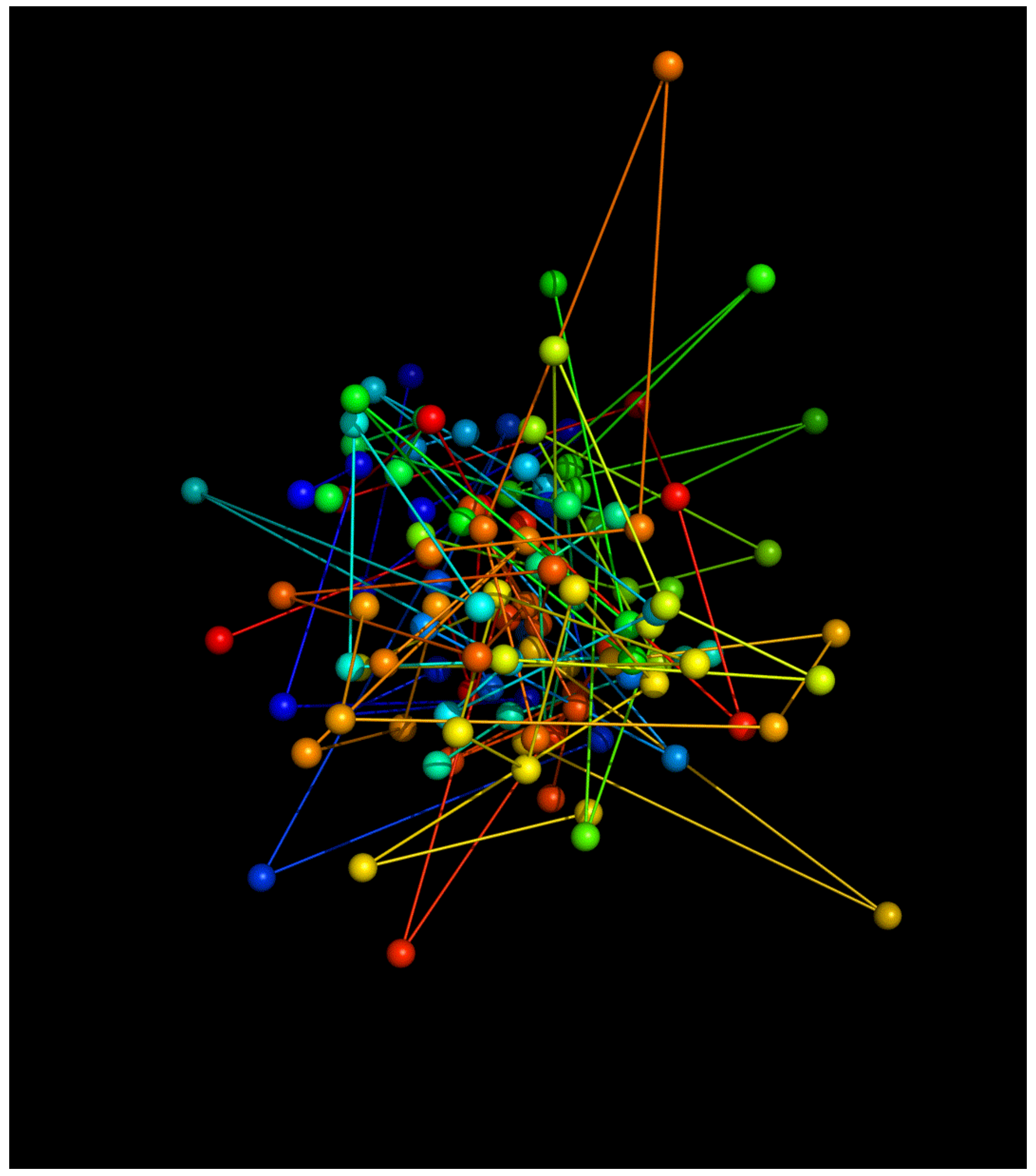


random Gaussian point cloud

$$= \quad x_T \quad \cdots \quad x_t \;\xleftarrow{\qquad}\; \xrightarrow{\qquad}\; x_{t-1} \quad \cdots \quad x_0 \quad =$$



3D Cα coordinates of a protein

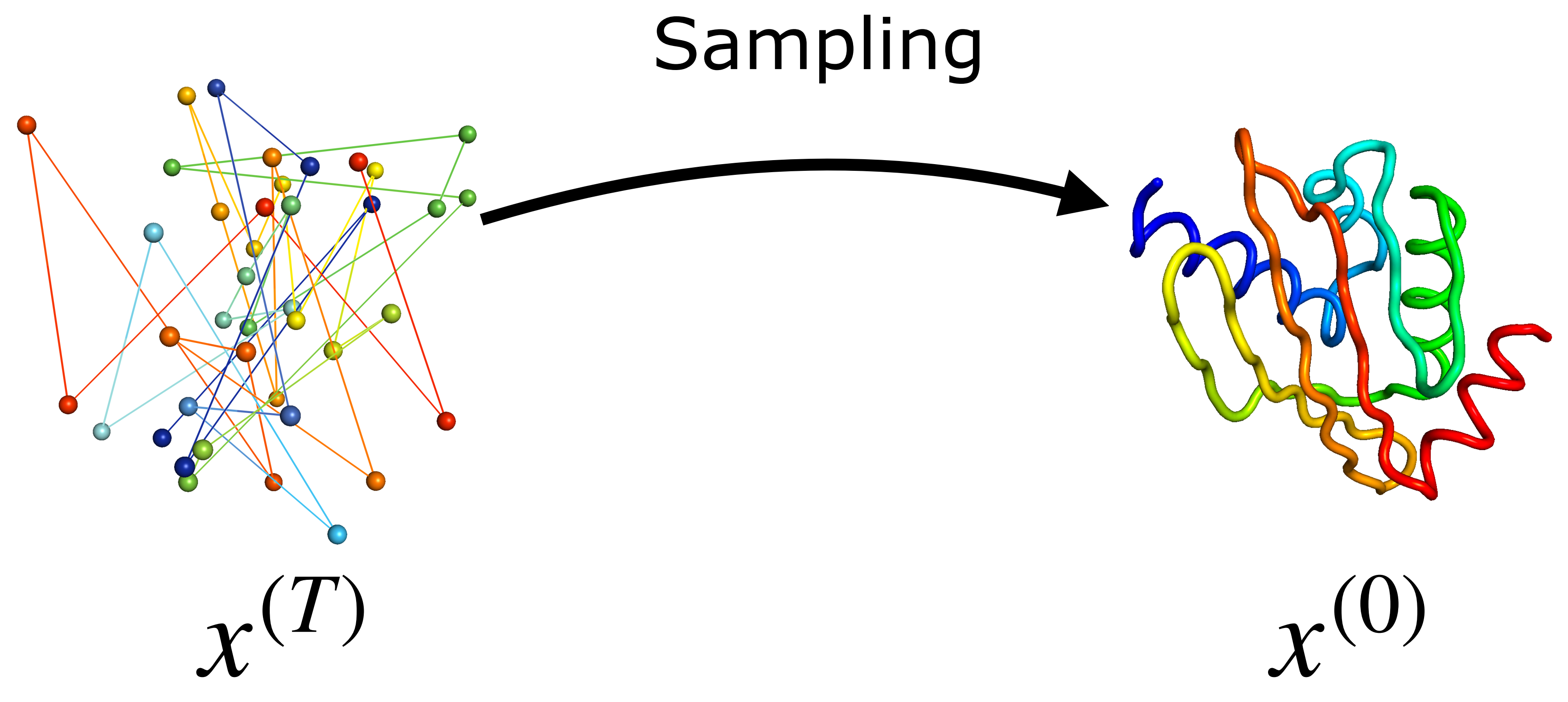


Future works:

- Incorporate SE(3) rigid bodies.
- Incorporate sequence.

Model details

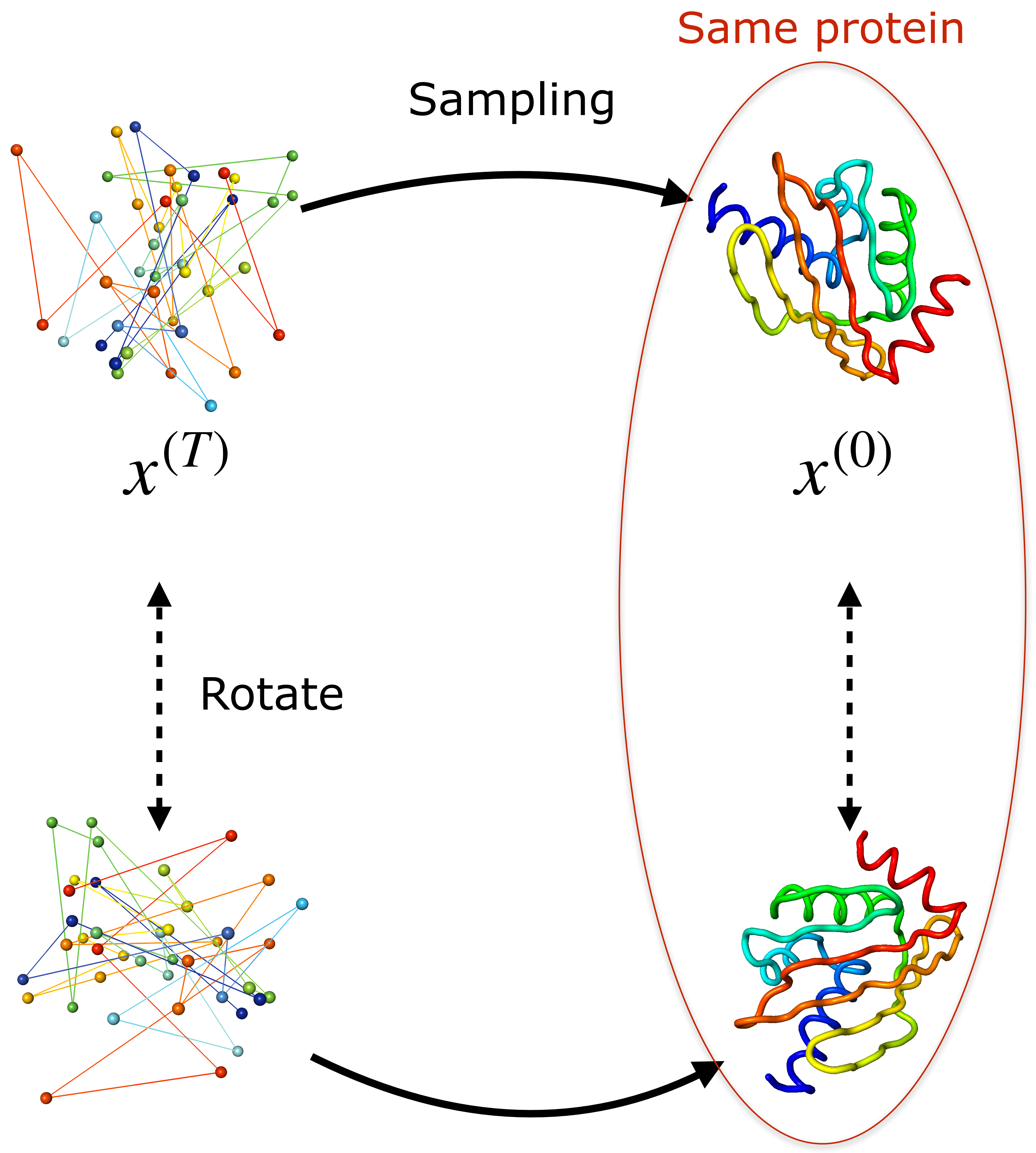
- **Input**:
  - $x_t \in \mathbb{R}^{N \times 3}$: zero-centered 3D backbone point clouds.
    - Treats protein as a fully connected graph.
  - $[s_1, \ldots, s_N]$ where s_i is sequence position index of residue i
    - Breaks permutation invariance.
  - $t \in [1, \ldots, T]$



Model details

- **Input**:
  - $x_t \in \mathbb{R}^{N \times 3}$: zero-centered 3D backbone point clouds.
    - Treats protein as a fully connected graph.
  - $[s_1, \ldots, s_N]$ where s_i is sequence position index of residue i
    - Breaks permutation invariance.
  - $t \in [1, \ldots, T]$

- **E(3) Equivariant diffusion model**:
  - Goal: $p(x_0) = p(Rx_0),\ R \in SO(3)$
  - Starting from ***invariant*** distribution with a ***equivariant*** noise prediction model leads to ***equivariant*** samples. [GeoDiff Xu et al.]
  - Distribution is invariant to E(3) but samples are equivariant.

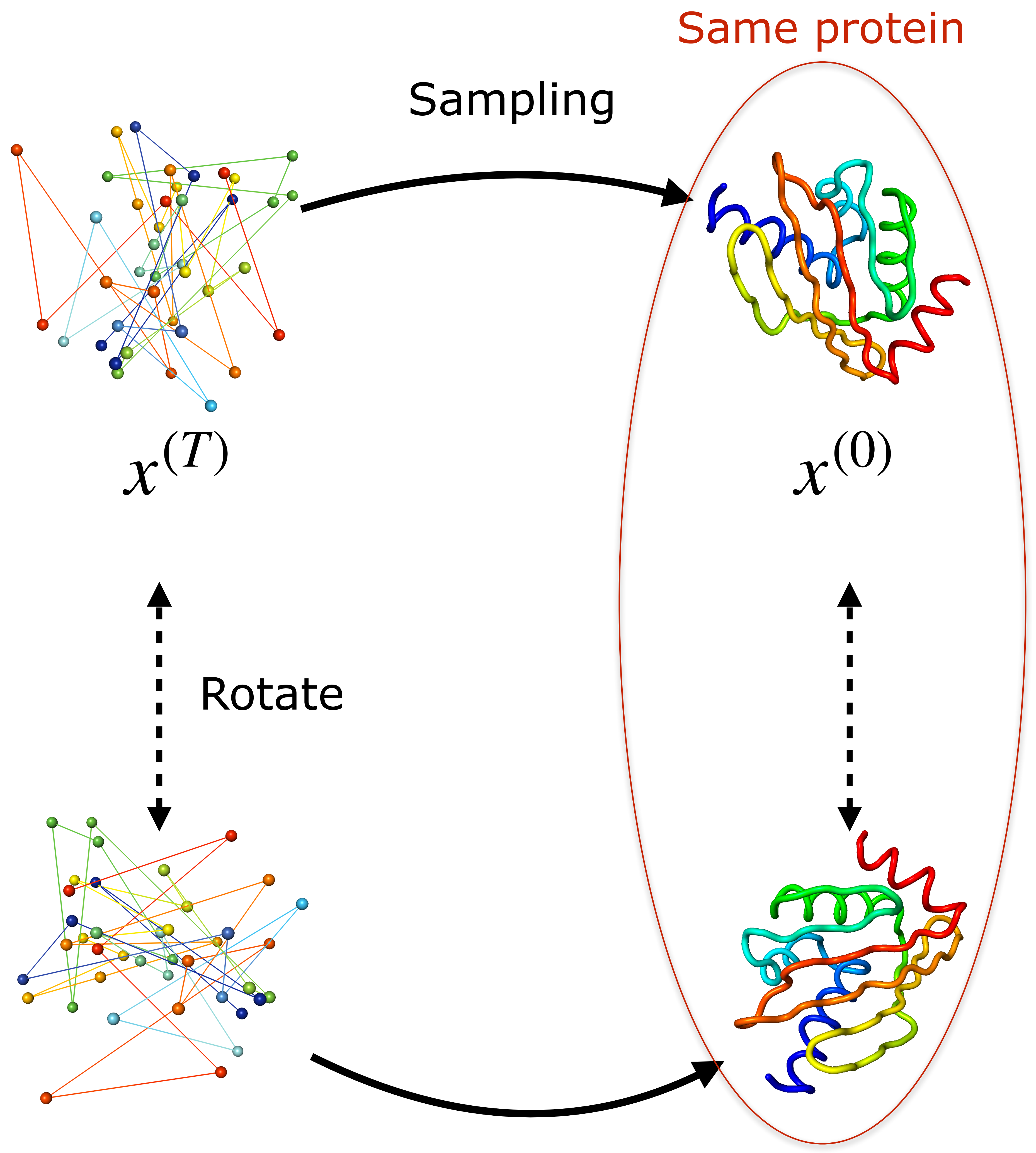


Model details

- **Input**:
  - $x_t \in \mathbb{R}^{N \times 3}$: zero-centered 3D backbone point clouds.
    - Treats protein as a fully connected graph.
  - $[s_1, \ldots, s_N]$ where s_i is sequence position index of residue i
    - Breaks permutation invariance.
  - $t \in [1, \ldots, T]$

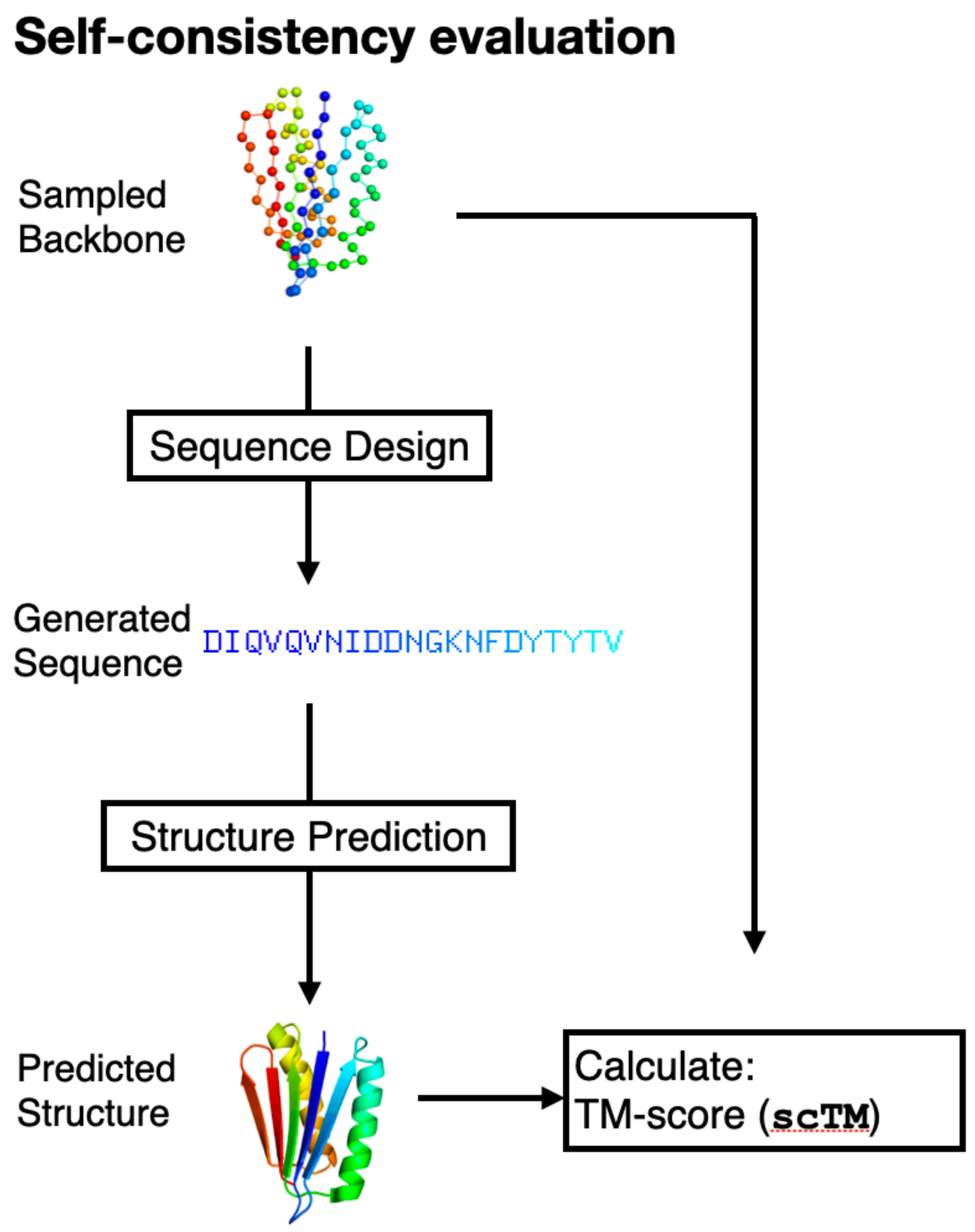
- **E(3) Equivariant diffusion model**:
  - Goal: $p(x_0) = p(Rx_0),\ R \in SO(3)$
  - Starting from ***invariant*** distribution with a ***equivariant*** noise prediction model leads to ***equivariant*** samples. [GeoDiff Xu et al.]
  - Distribution is invariant to E(3) but samples are equivariant.

- **Neural network**:
  - E(3) equivariant graph neural network (EGNN) [Satorras et al.]
  - Property that Euclidean transforms equally affect output.
    $R\ \epsilon_\theta(x) = \epsilon_\theta(Rx) \rightarrow$ equivariant noise prediction



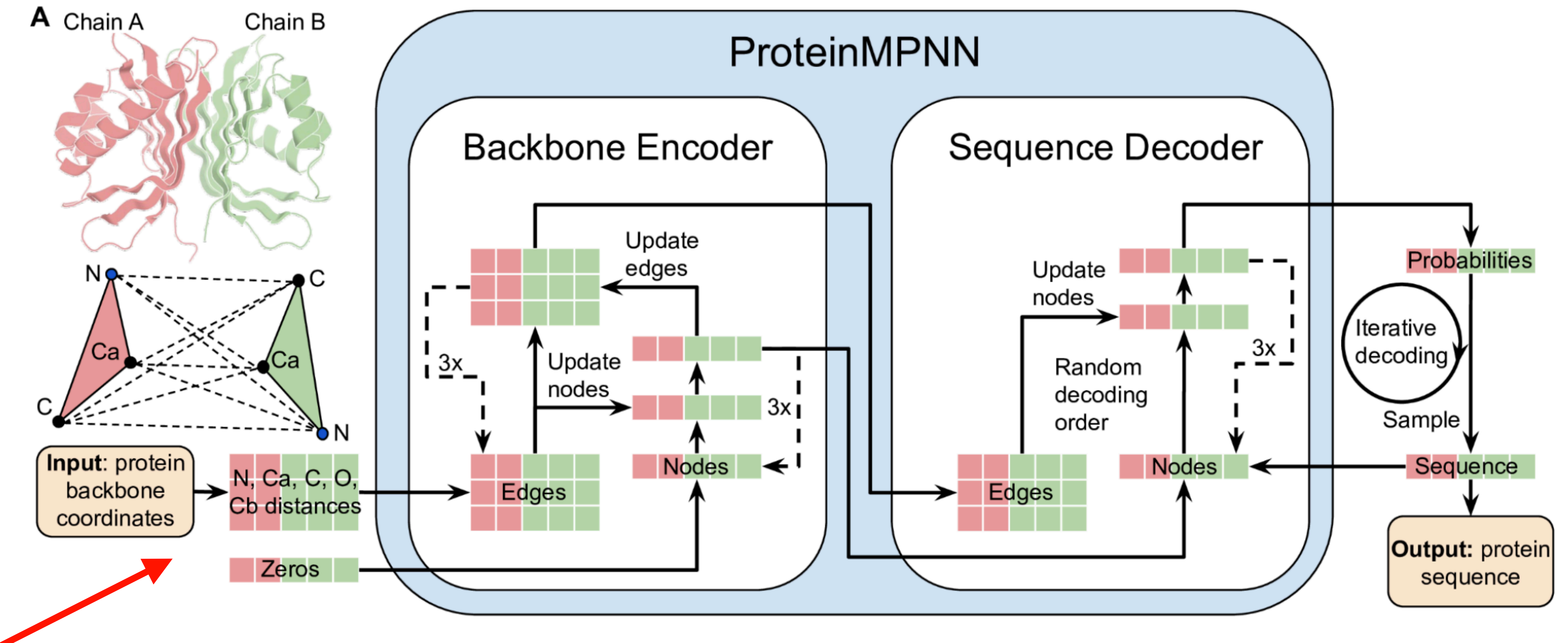
Designability criterion

- A 3D structure is **designable** if a protein sequence can be found that folds into the same 3D structure.
- Use **sequence design** to search for likely sequences for a backbone.
- Use **structure prediction** (folding) to verify backbone samples can be designed from a protein sequence.
  - >0.5 TM-score indicates roughly same structure.



Self-consistency components

- Sequence design: **ProteinMPNN**
  - State-of-the-art fixed backbone sequence design method.

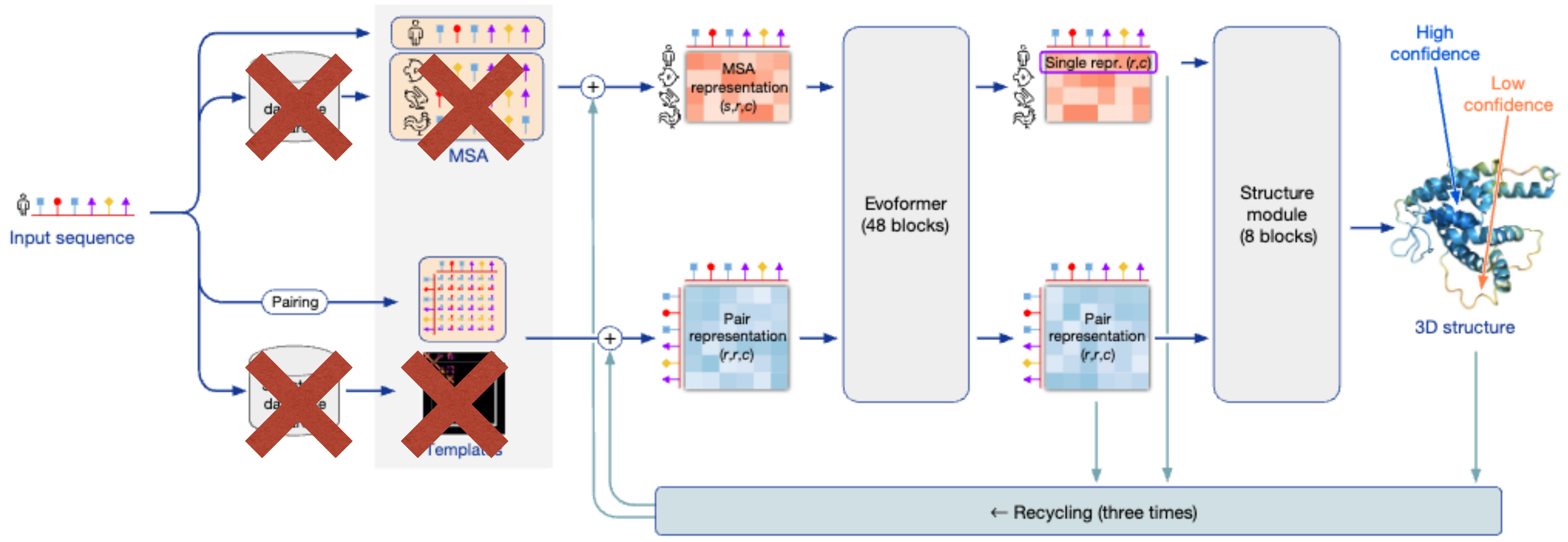


Only Ca distances

[Dauparas et al]

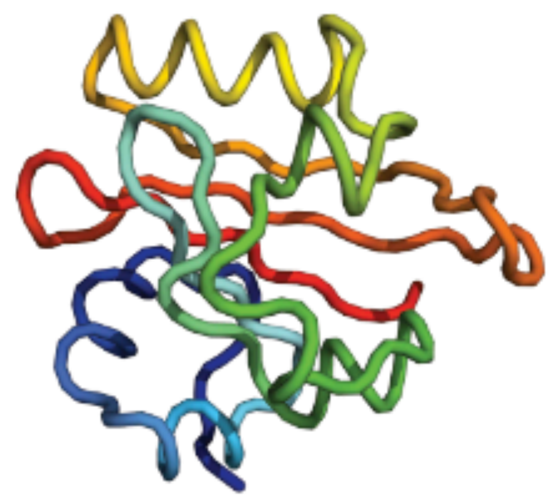
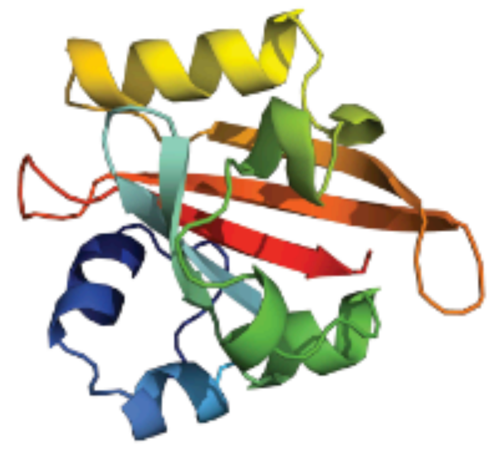
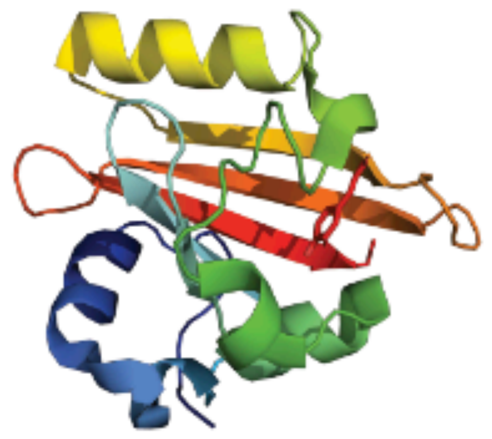
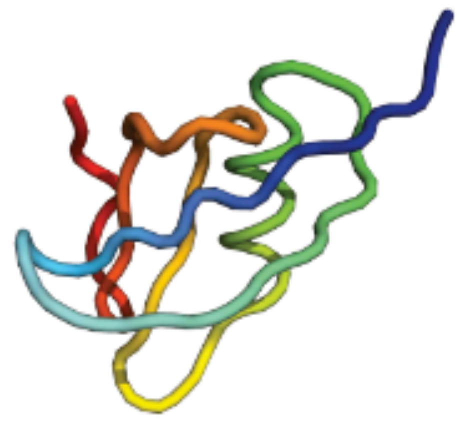
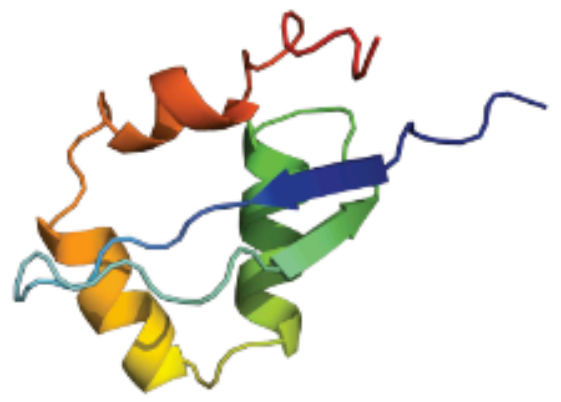
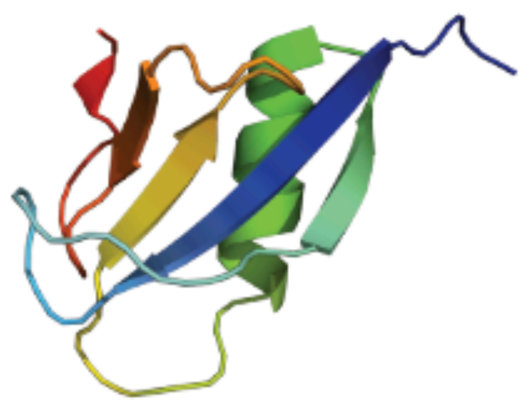
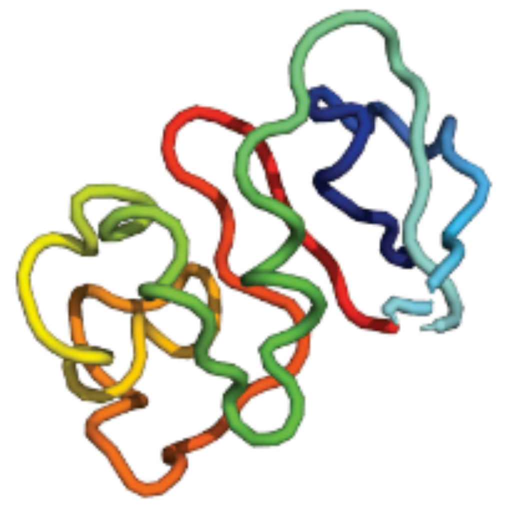
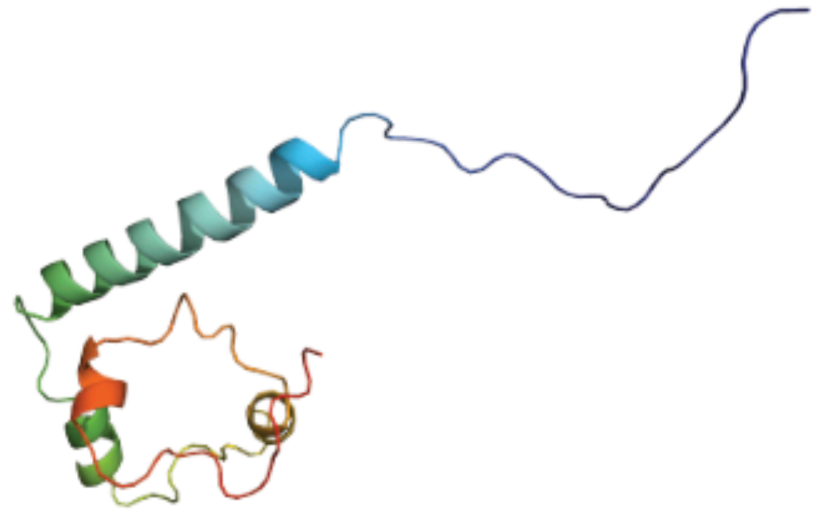
- Structure prediction: **AlphaFold2**
  - No MSA. No template. Allows for fast inference. Only include query sequence.

  - Use released CASP14 weights.



