

MOLECULE GENERATION WITH GRAPH NEURAL NETWORKS AND PROBABILISTIC DIFFUSION

Gerrit Großmann

26.07.2024

github.com/gerritgr/diffusion_gnn_tutorial

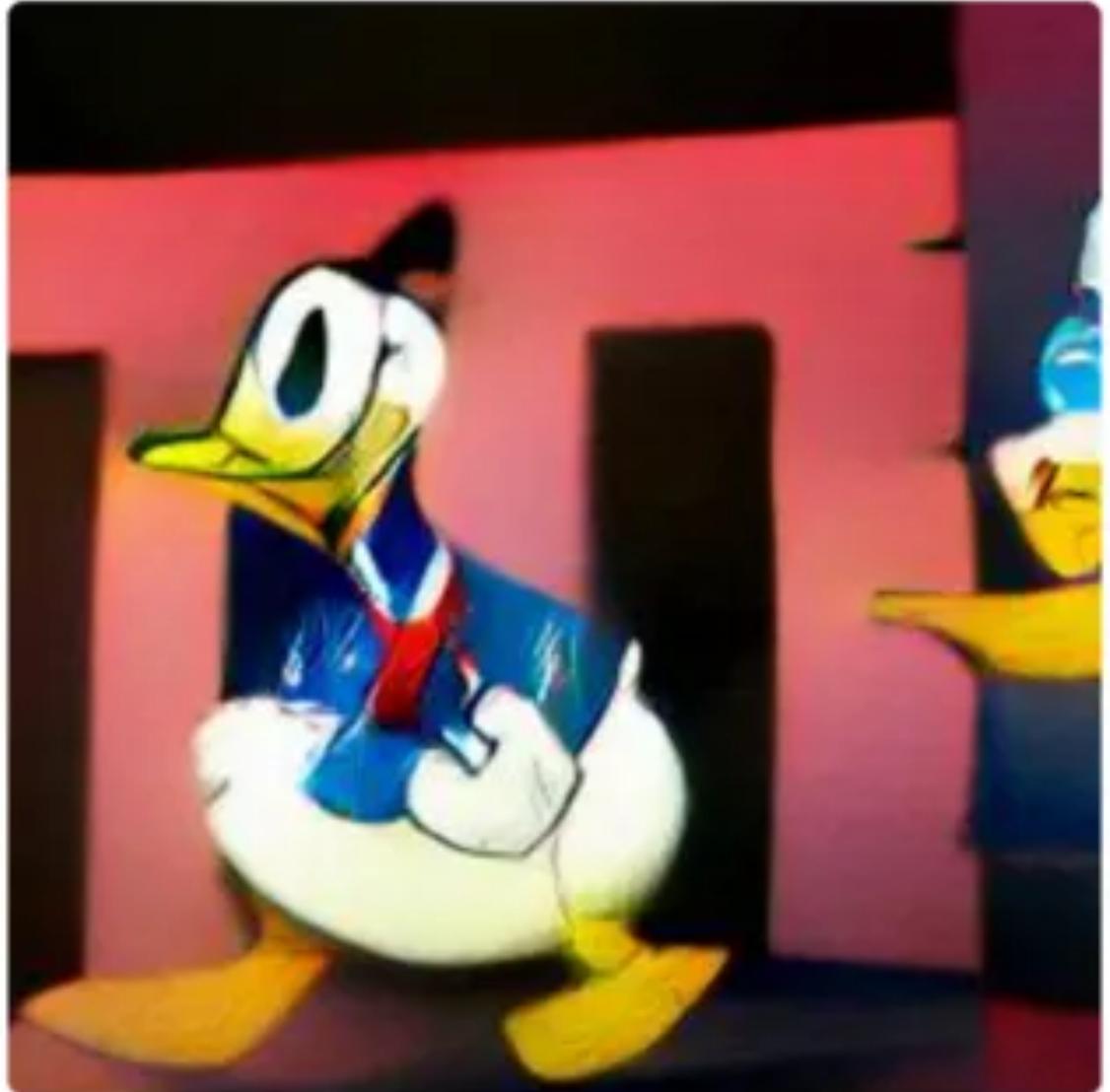
NextAID
Neuro-explicit AI for Drug Discovery

dfki
Deutsches Forschungszentrum
für Künstliche Intelligenz
*German Research Center for
Artificial Intelligence*

ScaDS.AI
DRESDEN LEIPZIG

Motivation

2022



2023



2024

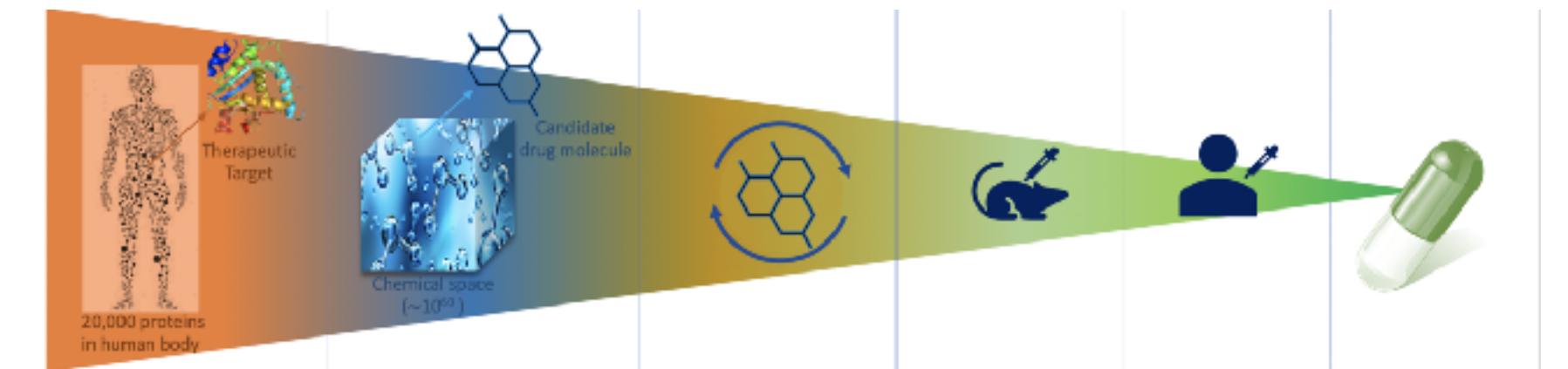


<https://www.sciencefocus.com/news/dall-e-mini-creator-explains-blurred-faces-going-viral-and-the-future-of-the-project>

youtube.com/watch?v=TU1gMlol0kc&ab_channel=MagnaAI

Motivation

“What works for images should also work for medicine!”



[linkedin.com/pulse/ai-drug-discovery-vijay-morampudi](https://www.linkedin.com/pulse/ai-drug-discovery-vijay-morampudi)

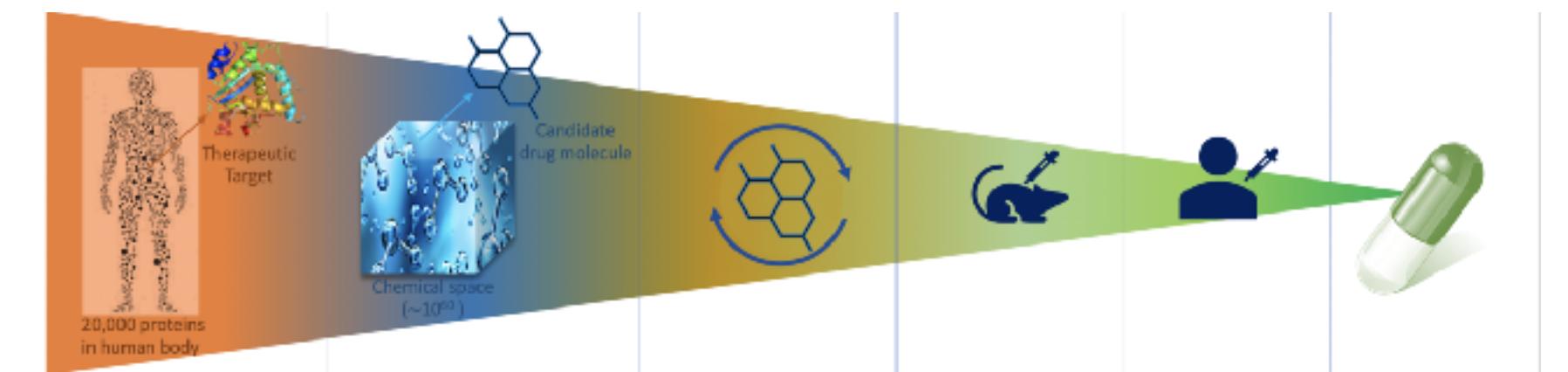
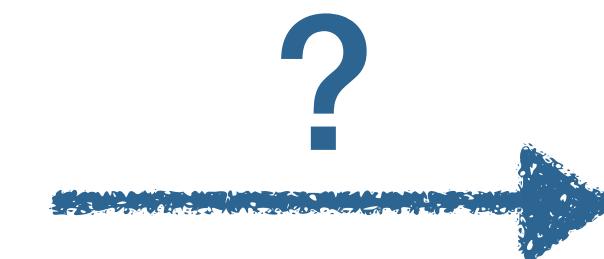
Jeroen Pixel twitter.com/pixelprotest_/status/1743227361888186773

Motivation

“What works for images should also work for medicine!”



“Painting images and making drugs is not the same thing!”



[linkedin.com/pulse/ai-drug-discovery-vijay-morampudi](https://www.linkedin.com/pulse/ai-drug-discovery-vijay-morampudi)

Jeroen Pixel twitter.com/pixelprotest_/status/1743227361888186773

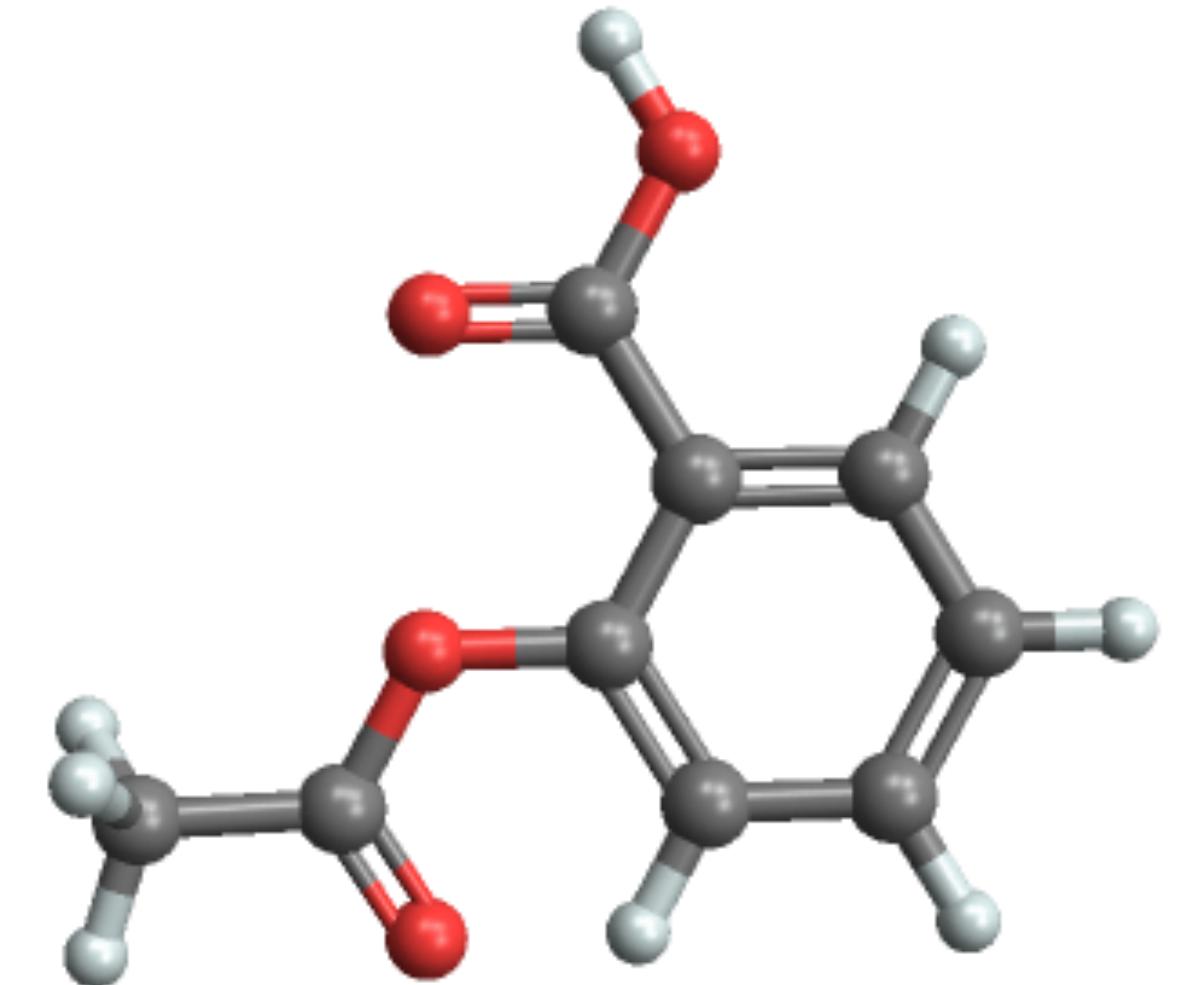
Motivation



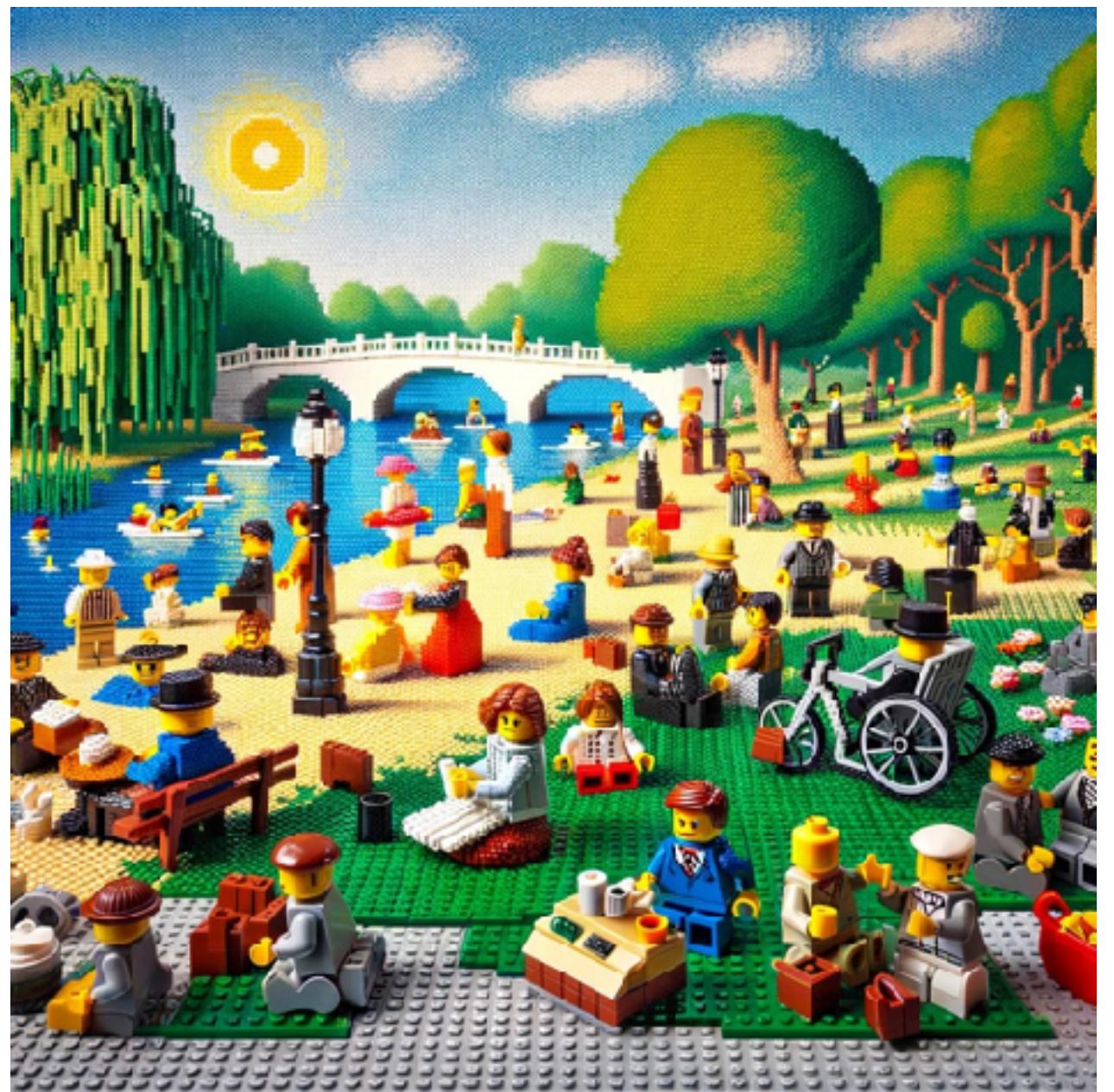
Jeroen Pixel twitter.com/pixelprotest_/status/1743227361888186773

- High Dimensionality
- Large amount of data
- Complex manifold
- World model
- Conditional generation
- Ethically dubious potential

- Domain knowledge
- 3D objects
- Evaluation
- Discrete variables

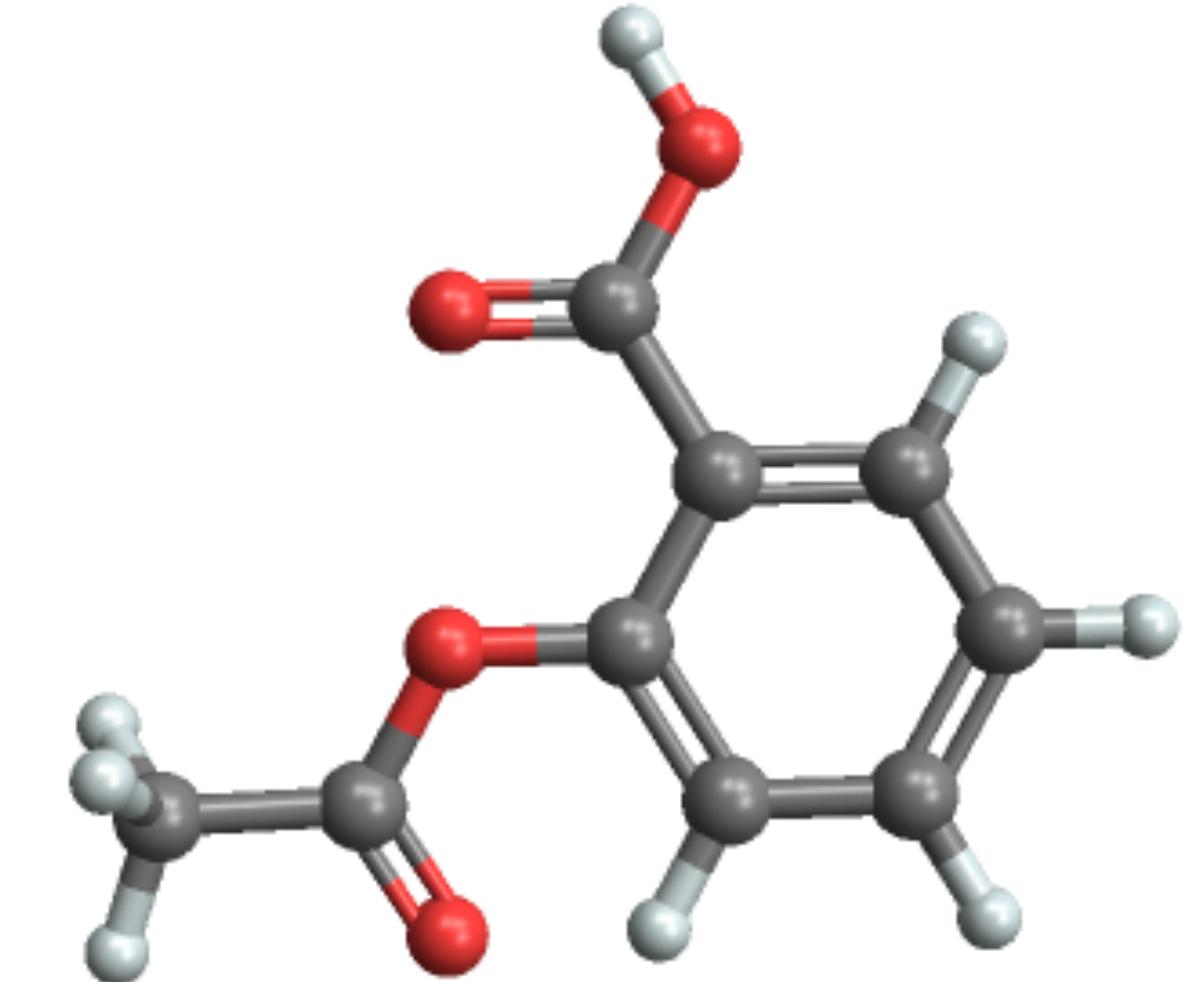


Motivation



Jeroen Pixel twitter.com/pixelprotest_/status/1743227361888186773

- High Dimensionality
- Large amount of data
- Complex manifold
- World model
- Conditional generation
- Ethically dubious potential
- Domain knowledge
- 3D objects
- Evaluation
- Discrete variables



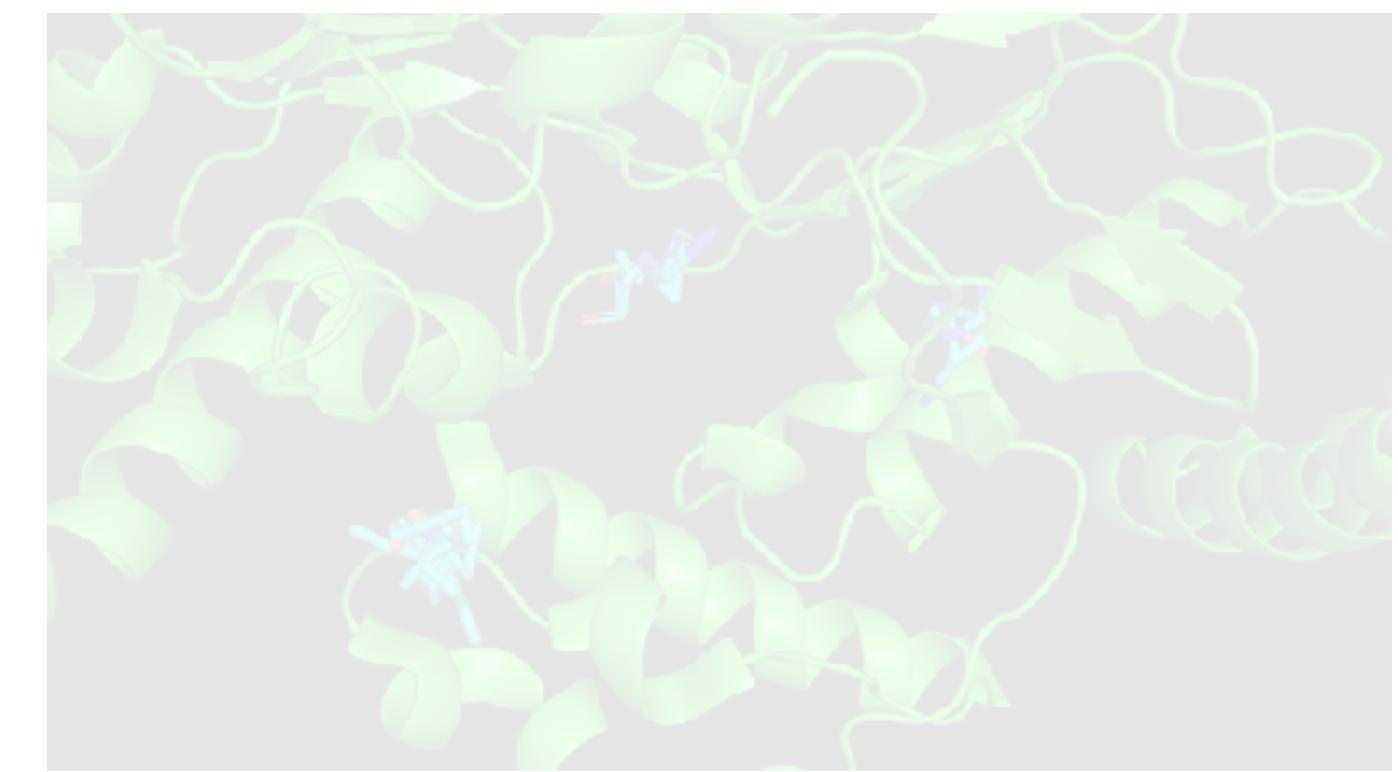
Outline



What are diffusion models
and how do they work?



Graph neural networks



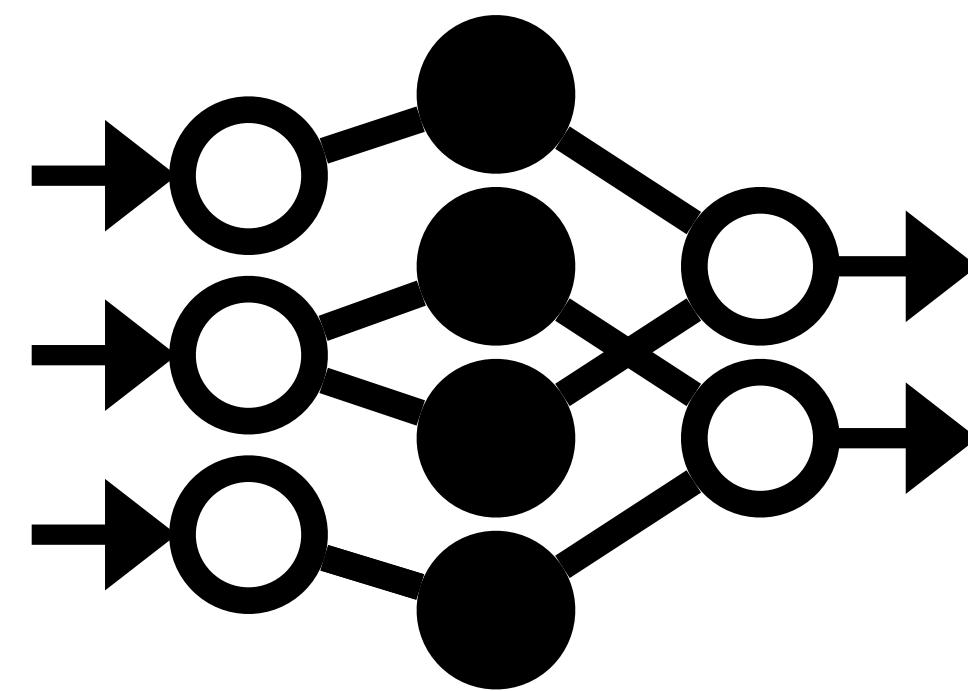
github.com/HannesStark/FlowSite

SOTA, challenges, and
limitations for
de novo drug design

Outline



What are diffusion models
and how do they work?



Graph neural networks



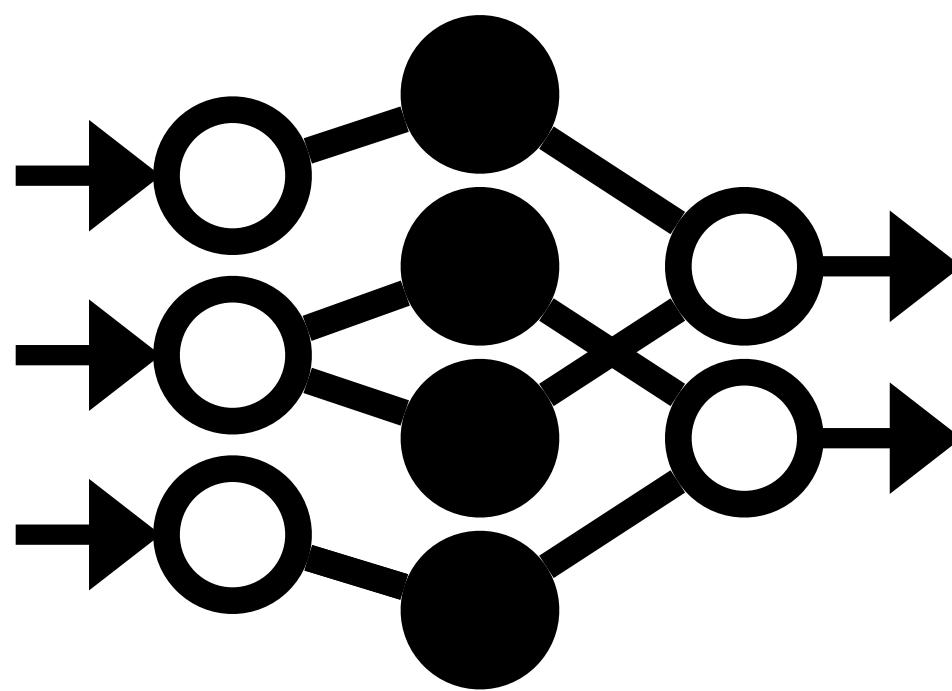
github.com/HannesStark/FlowSite

SOTA, challenges, and
limitations for
de novo drug design

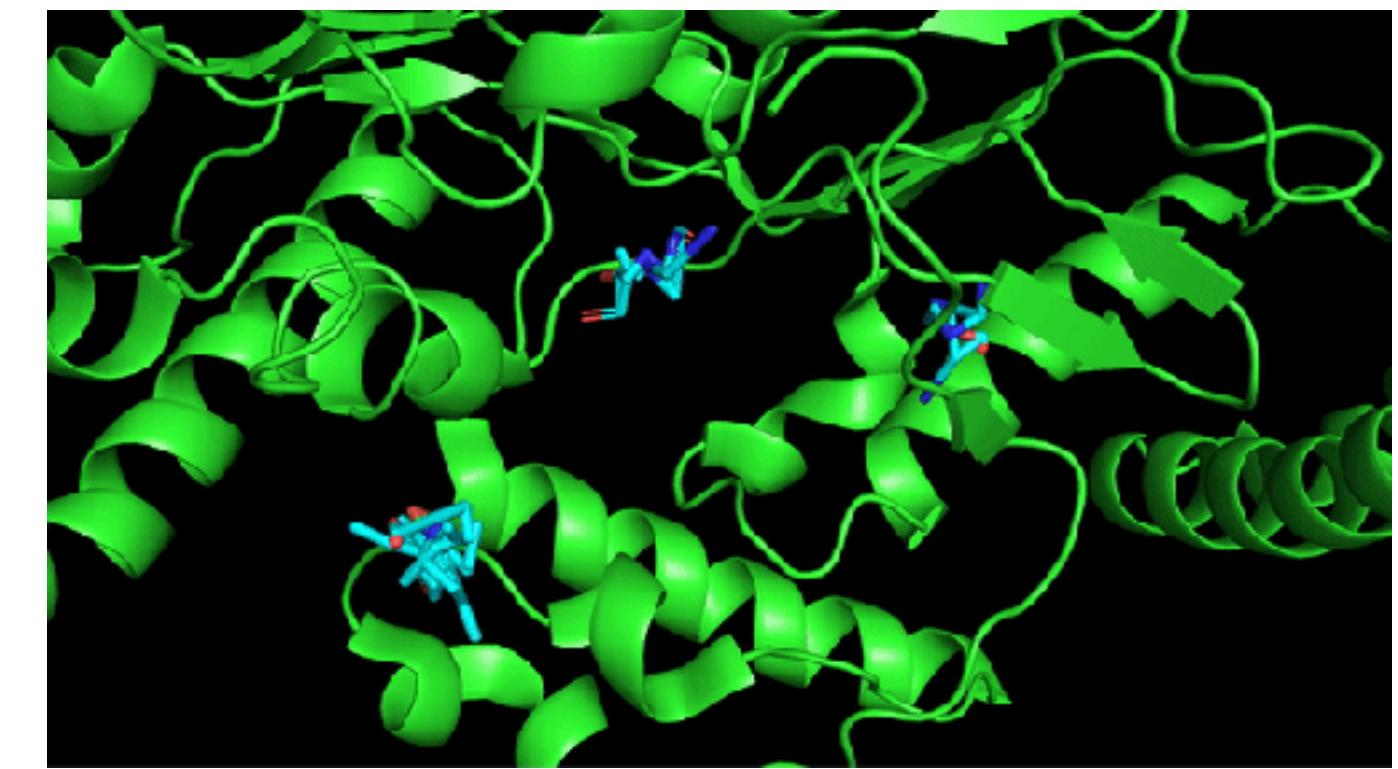
Outline



What are diffusion models
and how do they work?

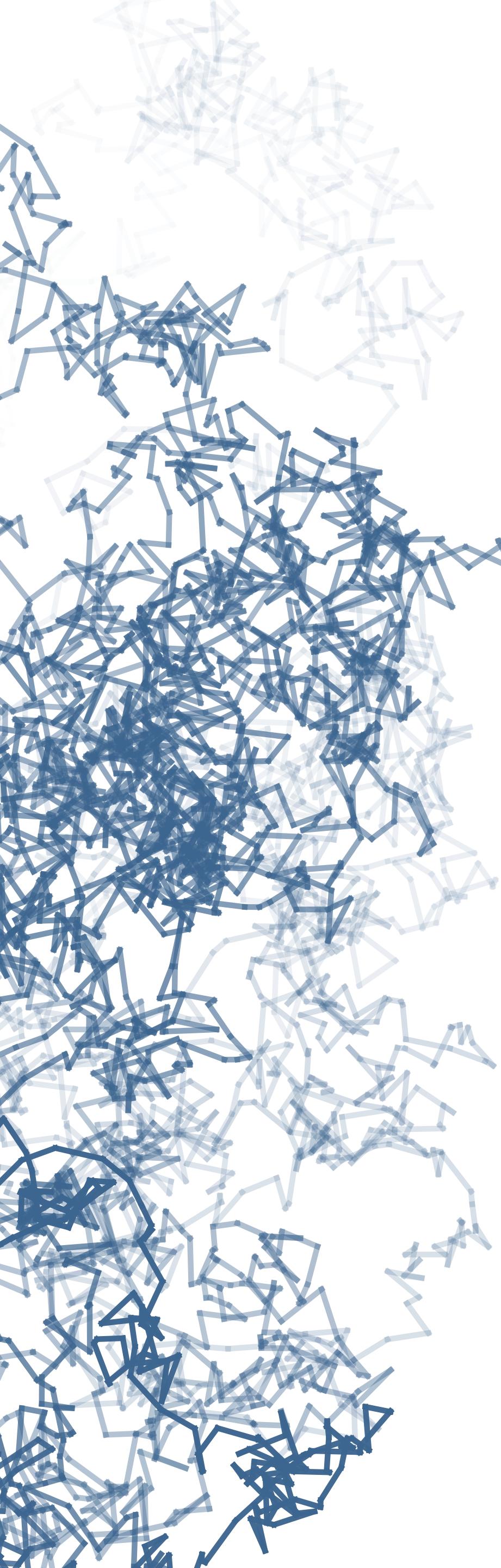


Graph neural networks



github.com/HannesStark/FlowSite

SOTA, challenges, and
limitations for
de novo drug design



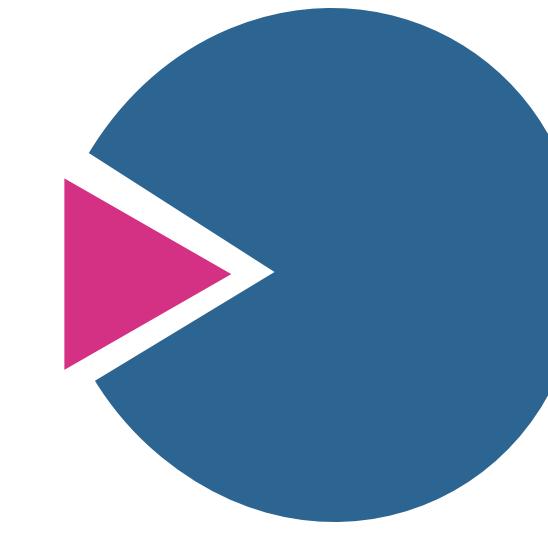
Part I

DRUG DISCOVERY

How Drugs Work

Most medical intervention are based on the fact that
a **small molecule** fits into a **large molecule**.

small molecule
drug
ligand
inhibitor

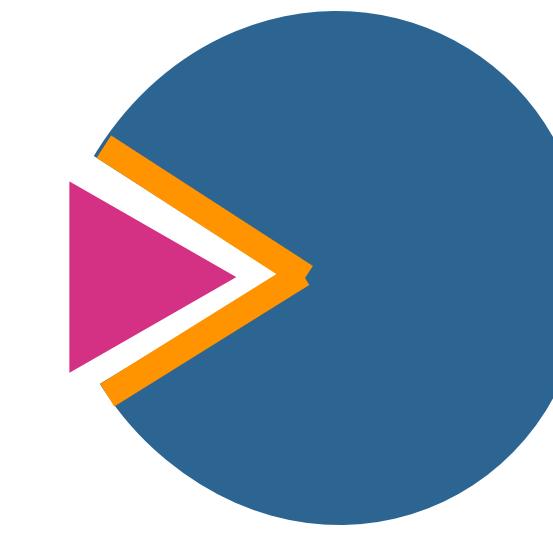


protein
enzyme
receptor
drug target

How Drugs Work

Most medical intervention are based on the fact that
a **small molecule** fits into a **large molecule**.

small molecule
drug
ligand
inhibitor



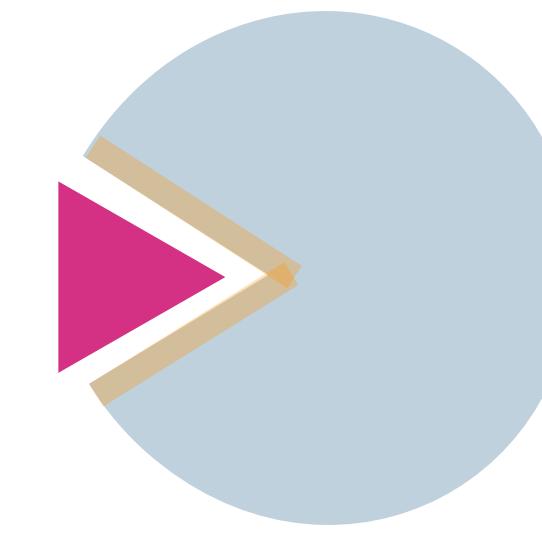
protein
enzyme
receptor
drug target

binding site
pocket

What is a Small Molecule

Most medical intervention are based on the fact that
a **small molecule** fits into a **large molecule**.

small molecule
drug
ligand
inhibitor



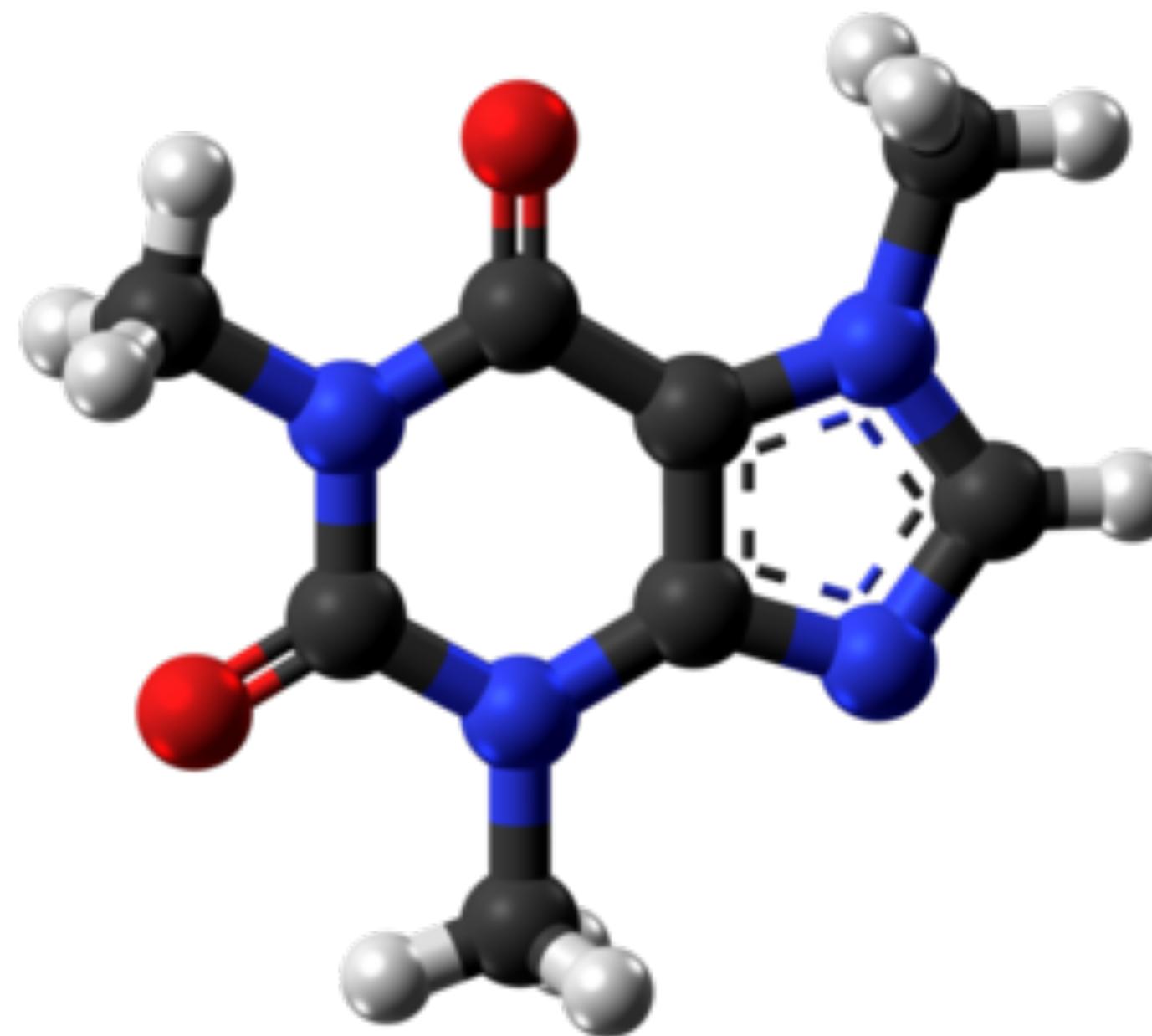
protein
enzyme
receptor
drug target

binding site
pocket

What is a (Small) Molecule

Molecule

- group of atoms connected through chemical bonds
- electrostatic force shapes geometry
- properties emerge from its constituent atoms
- dynamic and complex three-dimensional objects

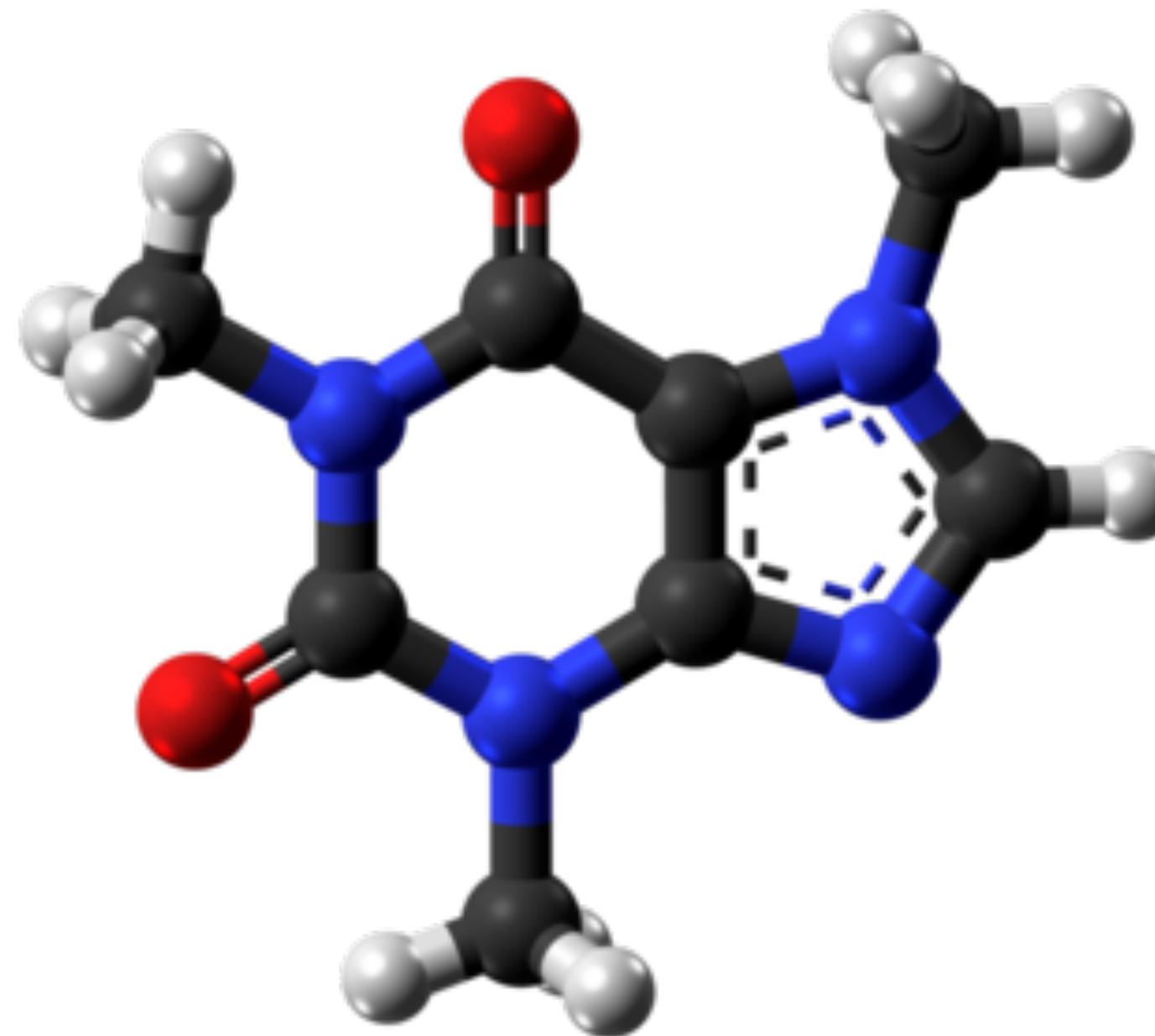


Small and organic molecule

- typically < 100 atoms
- carbon-hydrogen or carbon-carbon bonds
- modulates a biological process
- druglikeness (bioavailability, solubility, stability)

What is a (Small) Molecule

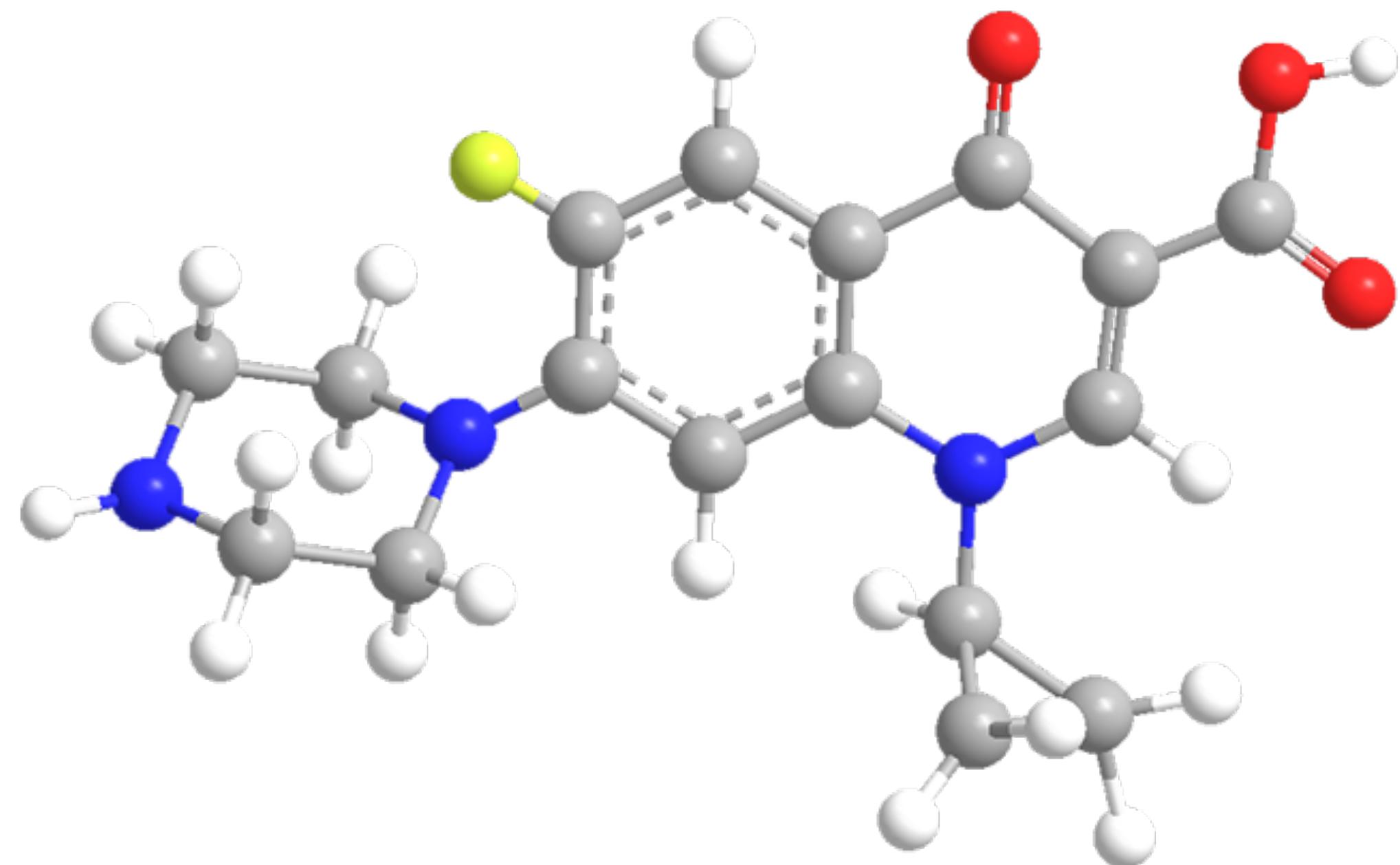
Molecule



Small and organic molecule

- typically < 100 atoms
- carbon-hydrogen or carbon-carbon bonds
- modulates a biological process
- druglikeness (bioavailability, solubility, stability)

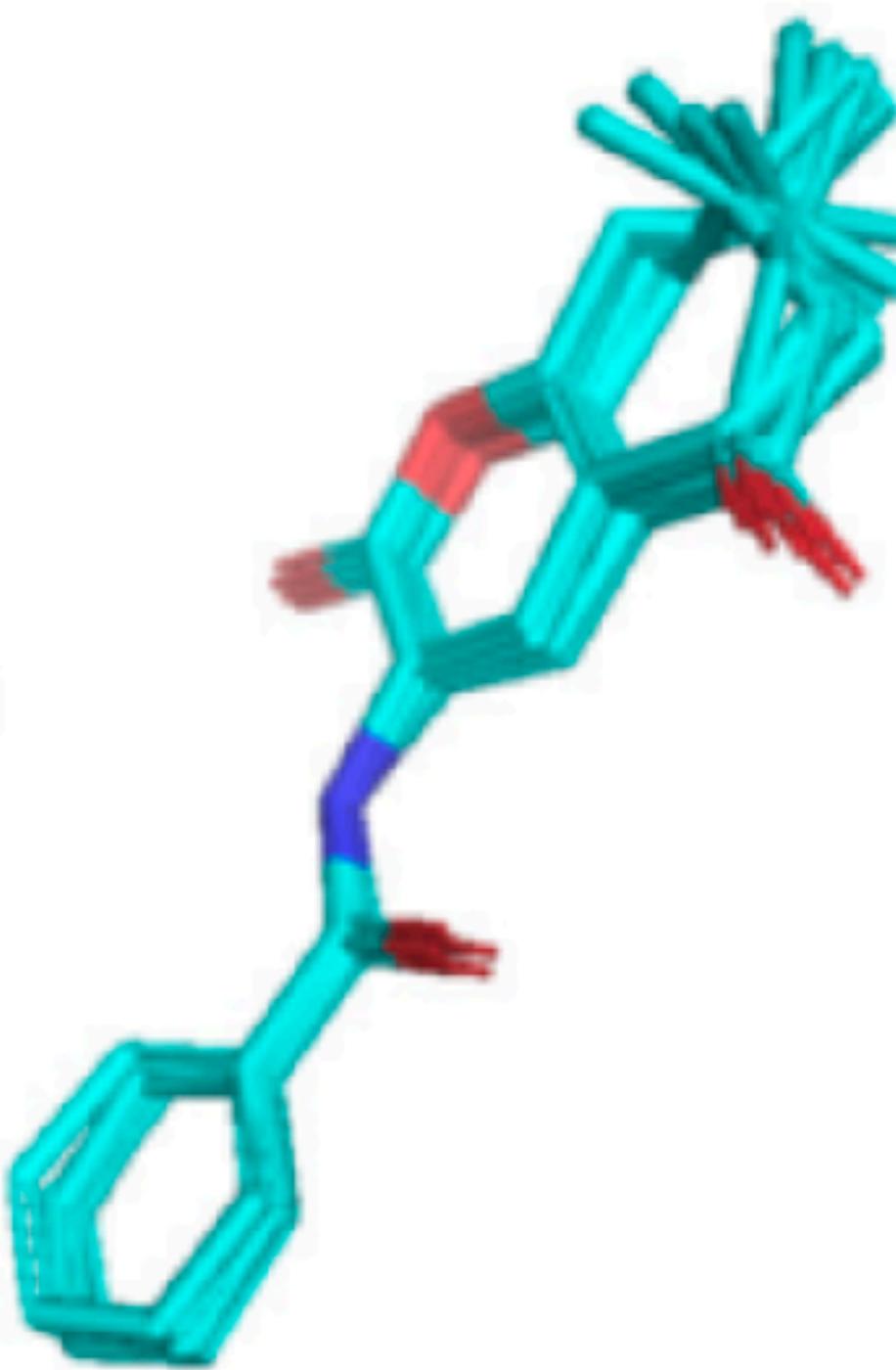
Drug-like Molecules



Most Common Elements

- Carbon (C)
- Hydrogen (H)
- Nitrogen (N)
- Oxygen (O)
- Fluorine (F)
- Sulfur (S)
- Chlorine (Cl)
- Phosphorus (P)

Conformers



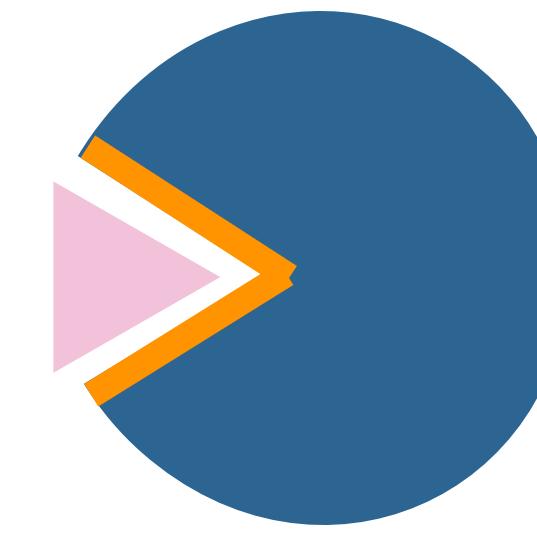
- Molecular graph does not determine 3d geometry
- Molecules are not static
- Different configurations due to single bond rotations

Better Informed Distance Geometry: Using What We Know To Improve Conformation Generation (Riniker et al.)

What is a Protein?

Most medical intervention are based on the fact that
a **small molecule** fits into a **large molecule**.

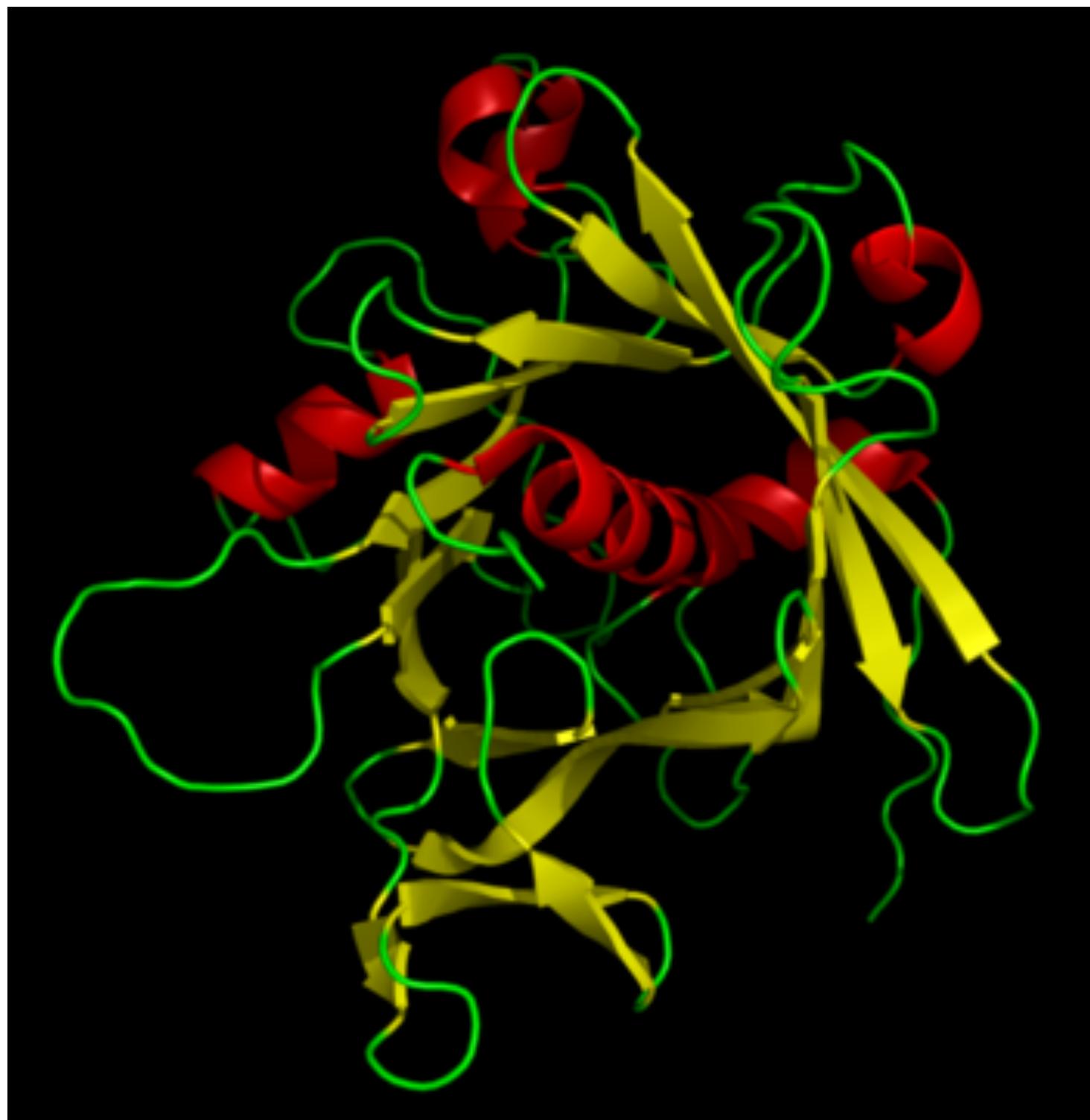
small molecule
drug
ligand
inhibitor



protein
enzyme
receptor
drug target

binding site
pocket

What is a Protein?



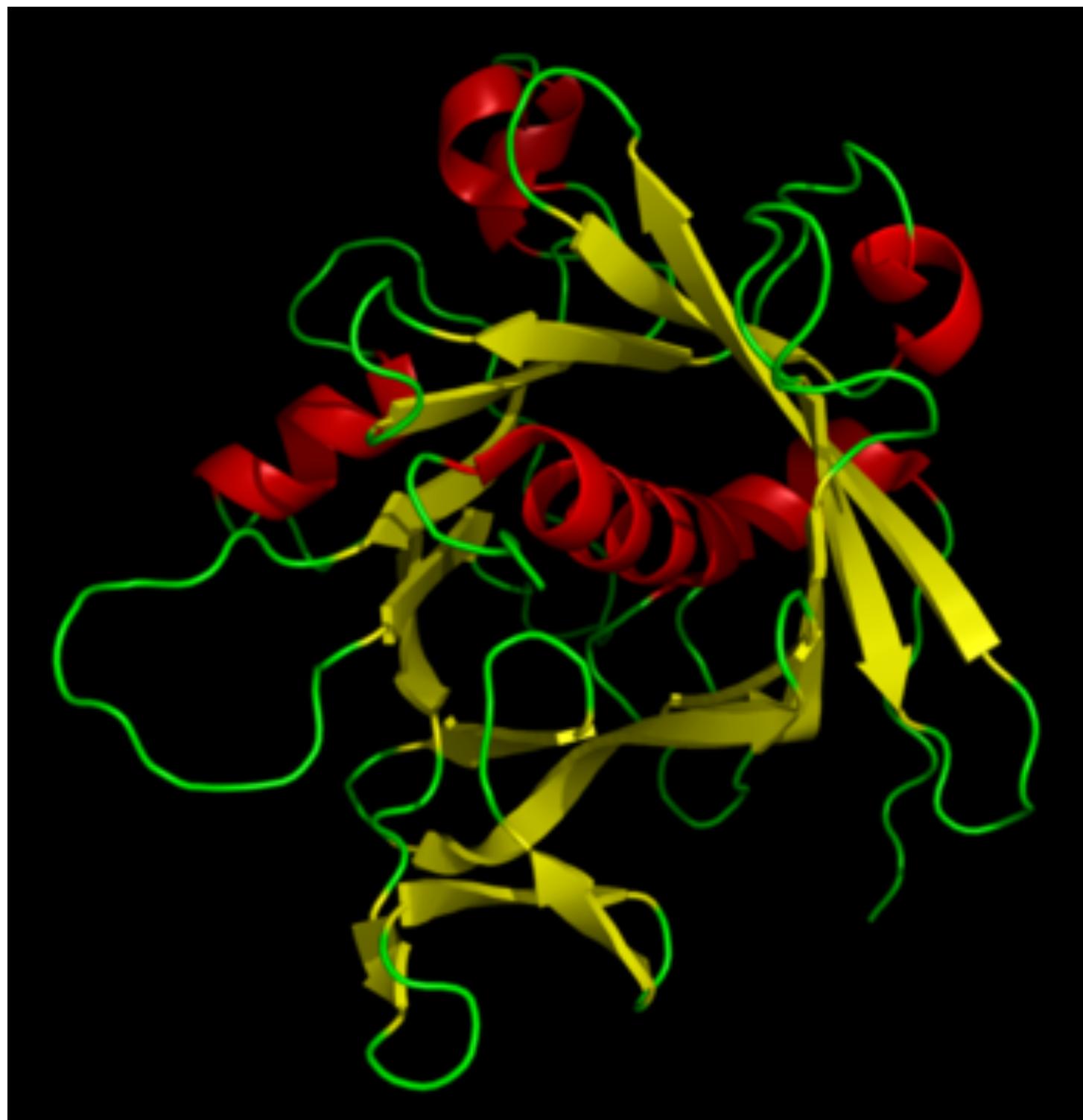
Protein

- large, complex molecules
- several thousand atoms
- chain of amino acids
- forms 3D structure in a highly non-trivial way

Druggable Target

- presence of a binding site
- minimize off-target effects
- modifiable activity:
 - inhibit or activate enzyme activity
 - activate or block receptor

What is a Protein?



Protein

- large, complex molecules
- several thousand atoms
- chain of amino acids
- forms 3D structure in a highly non-trivial way

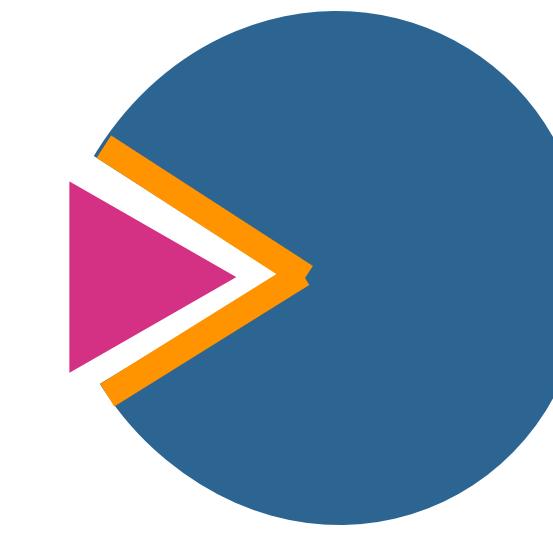
Druggable Target

- presence of a binding site
- minimize off-target effects
- modifiable activity:
 - inhibit or activate enzyme activity
 - activate or block receptor

How Drugs Work

Most medical intervention are based on the fact that
a **small molecule** fits into a **large molecule**.

small molecule
drug
ligand
inhibitor



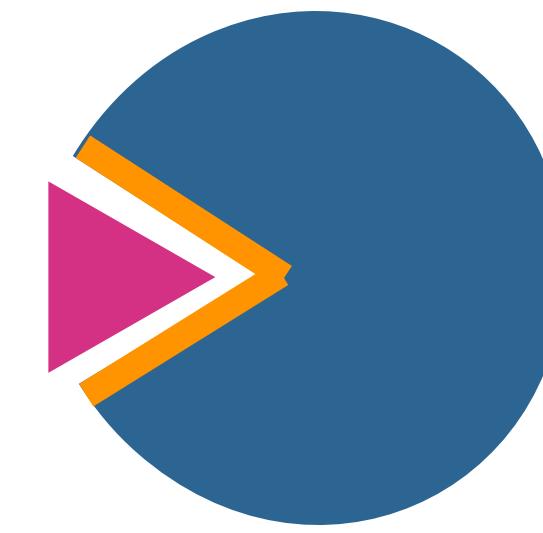
protein
enzyme
receptor
drug target

binding site
pocket

How Drugs Work

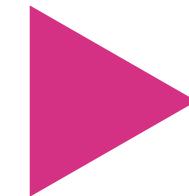
Most medical intervention are based on the fact that
a **small molecule** fits into a **large molecule**.

small molecule
drug
ligand
inhibitor

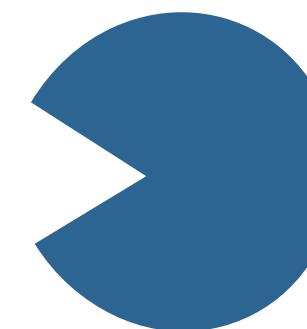


protein
enzyme
receptor
drug target

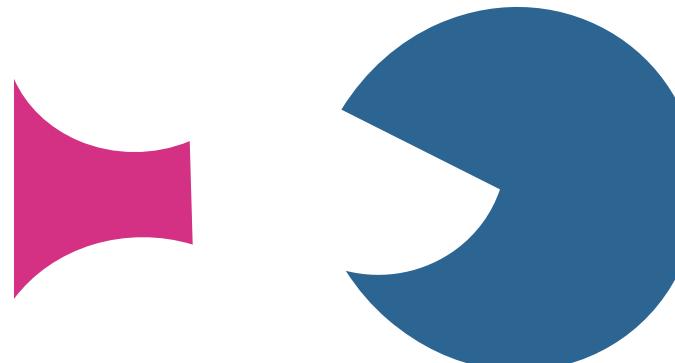
binding site
pocket



Design small molecule



Design large molecule

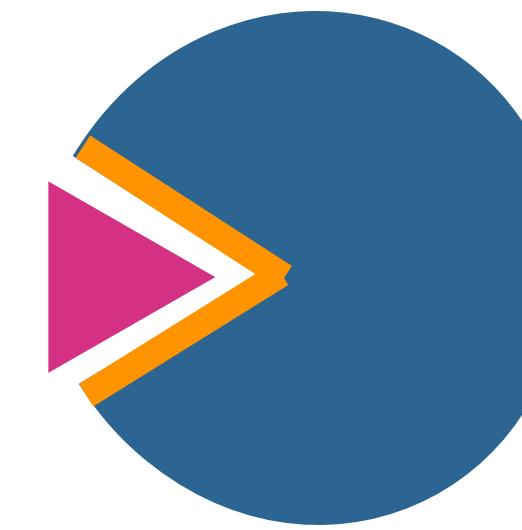


Determine if small molecule
fits into large molecule

How Drugs Work

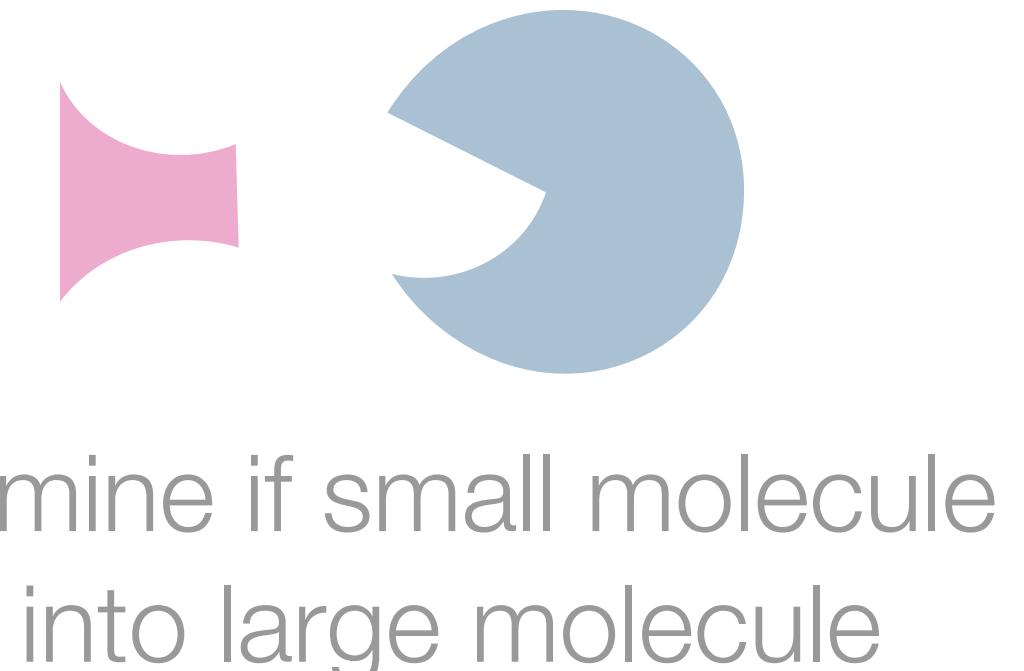
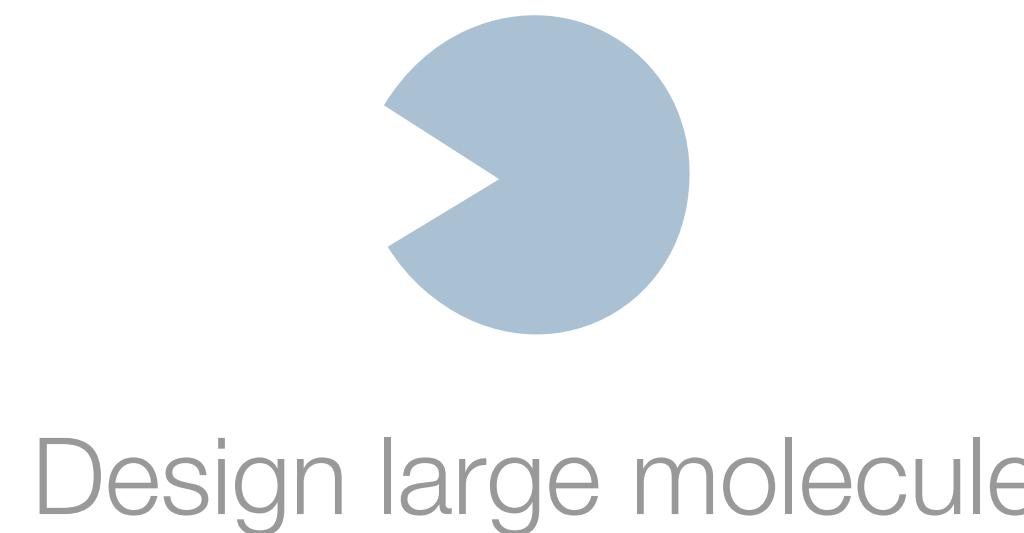
Most medical intervention are based on the fact that
a **small molecule** fits into a **large molecule**.

small molecule
drug
ligand
inhibitor

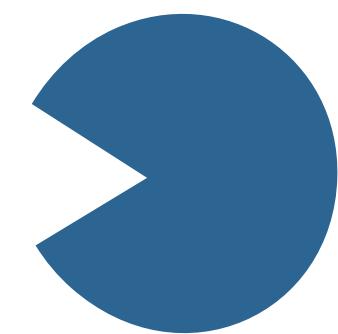


protein
enzyme
receptor
drug target

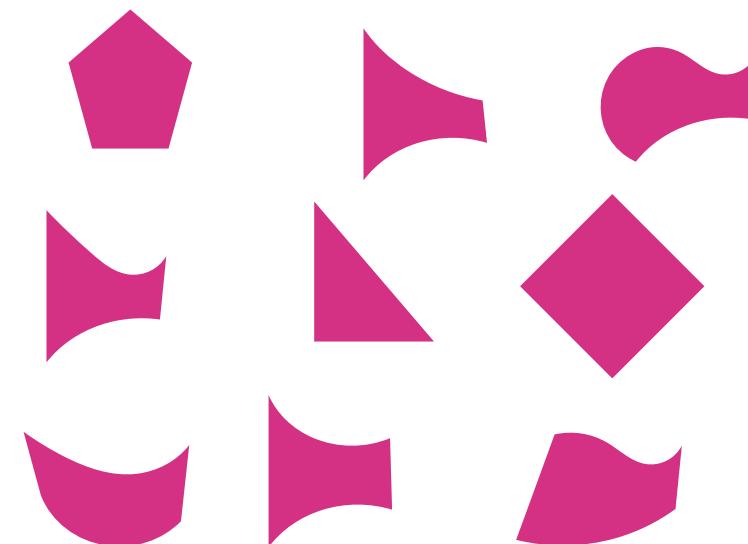
binding site
pocket



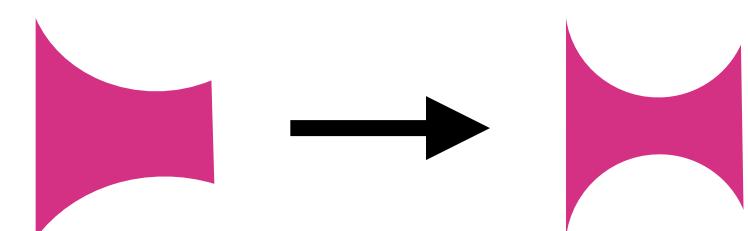
Pipeline



Identifying a **protein** that is involved in a disease process and can be **targeted** by a **drug** to produce a therapeutic effect.

