

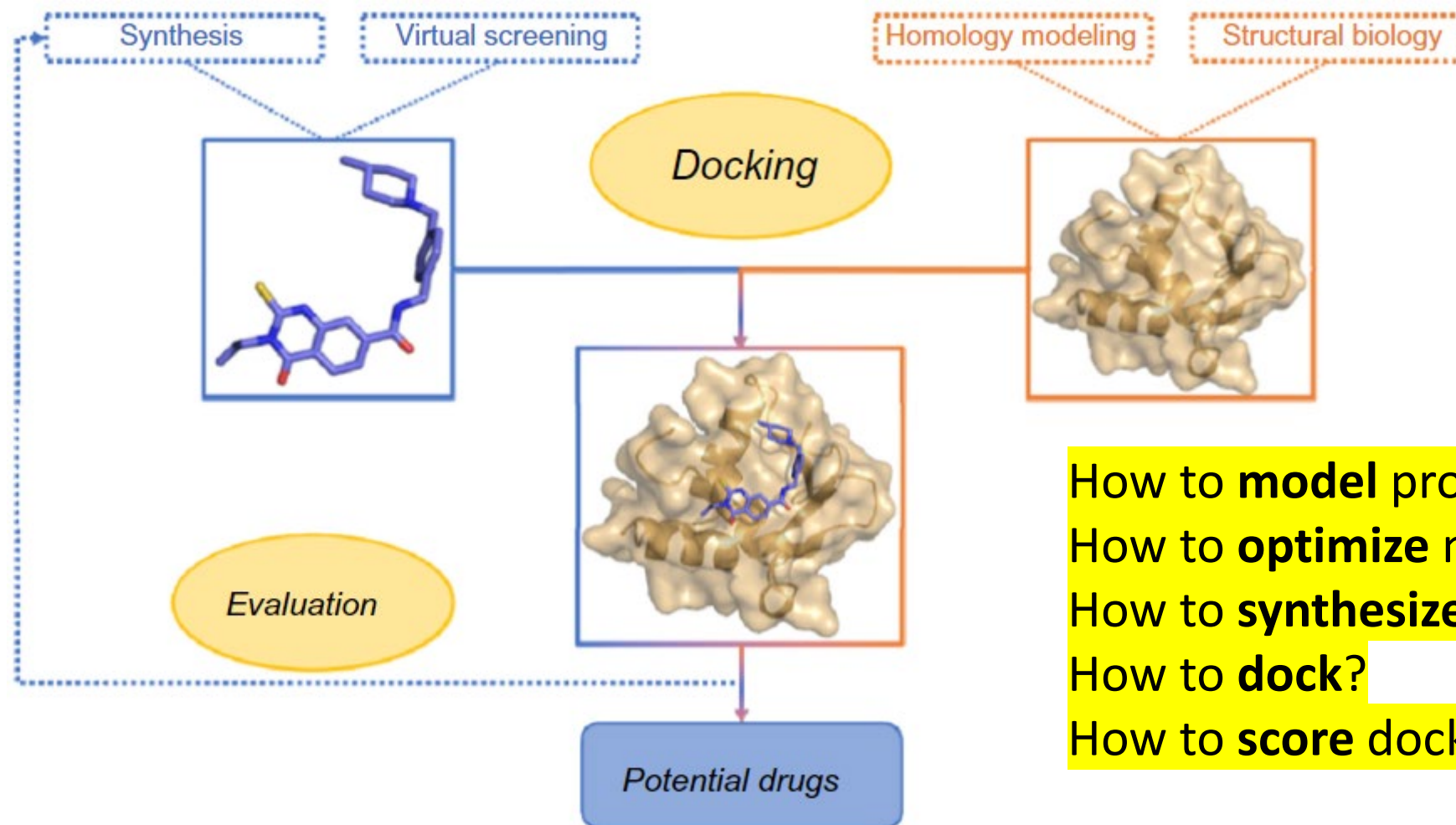
The background of the slide is a photograph of a modern, multi-story glass and concrete building, likely a university or research facility. In the foreground, several flagpoles are visible, flying various flags: a red flag with 'TU/e' and 'EINDHOVEN UNIVERSITY OF' text, the European Union flag, and a red and white checkered flag. The sky is blue with scattered white clouds.