[Jumper et al]

Characterization of self-consistency

	Backbone design	AF2 prediction	PDB parent
scTM > 0.9	 Length 125	 scTM=0.91	 PDB ID=1f9i, TMscore=0.98
scTM ~ 0.5	 Length 69	 scTM=0.5	 PDB ID=7c78, TMscore=0.65
scTM < 0.25	 Length 120	 scTM=0.24	None found

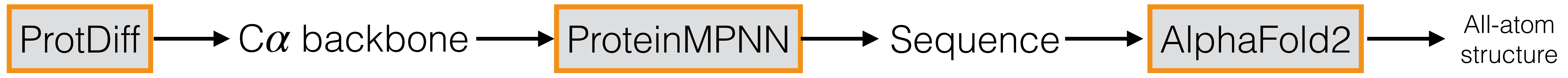
Unconditional sampling

Unconditional sampling

- Trained with around 4K short (<128 residue) proteins from PDB

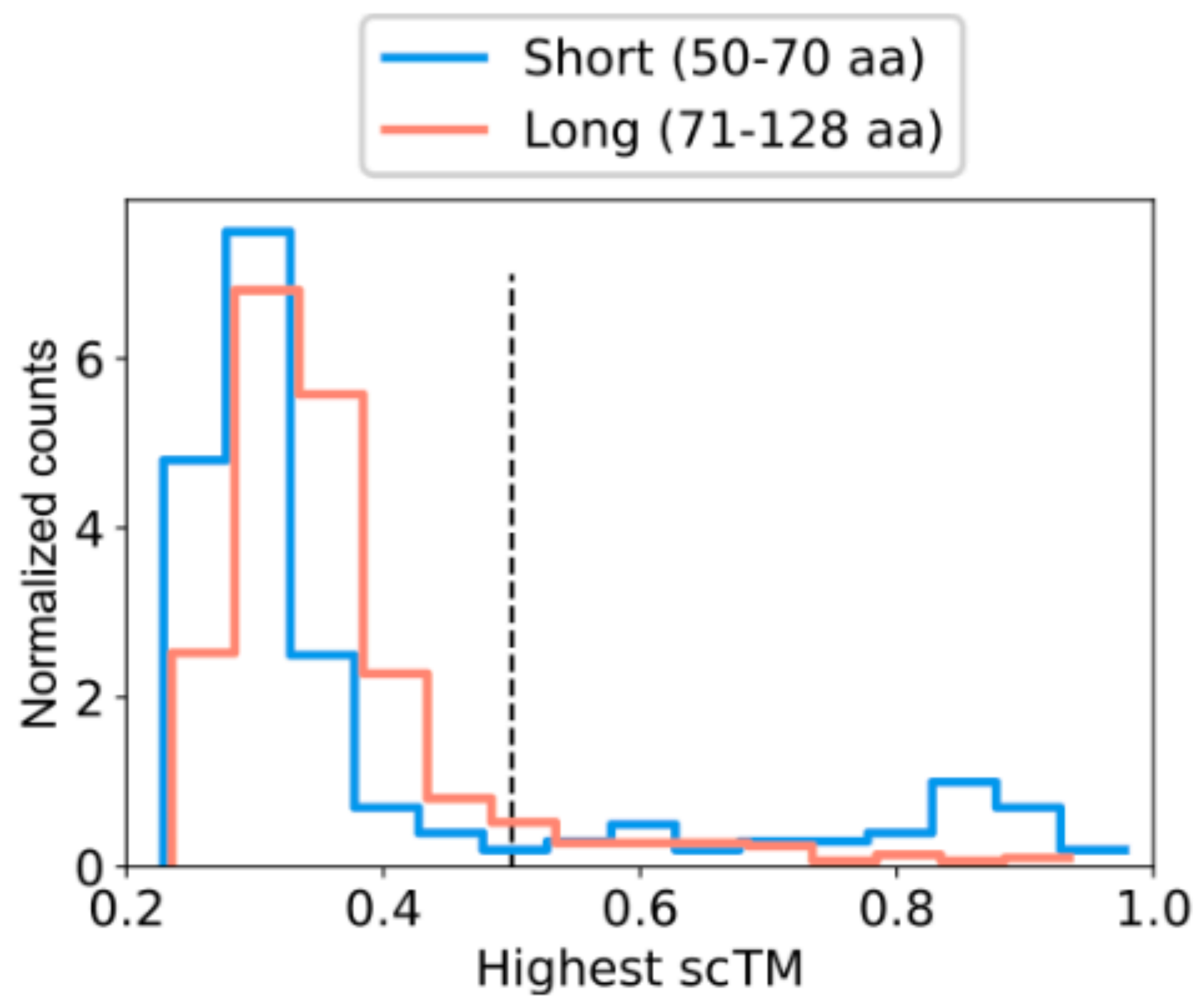
Unconditional sampling

- Trained with around 4K short (<128 residue) proteins from PDB
- Full sampling pipeline:



Unconditional sampling

- Trained with around 4K short (<128 residue) proteins from PDB
- Full sampling pipeline: $$\begin{tikzcd}\boxed{\text{ProtDiff}} \arrow[r] & C\alpha\text{ backbone} \arrow[r] & \boxed{\text{ProteinMPNN}} \arrow[r] & \text{Sequence} \arrow[r] & \boxed{\text{AlphaFold2}} \arrow[r] & \text{All-atom structure}\end{tikzcd}$$
- Compare the end structures through alignment (scTM)

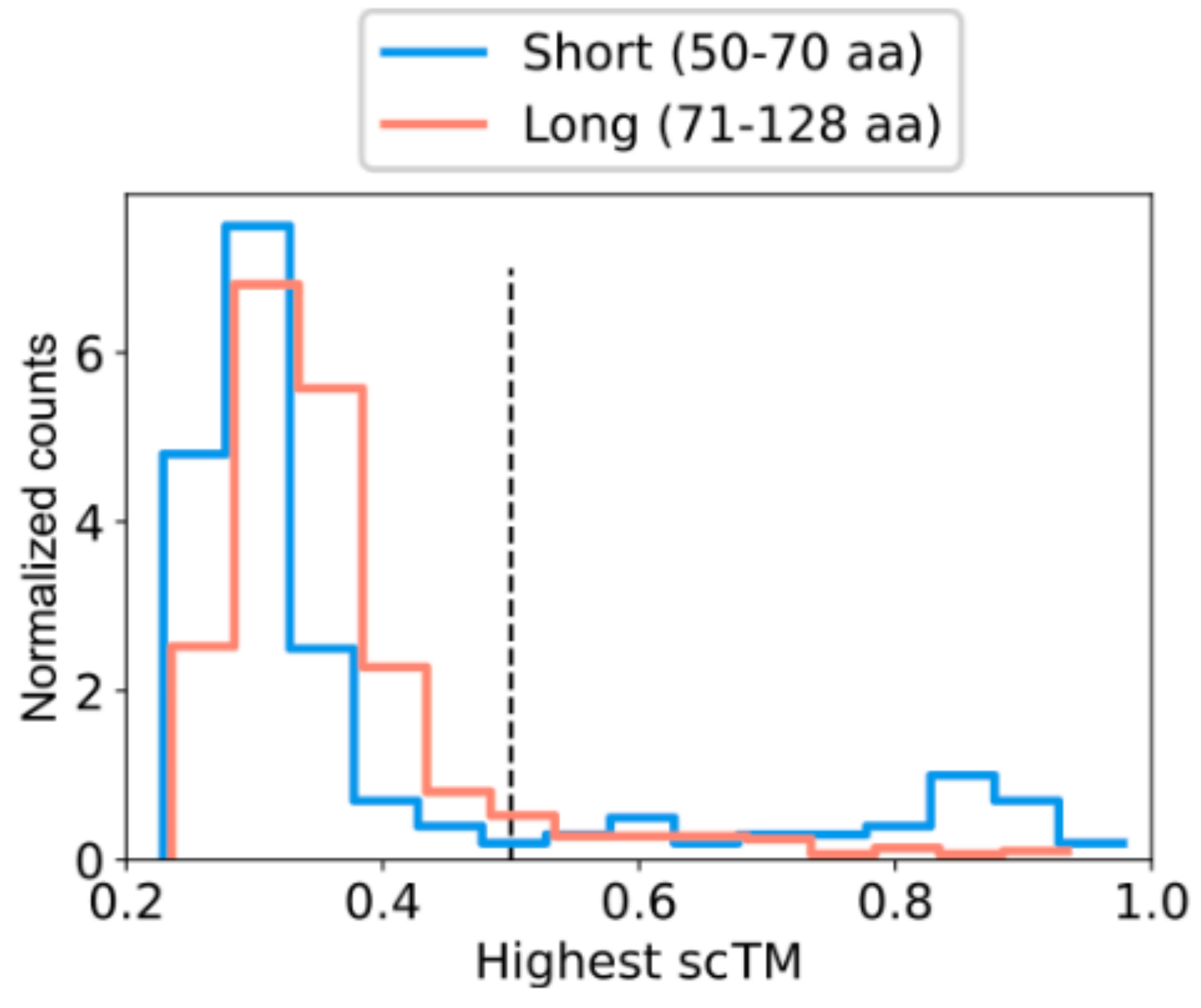


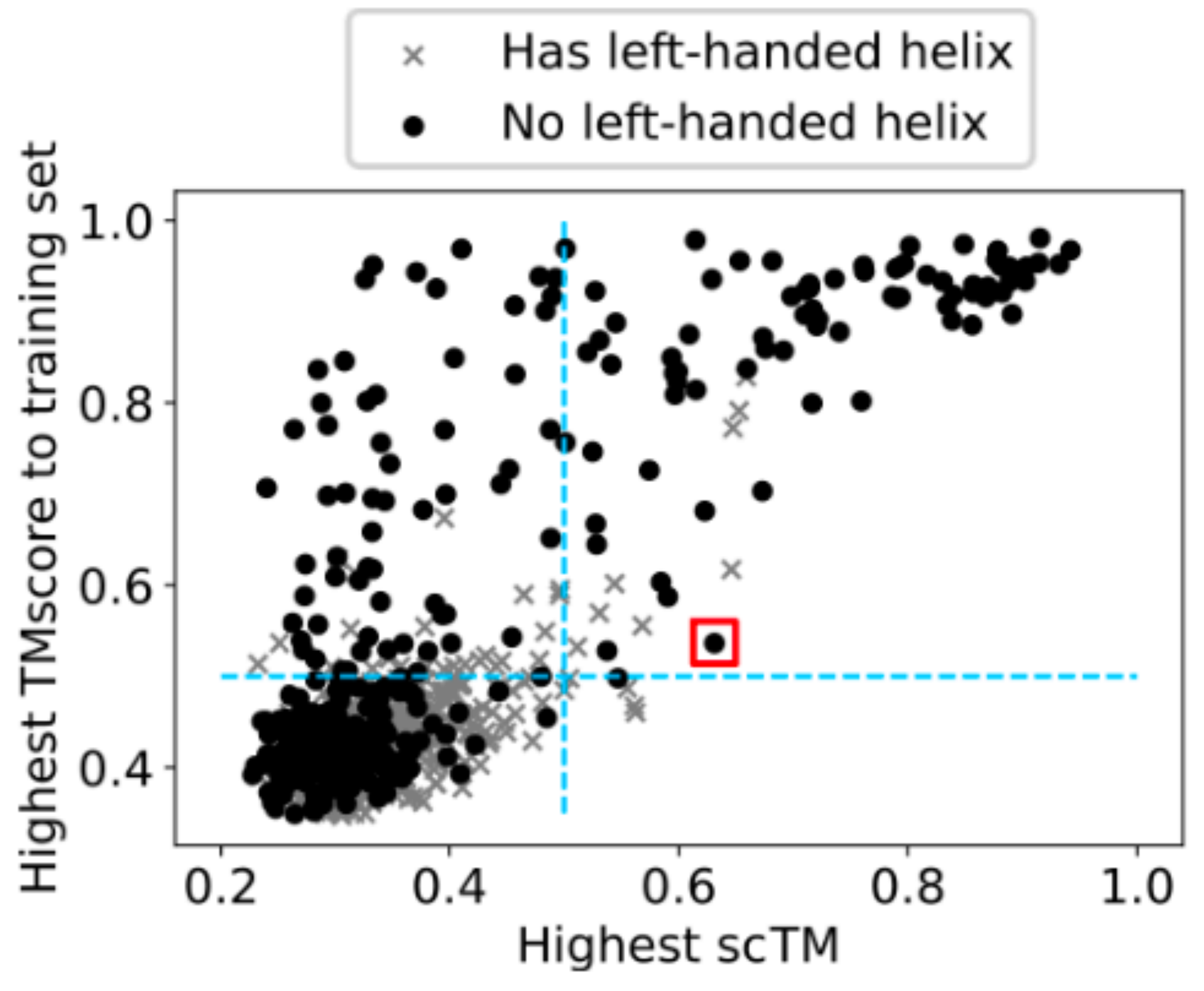
$\longrightarrow$

“designable”

Unconditional sampling

- Trained with around 4K short (<128 residue) proteins from PDB
- Full sampling pipeline: $$\begin{tikzcd} \boxed{\text{ProtDiff}} \arrow[r] & C\alpha\text{ backbone} \arrow[r] & \boxed{\text{ProteinMPNN}} \arrow[r] & \text{Sequence} \arrow[r] & \boxed{\text{AlphaFold2}} \arrow[r] & \text{All-atom structure} \end{tikzcd}$$
- Compare the end structures through alignment (scTM)



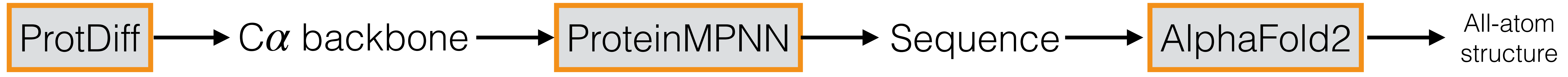


$\longrightarrow$

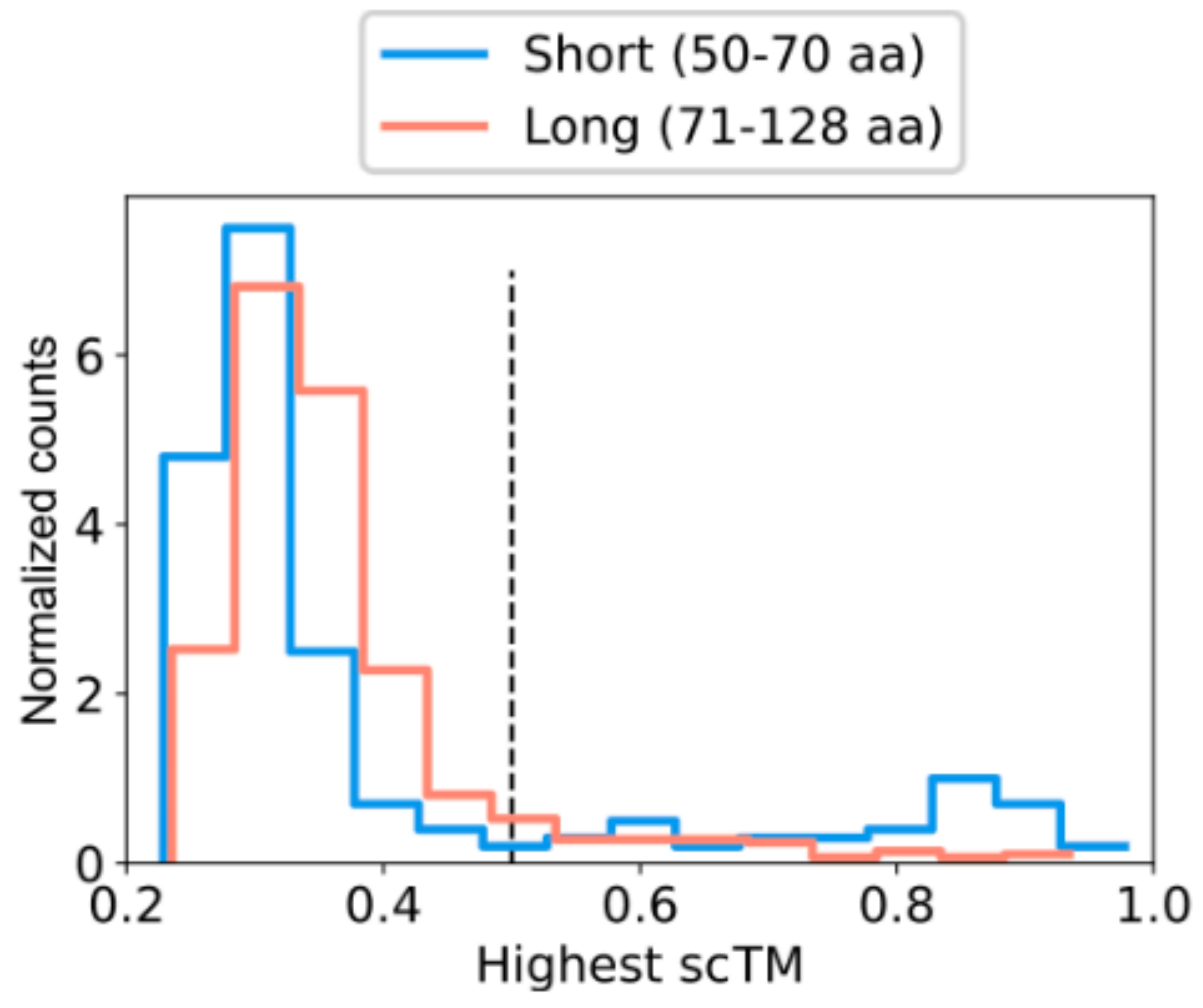
"designable"

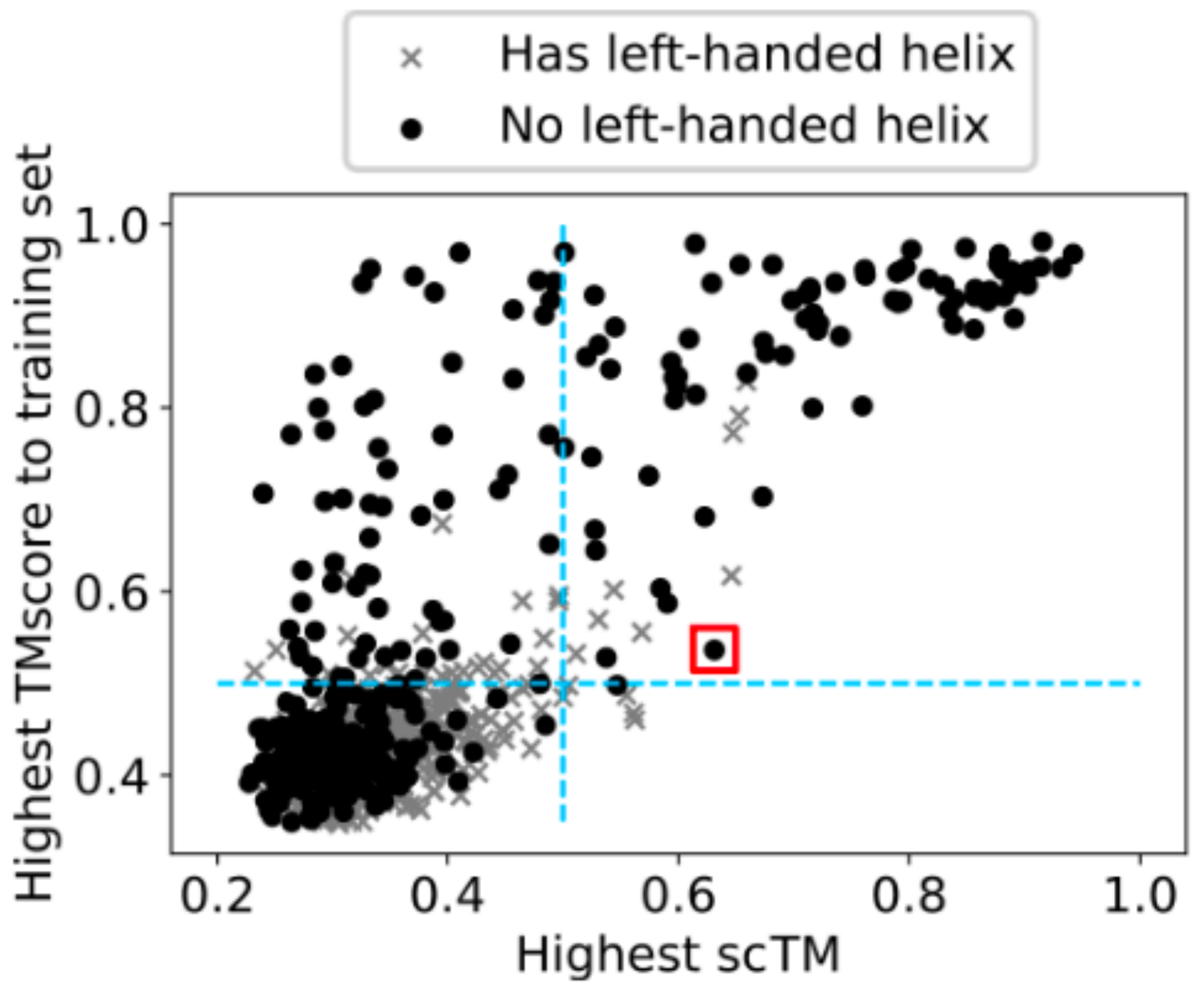
Unconditional sampling

- Trained with around 4K short (<128 residue) proteins from PDB
- Full sampling pipeline:



- Compare the end structures through alignment (scTM)



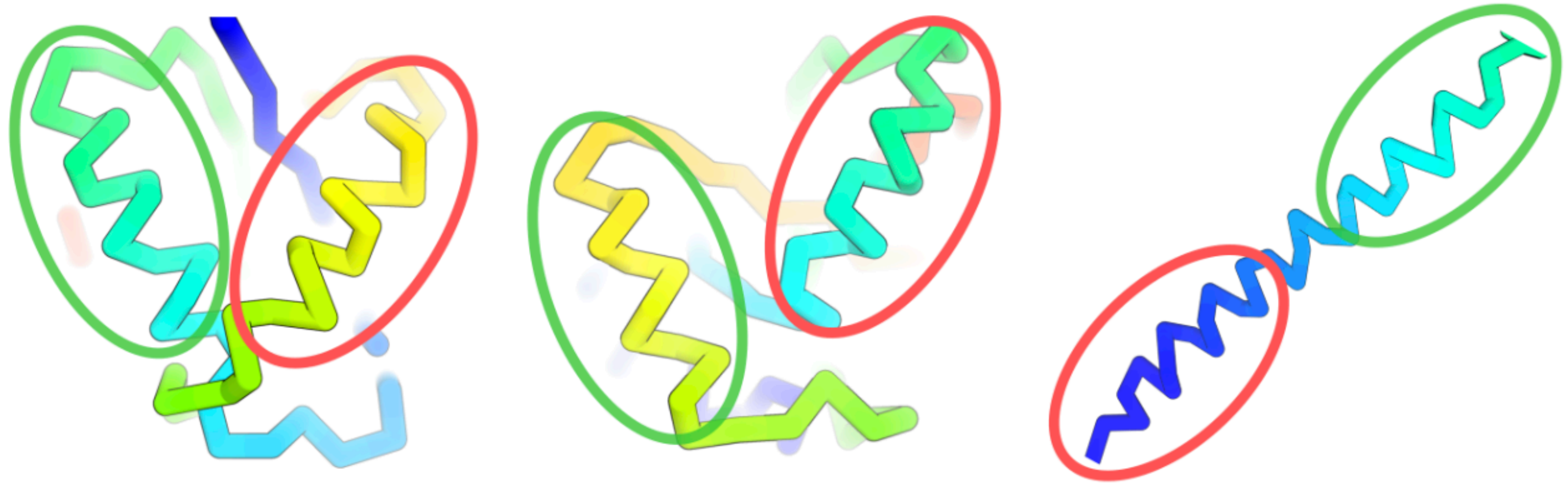


$\longrightarrow$

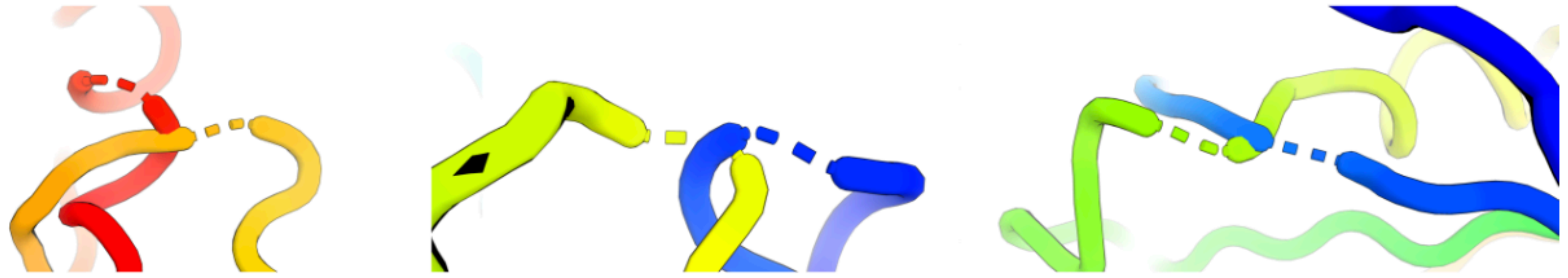
"designable"

Specific failure mode

- The model can generate left-handed helices (in red)

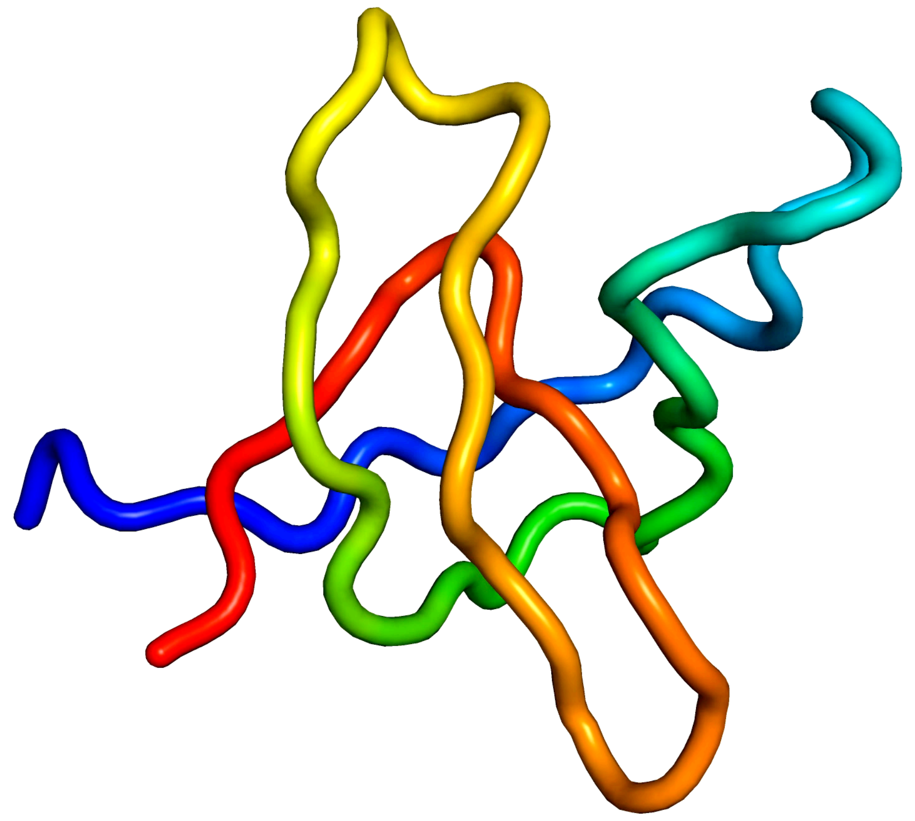


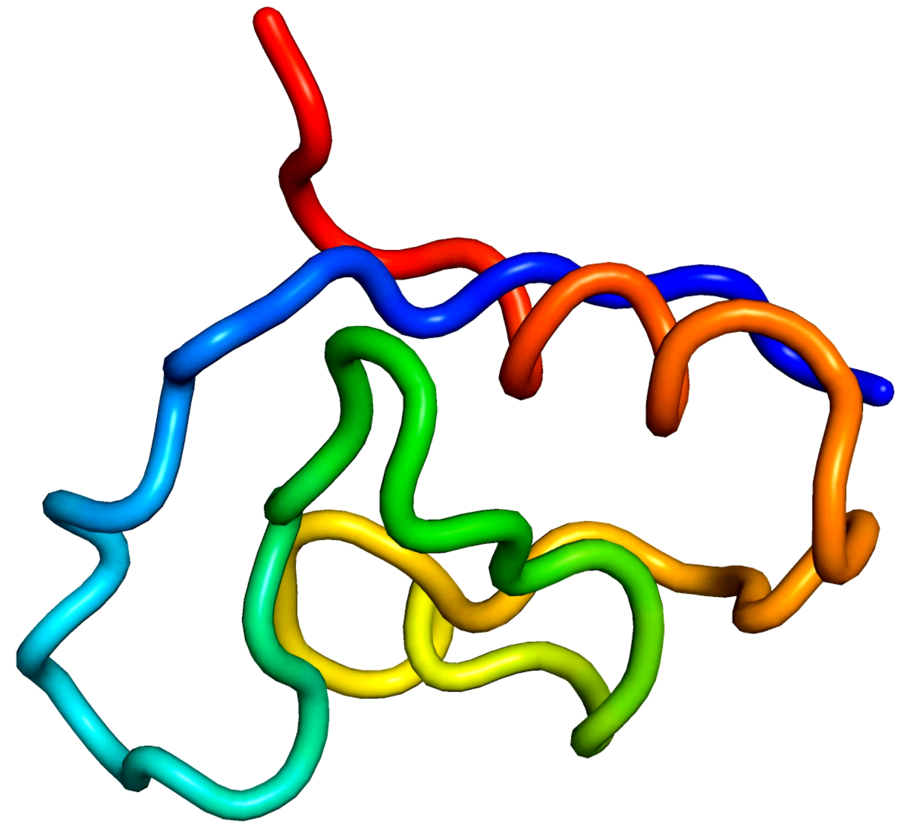
- Chain breaks can occur



Sample interpolations

- Noise interpolated samples

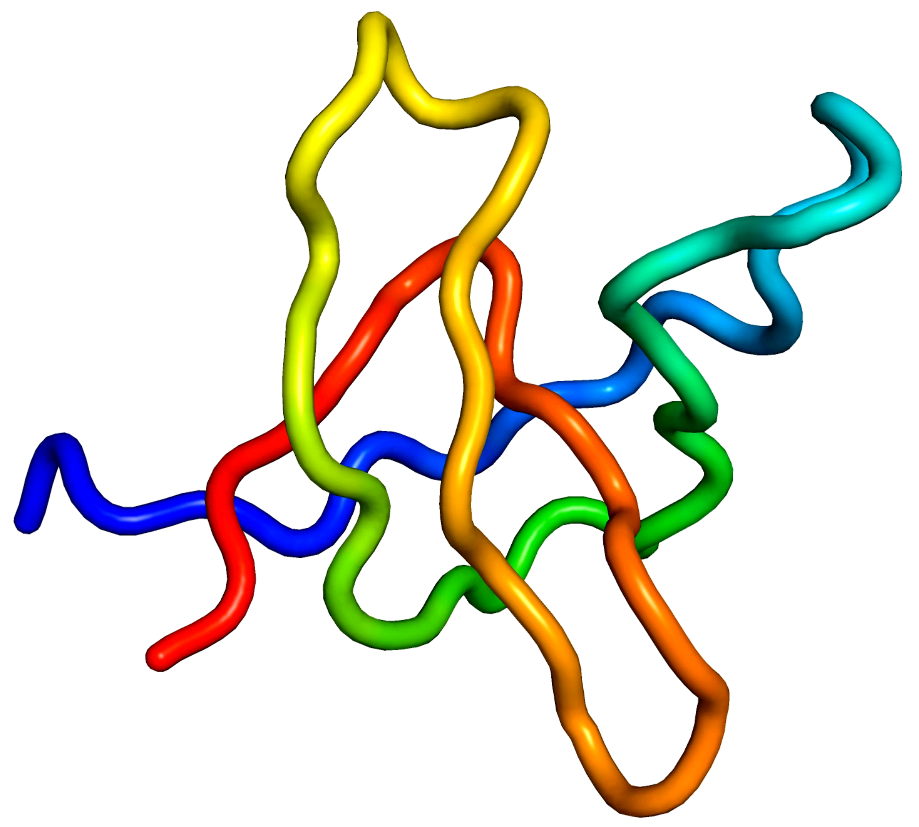


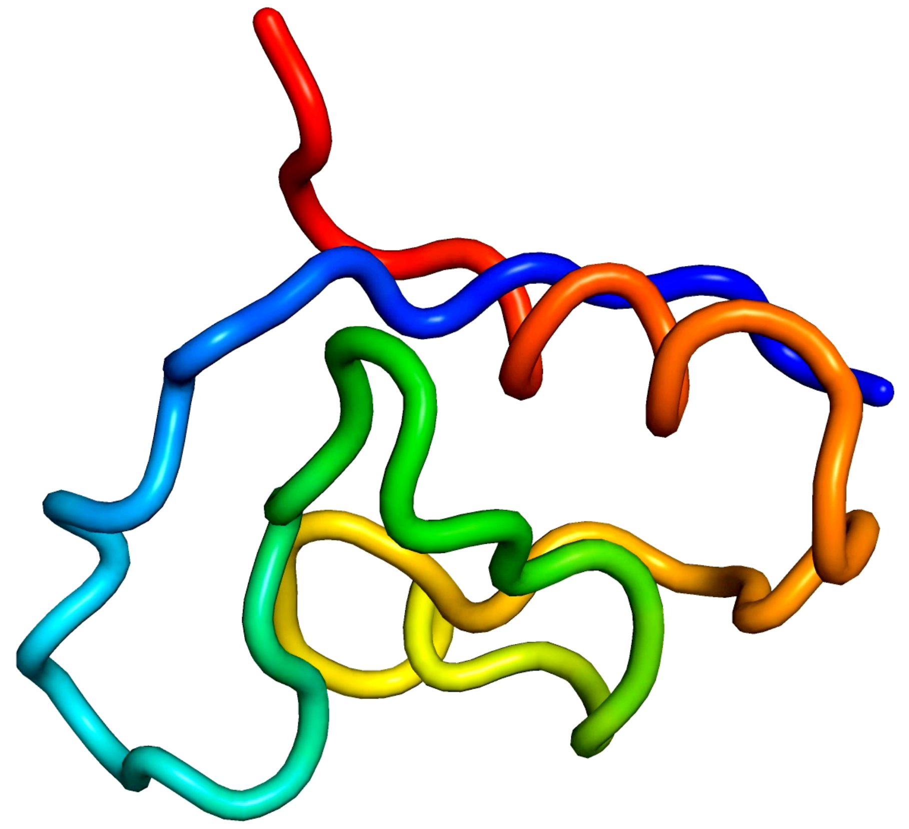


For samples with noise $\epsilon^{(0:T)}$ and $\tilde{\epsilon}^{(0:T)}$ we interpolate with noise set to $\sqrt{\alpha}\epsilon^{(0:T)} + \sqrt{1-\alpha}\tilde{\epsilon}^{(0:T)}$

Sample interpolations

- Noise interpolated samples





For samples with noise $\epsilon^{(0:T)}$ and $\tilde{\epsilon}^{(0:T)}$ we interpolate with noise set to $\sqrt{\alpha}\epsilon^{(0:T)} + \sqrt{1-\alpha}\tilde{\epsilon}^{(0:T)}$

Roadmap

Learning $p_\theta(x)$ [ProtDiff]

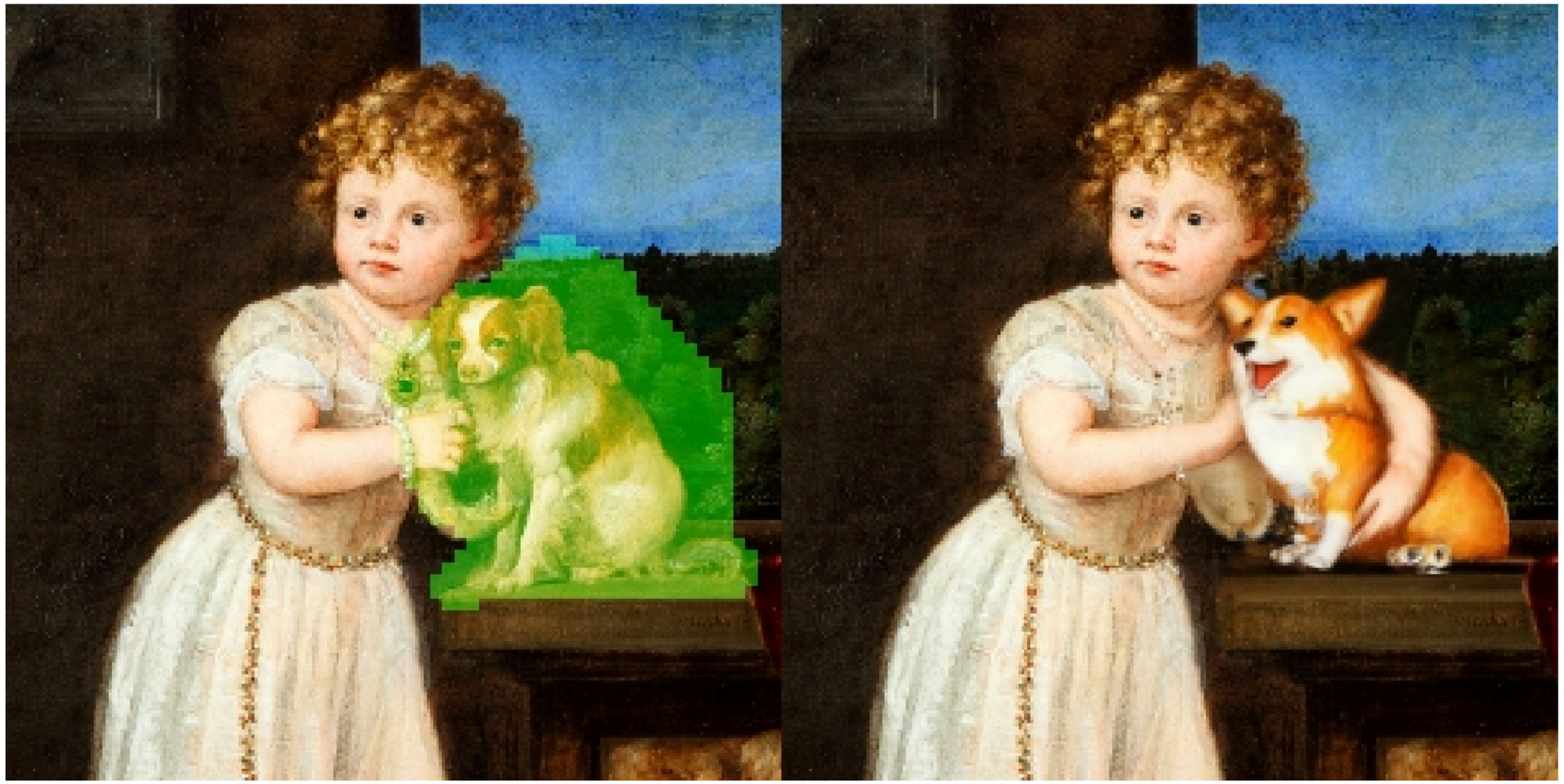
- Diffusion generative modeling background
- Adapting diffusion for protein backbones
- Model performance and limitations