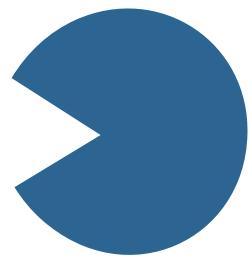


Use algorithms (docking + filtering) to **screen** virtual libraries of **small molecules** and predict which ones are likely to bind to the target.

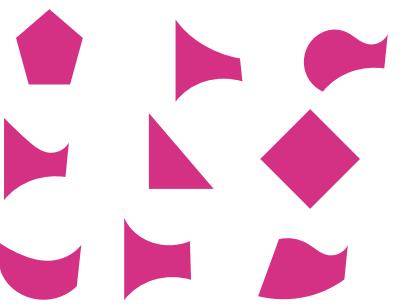


Refine **hits** to optimize their properties for use as drugs.

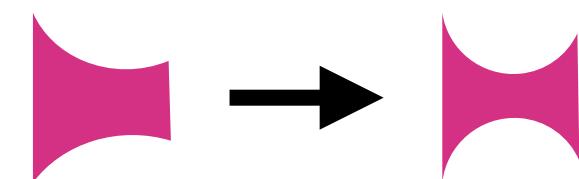
Pipeline



Identifying a **protein** that is involved in a disease process and can be **targeted** by a **drug** to produce a therapeutic effect.



Use algorithms (docking + filtering) to **screen** virtual libraries of **small molecules** and predict which ones are likely to bind to the target.



Refine **hits** to optimize their properties for use as drugs.

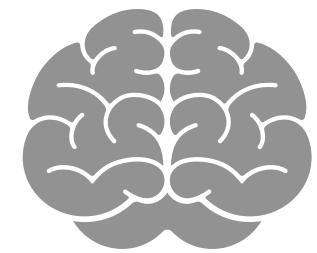
10^{60}

different drug-like molecules

Vast chemical space

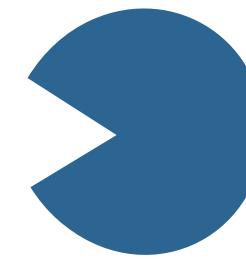


Expensive

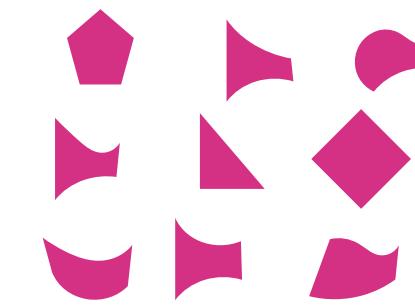


Human bias

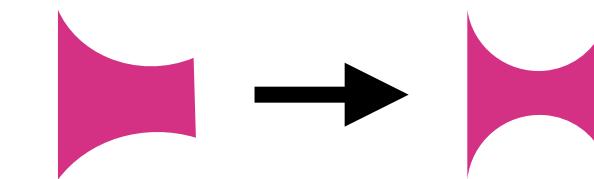
AI to the Rescue



Identifying a **protein** that is involved in a disease process and can be **targeted** by a **drug** to produce a therapeutic effect.



Use algorithms (docking + filtering) to **screen** virtual libraries of **small molecules** and predict which ones are likely to bind to the target.



Refine **hits** to optimize their properties for use as drugs.



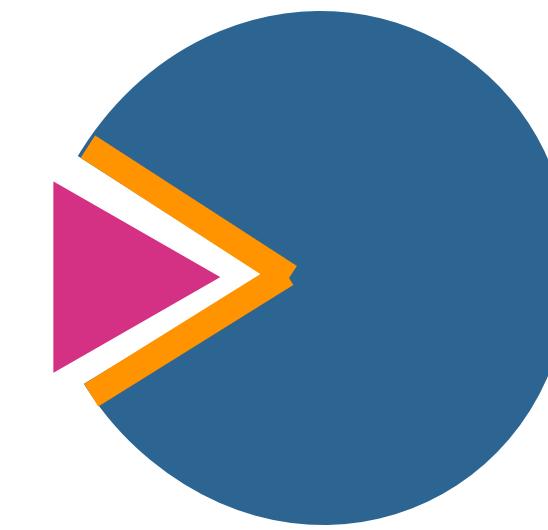
Compound generation

Generate novel chemical structures that are optimized for binding to the target.

How Drugs Work

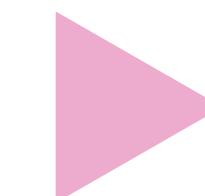
Most medical intervention are based on the fact that
a **small molecule** fits into a **large molecule**.

small molecule
drug
ligand
inhibitor



protein
enzyme
receptor
drug target

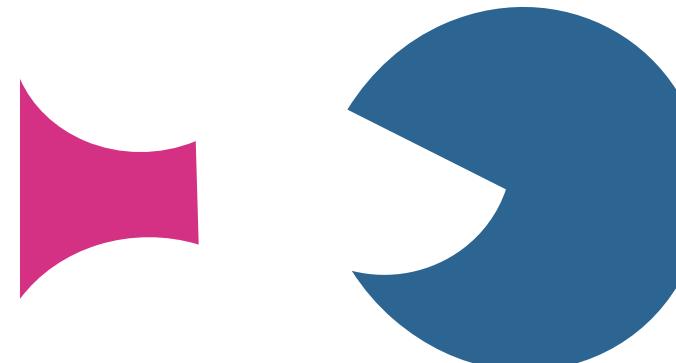
binding site
pocket



Design small molecule

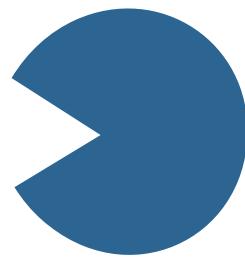


Design large molecule



Determine if small molecule
fits into large molecule

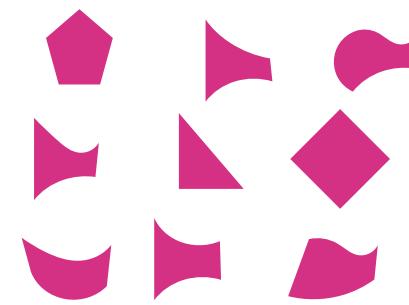
AI to the Rescue



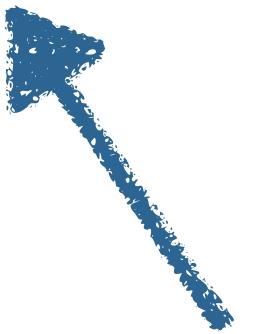
Identifying a **protein** that is involved in a disease process and can be **targeted** by a **drug** to produce a therapeutic effect.



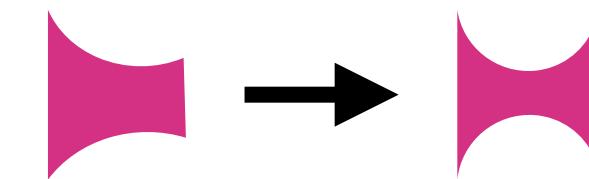
Protein structure and binding site prediction



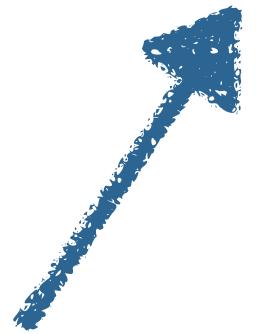
Use algorithms (docking + filtering) to **screen** virtual libraries of **small molecules** and predict which ones are likely to bind to the target.

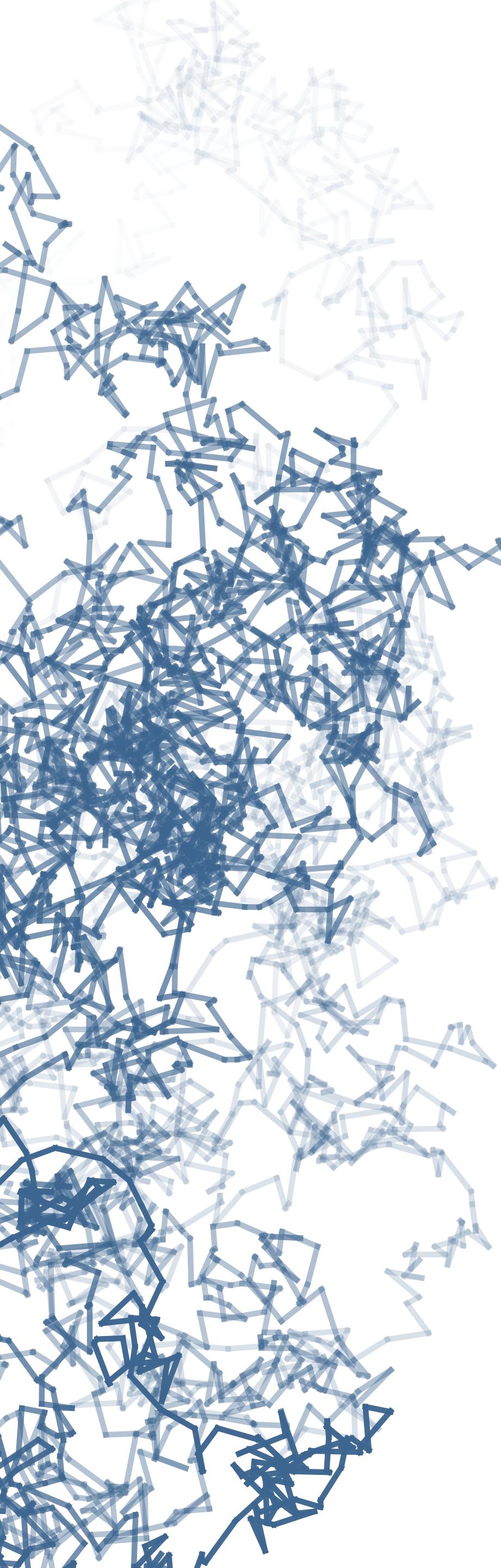


Binding affinity prediction



Refine **hits** to optimize their properties for use as drugs.

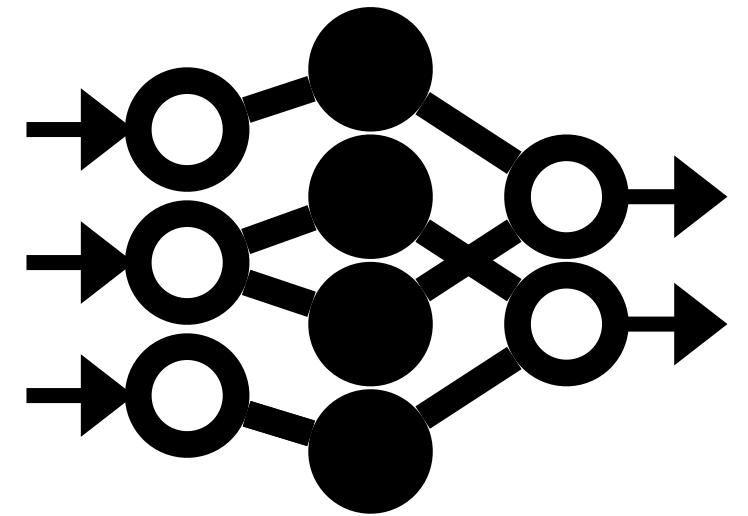




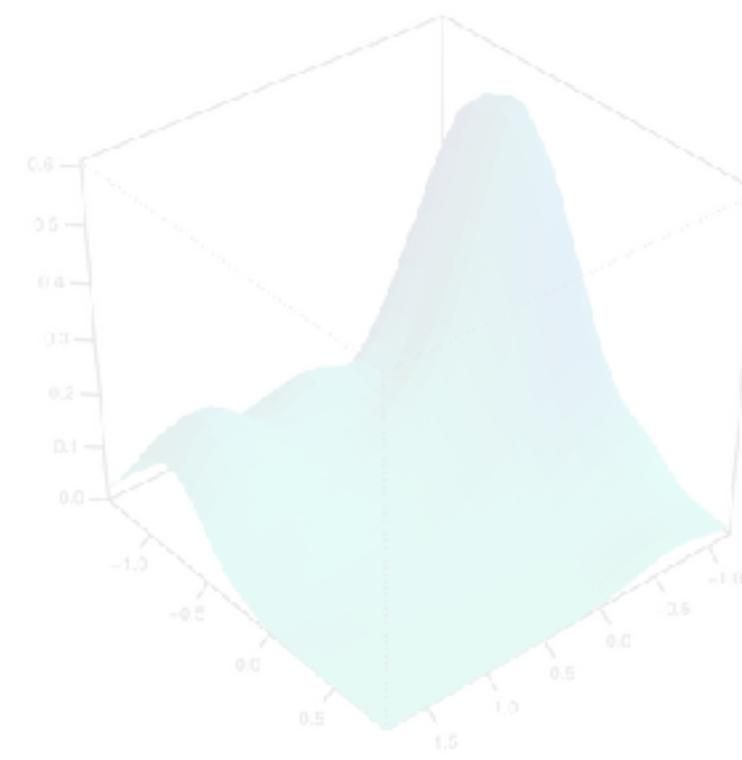
Part II

DIFFUSION MODELS

Generative Deep Learning



Neural network with millions of trainable parameters (weights)



Weights encode a probability distribution.

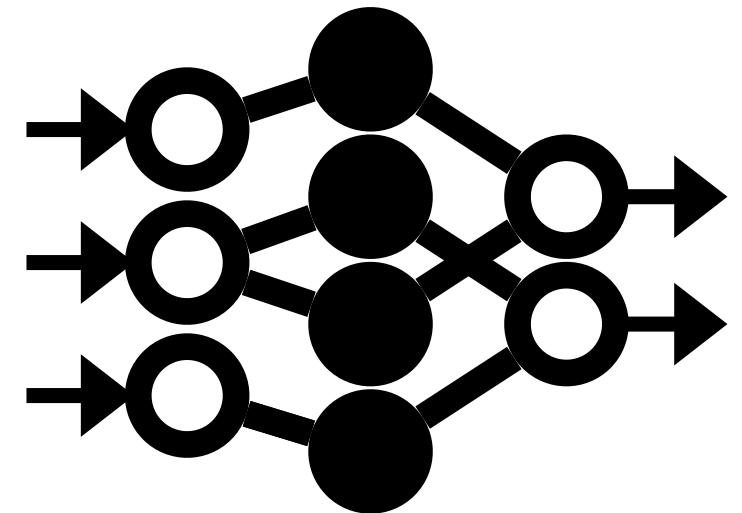


Training adjusts the weights based on training data

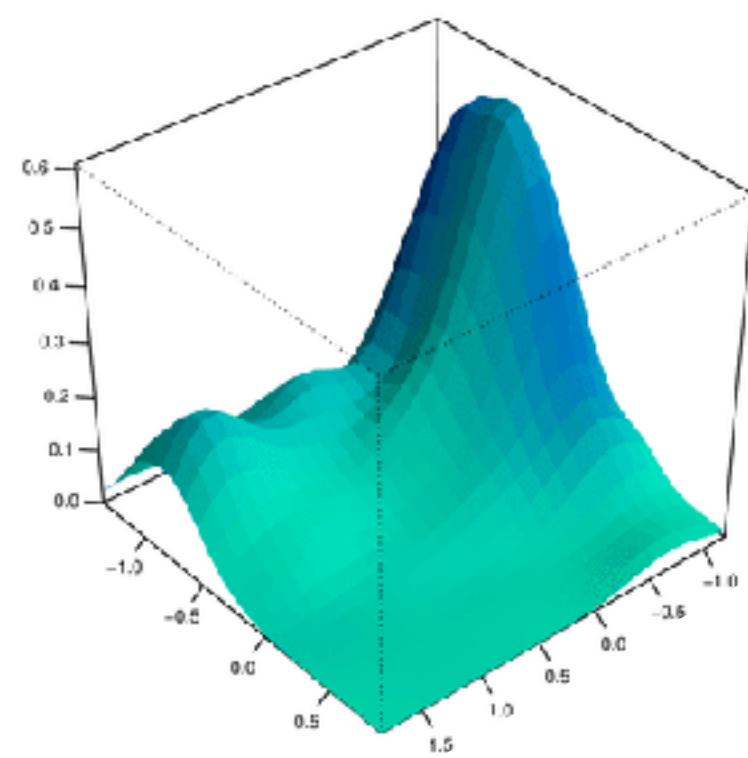
3

Sampling methods generate new data points.

Generative Deep Learning



Neural network with millions of trainable parameters (weights)



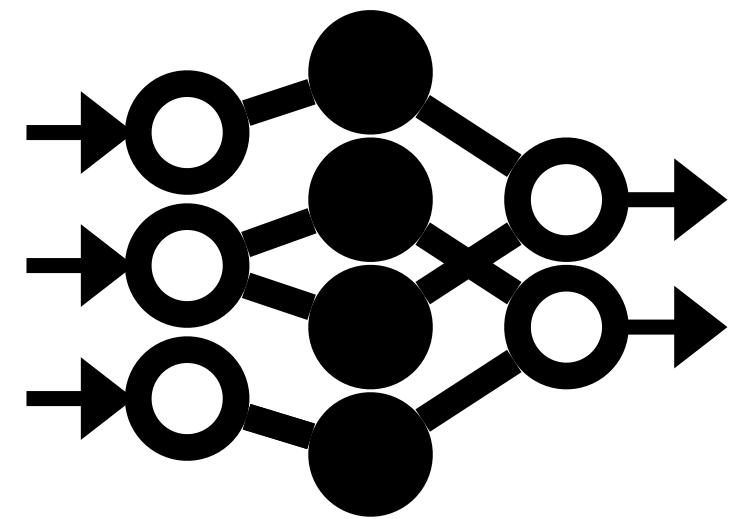
Weights encode a probability distribution.



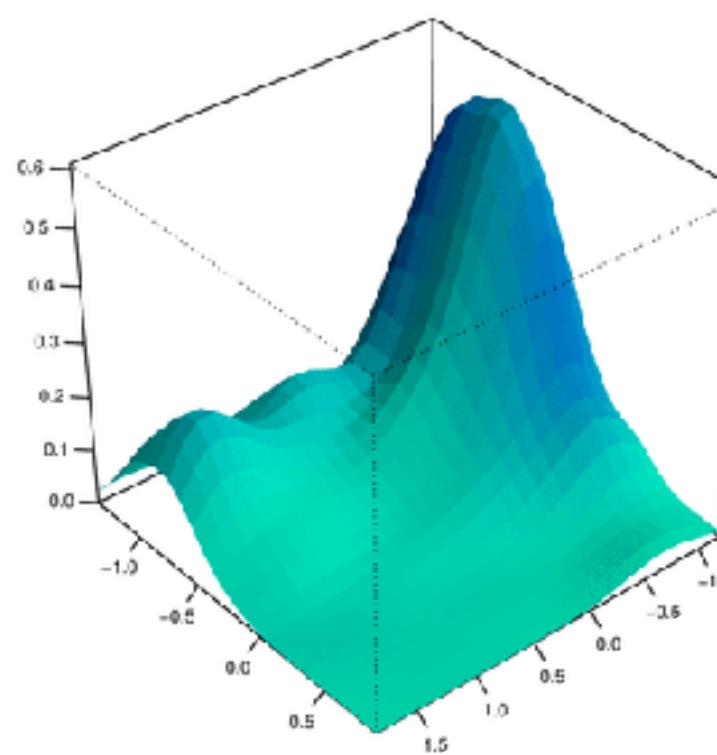
Training adjusts the weights based on training data

Sampling methods generate new data points.

Generative Deep Learning



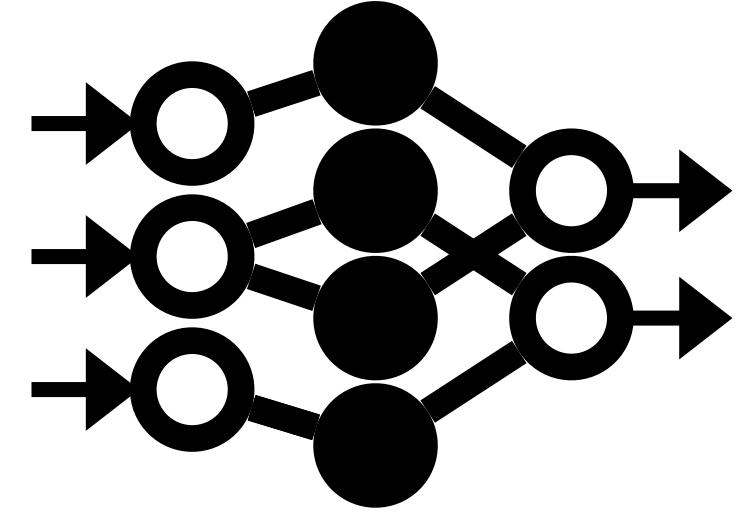
Neural network with millions of trainable parameters (weights)



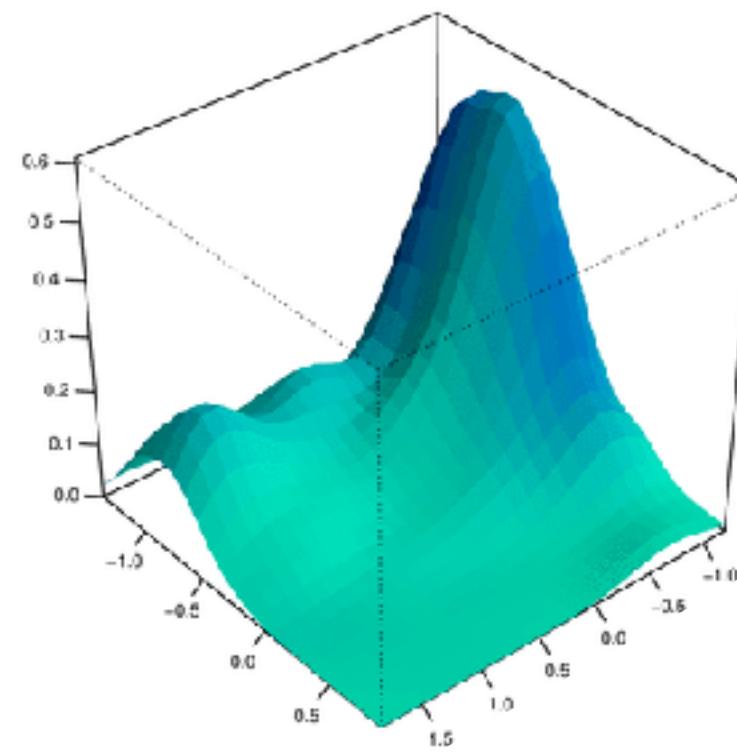
Weights encode a probability distribution.

Training adjusts the weights
based on training data

Generative Deep Learning



Neural network with millions of trainable parameters (weights)



Weights encode a probability distribution.

0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7 7 7 7 7 7 7
8 8 8 8 8 8 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9 9 9 9 9 9

3

Training adjusts the weights based on training data

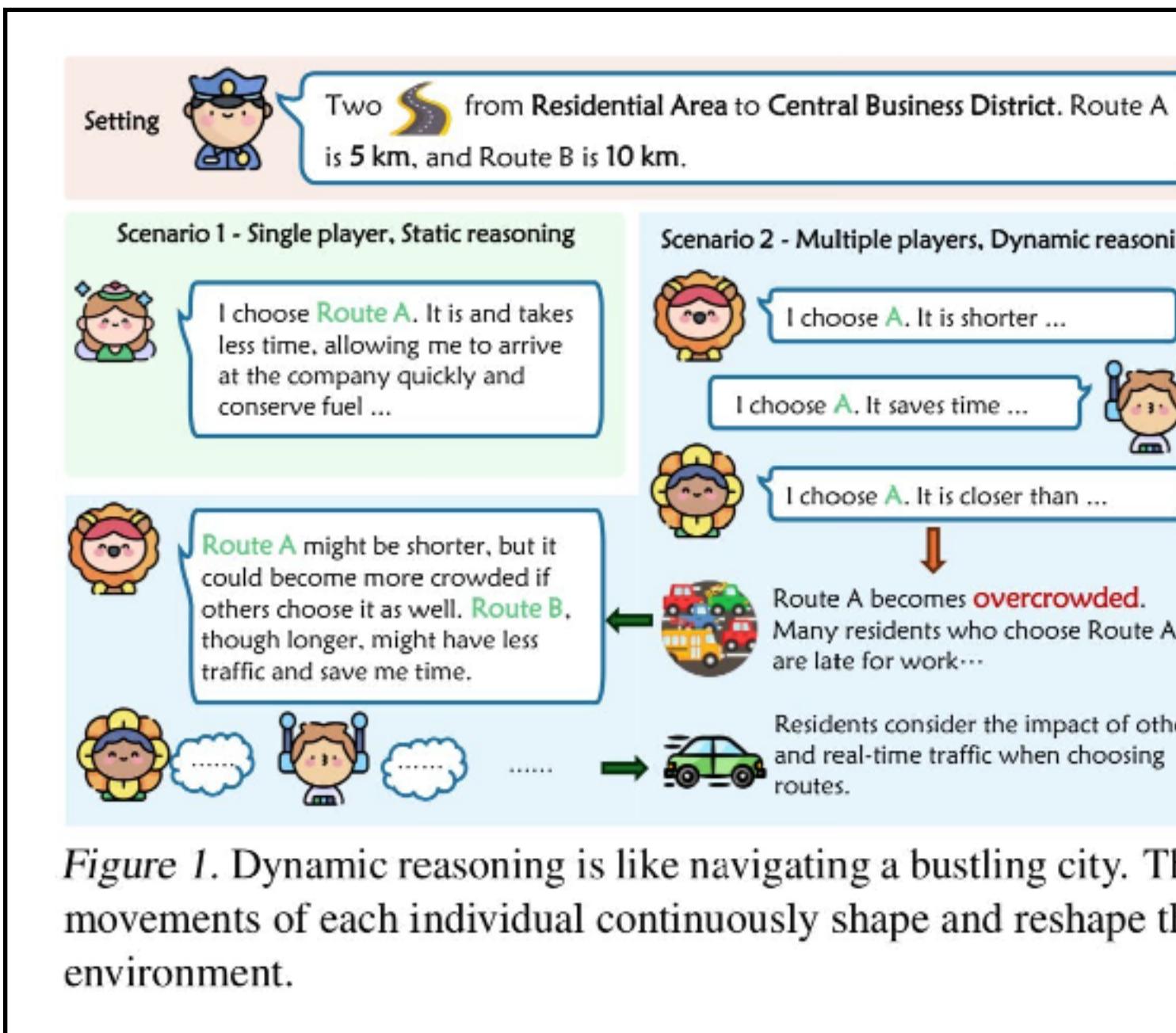
Sampling methods generate new data points.

Probabilistic Diffusion



Nucleus ☕ ✅ M
@EsotericCofe

hardest LLM paper vs easiest diffusion paper



twitter.com/EsotericCofe/status/1777280241884377474

$$\begin{aligned} L\text{-DPO-Diffusion}(\theta) \leq & -\mathbb{E}_{(\mathbf{x}_0^w, \mathbf{x}_0^l) \sim \mathcal{D}, t \sim \mathcal{U}(0, T),} \\ & \mathbf{x}_{t-1, t}^w \sim p_\theta(\mathbf{x}_{t-1, t}^w | \mathbf{x}_0^w), \\ & \mathbf{x}_{t-1, t}^l \sim p_\theta(\mathbf{x}_{t-1, t}^l | \mathbf{x}_0^l)} \\ & \log \sigma \left(\beta T \log \frac{p_\theta(\mathbf{x}_{t-1}^w | \mathbf{x}_t^w)}{p_{\text{ref}}(\mathbf{x}_{t-1}^w | \mathbf{x}_t^w)} - \beta T \log \frac{p_\theta(\mathbf{x}_{t-1}^l | \mathbf{x}_t^l)}{p_{\text{ref}}(\mathbf{x}_{t-1}^l | \mathbf{x}_t^l)} \right) \end{aligned} \quad (12)$$

Efficient training via gradient descent is now possible. However, sampling from reverse joint $p_\theta(\mathbf{x}_{t-1}, \mathbf{x}_t | \mathbf{x}_0, c)$ is still intractable and r of Eq. (9) has an expectation over $p_\theta(\mathbf{x}_{1:T} | \mathbf{x}_0)$. So we approximate the reverse process $p_\theta(\mathbf{x}_{1:T} | \mathbf{x}_0)$ with the forward $q(\mathbf{x}_{1:T} | \mathbf{x}_0)$ (an alternative scheme in Supp. S2). With some algebra, this yields:

$$\begin{aligned} L(\theta) = & -\mathbb{E}_{(\mathbf{x}_0^w, \mathbf{x}_0^l) \sim \mathcal{D}, t \sim \mathcal{U}(0, T), \mathbf{x}_t^w \sim q(\mathbf{x}_t^w | \mathbf{x}_0^w), \mathbf{x}_t^l \sim q(\mathbf{x}_t^l | \mathbf{x}_0^l)} \\ & \log \sigma(-\beta T) \\ & + \mathbb{D}_{\text{KL}}(q(\mathbf{x}_{t-1}^w | \mathbf{x}_{0,t}^w) \| p_\theta(\mathbf{x}_{t-1}^w | \mathbf{x}_t^w)) \\ & - \mathbb{D}_{\text{KL}}(q(\mathbf{x}_{t-1}^l | \mathbf{x}_{0,t}^l) \| p_{\text{ref}}(\mathbf{x}_{t-1}^l | \mathbf{x}_t^l)) \\ & - \mathbb{D}_{\text{KL}}(q(\mathbf{x}_{t-1}^l | \mathbf{x}_{0,t}^l) \| p_\theta(\mathbf{x}_{t-1}^l | \mathbf{x}_t^l)) \\ & + \mathbb{D}_{\text{KL}}(q(\mathbf{x}_{t-1}^l | \mathbf{x}_{0,t}^l) \| p_{\text{ref}}(\mathbf{x}_{t-1}^l | \mathbf{x}_t^l)). \end{aligned} \quad (13)$$

Using Eq. (1) and algebra, the above loss simplifies to:

$$\begin{aligned} L(\theta) = & -\mathbb{E}_{(\mathbf{x}_0^w, \mathbf{x}_0^l) \sim \mathcal{D}, t \sim \mathcal{U}(0, T), \mathbf{x}_t^w \sim q(\mathbf{x}_t^w | \mathbf{x}_0^w), \mathbf{x}_t^l \sim q(\mathbf{x}_t^l | \mathbf{x}_0^l)} \\ & \log \sigma(-\beta T \omega(\lambda_t)) \\ & \| \boldsymbol{\epsilon}^w - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t^w, t) \|_2^2 - \| \boldsymbol{\epsilon}^w - \boldsymbol{\epsilon}_{\text{ref}}(\mathbf{x}_t^w, t) \|_2^2 \\ & - (\| \boldsymbol{\epsilon}^l - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t^l, t) \|_2^2 - \| \boldsymbol{\epsilon}^l - \boldsymbol{\epsilon}_{\text{ref}}(\mathbf{x}_t^l, t) \|_2^2) \end{aligned} \quad (14)$$

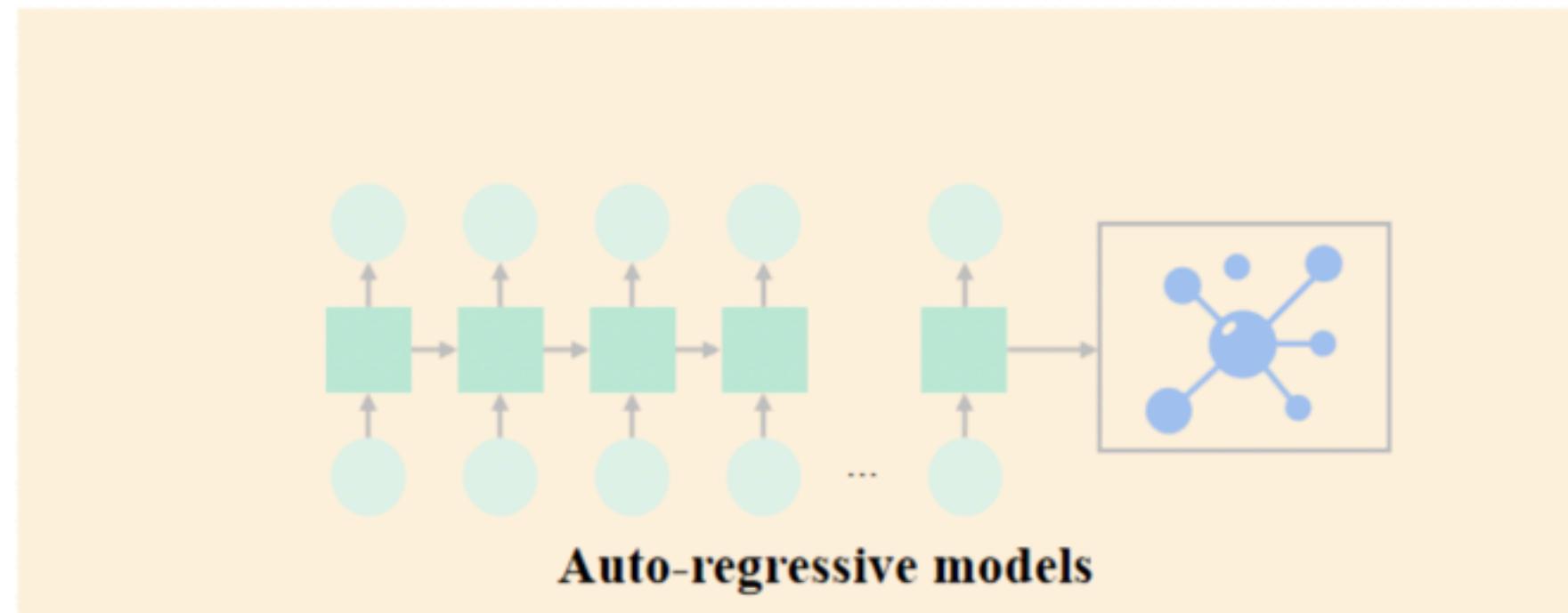
where $\mathbf{x}_t^* = \alpha_t \mathbf{x}_0^* + \sigma_t \boldsymbol{\epsilon}^*$, $\boldsymbol{\epsilon}^* \sim \mathcal{N}(0, I)$ is a draw from $q(\mathbf{x}_t^* | \mathbf{x}_0^*)$ (Eq. (2)). $\lambda_t = \alpha_t^2 / \sigma_t^2$ is the signal-to-noise ratio,

Probabilistic Diffusion

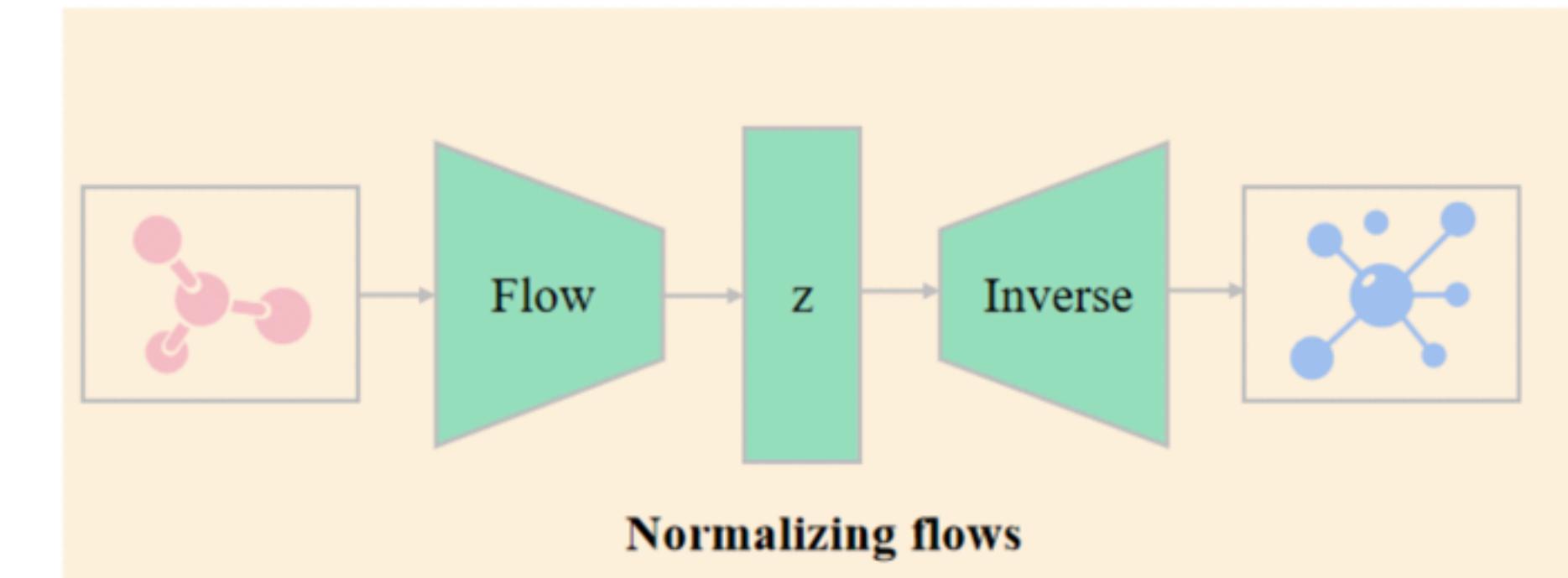


- historically rooted in non-equilibrium thermodynamics
- based on the theory of stochastic processes
- many different flavors and variations

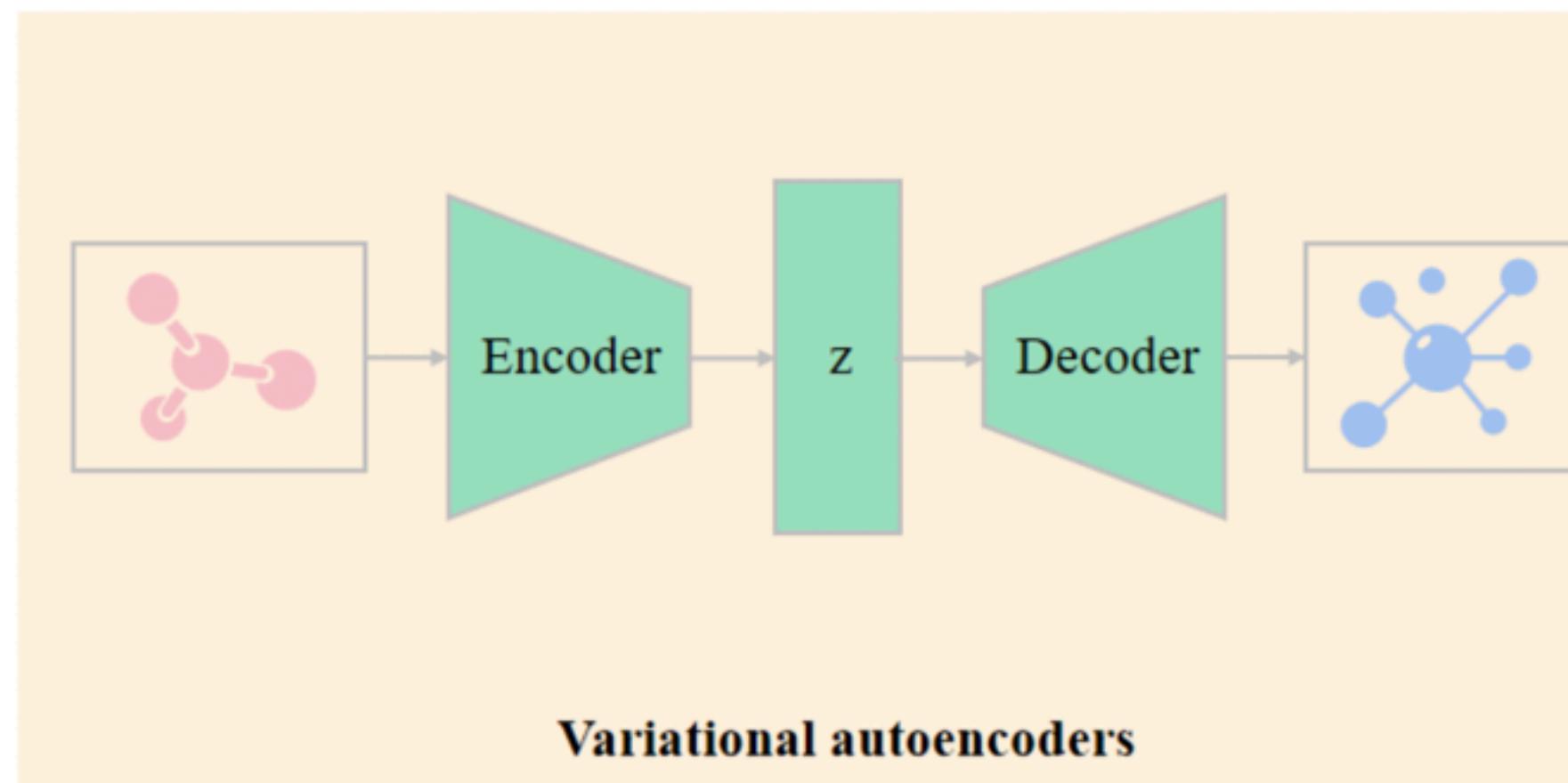
Probabilistic Diffusion



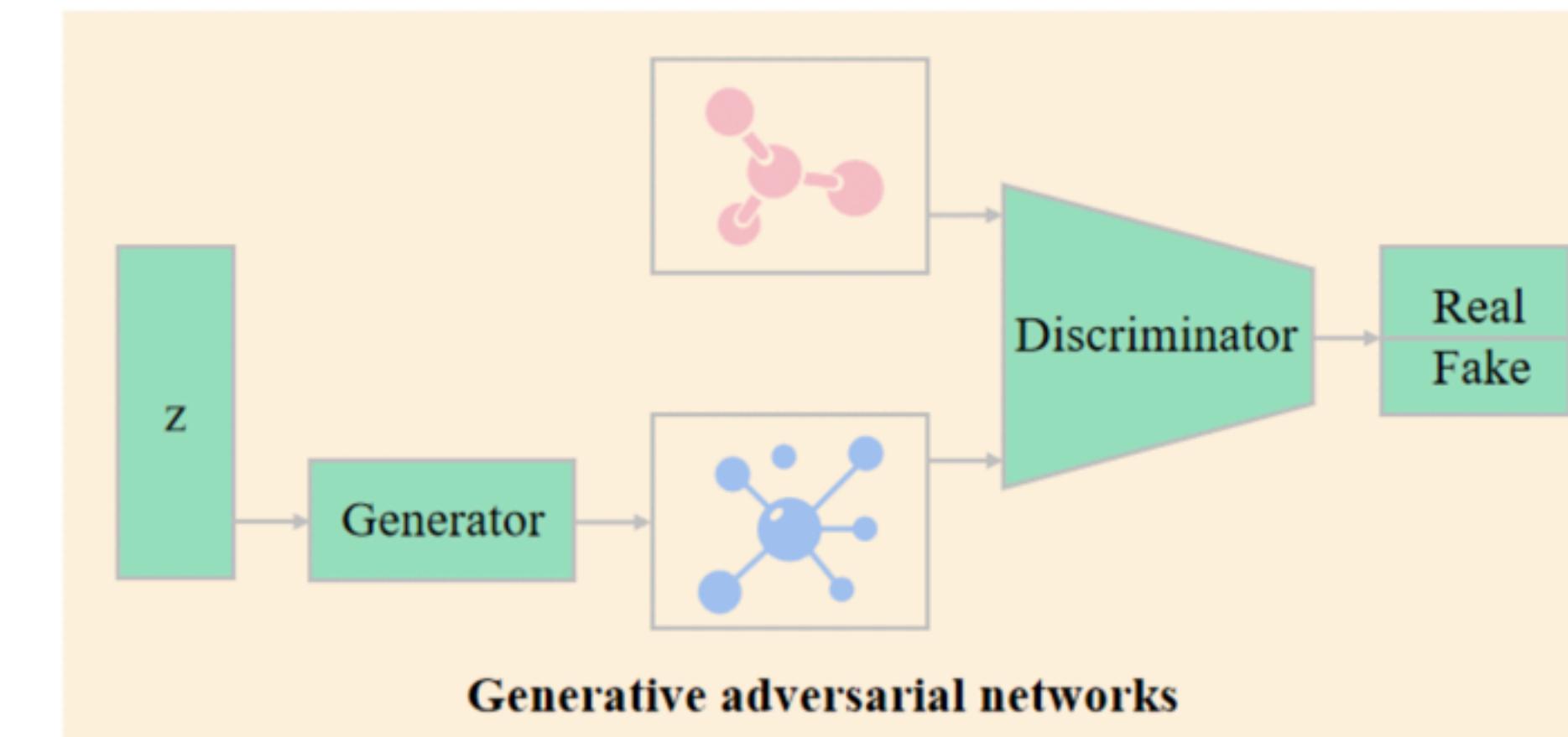
Auto-regressive models



Normalizing flows



Variational autoencoders



Generative adversarial networks

A Survey on Graph Diffusion Models: GenerativeAI in Science for Molecule, Protein and Material (Zhang et al.)

Probabilistic Diffusion

1.

Non-trainable part that **removes** information
(stochastically or deterministically)

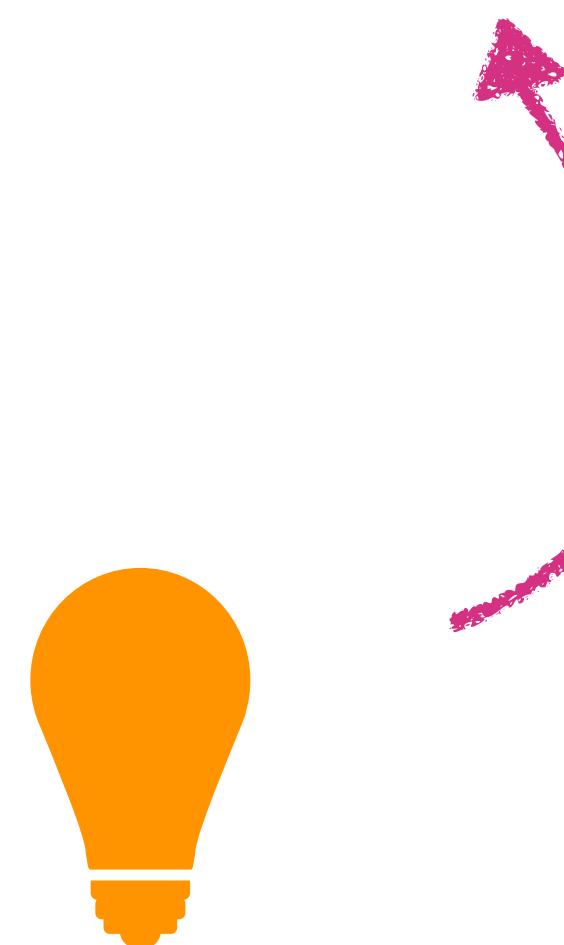
2.

Trainable part that **reconstructs** information
(typically with a stochastic component)

Probabilistic Diffusion

1.

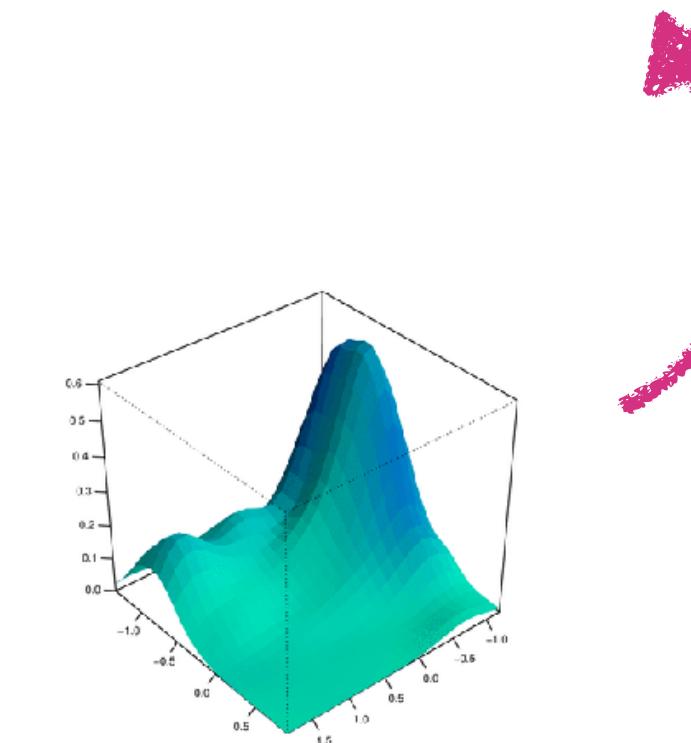
Non-trainable part that **removes** information
(stochastically or deterministically)



This is a trick to guide the training.

2.

Trainable part that **reconstructs** information
(typically with a stochastic component)



This is where we learn the distribution
(implicitly represented by NN weights).

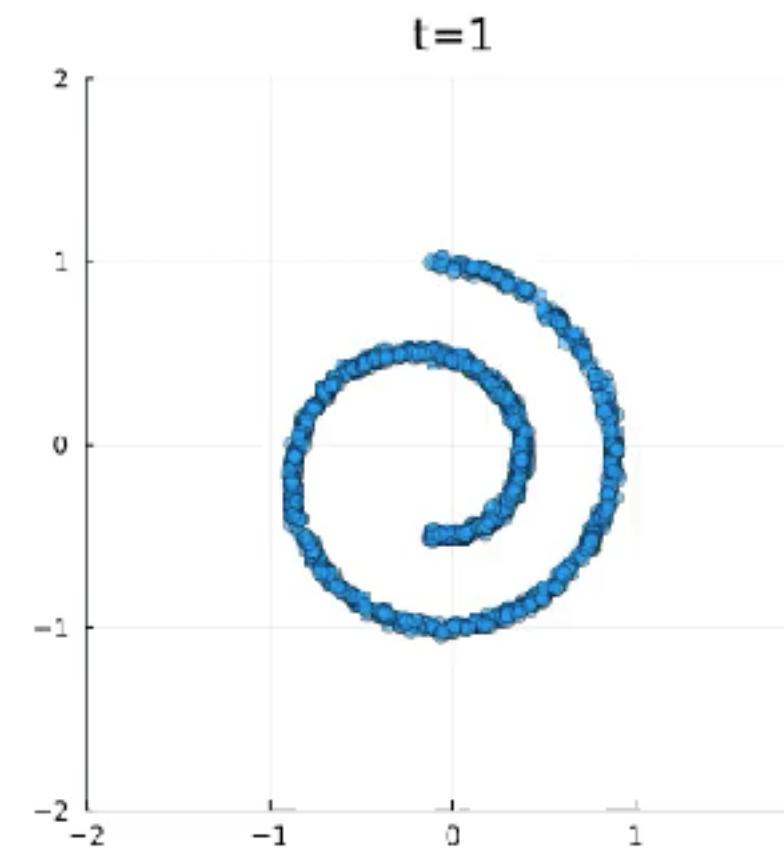
Probabilistic Diffusion

1.

Non-trainable part that **removes** information
(stochastically or deterministically)

2.

Trainable part that **reconstructs** information
(typically with a stochastic component)

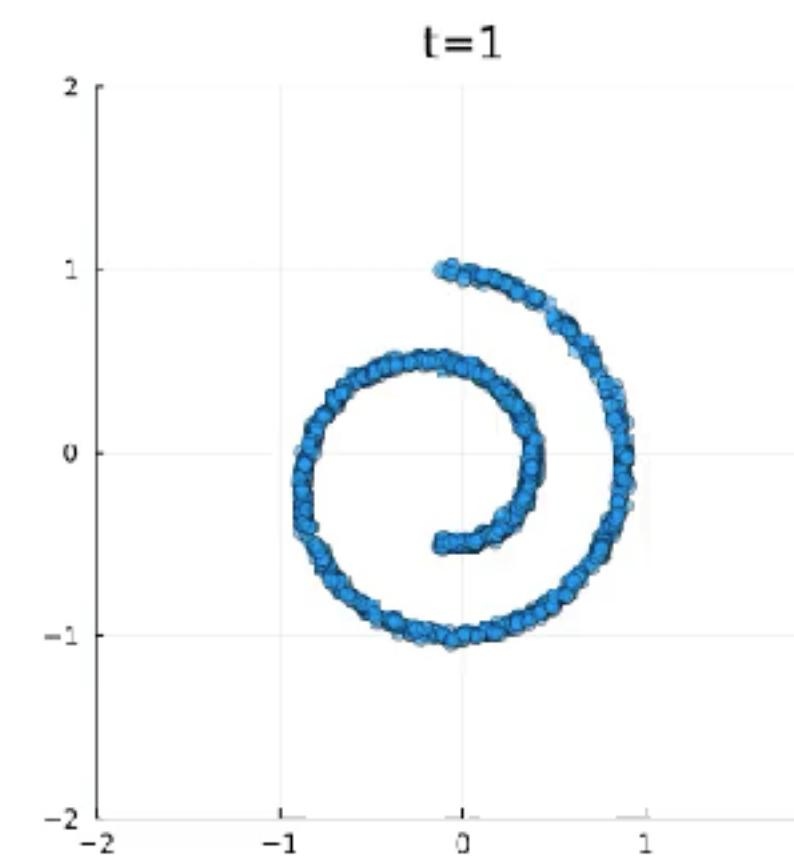


Obvious way: Degrade a sample by **adding Gaussian noise** to it.

Probabilistic Diffusion

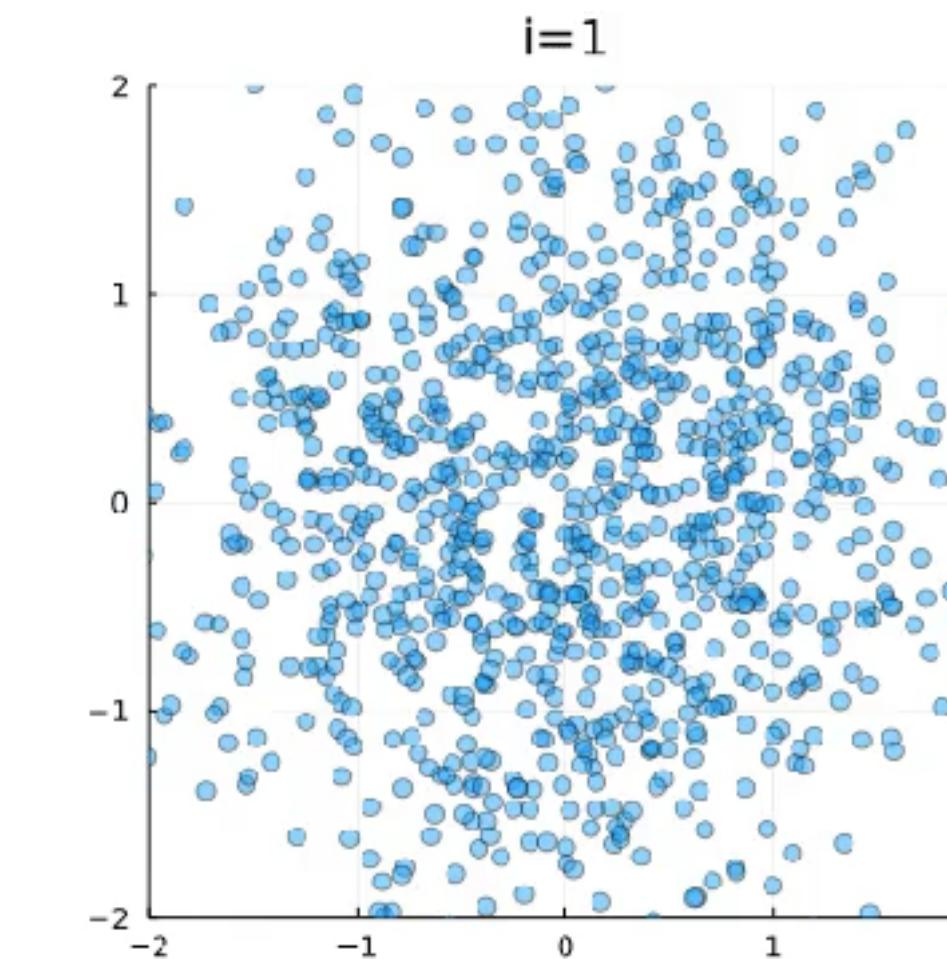
1.

Non-trainable part that **removes** information
(stochastically or deterministically)



2.

Trainable part that **reconstructs** information
(typically with a stochastic component)



Obvious way: Degrade a sample by **adding Gaussian noise** to it.

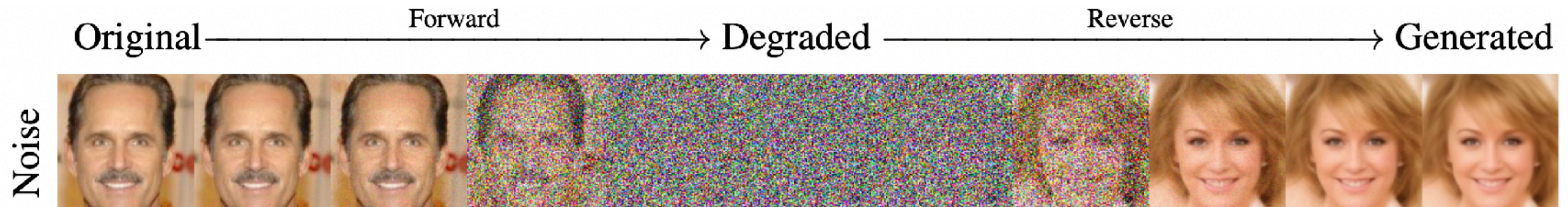
Probabilistic Diffusion

1.

Non-trainable part that **removes** information
(stochastically or deterministically)

2.

Trainable part that **reconstructs** information
(typically with a stochastic component)



Obvious way: Degrade a sample by **adding Gaussian noise** to it.

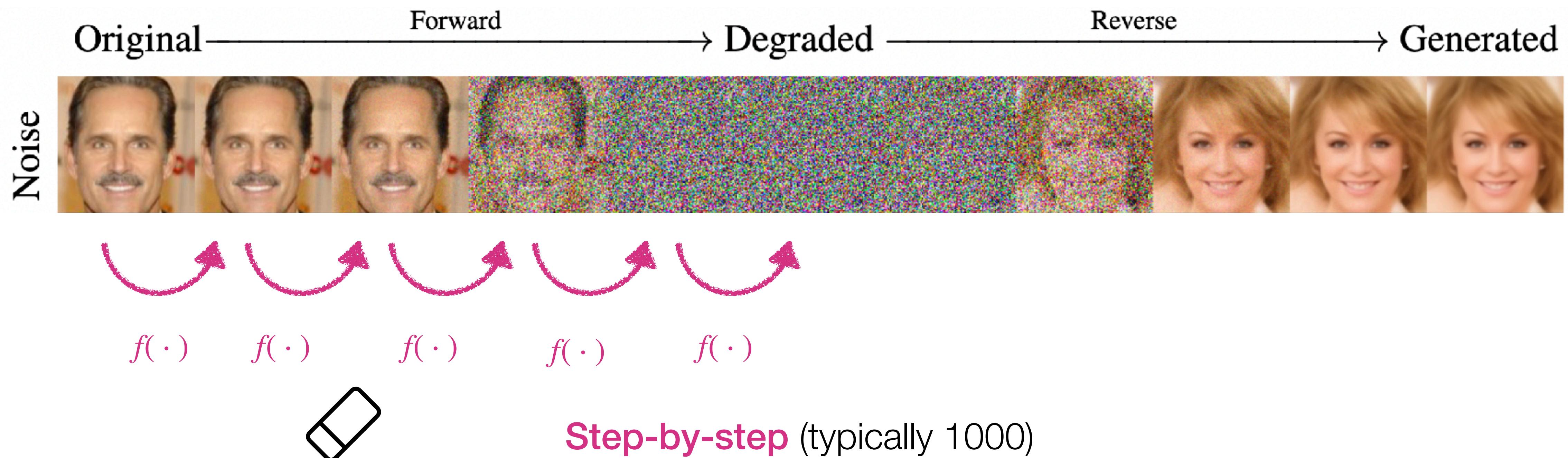
Probabilistic Diffusion

1.

Non-trainable part that **removes** information
(stochastically or deterministically)

2.

Trainable part that **reconstructs** information
(typically with a stochastic component)



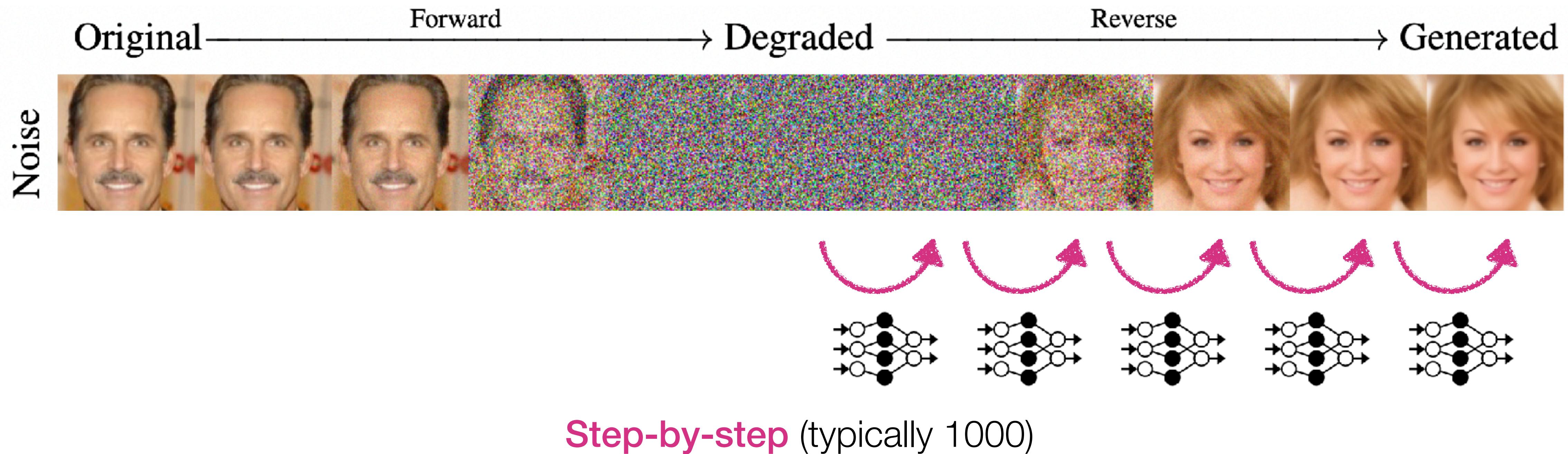
Probabilistic Diffusion

1.

Non-trainable part that **removes** information
(stochastically or deterministically)

2.

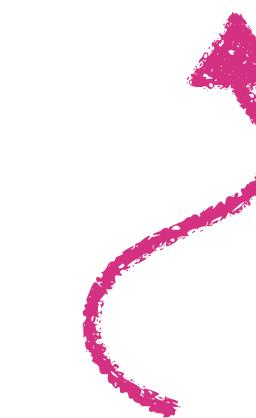
Trainable part that **reconstructs** information
(typically with a stochastic component)



Probabilistic Diffusion

1.

Non-trainable part that **removes** information
(stochastically or deterministically)



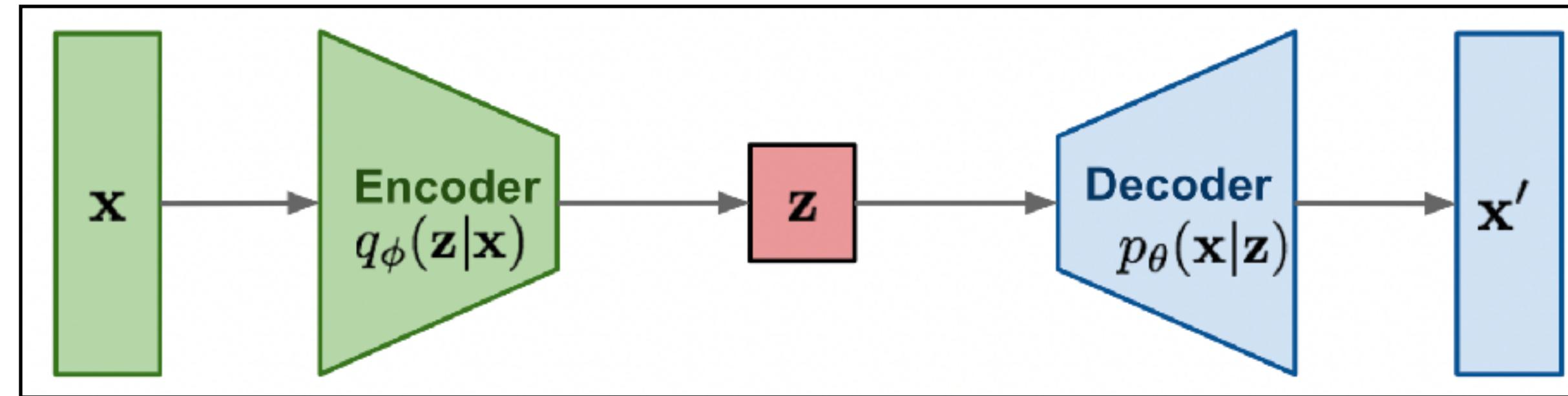
2.

Trainable part that **reconstructs** information
(typically with a stochastic component)

“What if I make the forward process **trainable**, too?”

“You get a very expensive **autoencoder**.“

Probabilistic Diffusion



“What if I make the forward process **trainable**, too?”

“You get a very expensive **autoencoder**.“

Probabilistic Diffusion

1.

Non-trainable part that **removes** information
(stochastically or deterministically)

2.

Trainable part that **reconstructs** information
(typically with a stochastic component)



“What if I make the forward process trainable and invertible and define the reverse process as the **inverted forward process**?“

“You get a normalizing flows!“

Probabilistic Diffusion

1.

Non-trainable part that **removes** information
(stochastically or deterministically)

2.

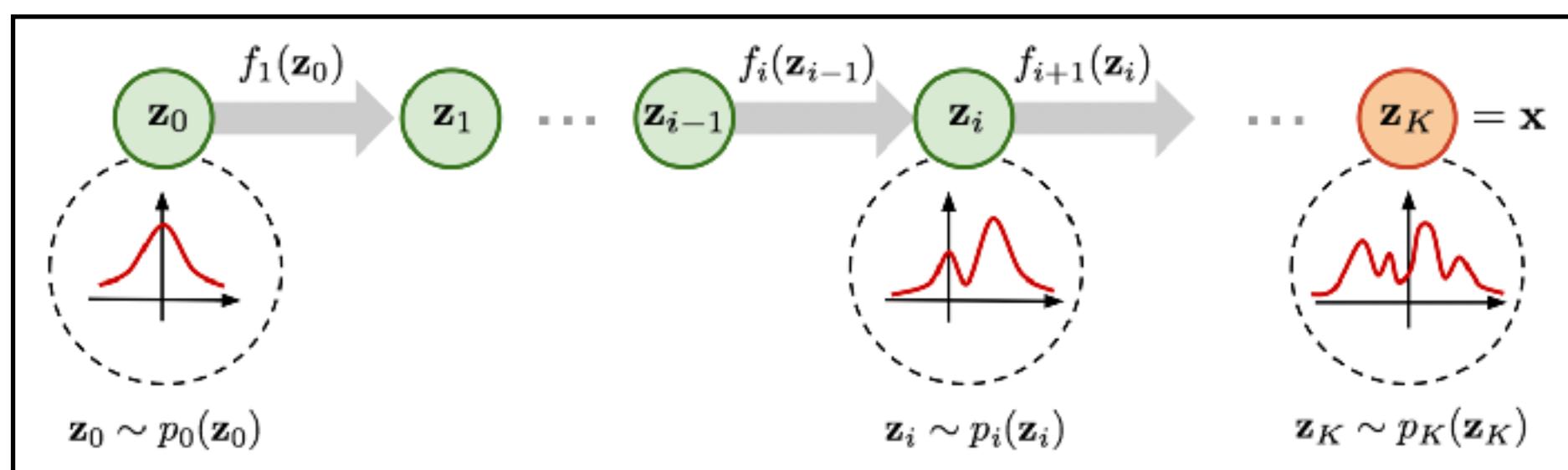
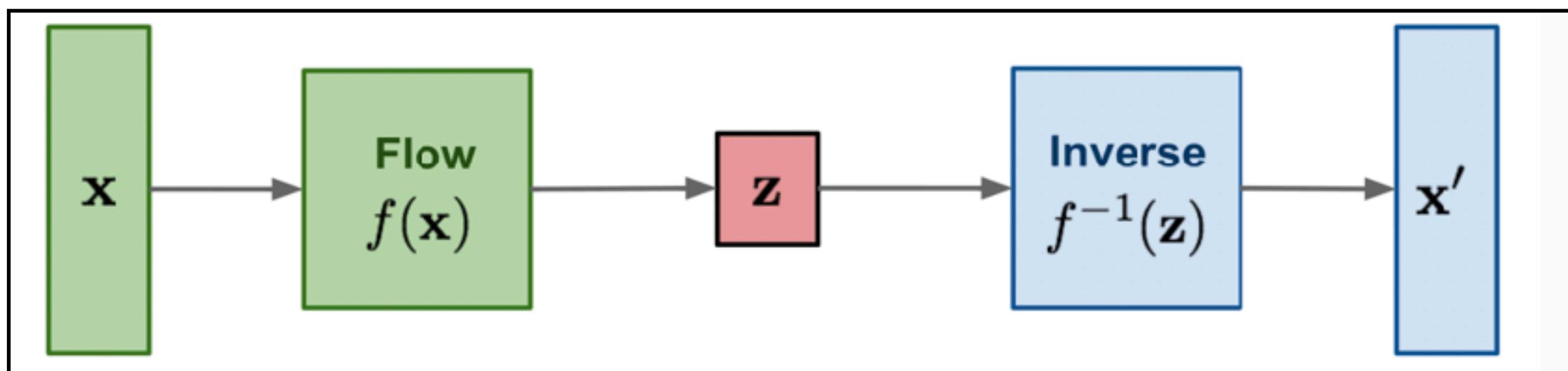
Trainable part that **reconstructs** information
(typically with a stochastic component)



“What if I make the forward process trainable and invertible and define the reverse process as the **inverted forward process**?“

“You get a normalizing flows!“

Probabilistic Diffusion



“What if I make the forward process trainable and invertible and define the reverse process as the **inverted forward process**?“

“You get a normalizing flows!“

Probabilistic Diffusion

1.

Non-trainable part that **removes** information
(stochastically or deterministically)



2.

Trainable part that **reconstructs** information
(typically with a stochastic component)

“What if I the forward process masks one dimension in each step?”

“You get an autoregressive model!”

Probabilistic Diffusion

1.

Non-trainable part that **removes** information
(stochastically or deterministically)



2.

Trainable part that **reconstructs** information
(typically with a stochastic component)

“What if the forward process masks one dimension in each step?”

“You get an autoregressive model!”