Enabling ML Techniques for SBI: An example of Drug Discovery

Jakub M. Tomczak
Associate Professor
Head of the Generative AI group, TU/e
Founder of Amsterdam AI Solutions

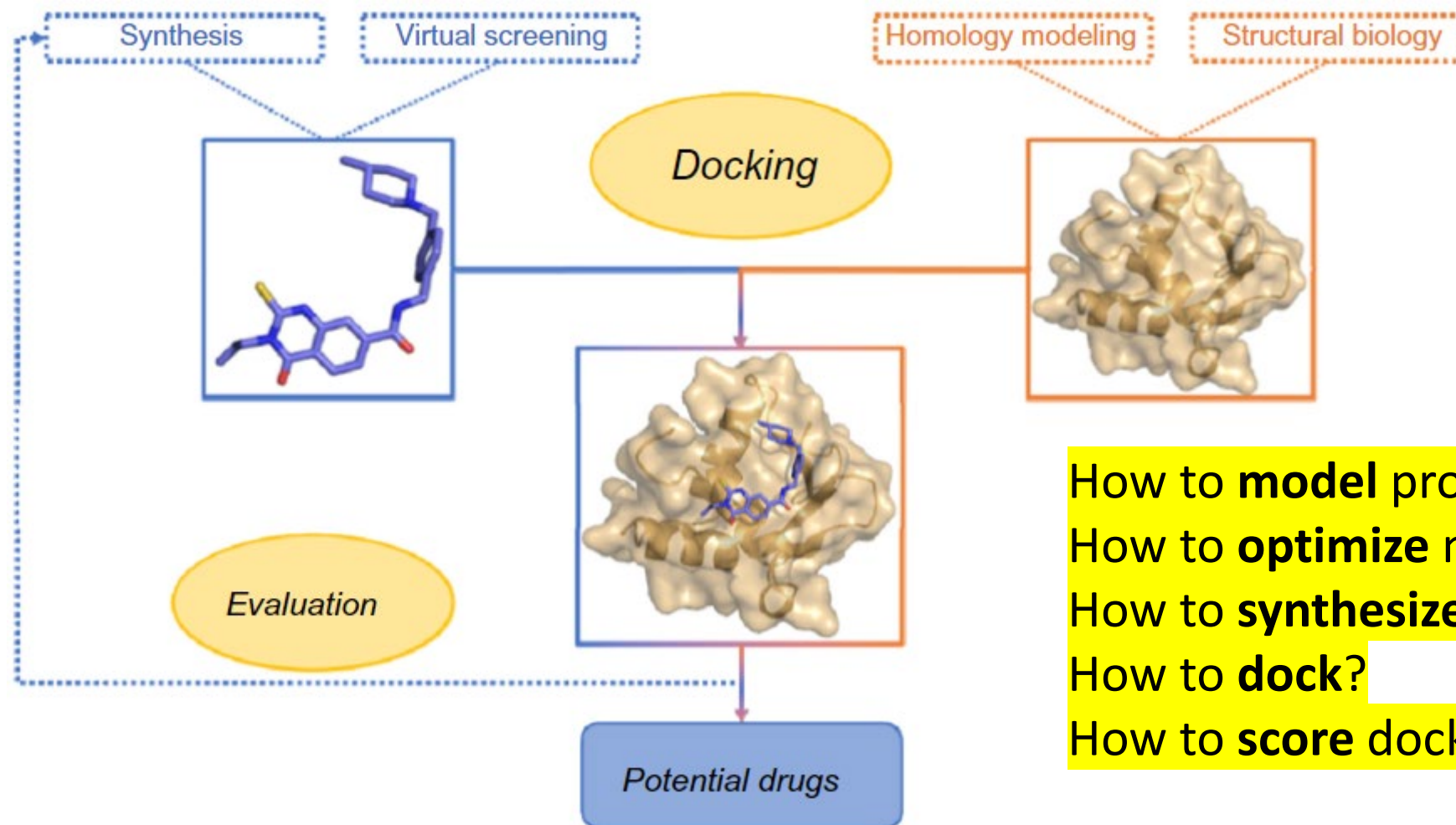
Simulator-based Inference

Why?