Sampling $x_S \sim p_\theta(x_S \mid x_M)$ [SMCDiff]

- Why conditional sampling vs. "naive" in-painting?
- Sequential Monte Carlo for exact sampling in the large-compute limit

Limitations, Related Work, and Future directions

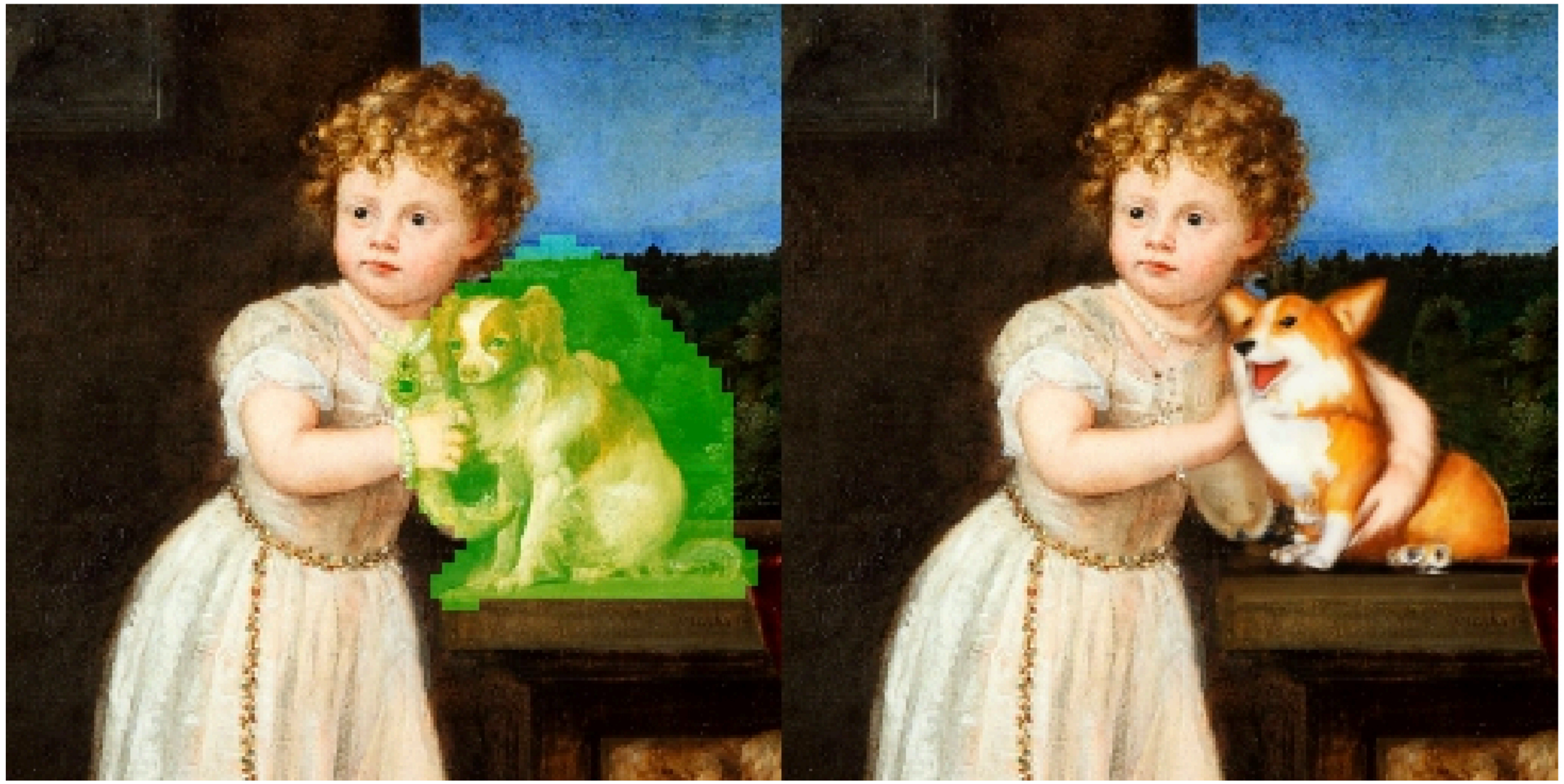
Inpainting with Diffusion Models



“A girl hugging a corgi on a pedestal”

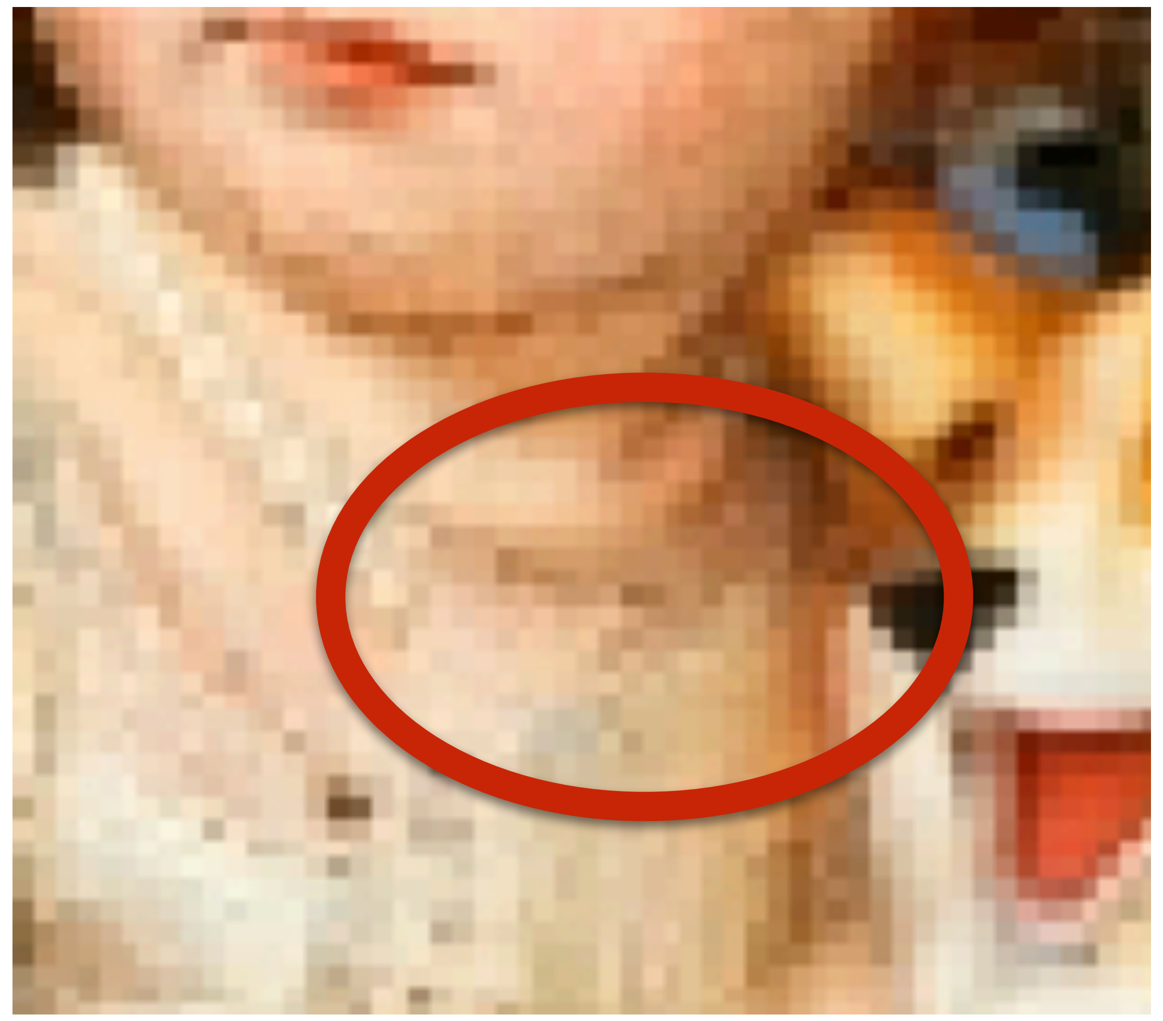
Nichol et al. “GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models” (2022)

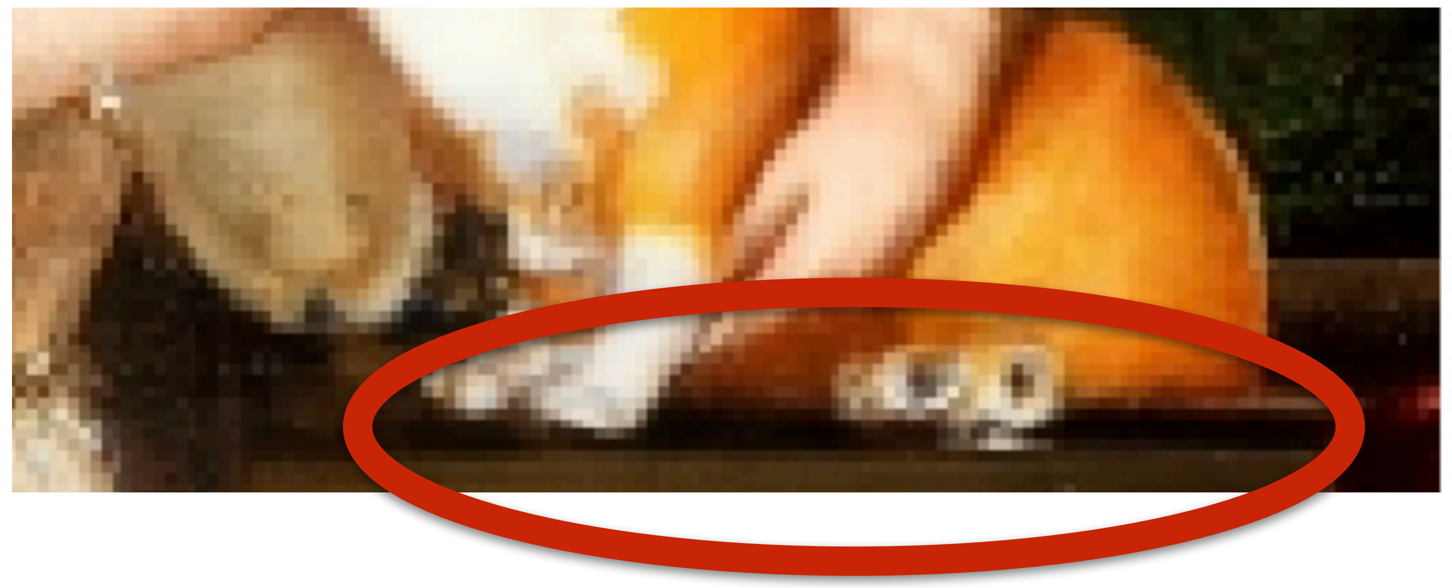
Inpainting with Diffusion Models



"A girl hugging a corgi on a pedestal"

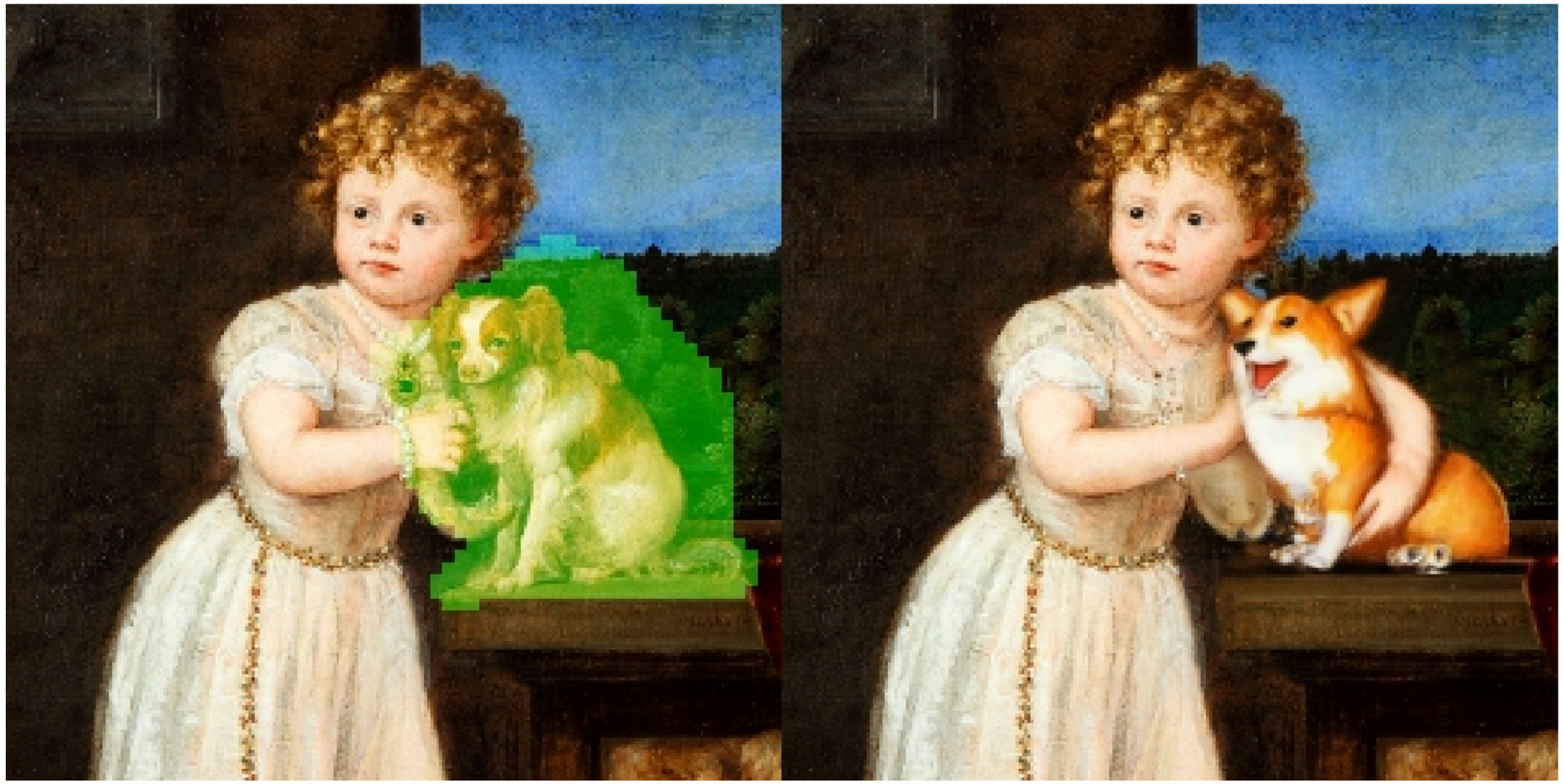
- Artifacts & edge effects in state-of-art inpainting





Nichol et al. "GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models" (2022)

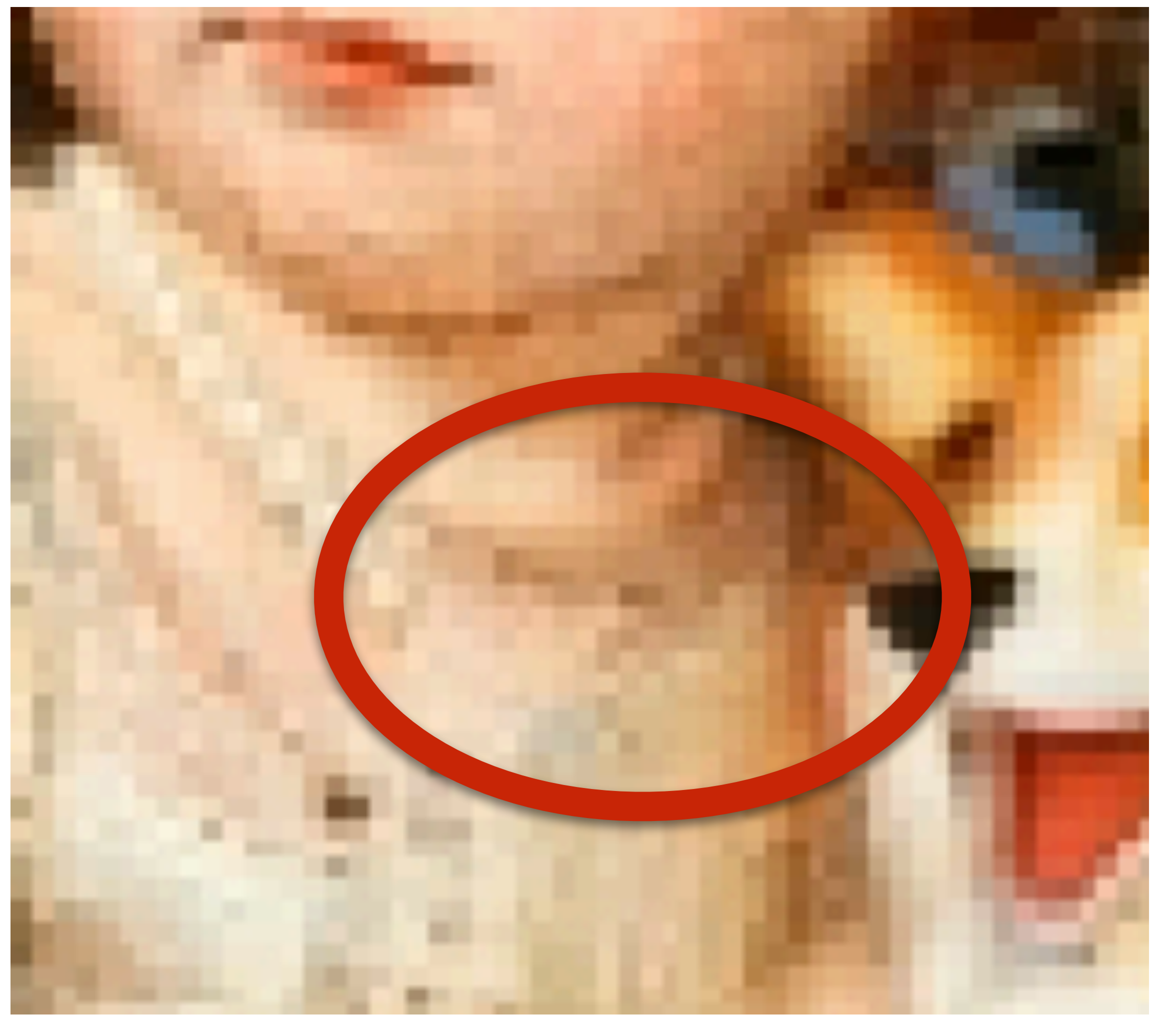
Inpainting with Diffusion Models



“A girl hugging a corgi on a pedestal”

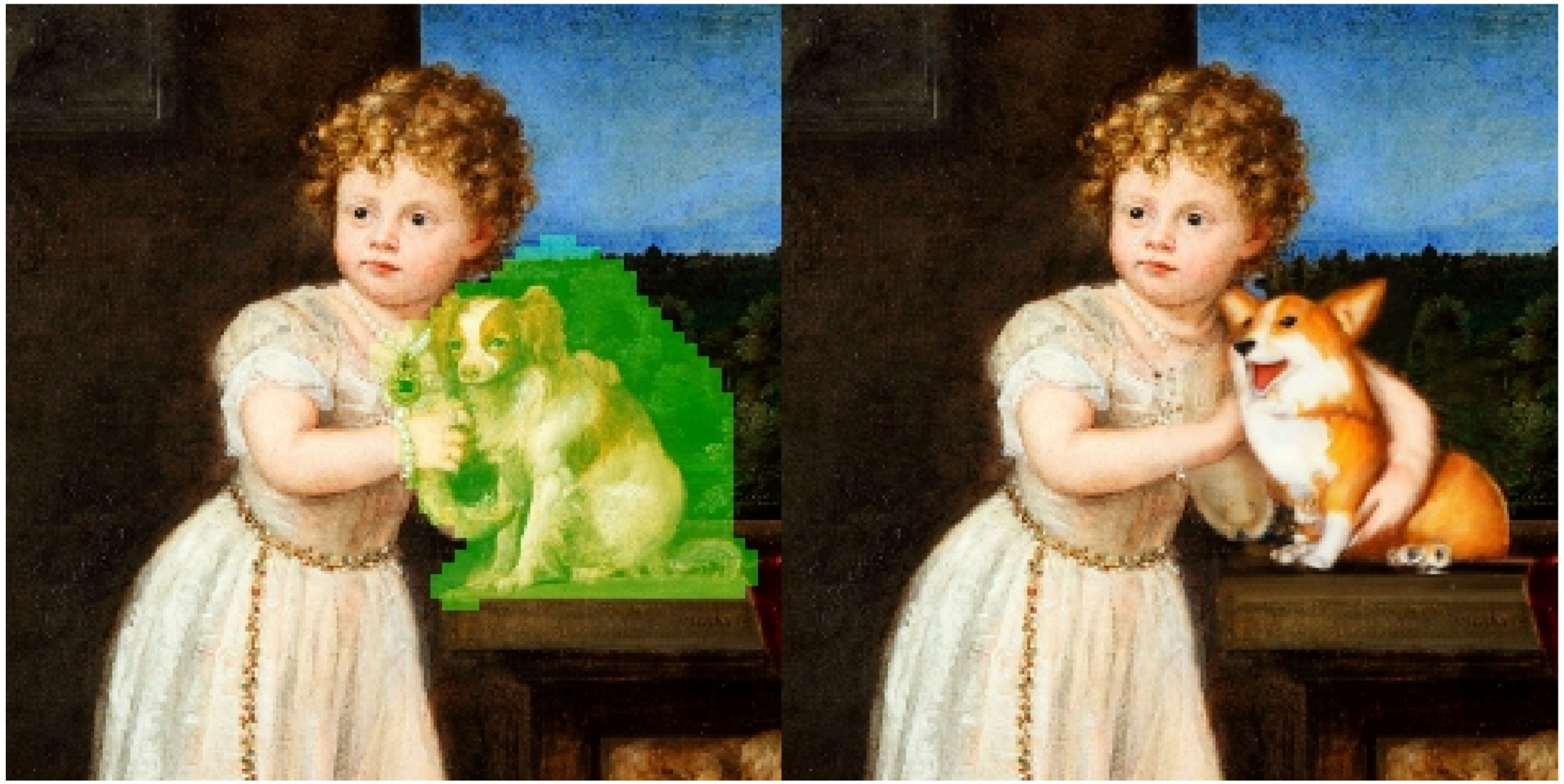
- For protein design, small errors break designability

- Artifacts & edge effects in state-of-art inpainting



Nichol et al. “GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models” (2022)

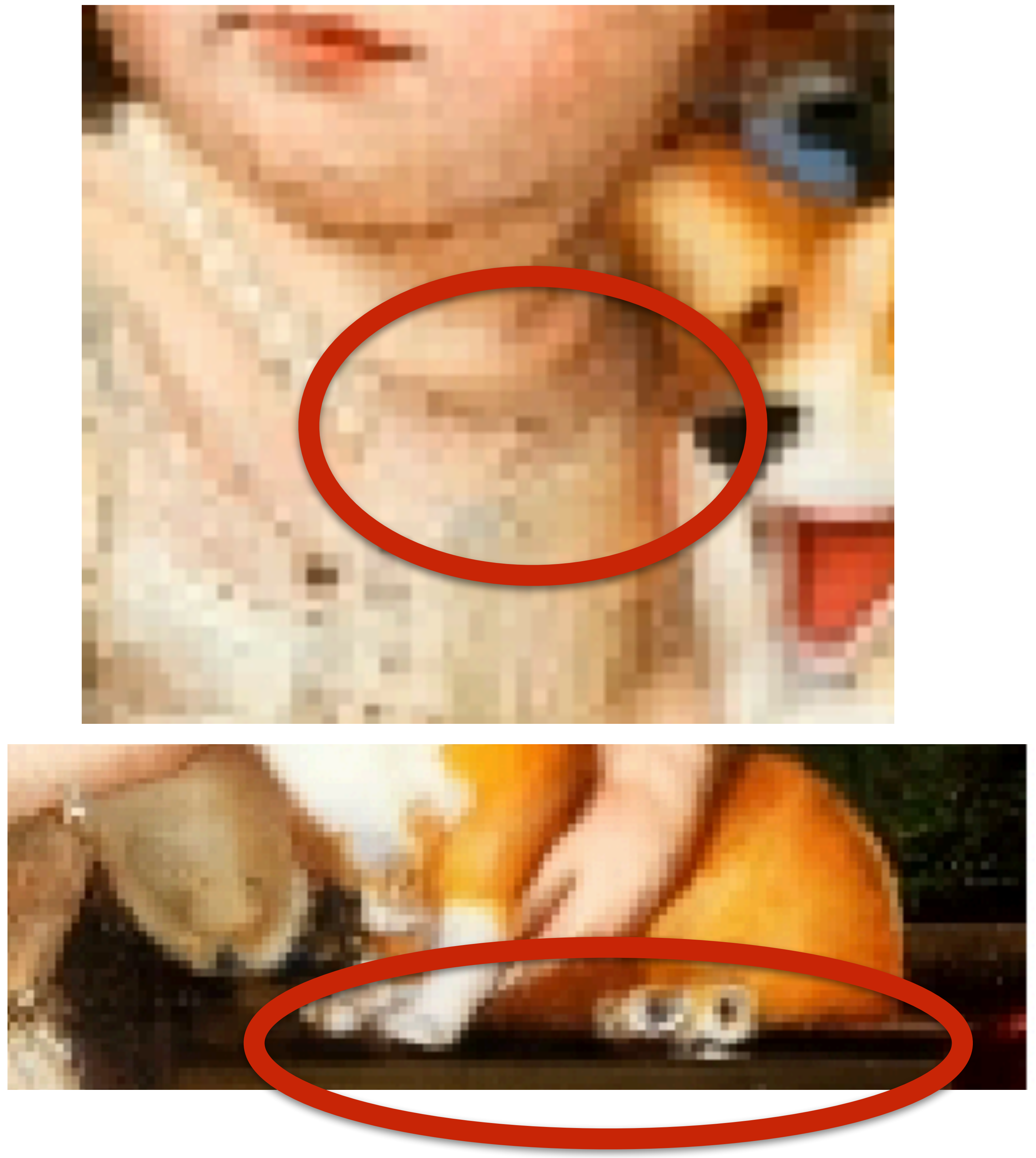
Inpainting with Diffusion Models



"A girl hugging a corgi on a pedestal"

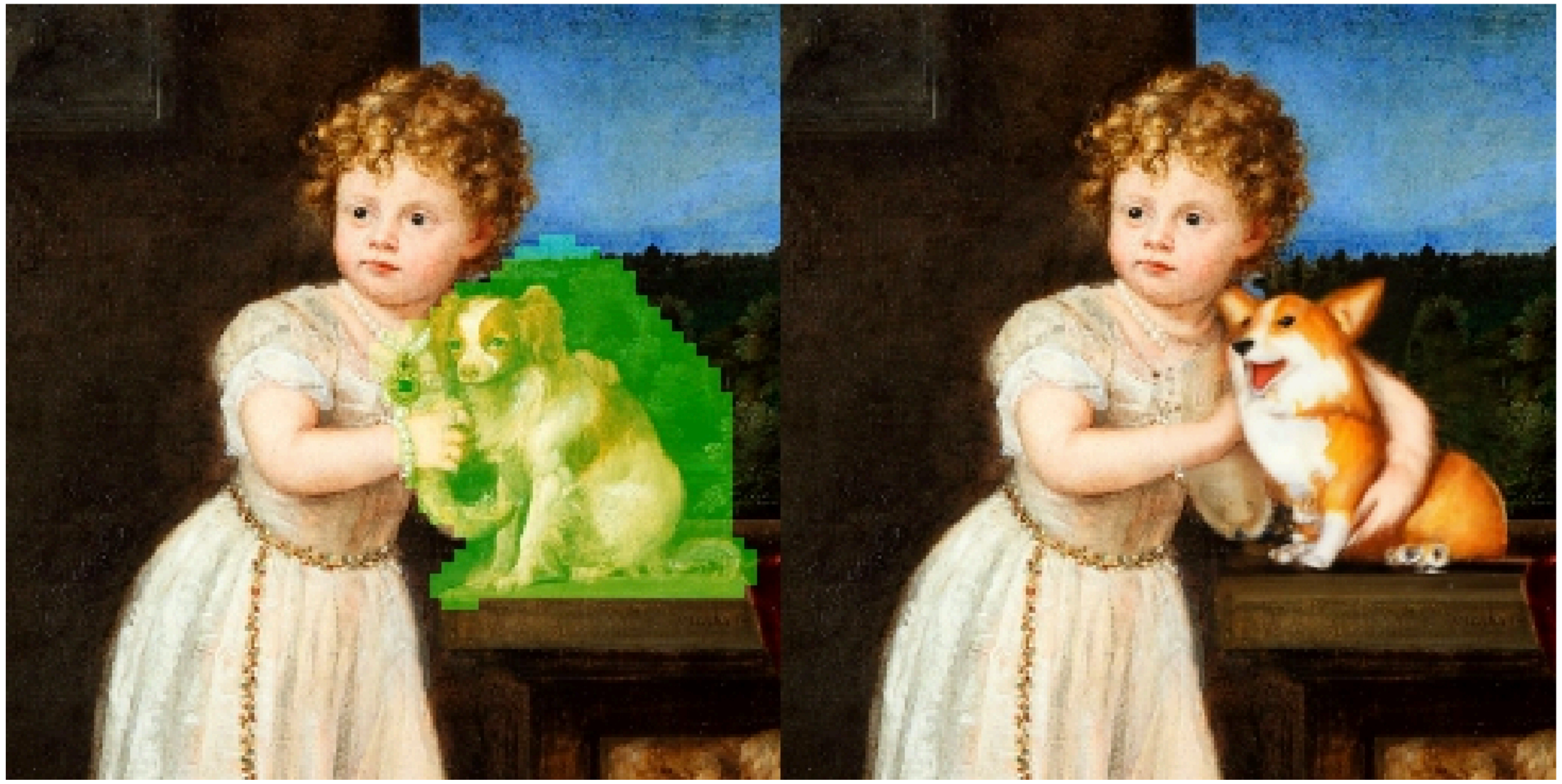
- For protein design, small errors break designability
- Why does this work at all?

- Artifacts & edge effects in state-of-art inpainting



Nichol et al. "GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models" (2022)

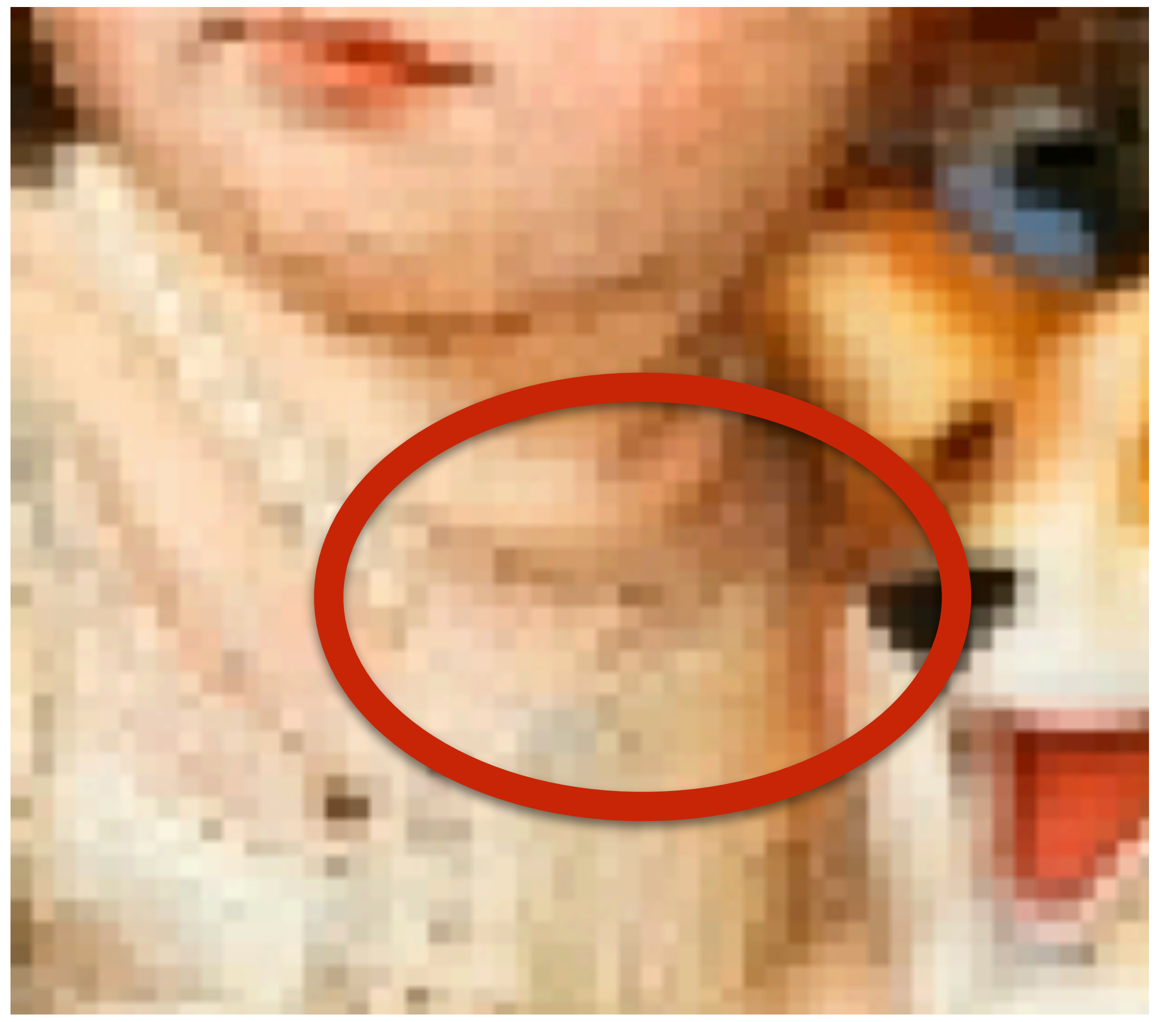
Inpainting with Diffusion Models


"A girl hugging a corgi on a pedestal"

- For protein design, small errors break designability
- Why does this work at all?

Hypothesis: Inpainting approximates conditional sampling.
Approximation error —> artifacts.

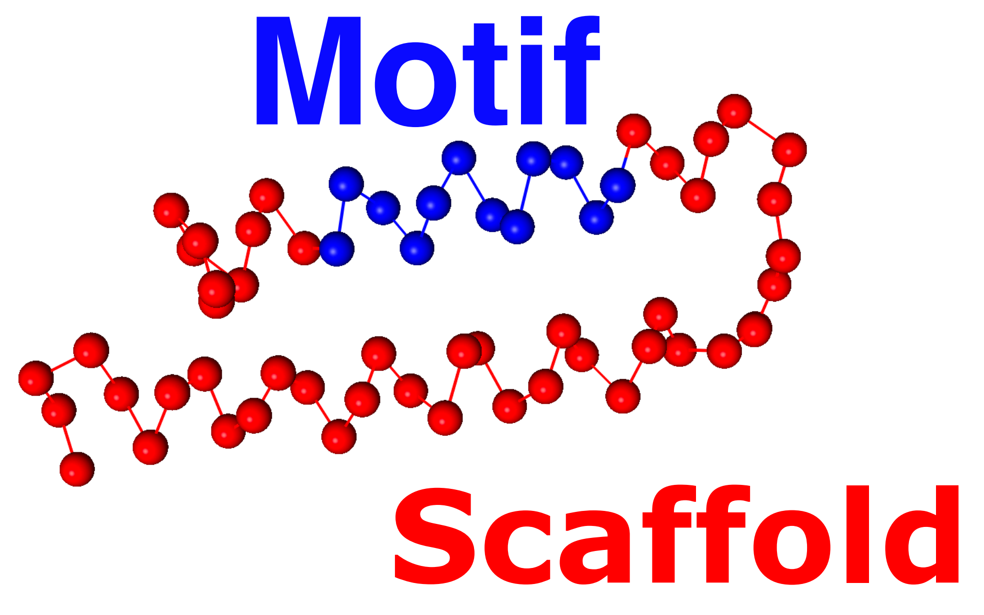
- Artifacts & edge effects in state-of-art inpainting



Nichol et al. "GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models" (2022)

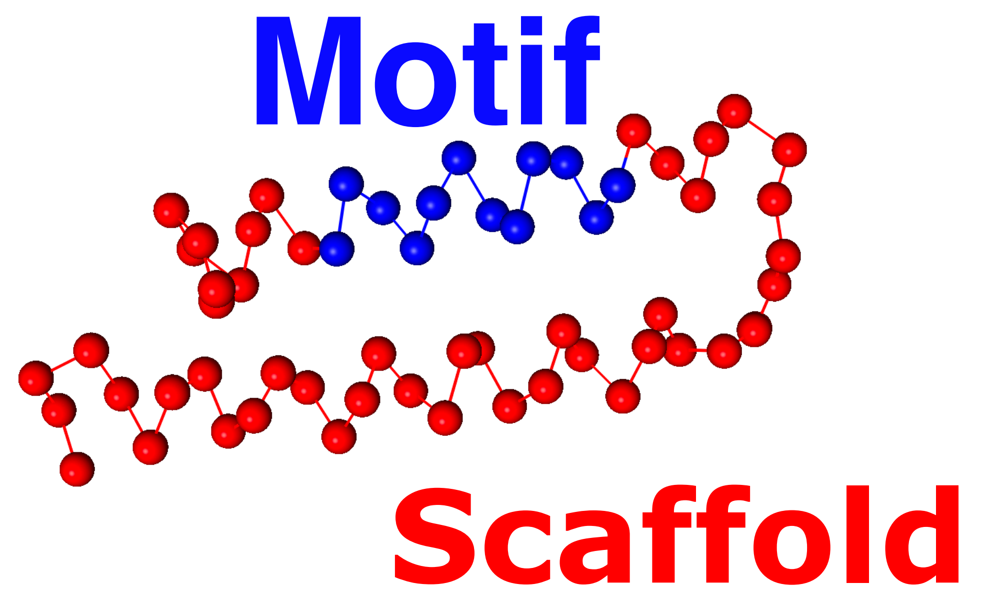
Conditional Sampling to Fix Inpainting?