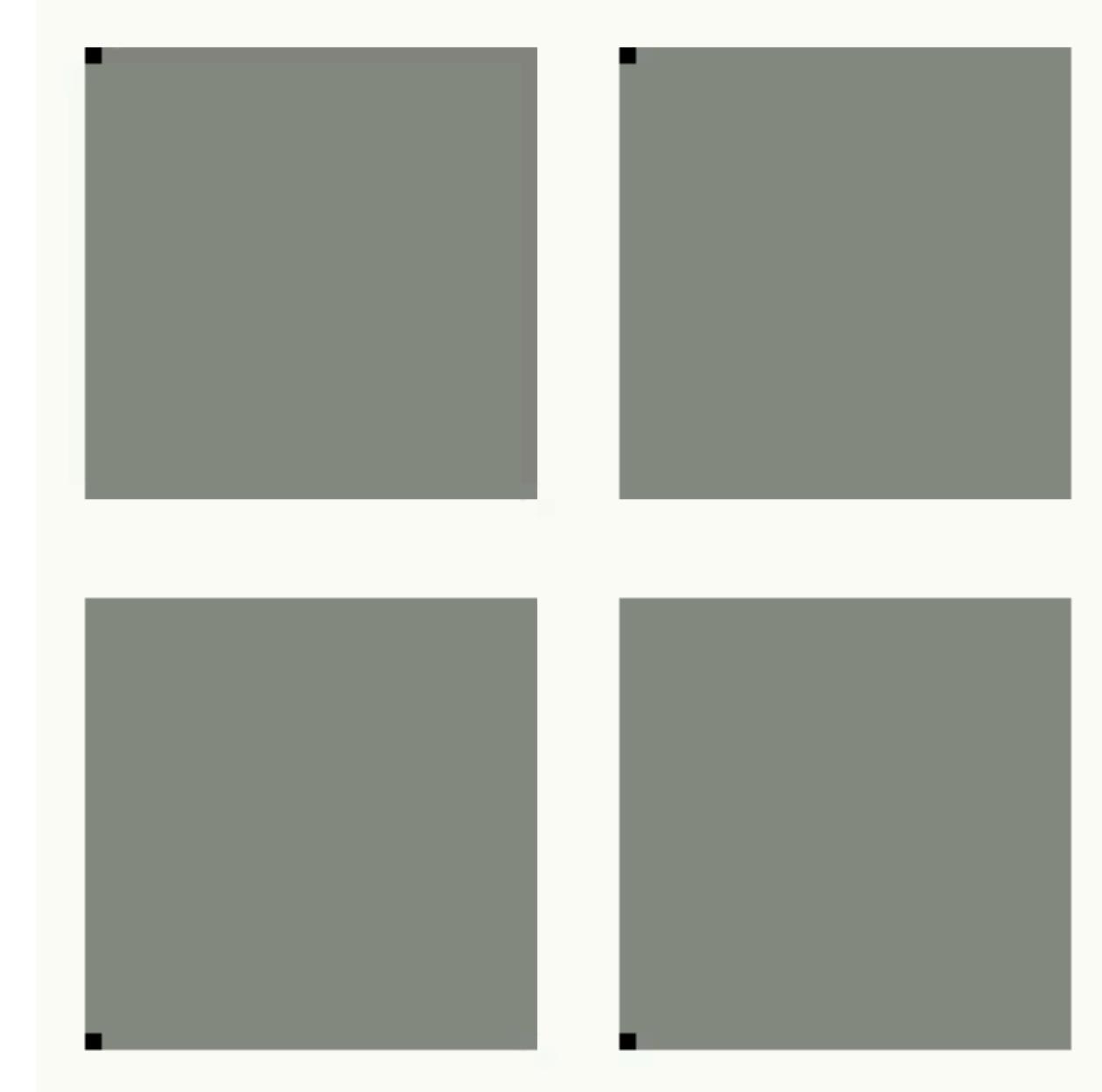
Probabilistic Diffusion



“What if the forward process masks one dimension in each step?”

“You get an autoregressive model!”

Probabilistic Diffusion

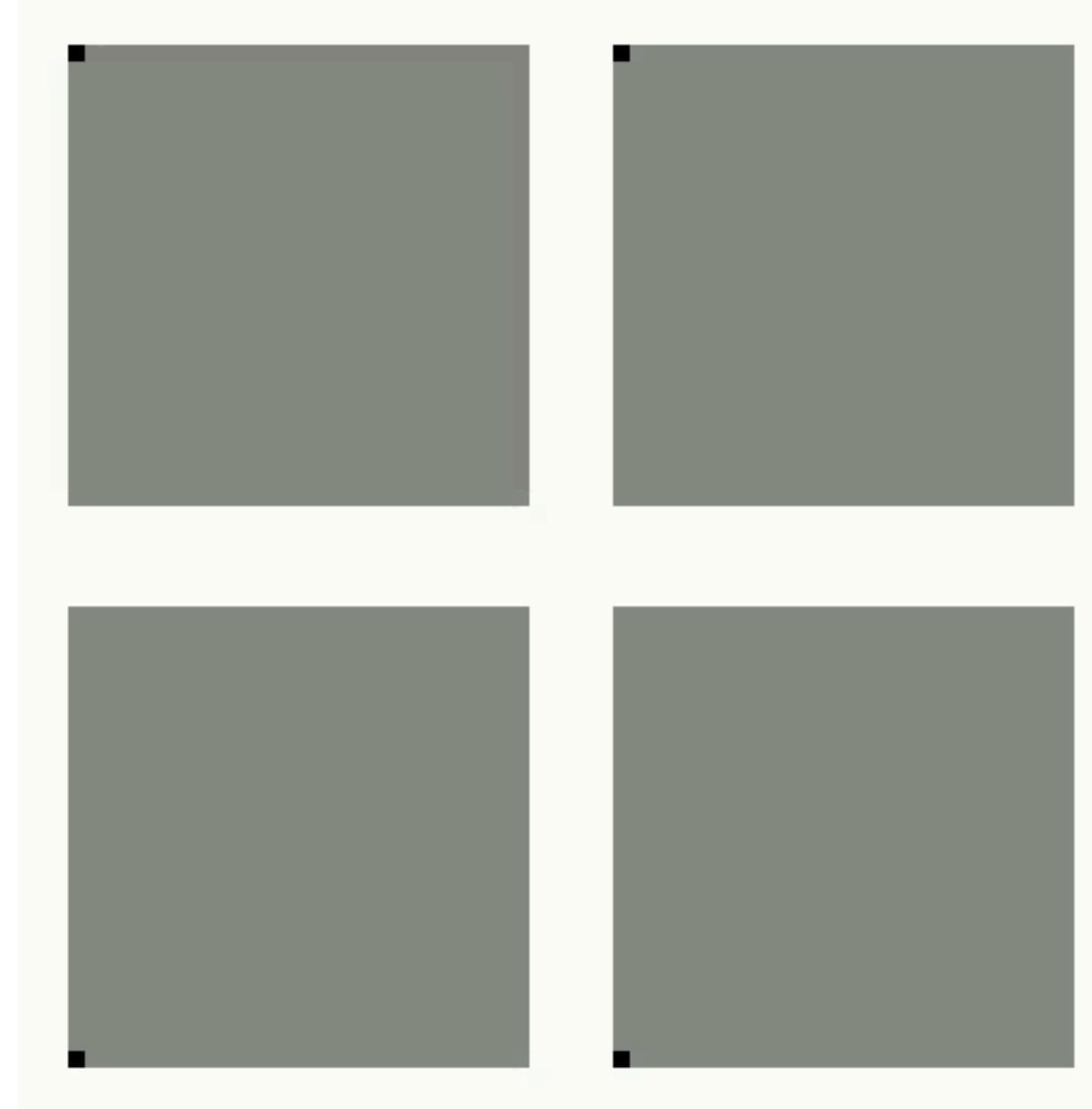


ajayjain.github.io/lmconv/

“What if the forward process masks one dimension in each step?”

“You get an autoregressive model!”

Personal Note



- **Fixed Steps:** The number of forward steps is fixed and equals the number of dimensions.
- **Inflexibility:** Autoregressive models remove information in a rigid manner (all information at once for each dimension).
- **Error Propagation:** Cannot correct past mistakes.

I think about diffusion models as **generalizations of auto-regressive models**.

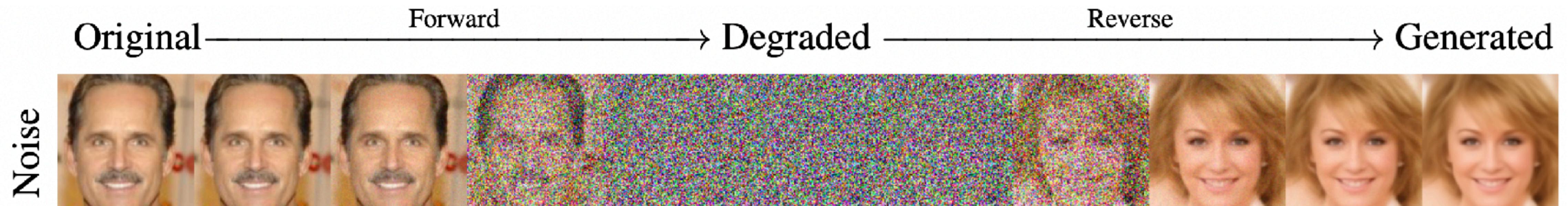
Probabilistic Diffusion

1.

Non-trainable part that **removes** information
(stochastically or deterministically)

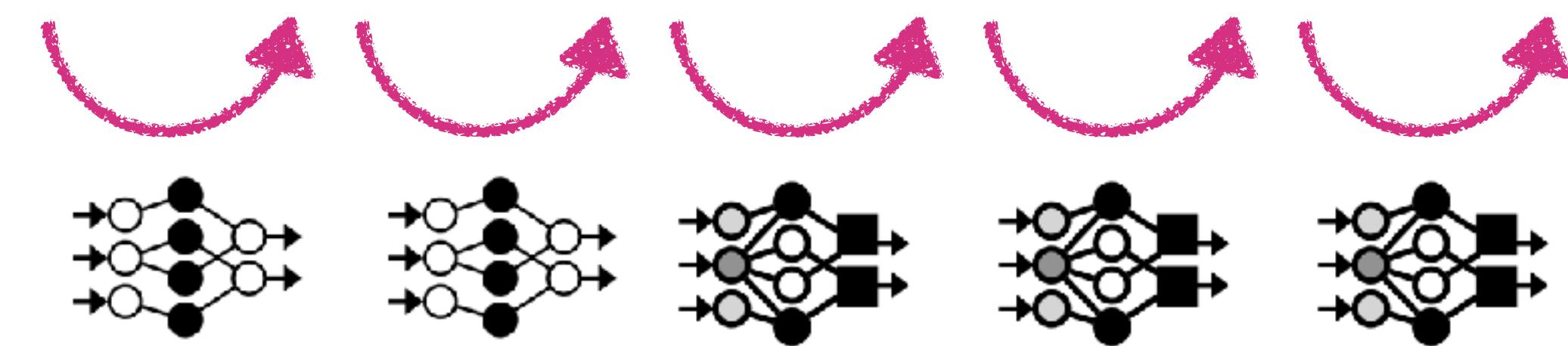
2.

Trainable part that **reconstructs** information
(typically with a stochastic component)

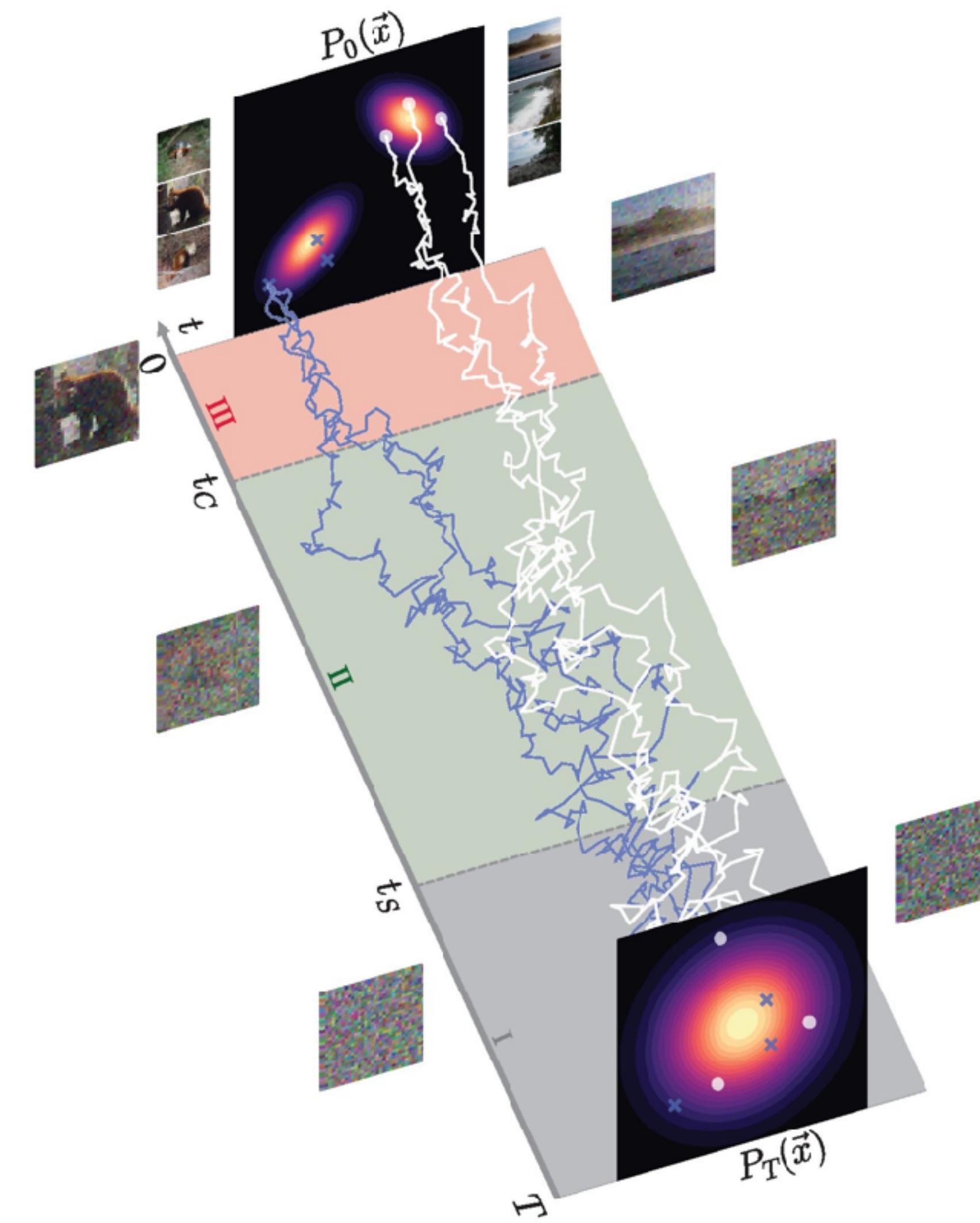


“Can you use different neural networks?”

“Generally yes, but not necessary!”



Probabilistic Diffusion



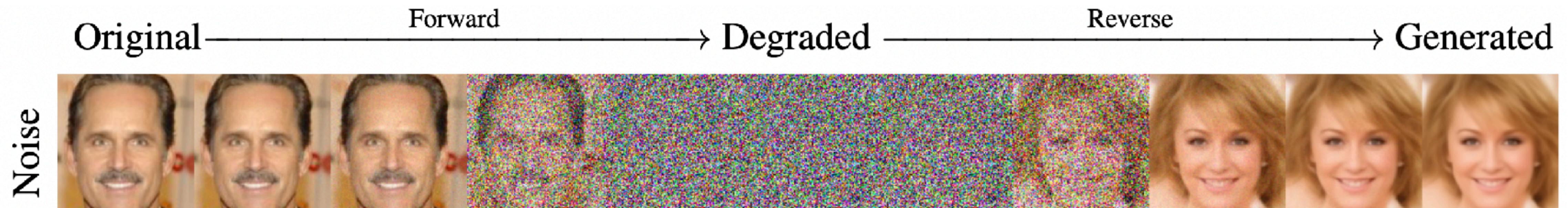
Probabilistic Diffusion

1.

Non-trainable part that **removes** information
(stochastically or deterministically)

2.

Trainable part that **reconstructs** information
(typically with a stochastic component)

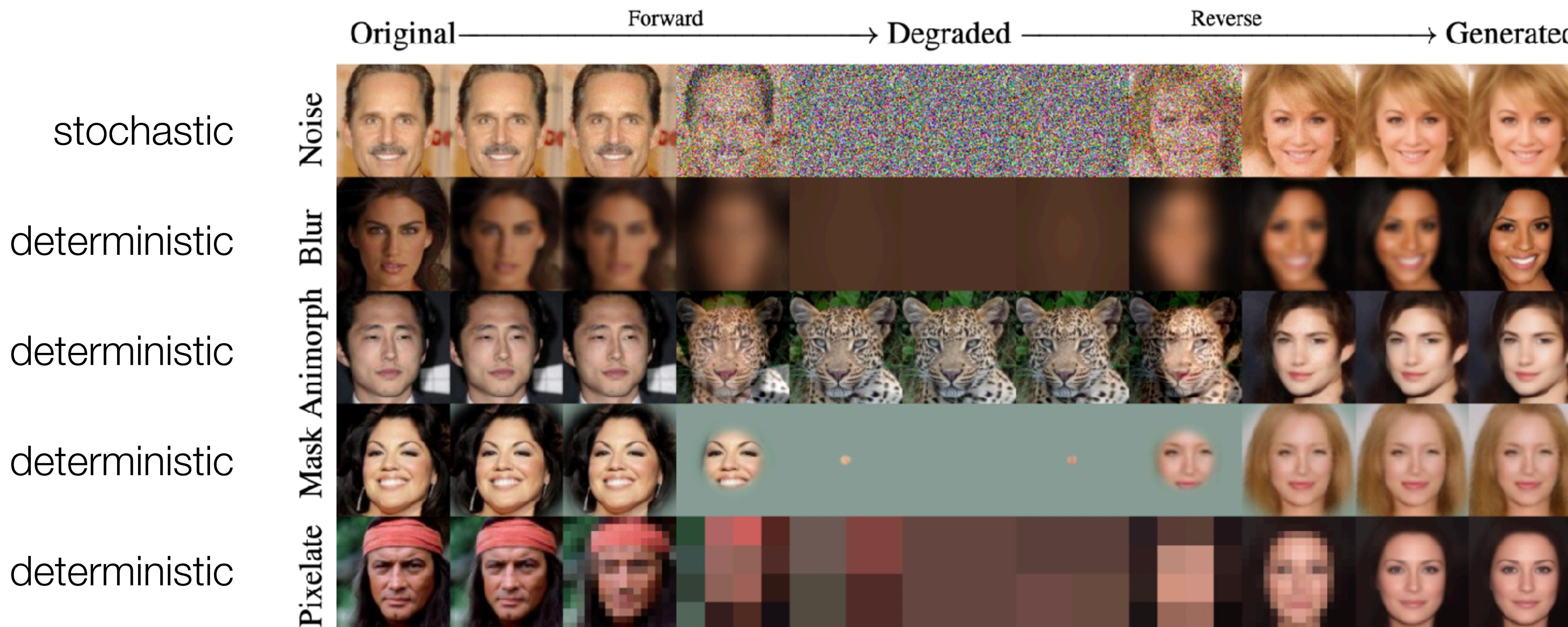


Obvious way: Degrade a sample by **adding Gaussian noise** to it.

Probabilistic Diffusion

Non-trainable part that **removes** information
(stochastically or deterministically)

Trainable part that **reconstructs** information
(typically with a stochastic component)

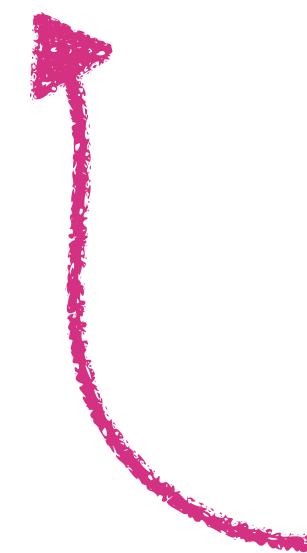


Cold Diffusion: Inverting Arbitrary Image Transforms Without Noise (Bansal et al.)

Probabilistic Diffusion

Non-trainable part that **removes** information
(stochastically or deterministically)

Trainable part that **reconstructs** information
(typically with a stochastic component)



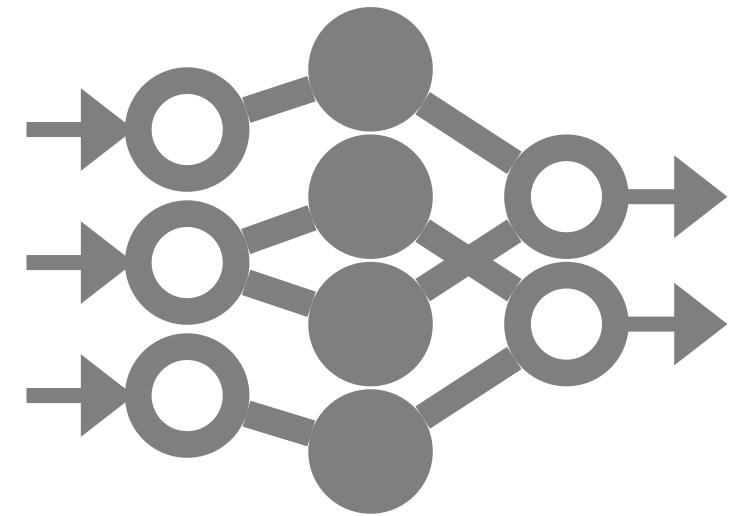
Take Home:

Generally, it is a good idea to add at least a little bit of noise to the forward process!

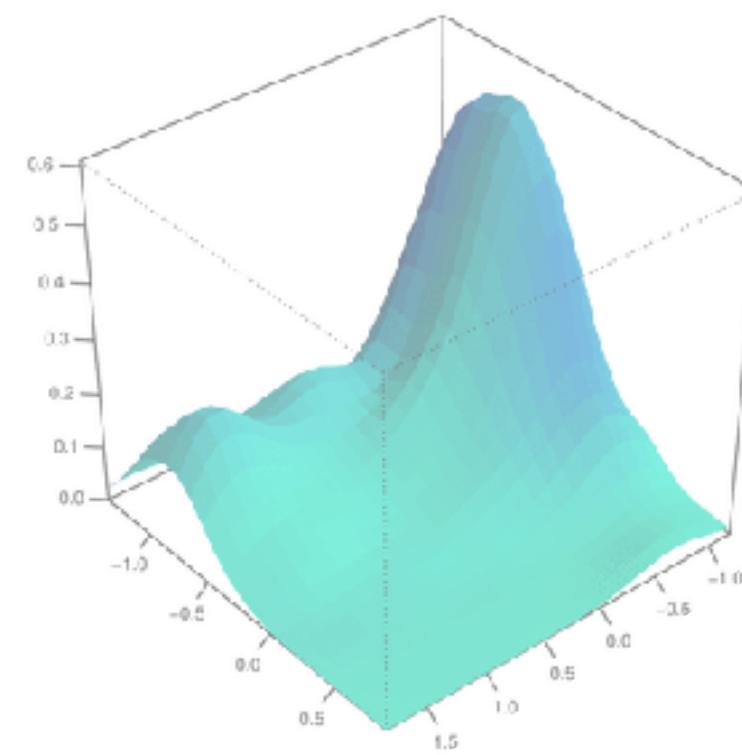


Ensures that the model encounters a wider **variety** of data during training!

Generative Deep Learning



Neural network with millions of trainable parameters (weights)



Weights encode a probability distribution.

0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7 7 7 7 7 7 7
8 8 8 8 8 8 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9 9 9 9 9 9

Training adjusts the weights based on training data

3

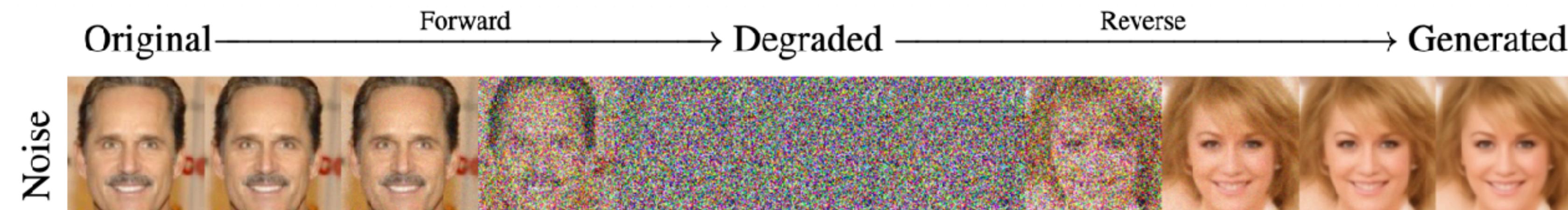
Sampling methods generate new data points.



How to Sample

Non-trainable part that **removes** information
(stochastically or deterministically)

Trainable part that **reconstructs** information
(typically with a stochastic component)



We have

- a function $f(\cdot)$ that removes information and
- a neural network that reconstructs information.

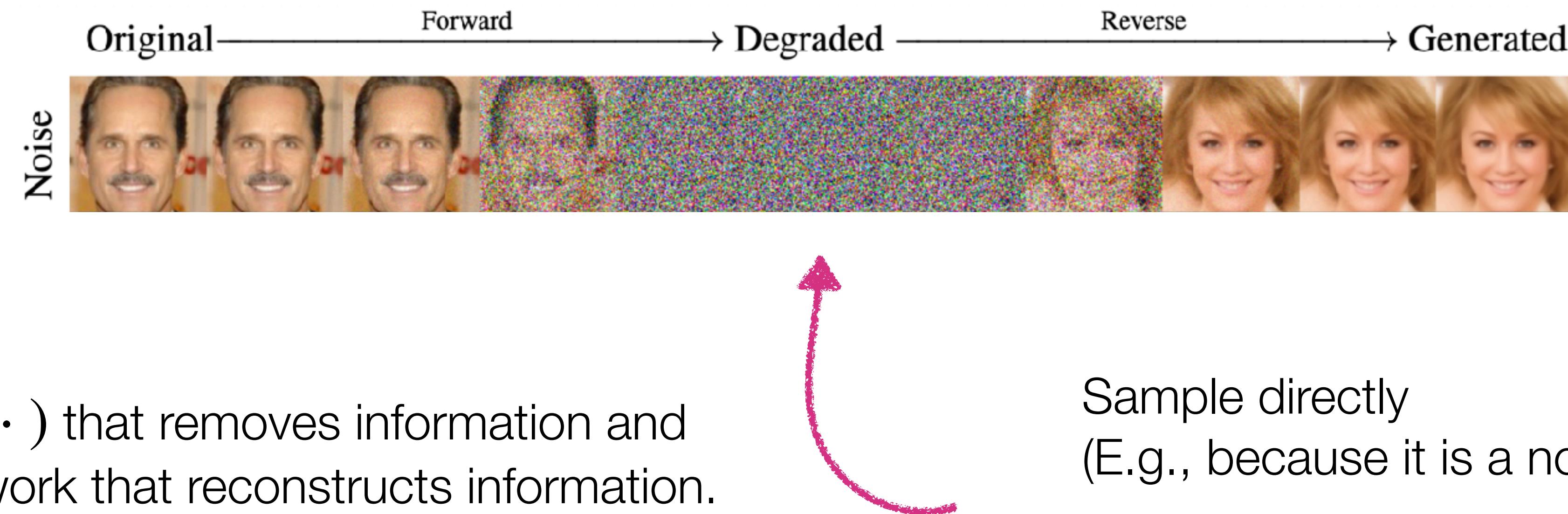
How do we generate samples?

- Sample from the **noise distribution**.
- Apply NN iteratively.

How to Sample

Non-trainable part that **removes** information
(stochastically or deterministically)

Trainable part that **reconstructs** information
(typically with a stochastic component)



We have

- a function $f(\cdot)$ that removes information and
- a neural network that reconstructs information.

How do we generate samples?

- Sample from the **noise distribution**.
- Apply NN iteratively.

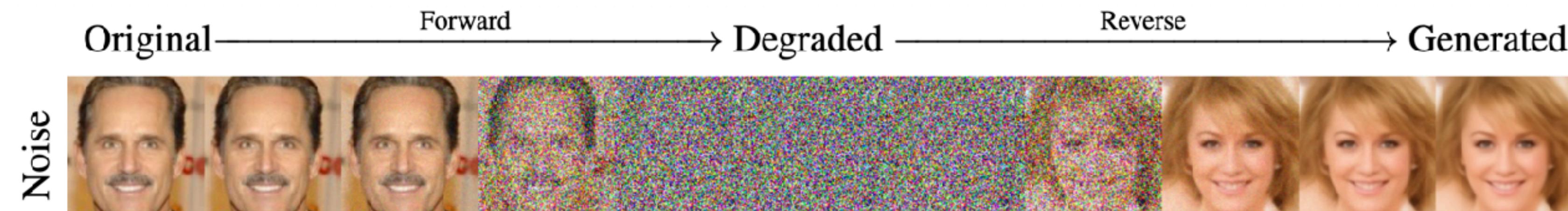
Sample directly
(E.g., because it is a normal distribution)

Add noise to random
sample from trainings set.

How to Sample

Non-trainable part that **removes** information
(stochastically or deterministically)

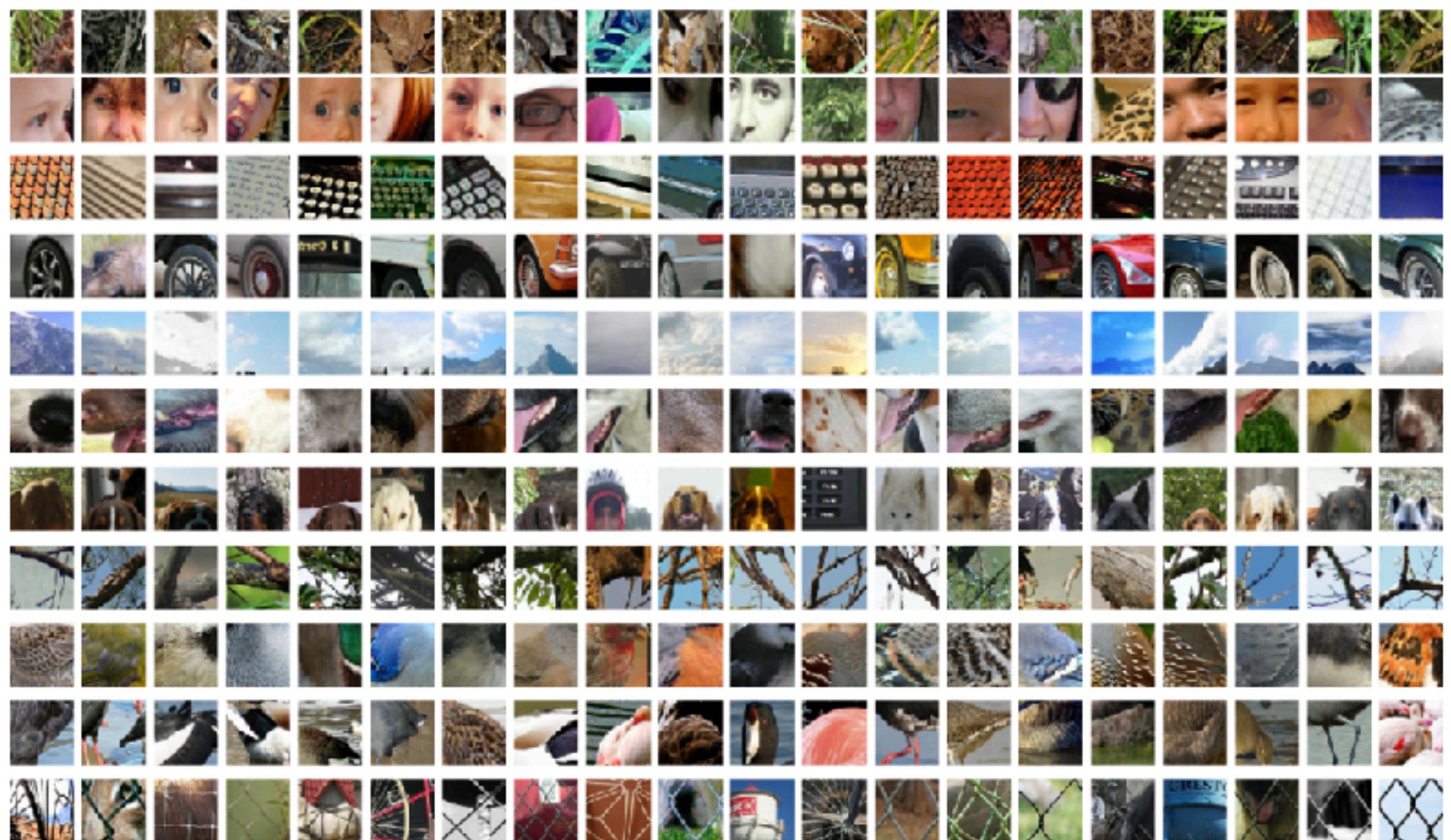
Trainable part that **reconstructs** information
(typically with a stochastic component)



“Can you also use a uniform distribution as the noise distribution?”

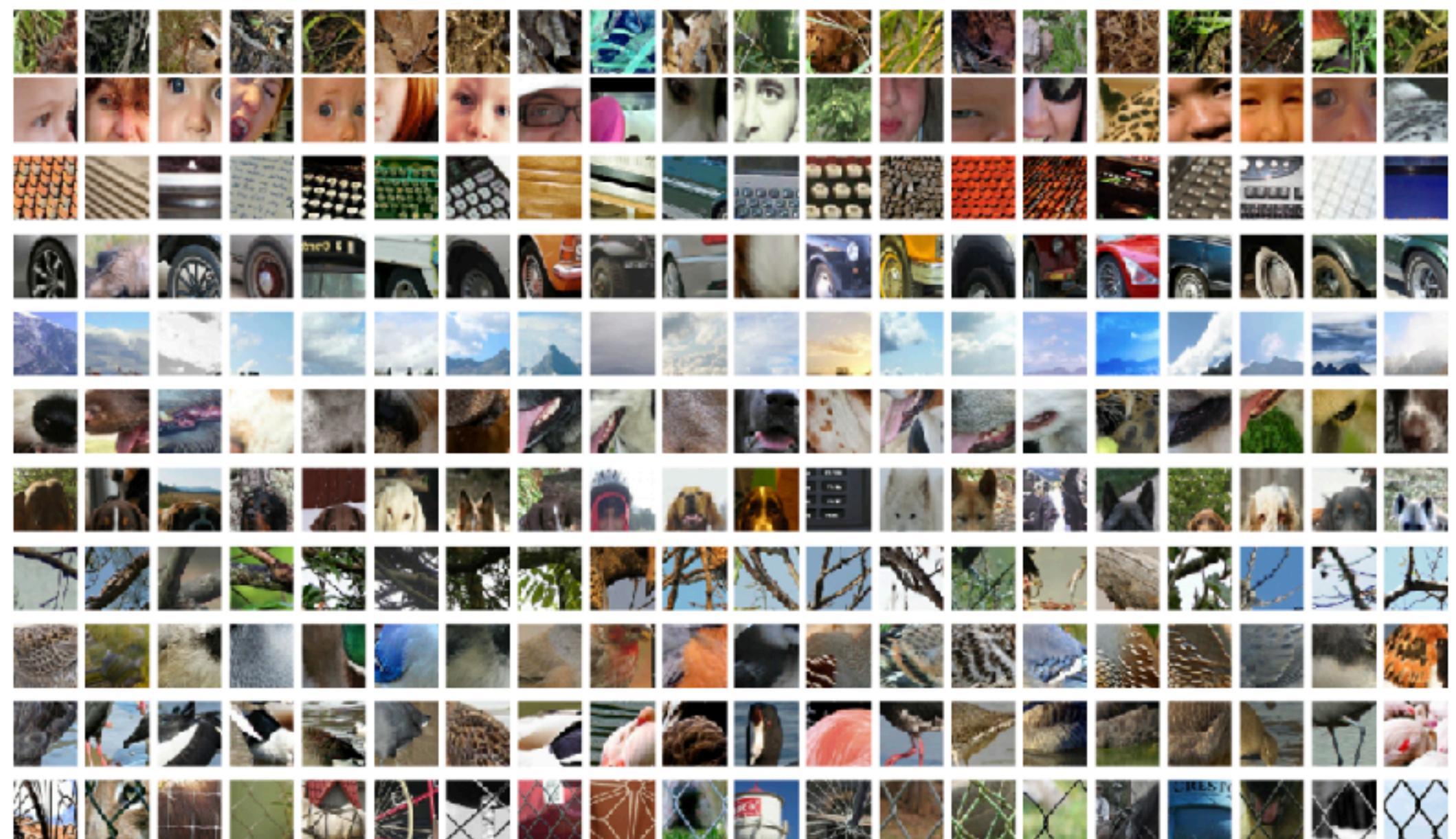
“Yes”

Conditional Sampling



semanticscholar.org/paper/Representation-Learning-with-Contrastive-Predictive-Oord-Li/b227f3e4c0dc96e5ac5426b85485a70f2175a205/figure/7

Conditional Sampling



A panda on a motorcycle with a red sombrero and a cigar, oil painting.



semanticscholar.org/paper/Representation-Learning-with-Contrastive-Predictive-Oord-Li/b227f3e4c0dc96e5ac5426b85485a70f2175a205/figure/7

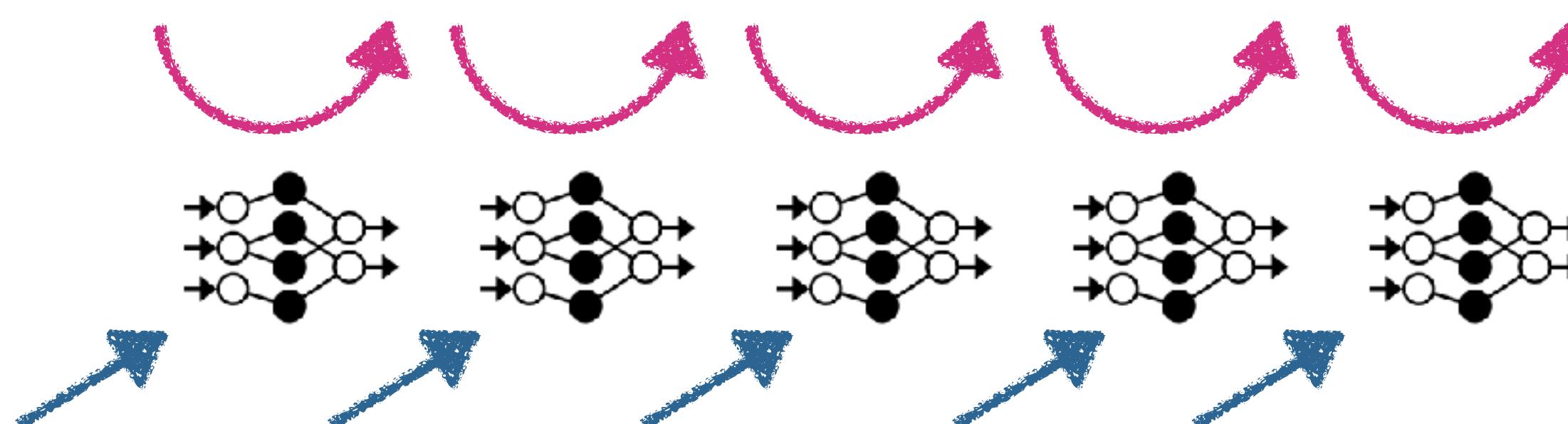
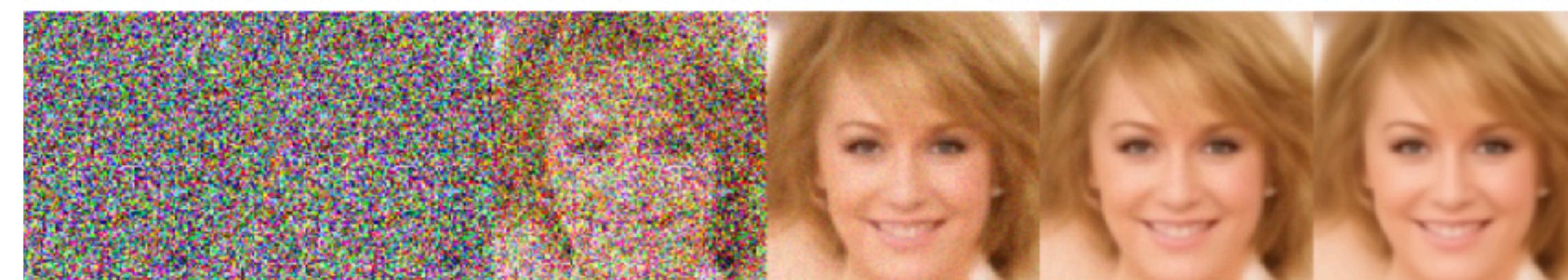
Conditional Sampling

1.

Non-trainable part that **removes** information
(stochastically or deterministically)

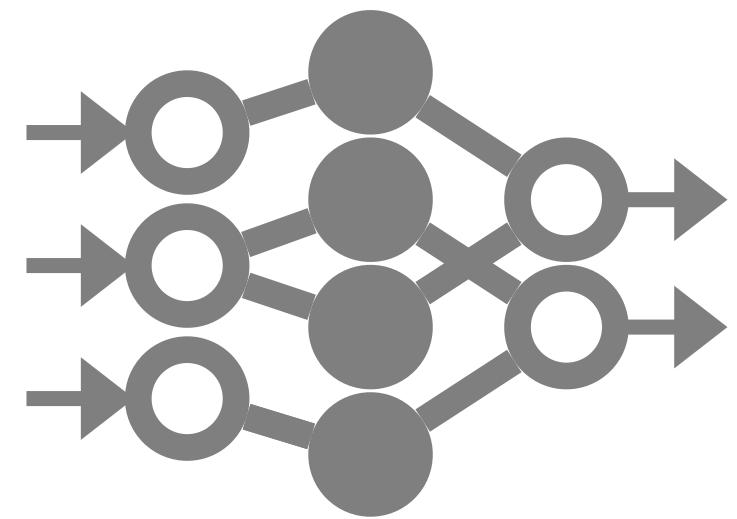
2.

Trainable part that **reconstructs** information
(typically with a stochastic component)

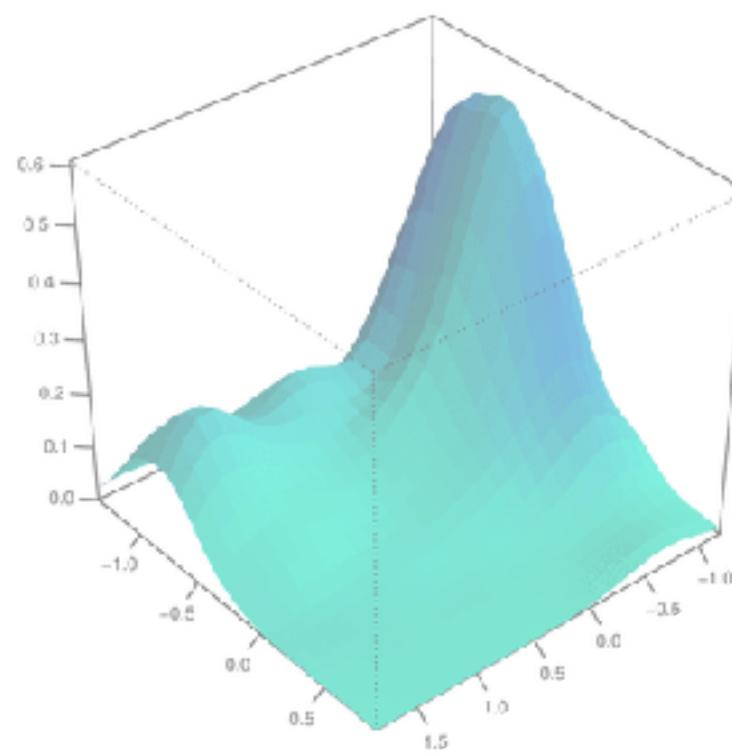


condition as conditional input

Generative Deep Learning



Neural network with millions of trainable parameters (weights)



Weights encode a probability distribution.

Training adjusts the weights
based on training data

3

Sampling methods generate new data points.



How to Train

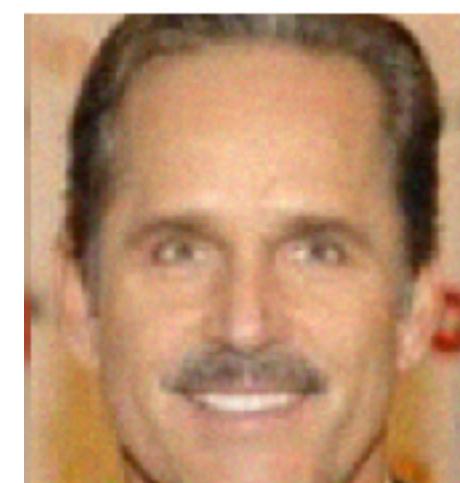
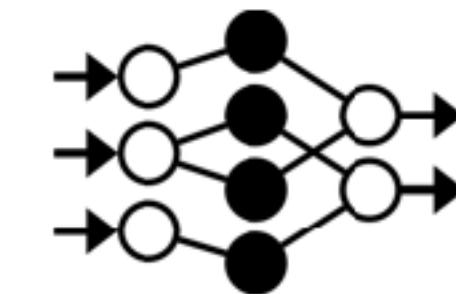
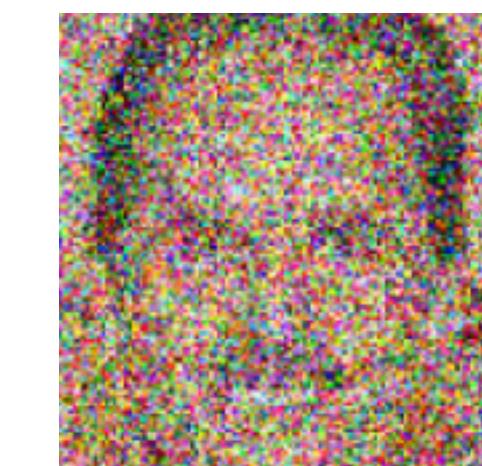
Non-trainable part that **removes** information
(stochastically or deterministically)

Trainable part that **reconstructs** information
(typically with a stochastic component)



Method 1

Train the NN to reconstruct
the information of exactly
one step.



How to Train

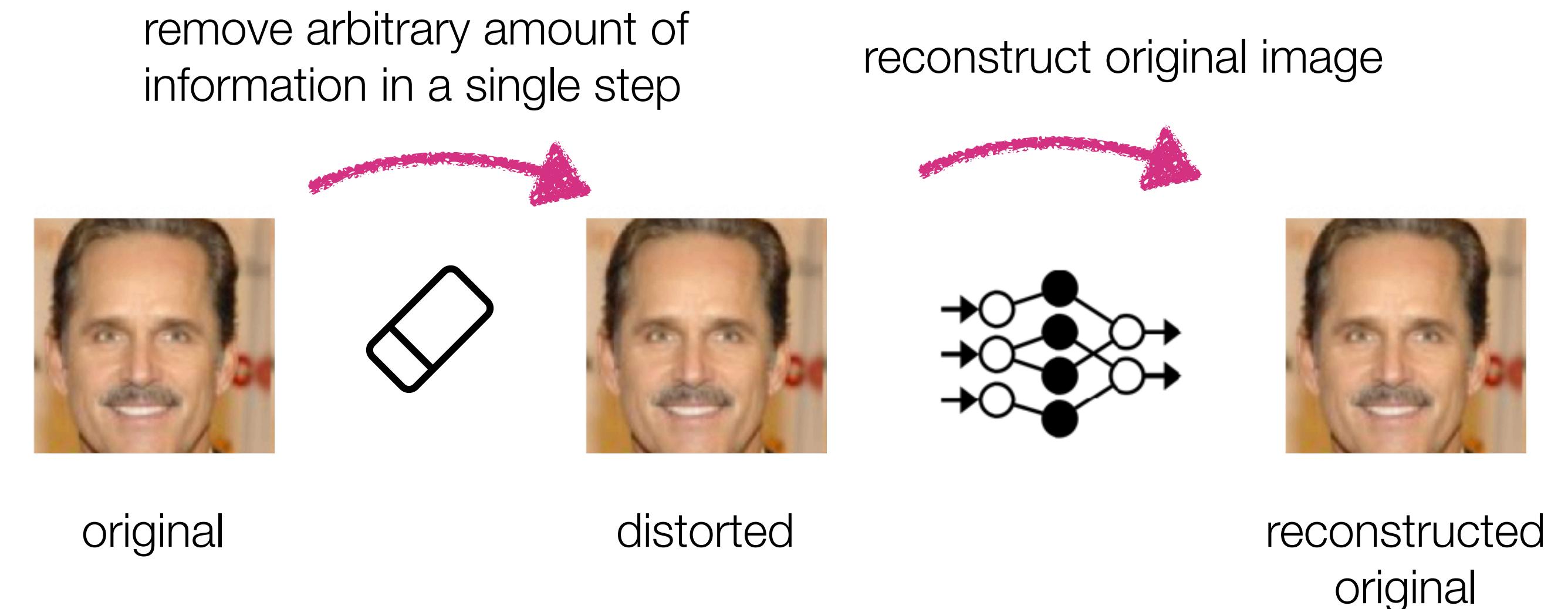
Non-trainable part that **removes** information
(stochastically or deterministically)

Trainable part that **reconstructs** information
(typically with a stochastic component)



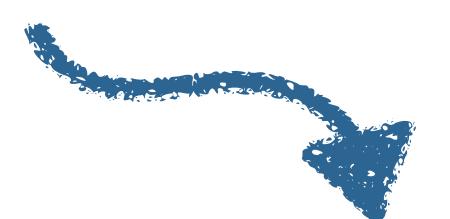
Method 2

1. Sample the amount of noise.
2. Add all this noise at once.
3. Reconstruct the whole image.



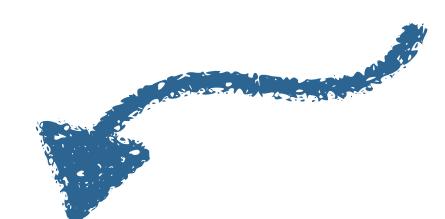
How to Train

This only works when we can perform multiple forward steps in a single step.



remove arbitrary amount of information in a single step

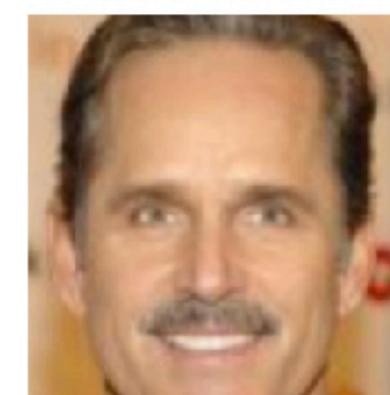
This only works when we account for this in the sampling process.



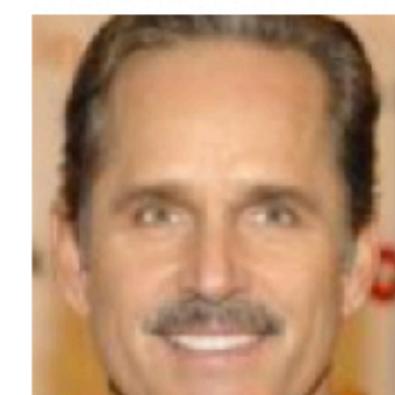
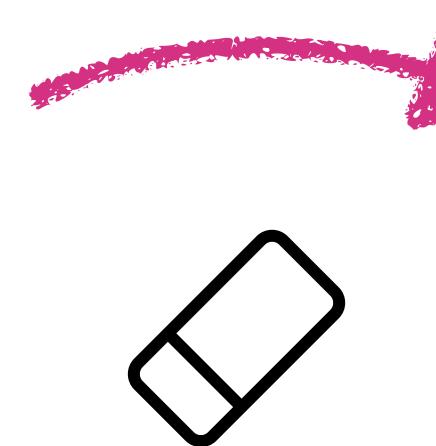
reconstruct original image

Method 2

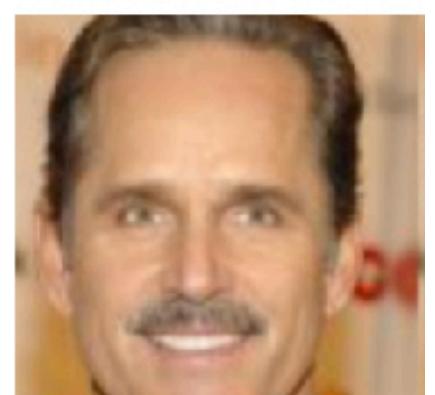
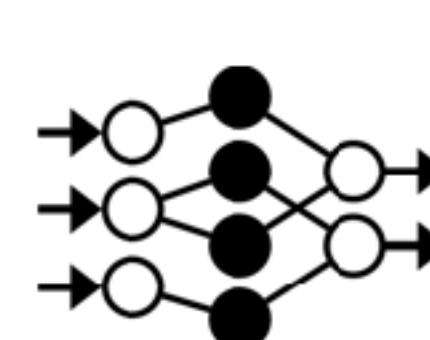
1. Sample the amount of noise.
2. Add all this noise at once.
3. Reconstruct the whole image.



original



distorted



reconstructed
original

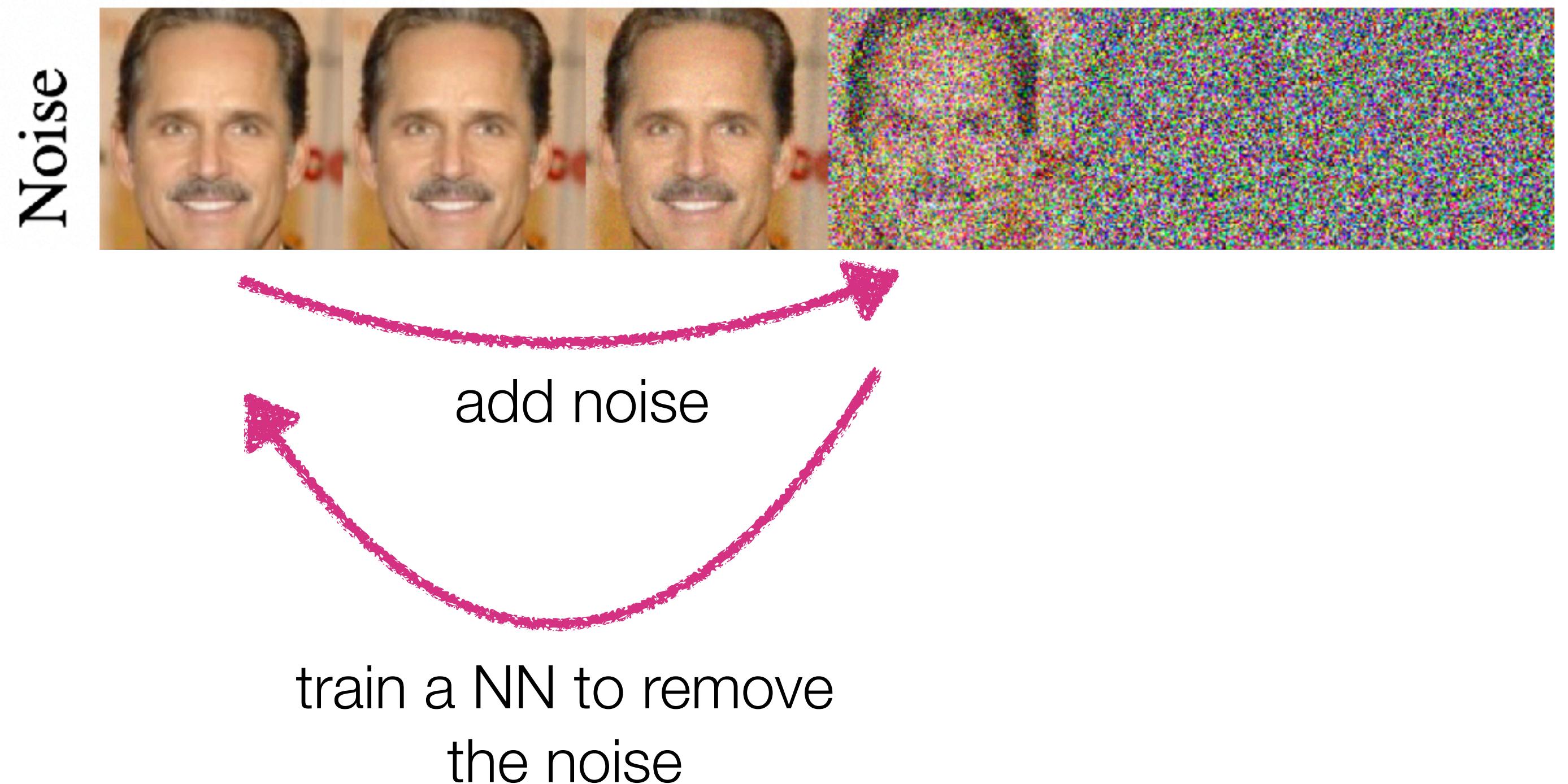
How to Train

Non-trainable part that **removes** information
(stochastically or deterministically)

Trainable part that **reconstructs** information
(typically with a stochastic component)

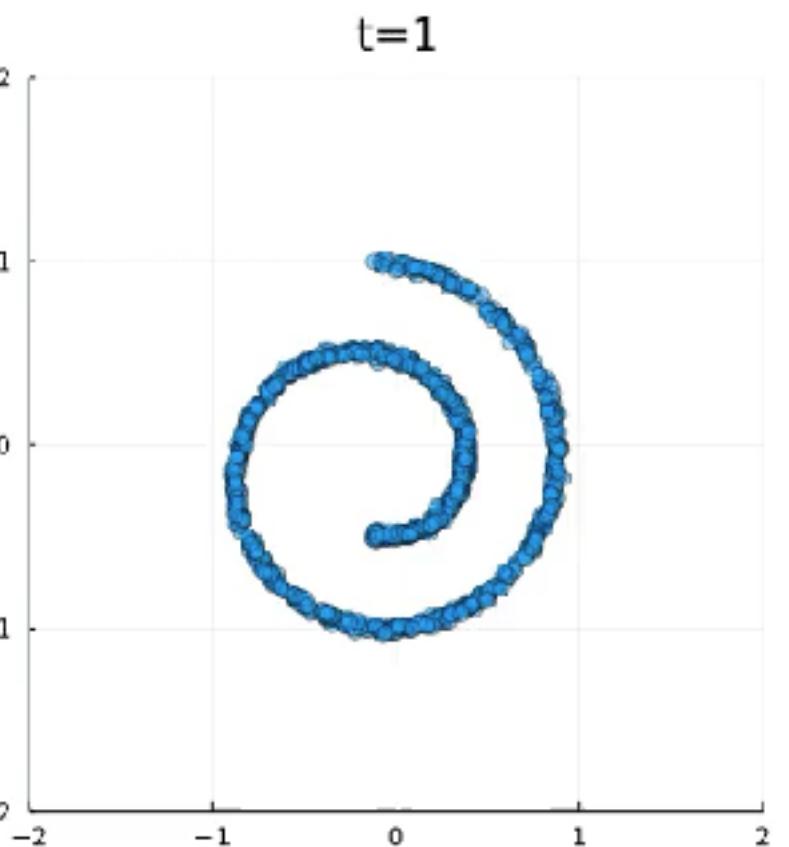
Method 2

1. Sample the amount of noise.
2. Add all this noise at once.
3. Reconstruct the whole image.



How to Train

Spiral forward



Method 2

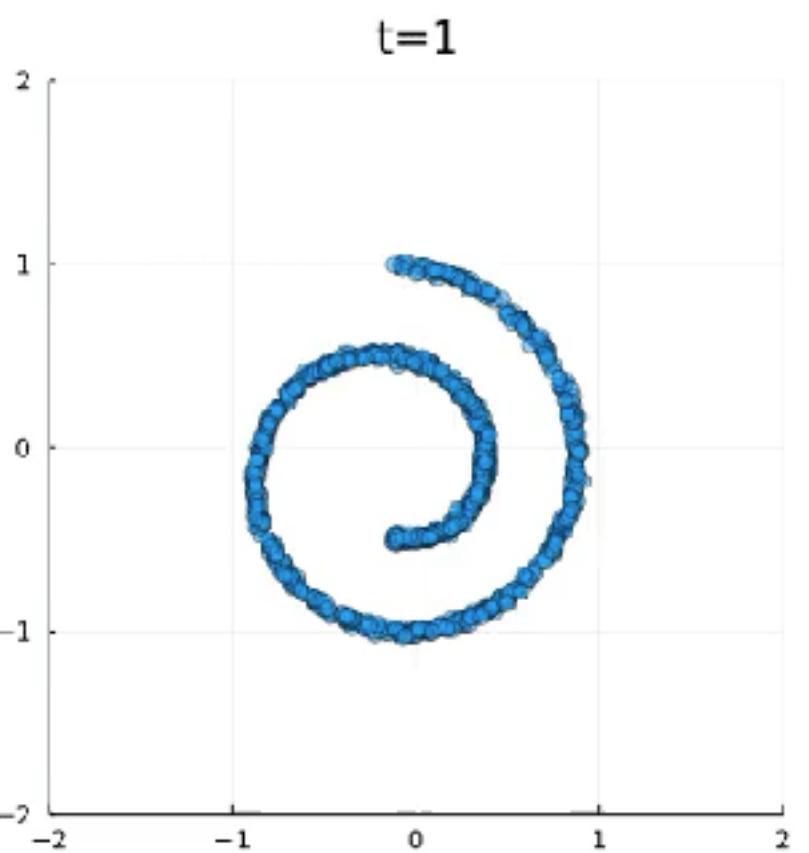
1. Sample the amount of noise.
2. Add all this noise at once.
3. Reconstruct the whole image.

How to Train

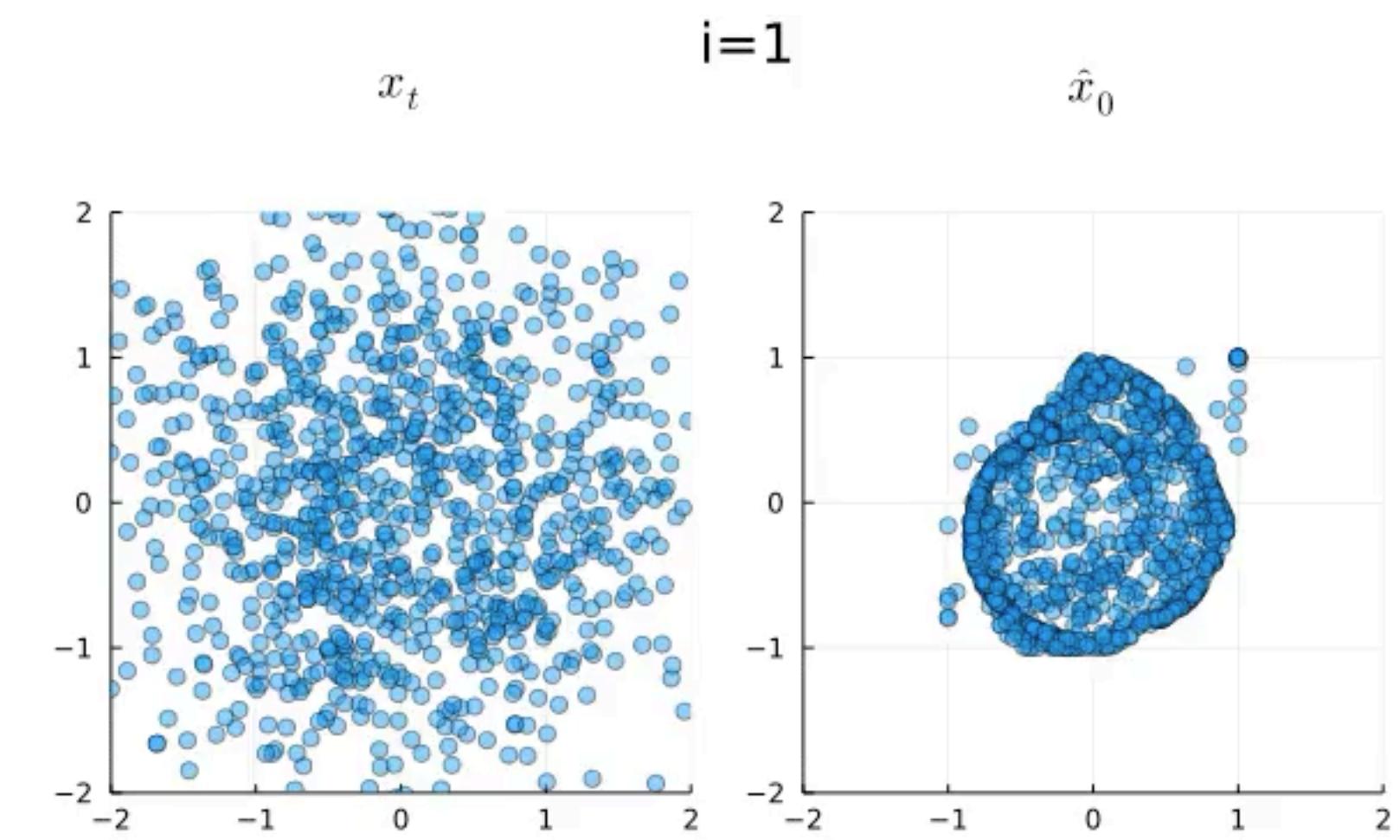
Method 2

1. Sample the amount of noise.
2. Add all this noise at once.
3. Reconstruct the whole image.

Spiral forward



Spiral reverse



Endpoint prediction at
the current time point

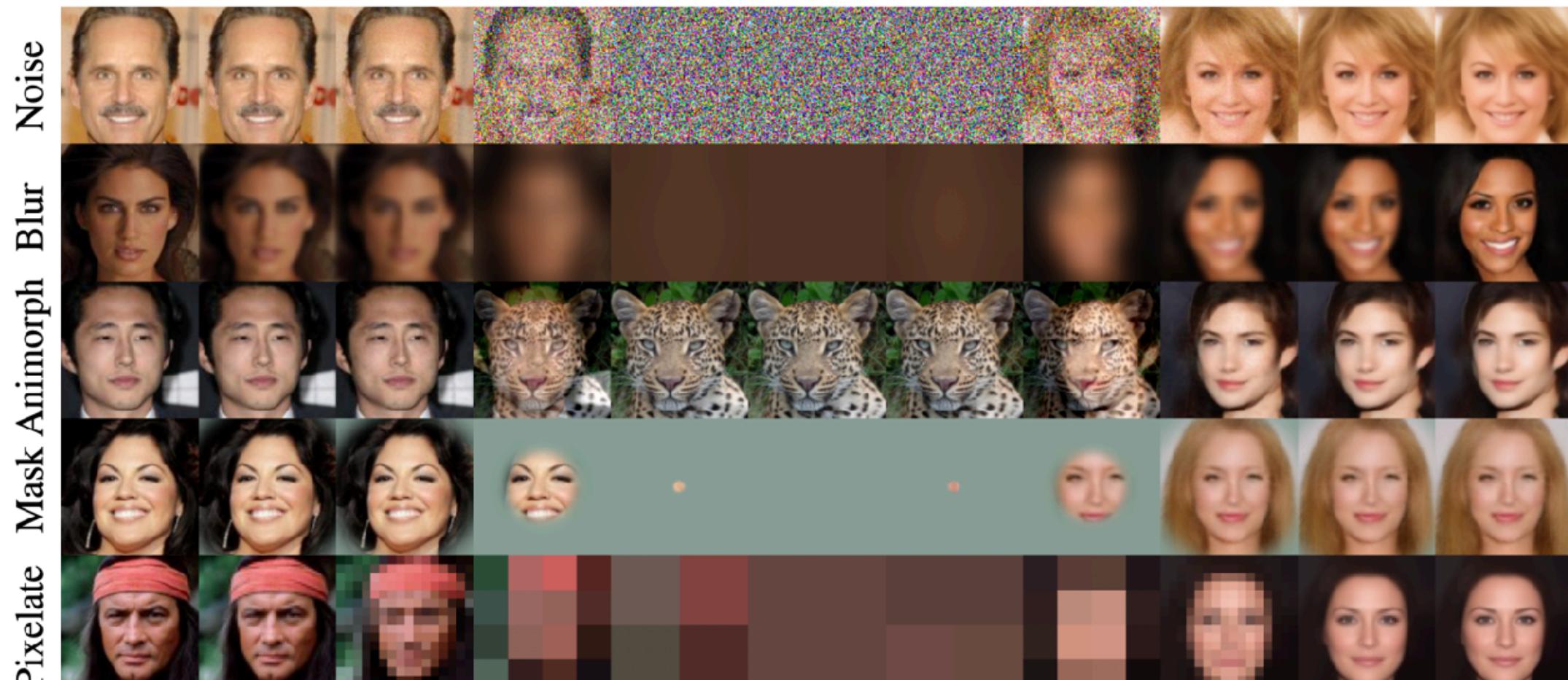


Go a small step into the direction of the
endpoint prediction
(And add some noise)

Probabilistic Diffusion

Take Home:

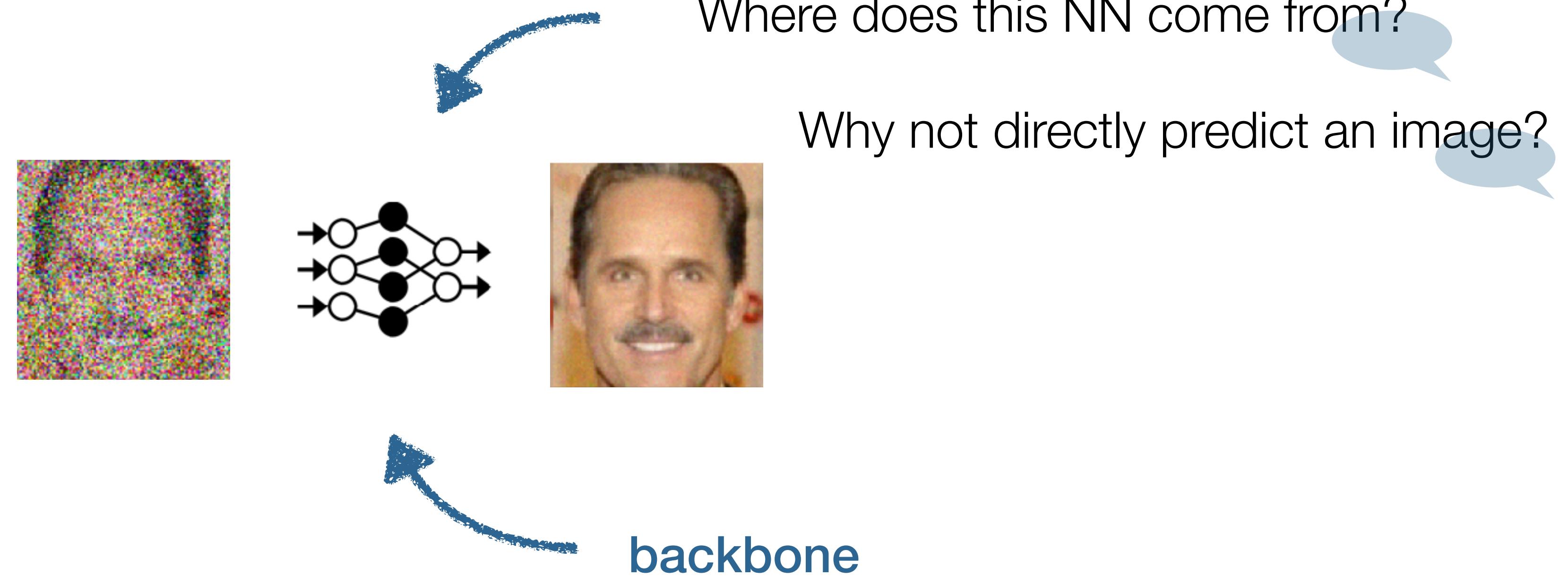
The design space of diffusion models is extremely large.



Design forward function such that:

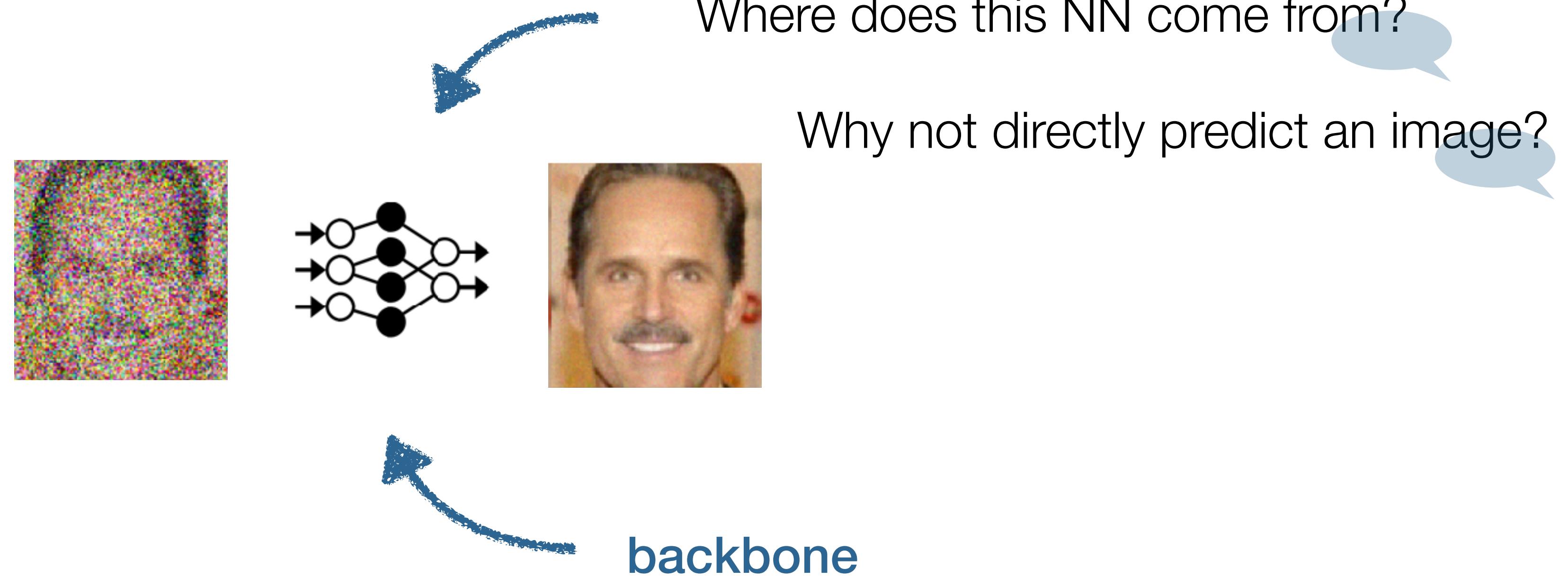
- Removes information gradually.
 - Often requires some hyper-param tuning.
- Noise distribution should be simple (e.g., Gaussian)
- Perform multiple forward steps in a single step.