How to **model** proteins?
How to **optimize** molecules?
How to **synthesize** molecules?
How to **dock**?
How to **score** docking?

Why?

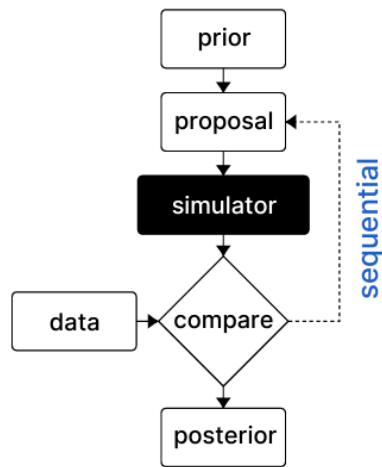


How to **model** proteins?
How to **optimize** molecules?
How to **synthesize** molecules?
How to **dock**?
How to **score** docking?

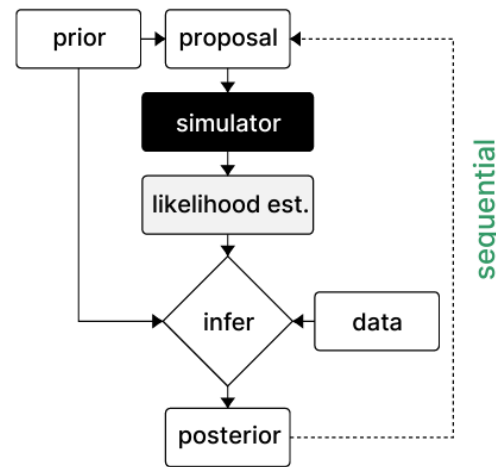
We can use a **simulator**

Various types of SBI

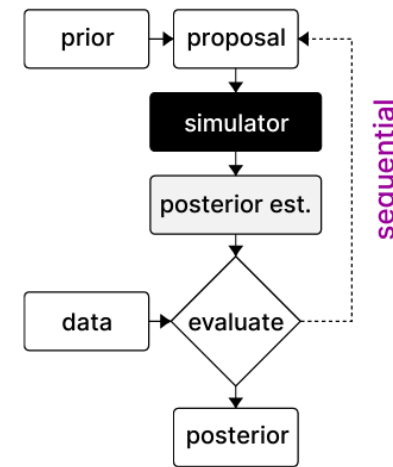
Monte Carlo ABC



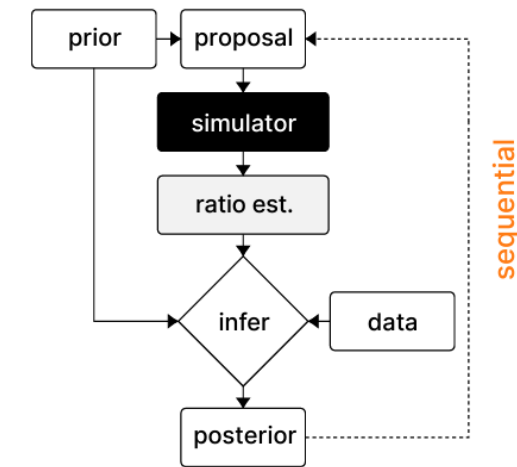
Likelihood Estimation



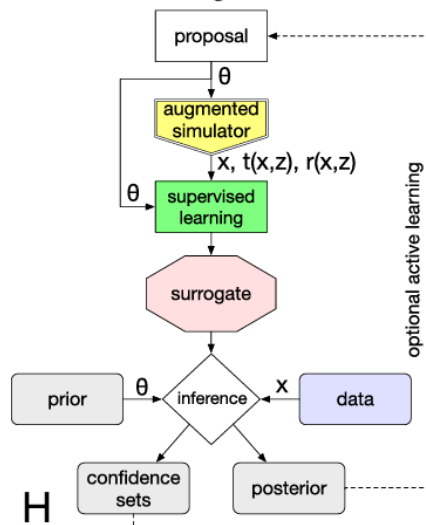
Posterior Estimation



Ratio Estimation



Amortized surrogates trained with augmented data



ML (e.g., Deep Generative Modeling) could be useful.