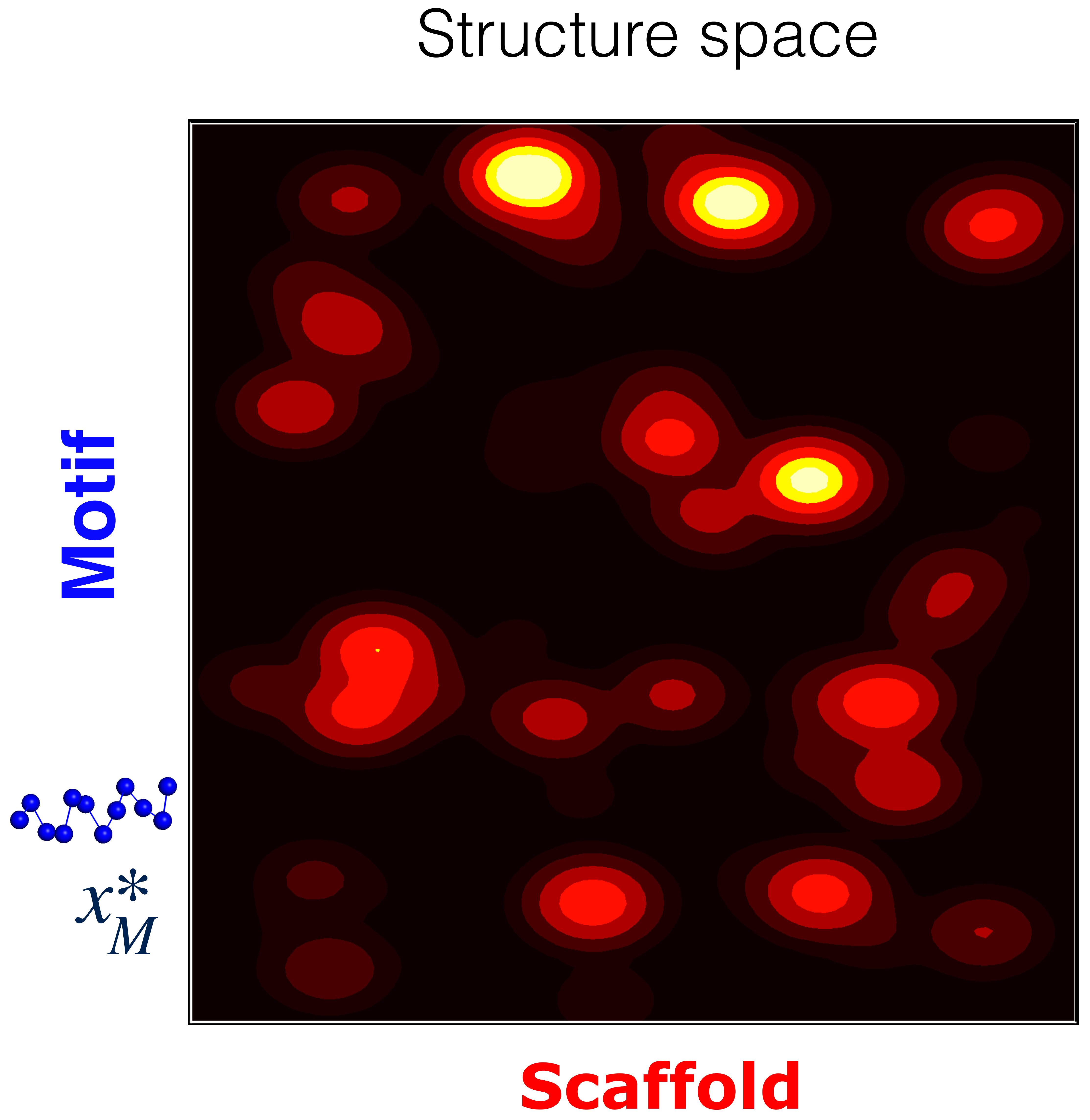
- Partition structure as $x = [x_M, x_S]$



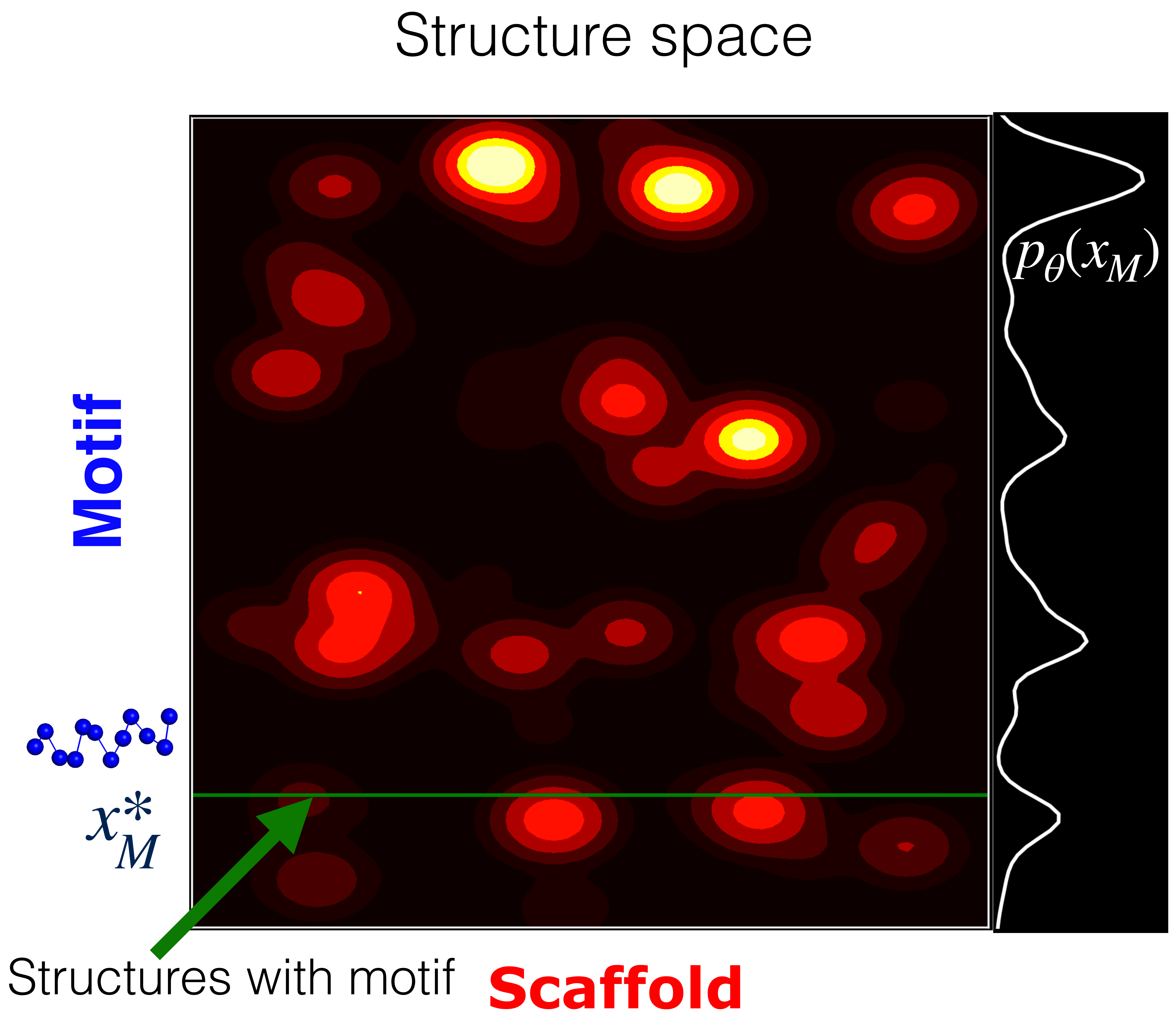
Conditional Sampling to Fix Inpainting?

- Partition structure as $x = [x_M, x_S]$

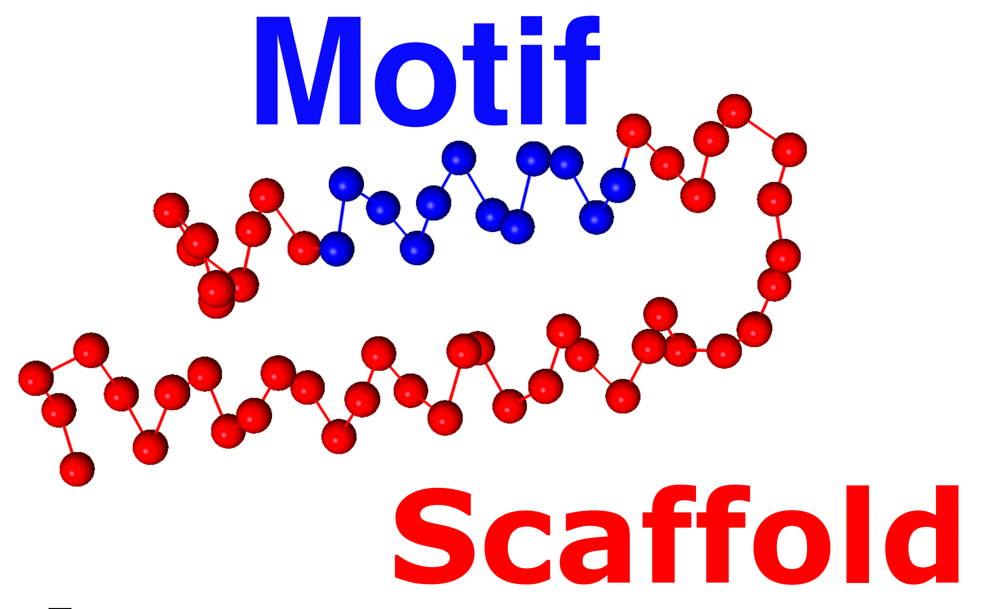




Conditional Sampling to Fix Inpainting?



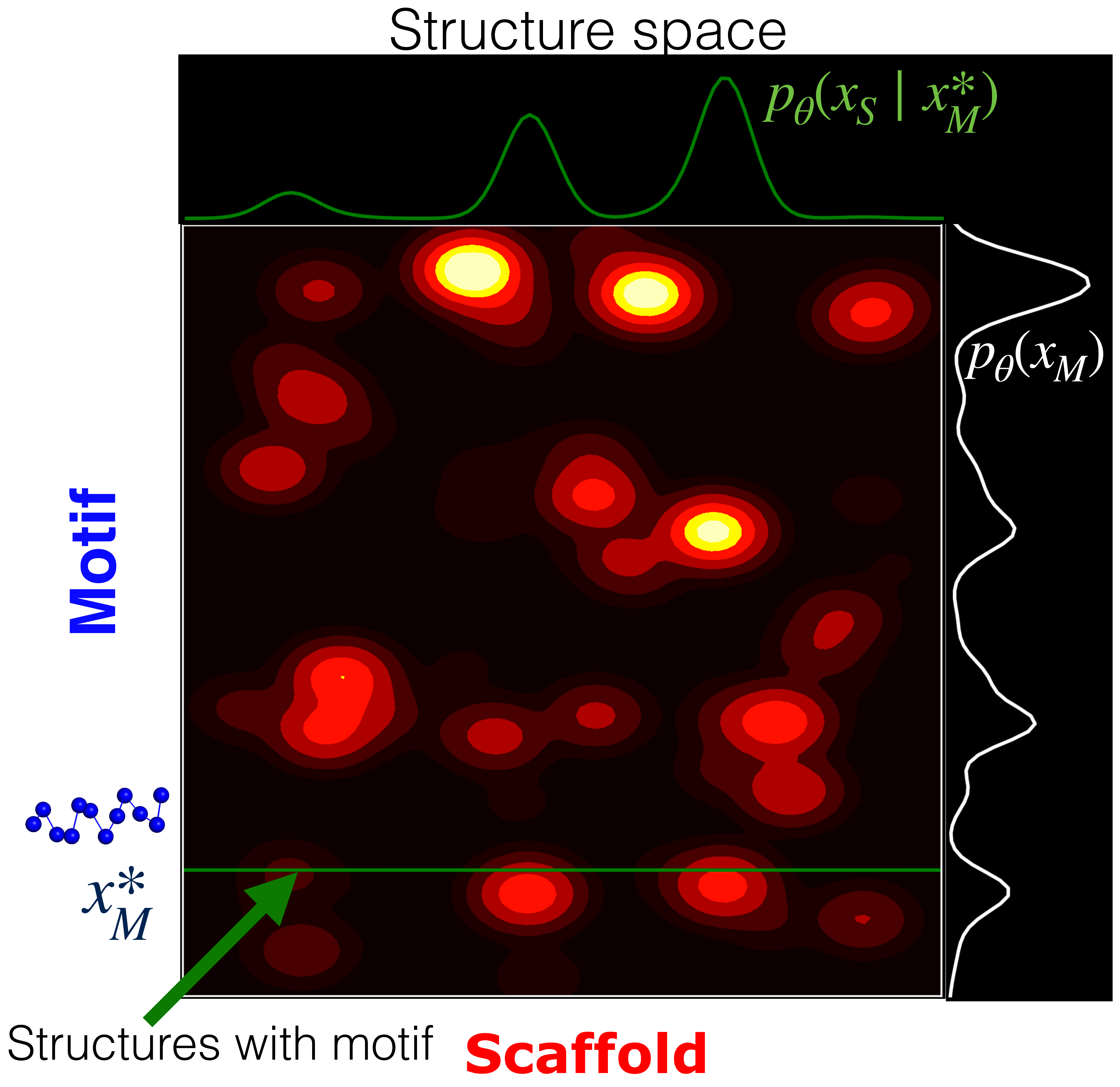
- Partition structure as $x = [x_M, x_S]$



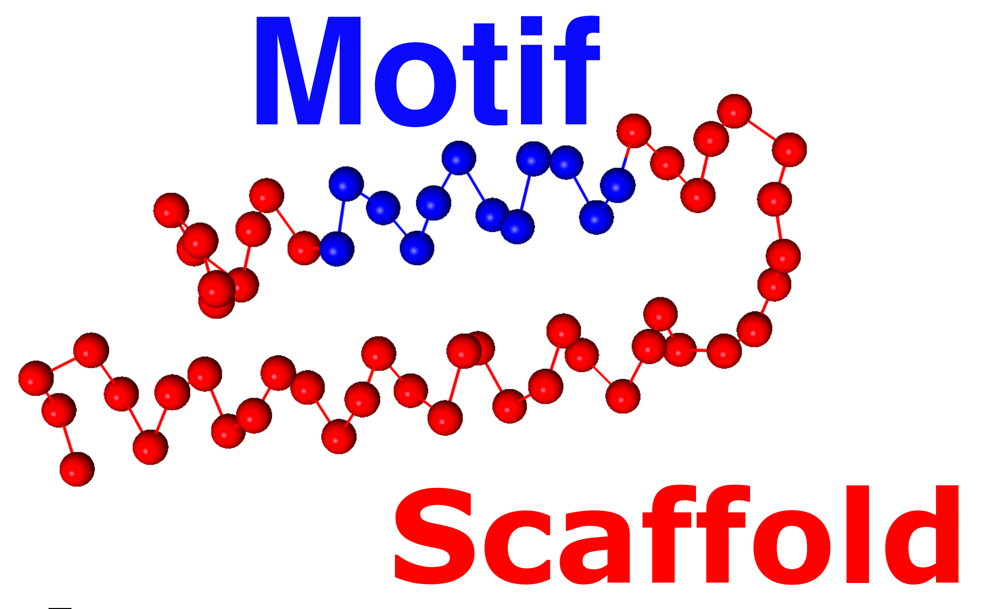
Our rationale: Assume

1. $p_\theta(x) > 0$ only if x is designable, and
2. $p_\theta(x_M^*) > 0$, for motif of interest x_M^*.

Conditional Sampling to Fix Inpainting?



- Partition structure as $x = [x_M, x_S]$

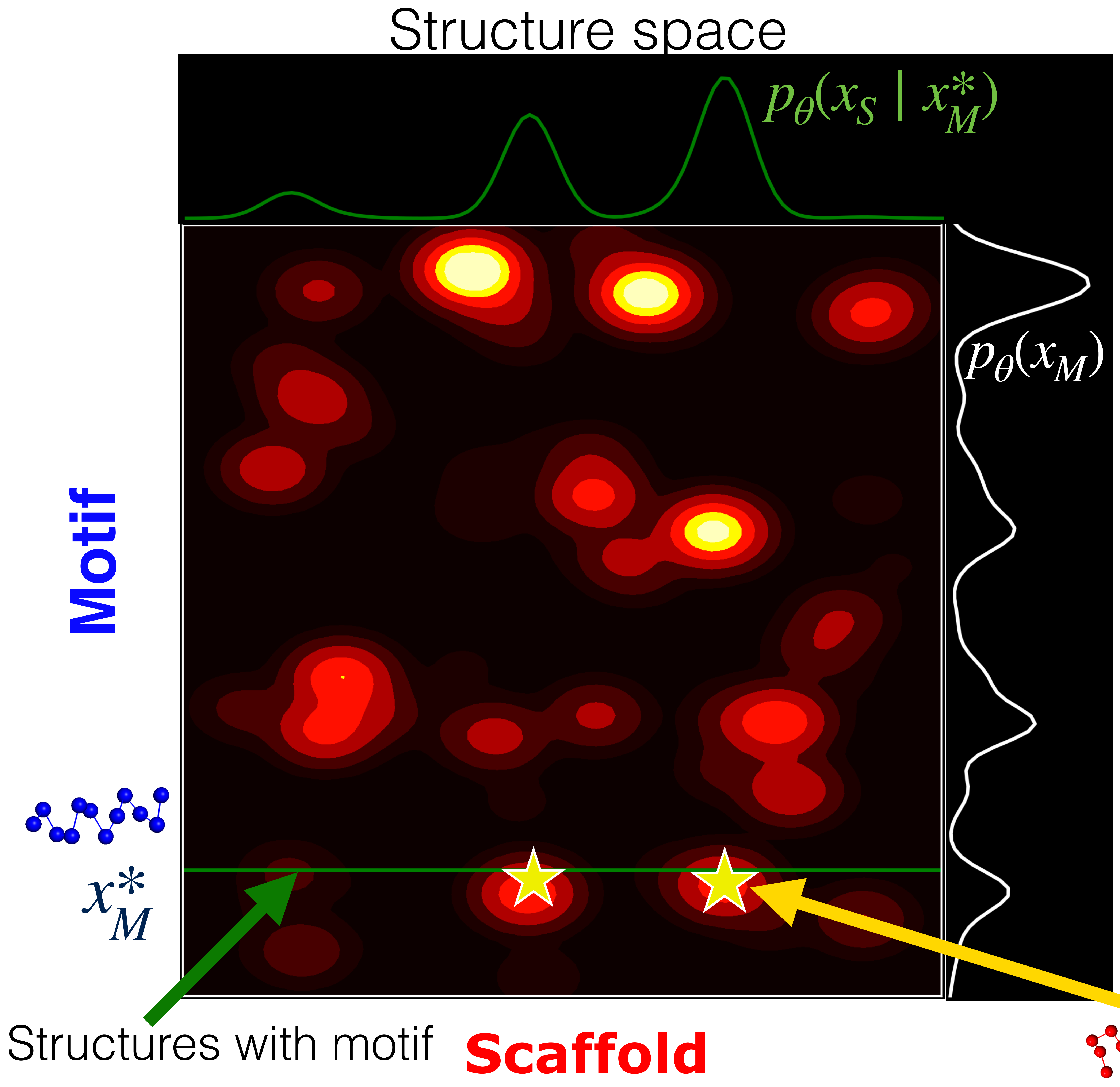


Our rationale: Assume

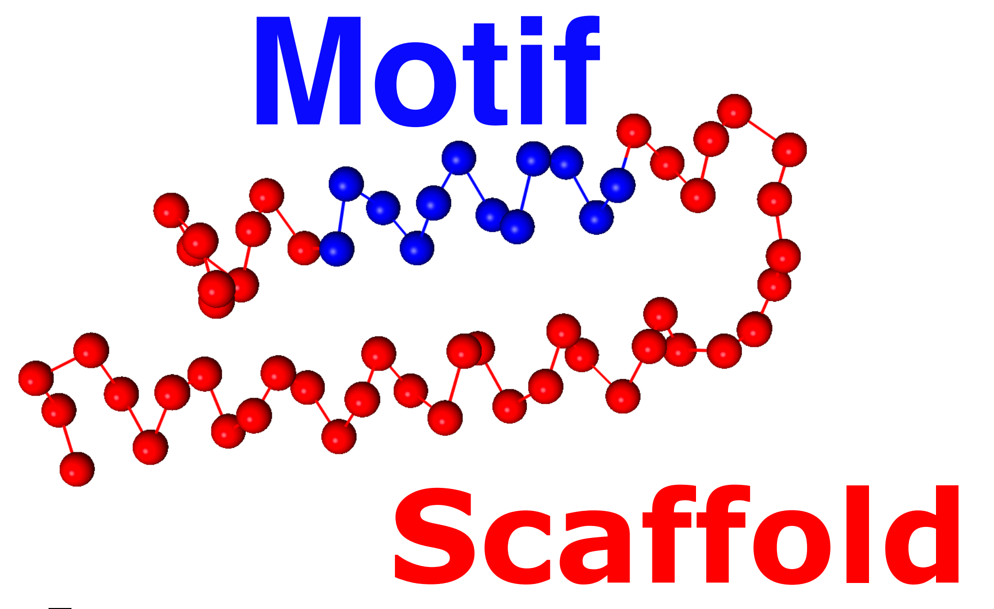
1. $p_\theta(x) > 0$ only if x is designable, and
2. $p_\theta(x_M^*) > 0$, for motif of interest x_M^*.

Then for any $x_S \sim p_\theta(x_S \mid x_M^*)$, $x = [x_M^*, x_S]$ is designable!

Conditional Sampling to Fix Inpainting?



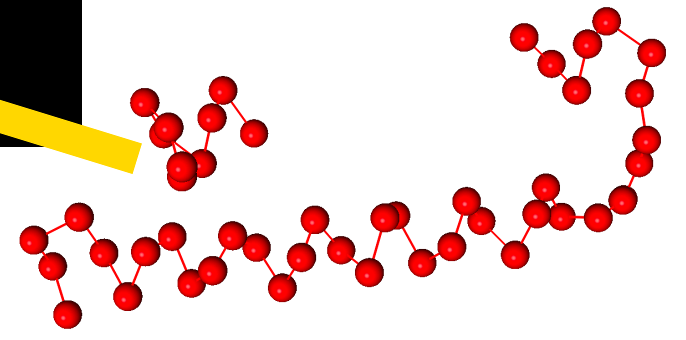
- Partition structure as $x = [x_M, x_S]$



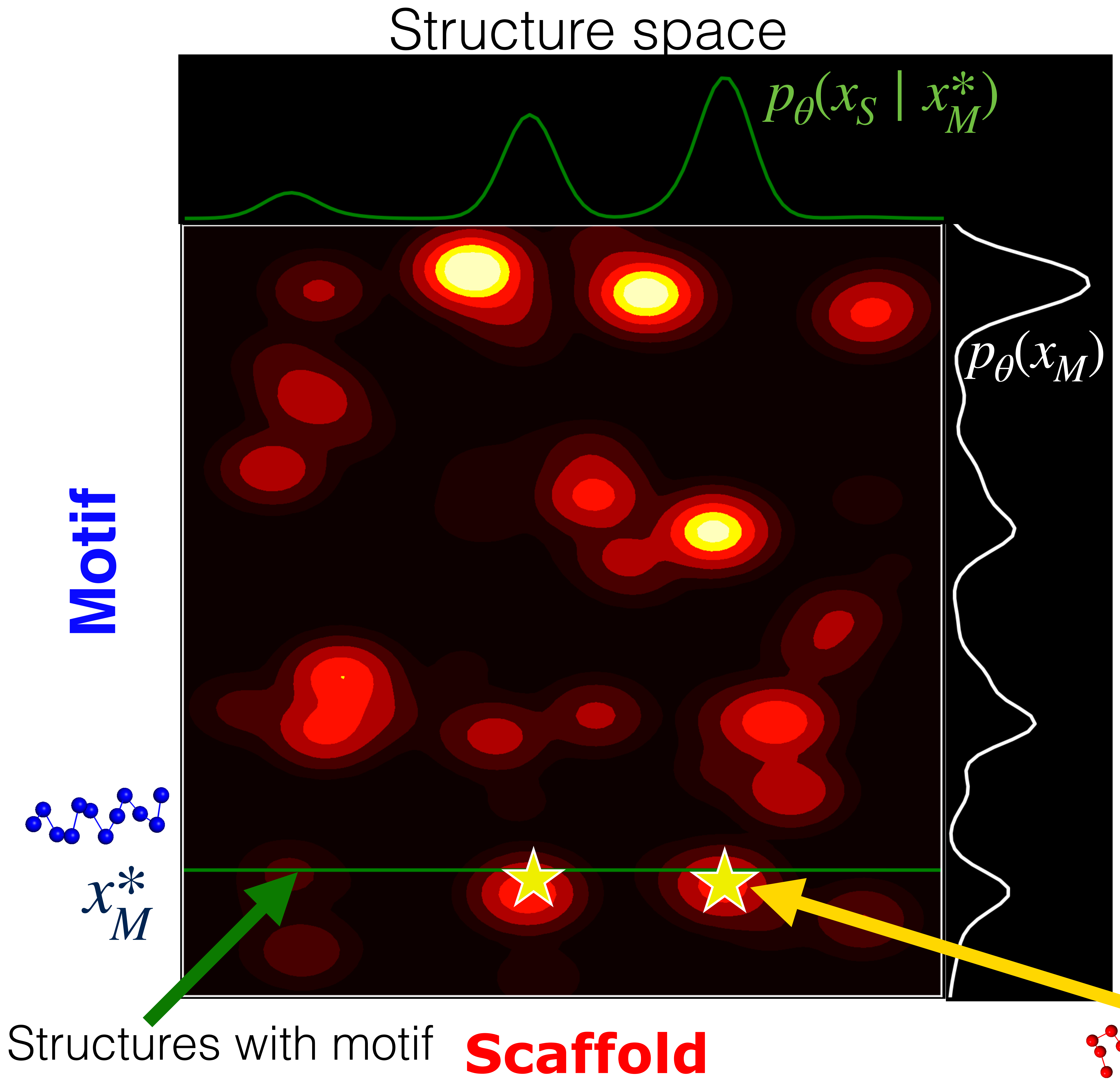
Our rationale: Assume

1. $p_\theta(x) > 0$ only if x is designable, and
2. $p_\theta(x_M^*) > 0$, for motif of interest x_M^*.

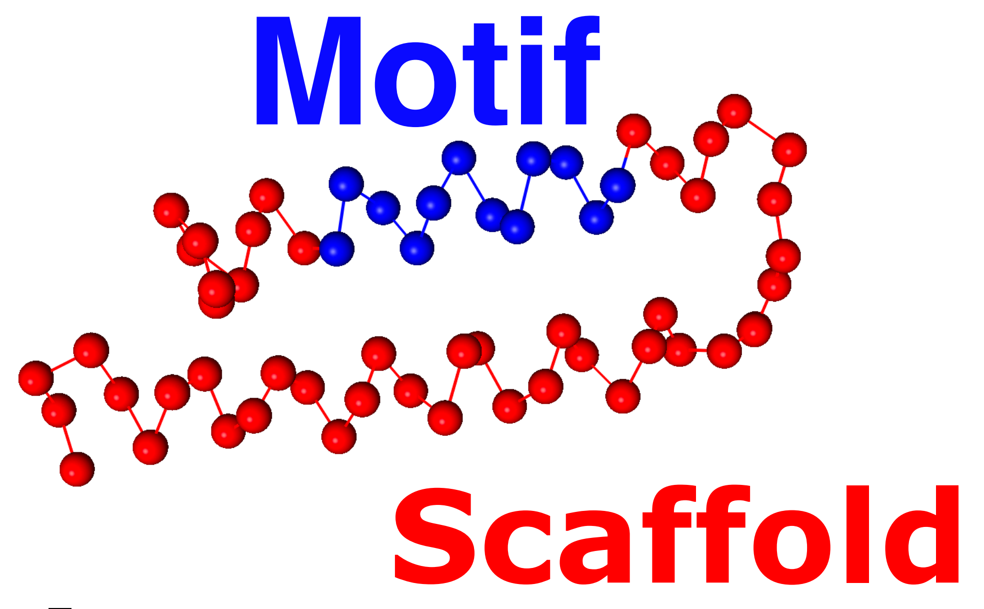
Then for any $x_S \sim p_\theta(x_S \mid x_M^*)$, $x = [x_M^*, x_S]$ is designable!



Conditional Sampling to Fix Inpainting?



- Partition structure as $x = [x_M, x_S]$

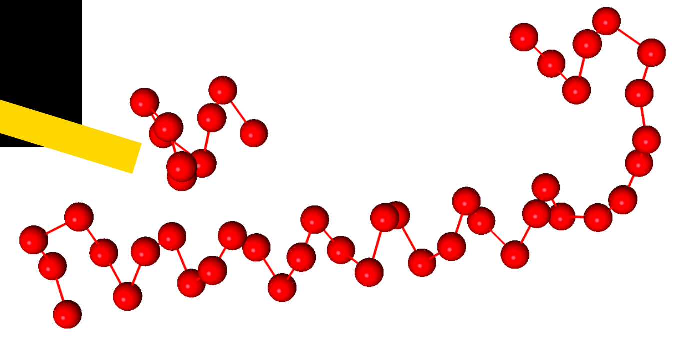


Our rationale: Assume

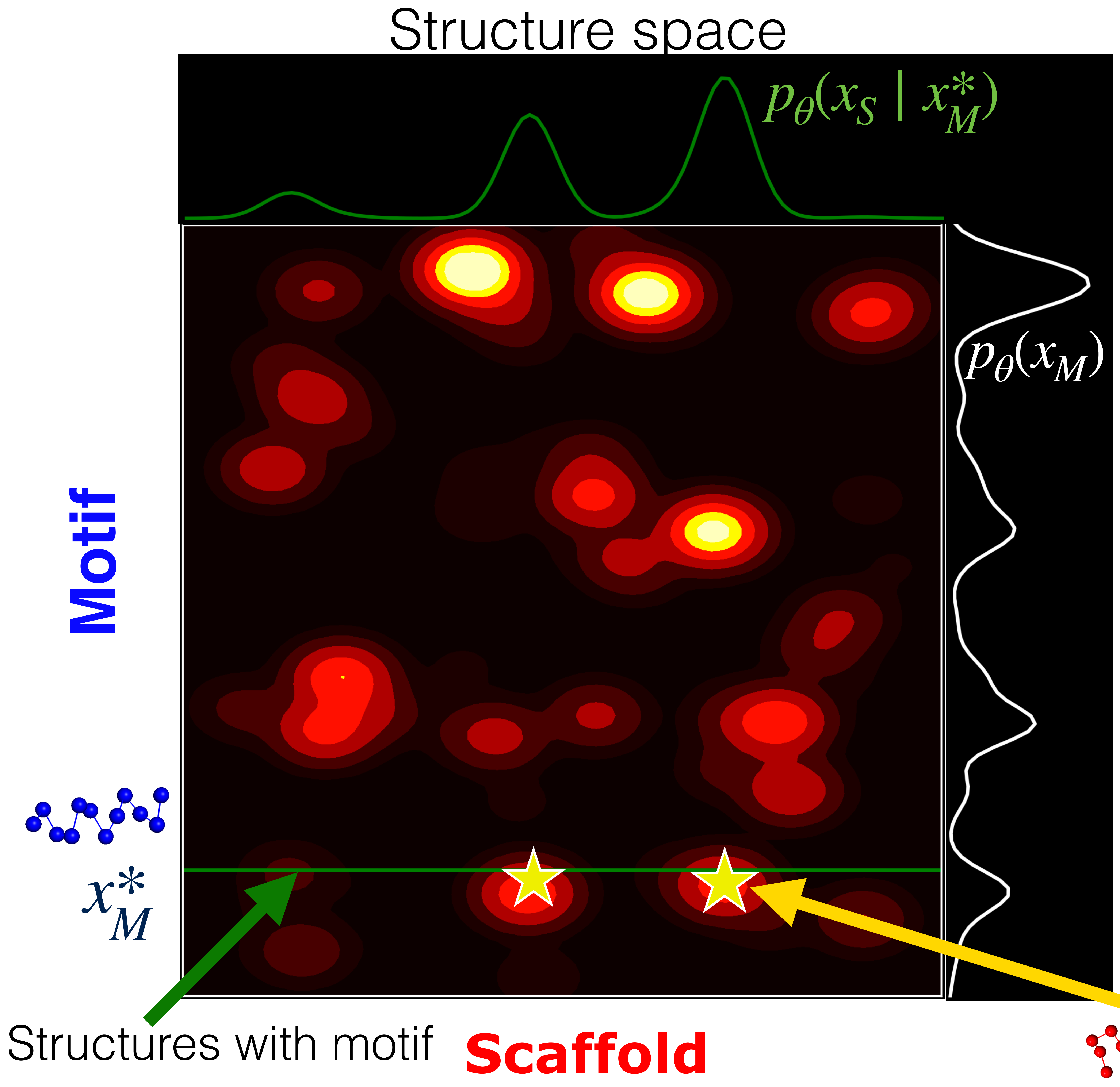
1. $p_\theta(x) > 0$ only if x is designable, and
2. $p_\theta(x_M^*) > 0$, for motif of interest x_M^*.

Then for any $x_S \sim p_\theta(x_S \mid x_M^*)$, $x = [x_M^*, x_S]$ is designable!

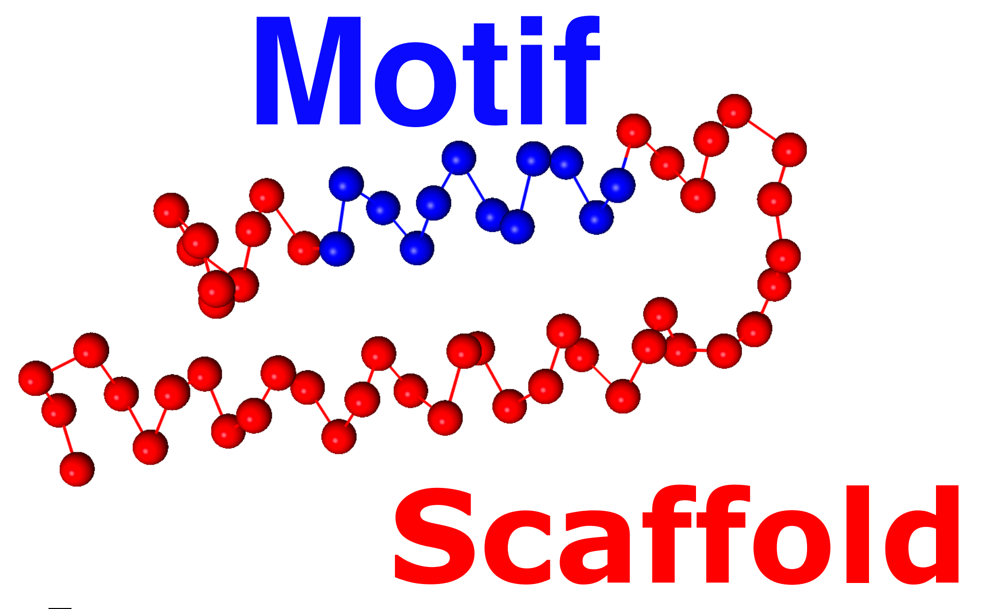
Result: With a good enough $p_\theta(x)$, motif-scaffolding $\leftrightarrow$ conditional sampling



Conditional Sampling to Fix Inpainting?