Isn't This Cheating?



- Diffusion models serve as **meta-algorithms**.
- They transform simple "**backbone**" NNs into powerful generative models.
- Multi-step method **empirically outperform** single-shot methods.

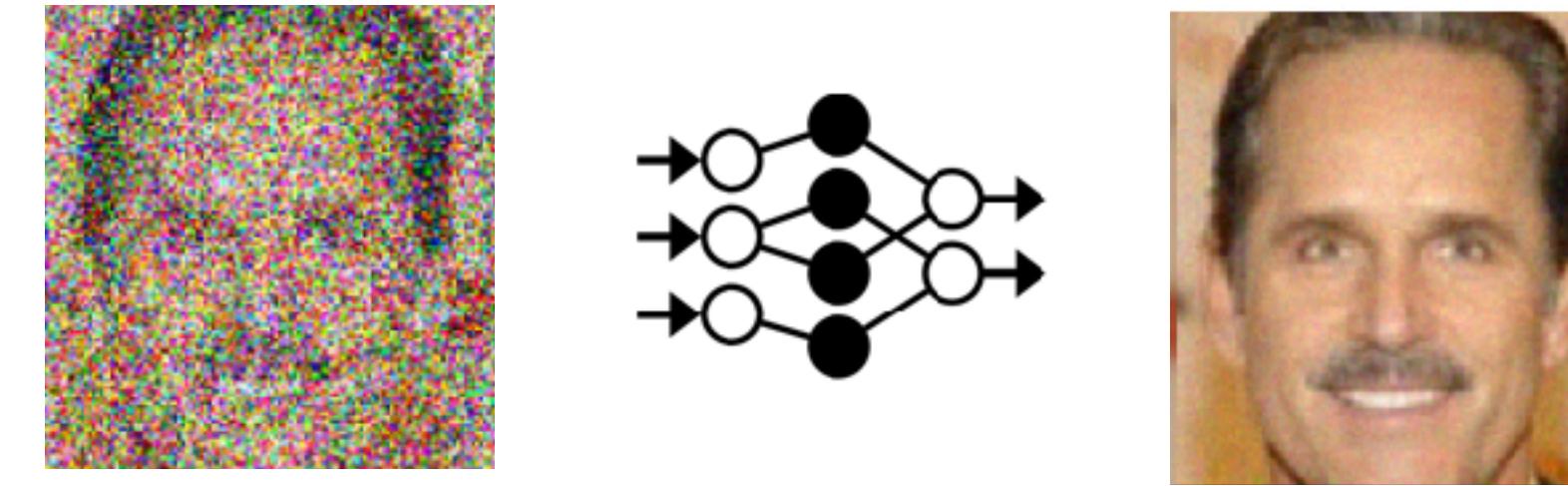
Isn't This Cheating?



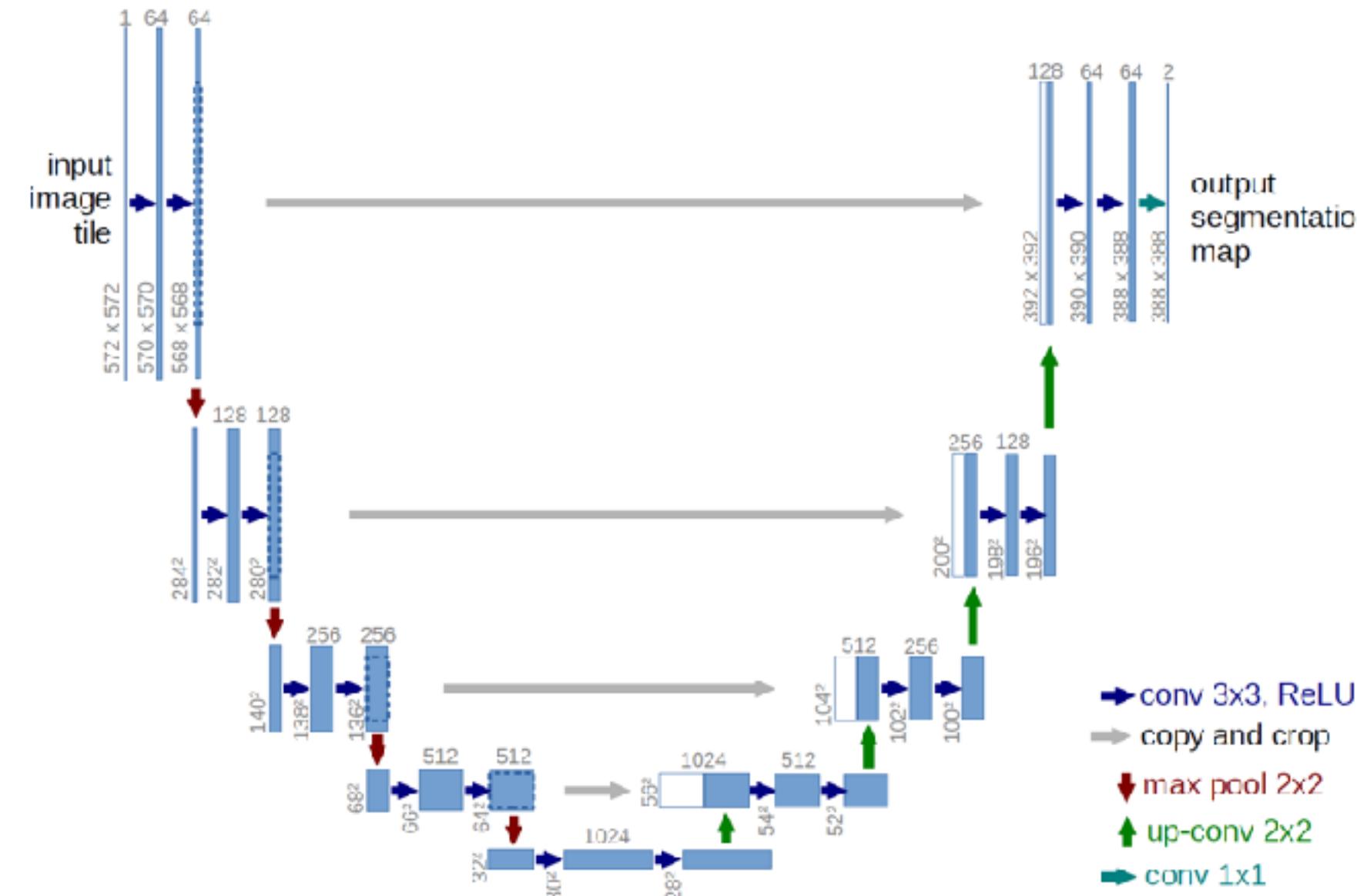
Take Home:

Put effort into building the best
backbone NN possible.

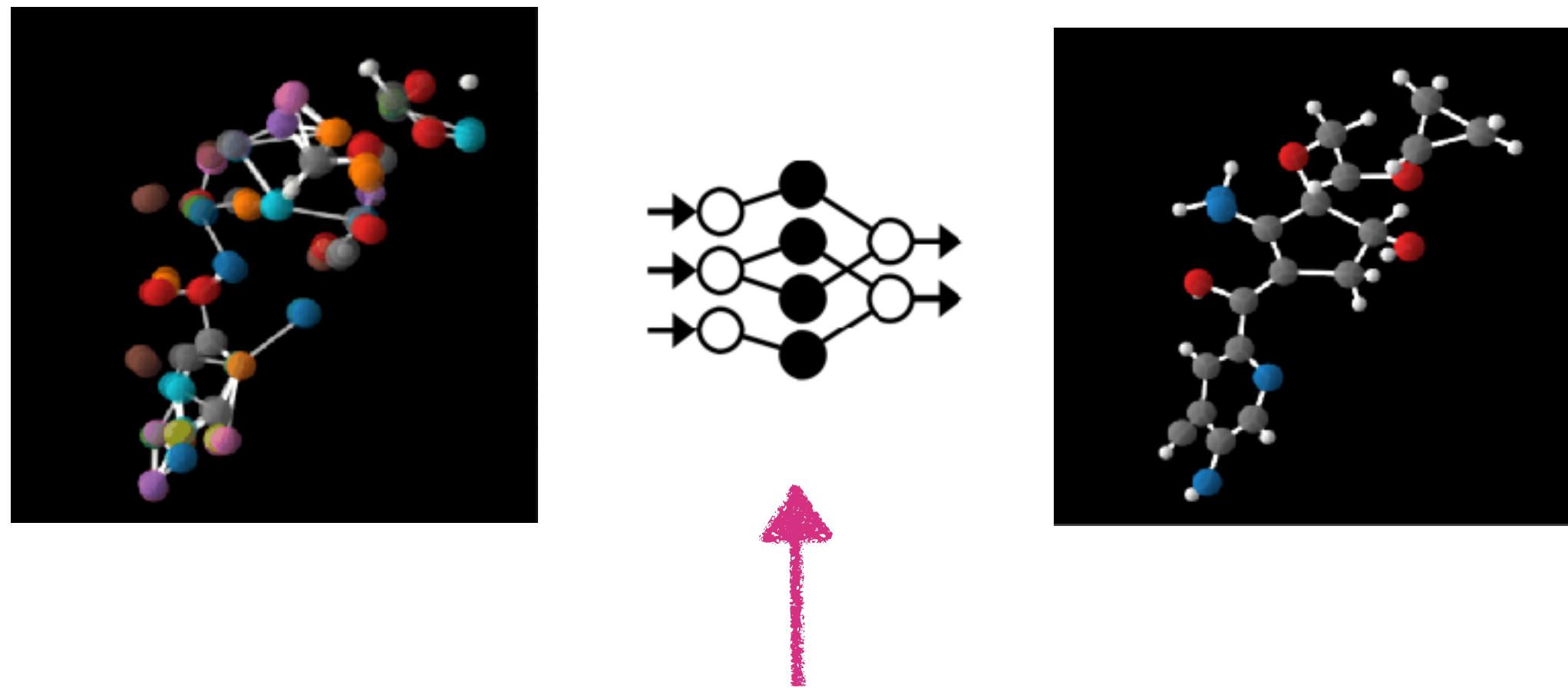
Isn't This Cheating?



U-Net



Isn't This Cheating?



Graph Neural Network

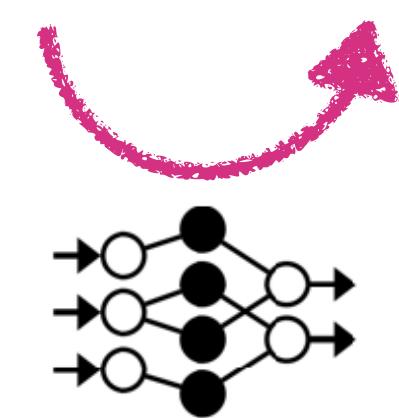
Summary

Forward process gradually removes information.

Reverse process gradually reconstructs information.



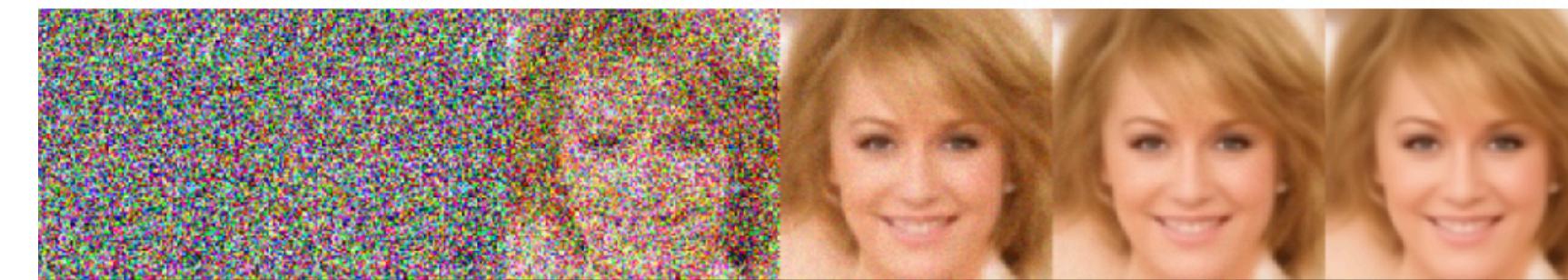
Noise distribution should be simple.



Backbone NN does the actual work.

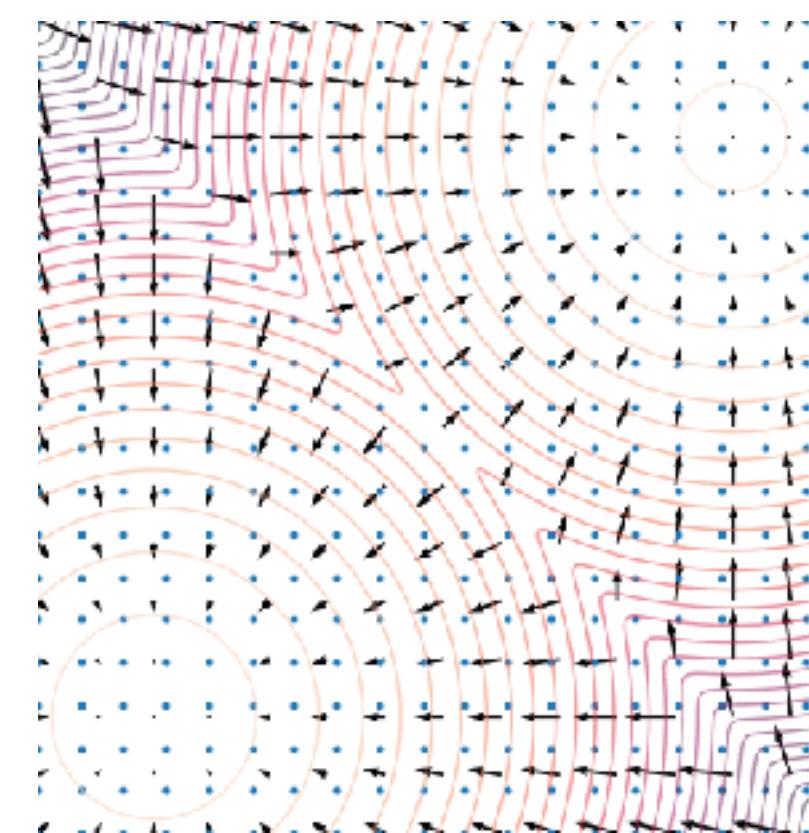
Diffusion in Literature

Denoising (DDPM)

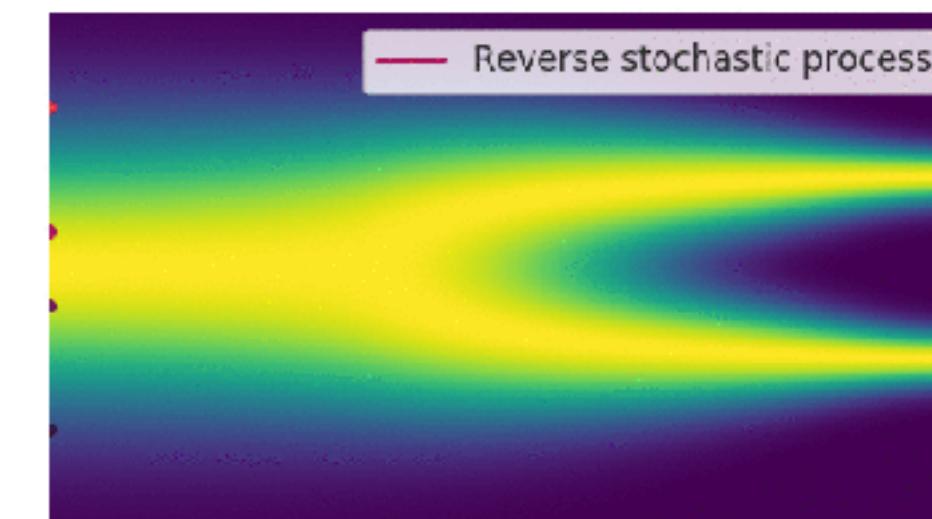
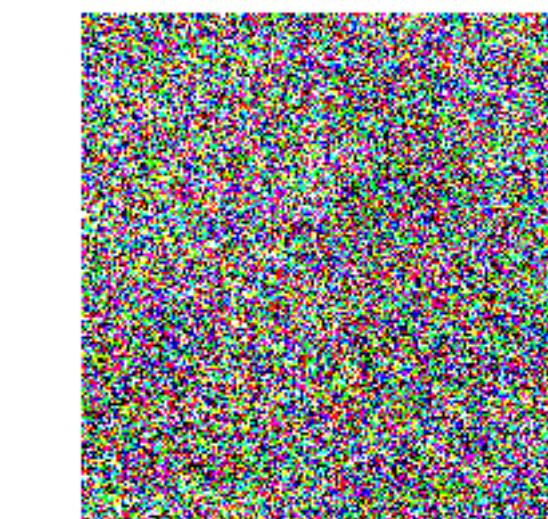


Cold Diffusion: Inverting Arbitrary Image Transforms Without Noise (Bansal et al.)

SDE

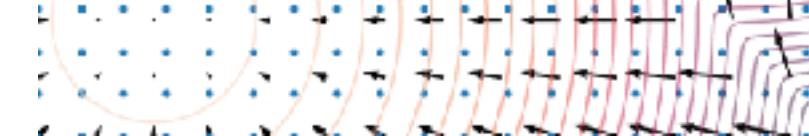


yang-song.net/blog/2021/score/

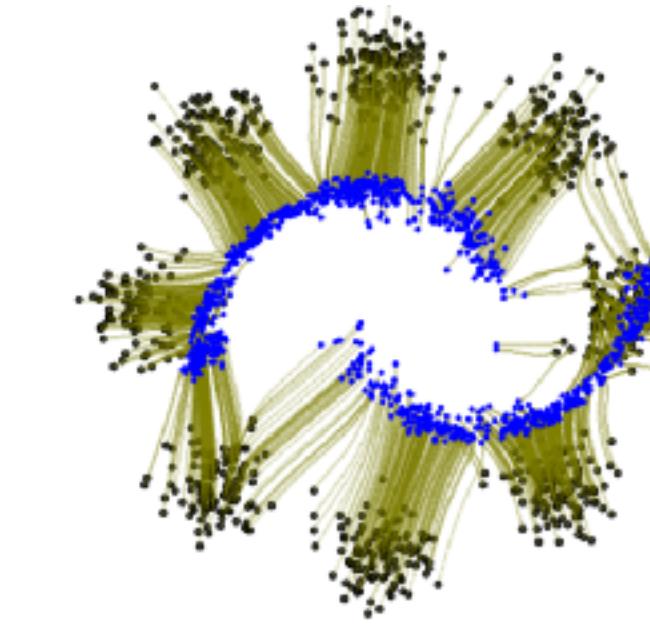
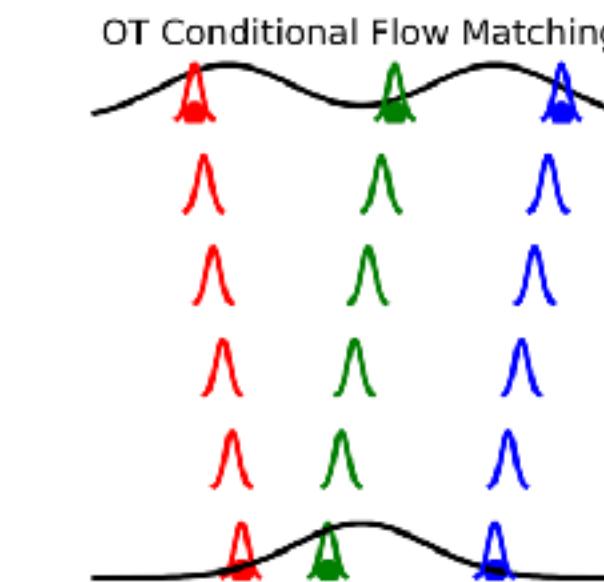


yang-song.net/blog/2021/score/

Score Matching

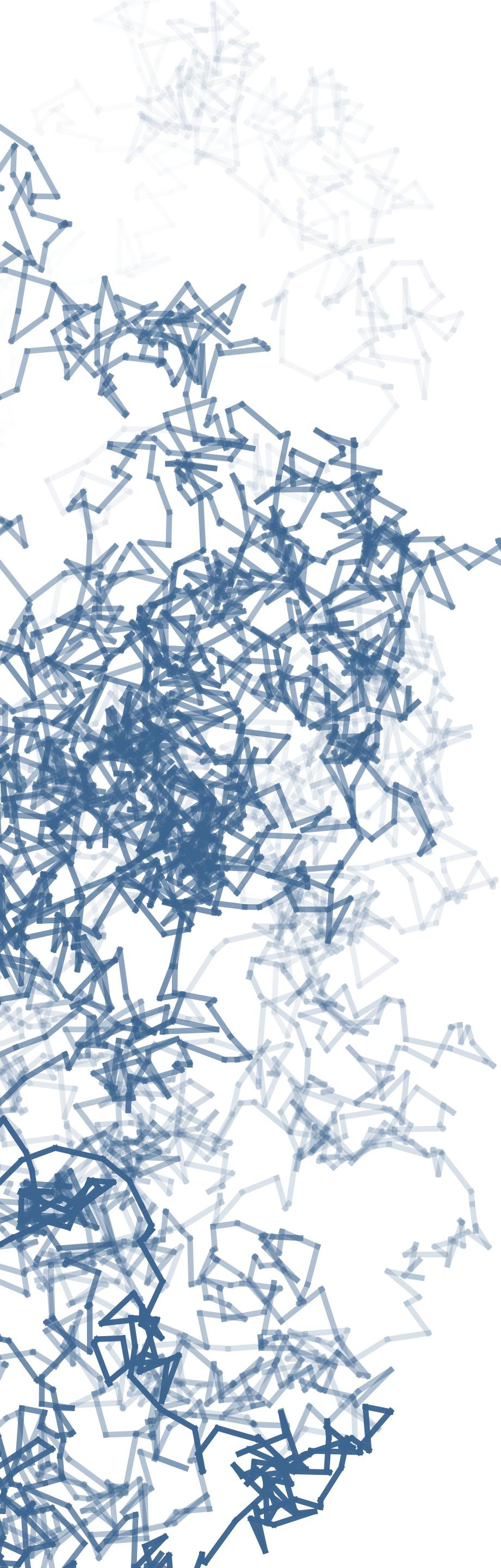


yang-song.net/blog/2021/score/



Flow Matching

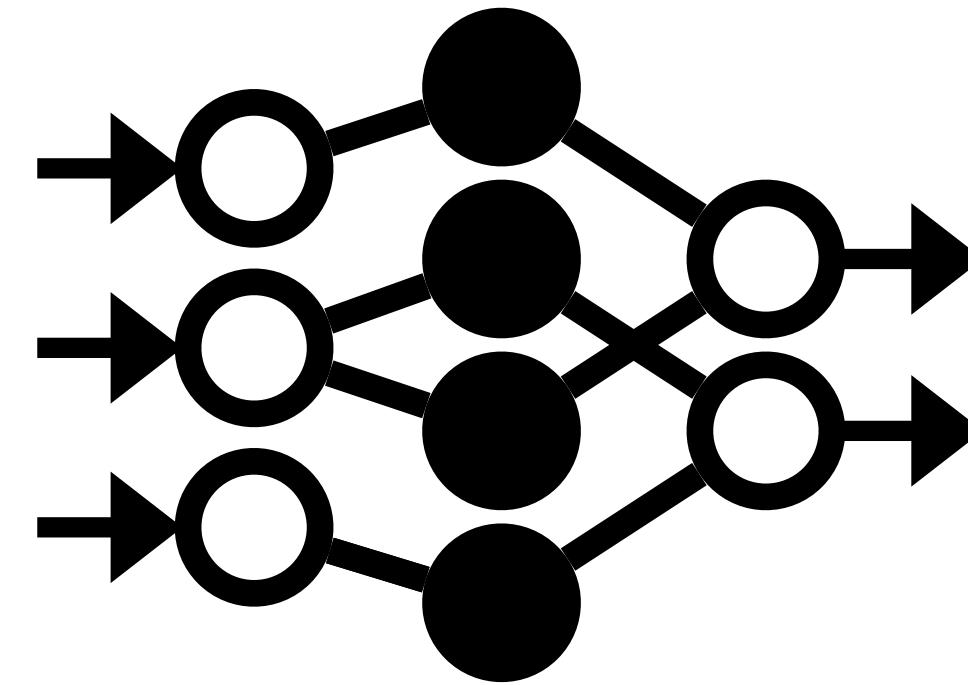
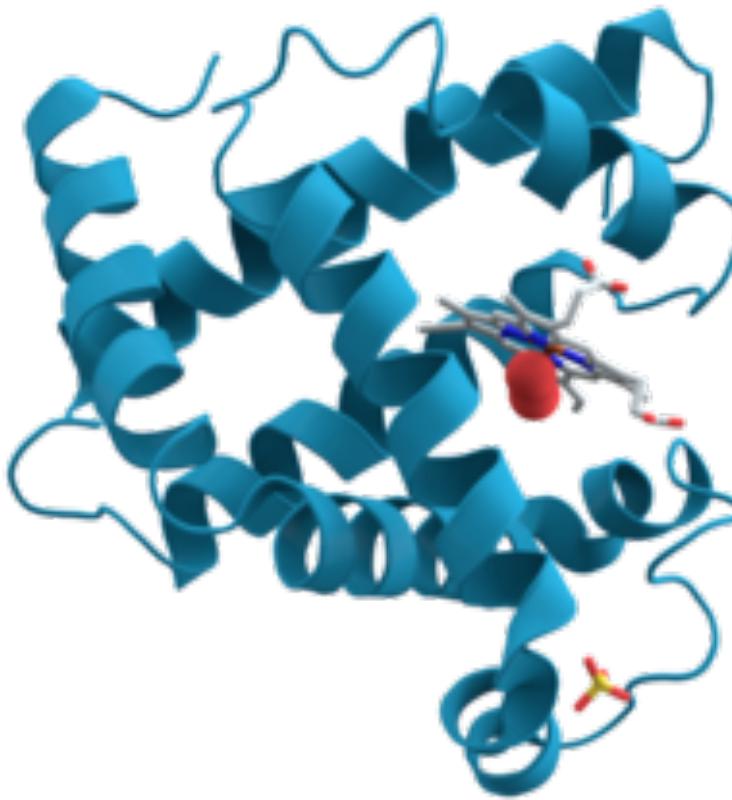
Improving and Generalizing Flow-Based Generative Models with Minibatch Optimal Transport (Tong et al.)



Part III

GRAPH NEURAL NETWORKS

Predictions on Graphs

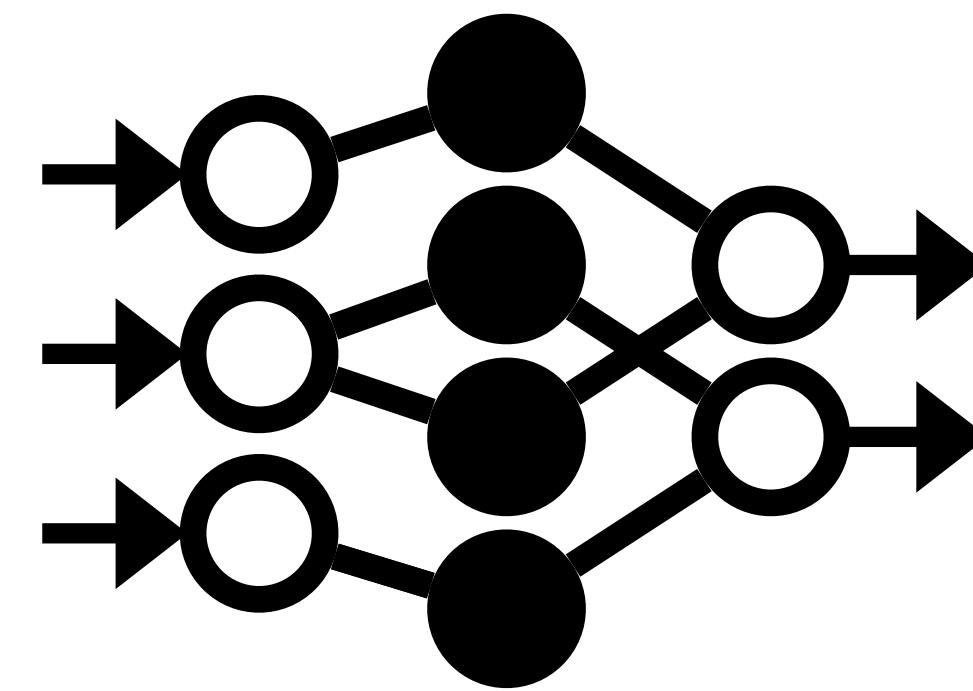
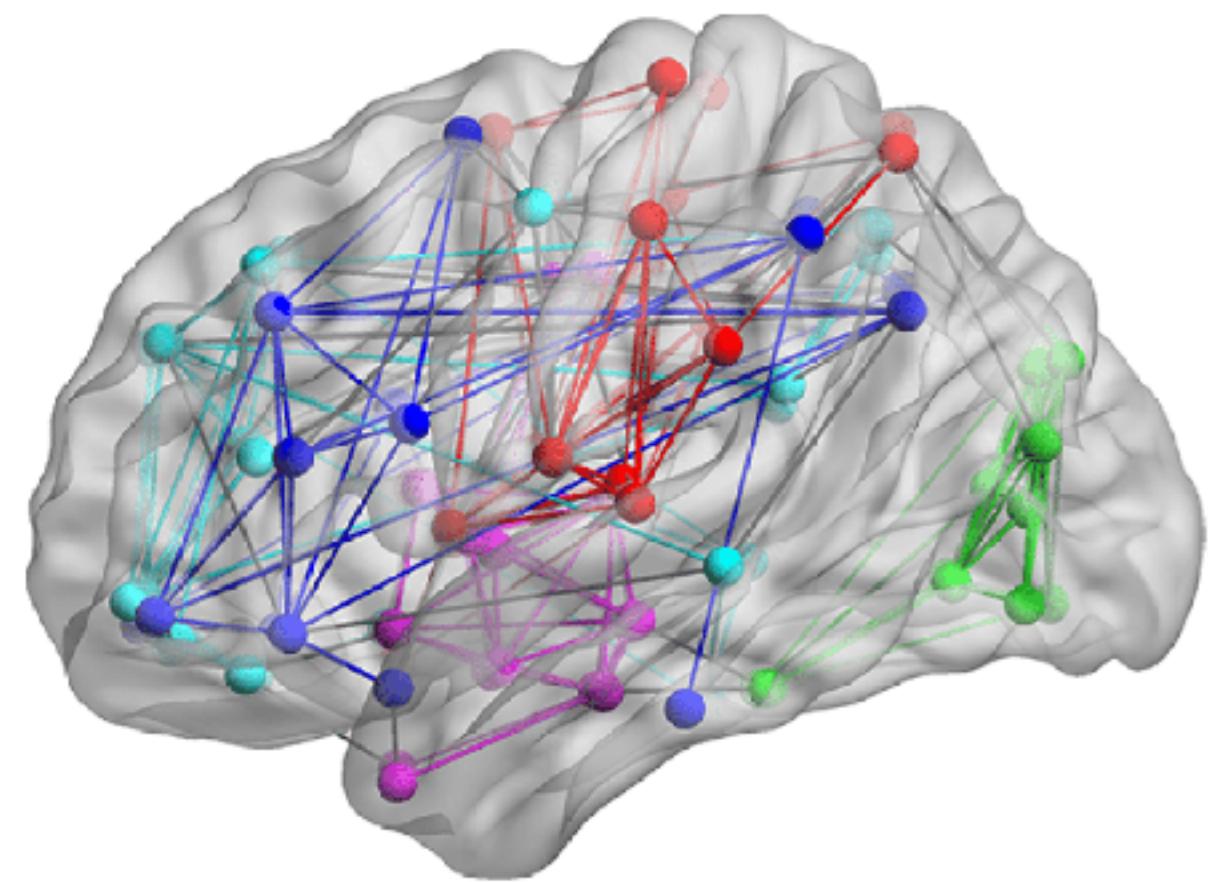


Kinase vs Phosphatase

In many ways, graphs are the main modality
of data we receive from nature.

- Petar Veličković

Predictions on Graphs

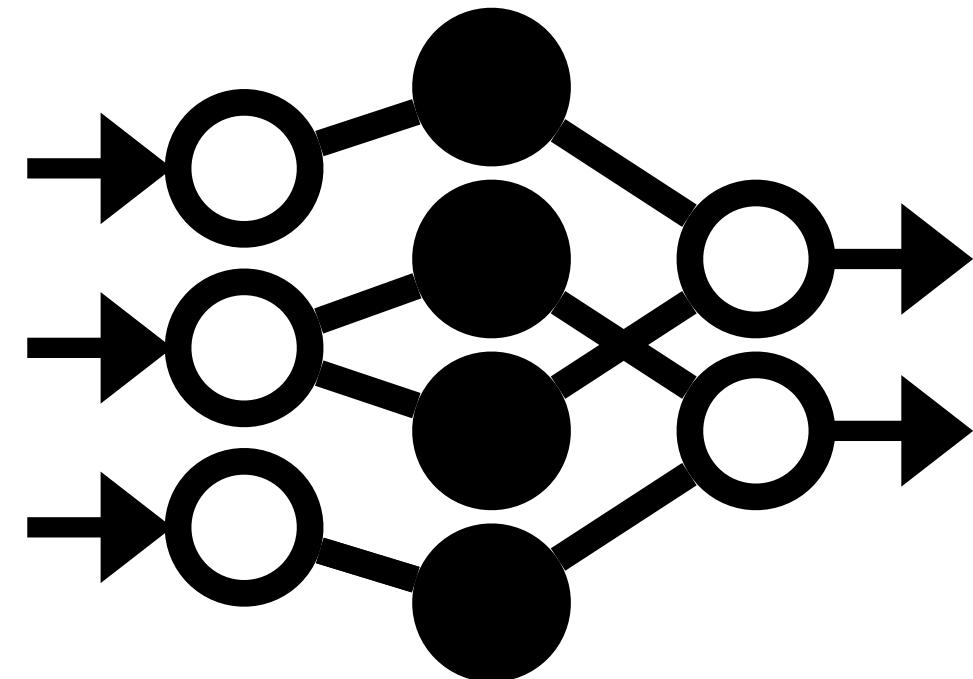
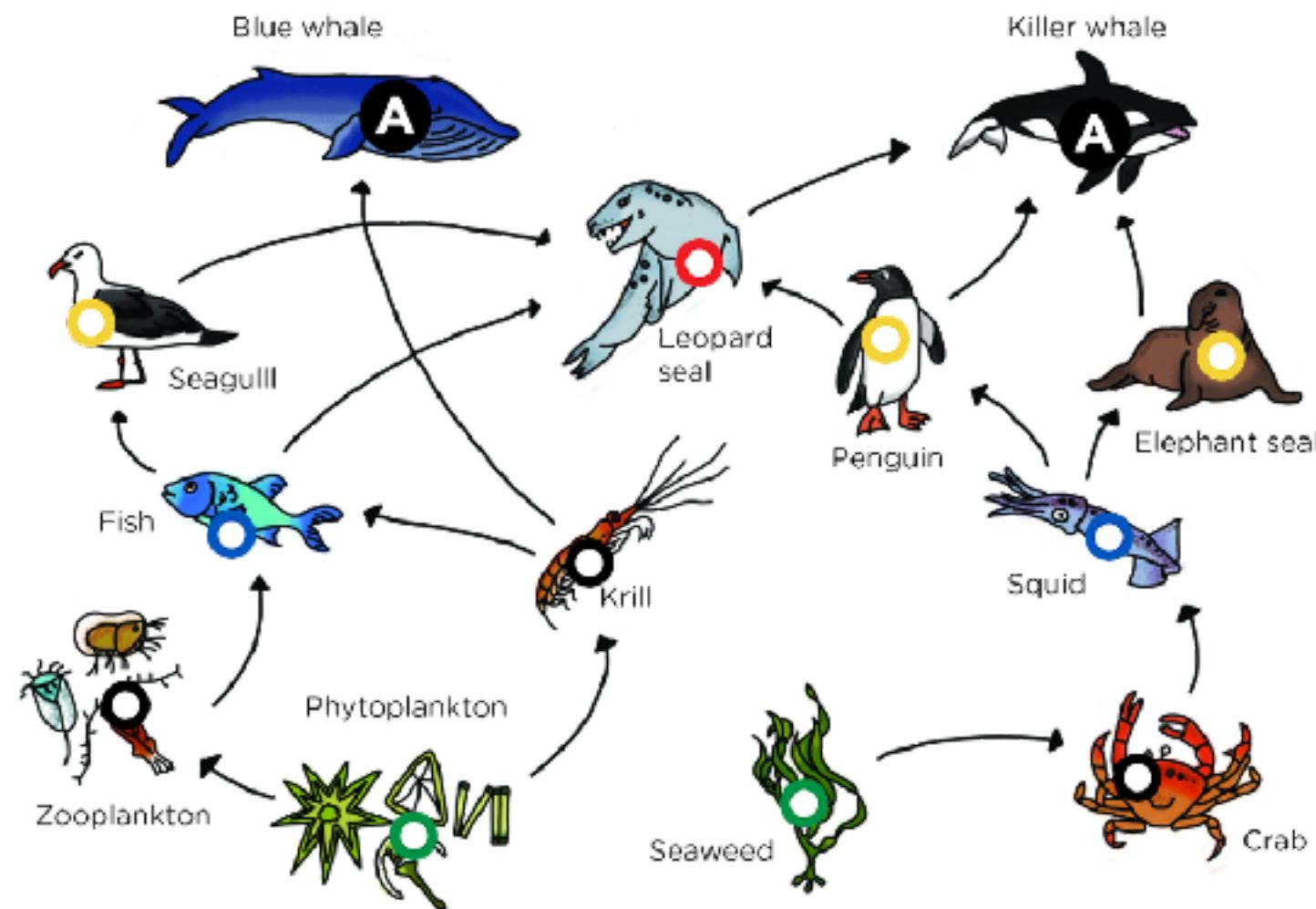


Attentive vs Distracted

In many ways, graphs are the main modality
of data we receive from nature.

- Petar Veličković

Predictions on Graphs

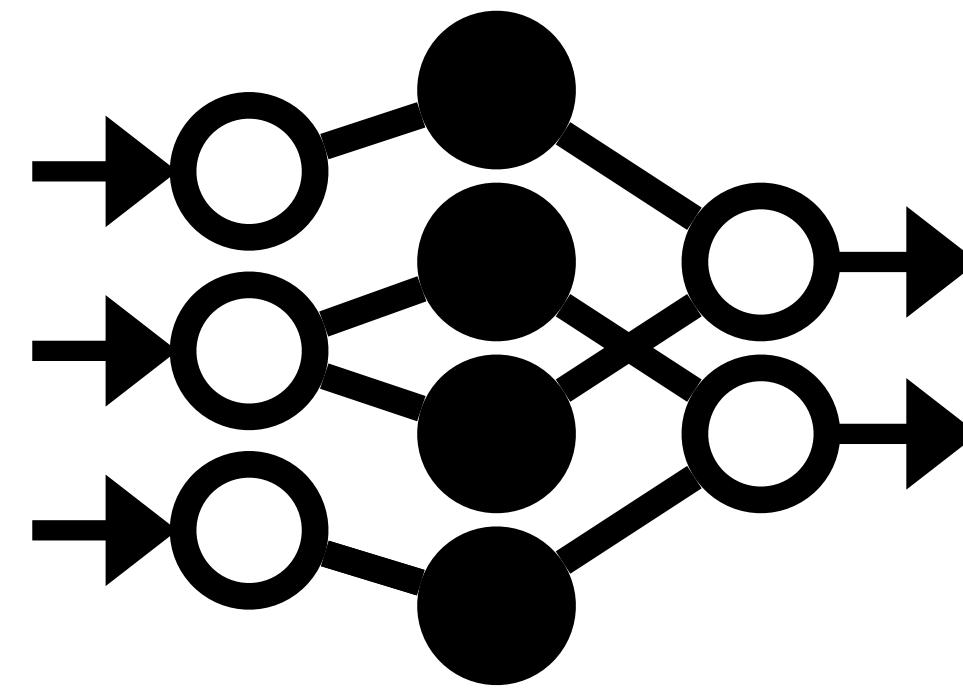
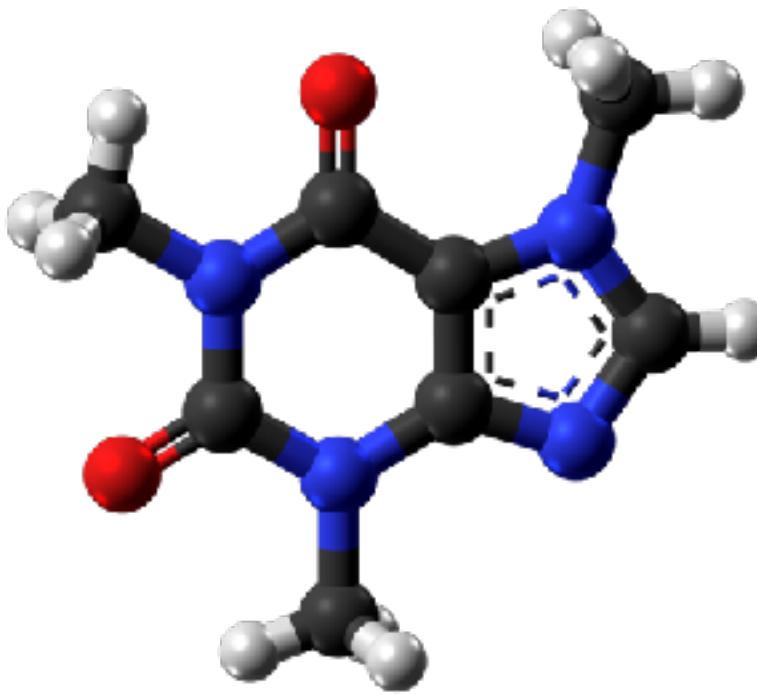


Stable vs Endangered

In many ways, graphs are the main modality
of data we receive from nature.

- Petar Veličković

Predictions on Graphs

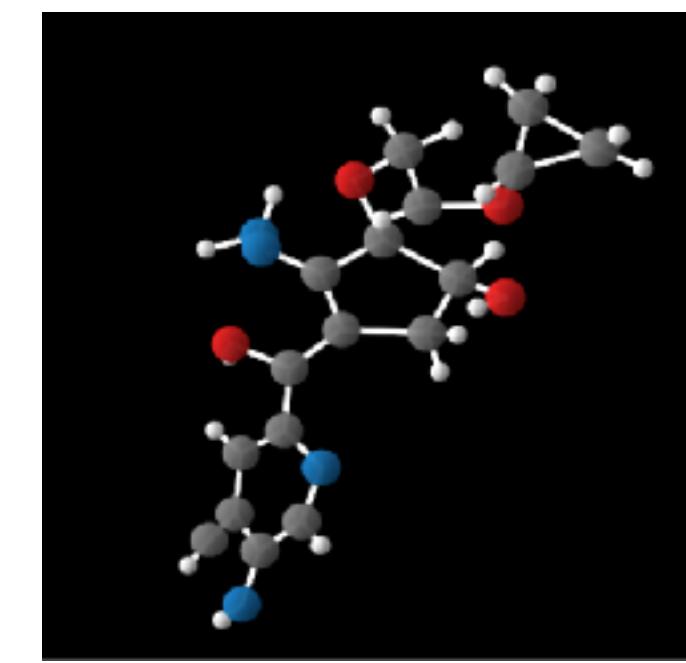
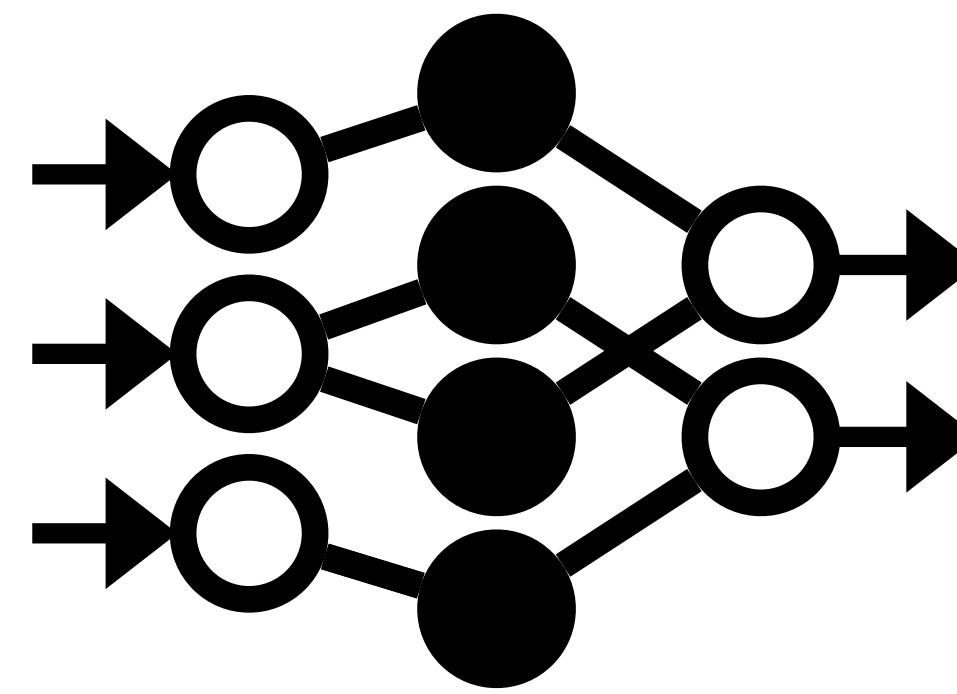
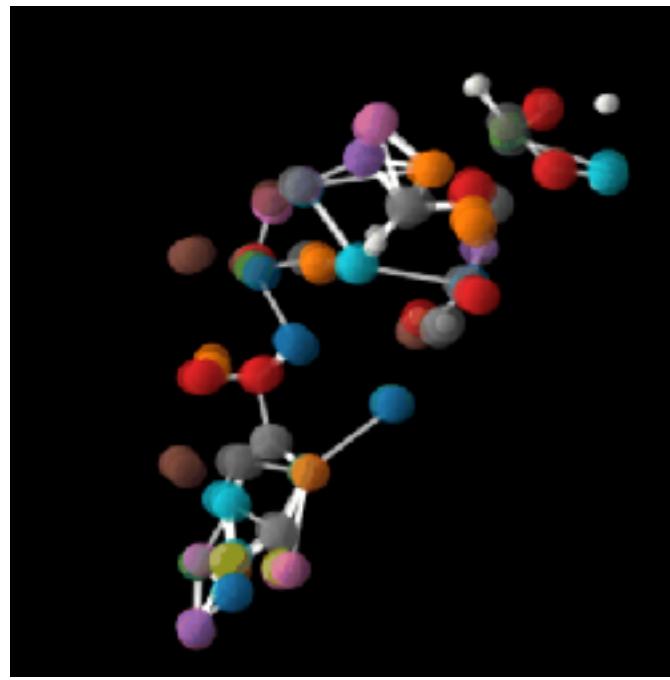


Toxic vs Non-Toxic

In many ways, graphs are the main modality
of data we receive from nature.

- Petar Veličković

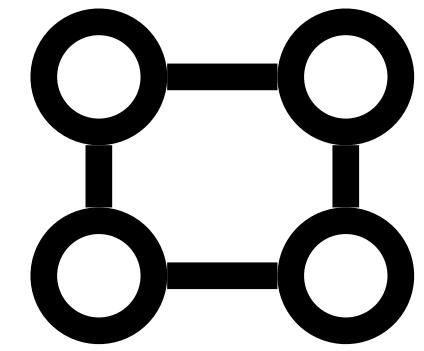
Predictions on Graphs



In many ways, graphs are the main modality
of data we receive from nature.

- Petar Veličković

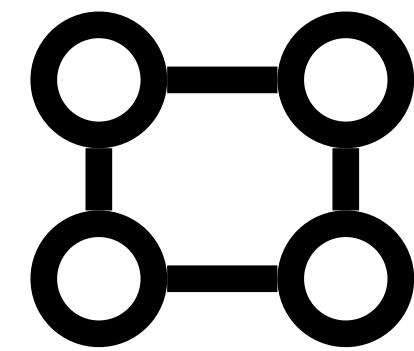
Predictions on Graphs



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

- **Variable Sizes:** Number of nodes and edges varies.
- **Non-Unique Representation:** The same graph can be represented by multiple adjacency matrices

Predictions on Graphs



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

- **Variable Sizes:** Number of nodes and edges varies.
- **Non-Unique Representation:** The same graph can be represented by multiple adjacency matrices

A graph neural network (**GNN**) is a trainable function that is **permutation invariant** or equivariant and takes arbitrary graphs as input.

Permutation Invariance

The diagram illustrates three different permutations of a simple 2x2 graph structure (two nodes connected by two parallel edges) and their corresponding 4x4 adjacency matrices.

Top row:

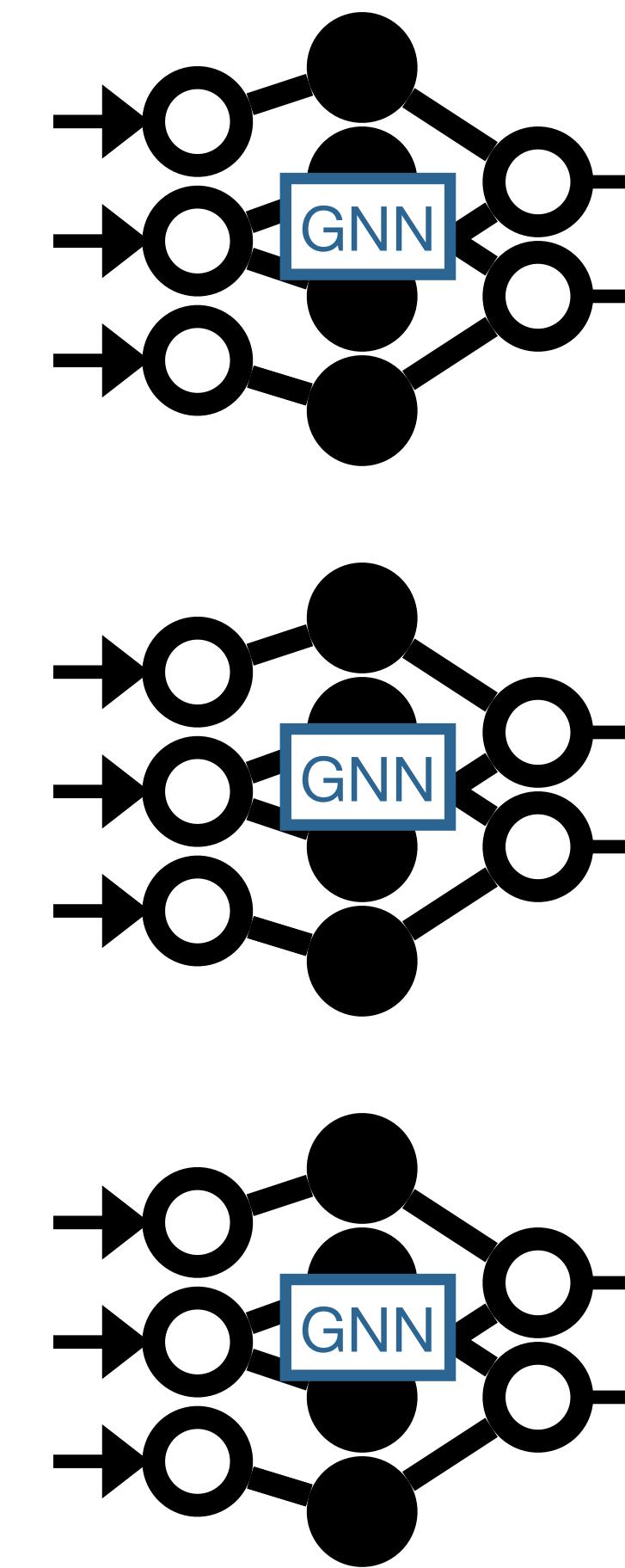
- Original graph: Two nodes connected by two parallel edges.
- Matrix: $\begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}$

Middle row:

- Permuted graph 1: Nodes swapped vertically.
- Matrix: $\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$

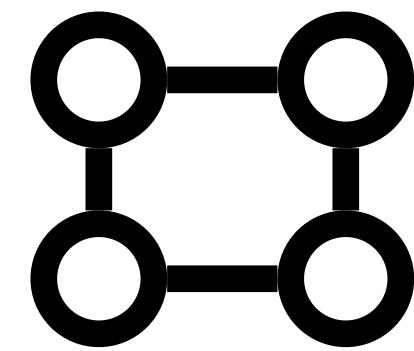
Bottom row:

- Permuted graph 2: Nodes swapped horizontally.
- Matrix: $\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$



Guaranteed to
produce the
same results.

Predictions on Graphs

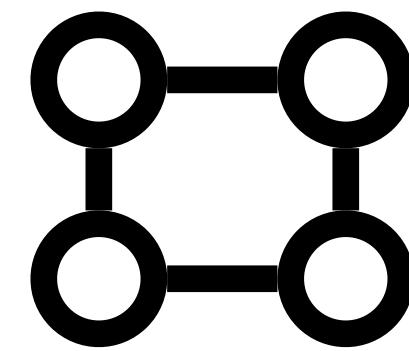


$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

- **Variable Sizes:** Number of nodes and edges varies.
- **Non-Unique Representation:** The same graph can be represented by multiple adjacency matrices

A graph neural network (**GNN**) is a trainable function that is permutation-invariant or **equivariant** and takes arbitrary graphs as input.

Predictions on Graphs



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

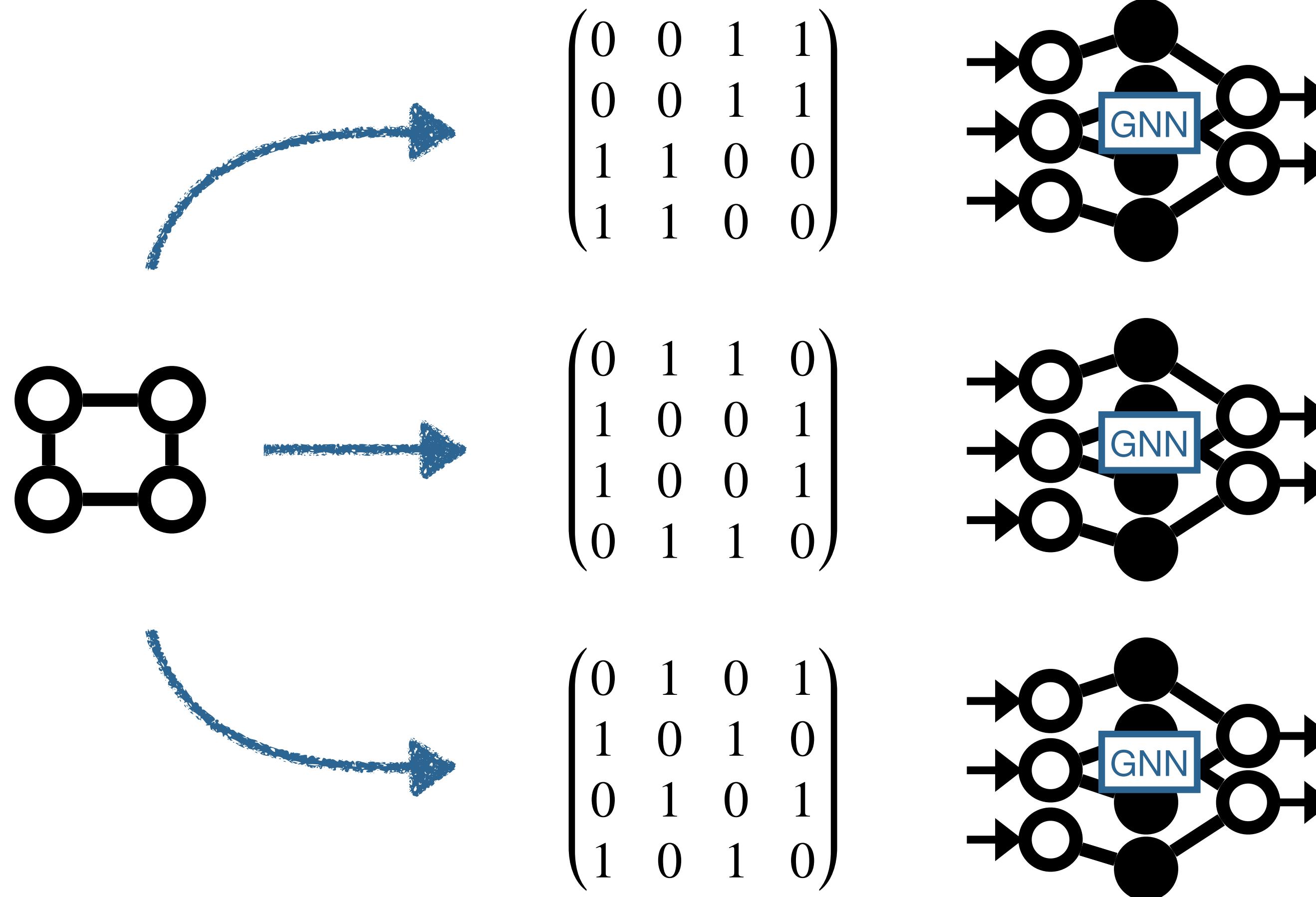
- **Variable Sizes:** Number of nodes and edges varies.
- **Non-Unique Representation:** The same graph can be represented by multiple adjacency matrices

For generating graphs, we want
permutation equivariance.



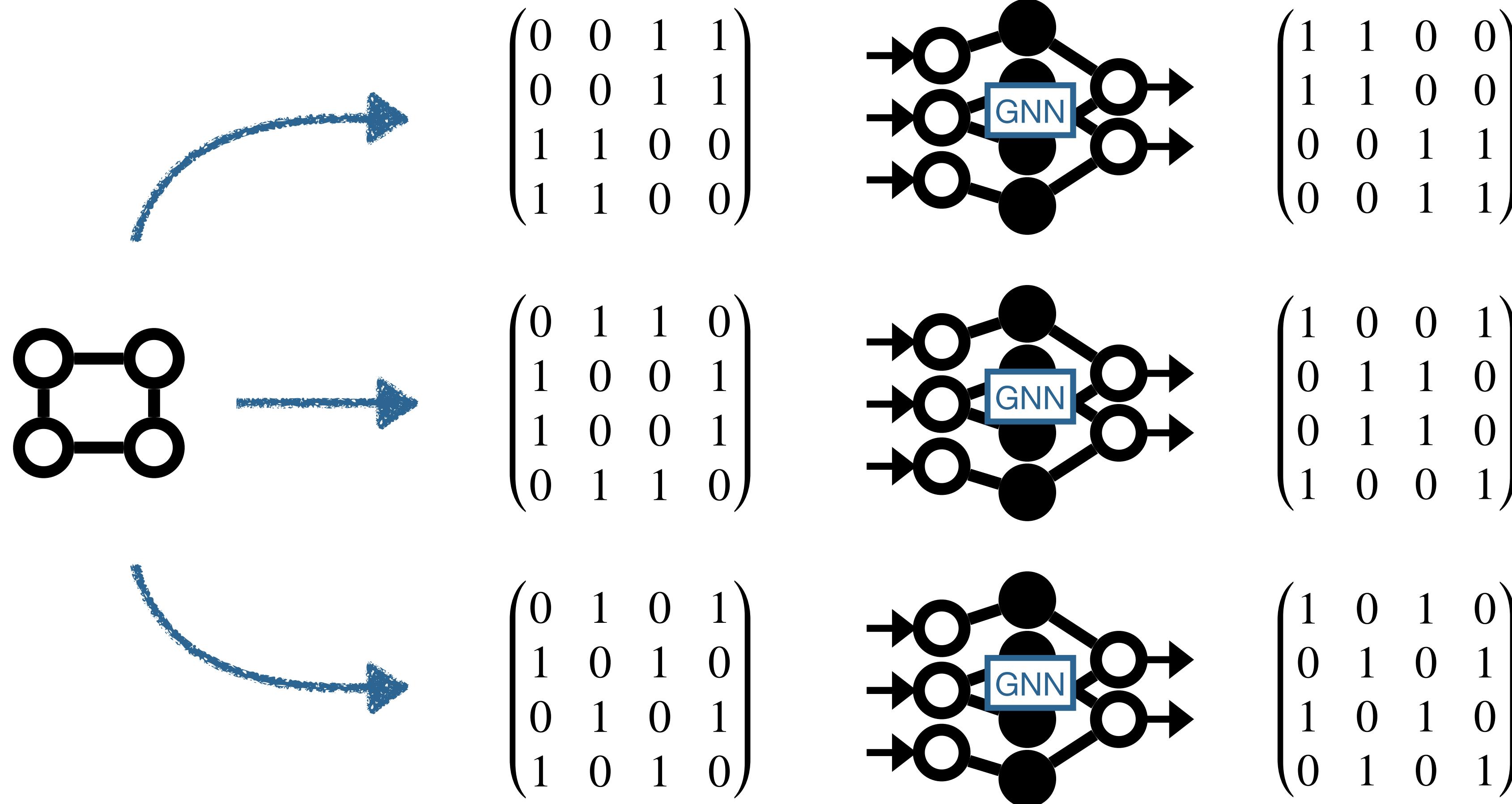
A graph neural network (**GNN**) is a trainable function that is permutation-invariant or **equivariant** and takes arbitrary graphs as input.

Predictions on Graphs



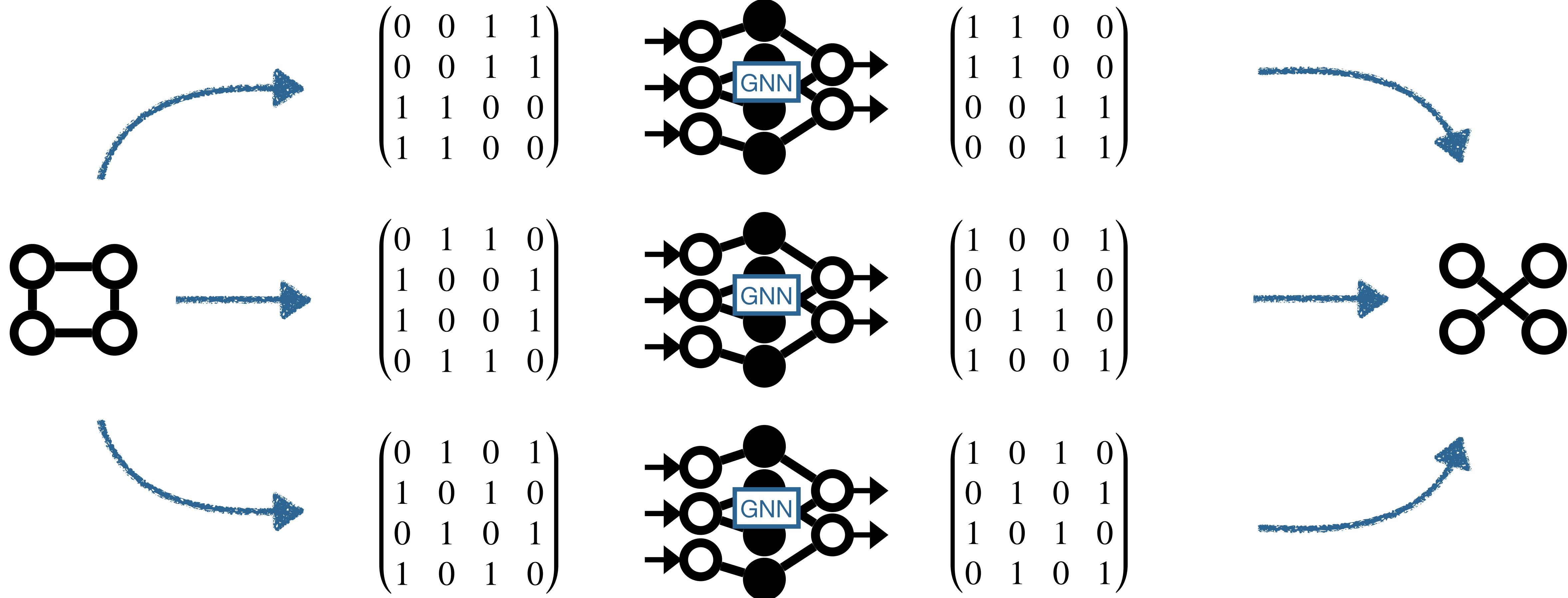
Guaranteed to
produce ~~the same~~
equivalent results.

Predictions on Graphs

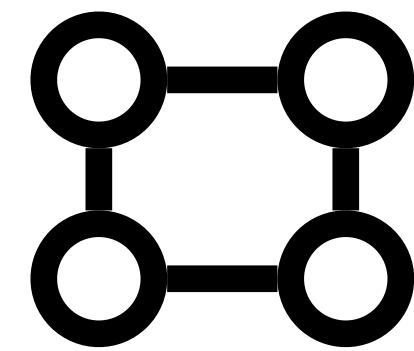


Guaranteed to
produce ~~the same~~
equivalent results.

Predictions on Graphs



Predictions on Graphs



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

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

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

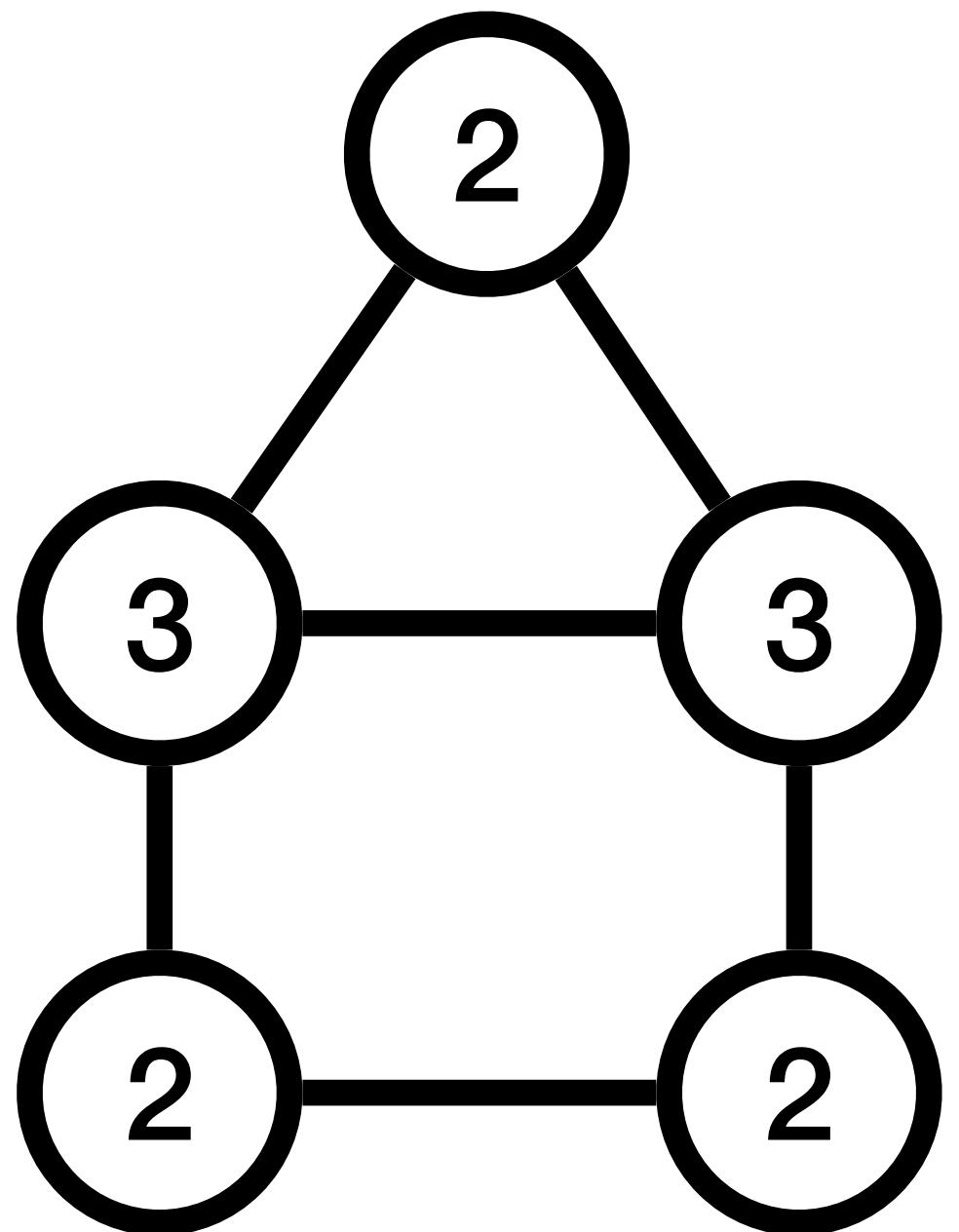
How to build a permutation equivariant GNN?



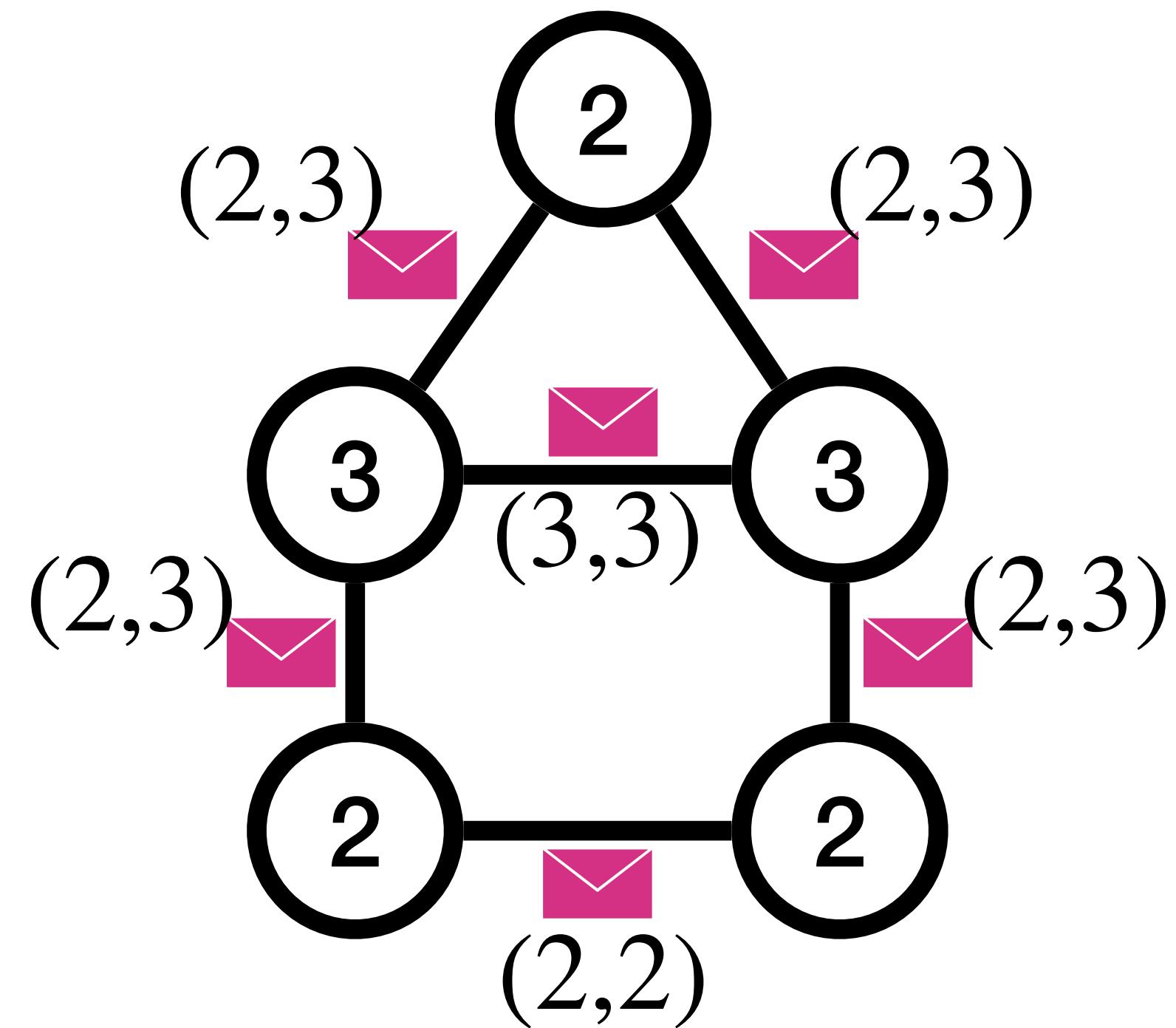
- **Variable Sizes:** Number of nodes and edges varies.
- **Non-Unique Representation:** The same graph can be represented by multiple adjacency matrices

A graph neural network (**GNN**) is a trainable function that is permutation-invariant or **equivariant** and takes arbitrary graphs as input.