When do we need simulators? Cannot we use **data** and **ML/DL**? Are they so crucial?

Molecule generation

Problem

Goal: Generate novel molecules (for a given target)

Constraints: Molecules that have certain desirable properties

Search space: $\sim 10^{60}$

Problem

Goal: Generate novel molecules (for a given target)

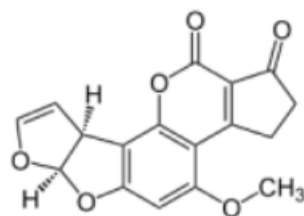
Constraints: Molecules that have certain desirable properties

Search space: $\sim 10^{60}$

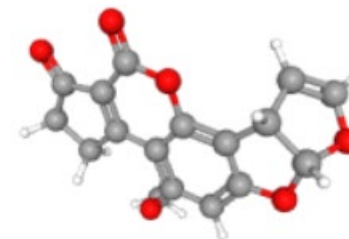
Representation of molecules:

COC1=C2C3=C(C(=O)CC3)C(=O)OC2=C4C5C=COC5OC4=C1

SMILES



Molecular graph



Molecular graph

+

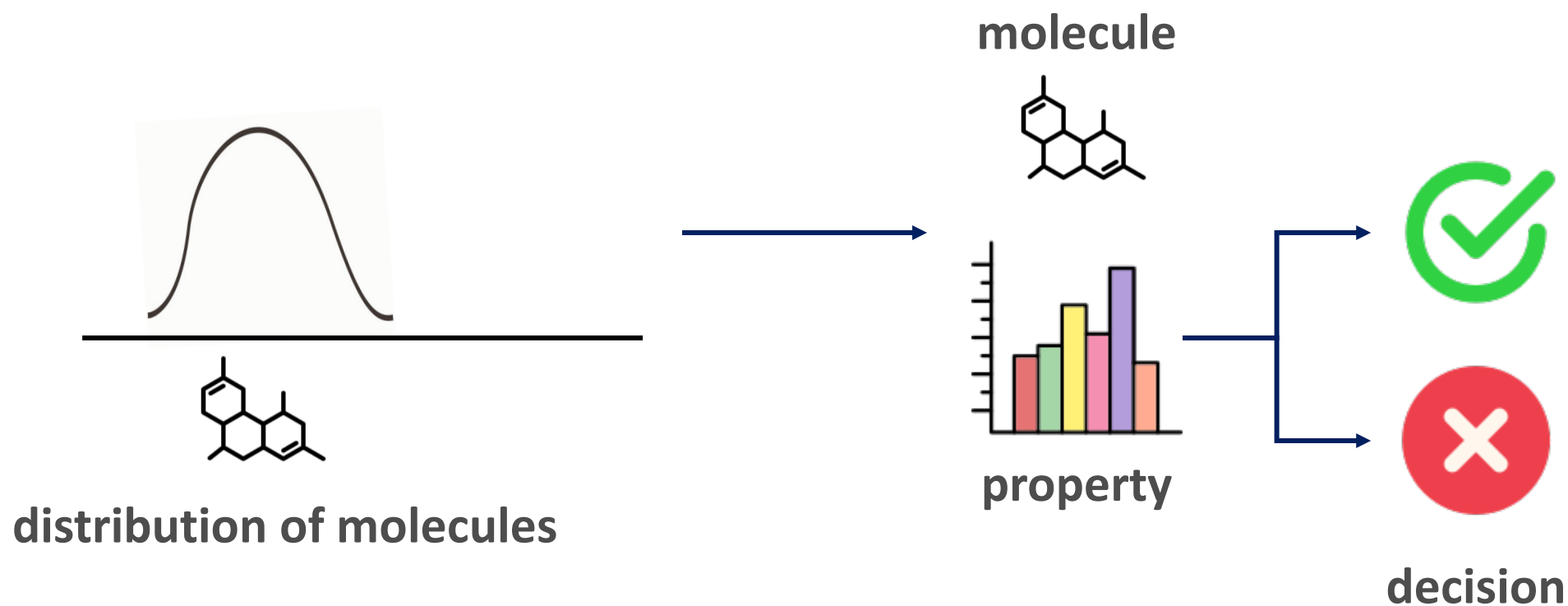
3D positions

Problem

Goal: Generate novel molecules (for a given target)