- Partition structure as $x = [x_M, x_S]$



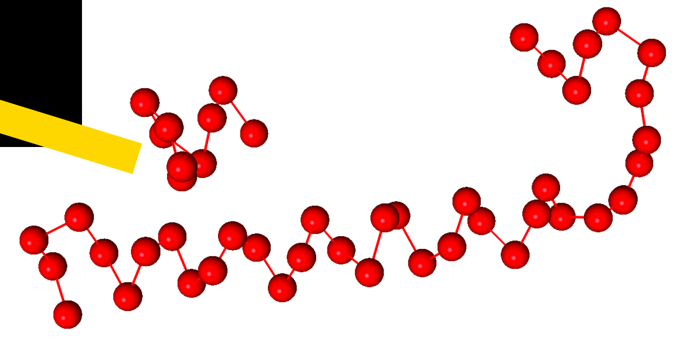
Our rationale: Assume

1. $p_\theta(x) > 0$ only if x is designable, and
2. $p_\theta(x_M^*) > 0$, for motif of interest x_M^*.

Then for any $x_S \sim p_\theta(x_S \mid x_M^*)$, $x = [x_M^*, x_S]$ is designable!

Result: With a good enough $p_\theta(x)$, motif-scaffolding $\leftrightarrow$ conditional sampling

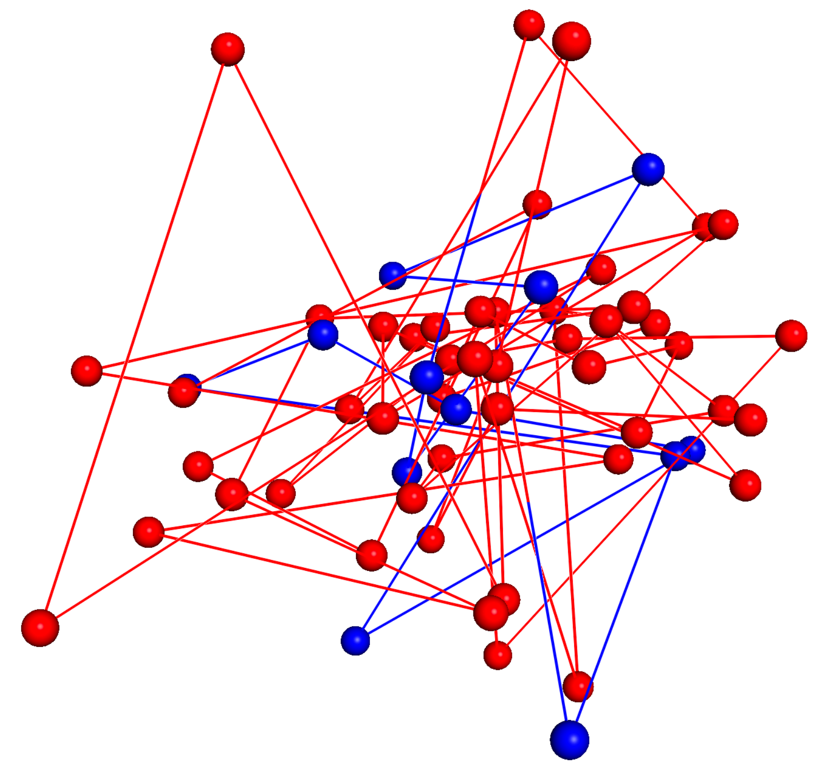
Challenge: How to sample $x_S \sim p_\theta(x_S \mid x_M^*)$



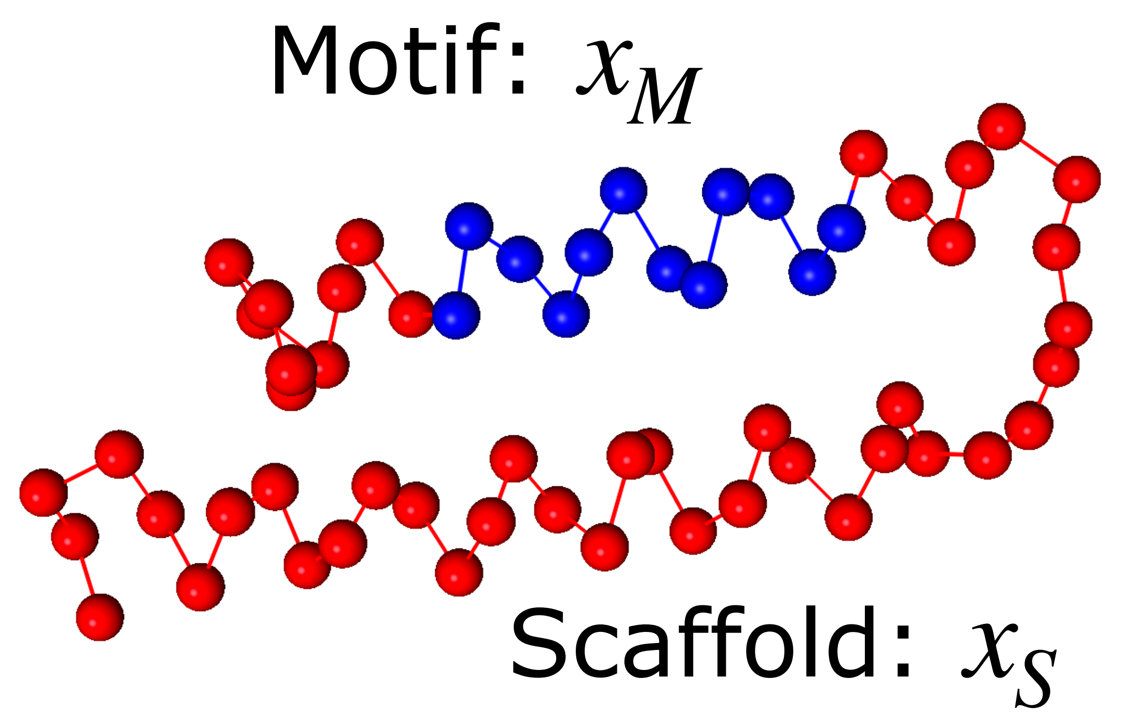
Inpainting with Diffusion Models

Unconditional Sampling:

$$x^{(t)} \sim p_\theta(x^{(t)} \,|\, x^{(t+1)})$$



$$= \quad \begin{matrix} x_M^{(T)} \\ x_S^{(T)} \end{matrix} \quad \cdots \quad \begin{matrix} x_M^{(t+1)} \\ x_S^{(t+1)} \end{matrix} \xrightarrow{} \begin{matrix} x_M^{(t)} \\ x_S^{(t)} \end{matrix} \quad \cdots \quad \begin{matrix} x_M^{(0)} \\ x_S^{(0)} \end{matrix} \quad =$$



Inpainting with Diffusion Models

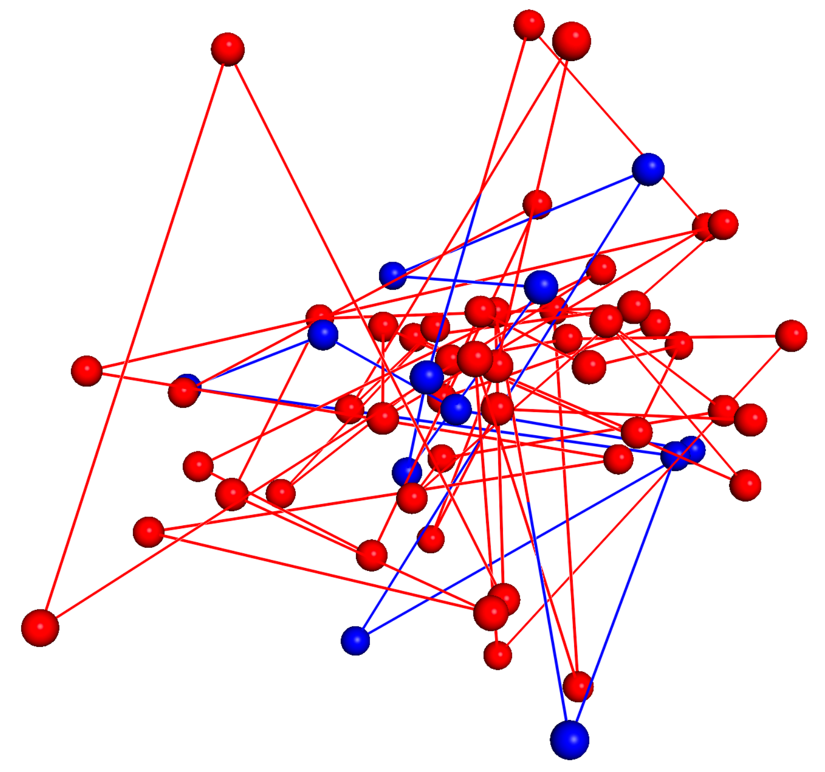
Unconditional Sampling:

$$x^{(t)} \sim p_\theta(x^{(t)} \mid x^{(t+1)})$$

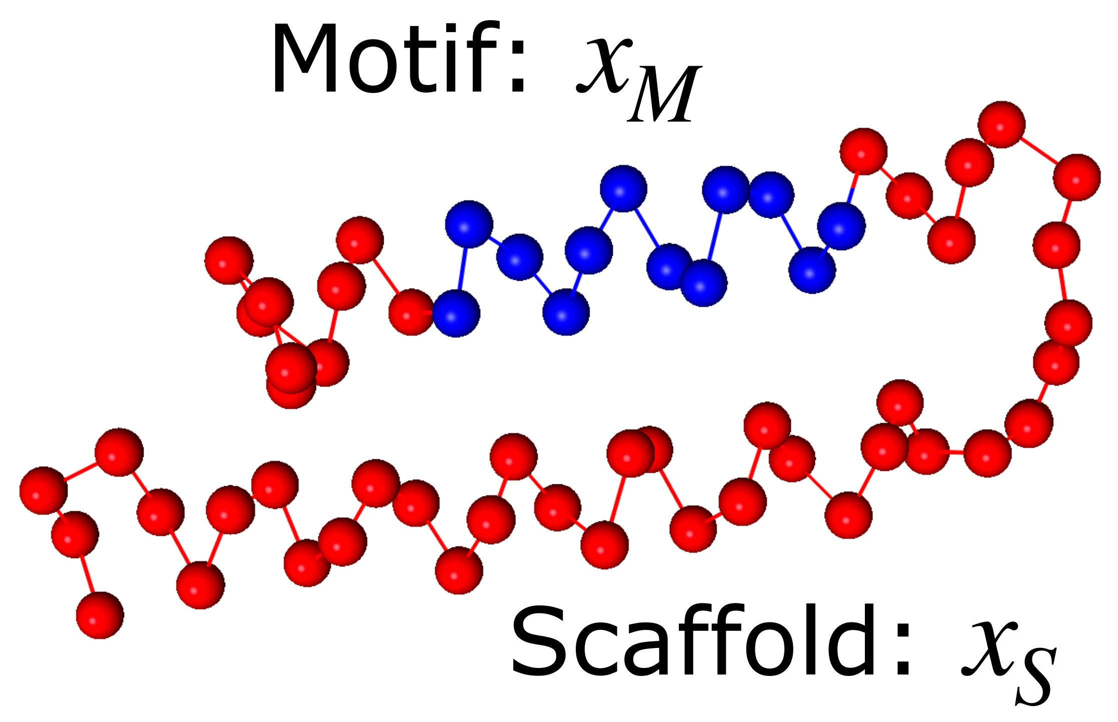
Conditional Sampling:

$$x_S^{(t)} \sim p_\theta(x_S^{(t)} \mid x_S^{(t+1)}, x_M^{(0)})$$

- Exact but intractable!



$$\begin{tikzcd} = & \begin{matrix} x_M^{(T)} \\ x_S^{(T)} \end{matrix} & \cdots & \begin{matrix} x_M^{(t+1)} \\ x_S^{(t+1)} \end{matrix} \arrow[r, bend right] & \begin{matrix} x_M^{(t)} \\ x_S^{(t)} \end{matrix} & \cdots & \begin{matrix} x_M^{(0)} \\ x_S^{(0)} \end{matrix} & = \end{tikzcd}$$



Inpainting with Diffusion Models

Unconditional Sampling:

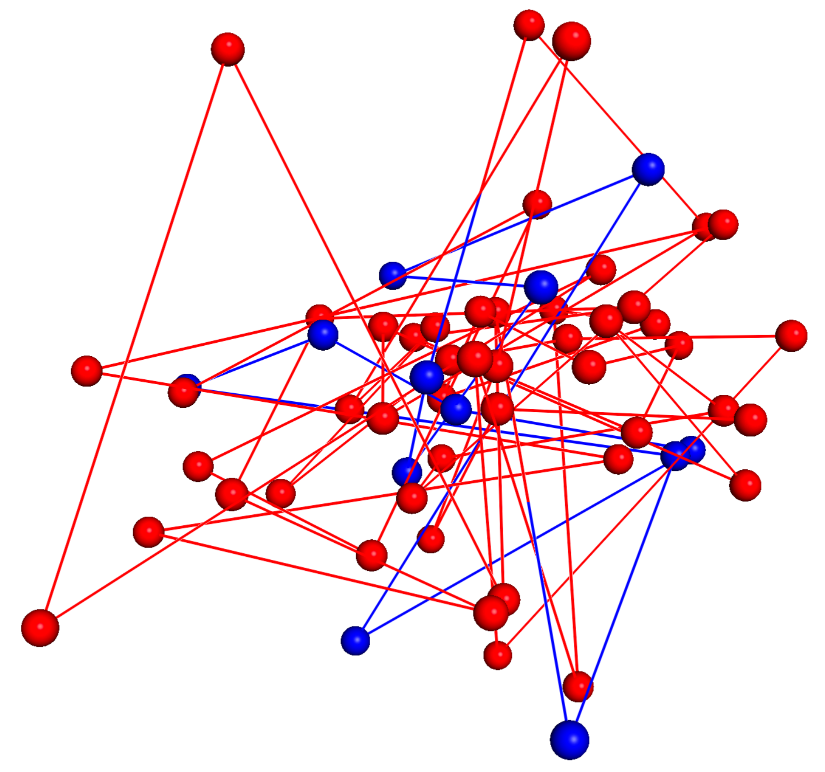
$$x^{(t)} \sim p_\theta(x^{(t)} \mid x^{(t+1)})$$

Conditional Sampling:

$$x_S^{(t)} \sim p_\theta(x_S^{(t)} \mid x_S^{(t+1)}, x_M^{(0)})$$

- Exact but intractable!

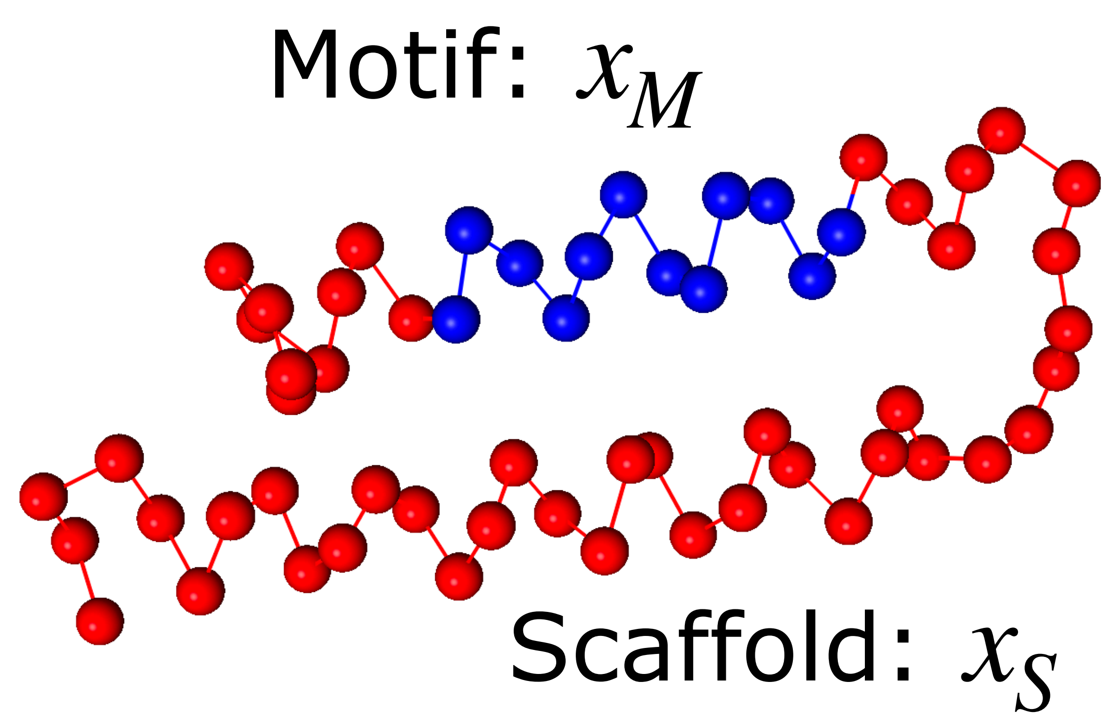
- Tractable alternative: Replacement approach [Song 2021, Meng 2021]



forward motif diffusion

$$= \begin{array}{c} x_M^{(T)} \\ x_S^{(T)} \end{array} \quad \cdots \quad \begin{array}{c} x_M^{(t+1)} \\ x_S^{(t+1)} \end{array} \quad \begin{array}{c} x_M^{(t)} \\ x_S^{(t)} \end{array} \quad \cdots \quad \begin{array}{c} x_M^{(0)} \\ x_S^{(0)} \end{array} =$$

reverse replacement sampling



$$x_S^{(t)} \sim p_\theta(x_S^{(t)} \mid x_S^{(t+1)}, x_M^{(t+1)}) \text{ with } x_M^{(t+1)} \sim q(x_M^{(t+1)} \mid x_M^{0})$$

Inpainting with Diffusion Models

Unconditional Sampling:

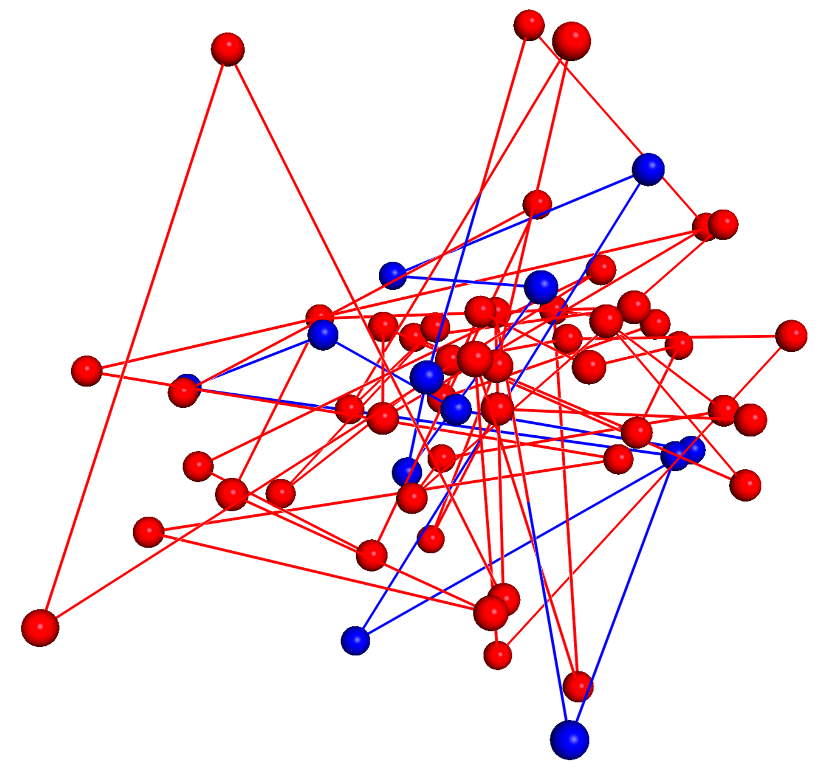
$$x^{(t)} \sim p_\theta(x^{(t)} \mid x^{(t+1)})$$

Conditional Sampling:

$$x_S^{(t)} \sim p_\theta(x_S^{(t)} \mid x_S^{(t+1)}, x_M^{(0)})$$

- Exact but intractable!

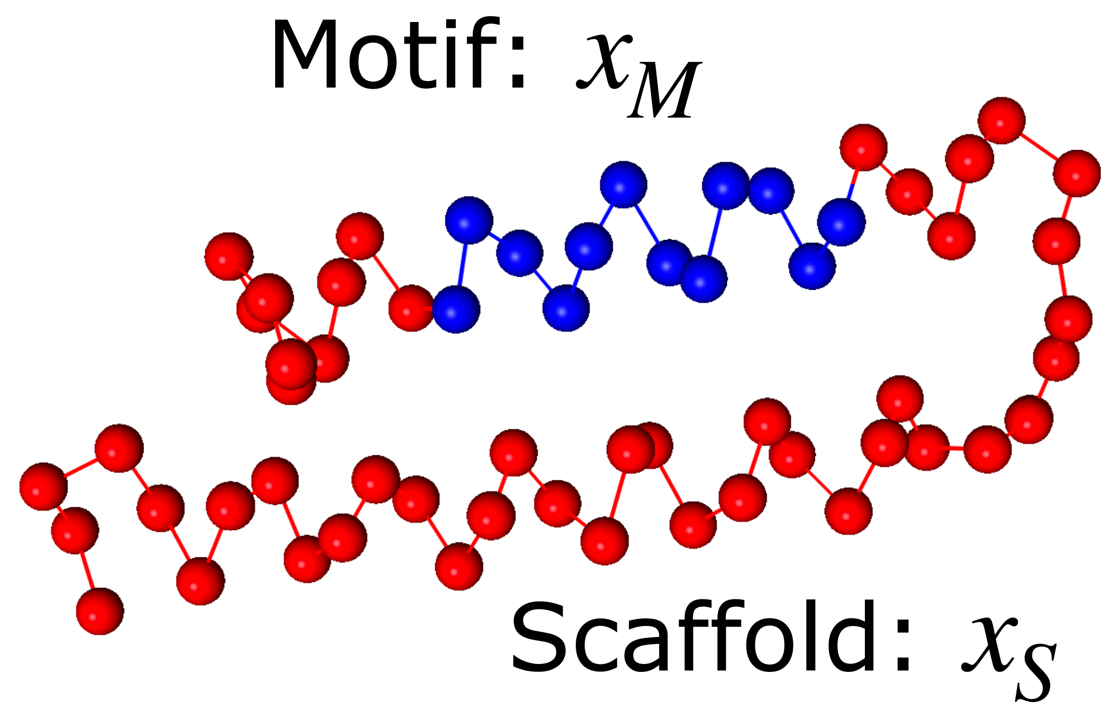
- Tractable alternative: Replacement approach [Song 2021, Meng 2021]



forward motif diffusion

$$= \begin{array}{c} x_M^{(T)} \\ x_S^{(T)} \end{array} \quad \cdots \quad \begin{array}{c} x_M^{(t+1)} \\ x_S^{(t+1)} \end{array} \quad \begin{array}{c} x_M^{(t)} \\ x_S^{(t)} \end{array} \quad \cdots \quad \begin{array}{c} x_M^{(0)} \\ x_S^{(0)} \end{array} =$$

reverse replacement sampling



$$x_S^{(t)} \sim p_\theta(x_S^{(t)} \mid x_S^{(t+1)}, x_M^{(t+1)}) \text{ with } x_M^{(t+1)} \sim q(x_M^{(t+1)} \mid x_M^0)$$

- Introduces approximation error and leads to chain breaks!

Improved inpainting with SMCDiff

Replacement Method:

$x_S^{(t)} \sim p_\theta(x_S^{(t)} \mid x_S^{(t+1)}, x_M^{(t+1)})$,

with $x_M^{(t+1)} \sim q(x_M^{(t+1)} \mid x_M^{(0)})$

- Inexact but tractable

Conditional Sampling:

$x_S^{(t)} \sim p_\theta(x_S^{(t)} \mid x_S^{(t+1)}, x_M^{(0)})$

- Exact but intractable!

Improved inpainting with SMCDiff

Replacement Method:

$x_S^{(t)} \sim p_\theta(x_S^{(t)} \mid x_S^{(t+1)}, x_M^{(t+1)})$,

with $x_M^{(t+1)} \sim q(x_M^{(t+1)} \mid x_M^{(0)})$

- Inexact but tractable

Is there something in-between?

Conditional Sampling:

$x_S^{(t)} \sim p_\theta(x_S^{(t)} \mid x_S^{(t+1)}, x_M^{(0)})$

- Exact but intractable!

Improved inpainting with SMCDiff

Replacement Method:

$$x_S^{(t)} \sim p_\theta(x_S^{(t)} \mid x_S^{(t+1)}, x_M^{(t+1)}),$$

with $x_M^{(t+1)} \sim q(x_M^{(t+1)} \mid x_M^{(0)})$

- Inexact but tractable

Is there something in-between?

Conditional Sampling:

$$x_S^{(t)} \sim p_\theta(x_S^{(t)} \mid x_S^{(t+1)}, x_M^{(0)})$$

- Exact but intractable!

Our proposal — Look ahead by a few steps:

$$x_S^{(t)} \sim p_\theta(x_S^{(t)} \mid x_S^{(t+1)}, x_M^{(t-1:t+1)}), \text{ with } x_M^{(t-1:t+1)} \sim q(x_M^{(t-1:t+1)} \mid x_M^{(0)})$$

Improved inpainting with SMCDiff

Replacement Method:

$$x_S^{(t)} \sim p_\theta(x_S^{(t)} \mid x_S^{(t+1)}, x_M^{(t+1)}),$$

with $x_M^{(t+1)} \sim q(x_M^{(t+1)} \mid x_M^{(0)})$

- Inexact but tractable

Is there something in-between?

Conditional Sampling:

$$x_S^{(t)} \sim p_\theta(x_S^{(t)} \mid x_S^{(t+1)}, x_M^{(0)})$$

- Exact but intractable!

Our proposal — Look ahead by a few steps:

$$x_S^{(t)} \sim p_\theta(x_S^{(t)} \mid x_S^{(t+1)}, x_M^{(t-1:t+1)}),\ \text{with } x_M^{(t-1:t+1)} \sim q(x_M^{(t-1:t+1)} \mid x_M^{(0)})$$

- Tractable with a sequential Monte Carlo (SMC) algorithm

Improved inpainting with SMCDiff

Replacement Method:

$x_S^{(t)} \sim p_\theta(x_S^{(t)} \mid x_S^{(t+1)}, x_M^{(t+1)}),$
with $x_M^{(t+1)} \sim q(x_M^{(t+1)} \mid x_M^{(0)})$

- Inexact but tractable

Is there something in-between?

Conditional Sampling:

$x_S^{(t)} \sim p_\theta(x_S^{(t)} \mid x_S^{(t+1)}, x_M^{(0)})$

- Exact but intractable!

Our proposal — Look ahead by a few steps:

$$x_S^{(t)} \sim p_\theta(x_S^{(t)} \mid x_S^{(t+1)}, x_M^{(t-1:t+1)}), \text{ with } x_M^{(t-1:t+1)} \sim q(x_M^{(t-1:t+1)} \mid x_M^{(0)})$$

- Tractable with a sequential Monte Carlo (SMC) algorithm

$$p_\theta(x_S^{(t)} \mid x_S^{(t+1)}, x_M^{(t-1:t+1)}) \propto p_\theta(x_S^{(t)}, x_M^{(t-1:t)} \mid x_S^{(t+1)}, x_M^{(t+1)}) \qquad \text{(Bayes' rule)}$$

Improved inpainting with SMCDiff

Replacement Method:

$$x_S^{(t)} \sim p_\theta(x_S^{(t)} \mid x_S^{(t+1)}, x_M^{(t+1)}),$$

with $x_M^{(t+1)} \sim q(x_M^{(t+1)} \mid x_M^{(0)})$

- Inexact but tractable

Is there something in-between?

Conditional Sampling:

$$x_S^{(t)} \sim p_\theta(x_S^{(t)} \mid x_S^{(t+1)}, x_M^{(0)})$$

- Exact but intractable!

Our proposal — Look ahead by a few steps:

$$x_S^{(t)} \sim p_\theta(x_S^{(t)} \mid x_S^{(t+1)}, x_M^{(t-1:t+1)}), \text{ with } x_M^{(t-1:t+1)} \sim q(x_M^{(t-1:t+1)} \mid x_M^{(0)})$$

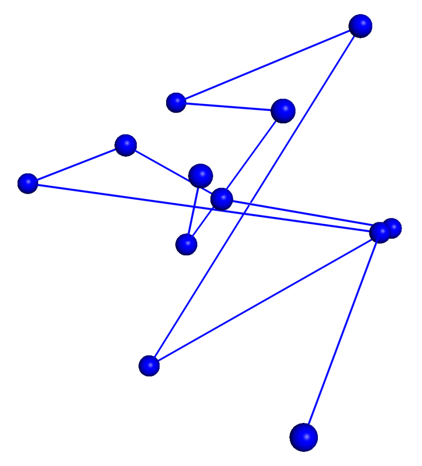
- Tractable with a sequential Monte Carlo (SMC) algorithm

$$p_\theta(x_S^{(t)} \mid x_S^{(t+1)}, x_M^{(t-1:t+1)}) \propto p_\theta(x_S^{(t)}, x_M^{(t-1:t)} \mid x_S^{(t+1)}, x_M^{(t+1)}) \qquad \text{(Bayes' rule)}$$

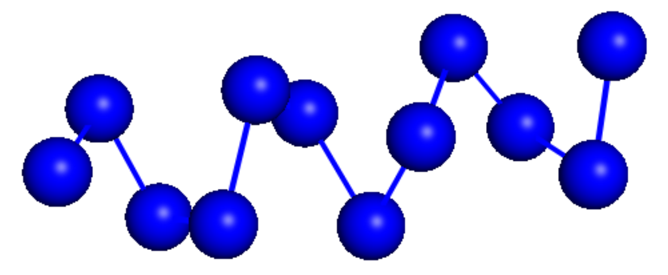
$$\propto \underbrace{p_\theta(x_S^{(t)} \mid x_S^{(t-1)}, x_M^{(t-1)})}_{\text{can sample}} \underbrace{p_\theta(x_M^{(t-1)} \mid x_S^{(t)}, x_M^{(t)})}_{\text{can compute}} \qquad \text{(Markov structure)}$$

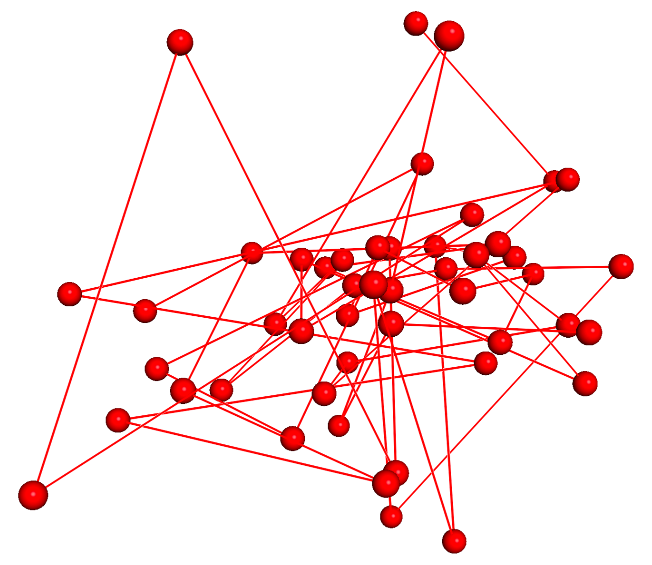
Improved inpainting with SMCDiff

$$\longleftarrow$$

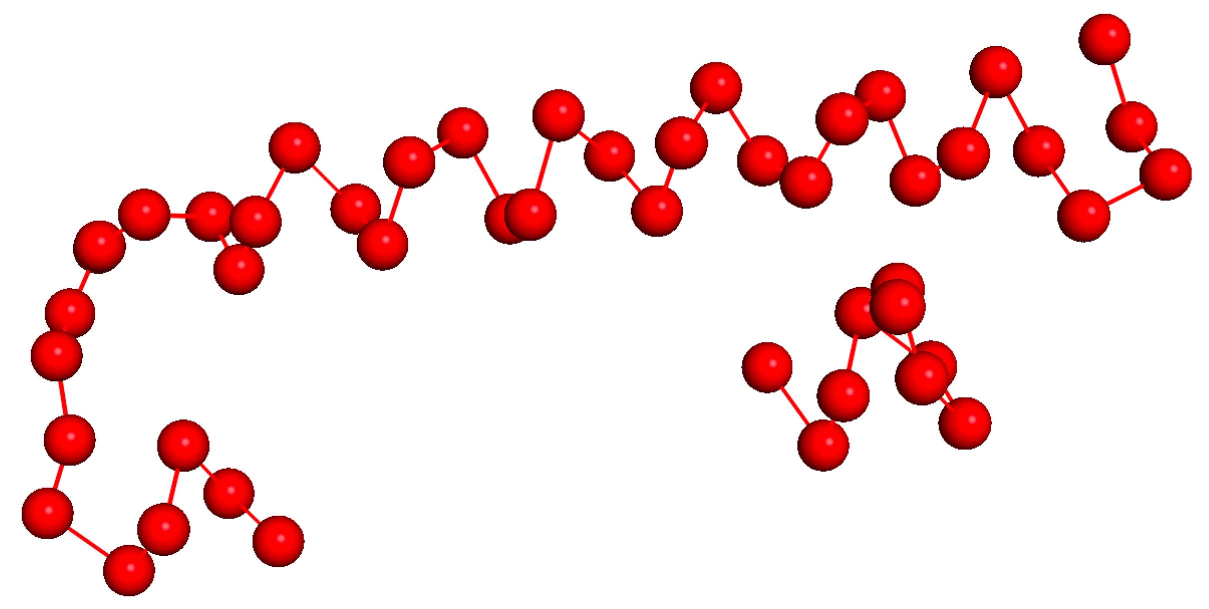


$$= \quad x_M^{(T)} \quad \cdots \quad x_M^{(t+1)} \quad x_M^{(t)} \quad x_M^{(t-1)} \quad \cdots \quad x_M^{(0)} \quad =$$





$$= \quad x_{S,k}^{(T)} \quad \cdots \quad x_{S,k}^{(t+1)} \quad x_{S,k}^{(t)} \quad x_{S,k}^{(t-1)} \quad \cdots \quad x_{S,k}^{(0)} \quad =$$