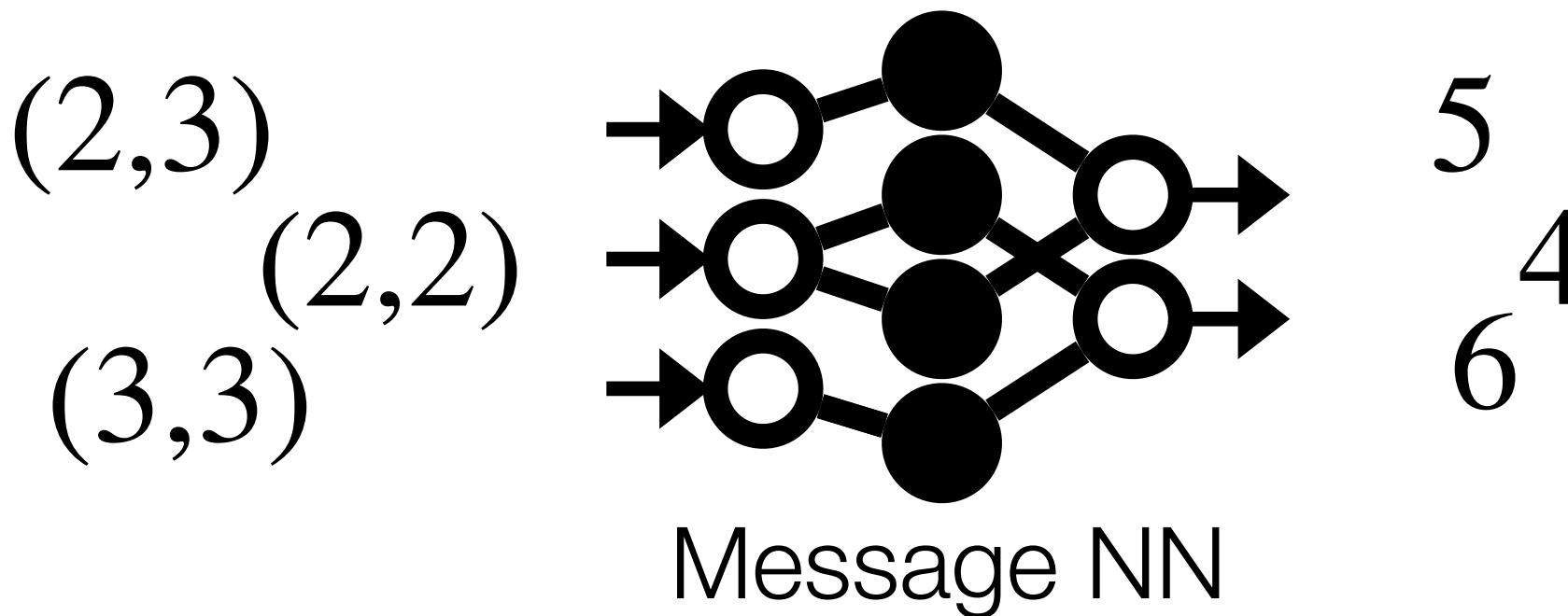
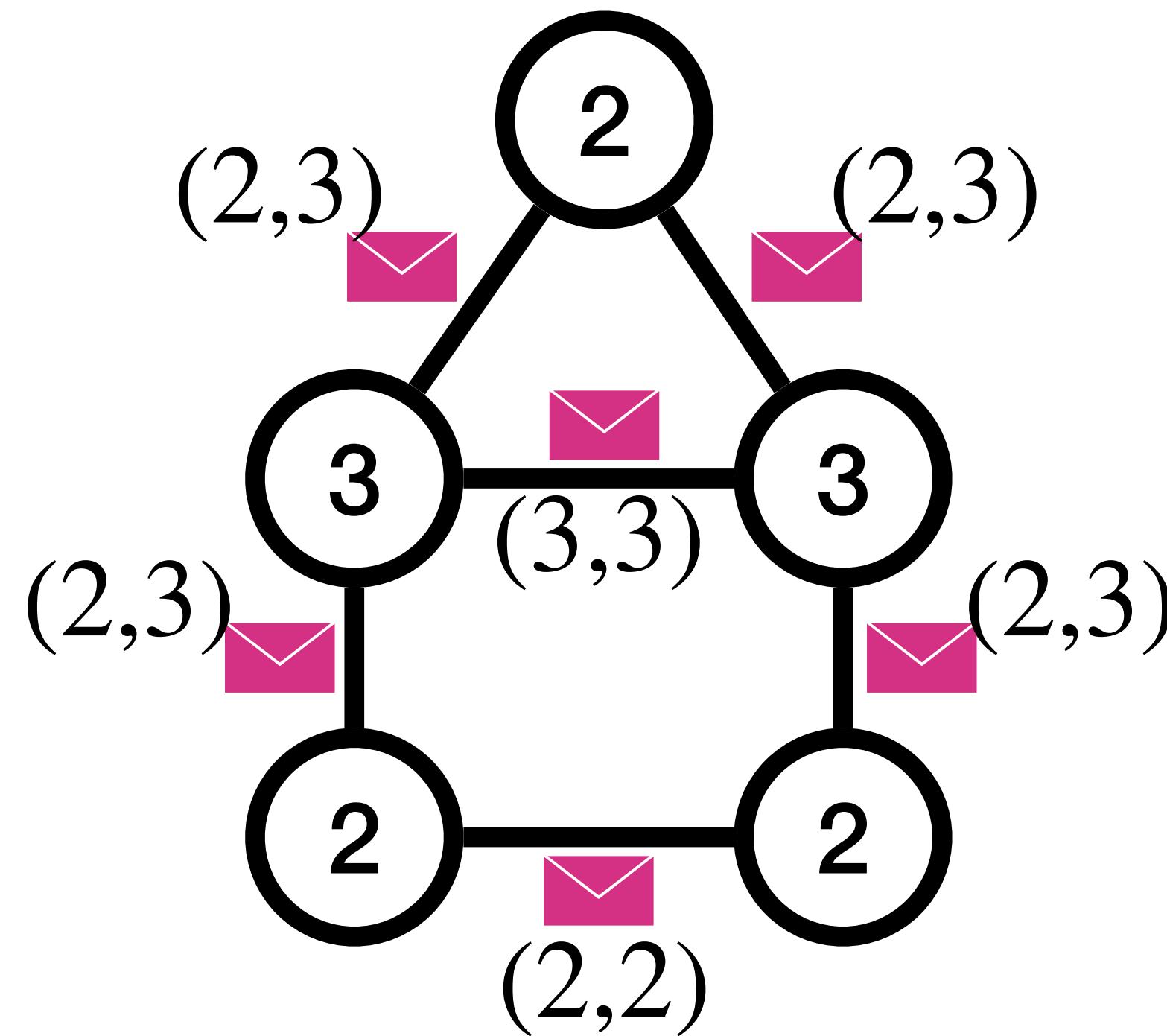
How to Build a GNN?



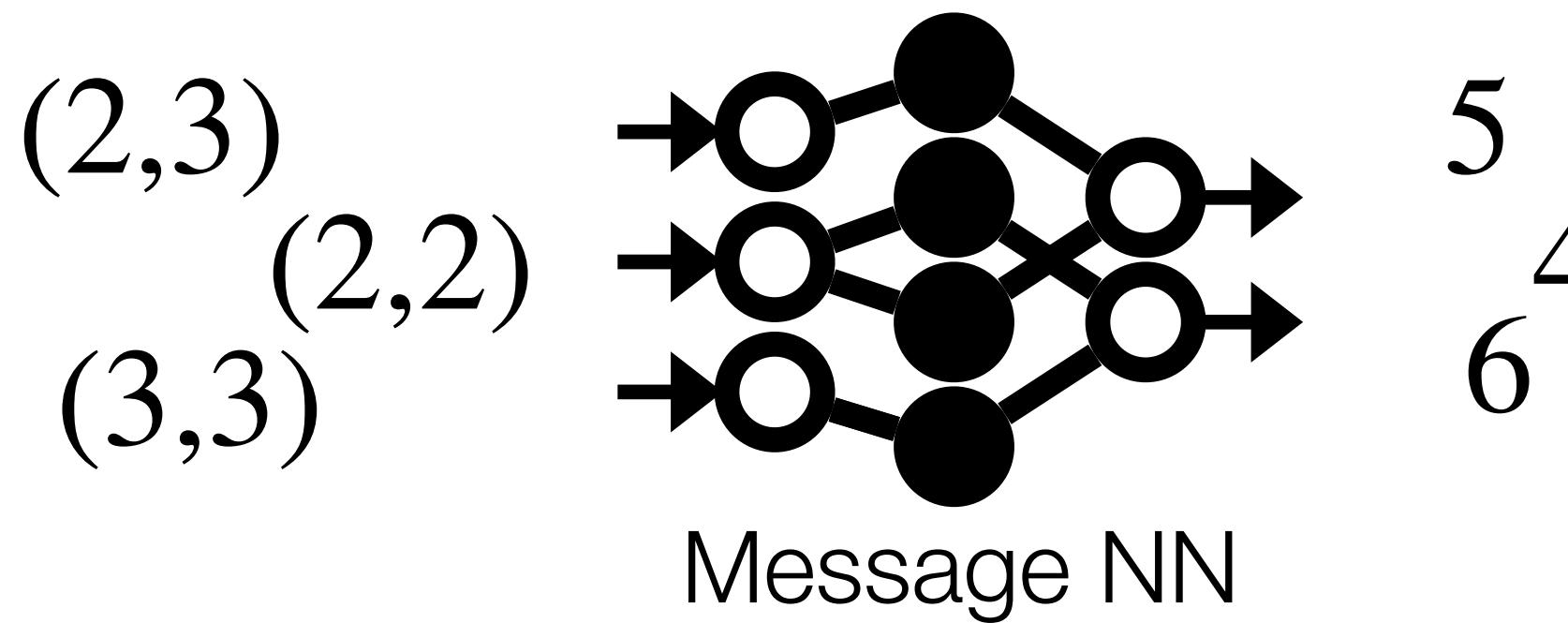
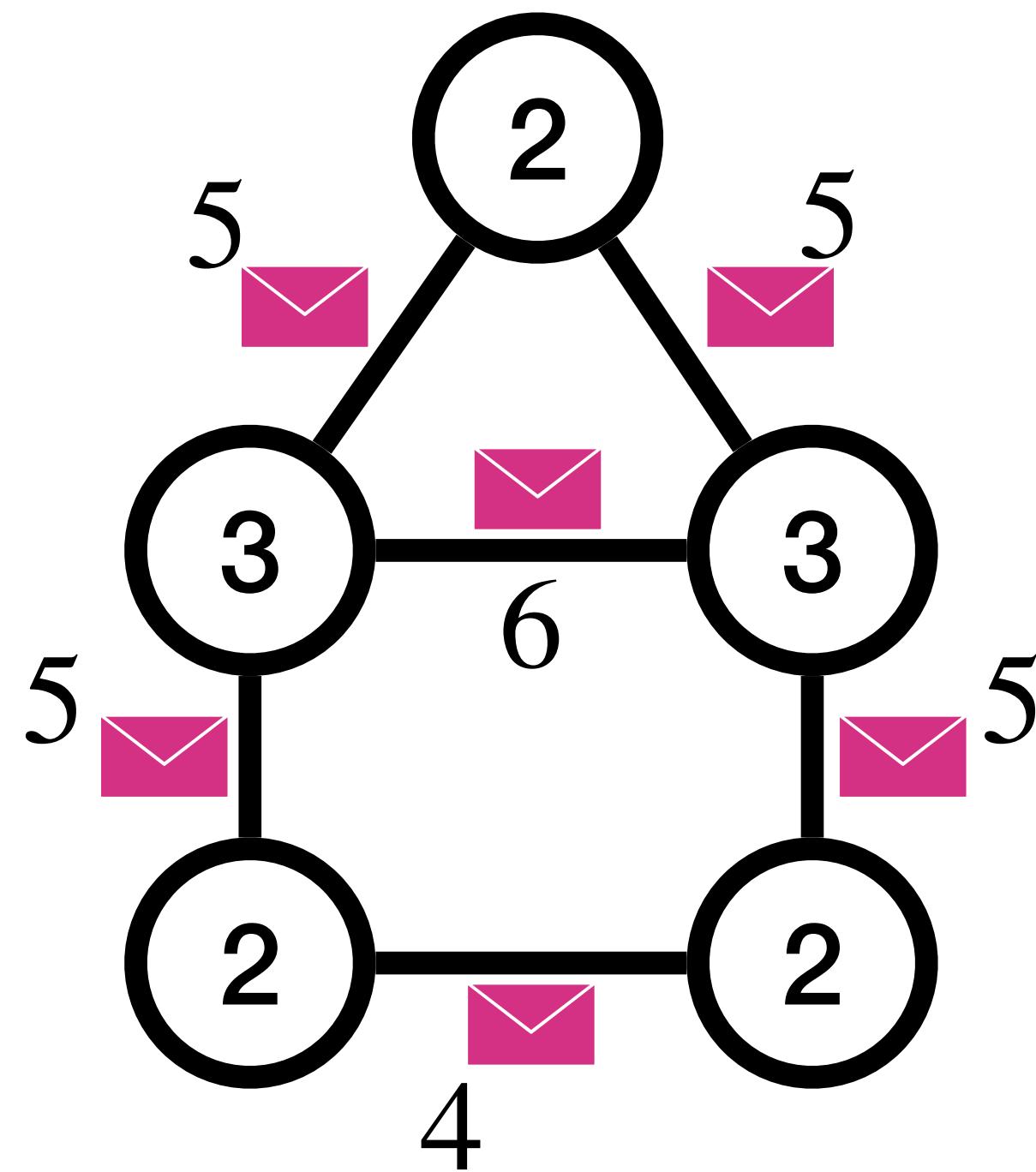
How to Build a GNN?



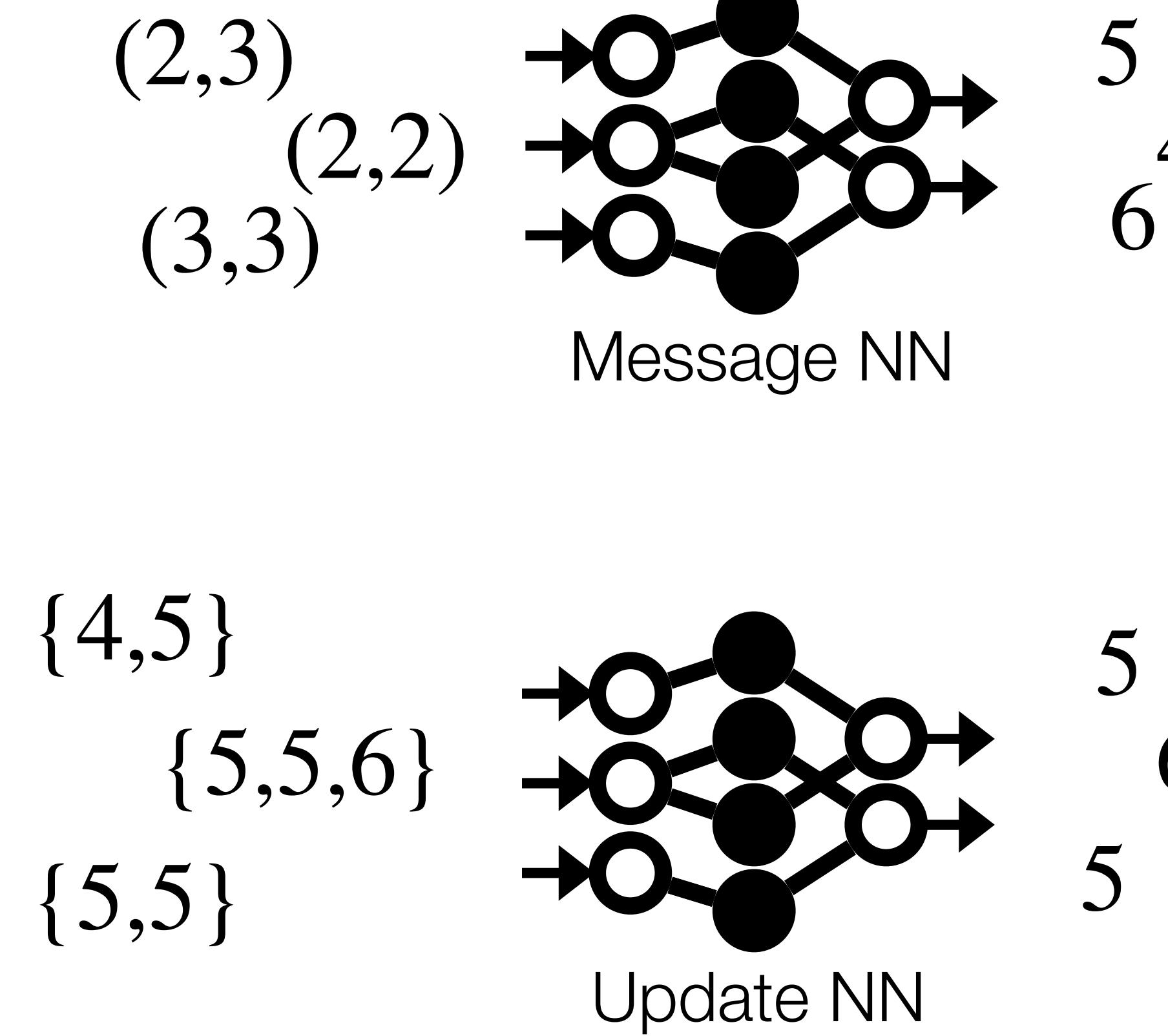
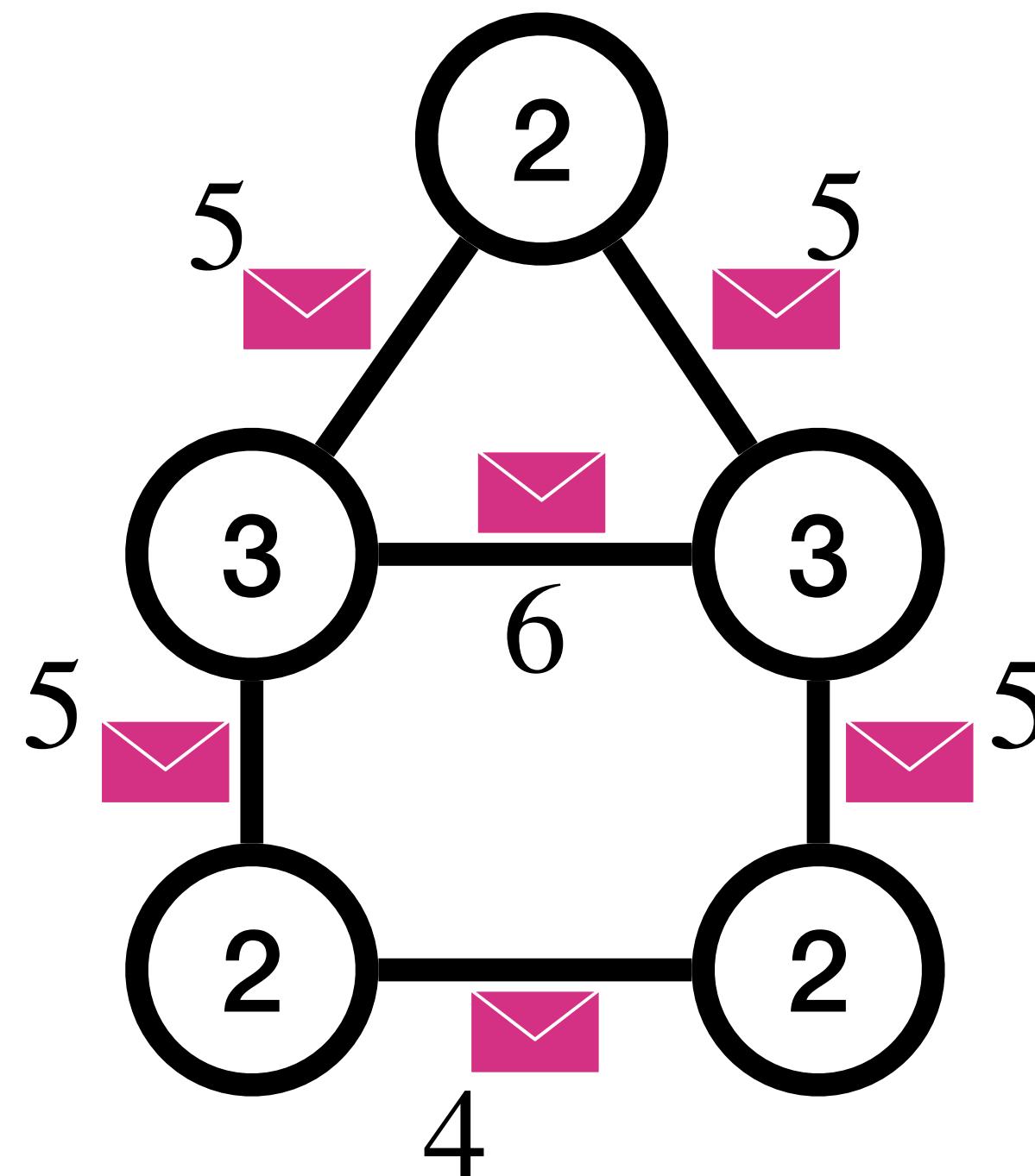
How to Build a GNN?



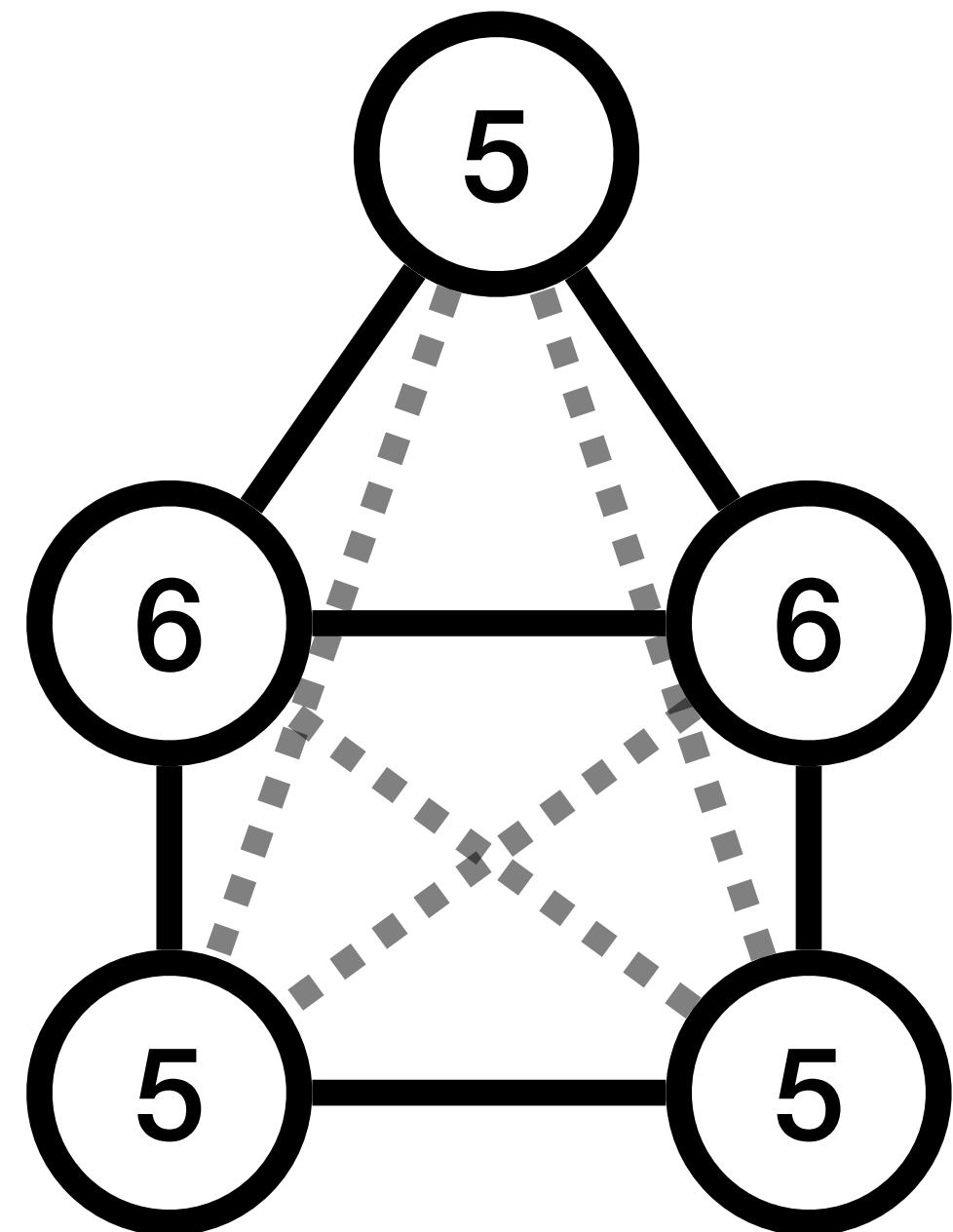
How to Build a GNN?



How to Build a GNN?



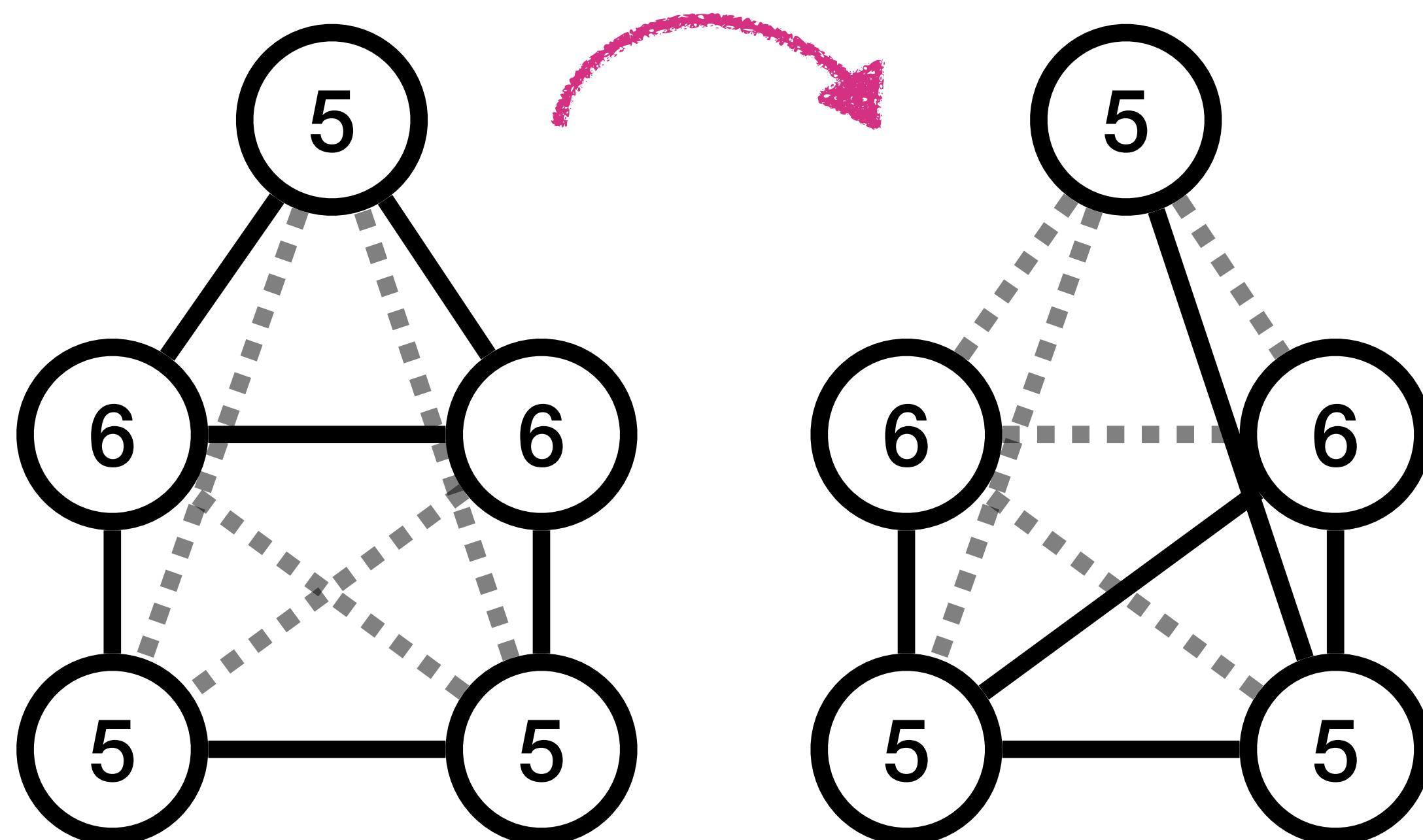
How to Build a GNN?



Graph regression:

- Use complete graph (all nodes are connected)
- Use edge features to encode graph structure (edge vs no-edge)
- Message passing takes edge features into account.

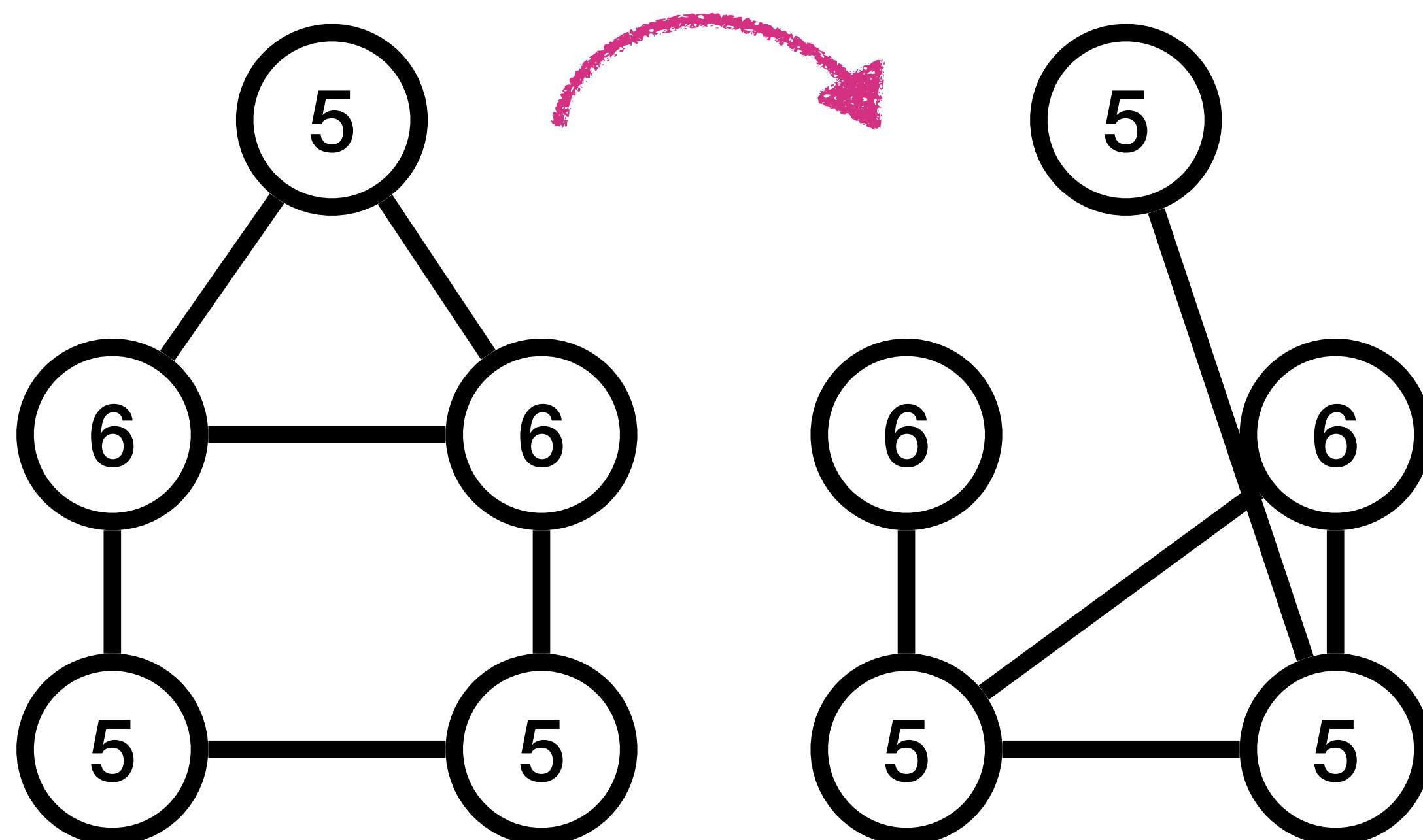
How to Build a GNN?



Graph regression:

- Use complete graph (all nodes are connected)
- Use edge features to encode graph structure (edge vs no-edge)
- Message passing takes edge features into account.

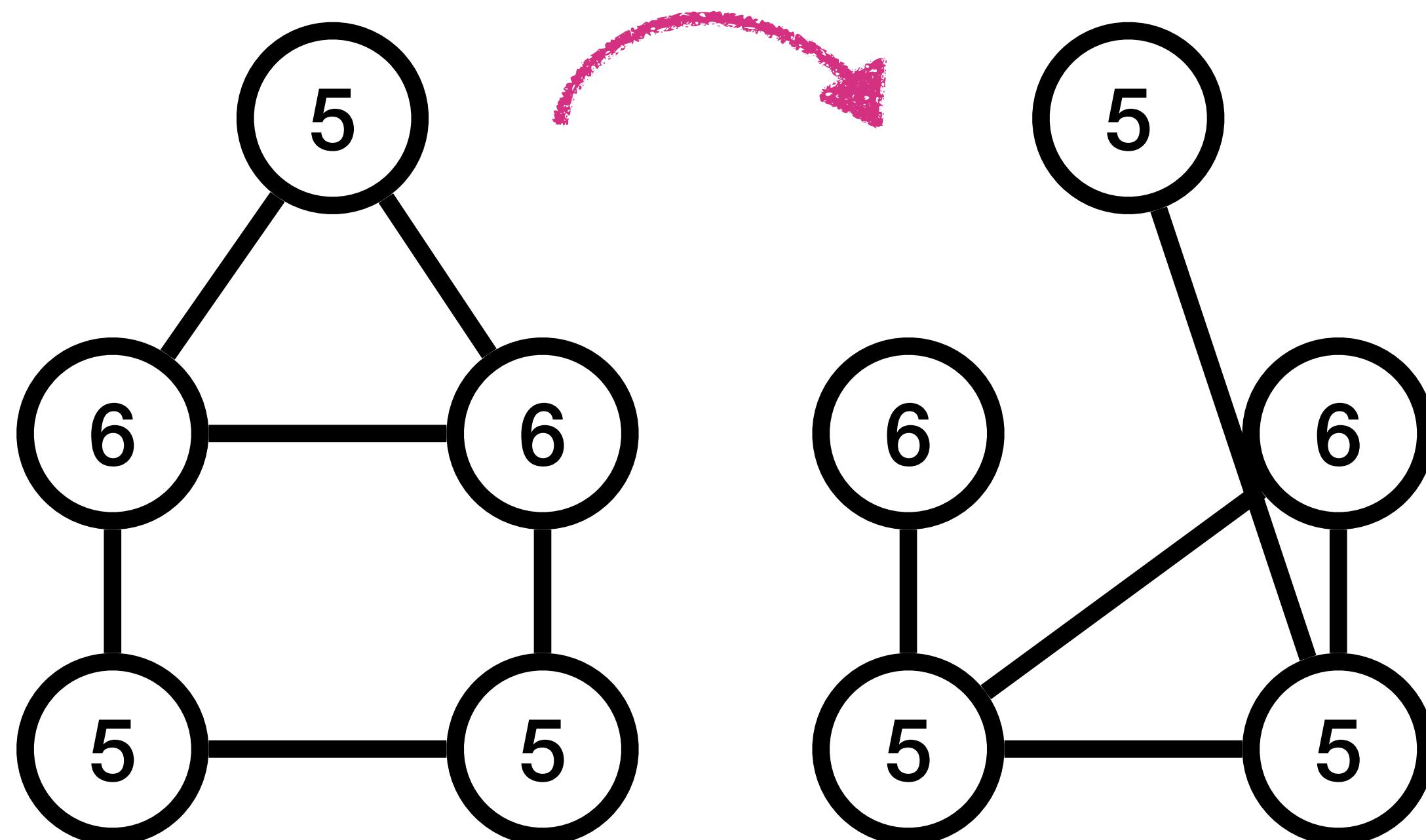
How to Build a GNN?



Graph regression:

- Use complete graph (all nodes are connected)
- Use edge features to encode graph structure (edge vs no-edge)
- Message passing takes edge features into account.

How to Build a GNN?



Properties of Message Passing Neural Networks:

- Theoretically well studied
- Not very powerful in their vanilla form, but easy to improve
- High computational costs on complete graphs

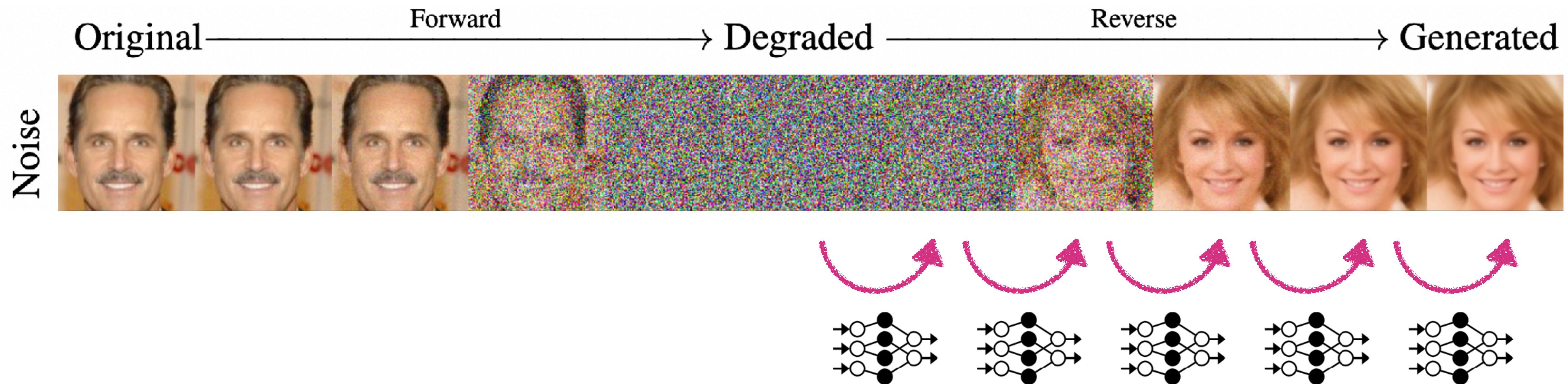
Diffusion on Graphs

1.

Non-trainable part that **removes** information
(stochastically or deterministically)

2.

Trainable part that **reconstructs** information
(typically with a stochastic component)



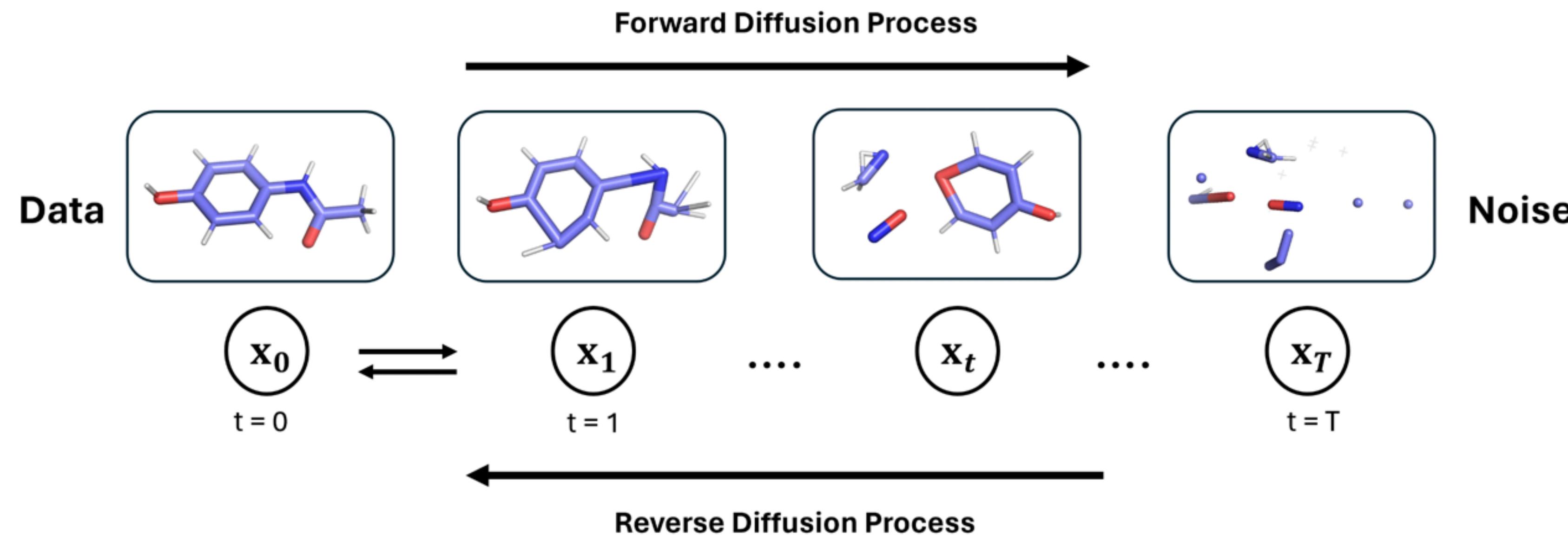
Diffusion on Graphs

1.

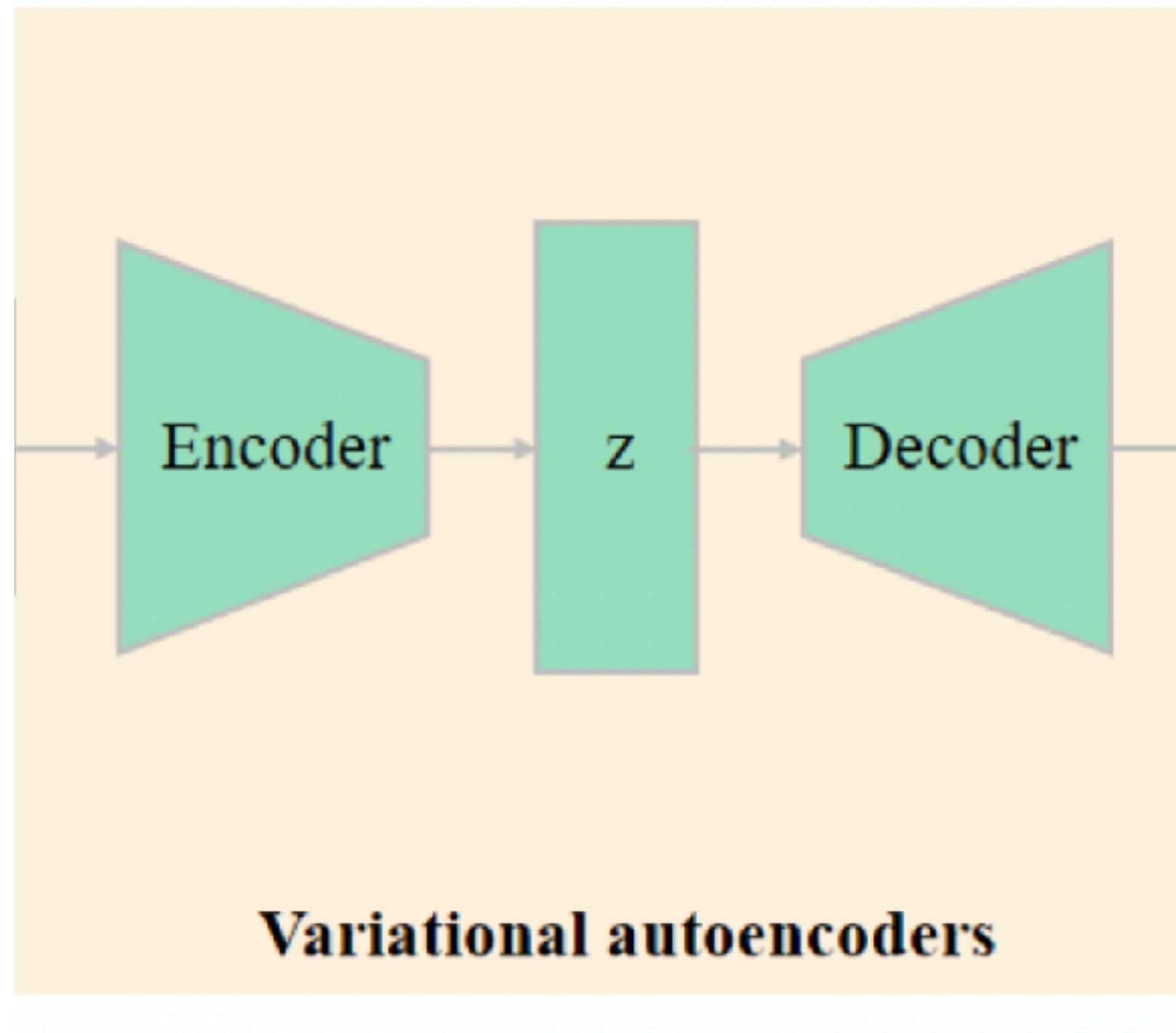
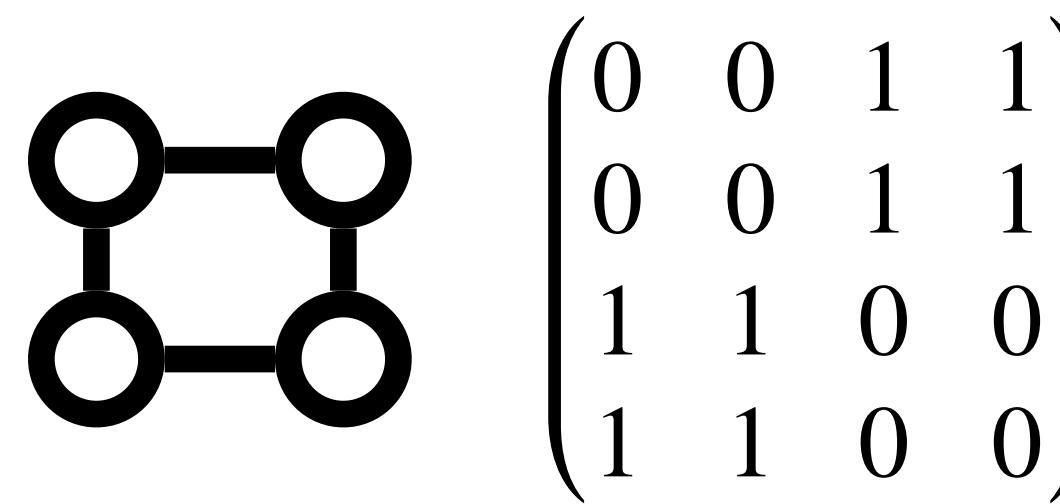
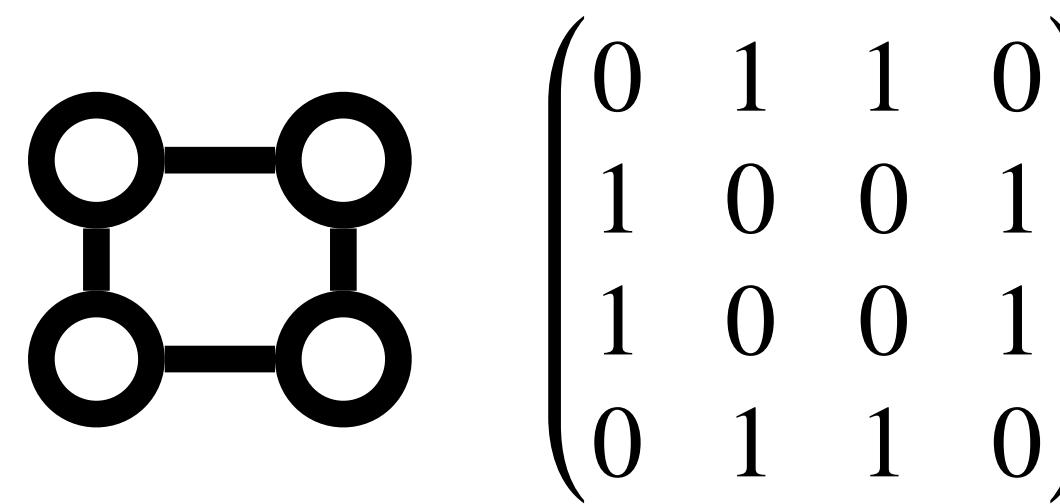
Non-trainable part that **removes** information
(stochastically or deterministically)

2.

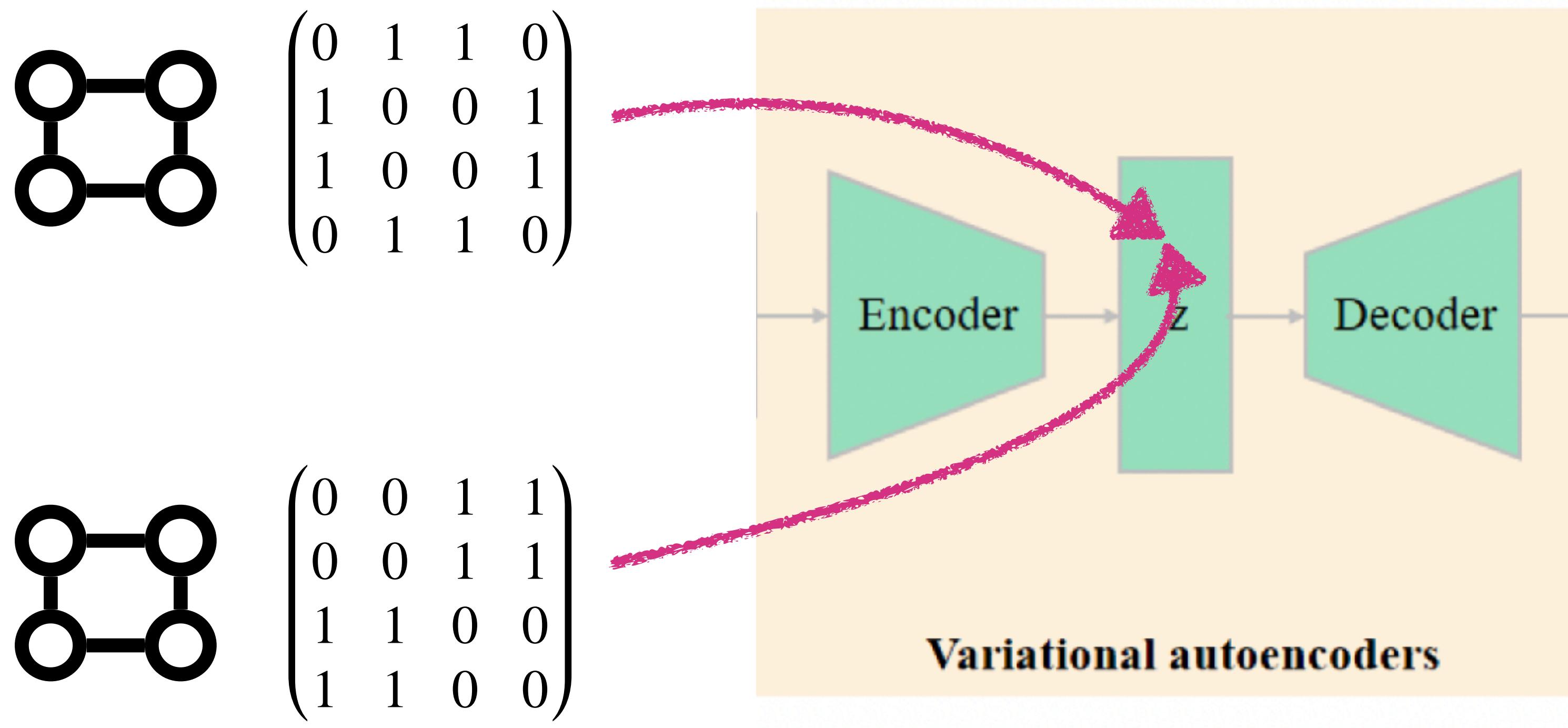
Trainable part that **reconstructs** information
(typically with a stochastic component)



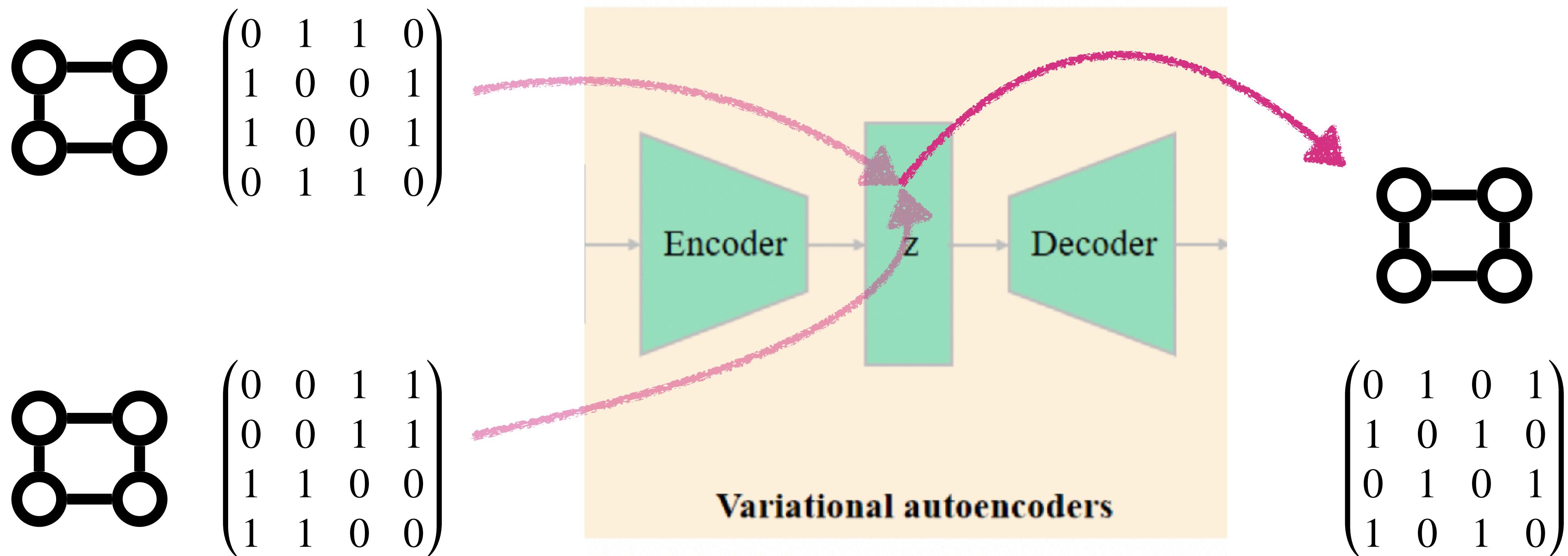
Diffusion Does Not Require Graph Matching



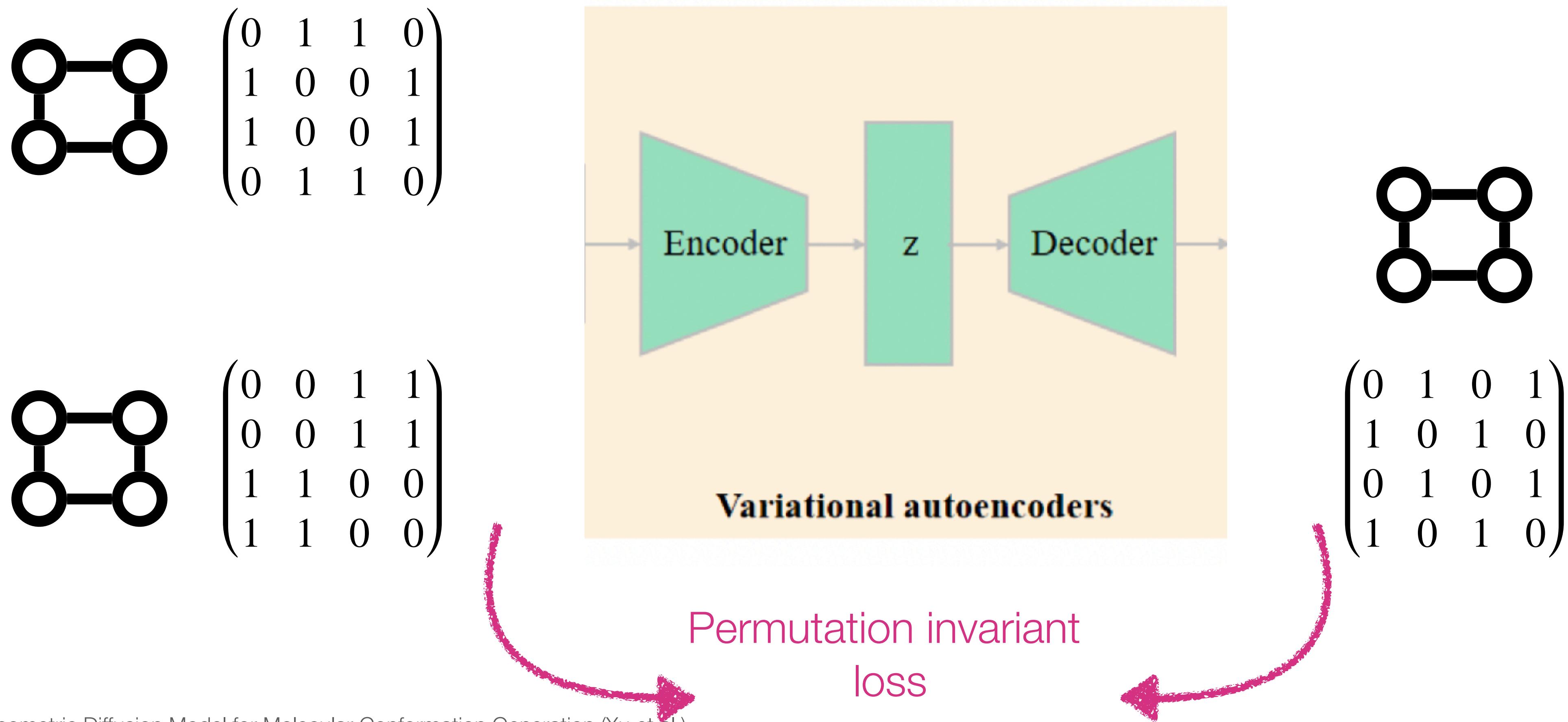
Diffusion Does Not Require Graph Matching



Diffusion Does Not Require Graph Matching



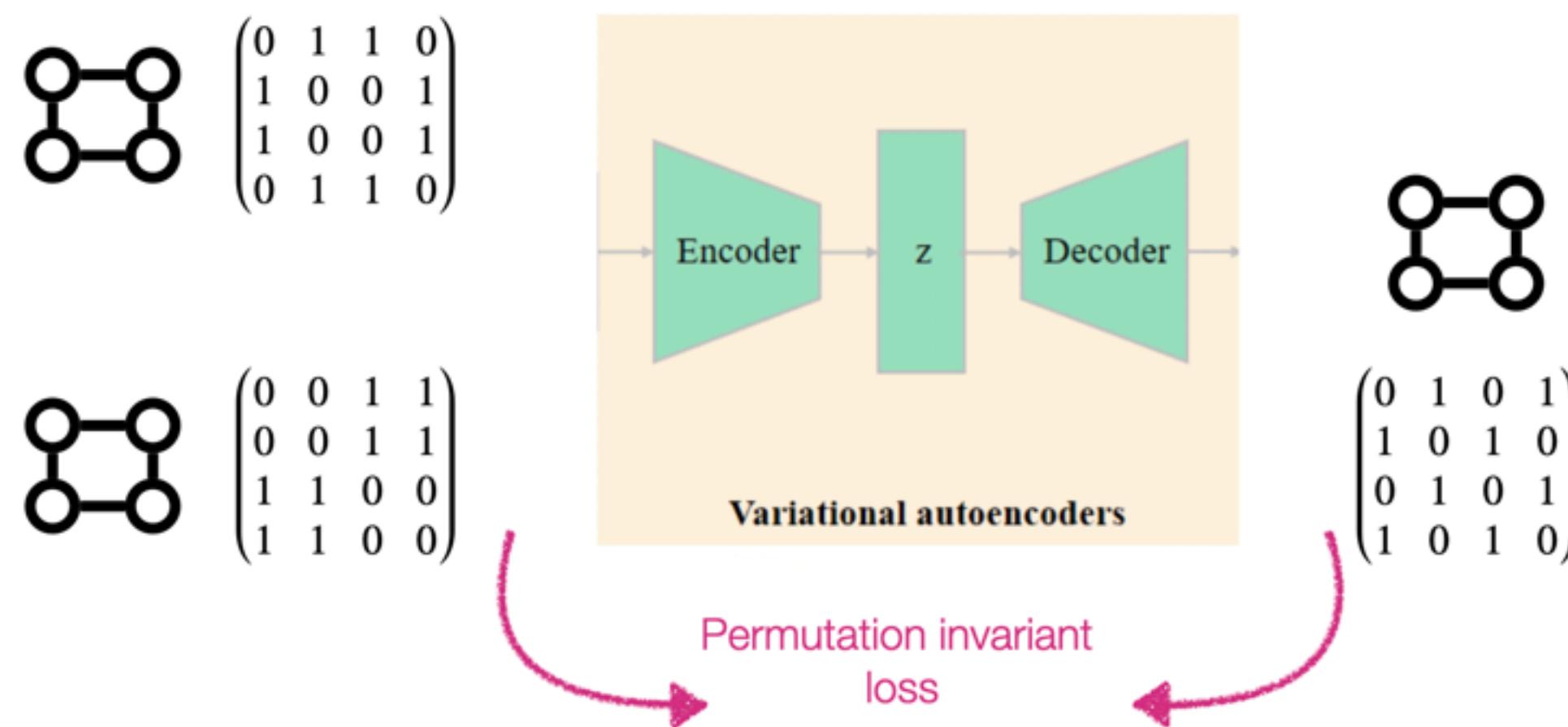
Diffusion Does Not Require Graph Matching



GeoDiff: A Geometric Diffusion Model for Molecular Conformation Generation (Xu et al.)

Equivariant Flows: Exact Likelihood Generative Learning for Symmetric Densities (Koehler et al.)

Diffusion Does Not Require Graph Matching



Diffusion keeps track of node identities:

Use **permutation equivariant** transformations

+

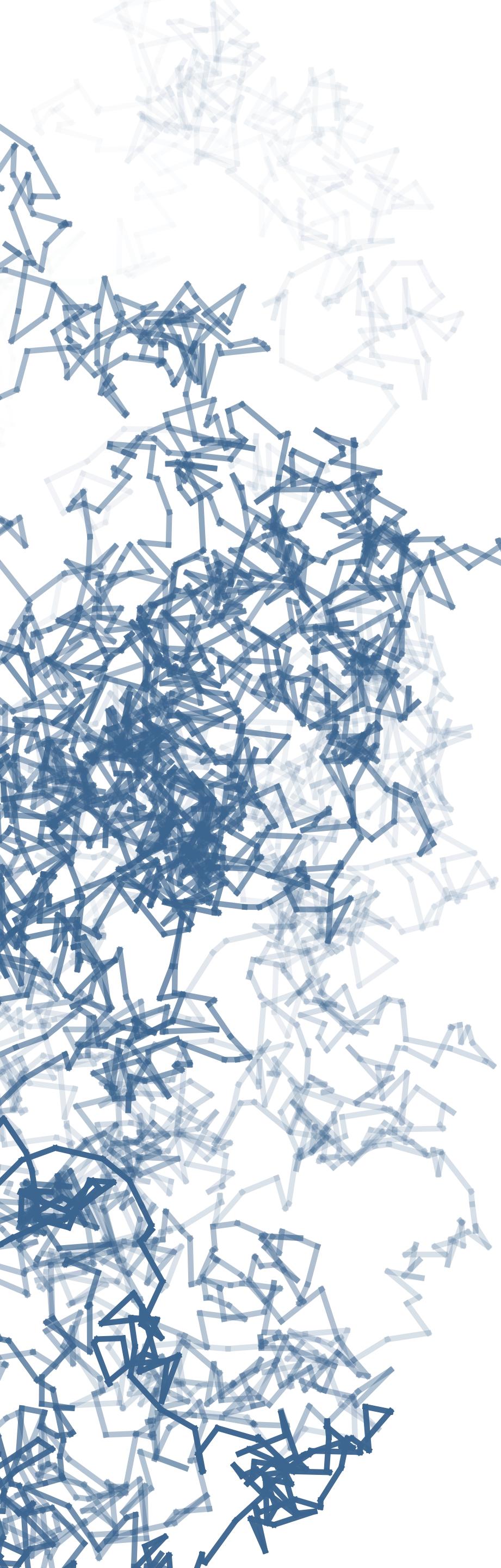
Use **element-wise loss** function

+

Reasonable noise distribution

=

Generation is faithful to invariance of node ordering in the data distribution

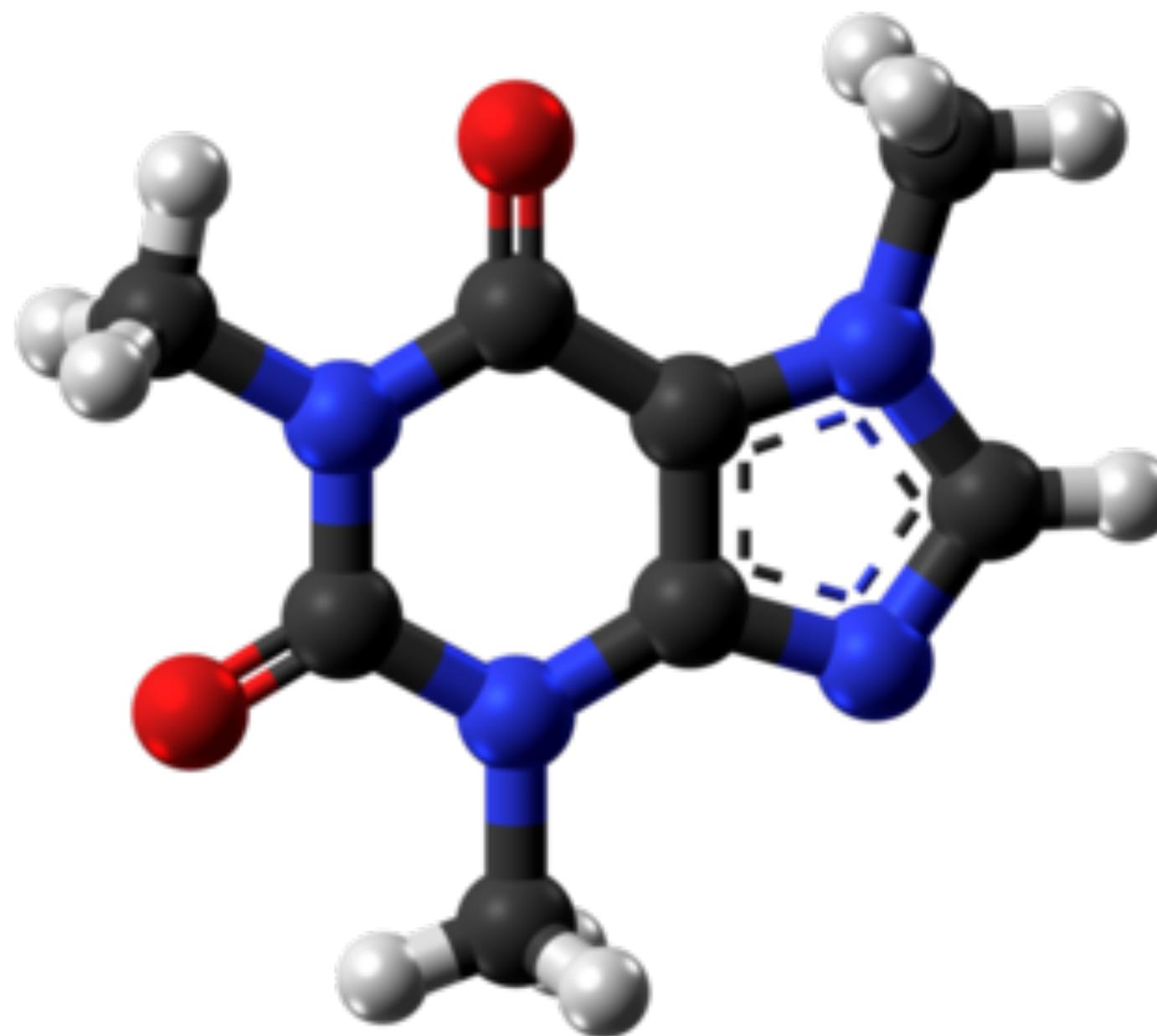


Part IV

DIFFUSION FOR DRUG DESIGN

Molecules as Graphs

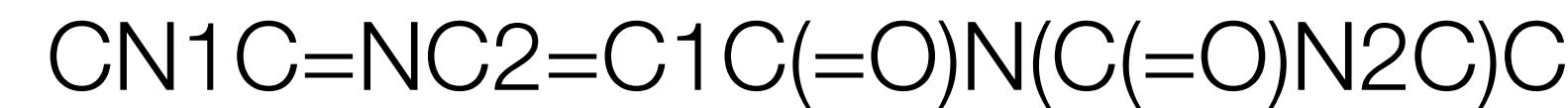
Molecule



- group of atoms connected through chemical bonds
- electrostatic force shapes geometry
- properties emerge from its constituent atoms
- dynamic and complex three-dimensional objects

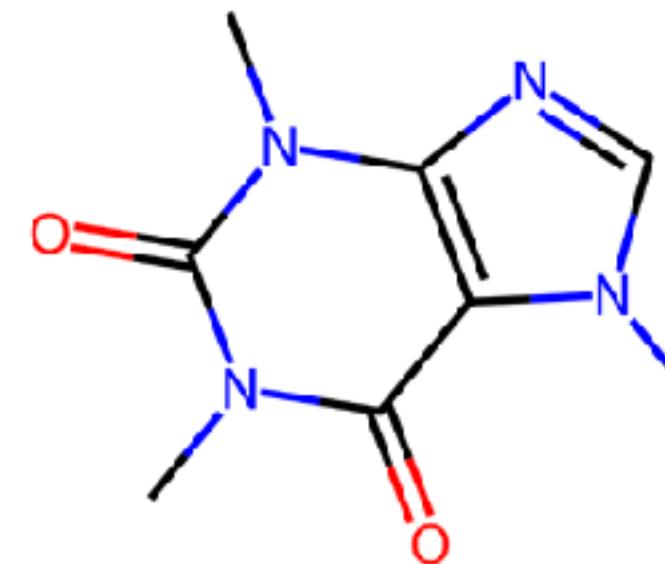
Molecule Representations

Text-based



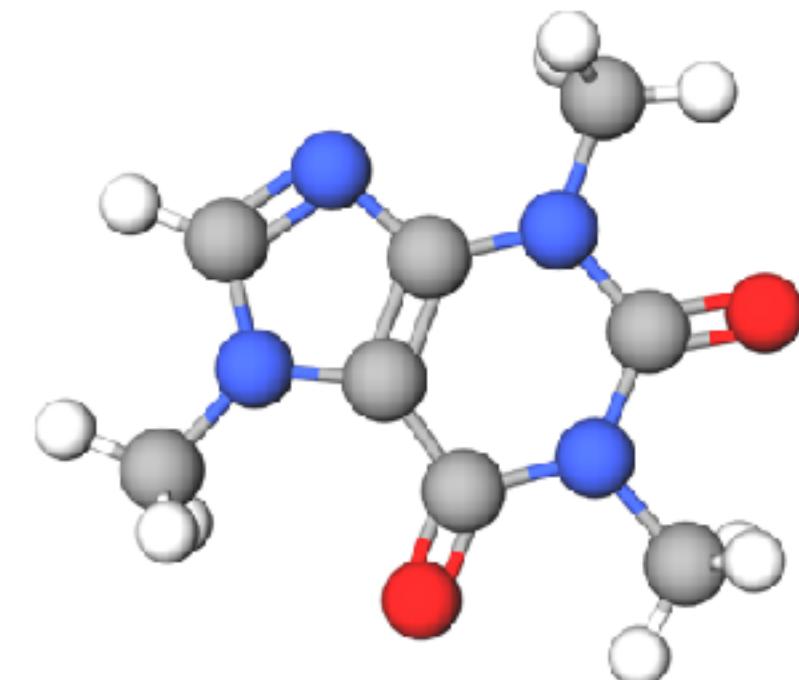
Smiles, Selfies, InChI, ...

Graph-based



Nodes = Atoms
Node feature = element
Edges = Covalent bonds
Edge feature = Bond type

Point Cloud



- Nodes in the graph have additional **3D spatial annotations**.
- Unconnected points are represented without explicit edge (bond) information.