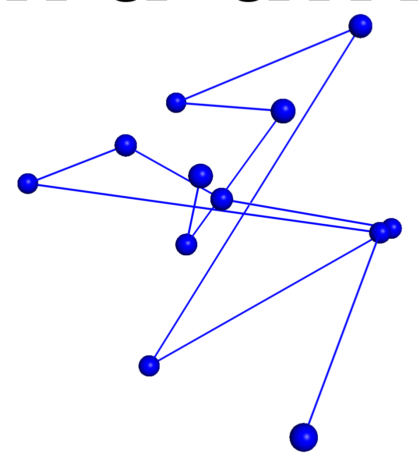


$$\longrightarrow$$

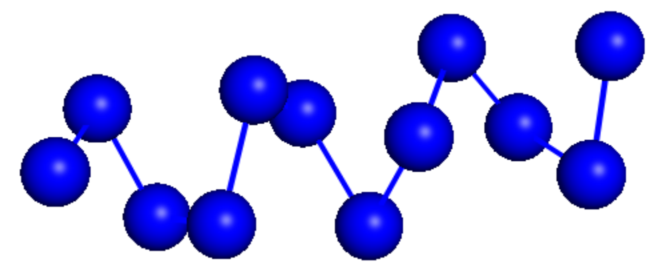
Improved inpainting with SMCDiff

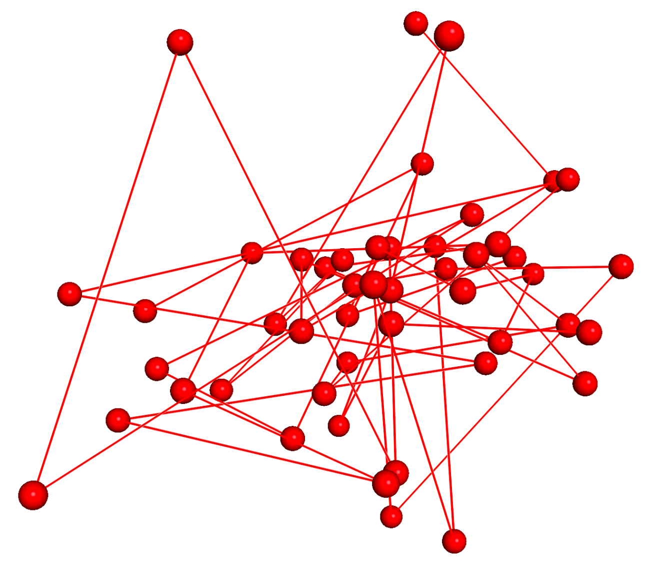
- **Forward diffuse motif**:

$$\longleftarrow$$

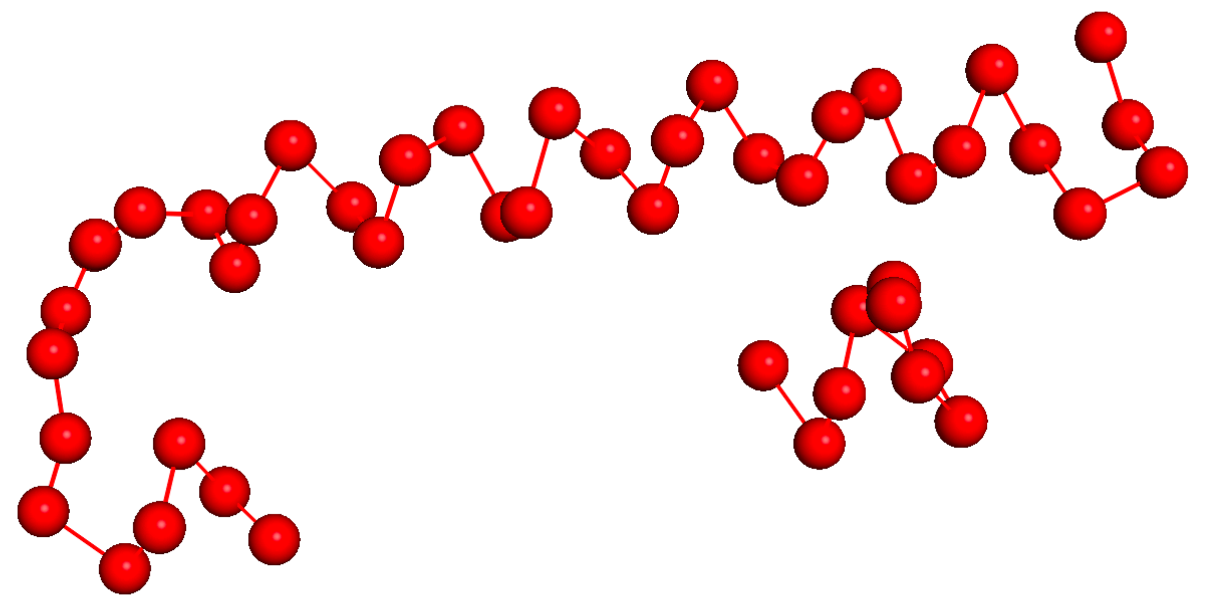


$$= \quad x_M^{(T)} \quad \cdots \quad x_M^{(t+1)} \quad x_M^{(t)} \quad x_M^{(t-1)} \quad \cdots \quad x_M^{(0)} \quad =$$





$$= \quad x_{S,k}^{(T)} \quad \cdots \quad x_{S,k}^{(t+1)} \quad x_{S,k}^{(t)} \quad x_{S,k}^{(t-1)} \quad \cdots \quad x_{S,k}^{(0)} \quad =$$

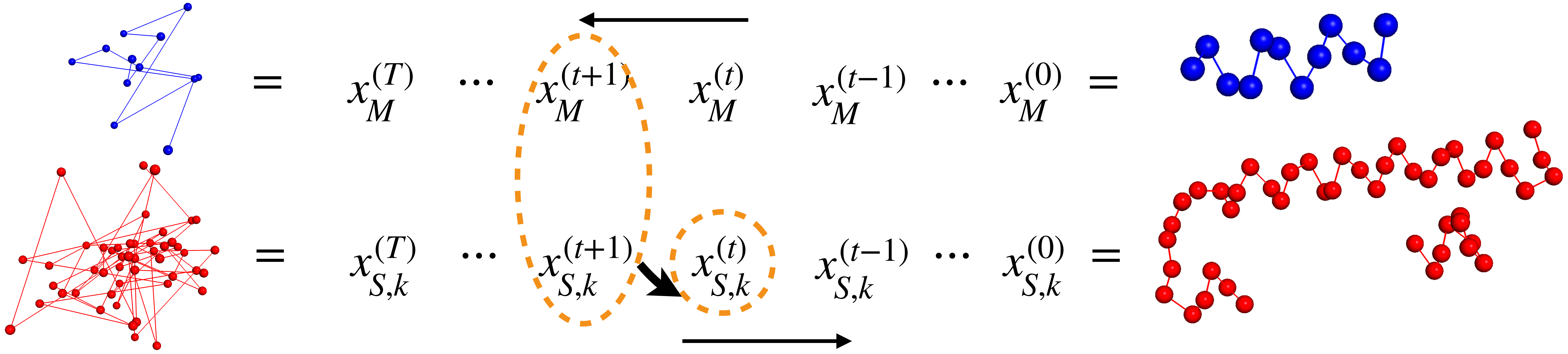


$$\longrightarrow$$

- **Reverse diffuse K weighted scaffold “particles”**

Improved inpainting with SMCDiff

- **Forward diffuse motif**:



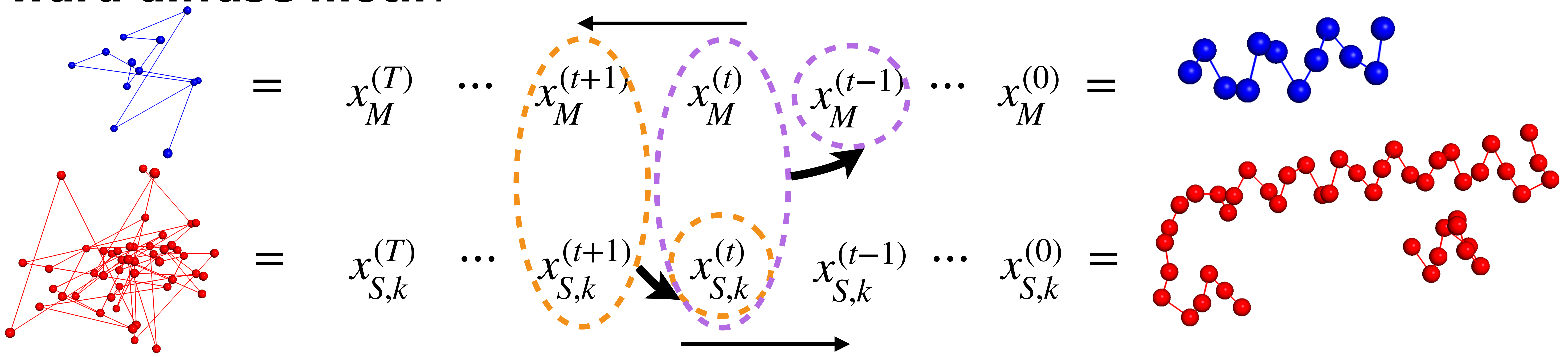
$= x_M^{(T)} \cdots x_M^{(t+1)} \; x_M^{(t)} \; x_M^{(t-1)} \cdots x_M^{(0)} =$
$= x_{S,k}^{(T)} \cdots x_{S,k}^{(t+1)} \; x_{S,k}^{(t)} \; x_{S,k}^{(t-1)} \cdots x_{S,k}^{(0)} =$

- **Reverse diffuse K weighted scaffold "particles"**

1. **Propose** scaffolds: $x_{S,k}^{(t)} \sim p_\theta(x_S^{(t)} \,|\, x_{S,k}^{(t+1)}, x_M^{(t+1)})$ for $k = 1, \ldots, K$

Improved inpainting with SMCDiff

- **Forward diffuse motif**:



$= x_M^{(T)} \cdots x_M^{(t+1)} \; x_M^{(t)} \; x_M^{(t-1)} \cdots x_M^{(0)} =$
$= x_{S,k}^{(T)} \cdots x_{S,k}^{(t+1)} \; x_{S,k}^{(t)} \; x_{S,k}^{(t-1)} \cdots x_{S,k}^{(0)} =$

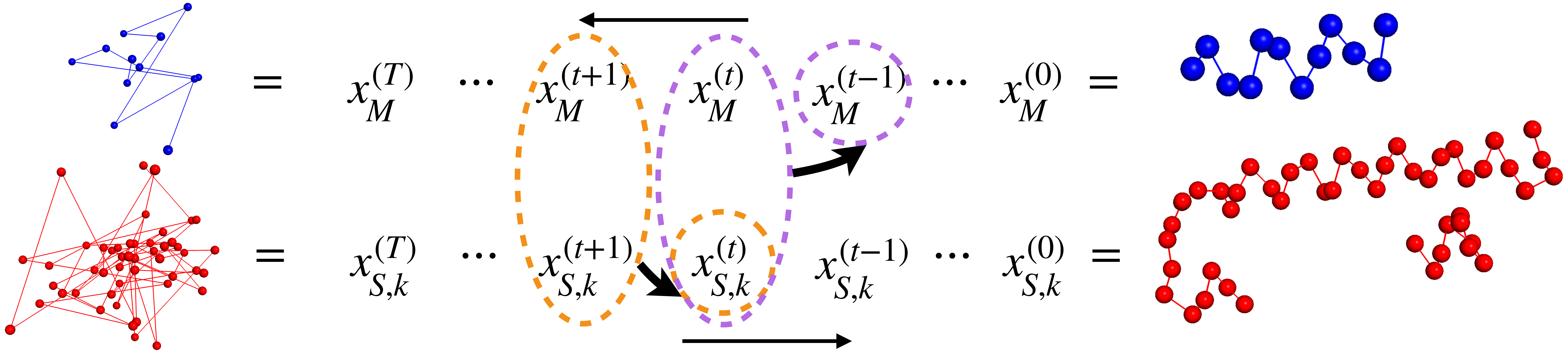
- **Reverse diffuse K weighted scaffold "particles"**

1. **Propose** scaffolds: $x_{S,k}^{(t)} \sim p_\theta(x_S^{(t)} \,|\, x_{S,k}^{(t+1)}, x_M^{(t+1)})$ for $k = 1, \ldots, K$

2. **Re-weight** by looking ahead: $w_k = p_\theta(x_M^{(t-1)} \,|\, x_M^{(t)}, x_{S,k}^{(t)}), \quad \tilde{w}_k = w_k / \sum_{k'=1}^{K} w_k$

Improved inpainting with SMCDiff

- **Forward diffuse motif**:



$= x_M^{(T)} \cdots x_M^{(t+1)} \; x_M^{(t)} \; x_M^{(t-1)} \cdots x_M^{(0)} =$
$= x_{S,k}^{(T)} \cdots x_{S,k}^{(t+1)} \; x_{S,k}^{(t)} \; x_{S,k}^{(t-1)} \cdots x_{S,k}^{(0)} =$

- **Reverse diffuse K weighted scaffold "particles"**

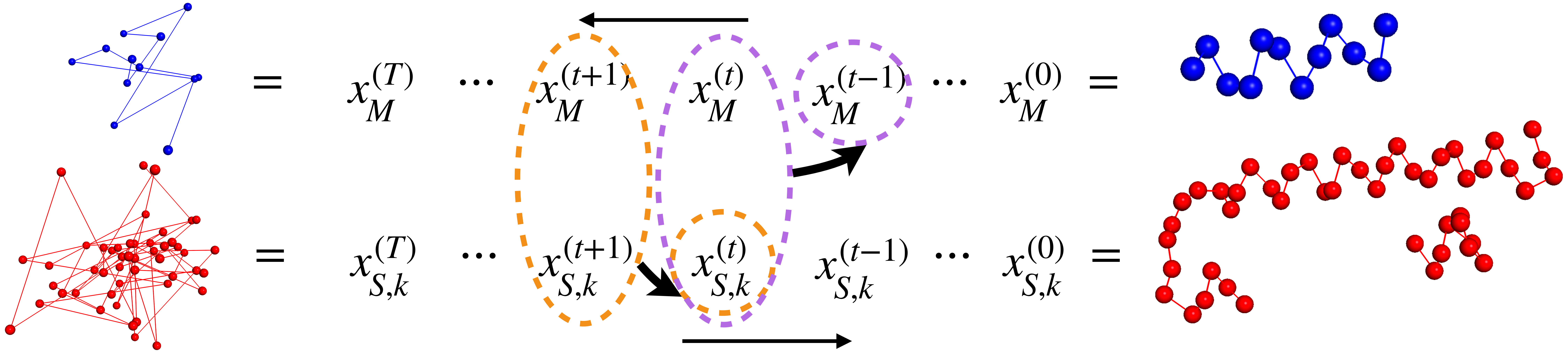
1. **Propose** scaffolds: $x_{S,k}^{(t)} \sim p_\theta(x_S^{(t)} \mid x_{S,k}^{(t+1)}, x_M^{(t+1)})$ for $k = 1, \ldots, K$

2. **Re-weight** by looking ahead: $w_k = p_\theta(x_M^{(t-1)} \mid x_M^{(t)}, x_{S,k}^{(t)}), \quad \tilde{w}_k = w_k / \sum_{k'=1}^{K} w_k$

3. **Resample** according to weight: $\{x_{S,k}^{(t)}\}_{k=1}^{K} \sim$ Multinomial$(\{x_{S,k'}^{(t)}\}, \{\tilde{w}_{k'}\}, K)$

Improved inpainting with SMCDiff

- **Forward diffuse motif**:



$= \quad x_M^{(T)} \quad \cdots \quad x_M^{(t+1)} \quad x_M^{(t)} \quad x_M^{(t-1)} \quad \cdots \quad x_M^{(0)} \quad =$
$= \quad x_{S,k}^{(T)} \quad \cdots \quad x_{S,k}^{(t+1)} \quad x_{S,k}^{(t)} \quad x_{S,k}^{(t-1)} \quad \cdots \quad x_{S,k}^{(0)} \quad =$

- **Reverse diffuse K weighted scaffold "particles"**

1. **Propose** scaffolds: $x_{S,k}^{(t)} \sim p_\theta(x_S^{(t)} \mid x_{S,k}^{(t+1)}, x_M^{(t+1)})$ for $k = 1, \ldots, K$

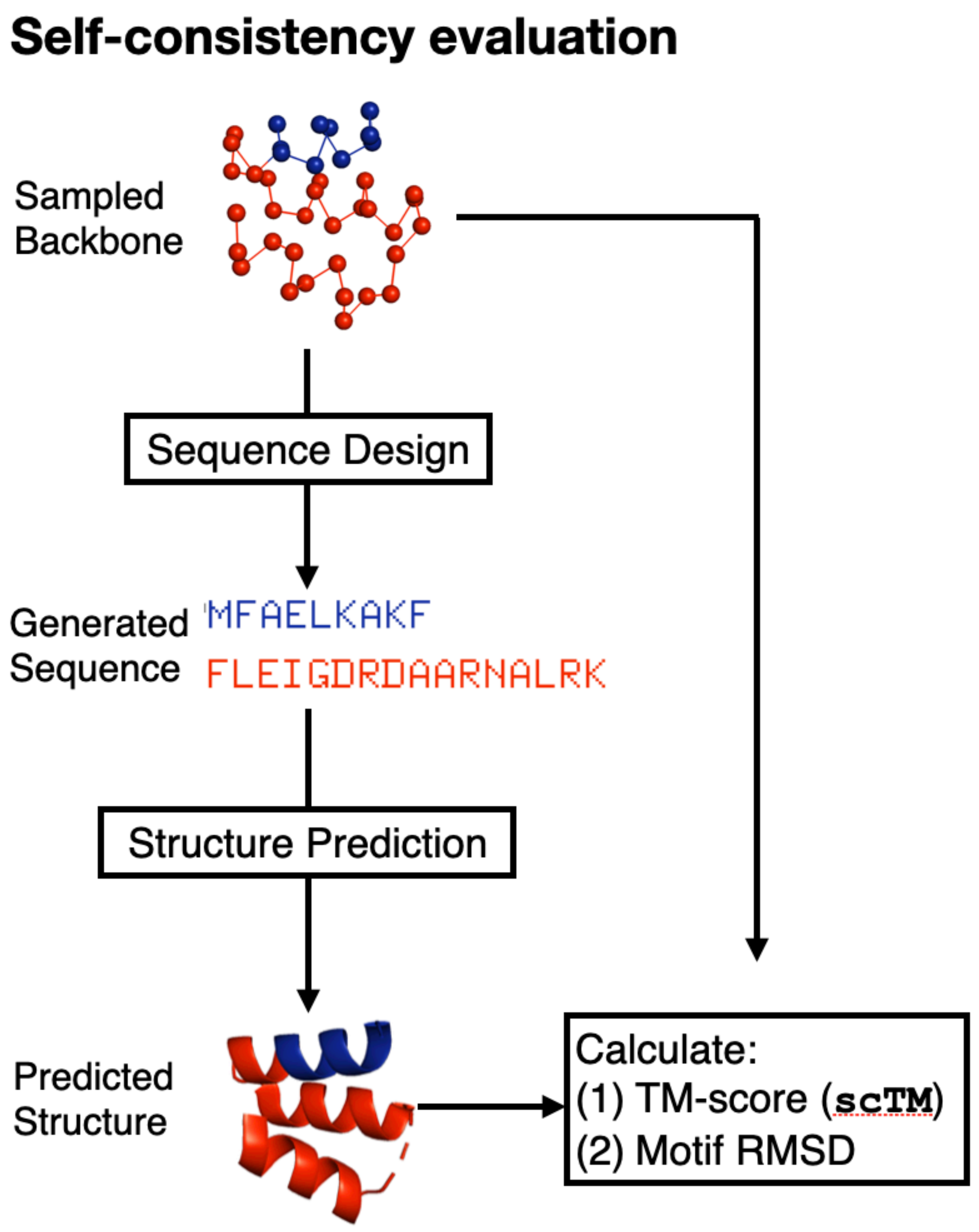
2. **Re-weight** by looking ahead: $w_k = p_\theta(x_M^{(t-1)} \mid x_M^{(t)}, x_{S,k}^{(t)}), \quad \tilde{w}_k = w_k / \sum_{k'=1}^{K} w_k$

3. **Resample** according to weight: $\{x_{S,k}^{(t)}\}_{k=1}^{K} \sim$ Multinomial$(\{x_{S,k'}^{(t)}\}, \{\tilde{w}_{k'}\}, K)$

Proposition (informal): *If* $p_\theta(x^{(0)}) = q(x^{(0)})$, *then* $x_{S,k}^{(0)} \xrightarrow[K\to\infty]{d} q(x_S^{(0)} \mid x_M^{(0)})$.

Motif self-consistency

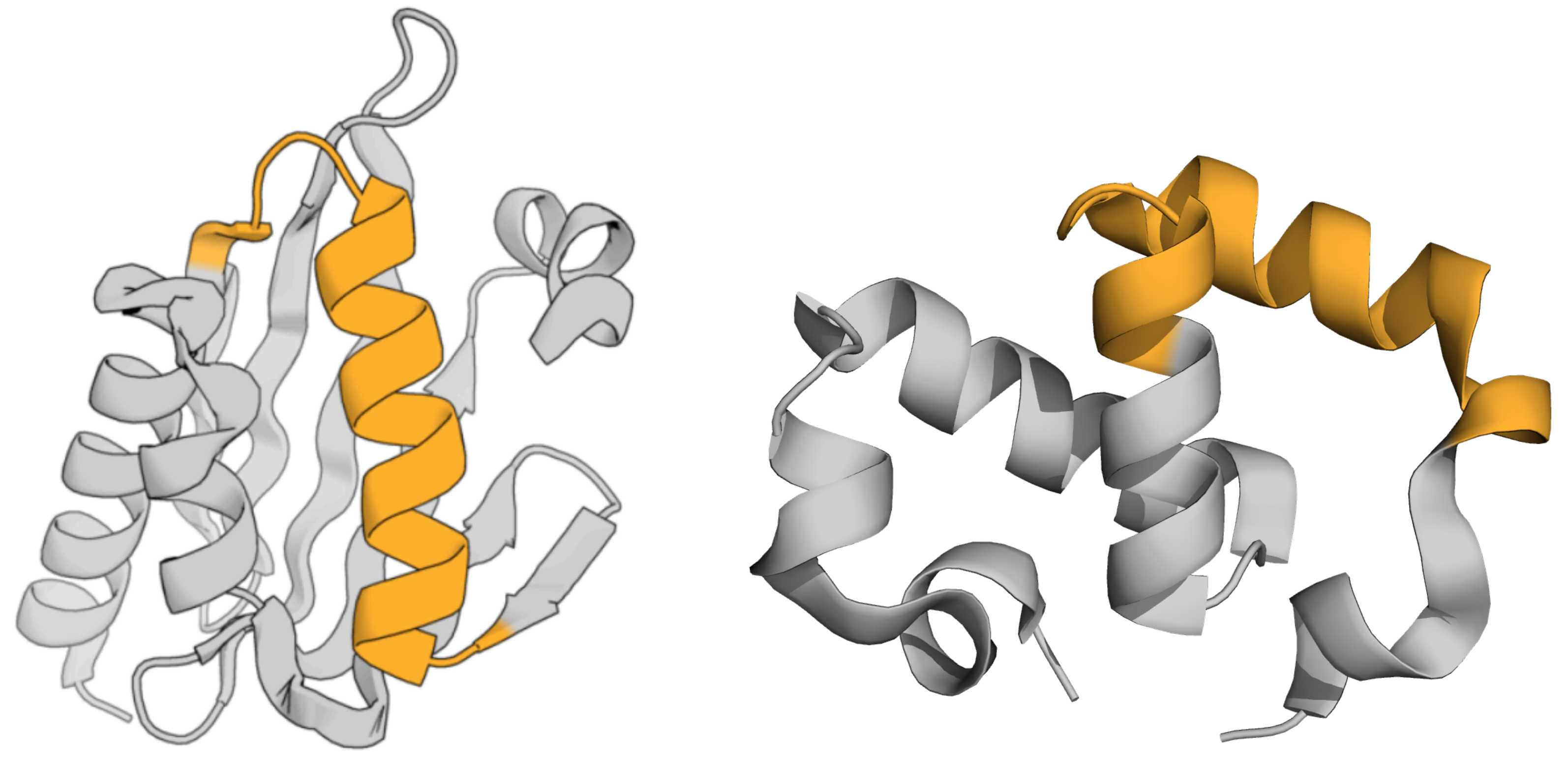
- Similar evaluation as before.
- Also calculate **Motif RMSD**
  - Achieving motif RMSD < 1.0A is imperative in motif-scaffolding.
- Criterion for successful scaffolding:
  - motif RMSD < 1.0A
  - **`scTM`** > 0.5



Motif-scaffolding case-studies

Evaluation Setup

- Pick motifs from structures in PDB
- Know at least one solution exists



	5trv	**6exz**
Base Motif Length	21 res.	20 res.
Length	118 res.	72 res.

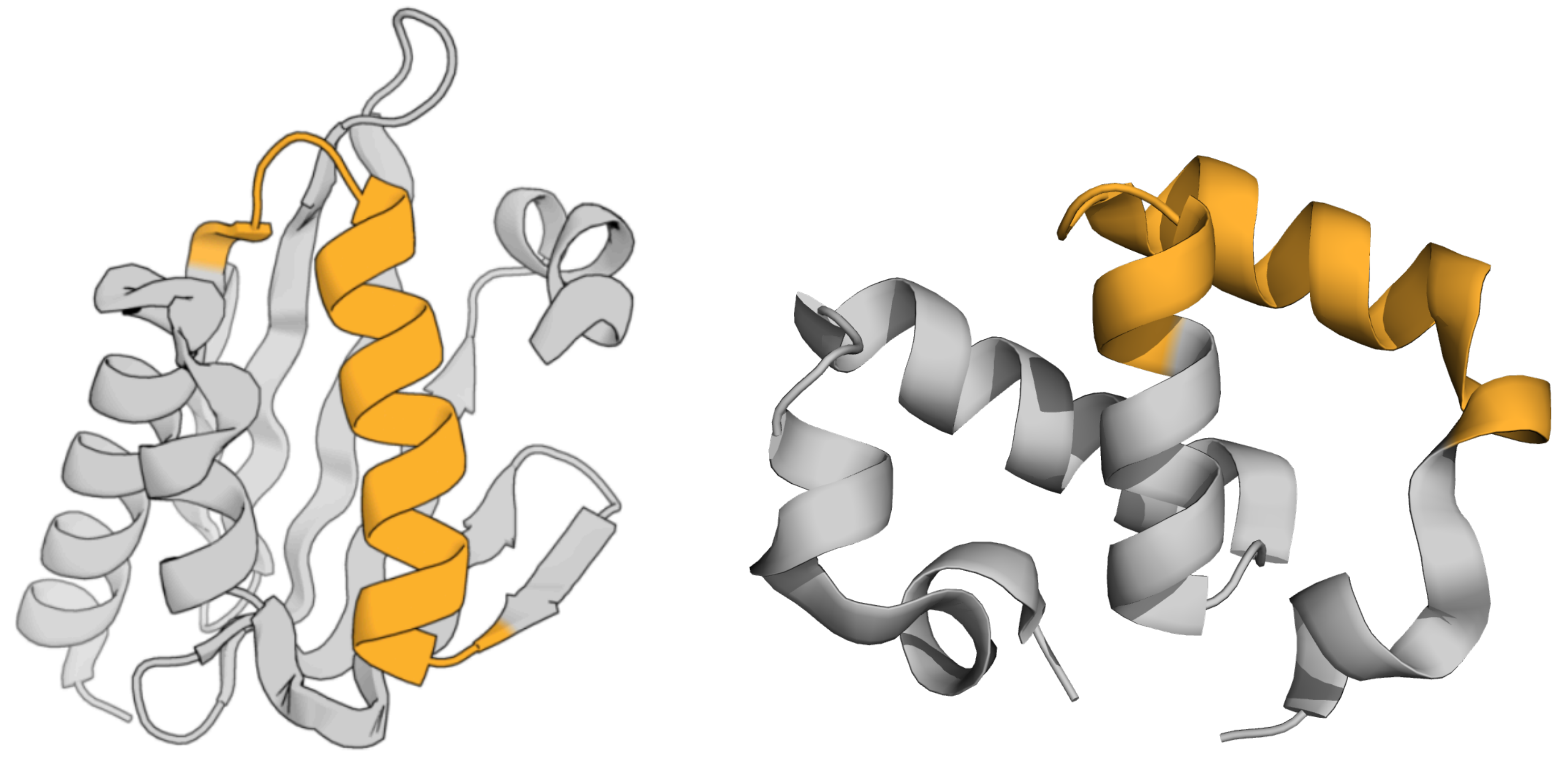
Motif-scaffolding case-studies

Evaluation Setup

- Pick motifs from structures in PDB
- Know at least one solution exists

We test dependence scaffold length

- Provide variable length “pads” of native scaffold



	5trv	6exz
Base Motif Length	21 res.	20 res.
Length	118 res.	72 res.

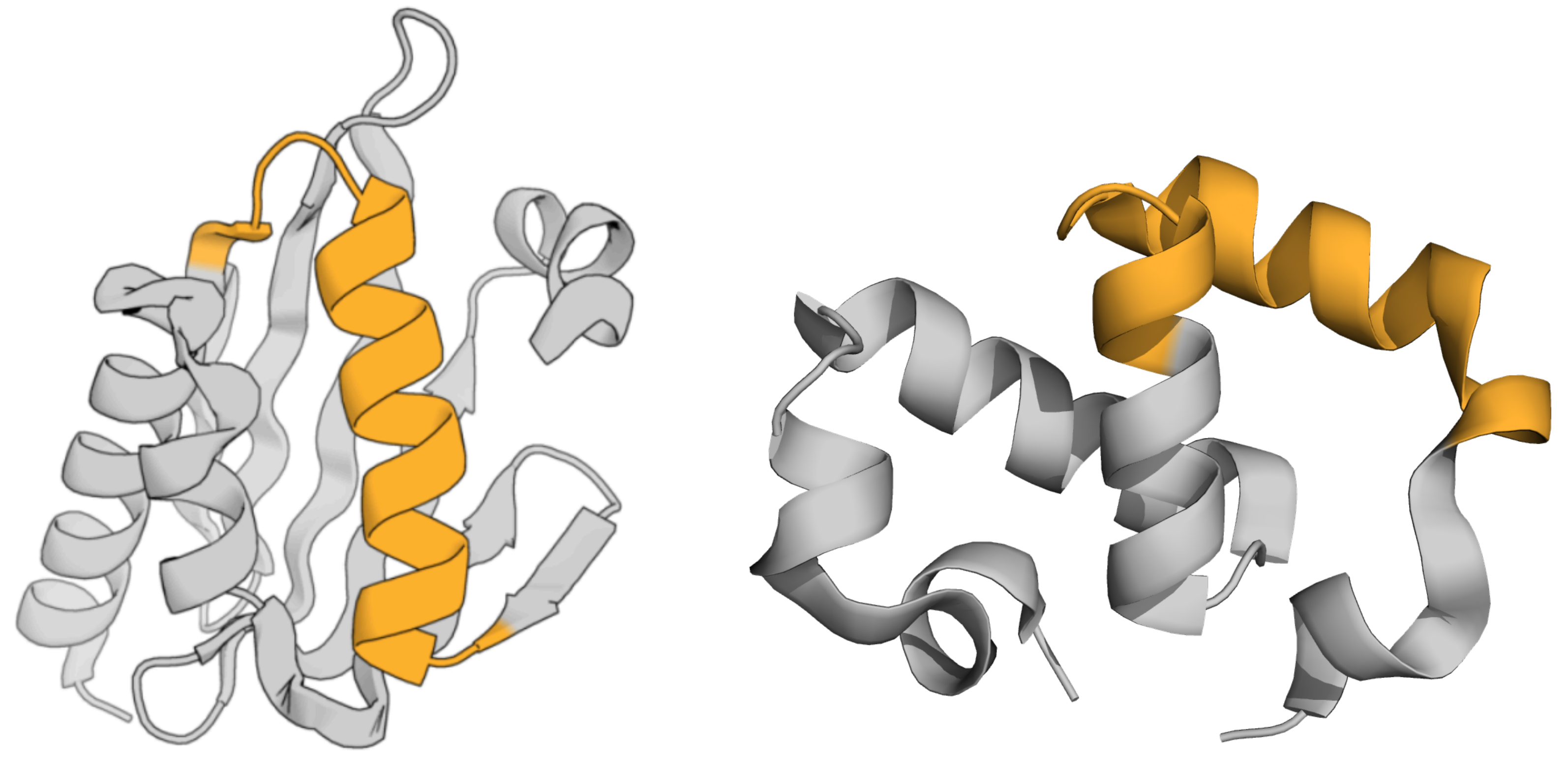
Motif-scaffolding case-studies

Evaluation Setup

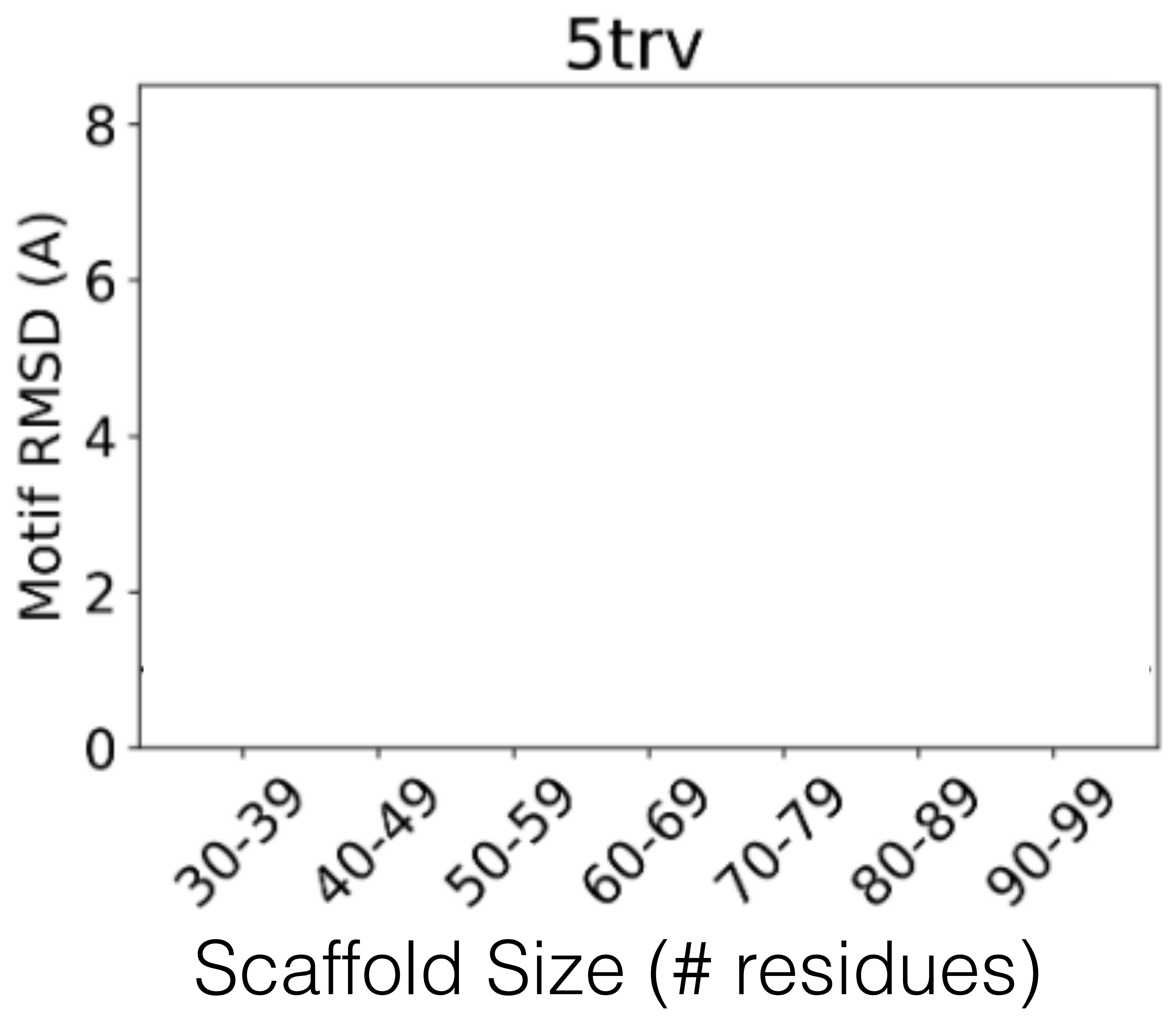
- Pick motifs from structures in PDB
- Know at least one solution exists

We test dependence scaffold length

- Provide variable length "pads" of native scaffold



	5trv	6exz
Base Motif Length	21 res.	20 res.
Length	118 res.	72 res.