Molecule Representations

Text-based

CN1C=NC2=C1C(=O)N(C(=O)N2C)C

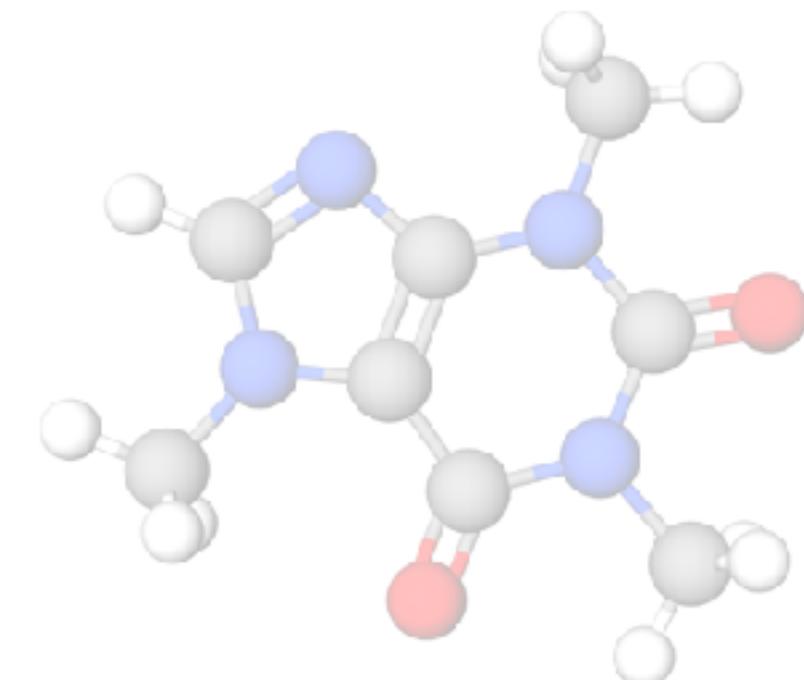
Smiles, Selfies, InChI, ...

Graph-based



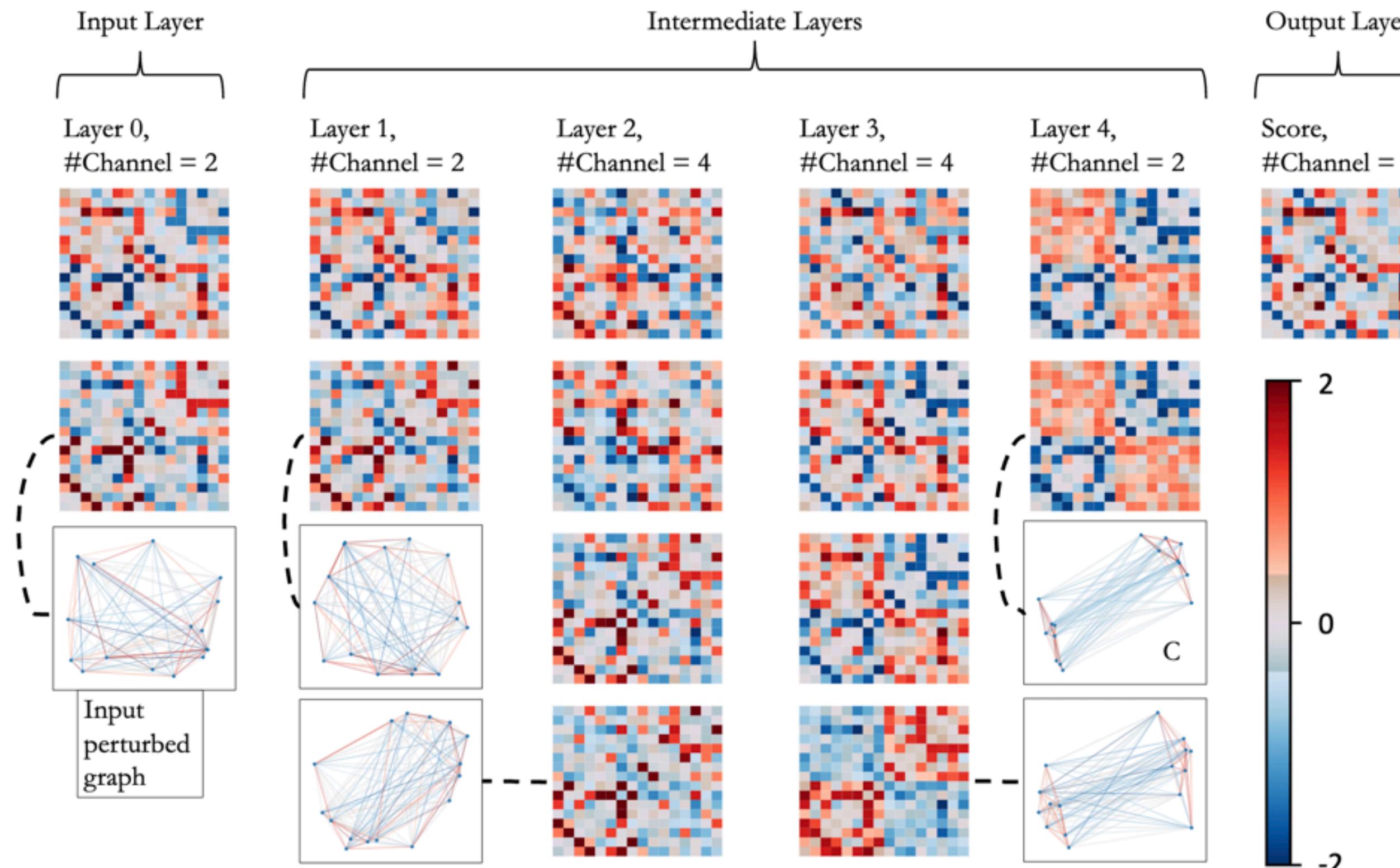
Nodes = Atoms
Node feature = element
Edges = Covalent bonds
Edge feature = Bond type

Point Cloud



- Nodes in the graph have additional **3D spatial annotations**.
- Unconnected points are represented without explicit edge (bond) information.

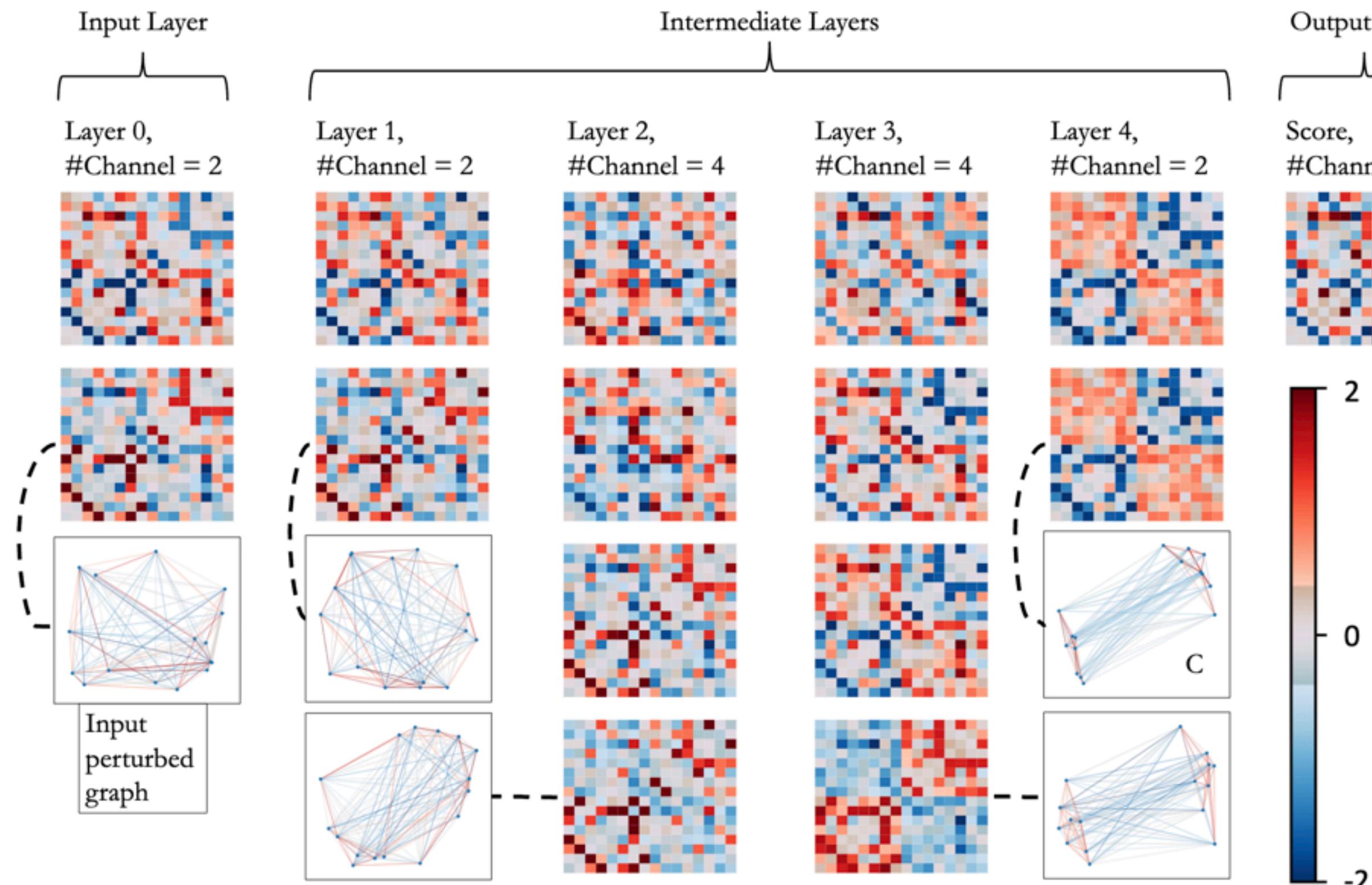
Score-Based Graph Generation



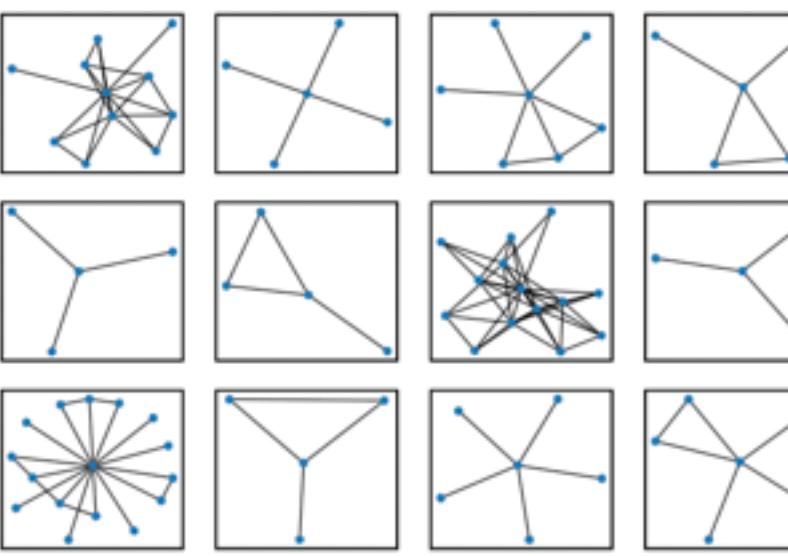
- First to do graph generation diffusion
- Permutation invariance handled via GNNs
- Relaxation of graph structure

Permutation Invariant Graph Generation via Score-Based Generative Modeling (Niu et al.)

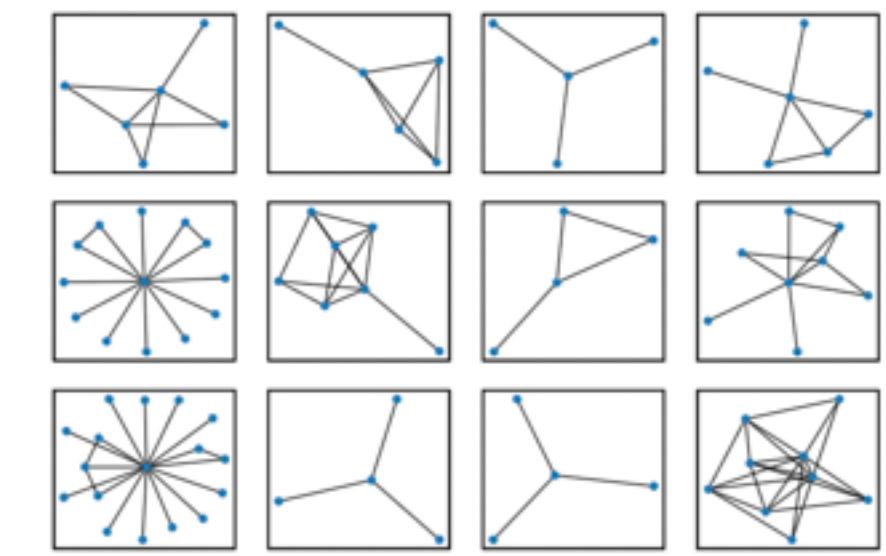
Score-Based Graph Generation



(d) Training data

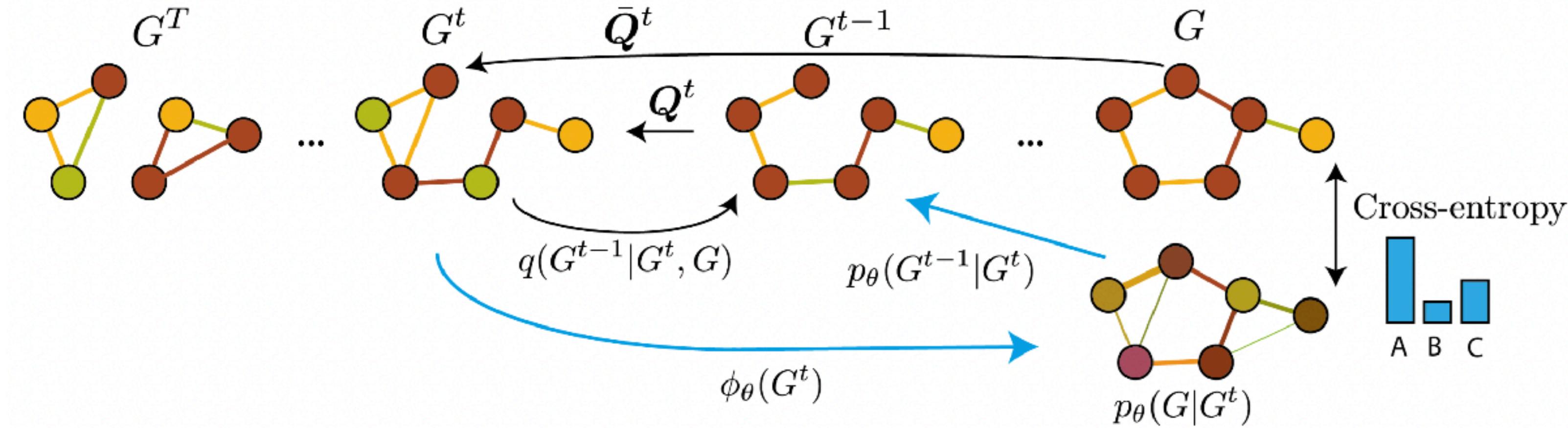


(e) EDP-GNN samples



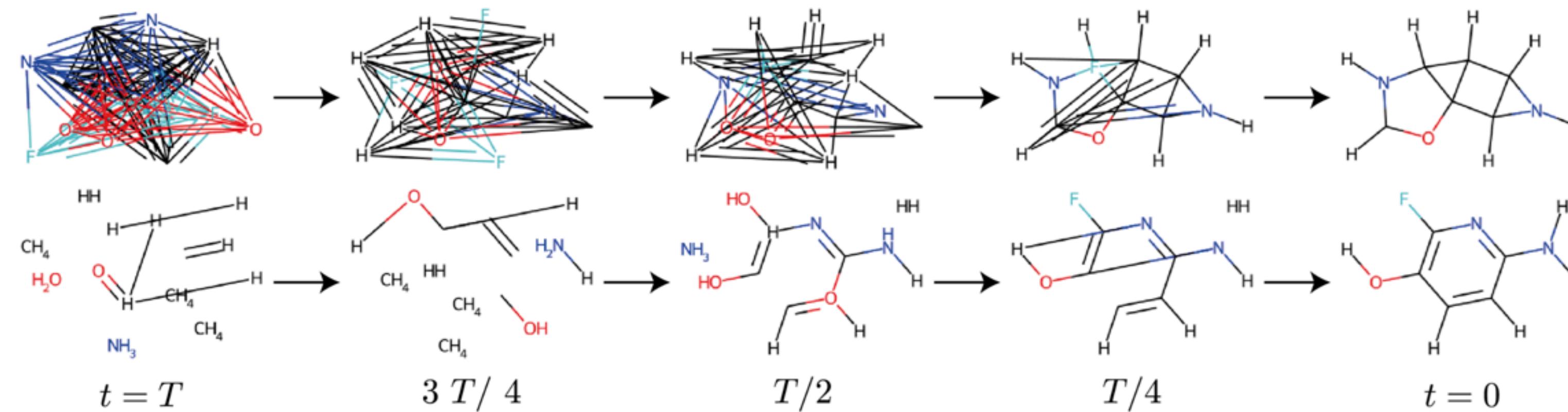
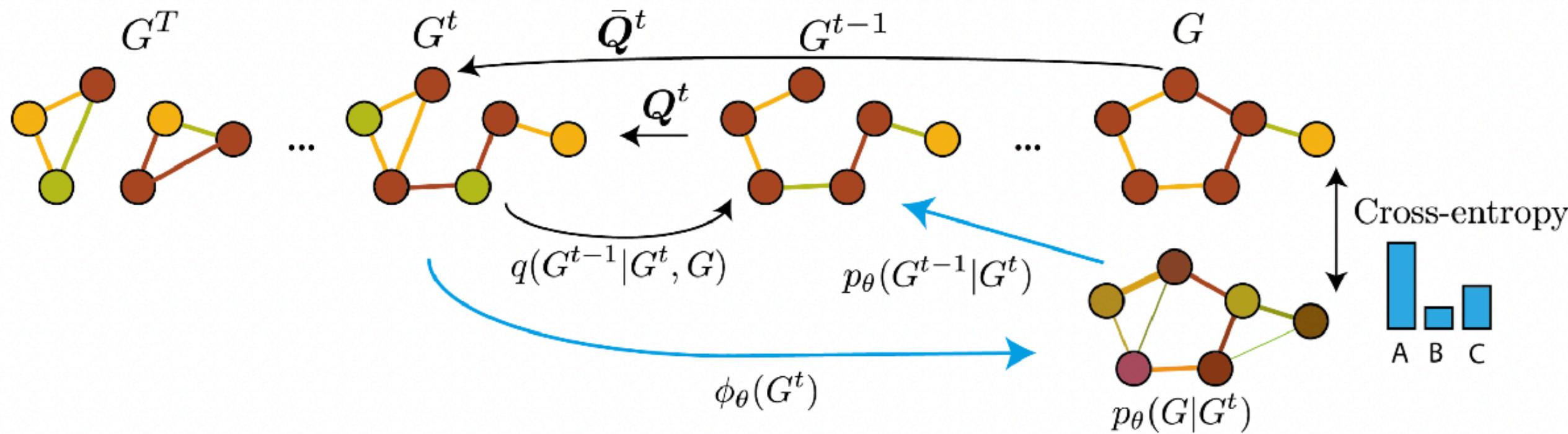
Permutation Invariant Graph Generation via Score-Based Generative Modeling (Niu et al.)

DiGress

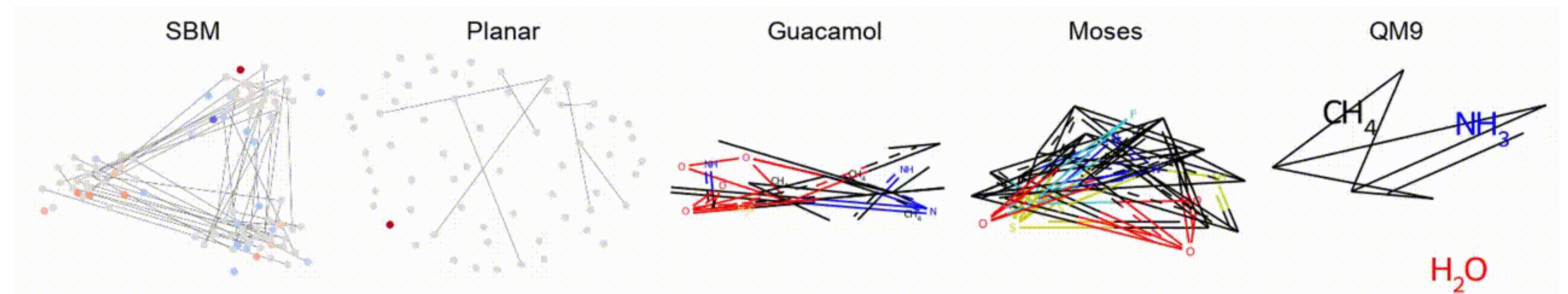


- Discrete diffusion by removing and adding edges
- Discrete jumps in categories (elements)

DiGress



DiGress



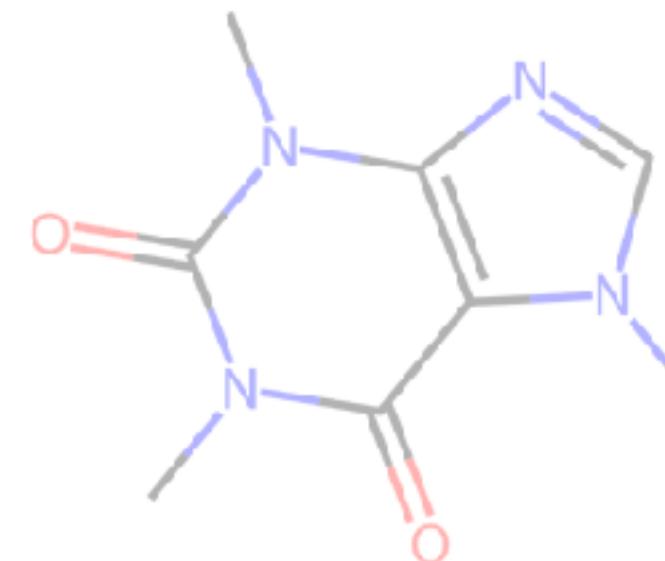
Molecule Representations

Text-based

CN1C=NC2=C1C(=O)N(C(=O)N2C)C

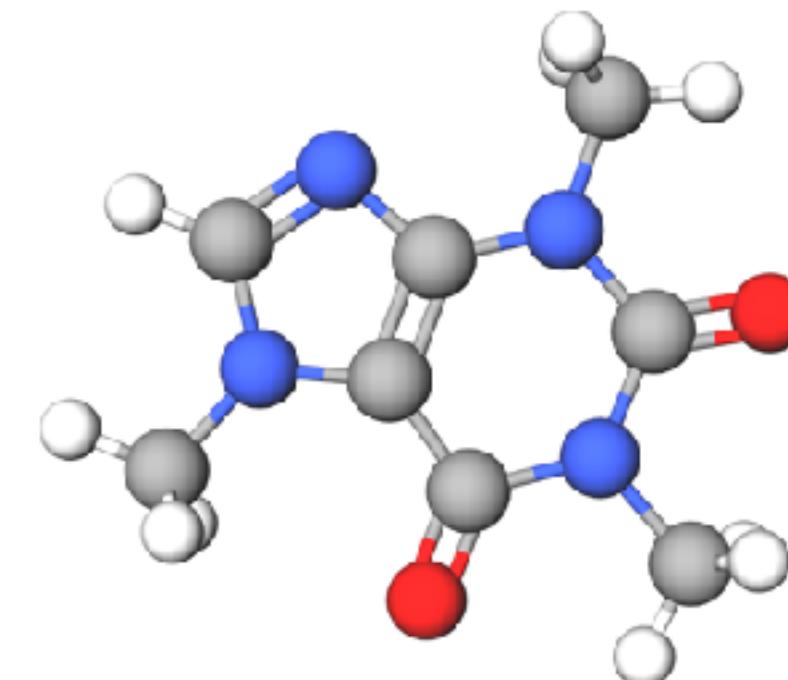
Smiles, Selfies, InChI, ...

Graph-based



Nodes = Atoms
Node feature = element
Edges = Covalent bonds
Edge feature = Bond type

Point Cloud



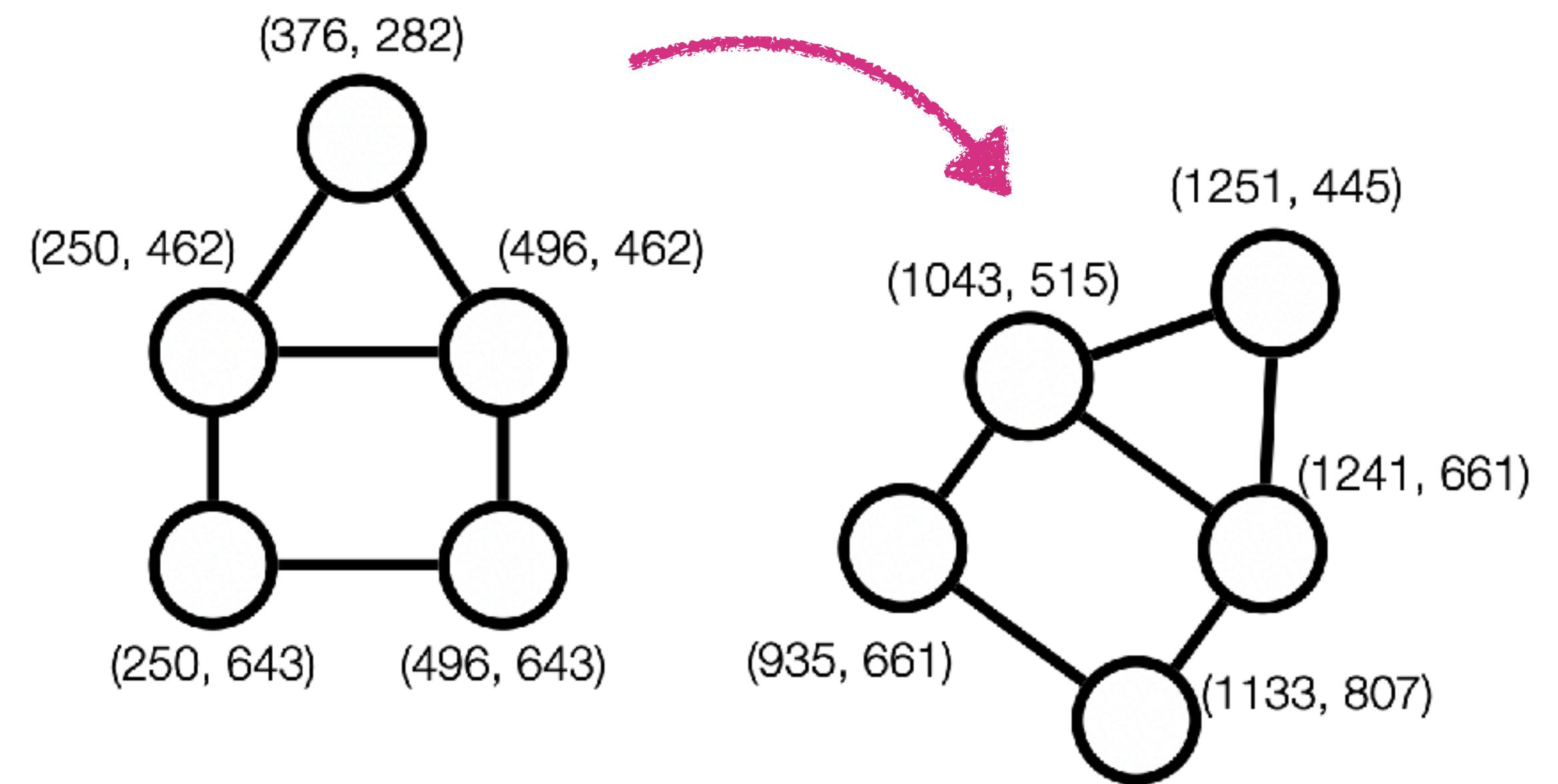
- Nodes in the graph have additional **3D spatial annotations**.
- Unconnected points are represented without explicit edge (bond) information.

Geometric GNNs: Key Concepts

- **Integration of Geometry:**
Add 3D coordinates to node feature.
- **Equivariance:** given **rotation** and **translation**
- **Enhanced Message Passing:**
Use geometric attributes like distances and angles.
- **Applications:**
Spatial reasoning, like molecular modeling, 3D object recognition, and physical simulations.

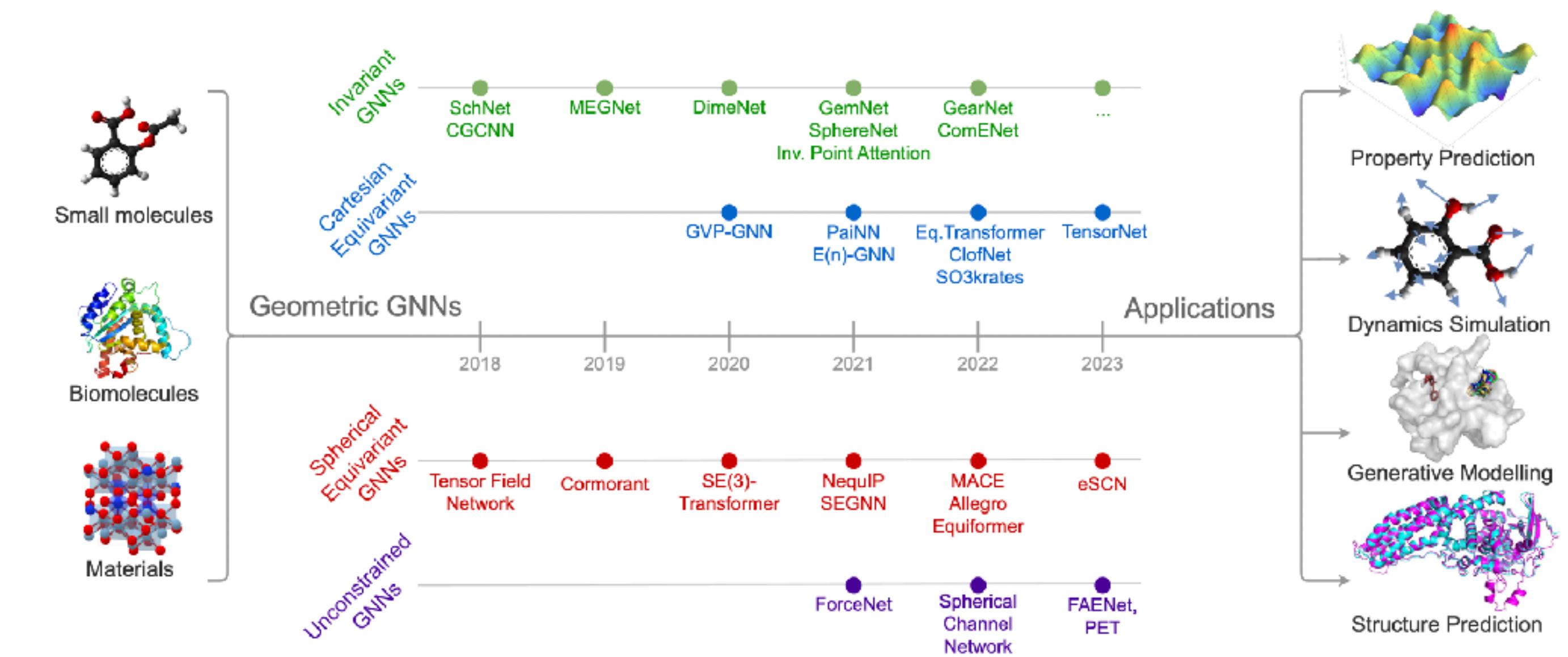
Geometric GNNs: Key Concepts

- **Integration of Geometry:**
Add 3D coordinates to node feature.
- **Equivariance:** given **rotation** and **translation**
- **Enhanced Message Passing:**
Use geometric attributes like distances and angles.
- **Applications:**
Spatial reasoning, like molecular modeling, 3D object recognition, and physical simulations.



Geometric GNNs: Key Concepts

- **Integration of Geometry:**
Add 3D (or 3D) coordinates to node feature.
- **Equivariance:** given **rotation** and **translation**
- **Enhanced Message Passing:**
Use geometric attributes like distances and angles.
- **Applications:**
Spatial reasoning, like molecular modeling, 3D object recognition, and physical simulations.



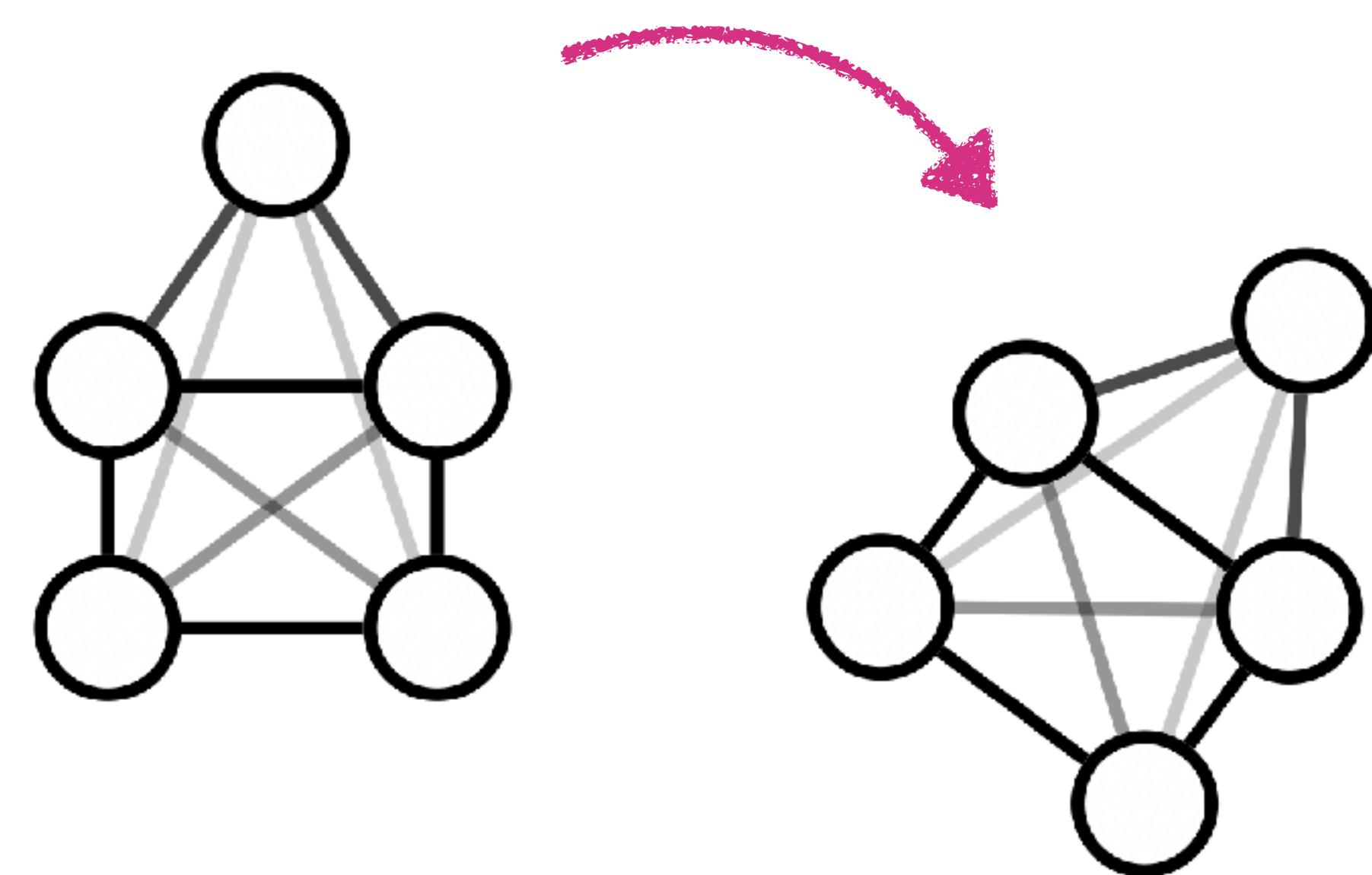
A Hitchhiker's Guide to Geometric GNNs for 3D Atomic Systems (Duval et al.)

Geometric GNNs: Key Concepts

- **Integration of Geometry:**
Add 3D coordinates to node feature.
- **Equivariance:** given **rotation** and **translation**
- **Enhanced Message Passing:**
Use geometric attributes like distances and angles.
- **Applications:**
Spatial reasoning, like molecular modeling, 3D object recognition, and physical simulations.

Geometric GNNs: Key Concepts

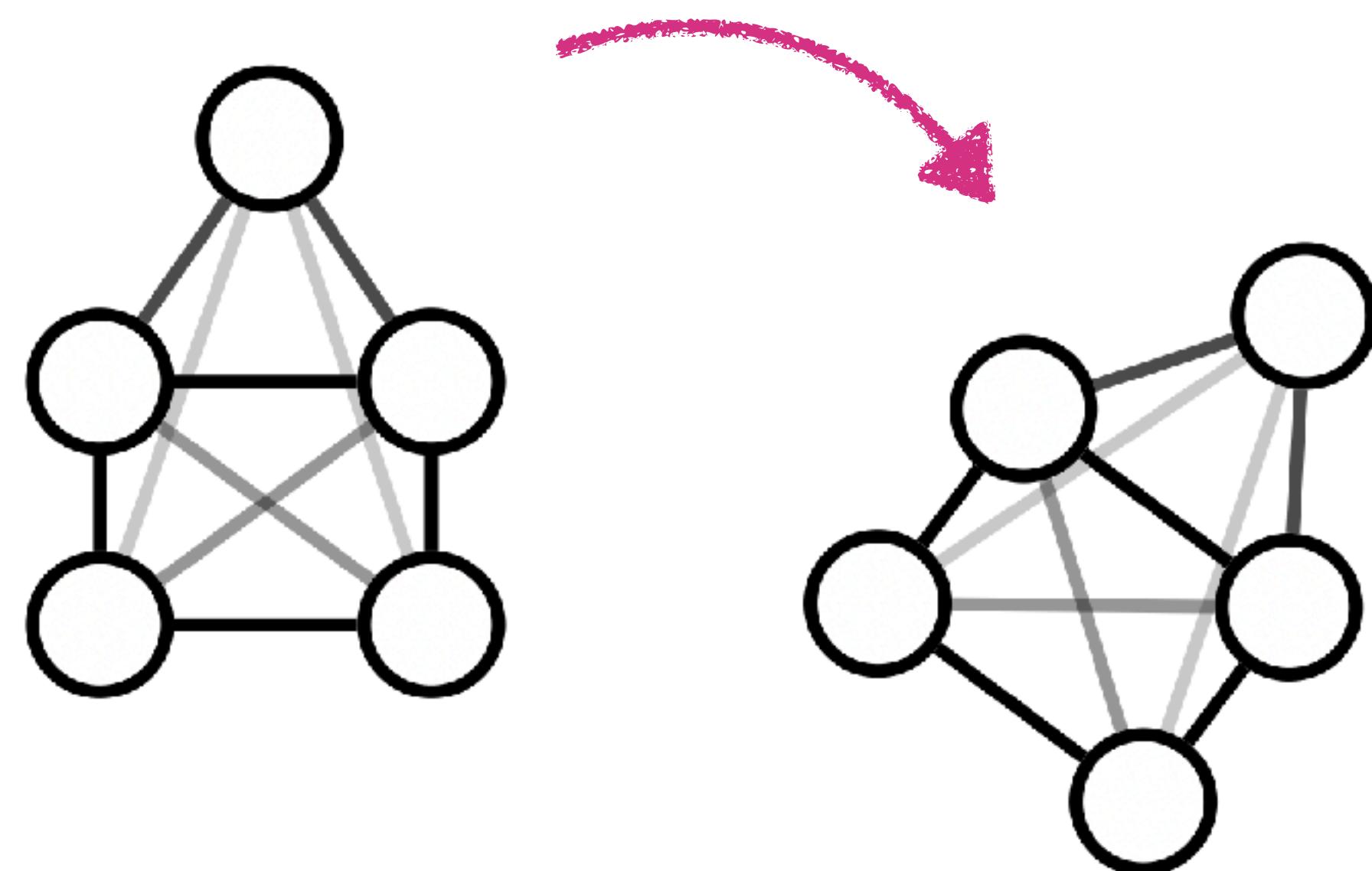
- **Integration of Geometry:**
Add 3D coordinates to node feature.
- **Equivariance:** given **rotation** and **translation**
- **Enhanced Message Passing:**
Use geometric attributes like distances and angles.
- **Applications:**
Spatial reasoning, like molecular modeling, 3D object recognition, and physical simulations.



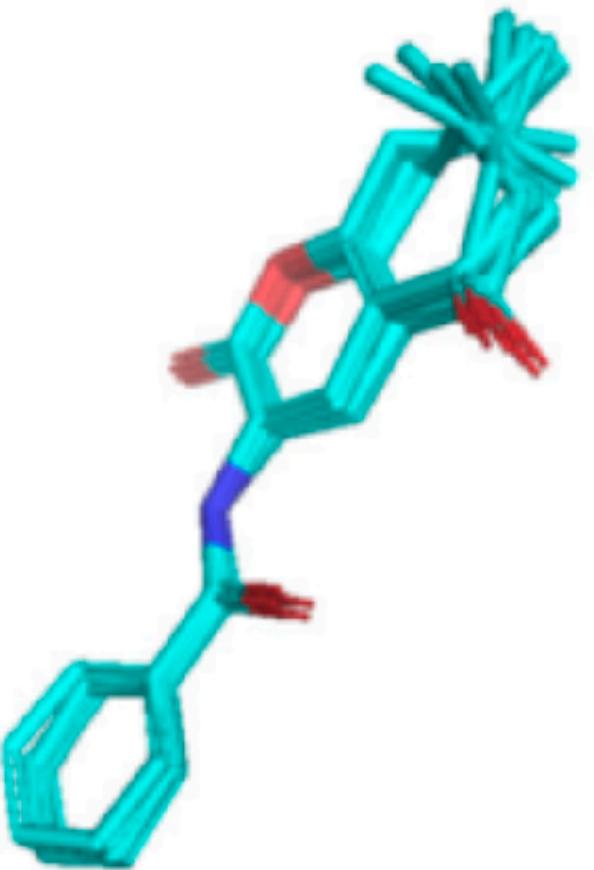
Geometric GNNs: Key Concepts

- **Integration of Geometry:**
Add 3D coordinates to node feature.
- **Equivariance:** given **rotation** and **translation**
- **Enhanced Message Passing:**
Use geometric attributes like distances and angles.
- **Applications:**
Spatial reasoning, like molecular modeling, 3D object recognition, and physical simulations.

Equivariant towards reflections!



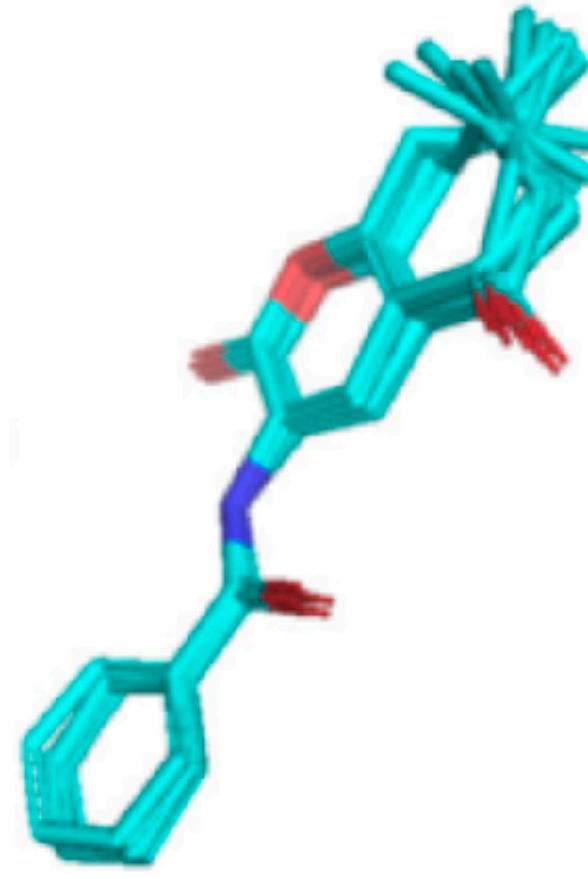
Conformer



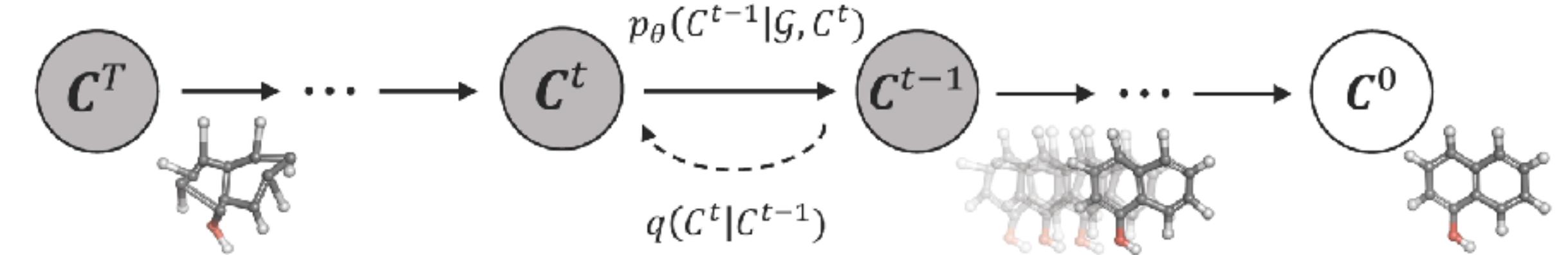
Better Informed Distance Geometry: Using What We
Know To Improve Conformation Generation (Riniker et al.)

- Molecular graph does not determine 3d geometry
- Molecules are not static
- Different configurations due to single bond rotations

Conformer



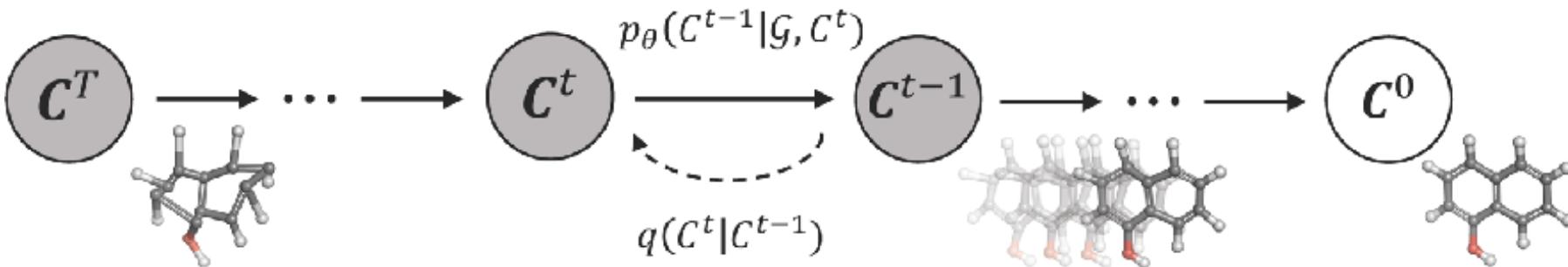
Better Informed Distance Geometry: Using What We Know To Improve Conformation Generation (Riniker et al.)



GeoDiff: a Geometric Diffusion Model for Molecular Conformation Generation (Xu et al.)

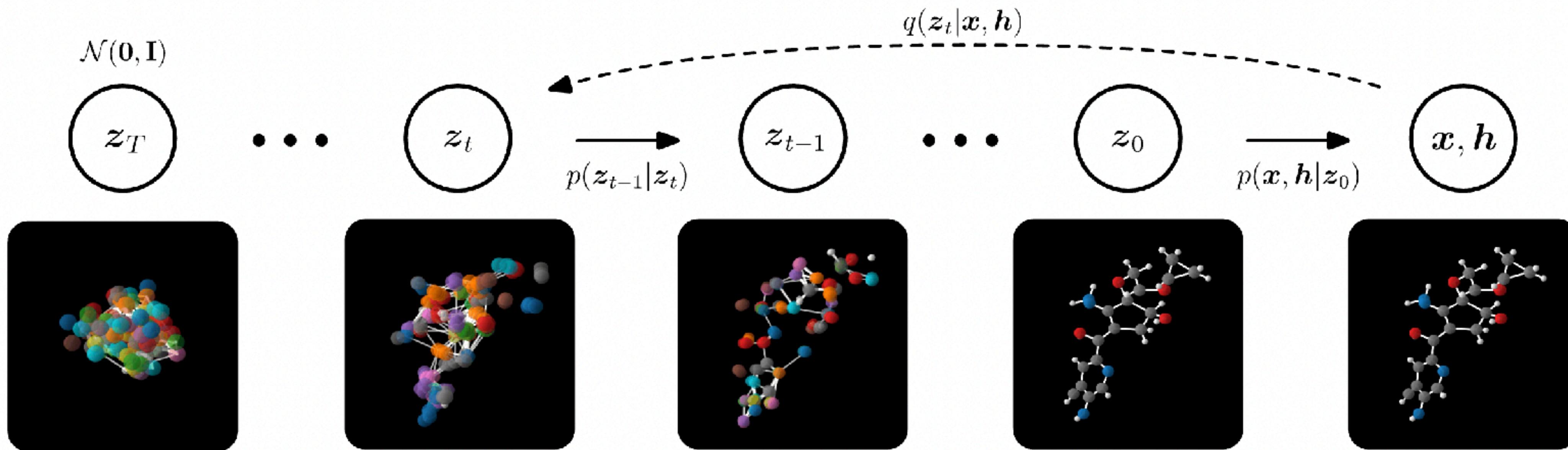
- Molecular graph does not determine 3d geometry
- Molecules are not static
- Different configurations due to single bond rotations

Conformer



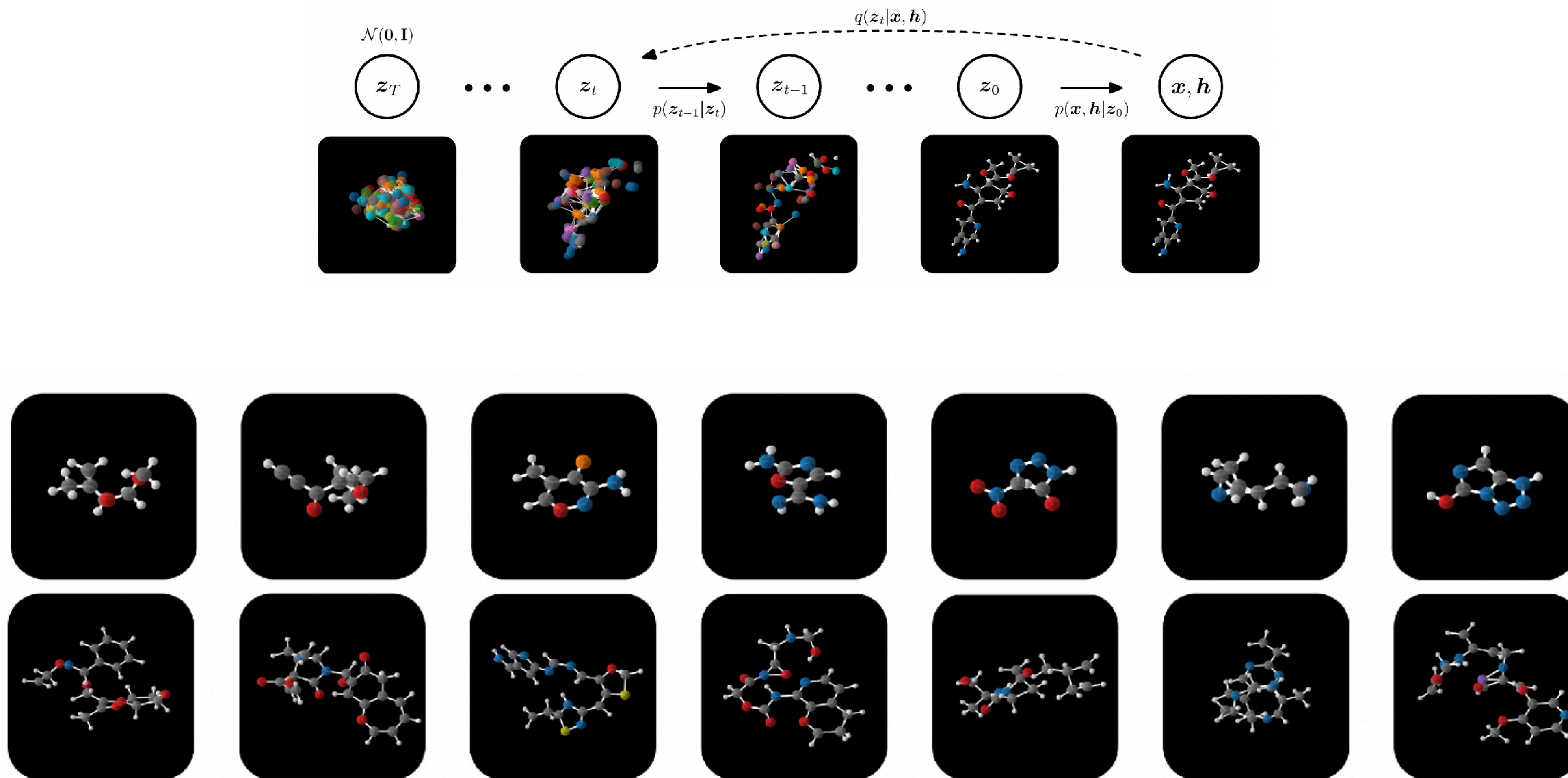
Graph				
Reference				
GeoDiff				
ConfGF				
GraphDG				

Molecule Generation in 3D

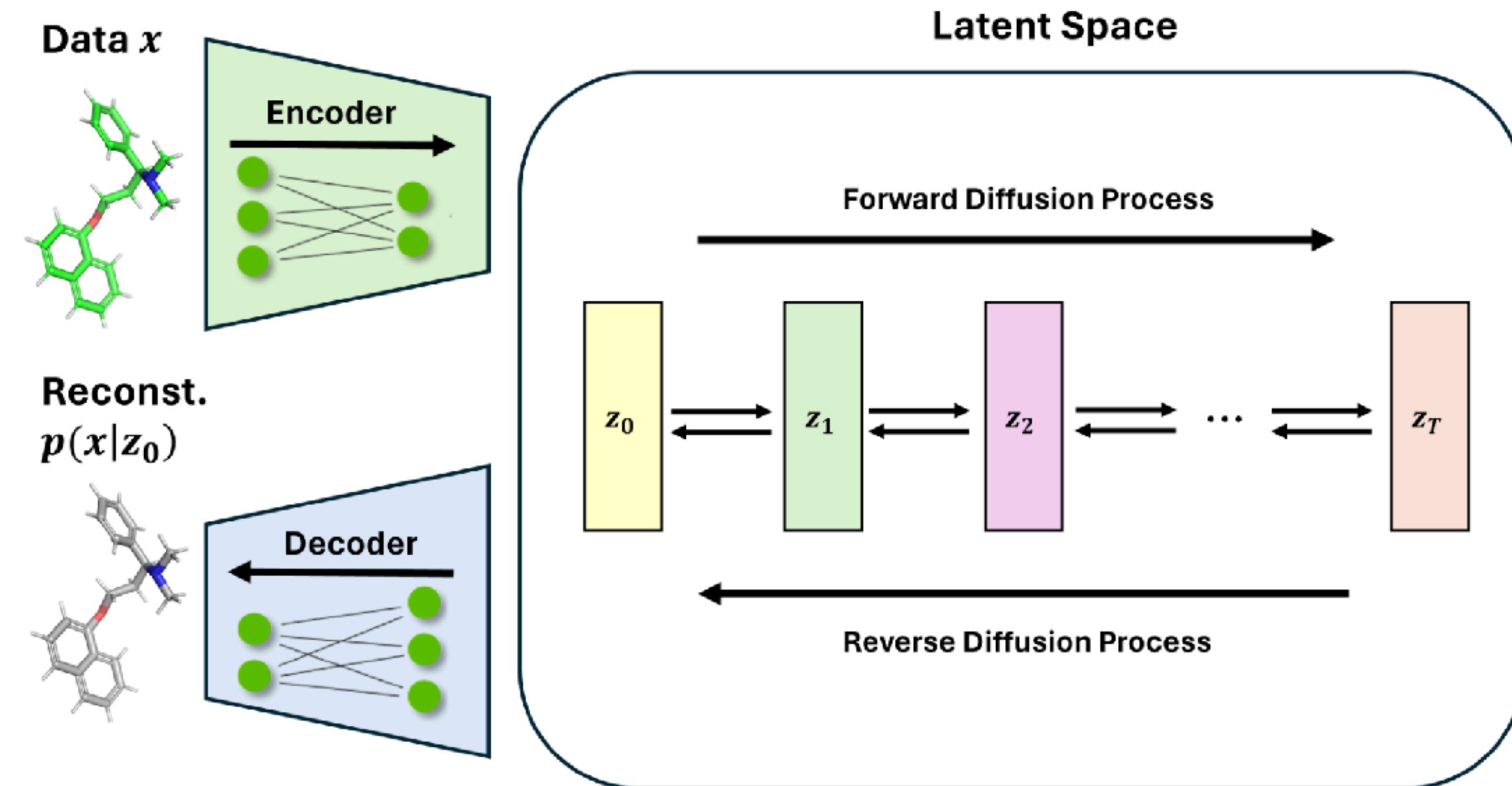


- First ones to achieve SOTA results for 3D molecule generation using diffusion

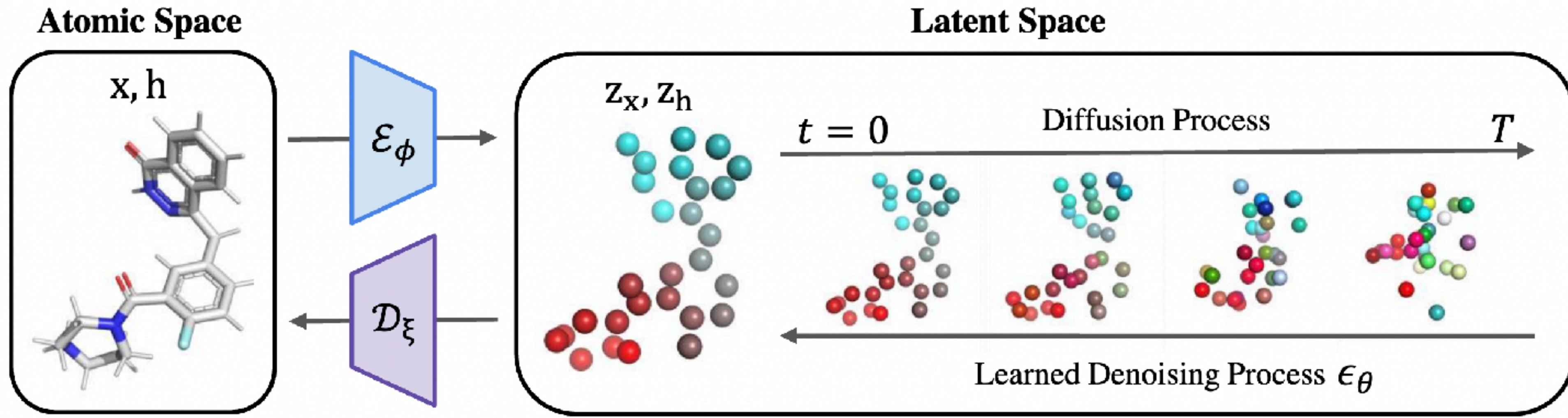
Molecule Generation in 3D



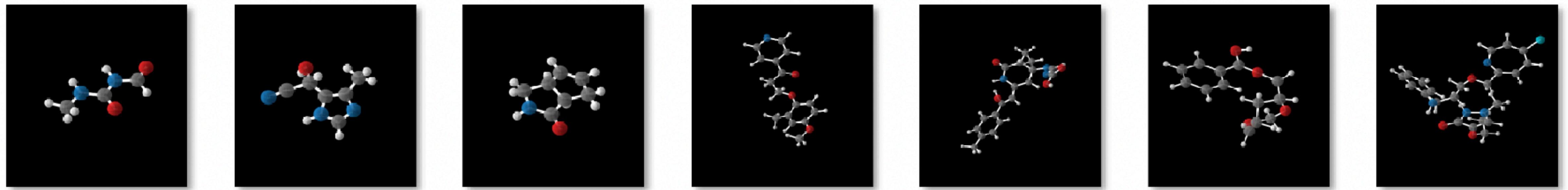
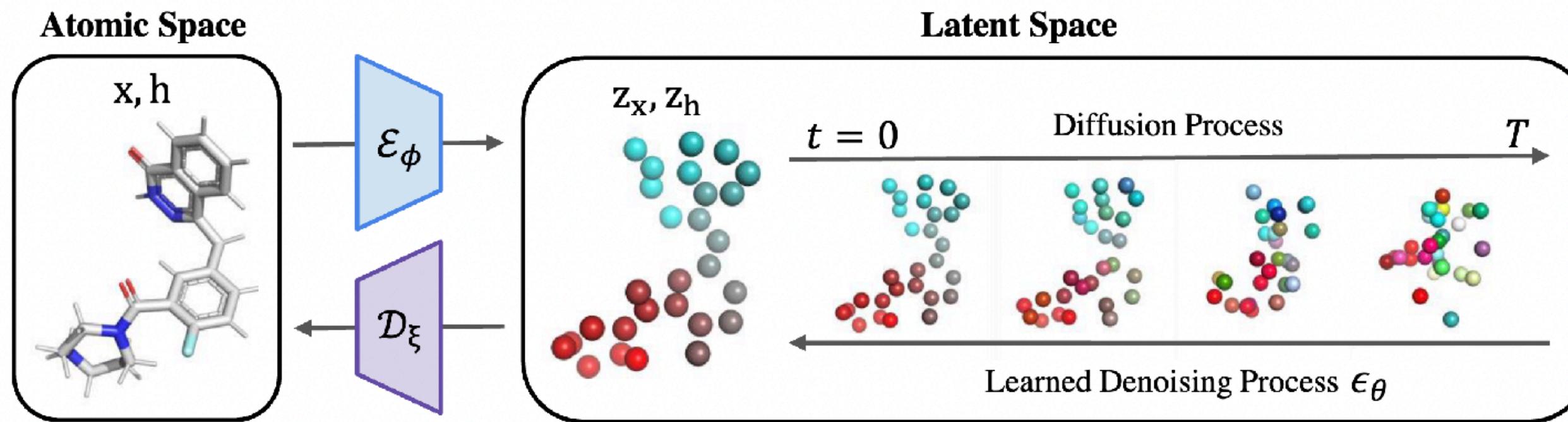
Latent Diffusion



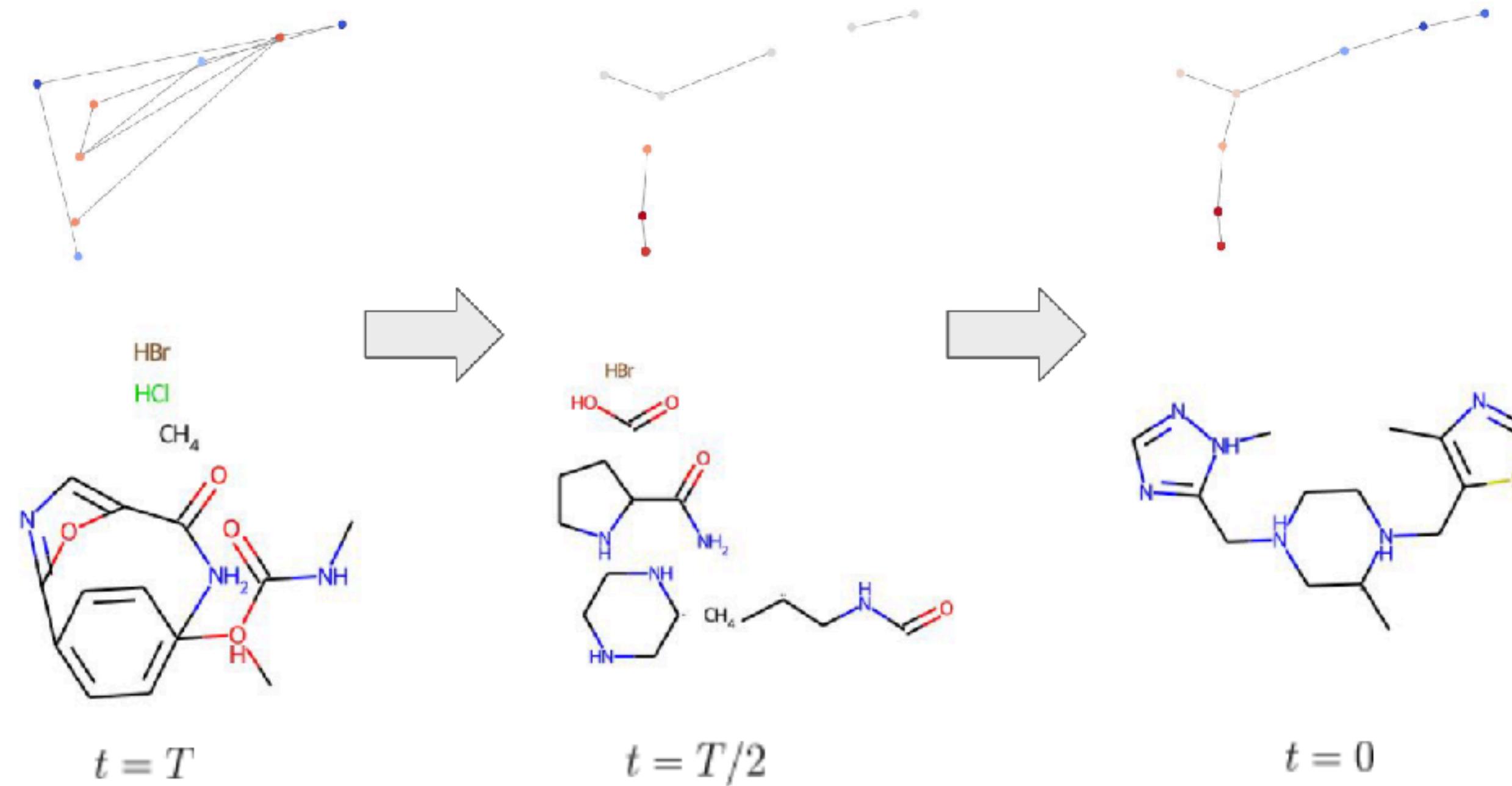
Latent Diffusion



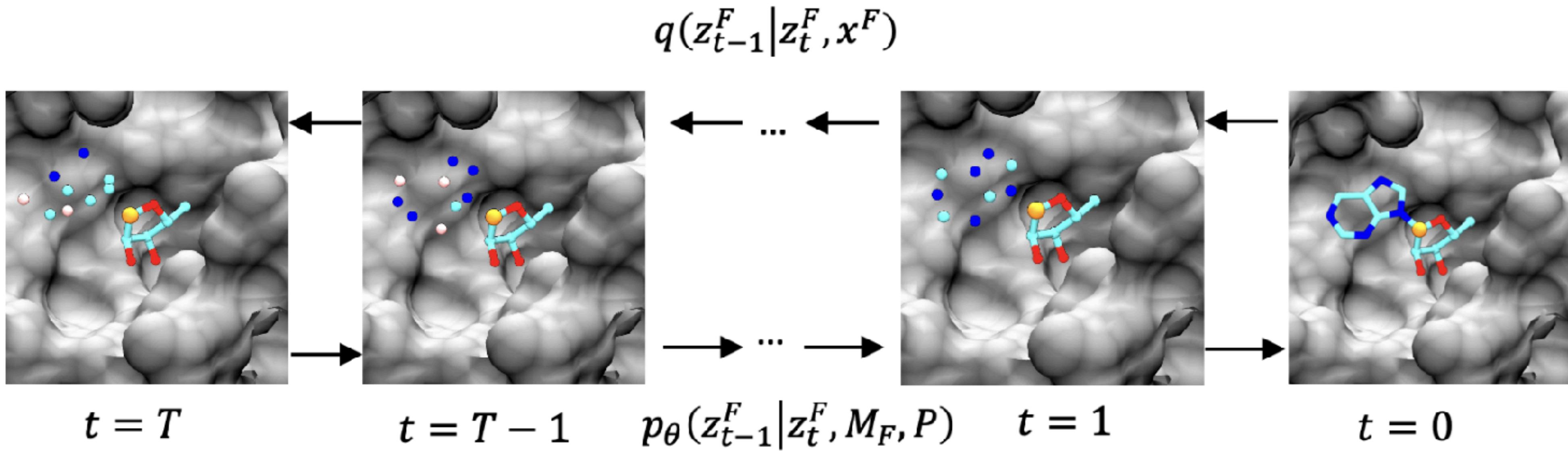
Latent Diffusion



Fragments



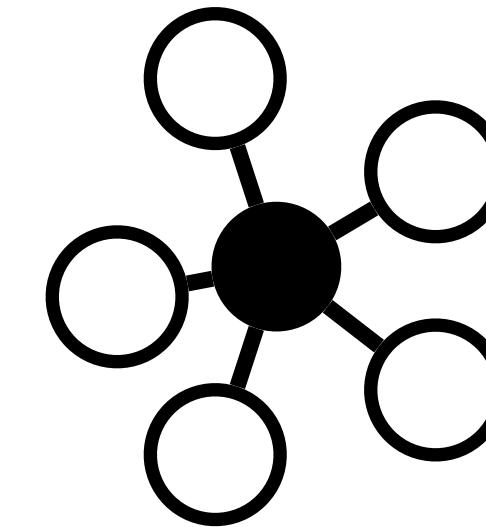
Pocket Conditioning



- Conditioned generation on protein pockets
- Extends user-provided scaffold
- Protein pocket is also processed as point cloud

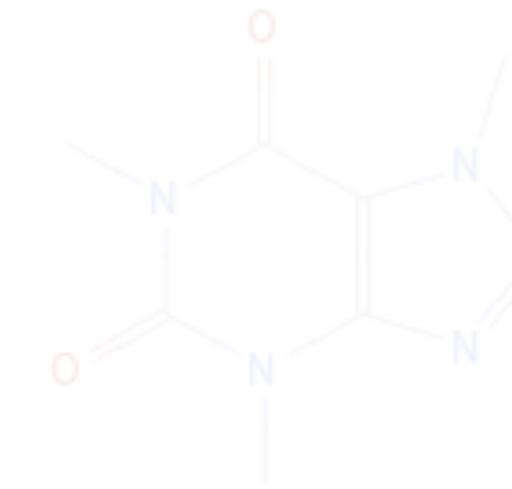
Evaluation

- **Validity:** Percentage of molecules with correct **valency**.
- **Novelty:** Percentage of molecules **not in training dataset**
- **Uniqueness:** Percentage of generated unique molecule

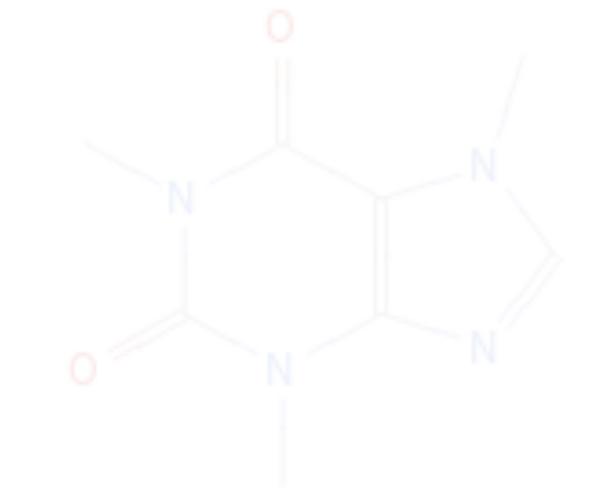


(Carbon atom with 5 hydrogen bonds)

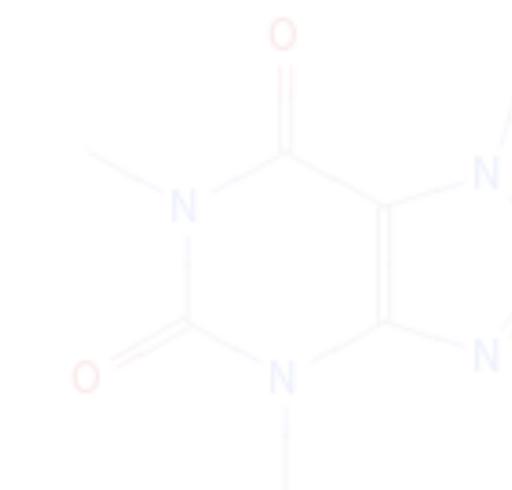
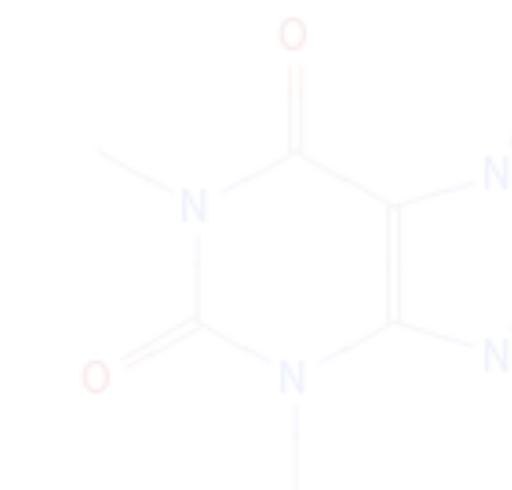
Train set:



Generated Molecule:

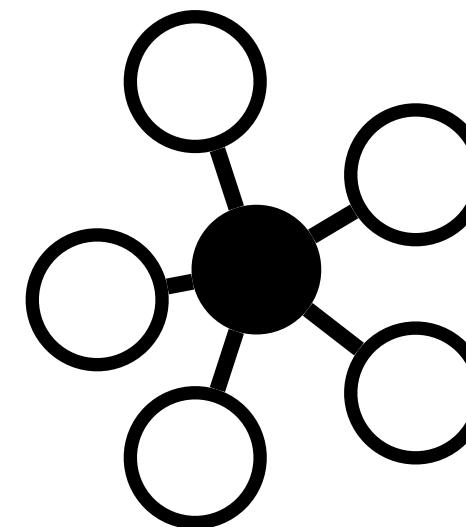


Generated Molecules:



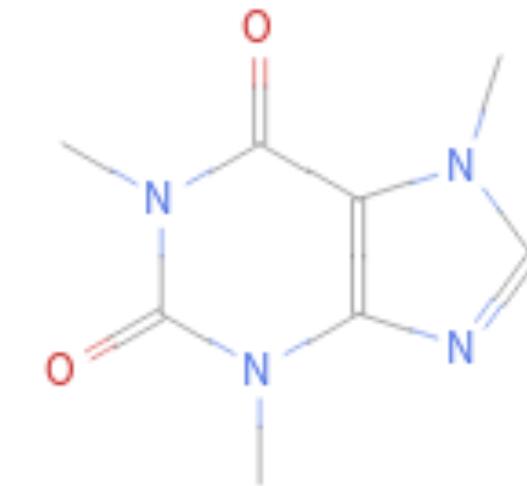
Evaluation

- **Validity:** Percentage of molecules with correct **valency**.
- **Novelty:** Percentage of molecules **not in training dataset**
- **Uniqueness:** Percentage of generated unique molecule

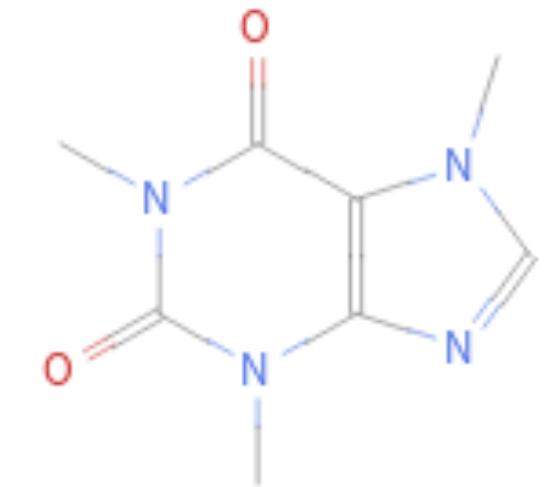


(Carbon atom with 5 hydrogen bonds)

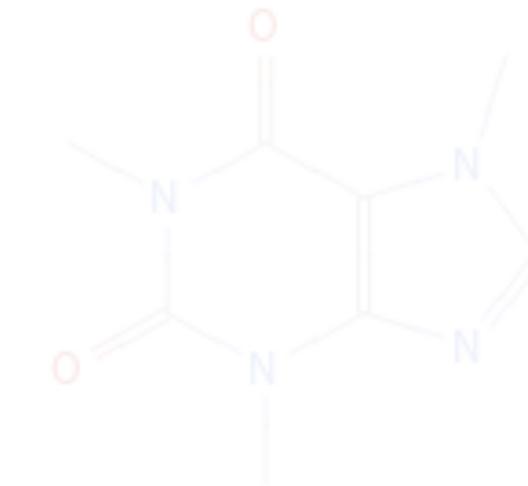
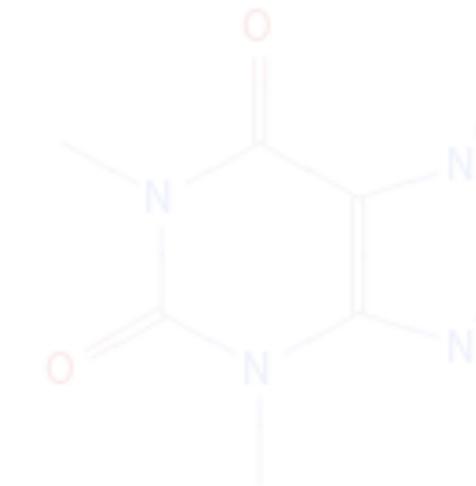
Train set:



Generated Molecule:

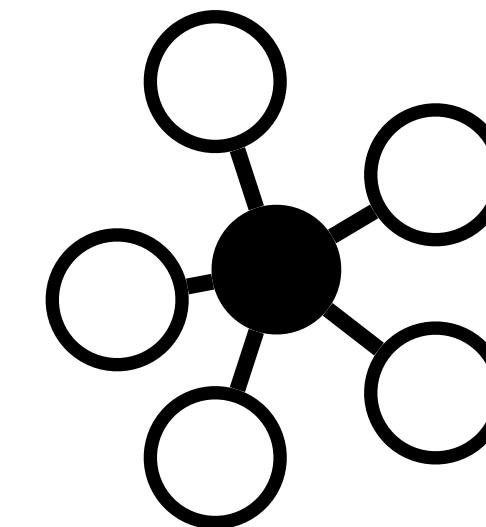


Generated Molecules:



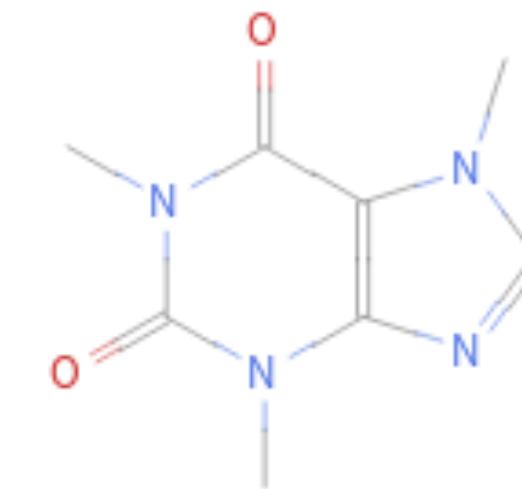
Evaluation

- **Validity:** Percentage of molecules with correct **valency**.
- **Novelty:** Percentage of molecules **not in training dataset**
- **Uniqueness:** Percentage of generated unique molecule

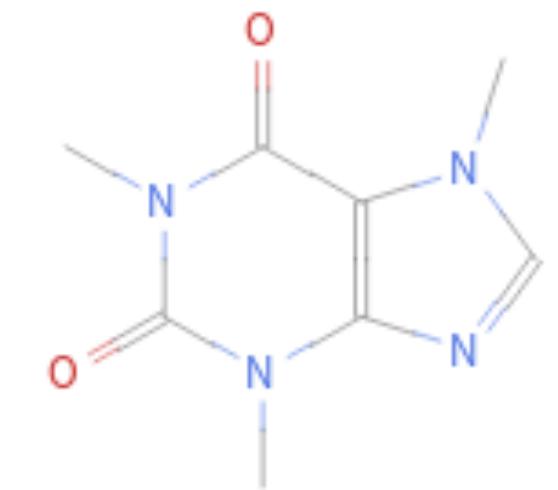


(Carbon atom with 5 hydrogen bonds)

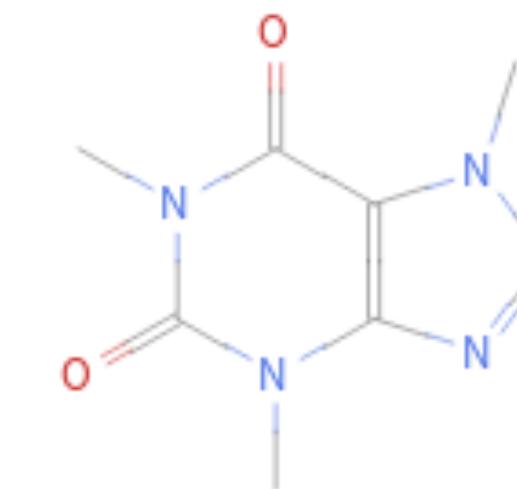
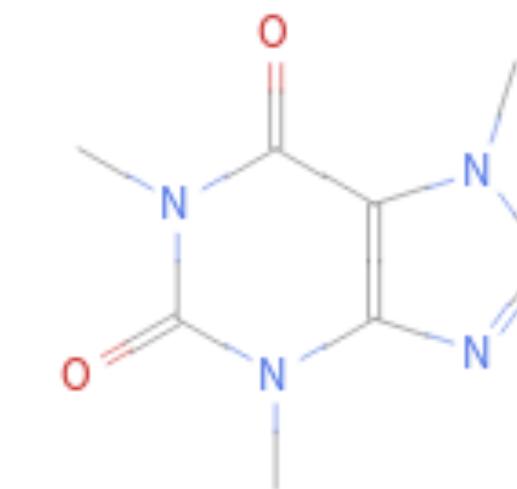
Train set:



Generated Molecule:



Generated Molecules:



Evaluation

- **Validity:** Percentage of molecules with correct **valency**.
- **Novelty:** Percentage of molecules **not in training dataset**
- **Uniqueness:** Percentage of generated unique molecule

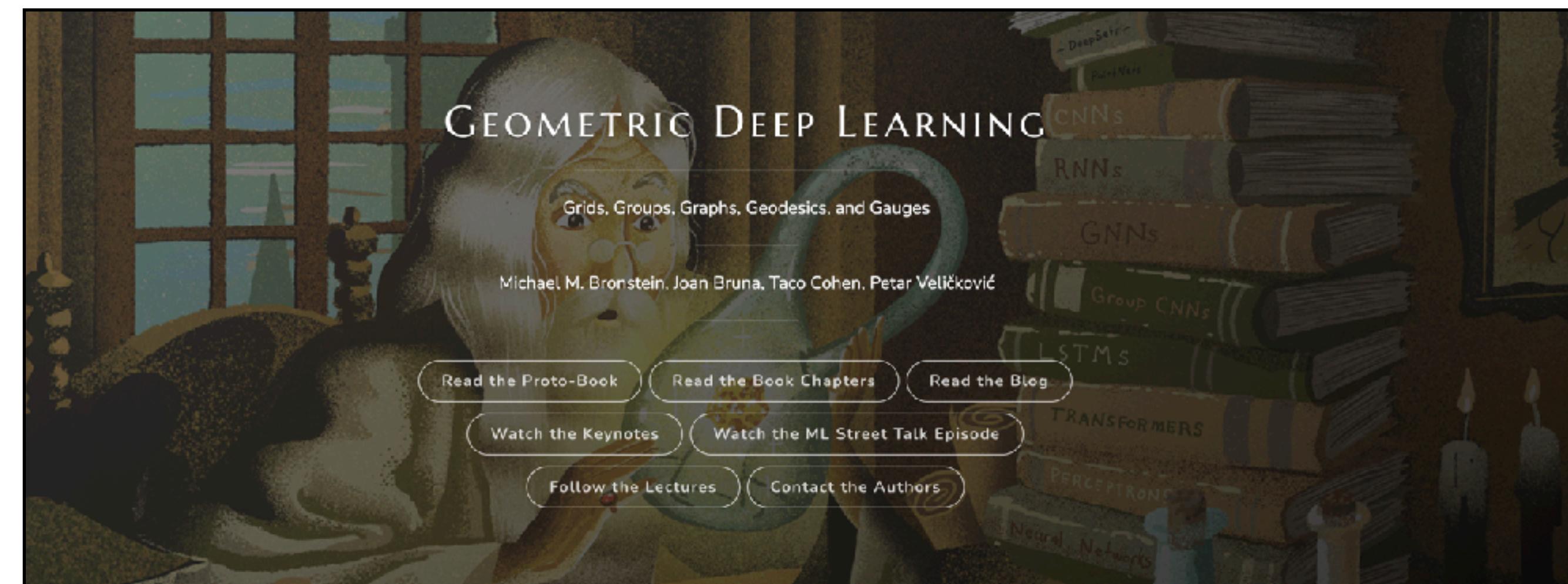
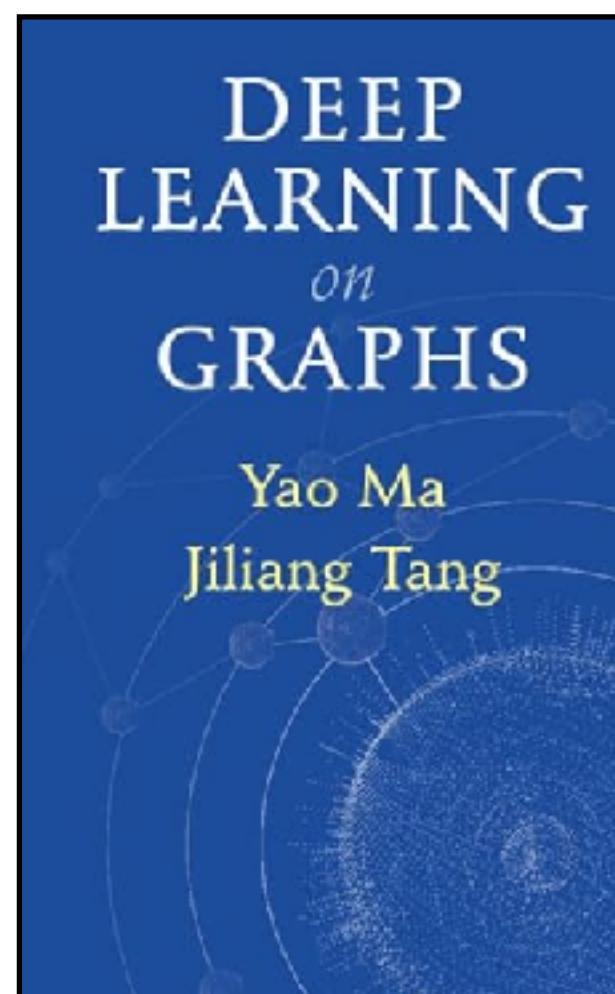
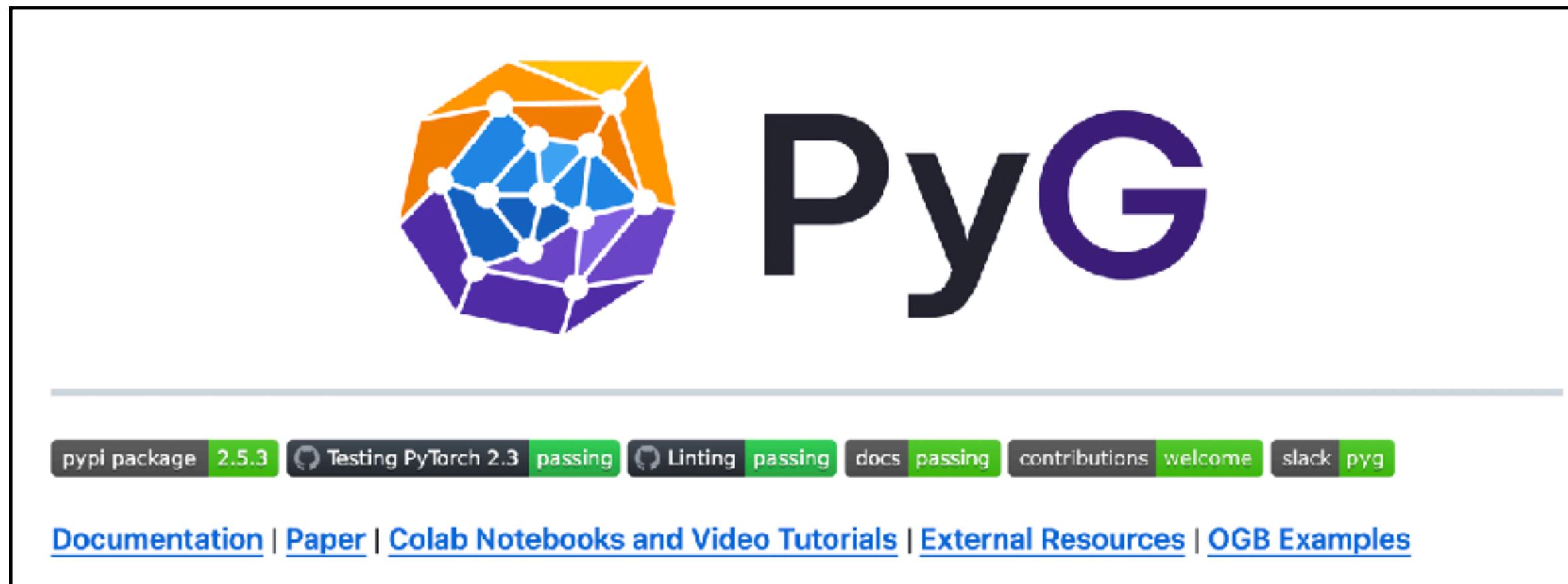
- **Lipophilicity Assessment:** Measures absorption
- **Quantitative Estimate of Druglikeness:** Based on many molecular properties
- **Synthetic Accessibility Score:** Fragment scores + complexity penalty
- **Lipinski Rule of Five:** Drug-likeness based on molecular structure



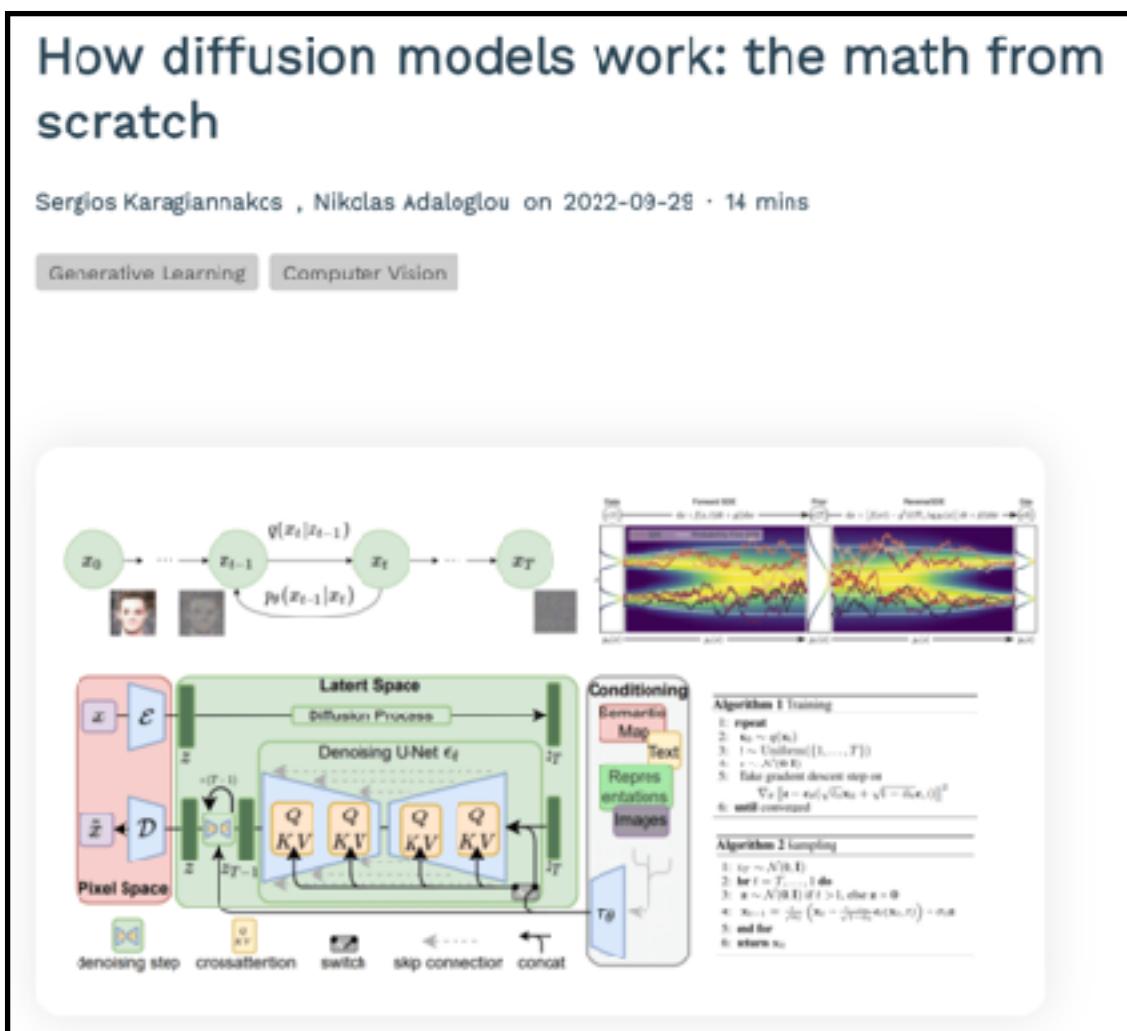
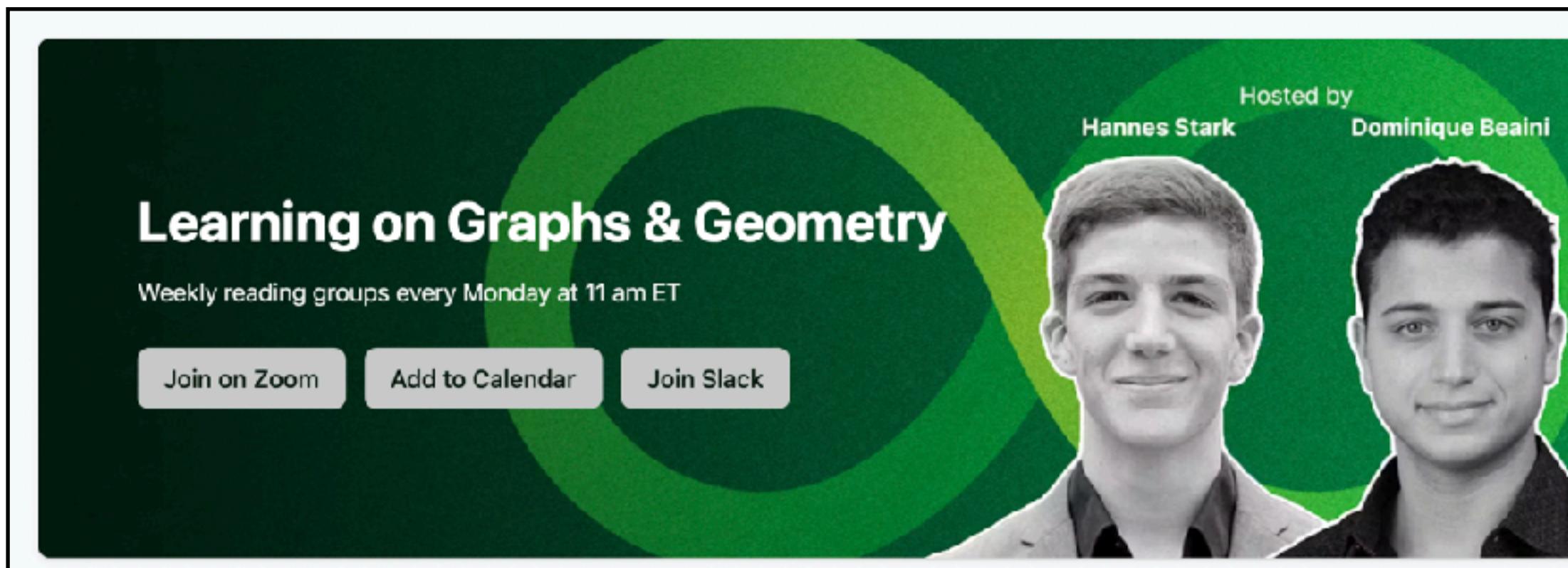
Part V

GETTING STARTED

Graph Neural Networks



Diffusion Models



Hugging Face

← Back to Articles

The Annotated Diffusion Model

Published June 7, 2022

Update on GitHub

▲ Upvote 33 +27

nielsr Niels Rogge kashif Kashif Rasul

Open in Colab

Awesome-Diffusion-Models Public Watch 2011

diffusion Add files via upload 01b3394 · 7 months ago 932 commits

docs rename title last year

template add website interface last year

website add website interface last year

README.md Add files via upload 7 months ago

license Create license 2 years ago

README MIT license

View raw License PDF Place with raw

This repository contains a collection of resources and papers on Diffusion Models.

Please refer to [this page](#) as this page may not contain all the information due to page constraints.

Contents

- Resources
 - Introductory Posts
 - Introductory Papers
 - Introductory Videos
 - Introductory Lectures
 - Tutorial and Jupyter Notebooks

Computational Drug Discovery

TeachOpenCADD

A teaching platform for computer-aided drug design (CADD) using open source packages and data.

Project TeachOpenCADD DOI

launch binder License CC BY 4.0 tag v2023.05.2 CI failing Docs failing downloads 9K total

closed pull requests 233 open pull requests 11 closed issues 111 issues 42 open

If you use TeachOpenCADD in a publication, please [cite](#) us! If you use TeachOpenCADD in class, please include a link back to our repository.

In any case, please [star](#) (and tell your students to star) those repositories you consider useful for your learning/teaching activities.

Description

The diagram illustrates the TeachOpenCADD workflow for drug discovery, organized into several main sections:

- Initial Data Sources:** T001 Query CHEMBL, T008 Query PDB.
- Filtering and Ensemble:** T002 Molecular filtering: ADME criteria, T003 Molecular filtering: Unwanted substr., T004 Ligand-based screening: Compound similarity, T005 Ligand clustering, T006 Maximum common substructures, T007 Ligand-based screening: Machine learning.
- Comparison and Web Services:** T009 Ligand-based ensemble pharmacophores, T010 Binding site comparison, T011 Query online API webservices, T012 Query KLIFS.
- Optimization and Detection:** T013 Query PubChem, T014 Binding site detection, T015 Protein-ligand docking, T016 Protein-ligand interactions.
- Simulations and Analysis:** T017 MC3 View, T018 Molecular dynamics simulations, T019 Molecular dynamics analysis, T020 Molecular dynamics analysis.
- Advanced Methods:** T021 One-hot encoding, T022 Ligand-based screening: Neural Networks, T023 Molecular representations, T024 Recurrent neural networks, T025 Graph neural networks, T026 E(3)-equivariant CNNs.
- Kinase Focus:** T027 Kinase similarity: Ligand profile, T028 Kinase similarity: Compare perspectives.

Legend:

- Database queries (blue)
- Cheminformatics (green)
- Structural bioinformatics (orange)
- Deep learning (purple)
- Kinase similarity (yellow)

Datasets

Dataset	No. Molecules	Data	Task
QM9 [48]	133,885	Organic molecules with up to nine atoms and their DFT calculated quantum chemical properties.	Unconditioned and conditioned 3D molecular generation.
GEOM-DRUG [49]	over 450,000	37 million molecular conformations for over 450,000 molecules.	Unconditioned and conditioned 3D molecular generation.
ZINC250k [50]	249,455	A drug-like subset of the ZINC database with bioactivity data for a portion of the molecules.	Unconditioned and conditioned generation of drug-like molecules.
MOSES [51]	1,936,962	filtered from ZINC Clean Leads collection.	Unconditioned and conditioned generation of 2D molecules.
CrossDocked [52]	18,450 complexes, 22.6 million poses	A collection of ligand-protein complexes where ligands are docked against several protein targets similar to one another.	SBDD tasks: target-aware 3D generation, molecular docking, etc.
PDBbind [53]	23,496 complexes	Biomolecular complexes in PDB with experimentally measured binding affinities, including 19,443 protein-ligand, 2,852 protein-protein, 1,052 protein-nucleic acid, and 149 nucleic acid-ligand complexes.	protein-protein docking and SBDD tasks: target-aware 3D generation, molecular docking, etc.

Datasets

Dataset	No. Molecules	Data	Task
QM9 [48]	133,885	Organic molecules with up to nine atoms and their DFT calculated quantum chemical properties.	Unconditioned and conditioned 3D molecular generation.
GEOM-DRUG [49]	over 450,000	37 million molecular conformations for over 450,000 molecules.	Unconditioned and conditioned 3D molecular generation.
ZINC250k [50]	249,455	A drug-like subset of the ZINC database with bioactivity data for a portion of the molecules.	Unconditioned and conditioned generation of drug-like molecules.
MOSES [51]	1,936,962	filtered from ZINC Clean Leads collection.	Unconditioned and conditioned generation of 2D molecules.
CrossDocked [52]	18,450 complexes, 22.6 million poses	A collection of ligand-protein complexes where ligands are docked against several protein targets similar to one another.	SBDD tasks: target-aware 3D generation, molecular docking, etc.
PDBbind [53]	23,496 complexes	Biomolecular complexes in PDB with experimentally measured binding affinities, including 19,443 protein-ligand, 2,852 protein-protein, 1,052 protein-nucleic acid, and 149 nucleic acid-ligand complexes.	protein-protein docking and SBDD tasks: target-aware 3D generation, molecular docking, etc.

Literature

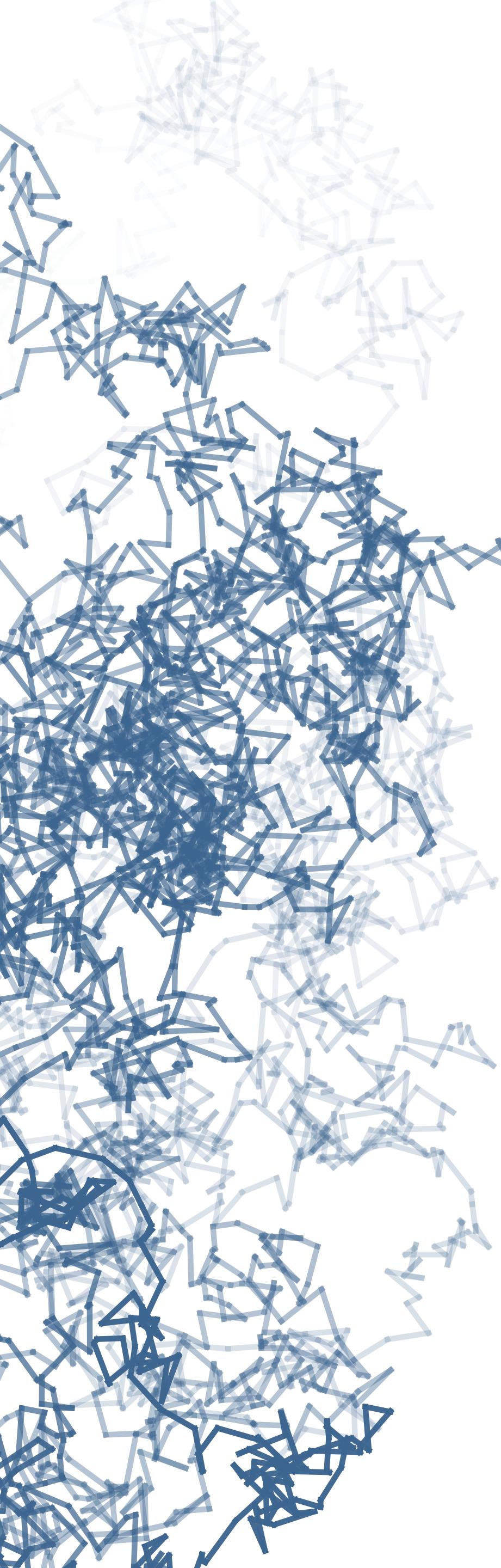
The screenshot shows a GitHub repository's README page. The title is "Molecule Generation with Graph Neural Networks and Probabilistic Diffusion". Below the title, there is a brief description: "Talk/Tutorial on 'Molecule Generation with Graph Neural Networks and Probabilistic Diffusion' as part of the ScaDS.AI Summer School 2024." A bulleted list of topics follows:

- Reviews
- Foundations of Diffusion Models
- Foundations of GNNs
- Diffusion Modes for Molecular Graphs
- Diffusion Modes for Molecular Point Clouds
- Diffusion in the Latent Space
- Broader Applications (Proteins, Docking, Force Fields, Antibodies, Retrosynthesis)

Below this, a section titled "Reviews" lists five items:

1. 2024 - Diffusion Models in *De Novo* Drug Design
Amira Alakhdar, Barnabas Poczos, Newell Washburn
2. 2024 - Diffusion models in protein structure and docking
Jason Yim, Hannes Stärk, Gabriele Corso, Bowen Jing, Regina Barzilay, Tommi S. Jaakkola
3. 2024 - Machine learning-aided generative molecular design
Yuanqi Du, Arjan R. Jamasb, Jeff Guo, Tianfan Fu, Charles Harris, Yingheng Wang, Chenru Duan, Pietro Liò, Philippe Schwaller, Tom L. Blundell
4. 2023 - A Survey on Graph Diffusion Models: Generative AI in Science for Molecule, Protein and Material
Mengchun Zhang, Maryam Qamar, Taegoo Kang, Yuna Jung, Chenshuang Zhang, Sung-Ho Bae, Chaoning Zhang
5. 2023 - Diffusion Models: A Comprehensive Survey of Methods and Applications
Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, Ming-Hsuan Yang

github.com/gerritgr/diffusion_gnn_tutorial

The background features a complex, abstract geometric pattern composed of numerous thin, light blue and white lines forming intricate shapes like triangles and hexagons, creating a sense of depth and motion.

Part VI

CONCLUDING REMAKES

Challenges & Opportunities

The proteins we know have emerged over 3.5 billion years of evolutionary optimisation.

...

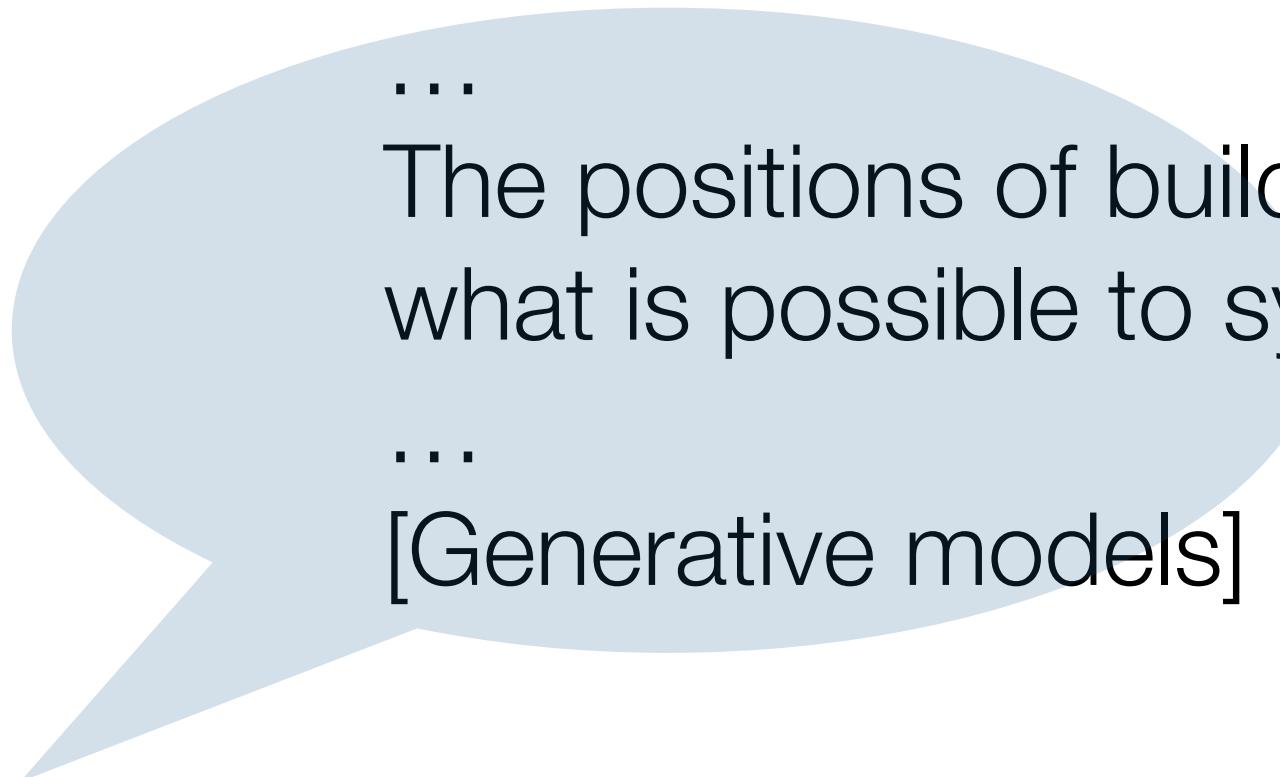
Since AlphaFold2 is trained on products of successful evolutionary optimisation, it is provided with an extremely strong signal

...

The positions of building blocks within a molecule are often explained not by function but by what is possible to synthesise with standard chemical processes

...

[Generative models] most likely to learn **synthesis biases** rather than function.



Michael Bronstein - The Road to Biology 2.0 Will Pass Through Black-Box Data

Challenges & Opportunities

The proteins we know have emerged over 3.5 billion years of evolutionary optimisation.

...

Since AlphaFold2 is trained on products of successful evolutionary optimisation, it is provided with an extremely strong signal

...

The positions of building blocks within a molecule are often explained not by function but by what is possible to synthesise with standard chemical processes

...

[Generative models] most likely to learn **synthesis biases** rather than function.

Michael Bronstein - The Road to Biology 2.0 Will Pass Through Black-Box Data

In other words: Molecular generative models cannot reveal substantial insights from data like *ChatGPT* with *language* or *AlphaFold* with *proteins*.

Challenges & Opportunities

The proteins we know have emerged over 3.5 billion years of evolutionary optimisation.

...

Since AlphaFold2 is trained on products of successful evolutionary optimisation, it is provided with an extremely strong signal

...

The positions of building blocks within a molecule are often explained not by function but by what is possible to synthesise with standard chemical processes

...

[Generative models] most likely to learn **synthesis biases** rather than function.

Michael Bronstein - The Road to Biology 2.0 Will Pass Through Black-Box Data

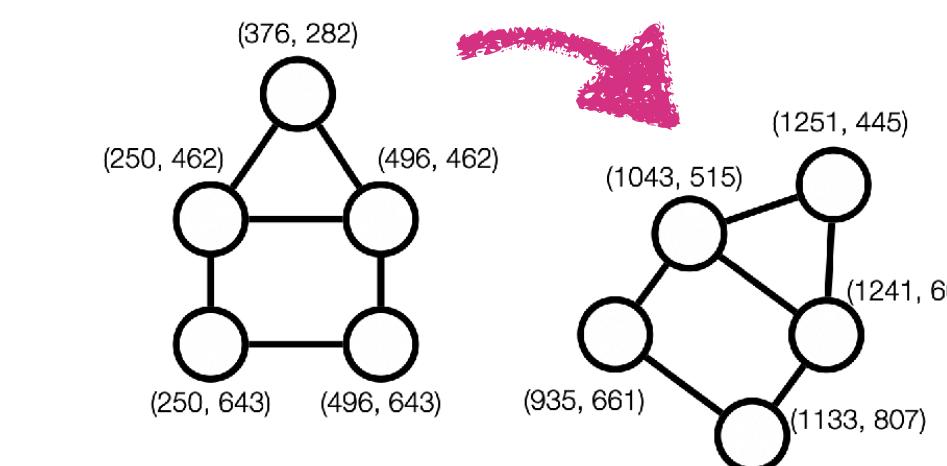
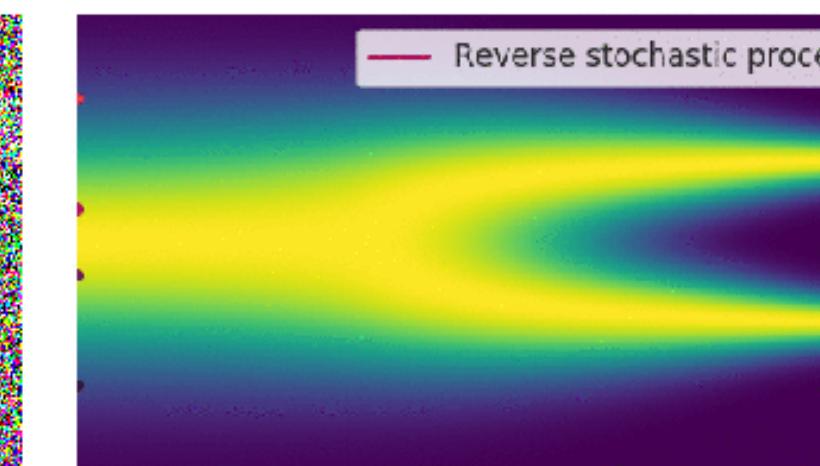
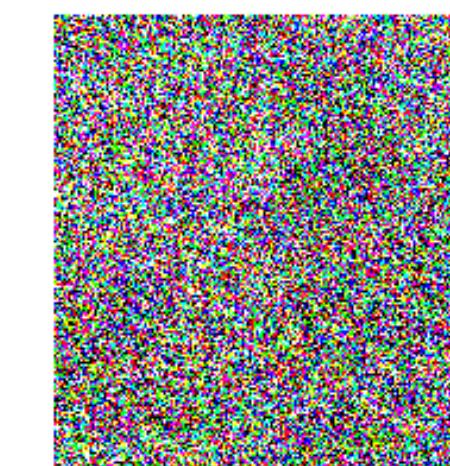
In other words (II): Training on QM9 to improve drugs is like learning on randomly generated text-sequences to learn about human communication.

Challenges & Opportunities

- **Benchmarks:** Existing benchmarks may not adequately evaluate model performance (in the domain of drug discovery).
- **Synthesizability:** Difficulty in generating molecules that are synthetically feasible.
- **Conditional Generation:** Current backbone NNs struggle with conditional molecule generation.
- **Best Practices:** Many arbitrary trick and hacks to make diffusion models and GNNs better.
- **Interpretability:** Models lack interpretability, making it hard to understand the reasoning behind predictions (e.g., why should this molecule fit into this pocket).
- **Synthesis biases:** No foundation model in sight.

Summary

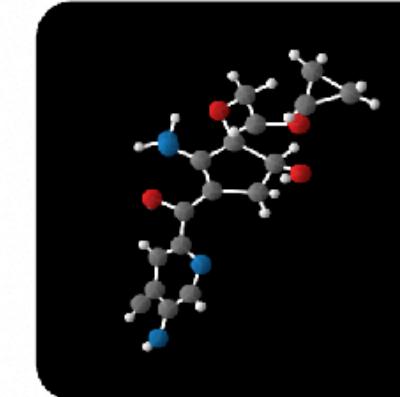
Method:



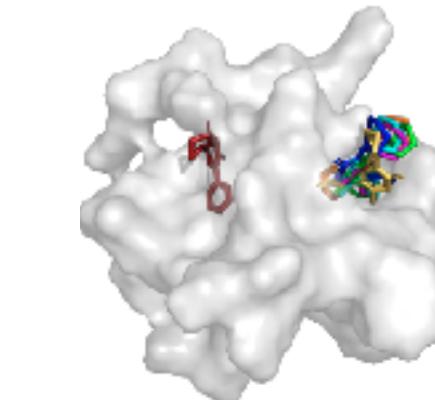
Diffusion Models

(Geometric) Graph Neural Networks

Applications:



Drug design

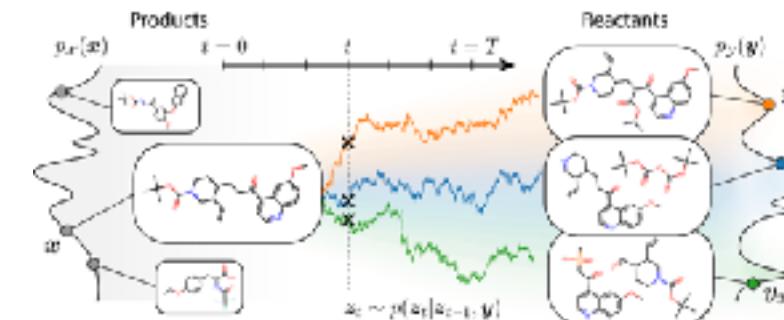


Docking



Structure Prediction

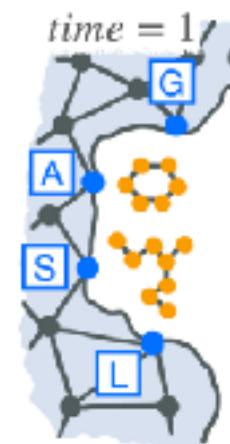
Future Directions:



Domain knowledge



pyg package 2.5.1 Testing PyTorch 2.3 passing Limiting passing docs passing contributions outcome black pyg
[Documentation](#) | [Paper](#) | [Colab Notebooks and Video Tutorials](#) | [External Resources](#) | [OGB Examples](#)



Benchmarks & Software

Better conditioning

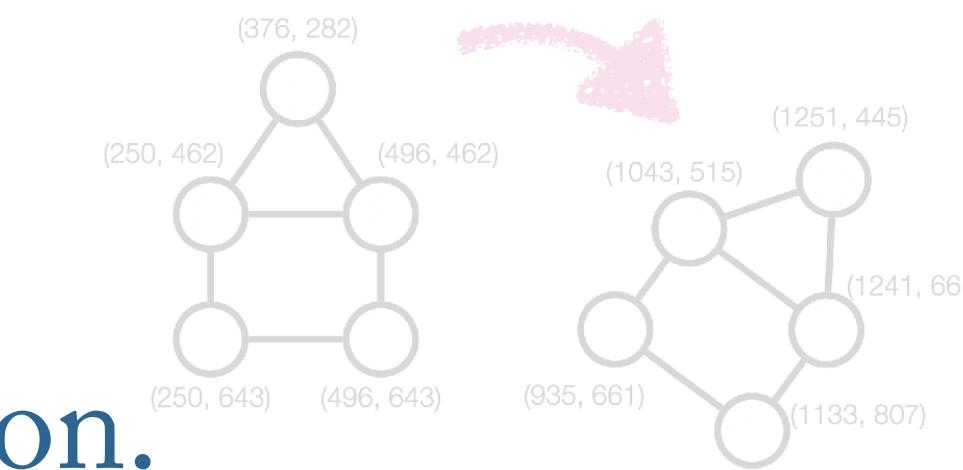
Summary

Method:



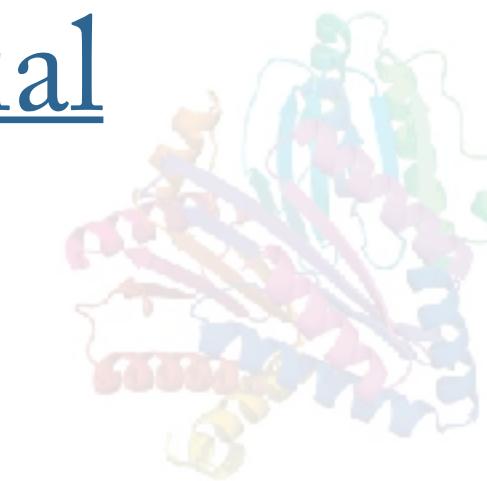
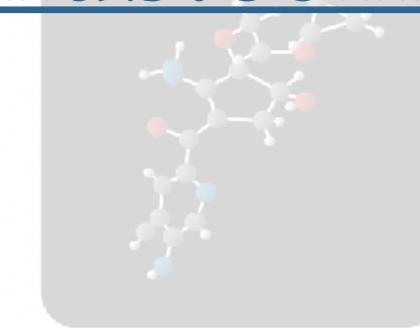
Thank you for your attention.

Diffusion Models



(Geometric) Graph Neural Networks

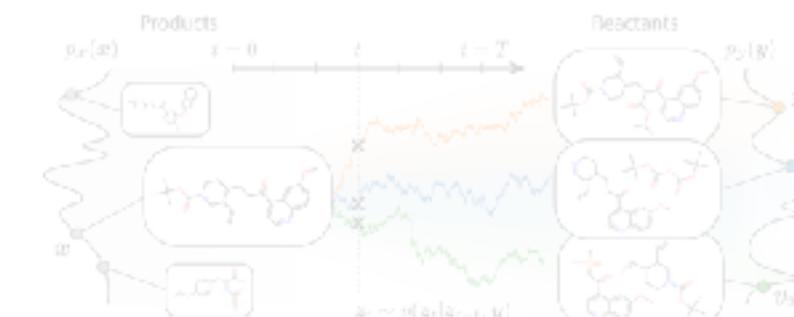
Applications:



Drug design? Docking? Structure Prediction?

gerrit.grossmann@dfki.de

Future Directions:



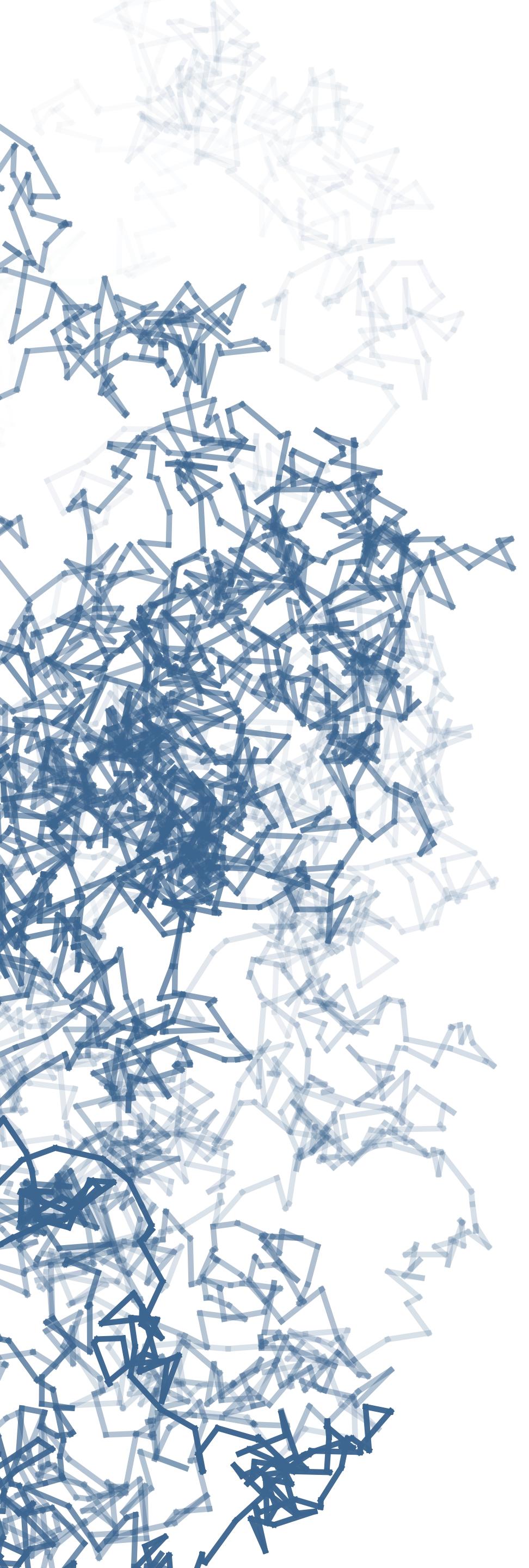
Domain knowledge



pytorch-geometric.readthedocs.io/en/latest/

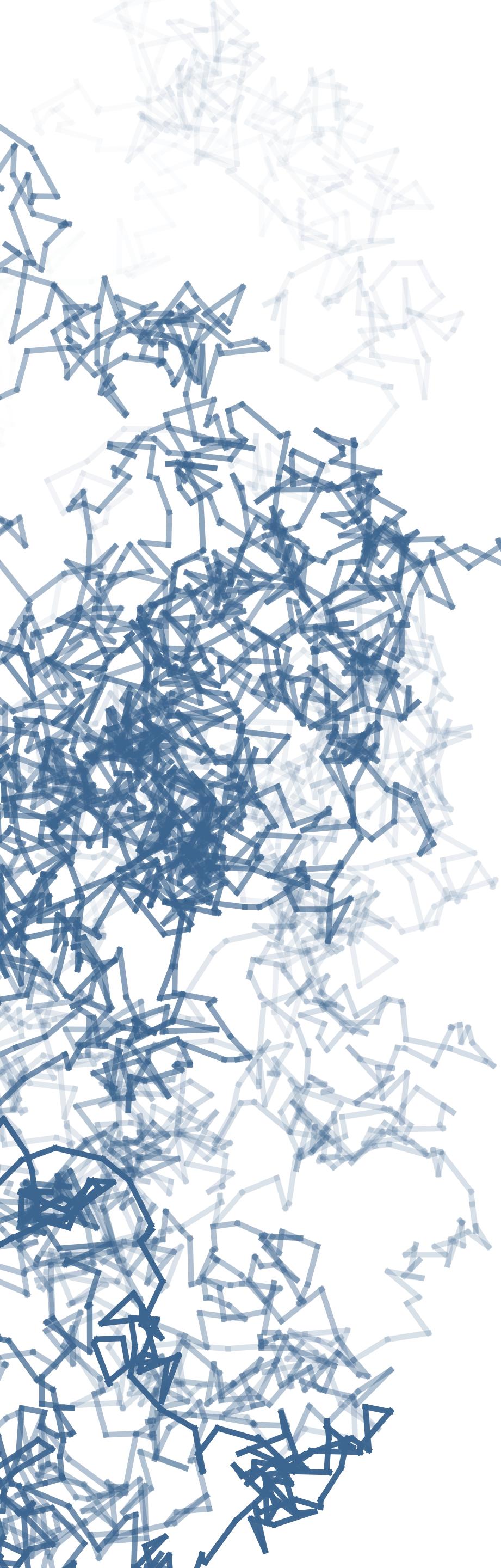


Better conditioning



Part VII

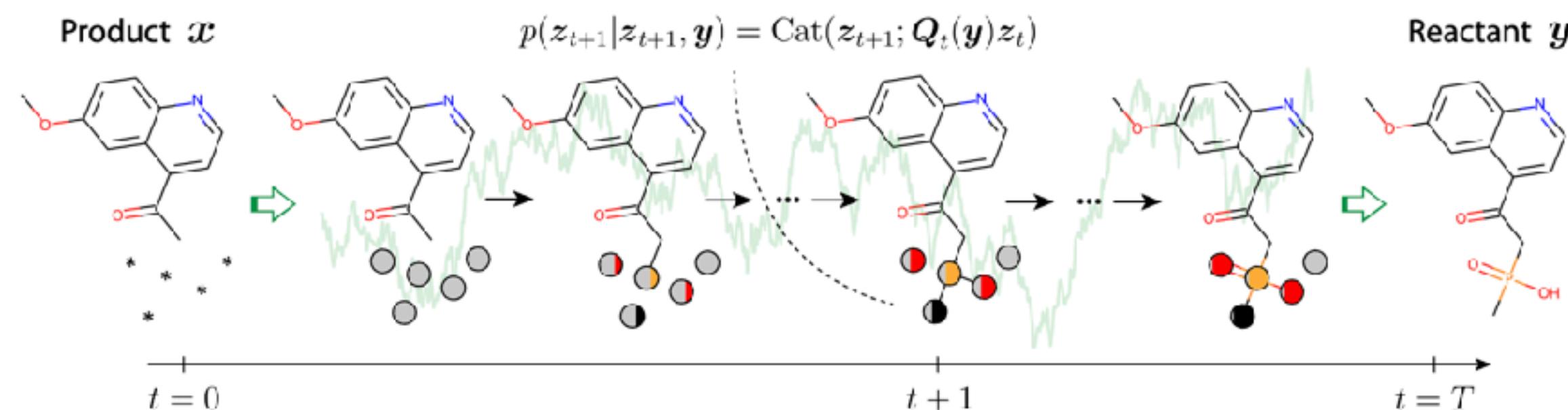
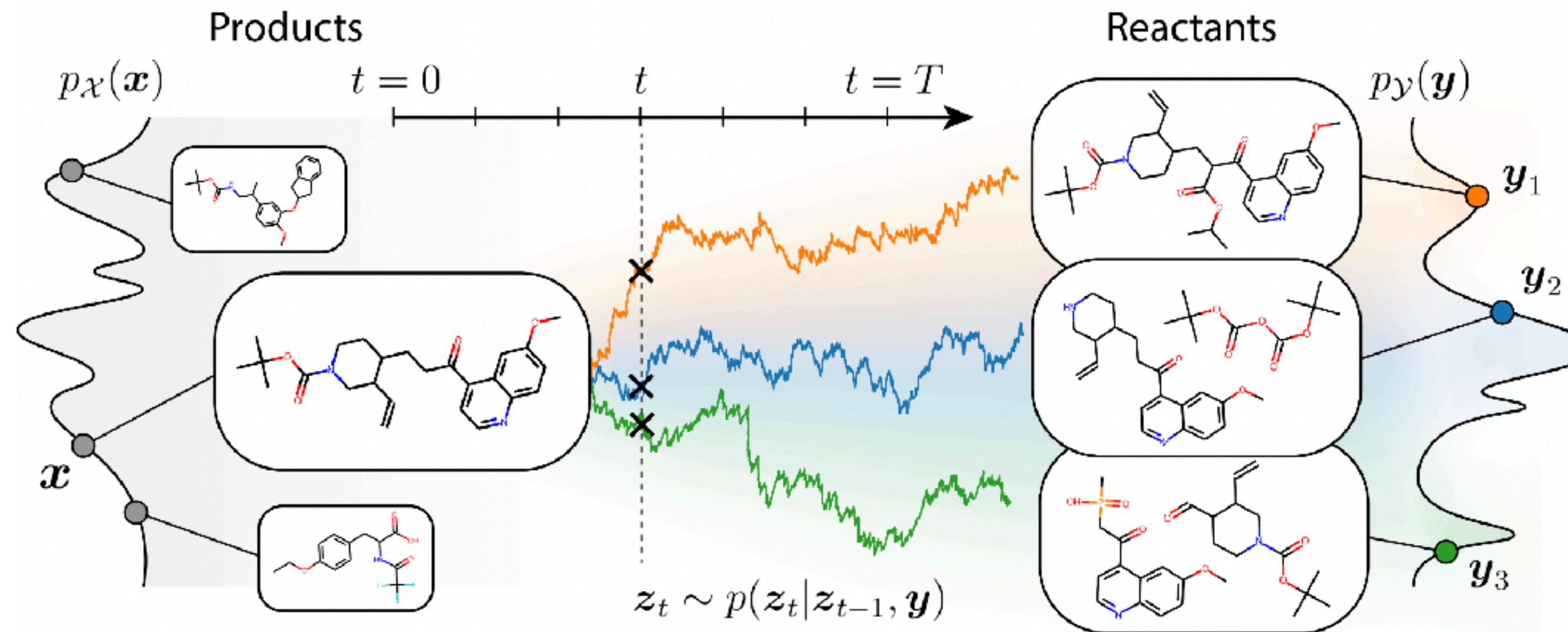
BACKUP



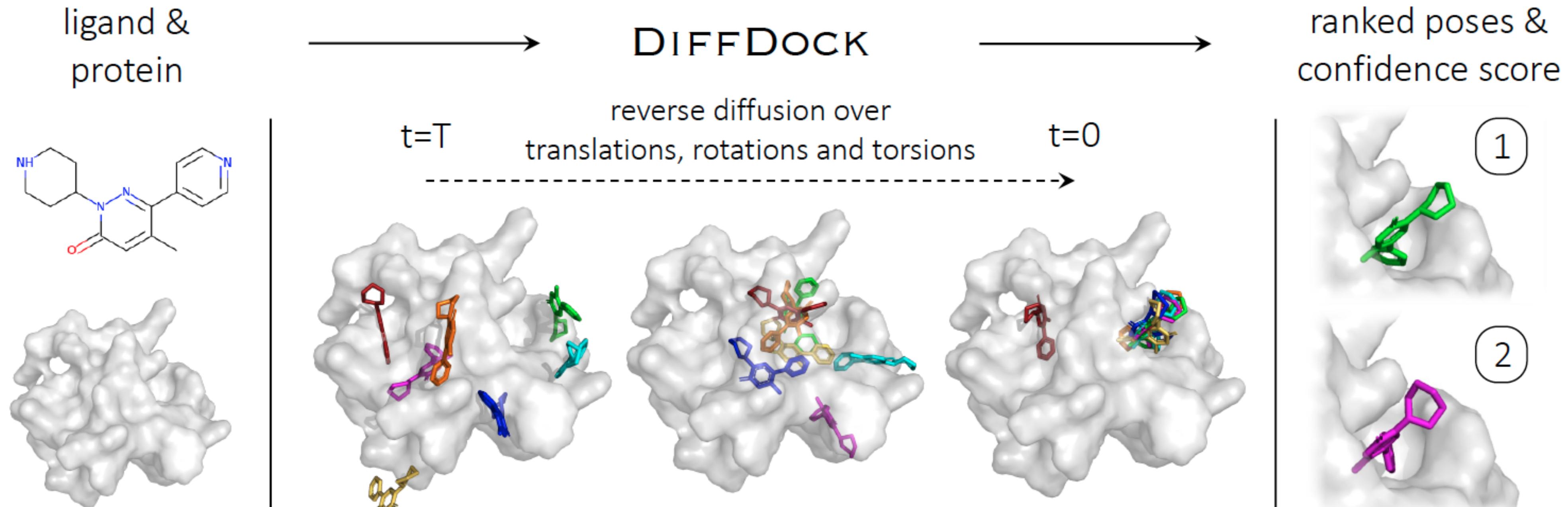
Part VI

BROADER APPLICATIONS

Retrosynthesis

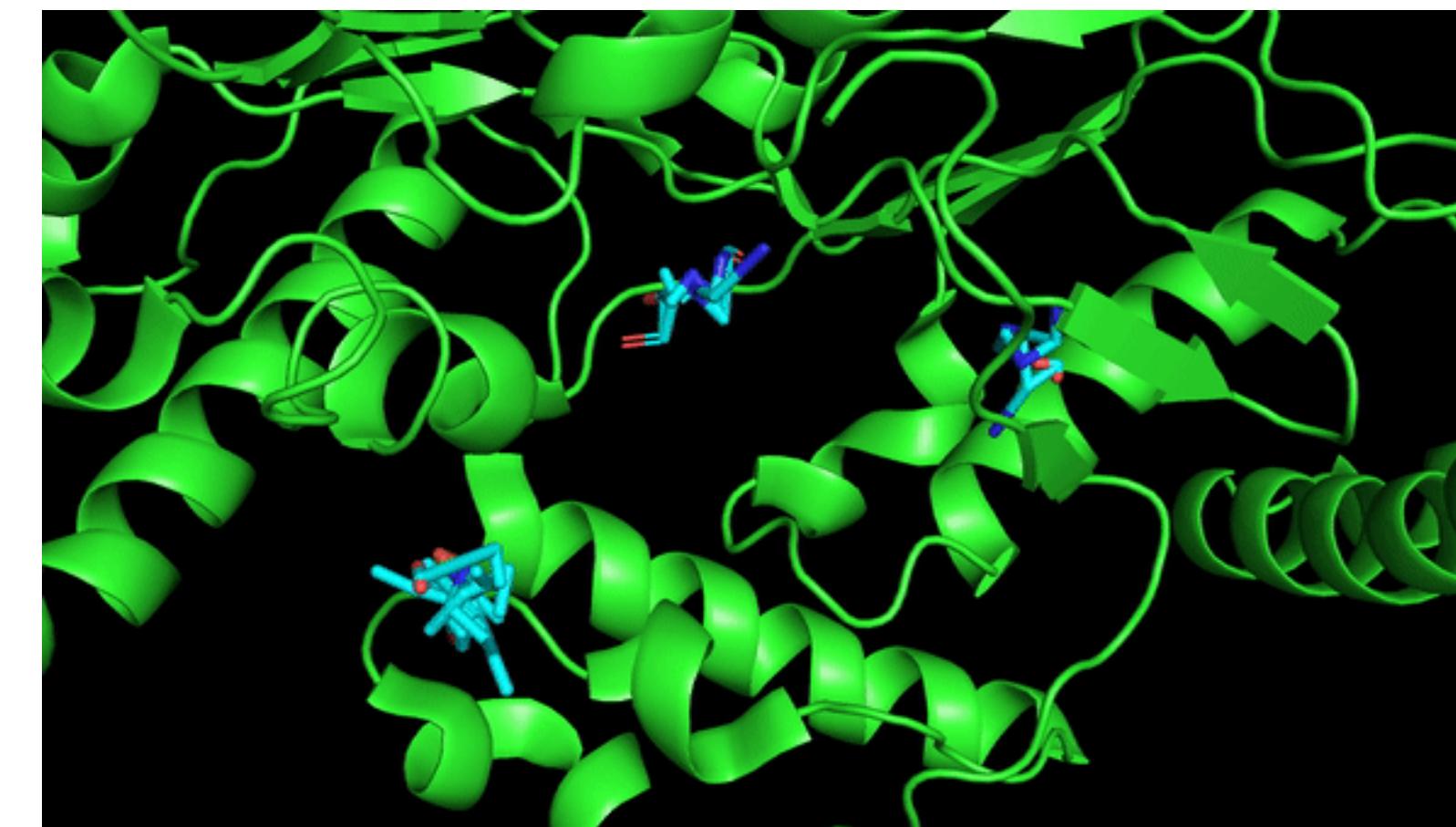
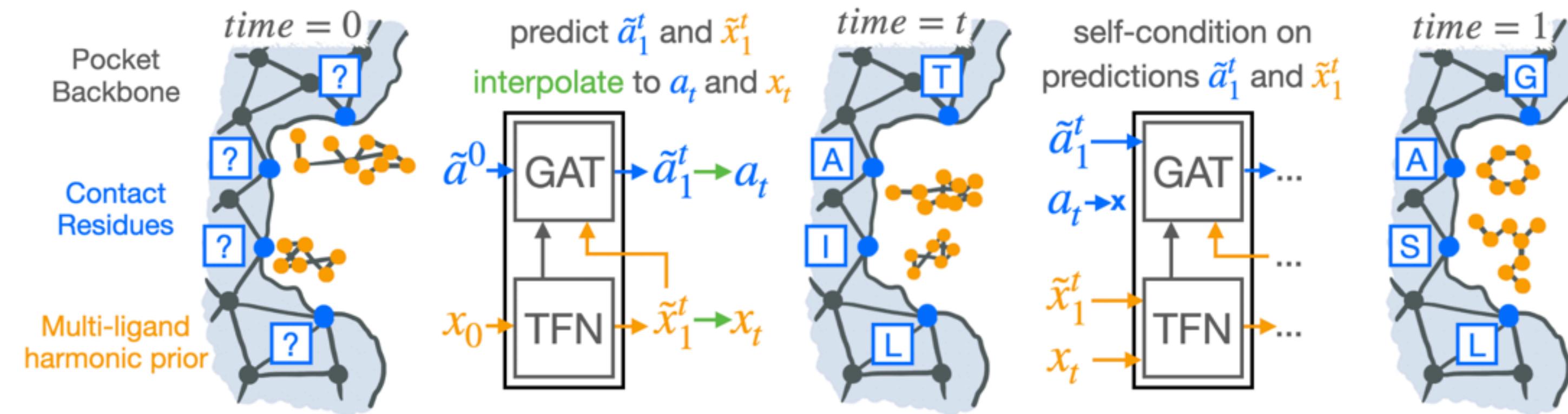


Docking



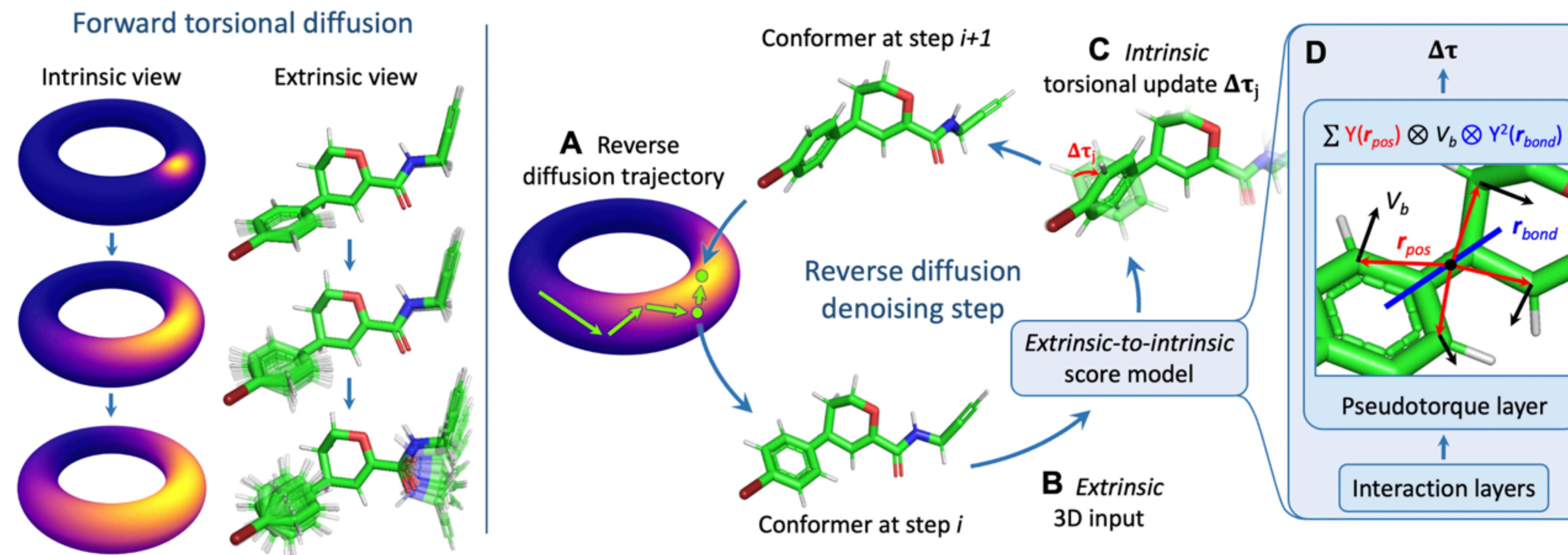
DiffDock: Diffusion Steps, Twists, and Turns for Molecular Docking (Corso et al.)

Flow Matching



Harmonic Self-Conditioned Flow Matching for Multi-Ligand Docking and Binding Site Design (Stärk et al.)