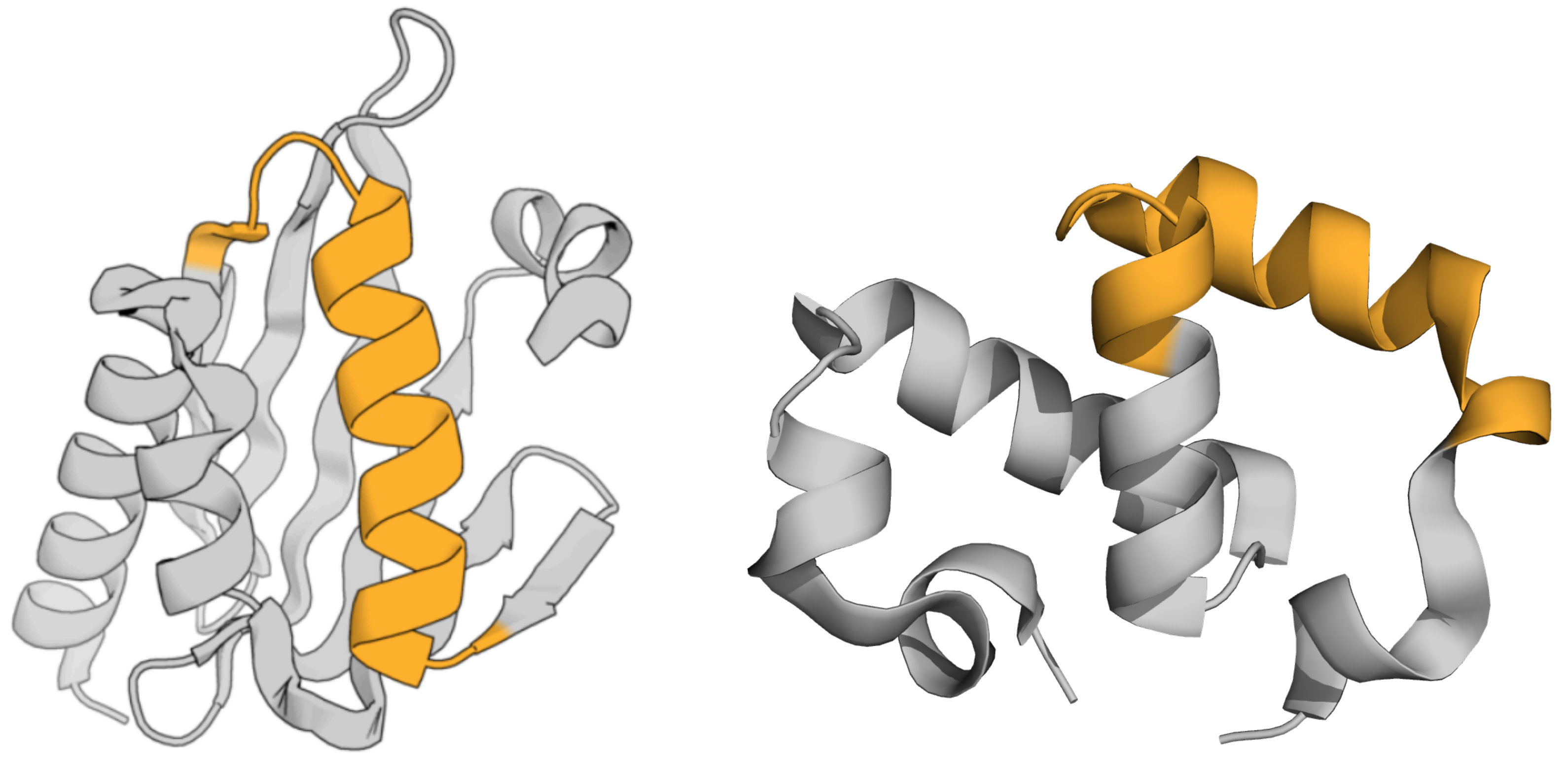
Motif-scaffolding case-studies

Evaluation Setup

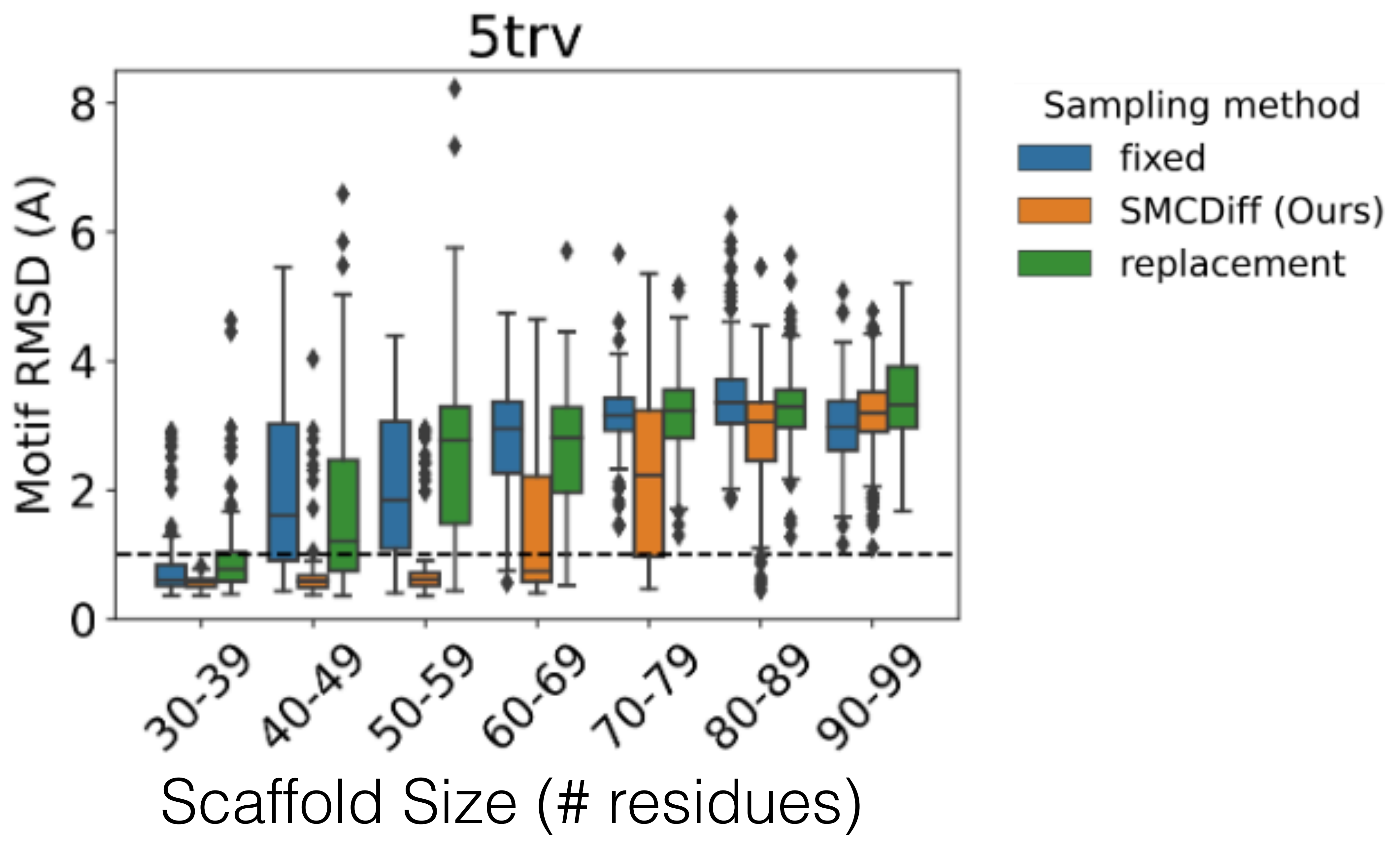
- Pick motifs from structures in PDB
- Know at least one solution exists

We test dependence scaffold length

- Provide variable length "pads" of native scaffold



	5trv	6exz
Base Motif Length	21 res.	20 res.
Length	118 res.	72 res.



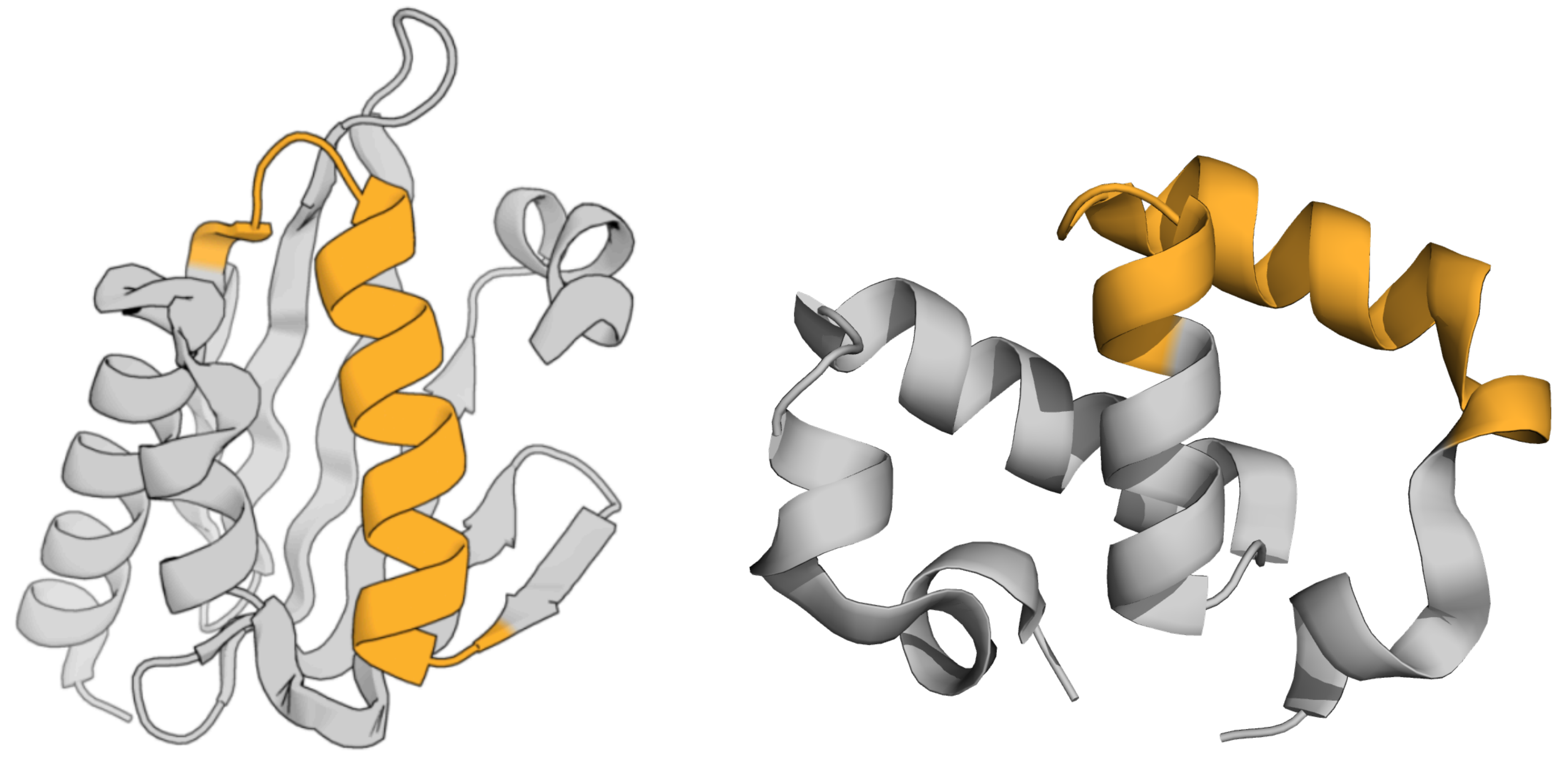
Motif-scaffolding case-studies

Evaluation Setup

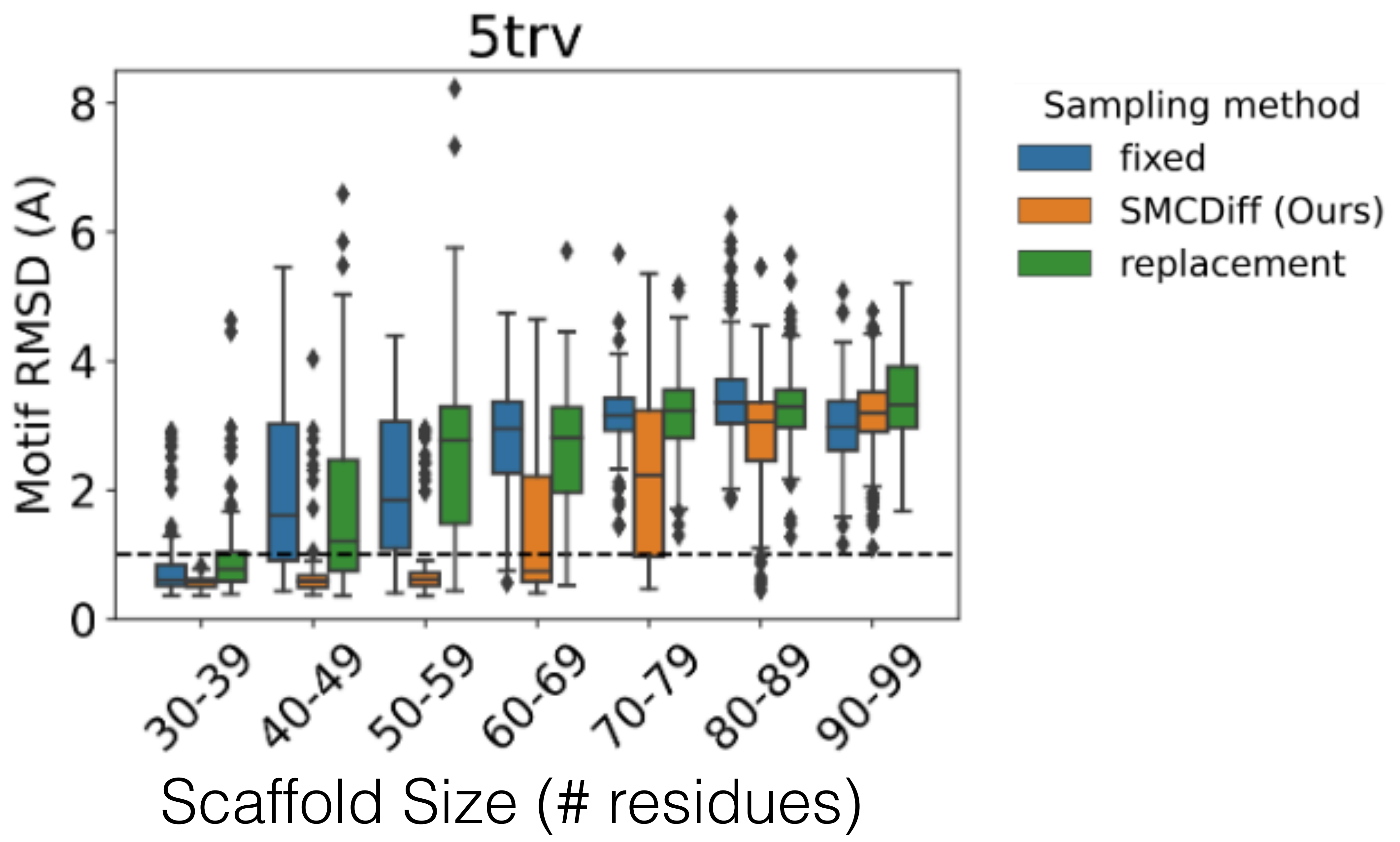
- Pick motifs from structures in PDB
- Know at least one solution exists

We test dependence scaffold length

- Provide variable length "pads" of native scaffold
- On 5trv, SMCDiff builds diverse scaffolds up to 80 residues
  - "Naive" inpainting fails beyond 50 residues



	5trv	6exz
Base Motif Length	21 res.	20 res.
Length	118 res.	72 res.



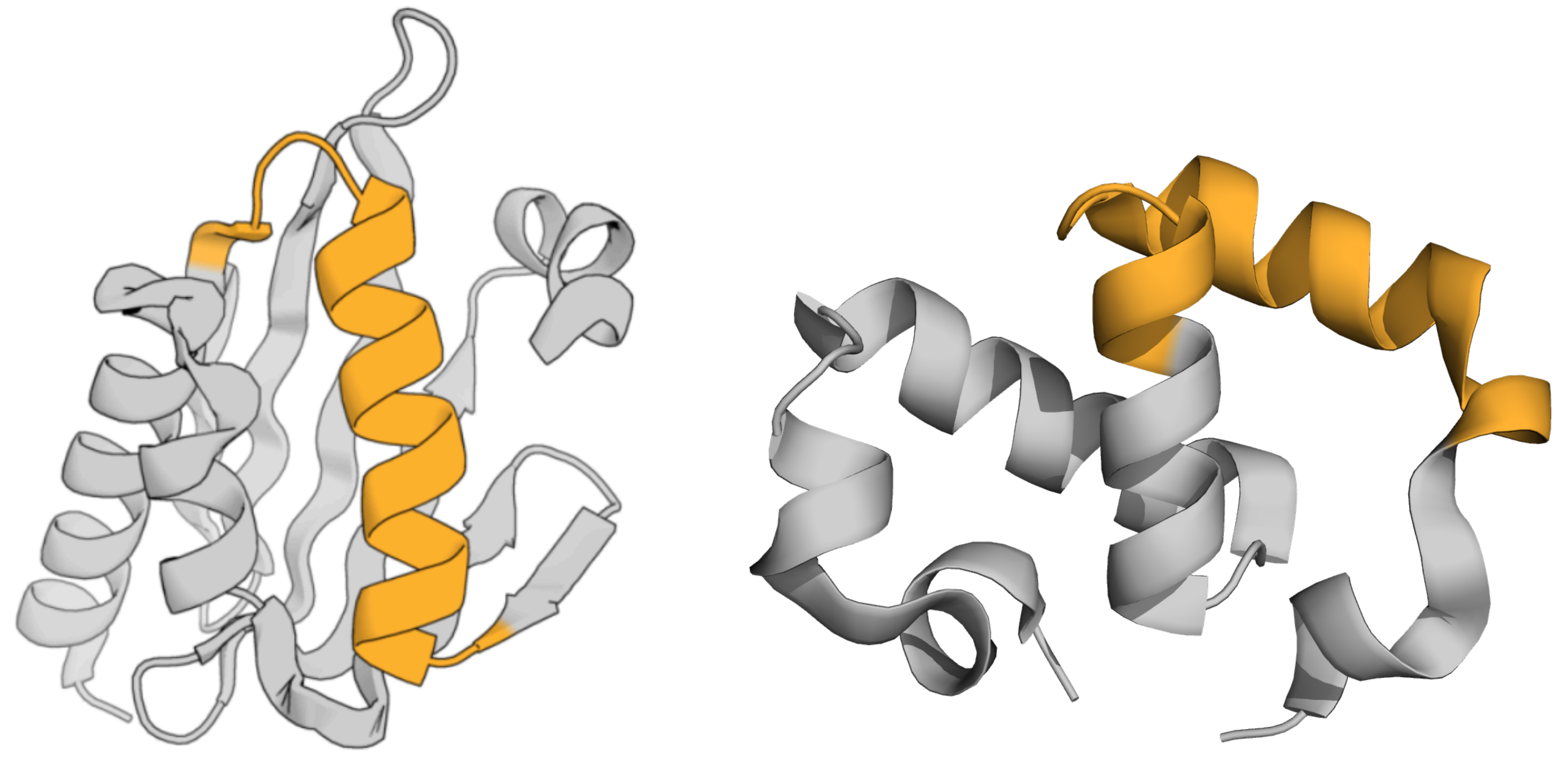
Motif-scaffolding case-studies

Evaluation Setup

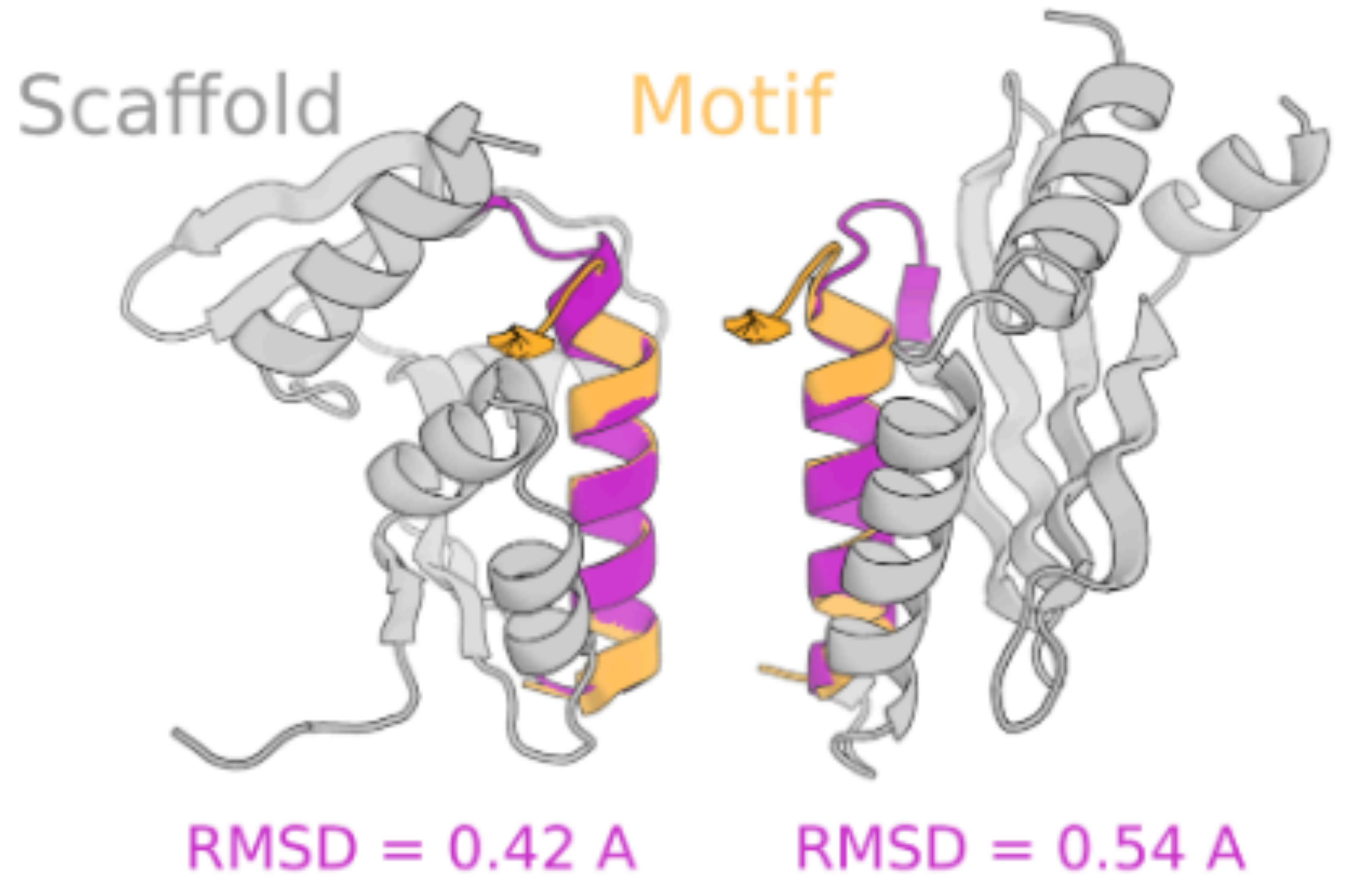
- Pick motifs from structures in PDB
- Know at least one solution exists

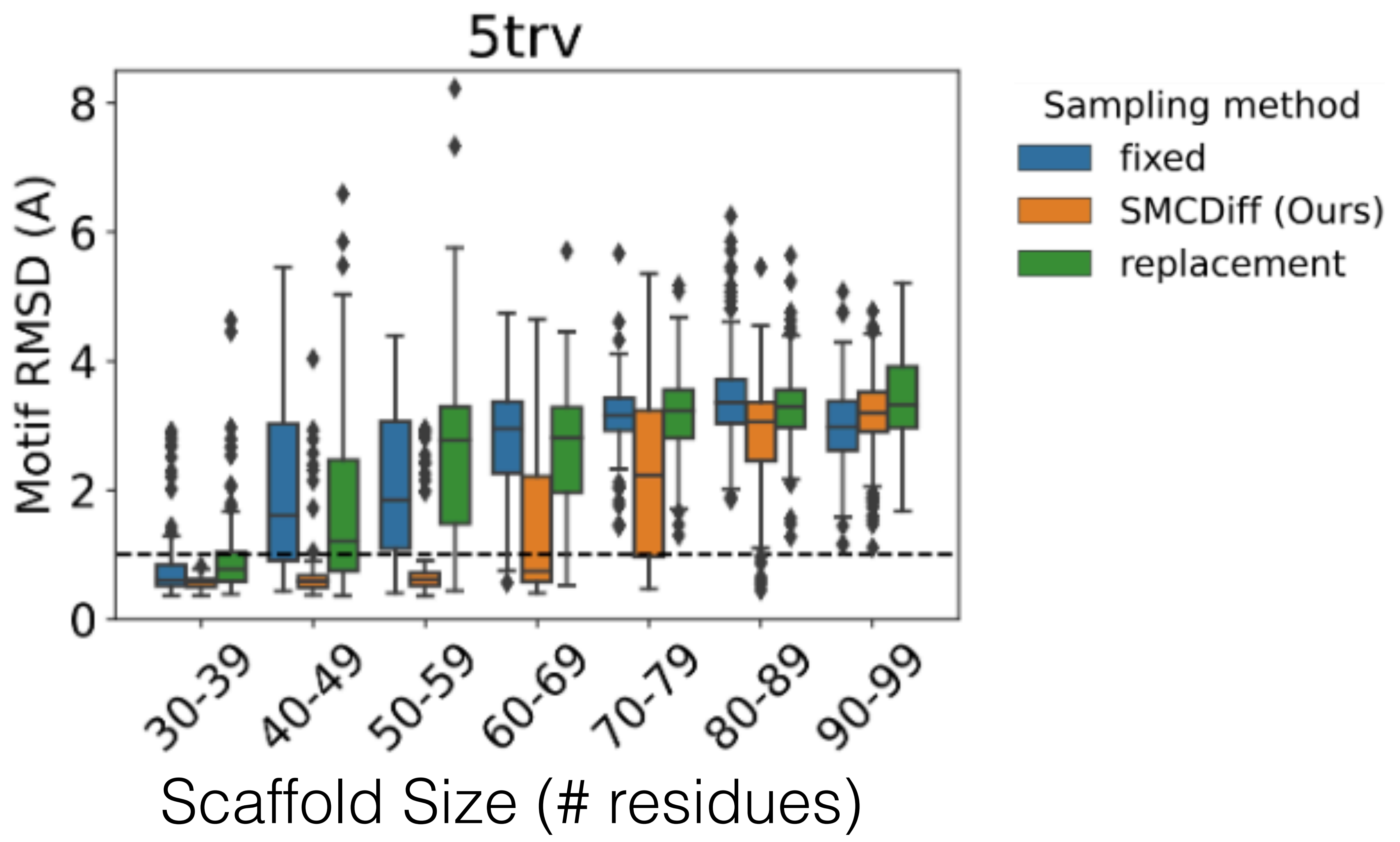
We test dependence scaffold length

- Provide variable length "pads" of native scaffold
- On 5trv, SMCDiff builds diverse scaffolds up to 80 residues
  - "Naive" inpainting fails beyond 50 residues



	5trv	6exz
Base Motif Length	21 res.	20 res.
Length	118 res.	72 res.





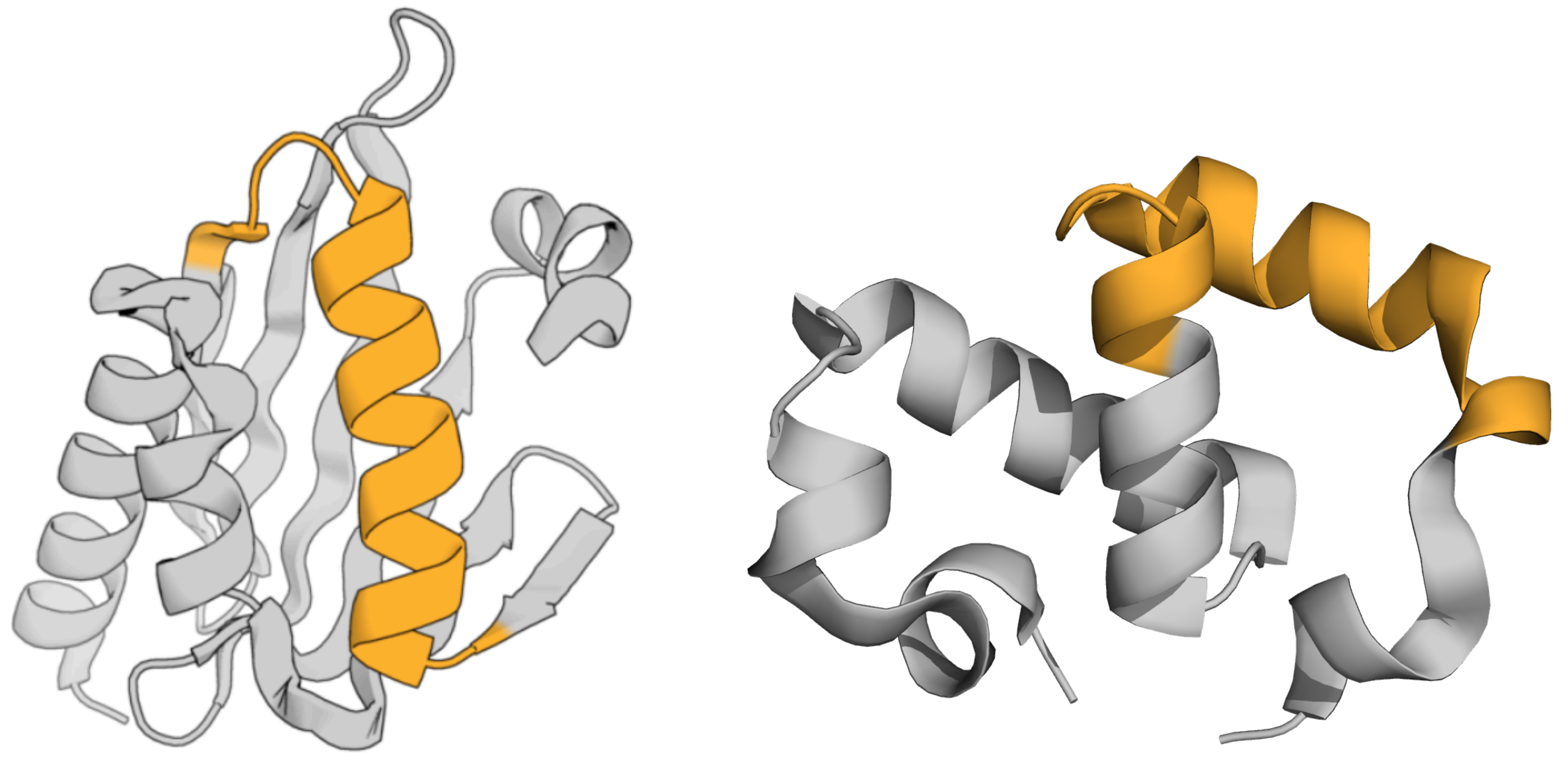
Motif-scaffolding case-studies

Evaluation Setup

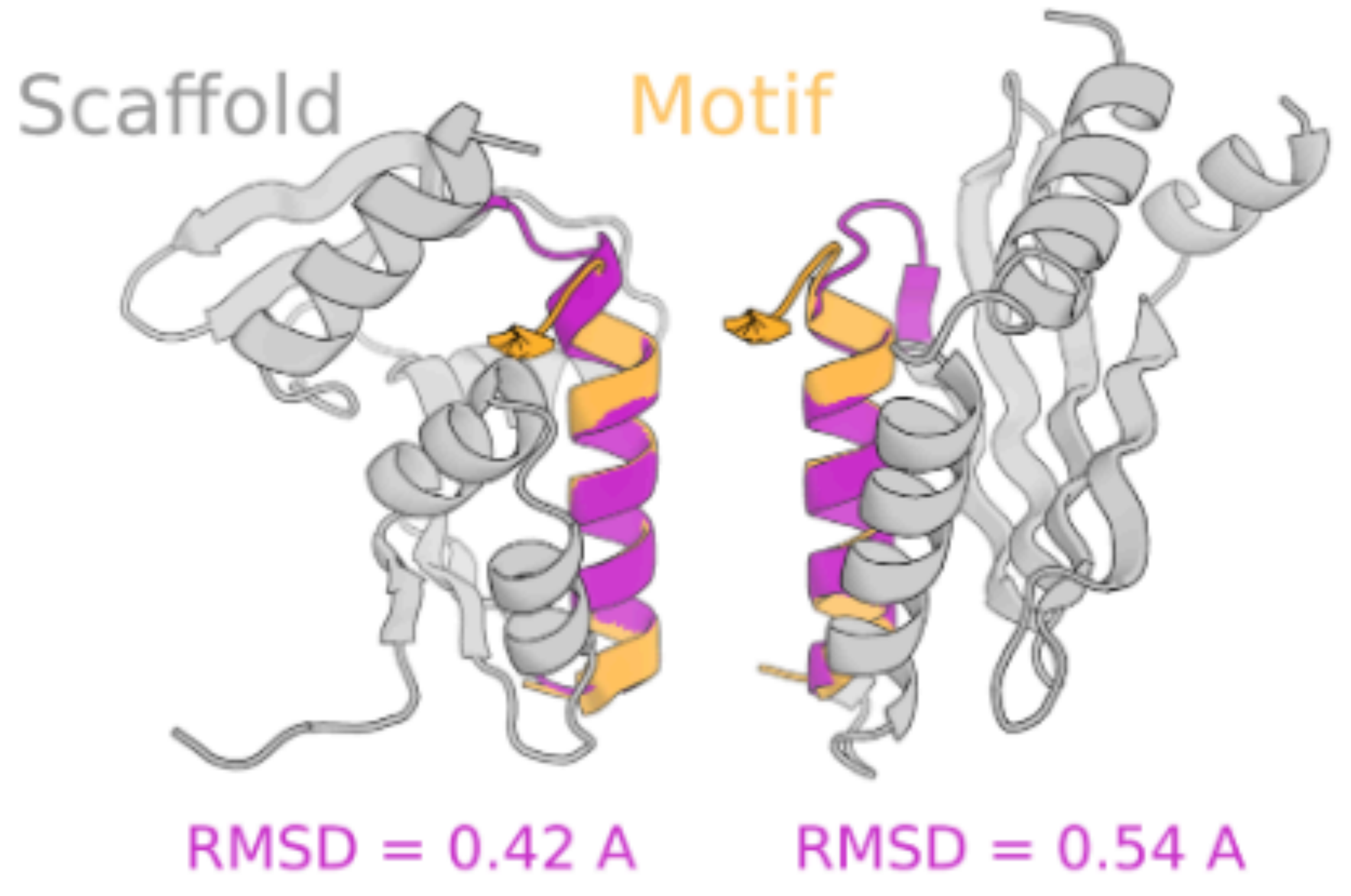
- Pick motifs from structures in PDB
- Know at least one solution exists

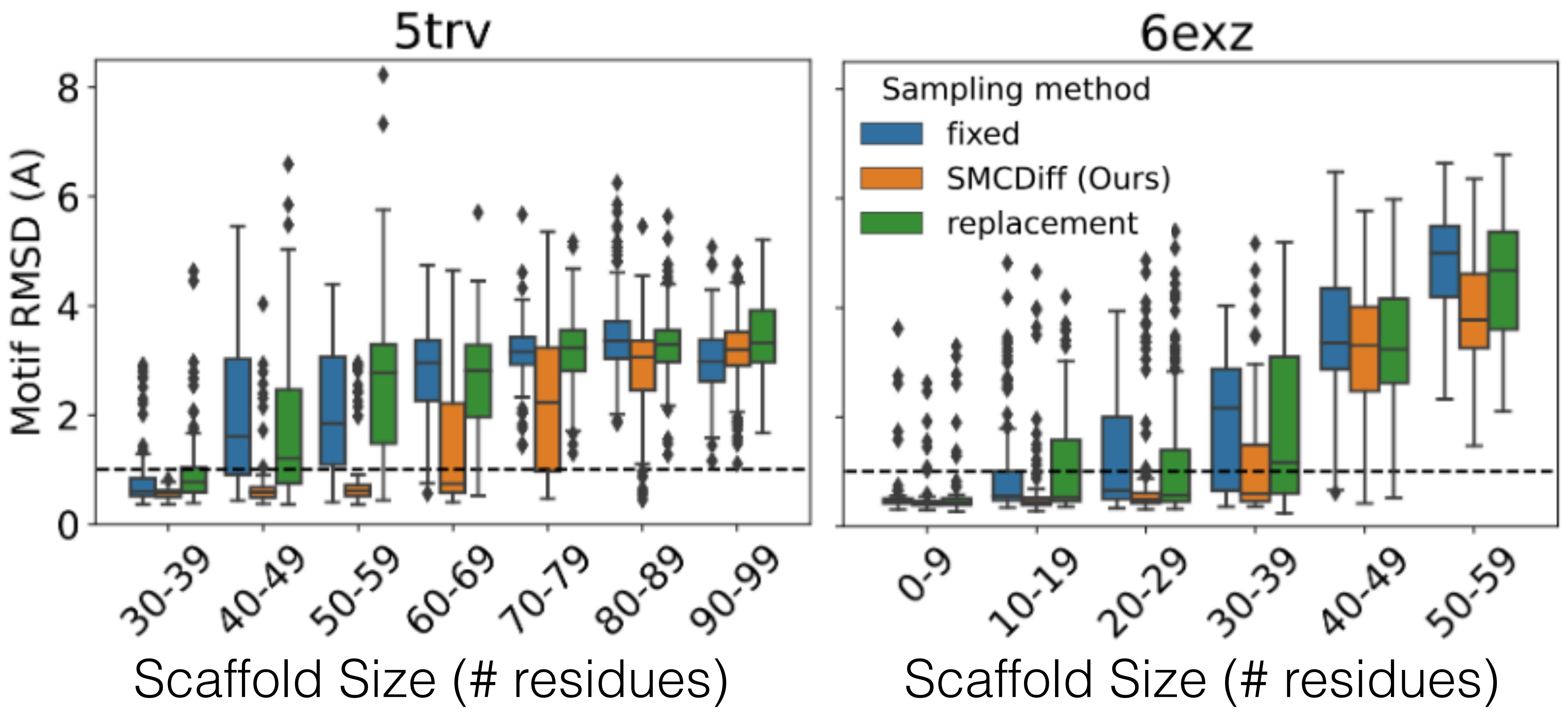
We test dependence scaffold length

- Provide variable length "pads" of native scaffold
- On 5trv, SMCDiff builds diverse scaffolds up to 80 residues
  - "Naive" inpainting fails beyond 50 residues



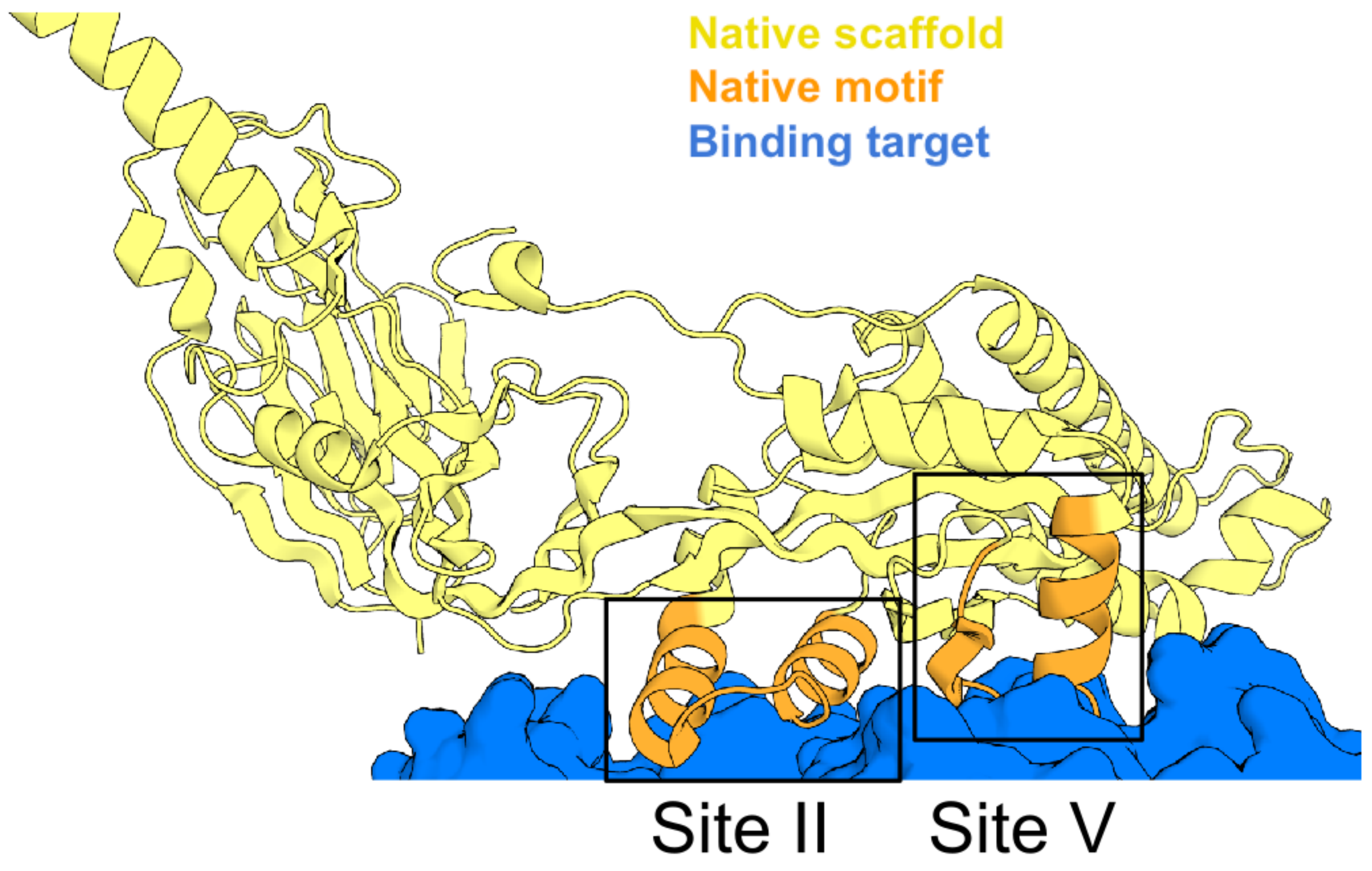
	5trv	6exz
Base Motif Length	21 res.	20 res.
Length	118 res.	72 res.



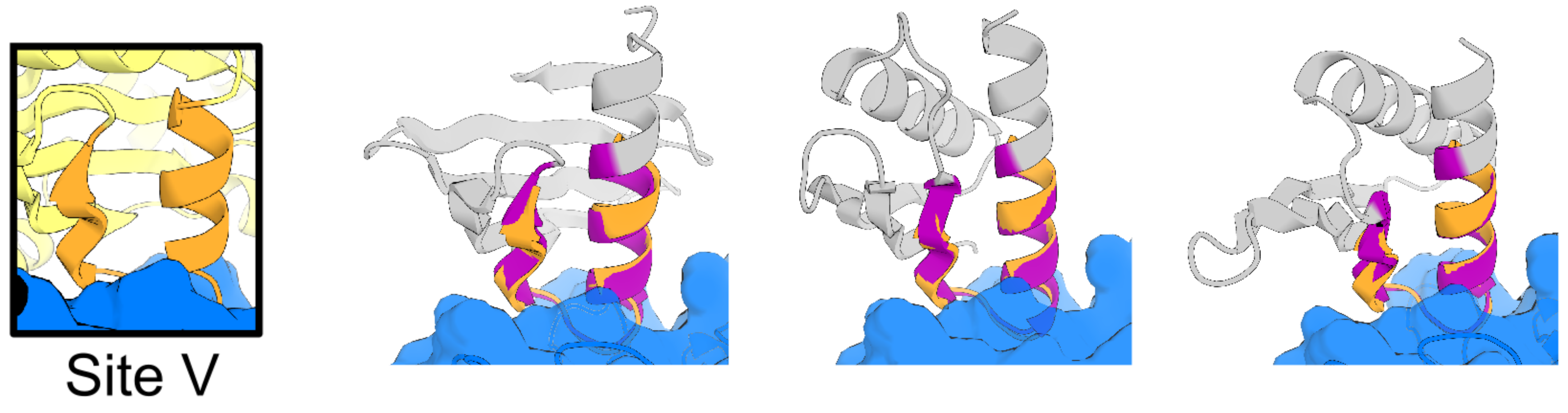


Failure case: RSV

Respiratory syncytial virus (RSV) neutralizing antibody binds site II and V.



- RSV known to be difficult to scaffold. Only recently [Wang+2022] successfully scaffolded Site V.



- SMCDiff fails to scaffold RSV.

[Figure adapted from Wang+ 2022]

Related work on motif-scaffolding & generative modeling

Related work on motif-scaffolding & generative modeling

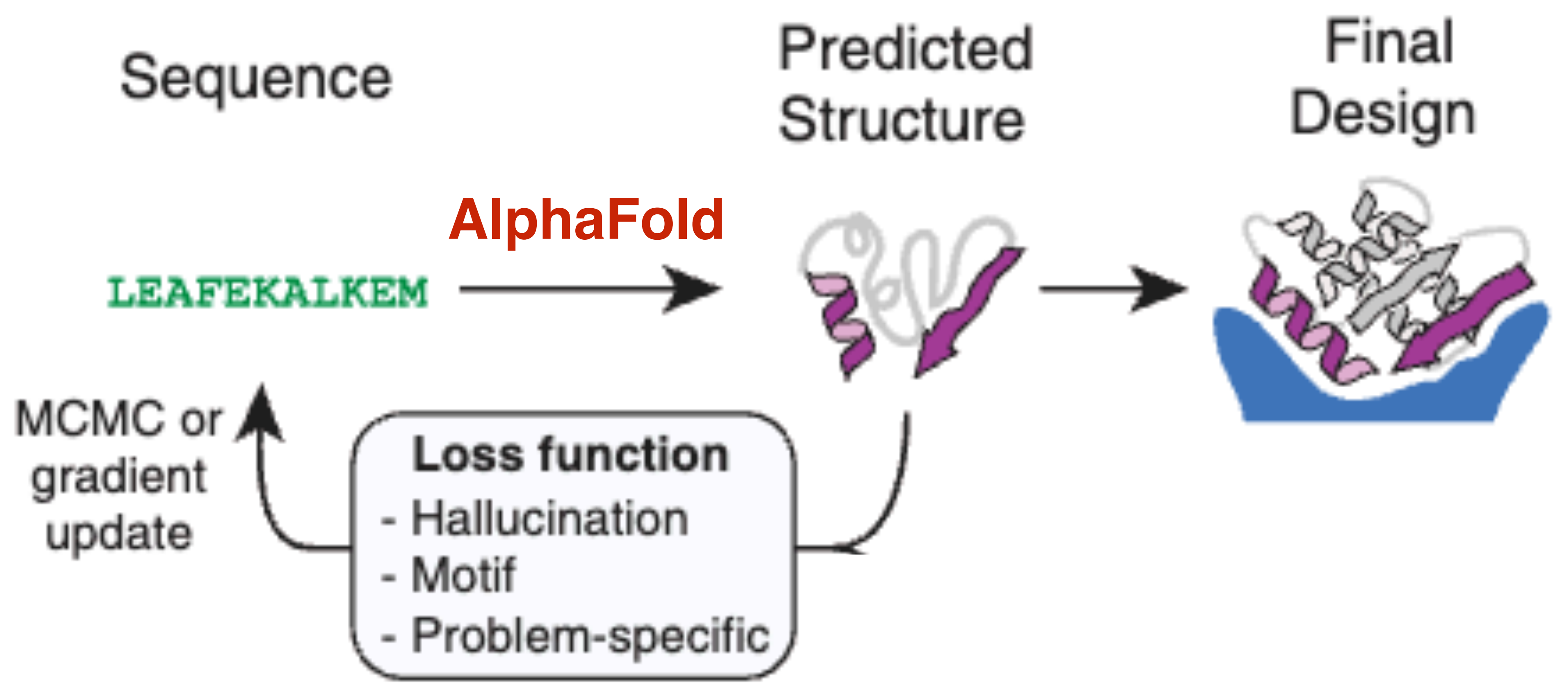
Non-generative motif-scaffolding is "state of the art", but has limitations:

Related work on motif-scaffolding & generative modeling

Non-generative motif-scaffolding is “state of the art”, but has limitations:

- Hallucination: search over on sequence input to AlphaFold

Hallucination [Anishchenko+, 2021; Wang+, 2022]

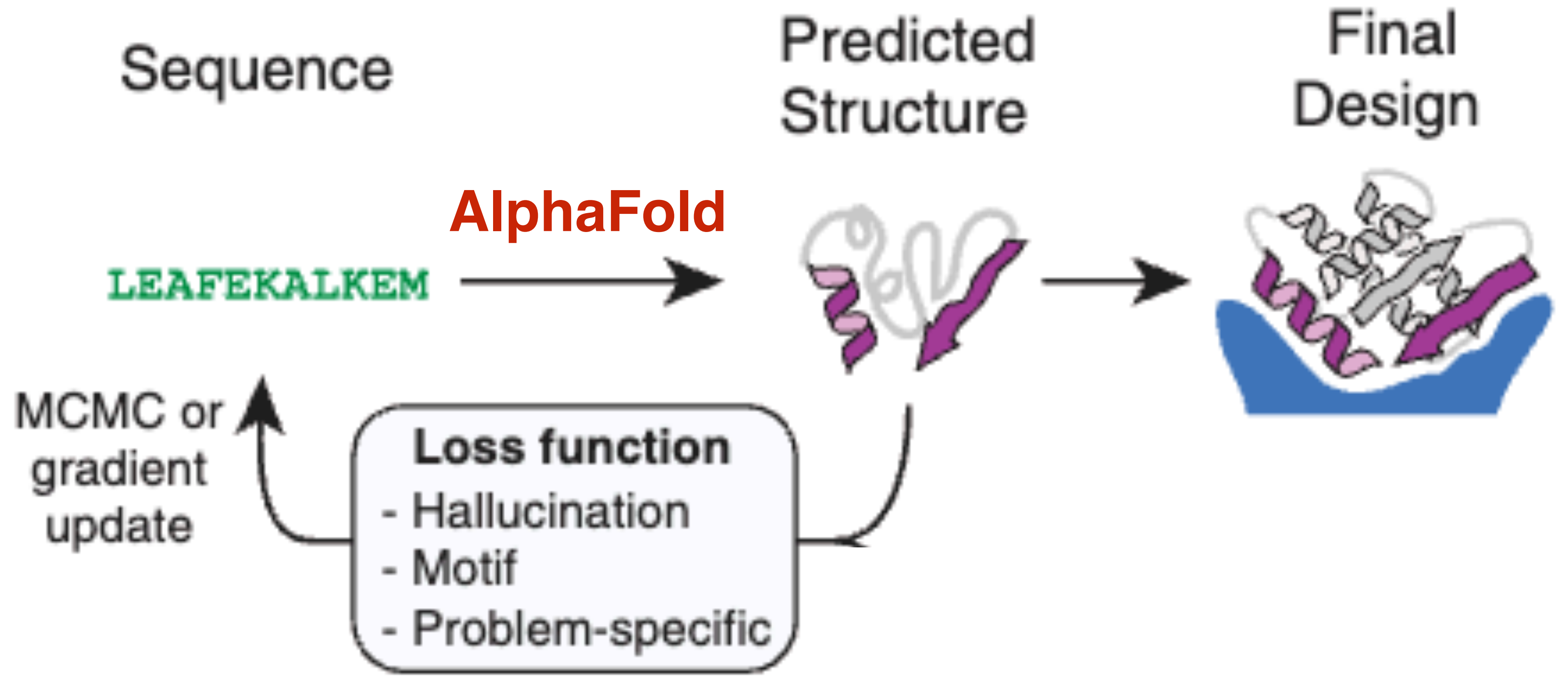


Related work on motif-scaffolding & generative modeling

Non-generative motif-scaffolding is "state of the art", but has limitations:

- Hallucination: search over on sequence input to AlphaFold
  - Susceptible to adversarial examples
  - Compute cost of hours to days with variable success rates

Hallucination [Anishchenko+, 2021; Wang+, 2022]

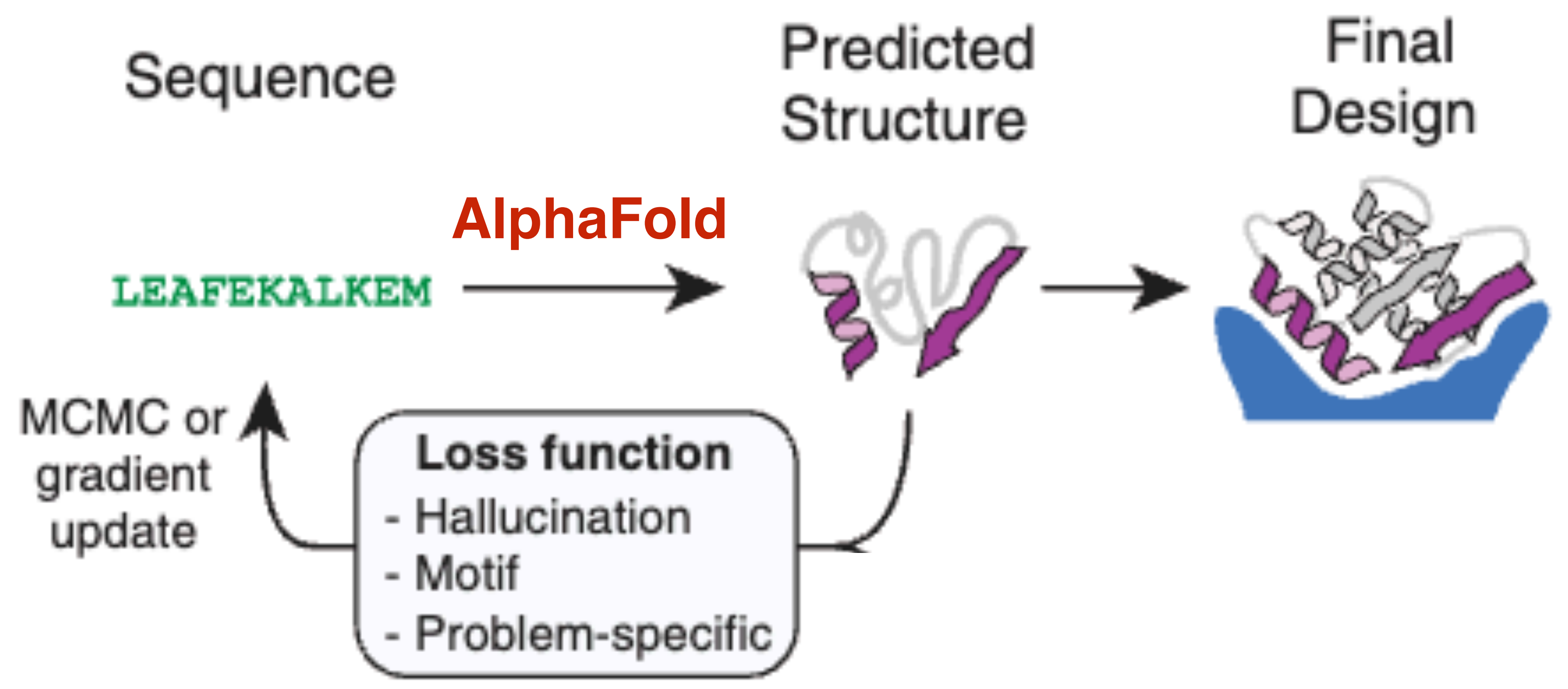


Related work on motif-scaffolding & generative modeling

Non-generative motif-scaffolding is "state of the art", but has limitations:

- Hallucination: search over on sequence input to AlphaFold
  - Susceptible to adversarial examples
  - Compute cost of hours to days with variable success rates
- RosettaFold "Missing information recovery" [Wang+, 2022]: Supervised retraining

Hallucination [Anishchenko+, 2021; Wang+, 2022]

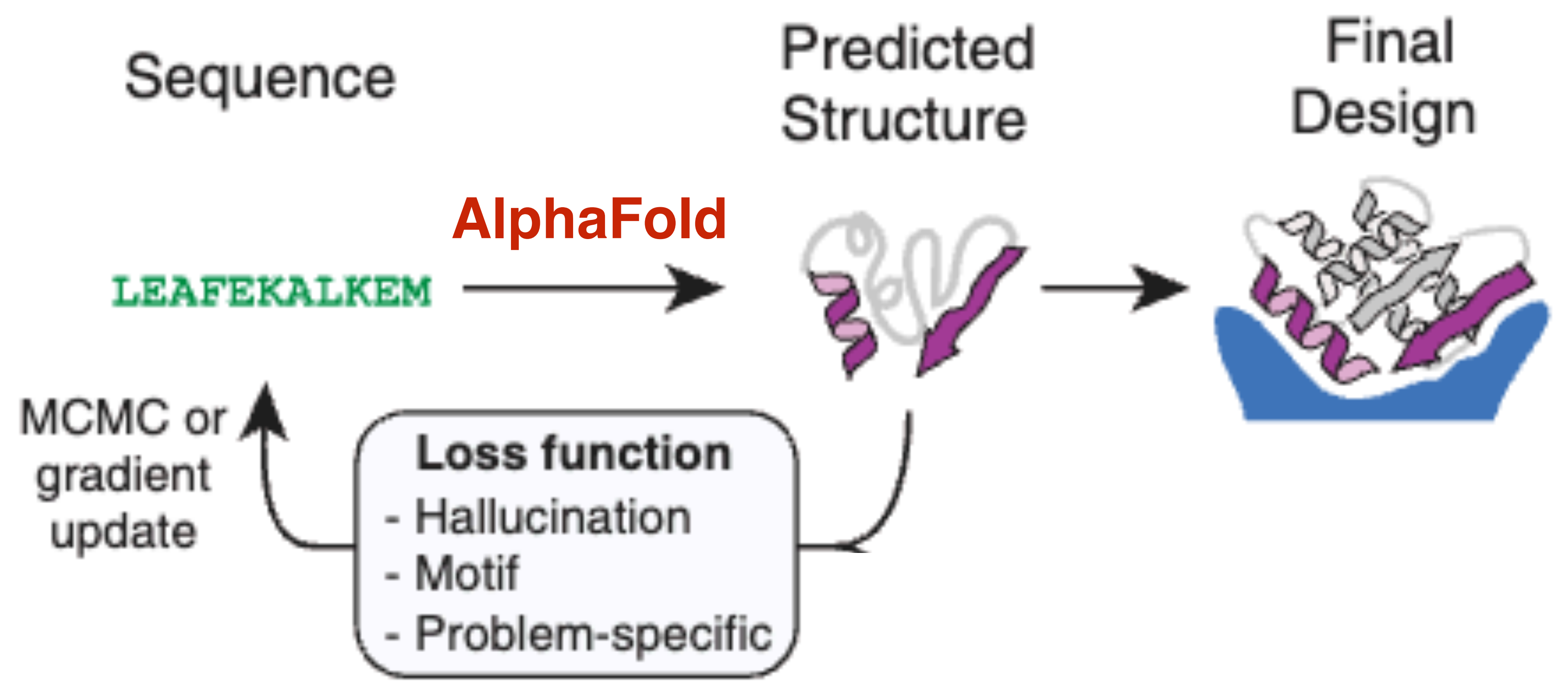


Related work on motif-scaffolding & generative modeling

Non-generative motif-scaffolding is "state of the art", but has limitations:

- Hallucination: search over on sequence input to AlphaFold
  - Susceptible to adversarial examples
  - Compute cost of hours to days with variable success rates
- RosettaFold "Missing information recovery" [Wang+, 2022]: Supervised retraining
  - Limited diversity, low performance for >40 residue scaffolds

Hallucination [Anishchenko+, 2021; Wang+, 2022]



Related work on motif-scaffolding & generative modeling

Related work on motif-scaffolding & generative modeling

Generative models have made recent strides:

Related work on motif-scaffolding & generative modeling

Generative models have made recent strides:

- Modeling distance matrices as images [Lin+2021, Anand+2017, Lee+2022]

Related work on motif-scaffolding & generative modeling

Generative models have made recent strides:

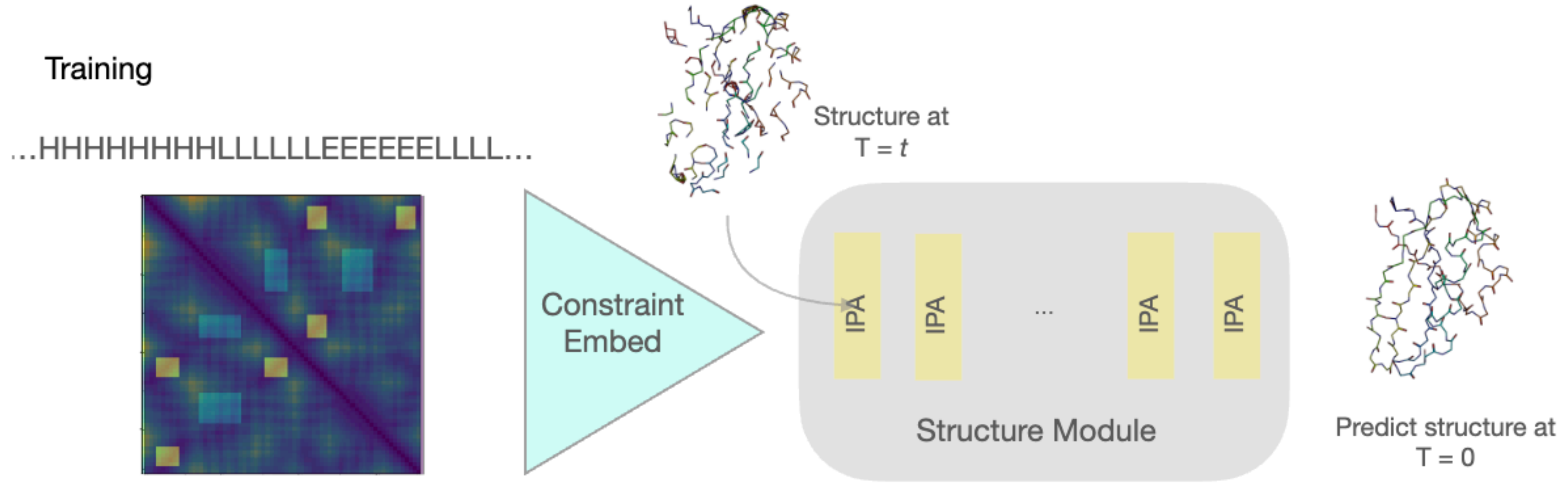
- Modeling distance matrices as images [Lin+2021, Anand+2017, Lee+2022]
  - Rely on non-differentiable "folding" as second step

Related work on motif-scaffolding & generative modeling

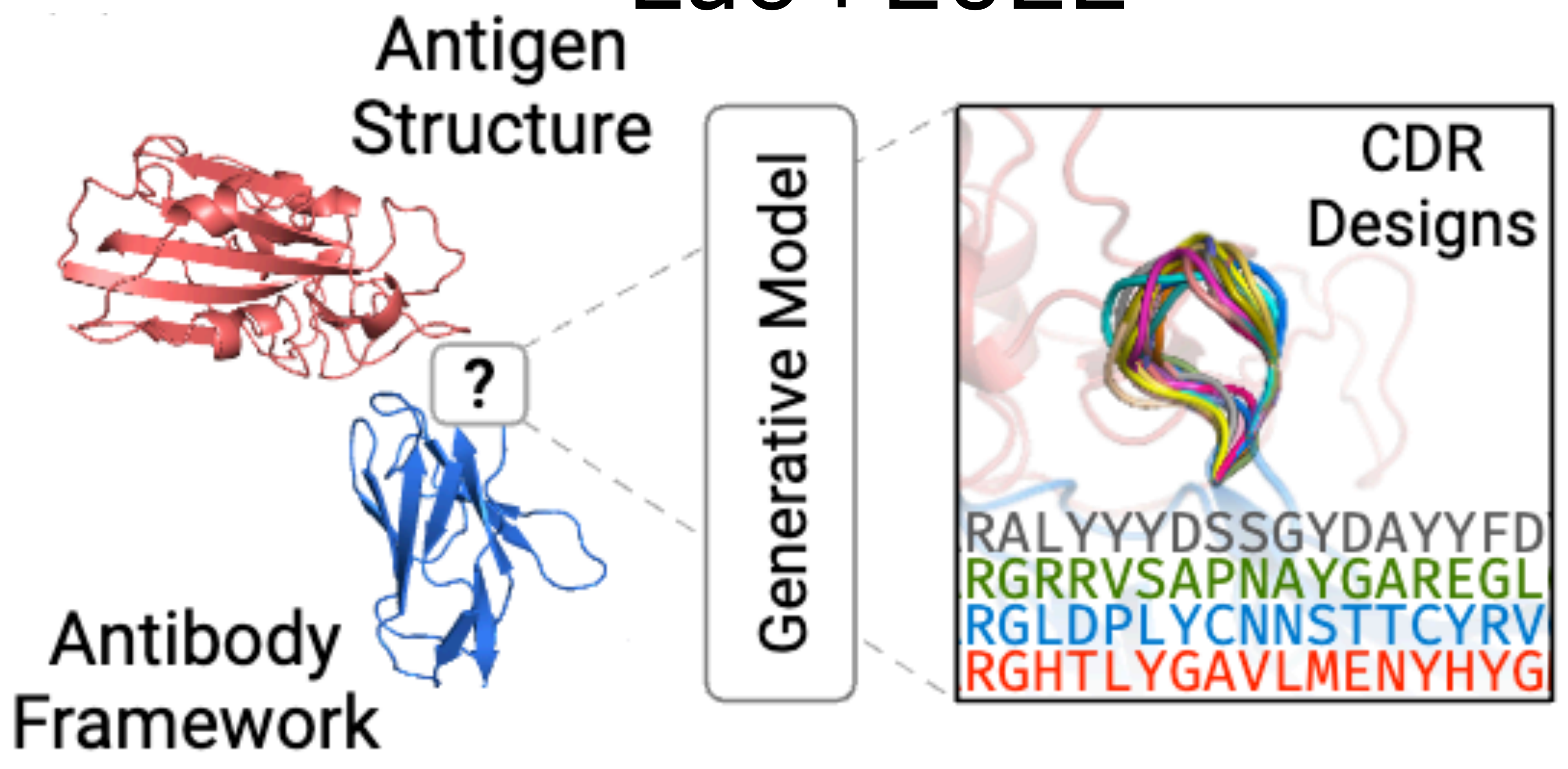
Generative models have made recent strides:

- Modeling distance matrices as images [Lin+2021, Anand+2017, Lee+2022]
  - Rely on non-differentiable "folding" as second step
- Concurrent work on diffusion in 3D [Anand+2022, Luo+2022]
  - No demonstrations of "unconditional" sampling

Anand+2022



Luo+2022



Related work on motif-scaffolding & generative modeling

Related work on motif-scaffolding & generative modeling

ProtDiff + SMCDiff Advantages:

- Unconditional sampling of diverse backbones
- Generative framework allows efficient sampling of diverse scaffolds
- Can scaffold up to 80 residues

Related work on motif-scaffolding & generative modeling

ProtDiff + SMCDiff Advantages:

- Unconditional sampling of diverse backbones
- Generative framework allows efficient sampling of diverse scaffolds
- Can scaffold up to 80 residues

ProtDiff Limitations:

- Doesn’t yet extend beyond motifs in train set
- Requires pre-specifying scaffold length and motif placement

Conclusions

Diffusion models enable a probabilistic approach to scaffolding motifs

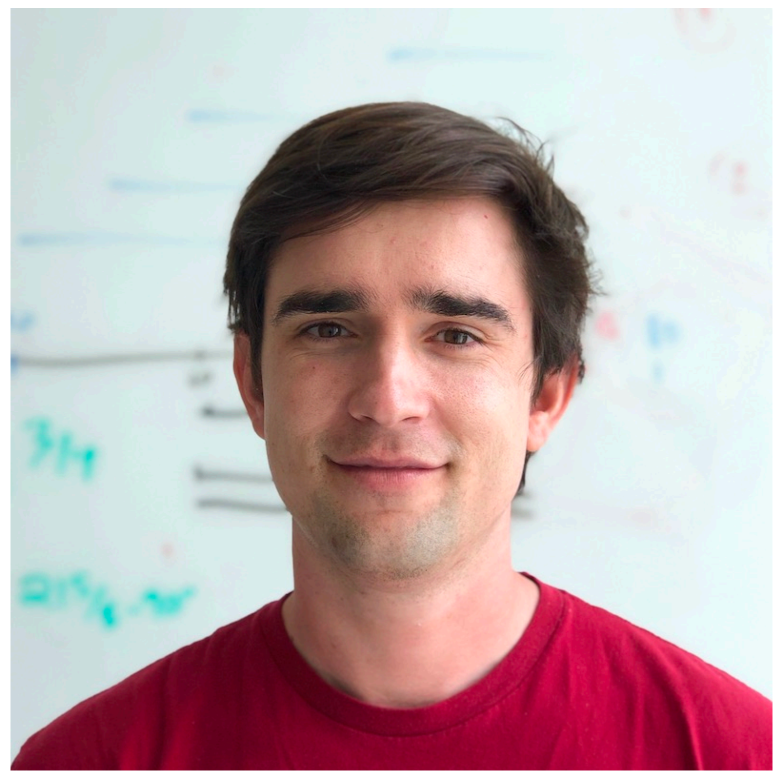
- **ProtDiff**, captures a distribution over diverse native backbones
- **SMCDiff**, provides accurate conditional samples, and outperforms naive inpainting on scaffolding problems

Conclusions

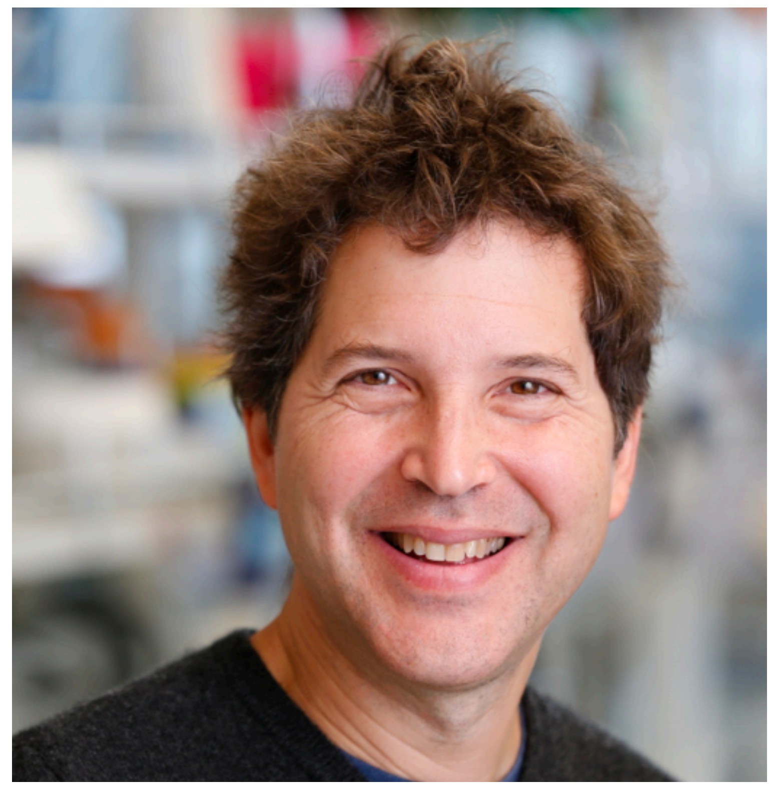
Diffusion models enable a probabilistic approach to scaffolding motifs

- **ProtDiff**, captures a distribution over diverse native backbones
- **SMCDiff**, provides accurate conditional samples, and outperforms naive inpainting on scaffolding problems

Collaborators


Doug Tischer

Tamara Broderick


David Baker

Regina Barzilay

Tommi Jaakkola

Primary Reference: "Diffusion probabilistic modeling of protein backbones in 3D for the motif-scaffolding problem." *arXiv preprint arXiv:2206.04119*

Contact: Brian Trippe (blt2114@columbia.edu), Jason Yim (jyim@mit.edu)

References

- **[Ho et al.]** Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 2020.
- **[Satorras et al.]** Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks. *International Conference on Machine Learning*, 2021.
- **[Hoogeboom et al.]** Emiel Hoogeboom, Victor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3D. *arXiv preprint* arXiv:2203.17003, 2022.
- **[Anand et al.]** Namrata Anand and Possu Huang. Generative modeling for protein structures. *Advances in Neural Information Processing Systems*, 2018.
- **[Anishchenko et al.]** Ivan Anishchenko, Samuel J Pellock, Tamuka M Chidyausiku, Theresa A Ramelot, Sergey Ovchinnikov, Jingzhou Hao, Khushboo Bafna, Christoffer Norn, Alex Kang, Asim K Bera, Frank DiMaio, Lauren Carter, Cameron M Chow, Gaetano T Montelione, and David Baker. De novo protein design by deep network hallucination. *Nature*, 2021.
- **[Wang et al.]** Jue Wang, Sidney Lisanza, David Juergens, Doug Tischer, Ivan Anishchenko, Minkyung Baek, Joseph L Watson, Jung Ho Chun, Lukas F Milles, Justas Dauparas, Marc Exposit, Wei Yang, Amijai Saragovi, Sergey Ovchinnikov, and David A. Baker. Deep learning methods for designing proteins scaffolding functional sites. *Science,* 2022.
- **[Xu et al.]** Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. GeoDiff: A Geometric Diffusion Model for Molecular Conformation Generation. *International Conference on Learning Representations*, 2022.
- **[Lin et al.]** Zeming Lin, Tom Sercu, Yann LeCun, and Alexander Rives. Deep generative models create new and diverse protein structures. *Machine Learning for Structural Biology Workshop, NeurIPS*, 2021.
- **[Song et al.]** Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *International Conference on Learning Representations*, 2021.
- **[Meng et al.]** Meng, Chenlin, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. *International Conference on Learning Representations,* 2021.