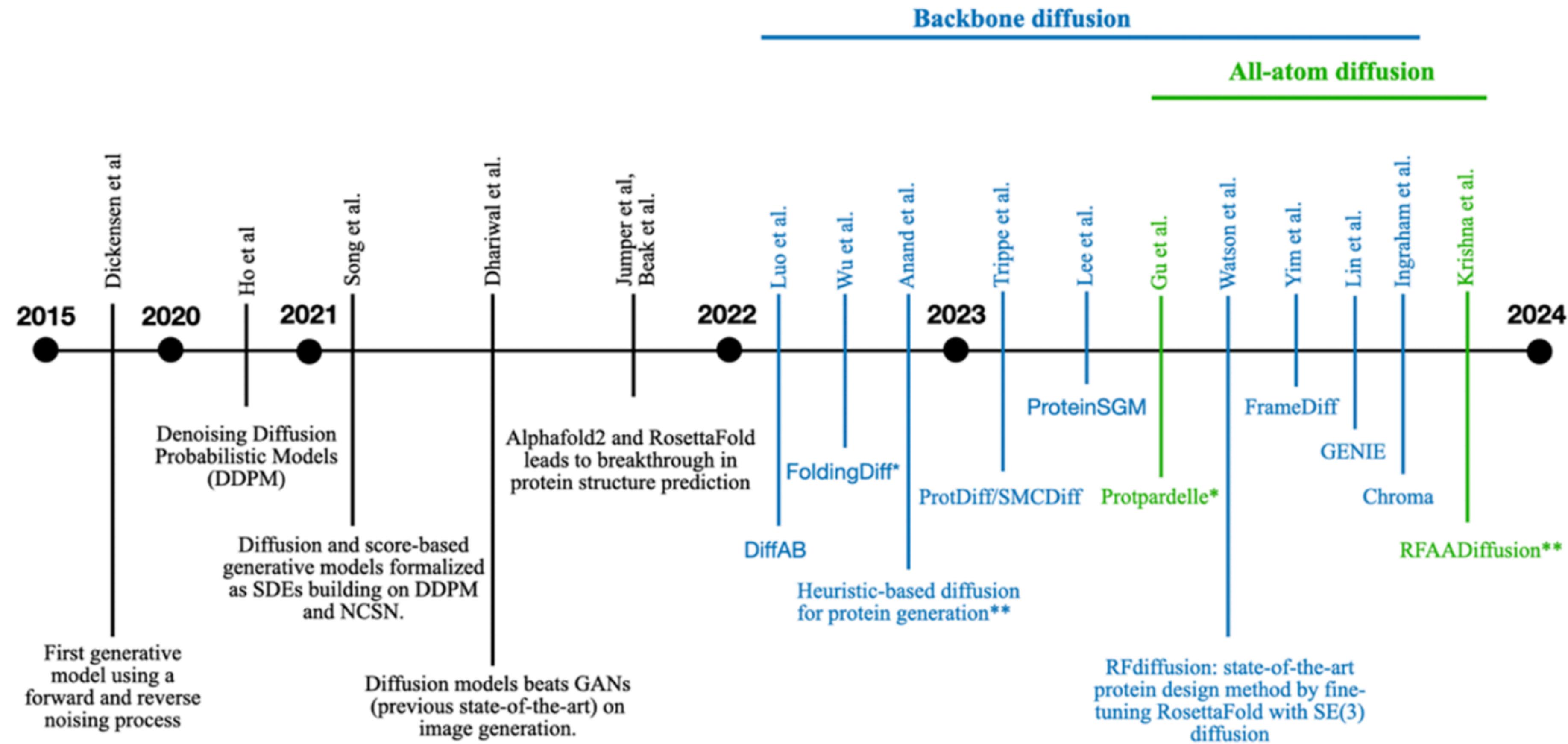
Torsional Diffusion



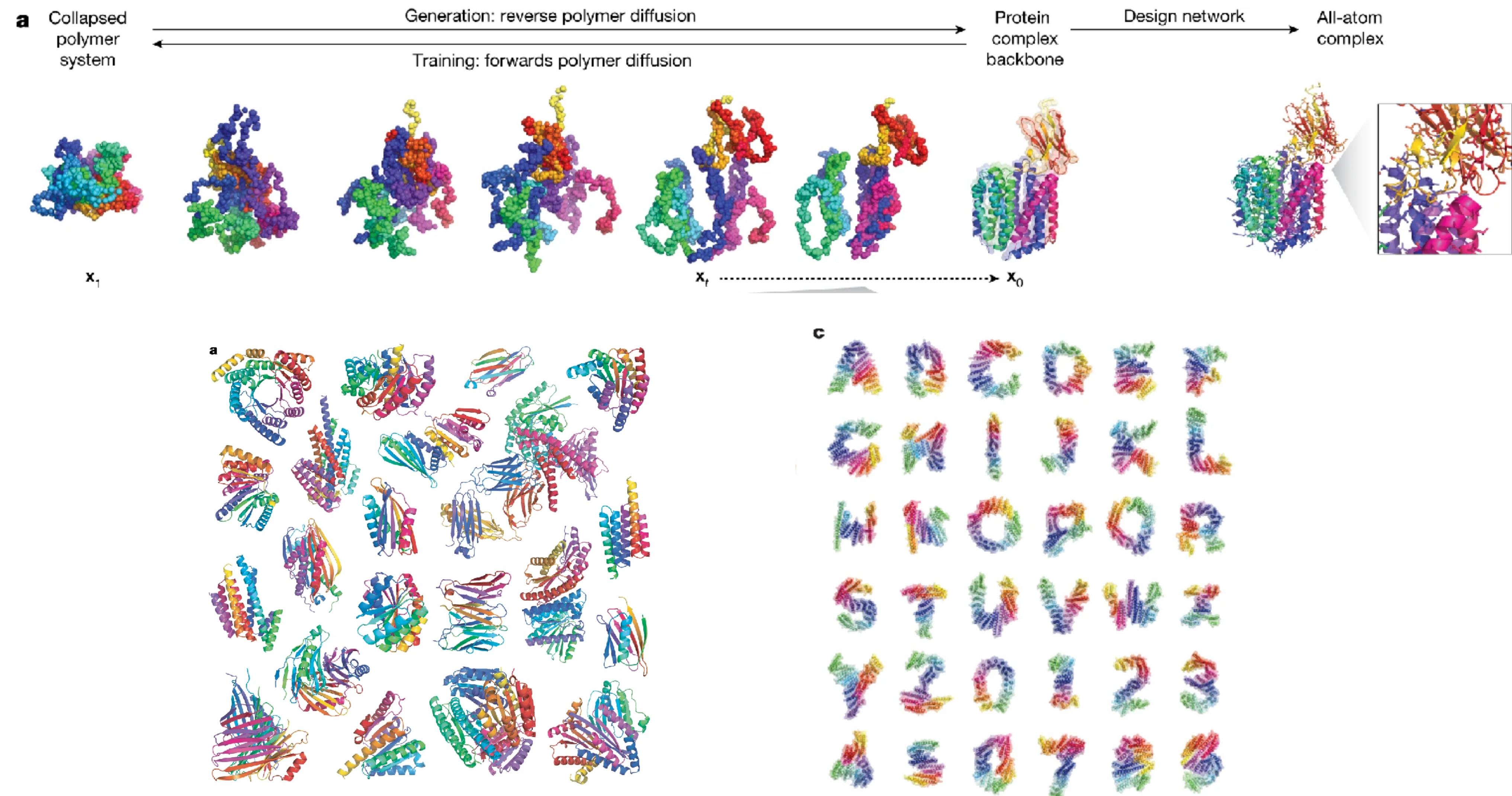
Torsional Diffusion for Molecular Conformer Generation (Jing et al.)

Proteins

Diffusion models for protein structure



Chroma



Antibodies

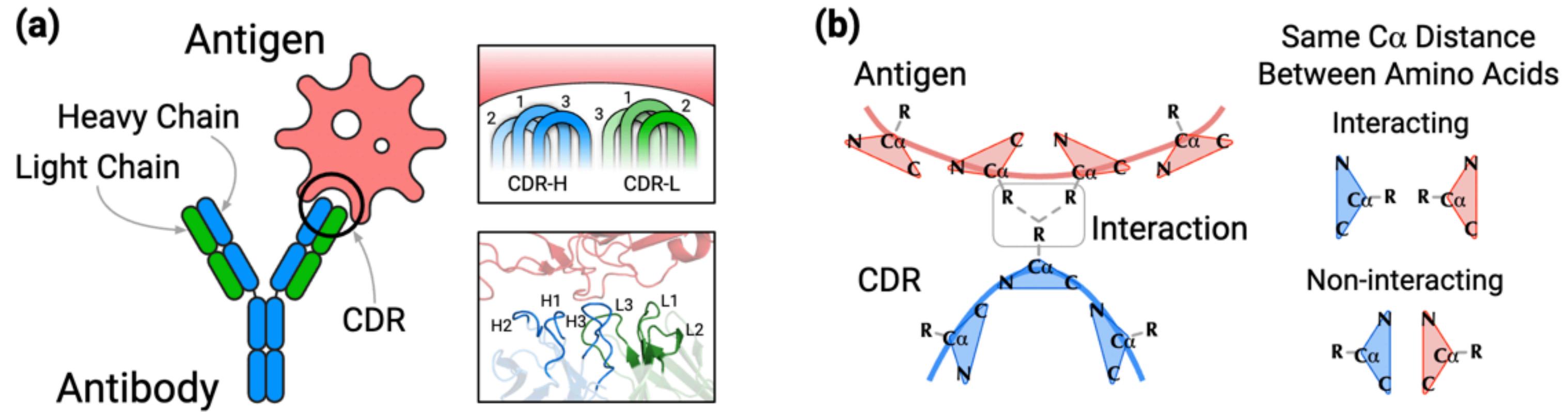
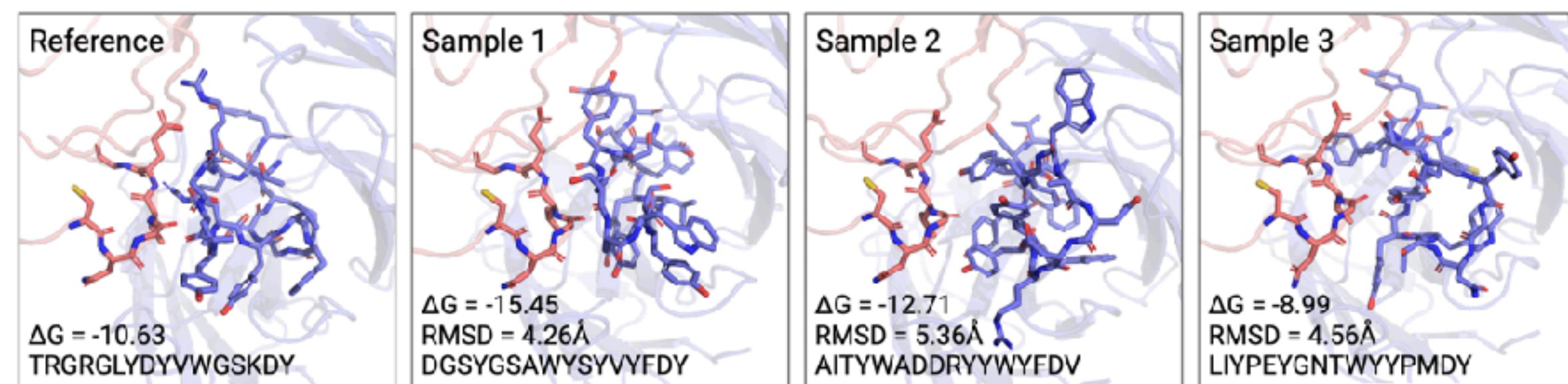
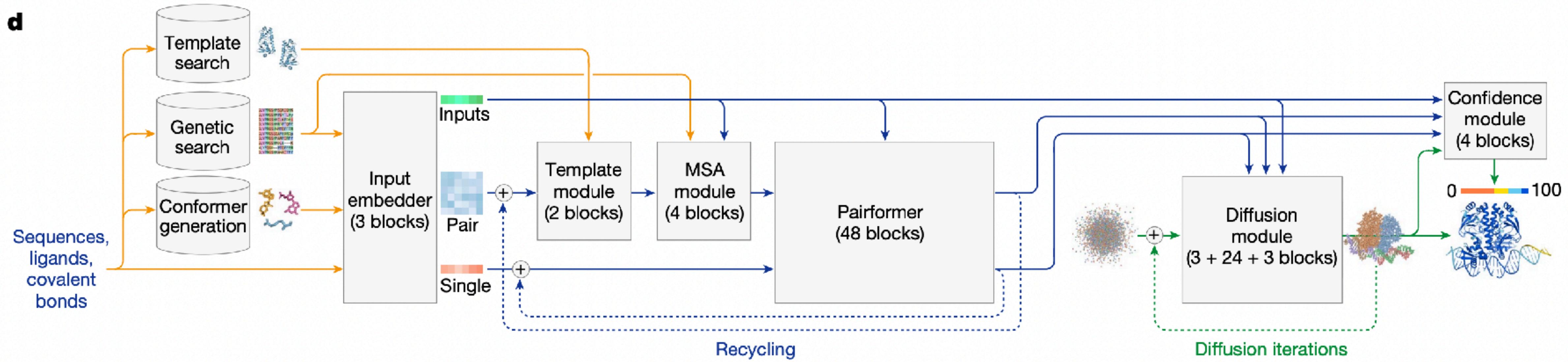


Figure 1: **(a)** Antibody-antigen complex structure and CDR structure. **(b)** The orientations of amino acids (represented by triangles) determine their side-chain orientations, which are key to inter-amino-acid interactions.



AlphaFold 3



Accurate structure prediction of biomolecular interactions with AlphaFold 3 (Abramson et al.)

Harmonic Self-Conditioned Flow Matching for Multi-Ligand Docking and Binding Site Design

HARMONIC SELF-CONDITIONED FLOW MATCHING FOR
MULTI-LIGAND DOCKING AND BINDING SITE DESIGN

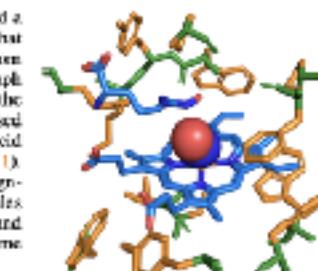
Hannes Stark¹, Bowen Jing¹, Regina Barzilay & Tommi Jaakkola¹
¹SAIL, Massachusetts Institute of Technology

ABSTRACT

A significant amount of protein function requires binding small molecules, including enzymatic catalysis. As such, designing binding pockets for small molecules has several impactful applications, ranging from drug synthesis to energy storage. Towards this goal, we first develop HARMONICFLow, an improved generative process over 3D protein-ligand binding structures based on our self-conditioned flow matching objective. FLOWSITE extends this flow model to jointly generate a protein pocket's discrete residue types and the molecule's binding 3D orientation. We show that HARMONICFlow improves upon state-of-the-art generative processes for docking in simplicity, generality, and average sample quality in pocket-level docking. Enabled by this structure modeling, FLOWSITE designs binding sites substantially better than baseline approaches.

1 INTRODUCTION

Designing proteins that can bind small molecules has many applications, ranging from drug synthesis to energy storage or gene editing. Indeed, a key part of any protein's function derives from its ability to bind and interact with other molecular species. For example, we may design proteins that act as antibodies that sequester toxins or design enzymes that enable chemical reactions through catalysts, which plays a major role in most biological processes. We develop FLOWSITE to address this design challenge by adapting a recent advance in deep learning (DL)-based protein design [Borodzik et al., 2022] and proteinaceous docking [Corso et al., 2022].



Specifically, we aim to design protein pockets to bind a certain small molecule (called ligand). We assume that we are given a protein pocket via the 3D backbone atom locations of its residues as well as the 2D chemical graph of the ligand. We do not assume any knowledge of the 3D structure or the binding pose of the ligand. Based on this information, our goal is to predict the amino acid identities at the given backbone locations (see Figure 1). We also consider the more challenging case of designing pockets that simultaneously bind multiple molecules and ions (which we call multi-ligand). Such multi-ligand binding pockets are important, for example, in enzyme design where two ligands correspond to reactants.

This task has not been addressed by deep learning yet. While deep learning has been successful in designing proteins that can bind to other proteins [Winston et al., 2022], designing (multi-)ligand binders is different and arguably harder in various aspects. For example, no evolutionary information is directly available, unlike when modeling interactions between amino acids only. Existing successes, such as designing 6 drug-binding proteins [Polizzi

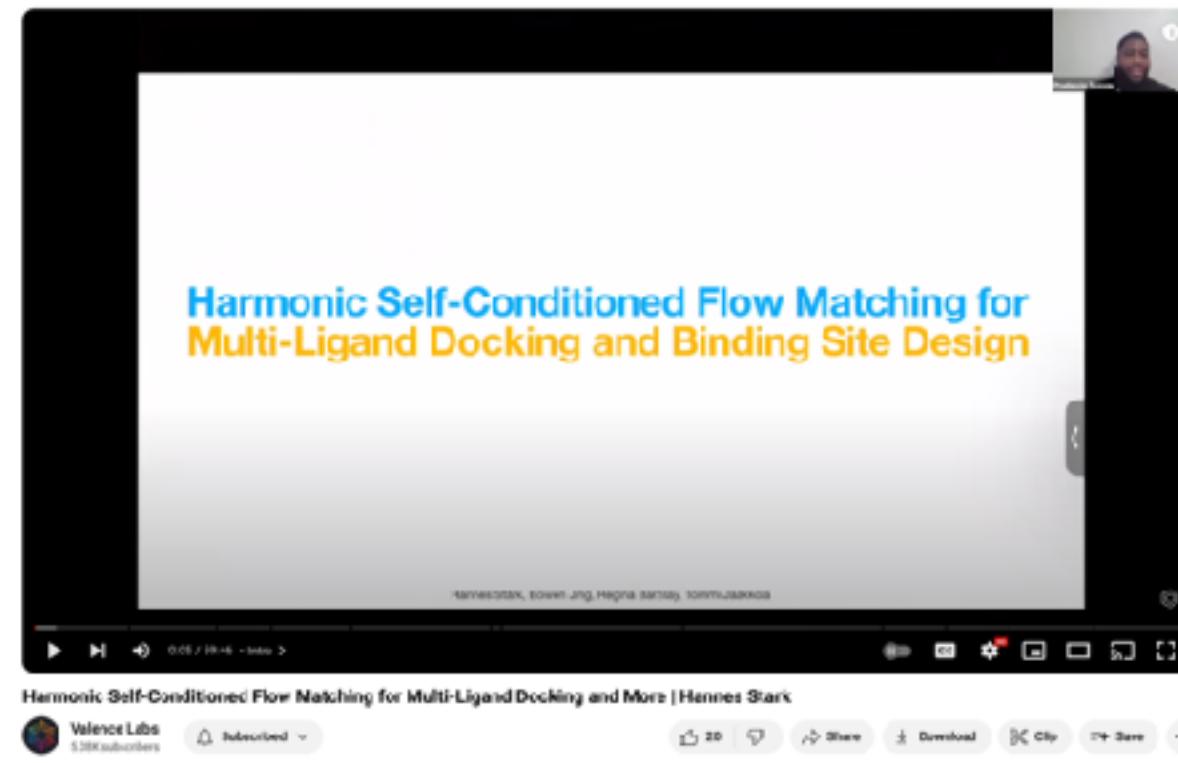


Figure 1: Binding site design. Given the backbone (green) and multi-ligand without smarts, FLOWSITE generates residue types and structure (orange) to bind the multi-ligand and its identity generated structure (blue). The majority of the pocket is omitted for visibility.

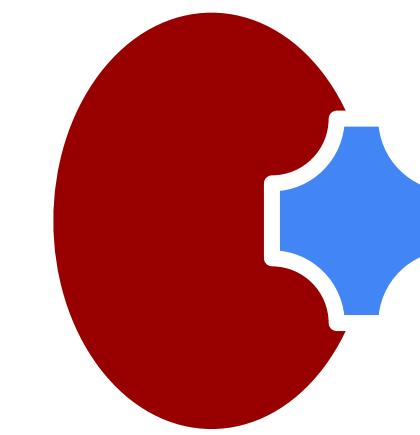
[github.com/
HannesStark/FlowSite](https://github.com/HannesStark/FlowSite)

Harmonic Self-Conditioned Flow Matching for Multi-Ligand Docking and Binding Site Design

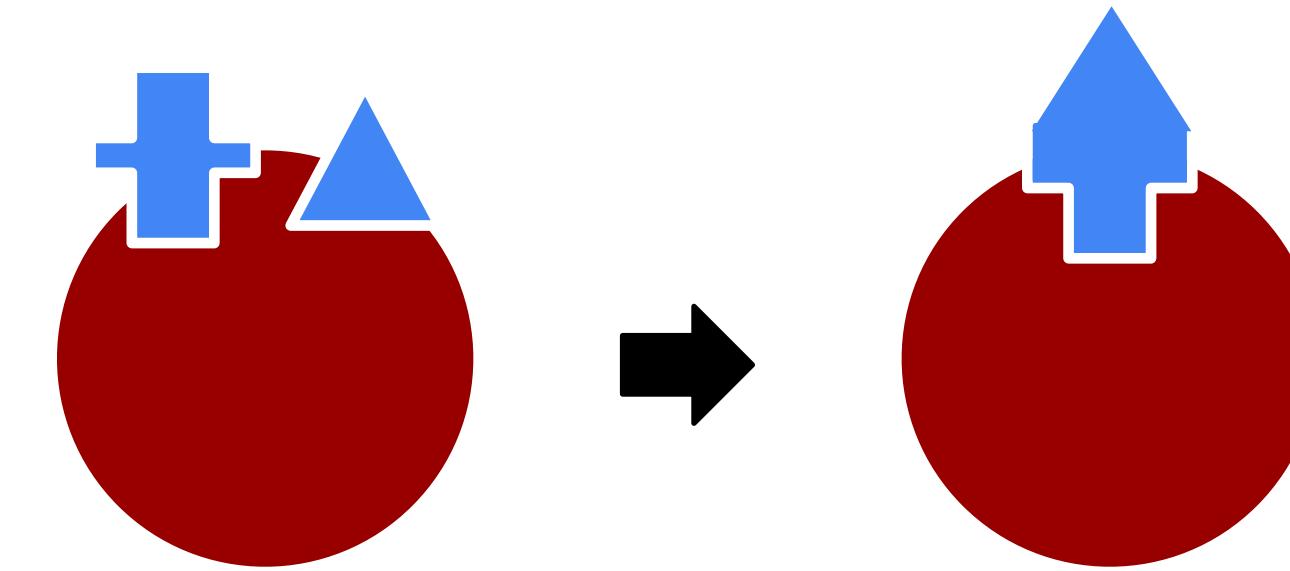
Harmonic Self-Conditioned Flow Matching
for Multi-Ligand Docking and Binding Site Design

Problem Setting

Single ligand binding site design



Multi ligand binding site design



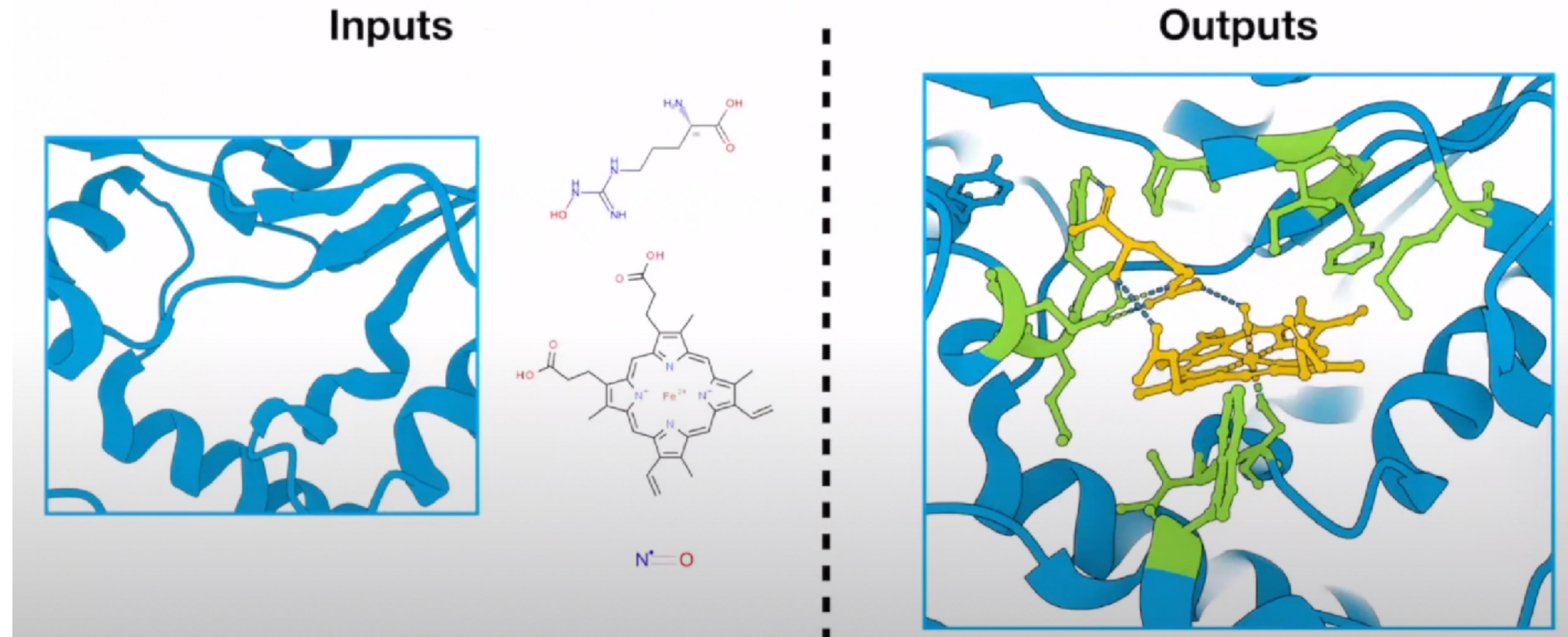
Given: toxic small **molecule**

Task: design **enzyme** to trap molecule

Problem Setting

Input: Enzyme backbone and 2D ligands

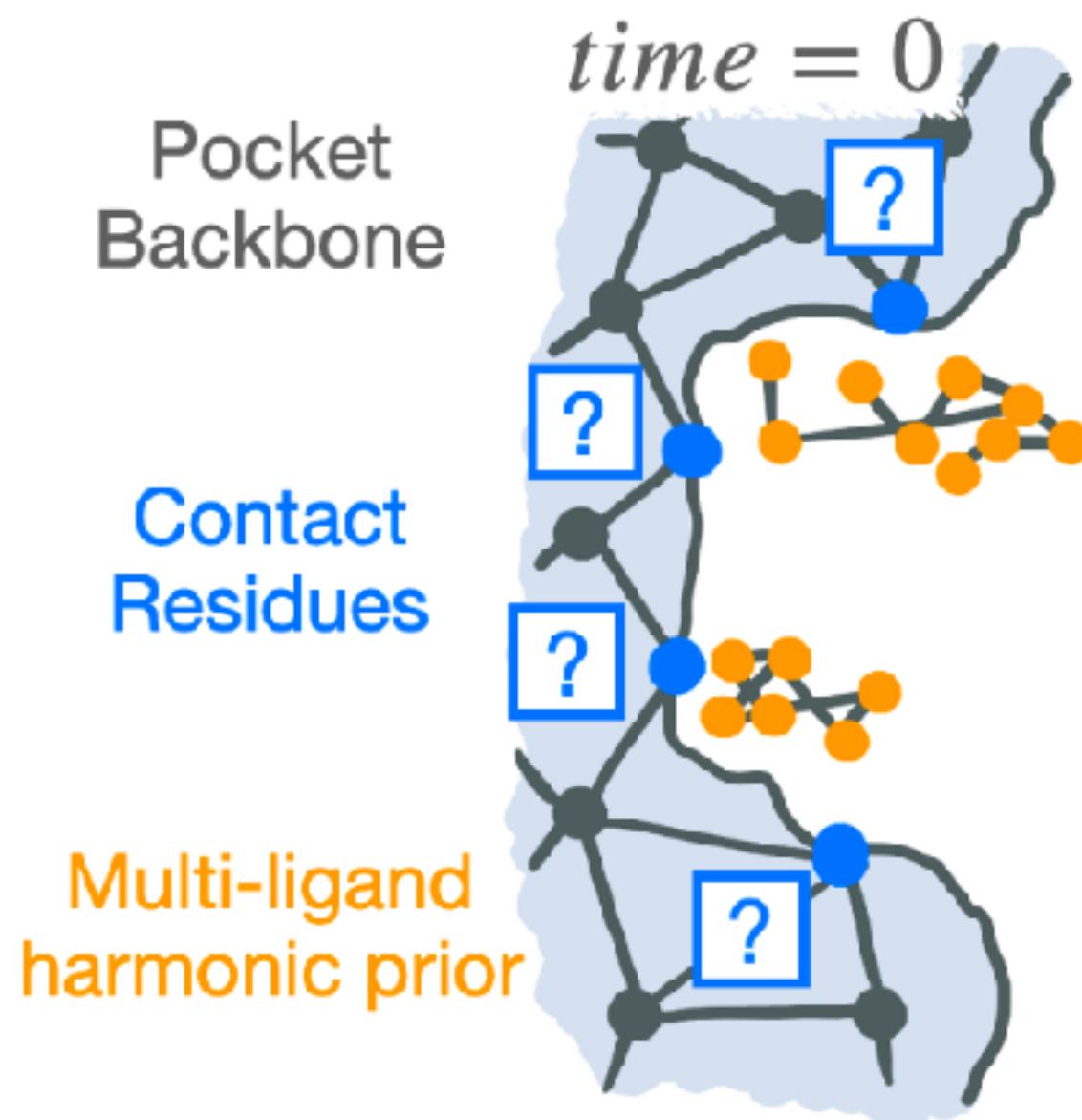
Output: Enzyme residues and 3D conformers of all ligands



Problem Setting

Input: Enzyme backbone and 2D ligands

Output: Enzyme residues and 3D conformers of all ligands

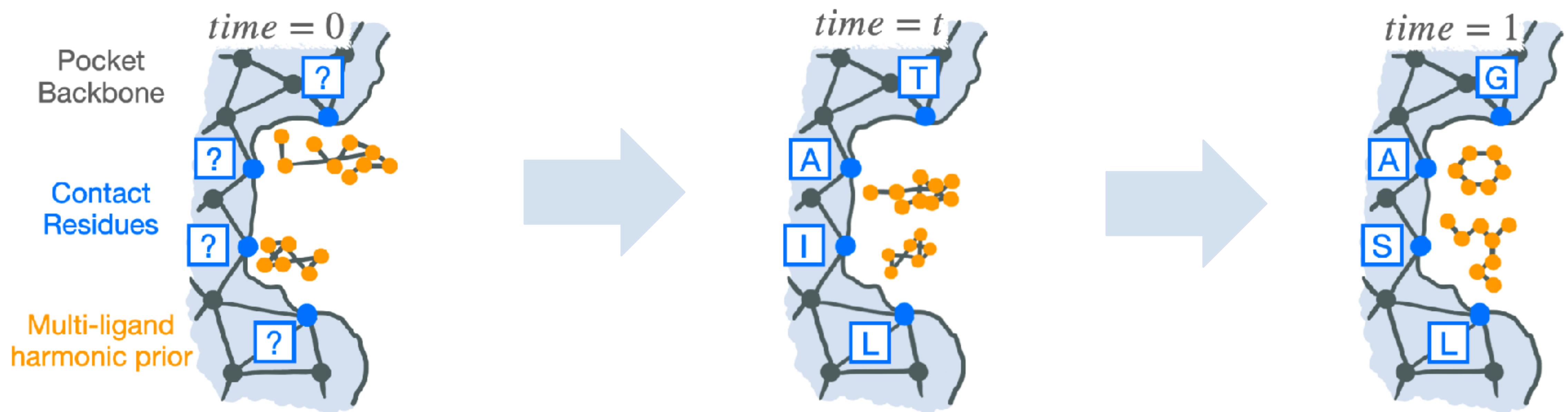


Start with random **residues** and
random number of **ligands** 3D
conformations of **ligands**.

Problem Setting

Input: Enzyme backbone and 2D ligands

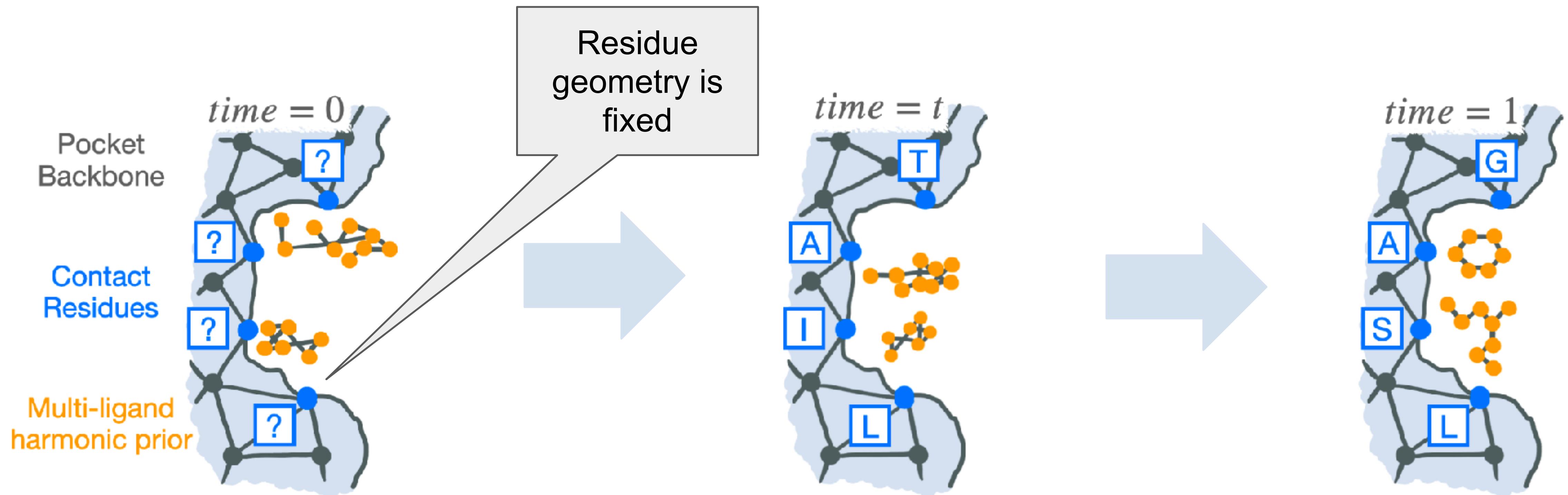
Output: Enzyme residues and 3D conformers of all ligands



Problem Setting

Input: Enzyme backbone and 2D ligands

Output: Enzyme residues and 3D conformers of all ligands



Harmonic Self-Conditioned Flow Matching

for Multi-Ligand Docking and Binding Site Design

Harmonic

Fancy prior instead of
normal distribution

Self-Conditioned

Simple trick to make
training more stable

Flow Matching

Computationally nice way to
model normalizing flows

Harmonic

Fancy prior instead of
normal distribution

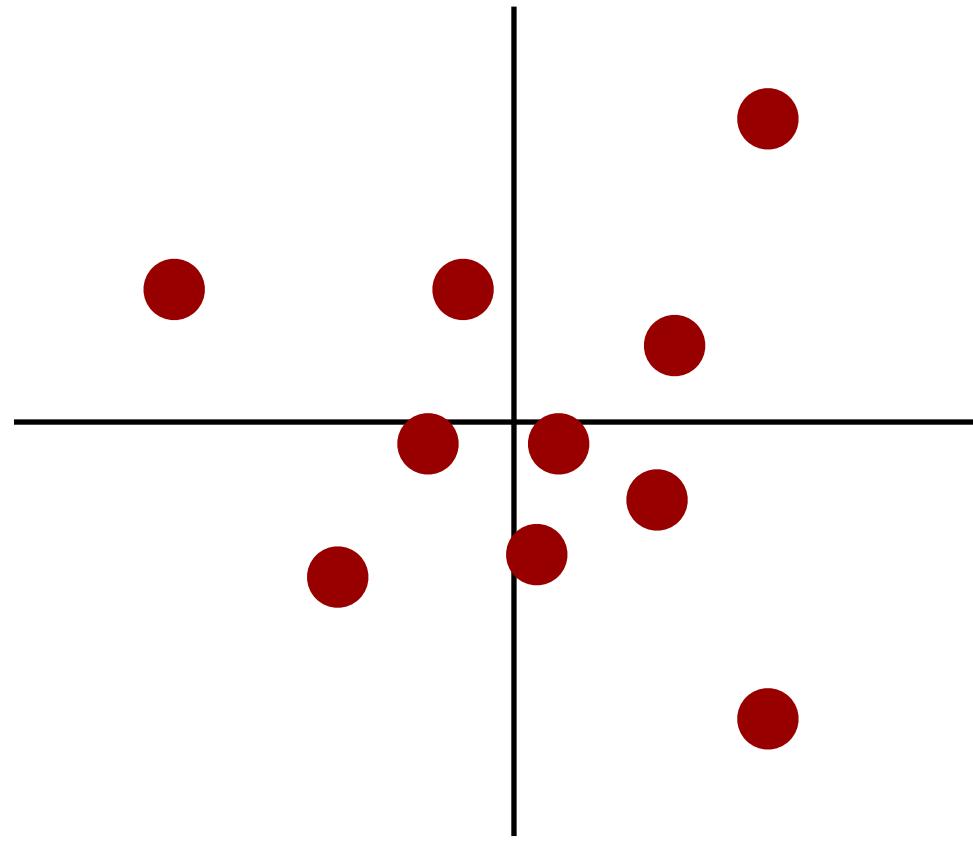
Self-Conditioned

Simple trick to make
training more stable

Flow Matching

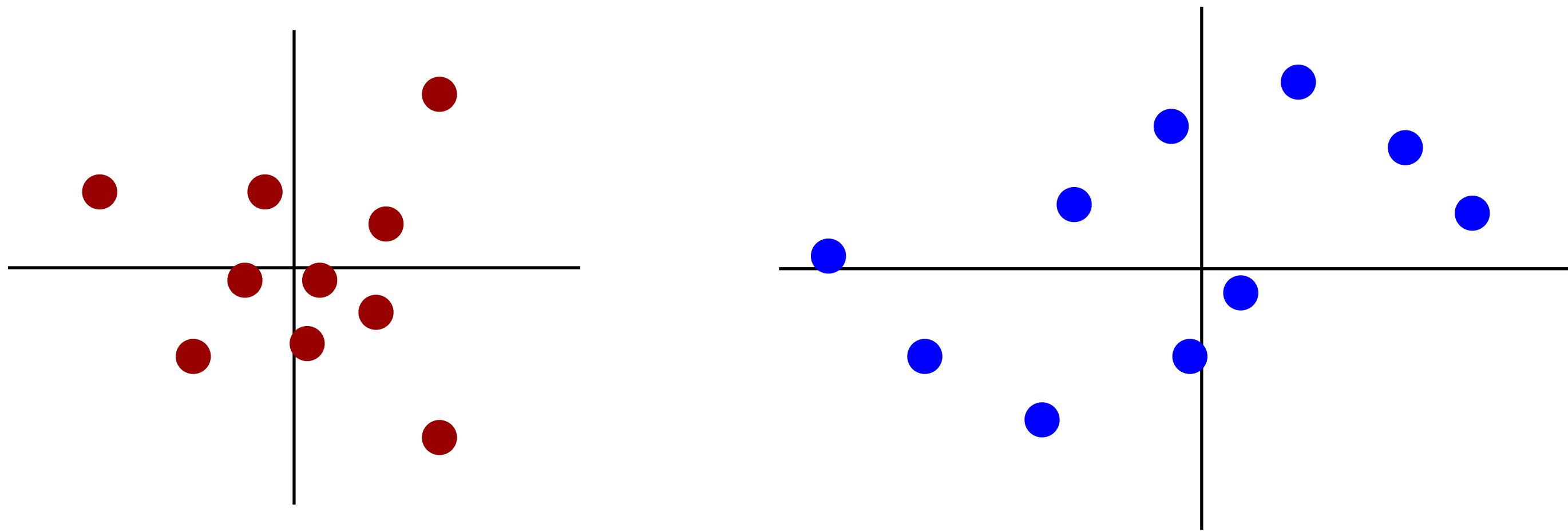
Computationally nice way to
model normalizing flows

Conditional Flow Matching



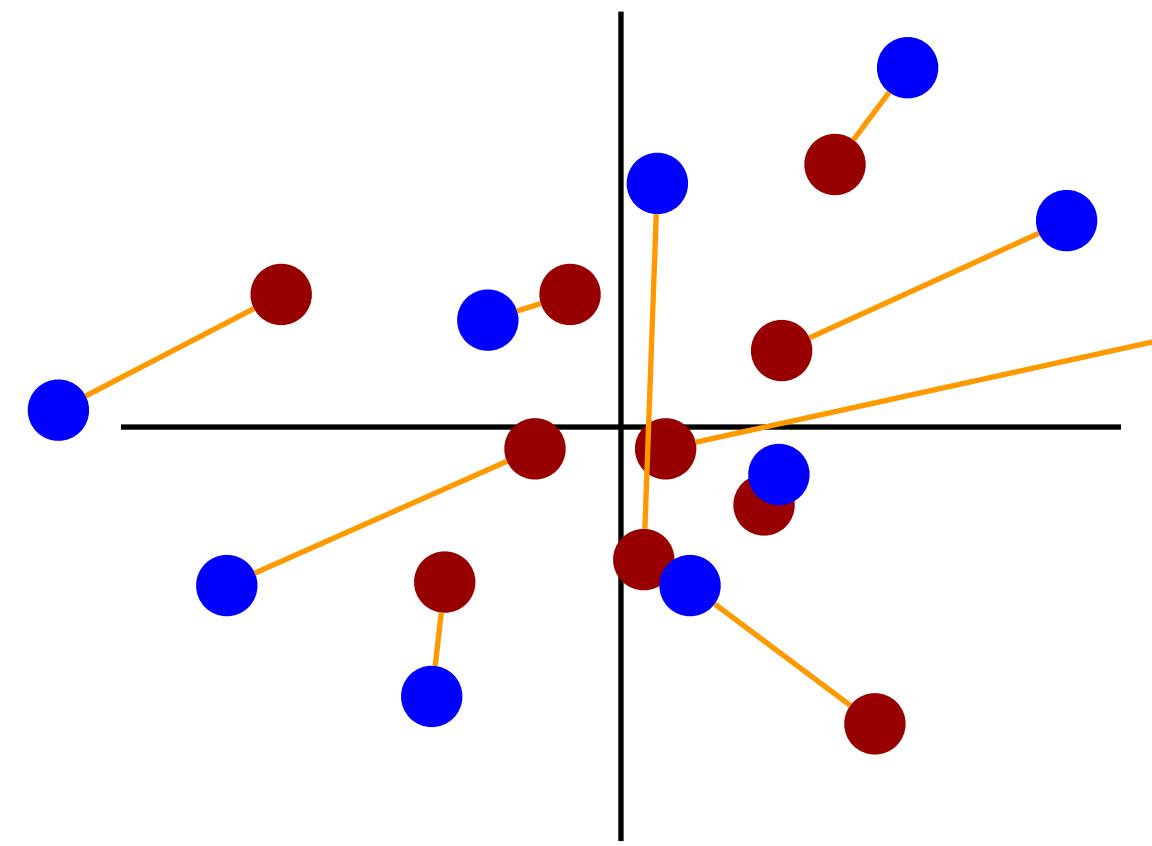
1. Randomly sample data points from your **prior**.
2. Sample data points from your **training data**.
3. Find a **1:1 matching** between the two empirical distributions.
4. Learn a time-varying vector field that transforms the **prior** into the **data** distribution using linear interpolation.

Conditional Flow Matching



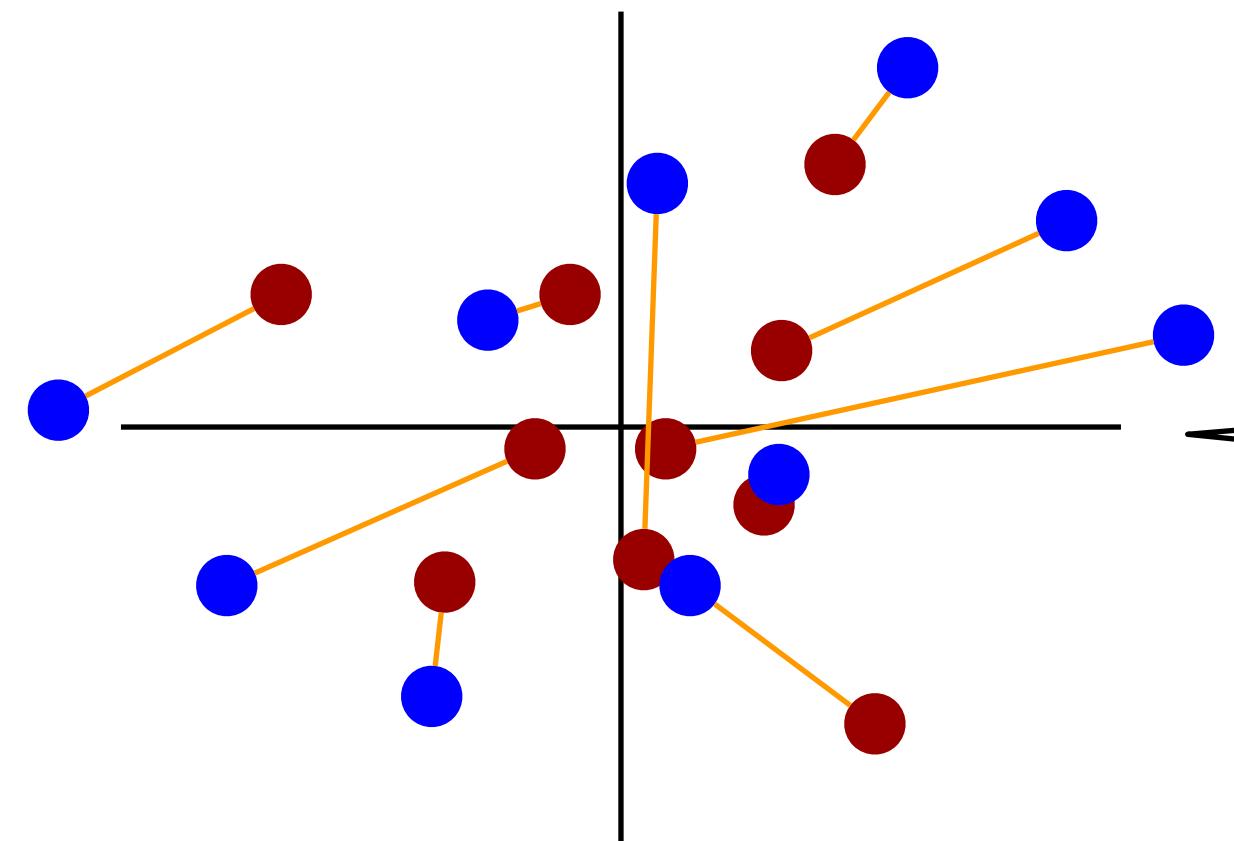
1. Randomly sample data points from your **prior**.
2. Sample data points from your **training data**.
3. Find a **1:1 matching** between the two empirical distributions.
4. Learn a time-varying vector field that transforms the **prior** into the **data** distribution using linear interpolation.

Conditional Flow Matching



1. Randomly sample data points from your **prior**.
2. Sample data points from your **training data**.
3. Find a **1:1 matching** between the two empirical distributions.
4. Learn a time-varying vector field that transforms the **prior** into the **data** distribution using linear interpolation.

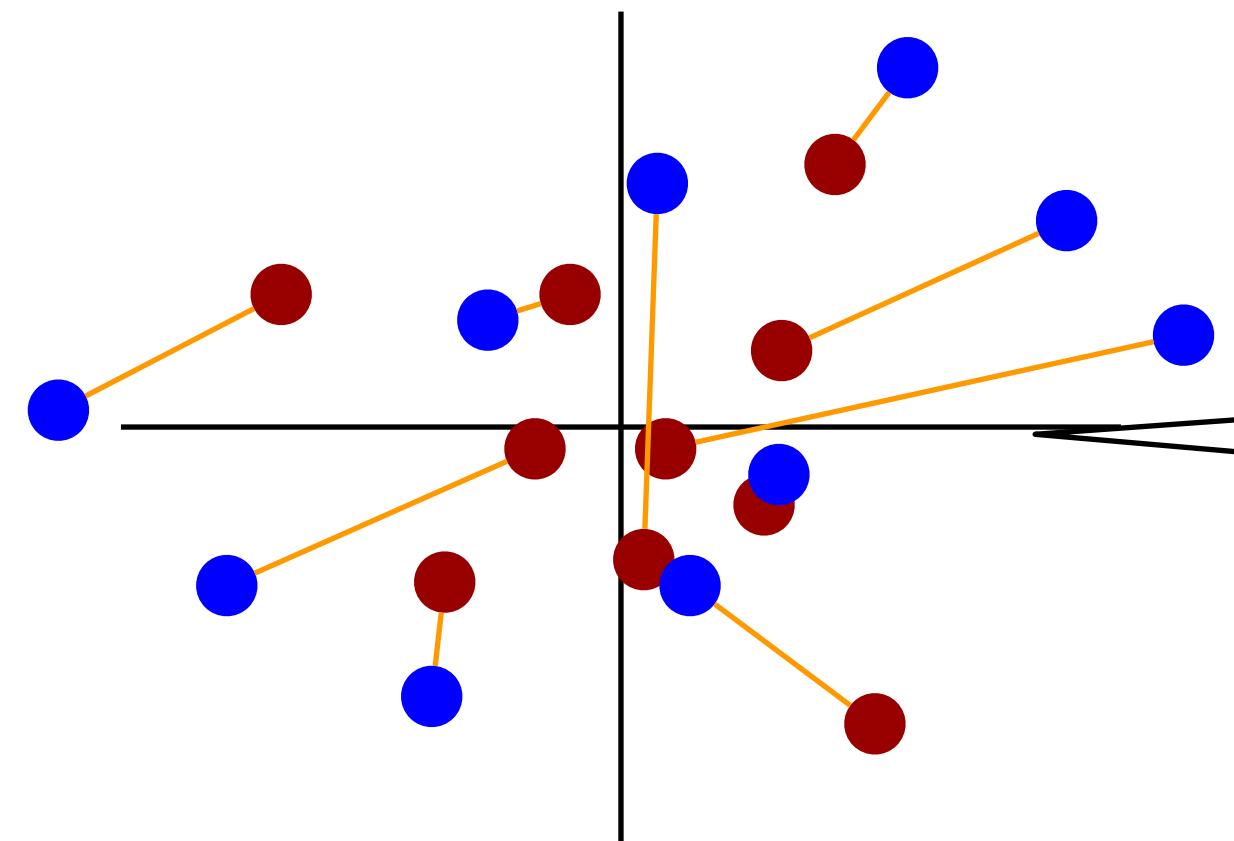
Conditional Flow Matching



It is called “**conditional**” because learn a vector field, conditioned on the start and end point.

1. Randomly sample data points from your **prior**.
2. Sample data points from your **training data**.
3. Find a **1:1 matching** between the two empirical distributions.
4. Learn a time-varying vector field that transforms the **prior** into the **data** distribution using linear interpolation.

Conditional Flow Matching

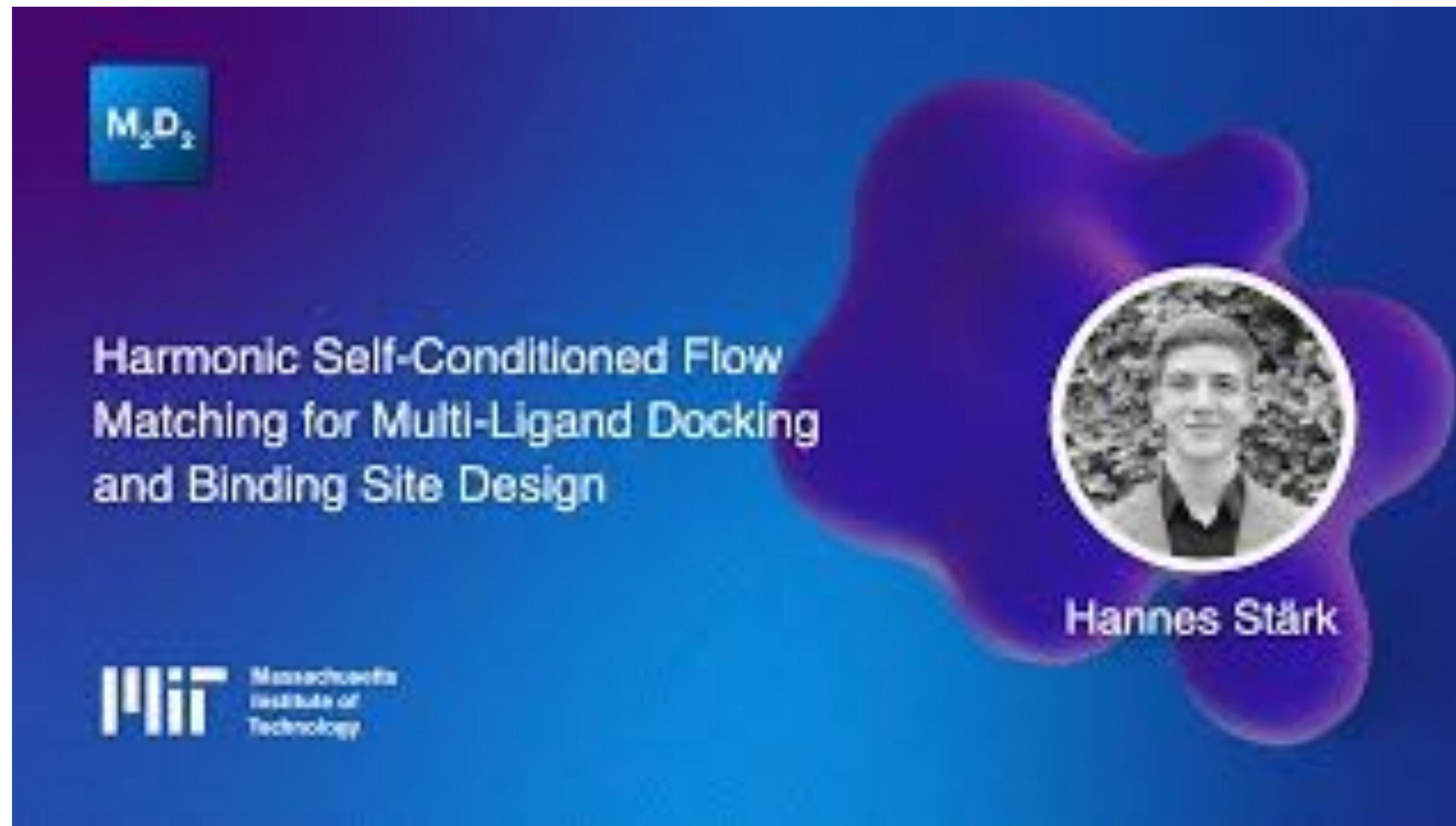


Diffusion Trick:

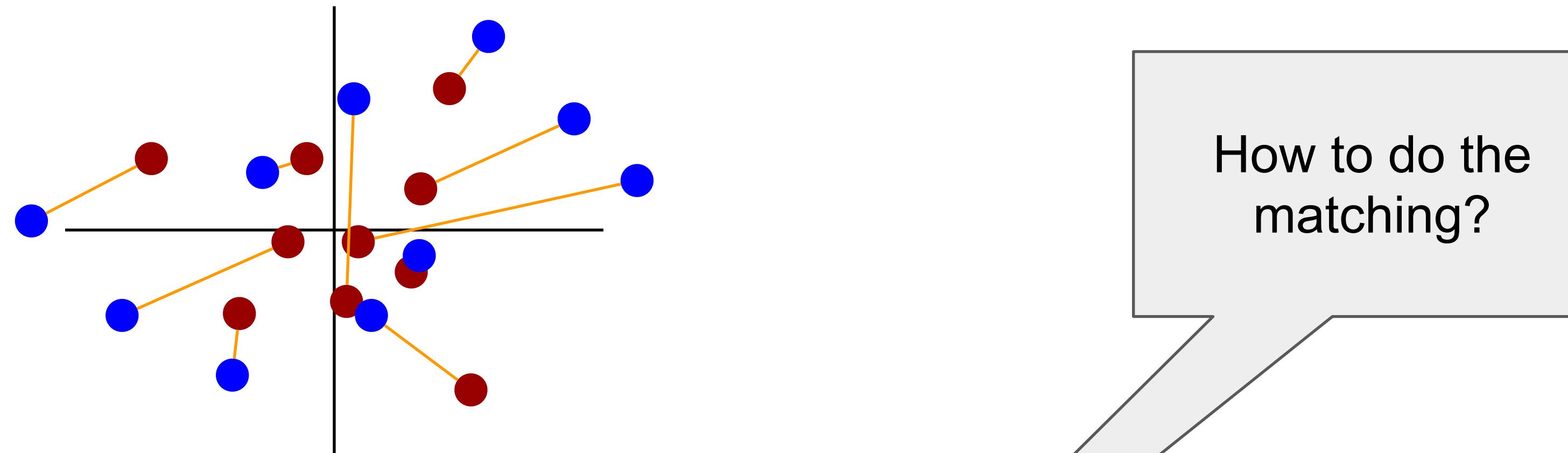
Implicitly learn the vector field by learning to predict the end point.

1. Randomly sample data points from your **prior**.
2. Sample data points from your **training data**.
3. Find a **1:1 matching** between the two empirical distributions.
4. Learn a time-varying vector field that transforms the **prior** into the **data** distribution using linear interpolation.

Inference



Conditional Flow Matching



1. Randomly sample data points from your **prior**.
2. Sample data points from your **training data**.
3. Find a **1:1 matching** between the two empirical distributions.
4. Learn a time-varying vector field that transforms the **prior** into the **data** distribution using linear interpolation.

Conditional Flow Matching

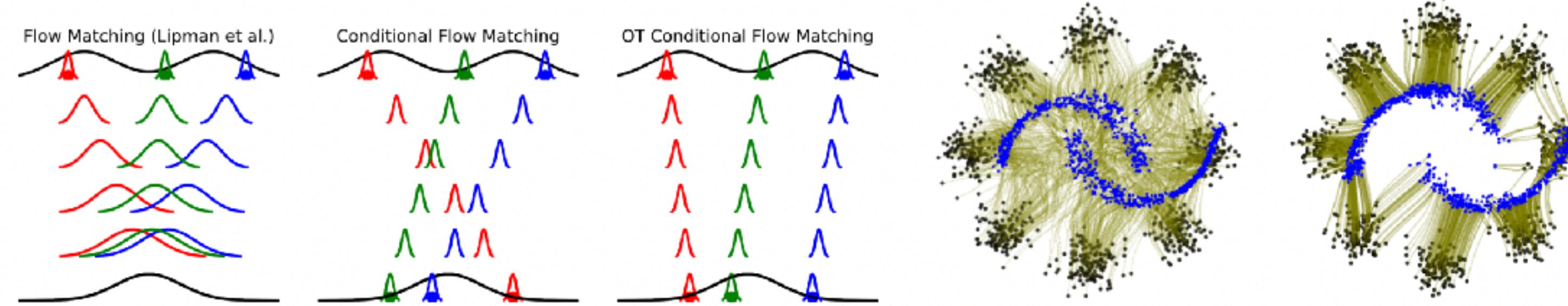


Figure 1: **Left:** Conditional flows from FM (Lipman et al., 2023), I-CFM (§3.2.2), and OT-CFM (§3.2.3). **Right:** Learned flows (green) from moons (blue) to 8gaussians (black) using I-CFM (centre-right) and OT-CFM (far right).

Improving and
Generalizing Flow-
Based Generative
Models with Minibatch
Optimal Transport

1. Randomly sample data points from your **prior**.
2. Sample data points from your **training data**.
3. Find a **1:1 matching** between the two empirical distributions.
4. Learn a time-varying vector field that transforms the **prior** into the **data** distribution using linear interpolation.

Conditional Flow Matching

Vector field
can be ill-
defined

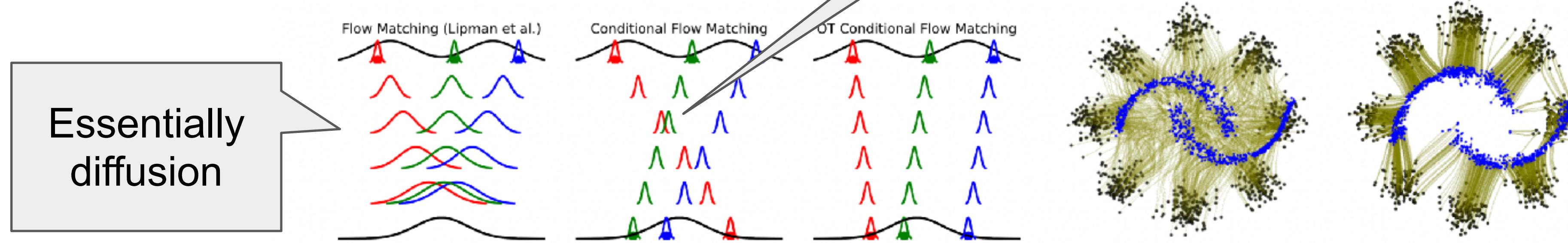


Figure 1: **Left:** Conditional flows from FM (Lipman et al., 2023), I-CFM (§3.2.2), and OT-CFM (§3.2.3). **Right:** Learned flows (green) from moons (blue) to 8gaussians (black) using I-CFM (centre-right) and OT-CFM (far right).

Improving and
Generalizing Flow-
Based Generative
Models with Minibatch
Optimal Transport

1. Randomly sample data points from your **prior**.
2. Sample data points from your **training data**.
3. Find a **1:1 matching** between the two empirical distributions.
4. Learn a time-varying vector field that transforms the **prior** into the **data** distribution using linear interpolation.

Conditional Flow Matching

README MIT license

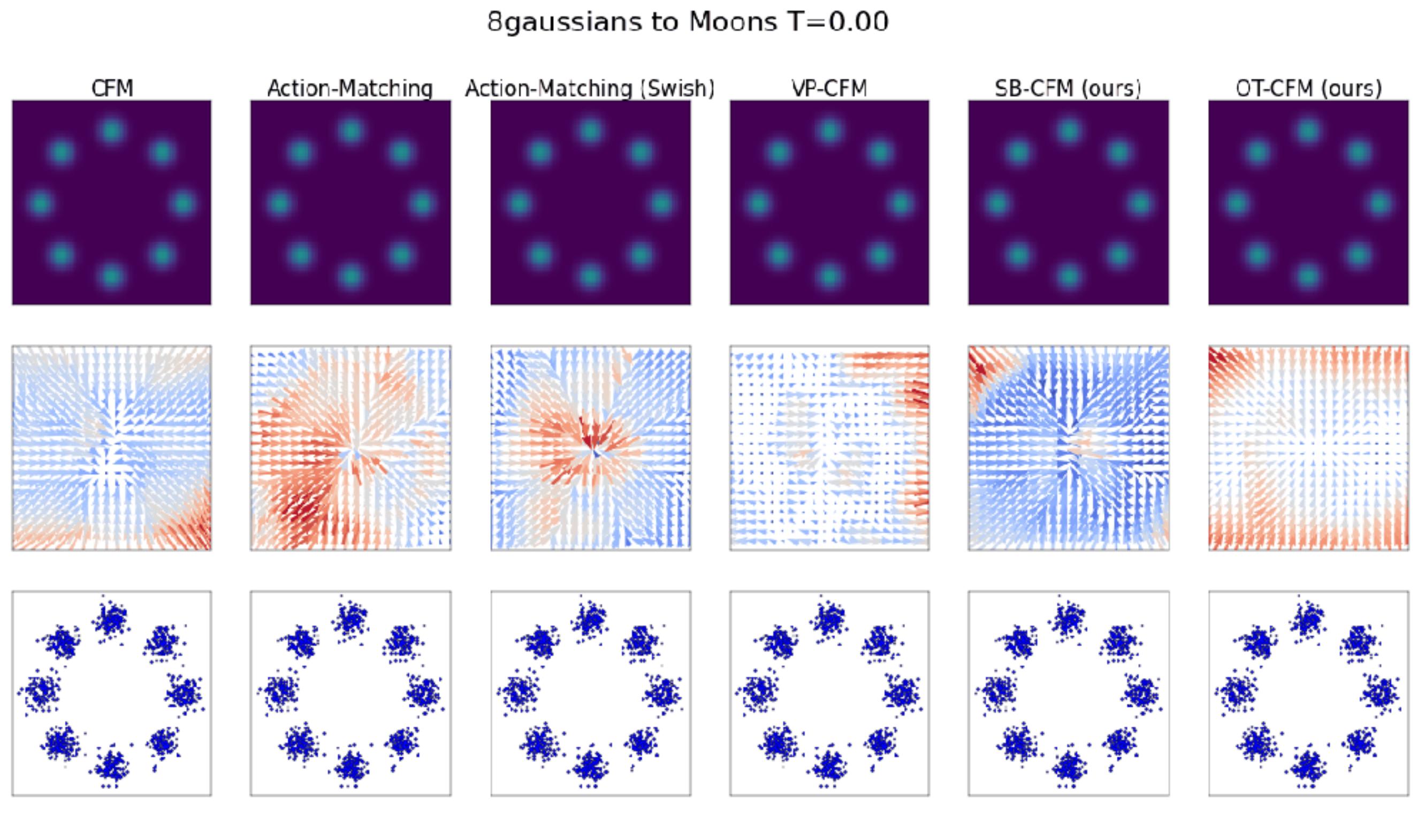
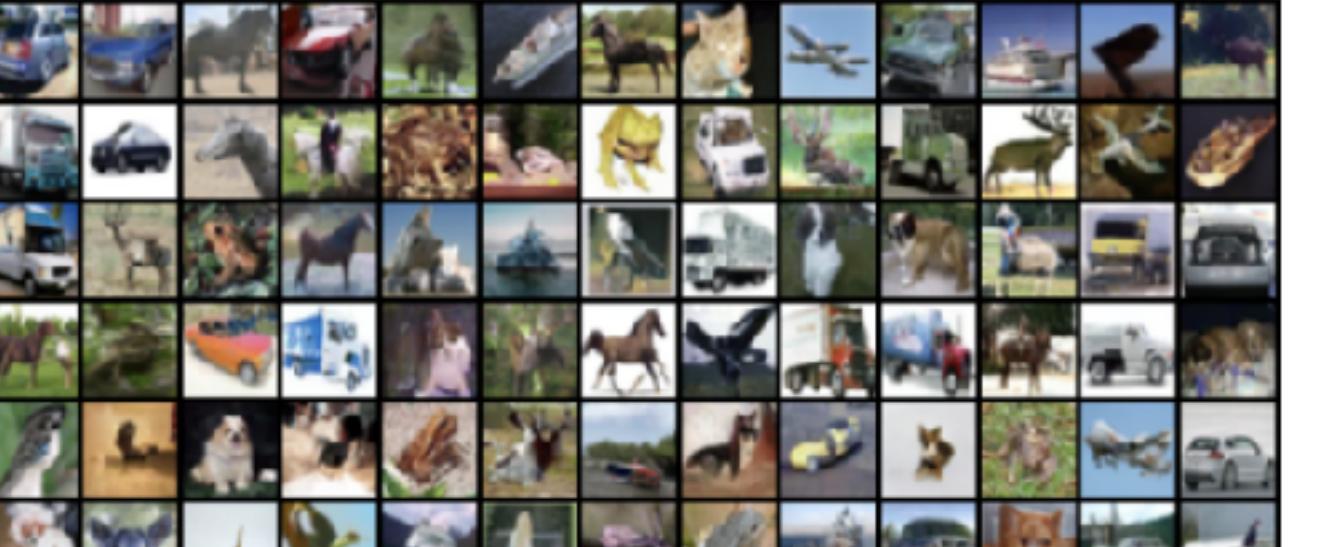
TorchCFM: a Conditional Flow Matching library

[paper arxiv.2302.00482](#) [paper arxiv.2307.03672](#) [PyTorch 1.8+](#) [Lightning 1.6+](#) [Config Hydra 1.2](#) [Code Style Black](#)
[Pre-commit enabled](#) [TorchCFM Tests passing](#) [codecov 36%](#) [Code Quality Main passing](#) [License MIT](#)
[Lightning+Hydra-Template](#)

Description

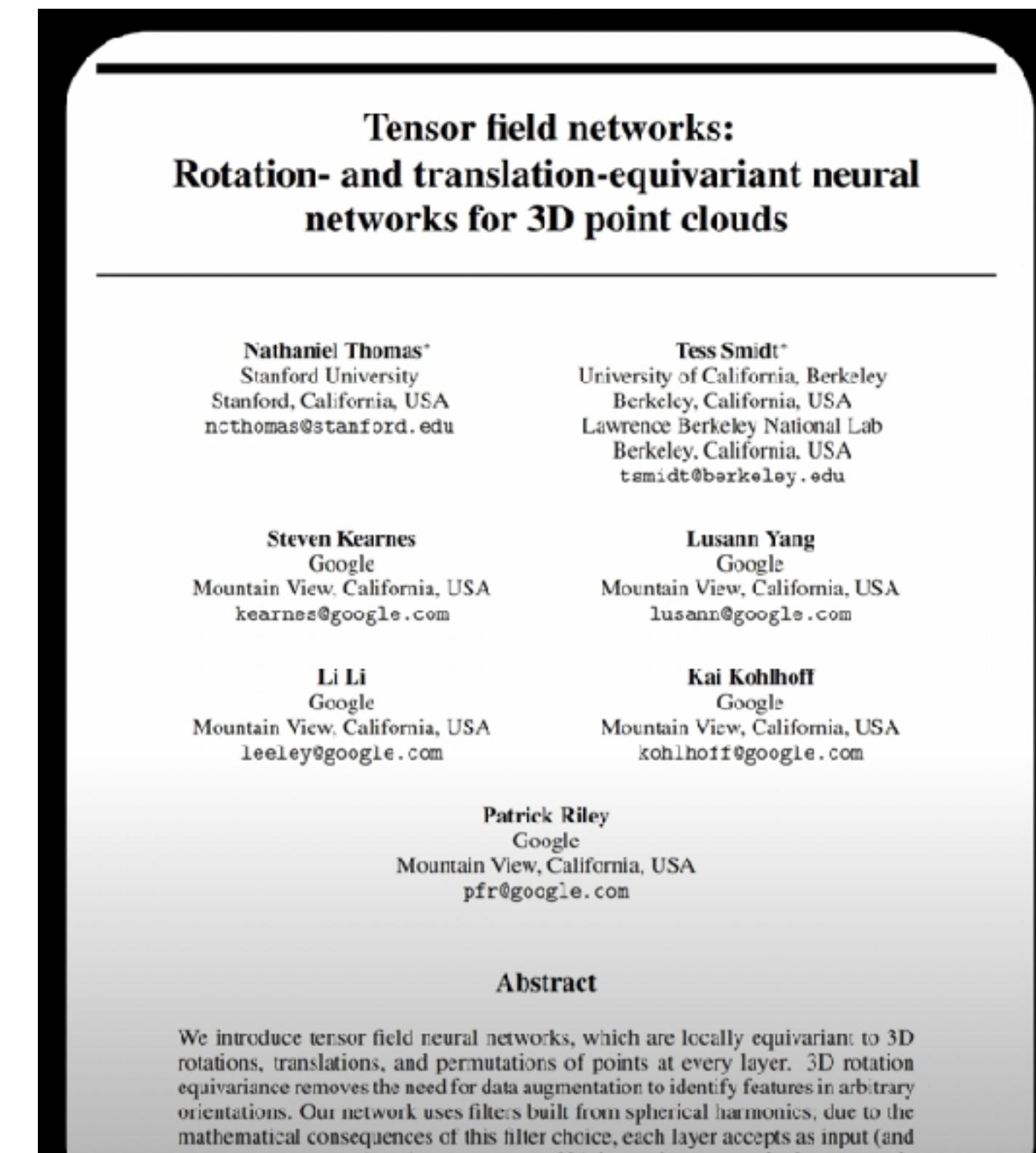
Conditional Flow Matching (CFM) is a fast way to train continuous normalizing flow (CNF) models. CFM is a simulation-free training objective for continuous normalizing flows that allows conditional generative modeling and speeds up training and inference. CFM's performance closes the gap between CNFs and diffusion models. To spread its use within the machine learning community, we have built a library focused on Flow Matching methods: TorchCFM. TorchCFM is a library showing how Flow Matching methods can be trained and used to deal with image generation, single-cell dynamics, tabular data and soon $SO(3)$ data.

Generated images with OT-CFM at iteration 500k (FID: 3.6)

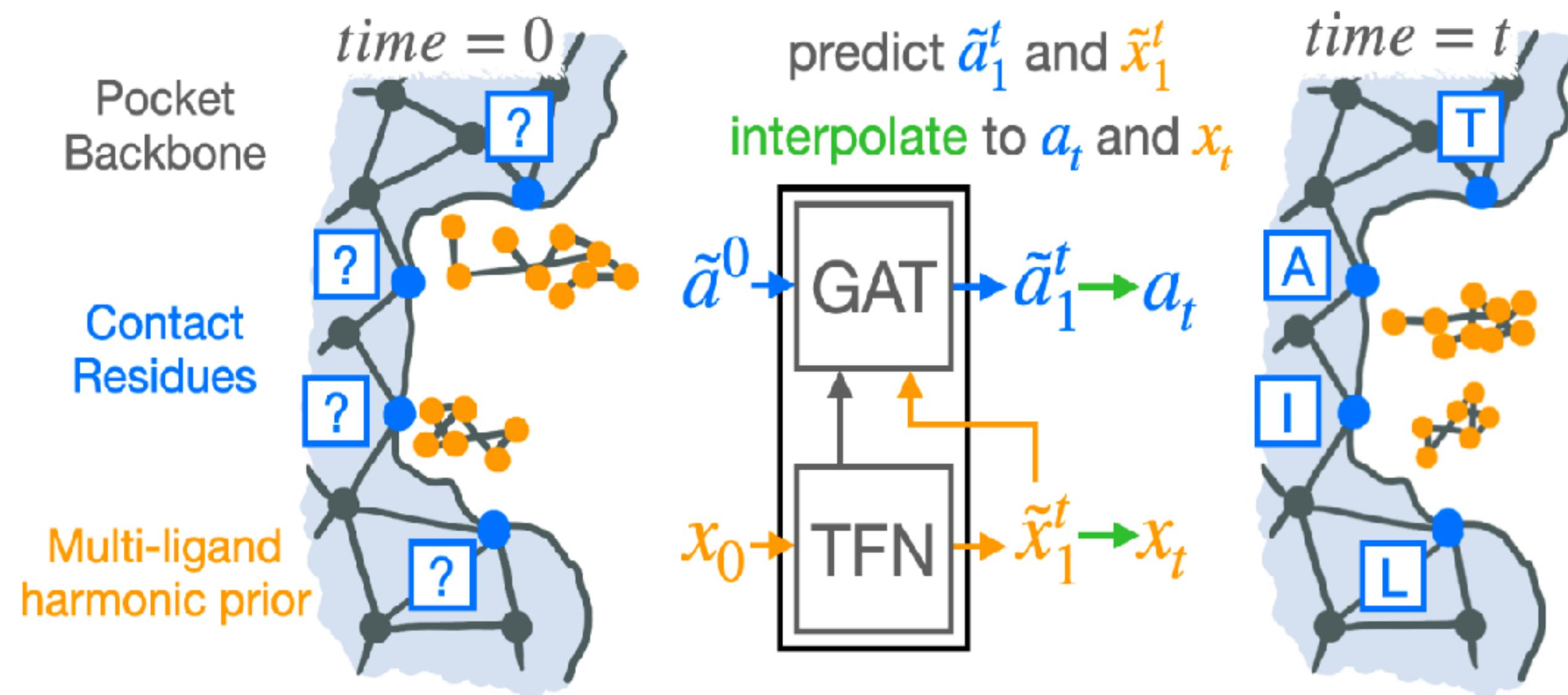


Tensor Field Networks

Backbone: TFNs (special geometric GNNs)



Combine **SchNet** continuous filters
+ basis functions from **spherical harmonics** networks



Take Home

- Conditional flow matching is a **robust and simple alternative to diffusion** and standard normalizing flows.
- Self-conditioning trick makes results better.
- **Tensor-field networks** are a great alternative to standard (E)GNNs.
- Matching samples from the data and prior distribution is a promising research direction.