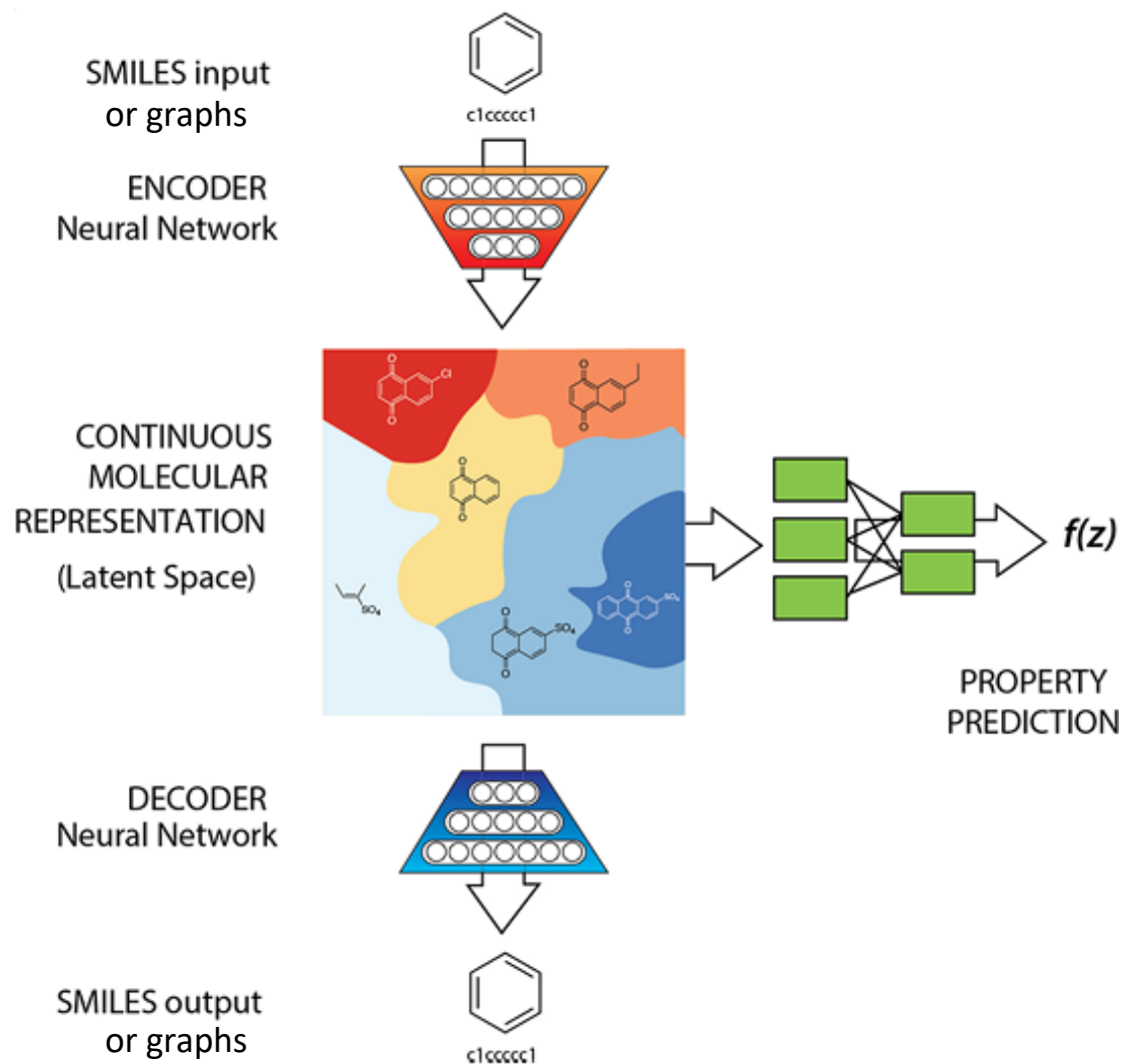
Constraints: Molecules that have certain desirable properties

Search space: $\sim 10^{60}$



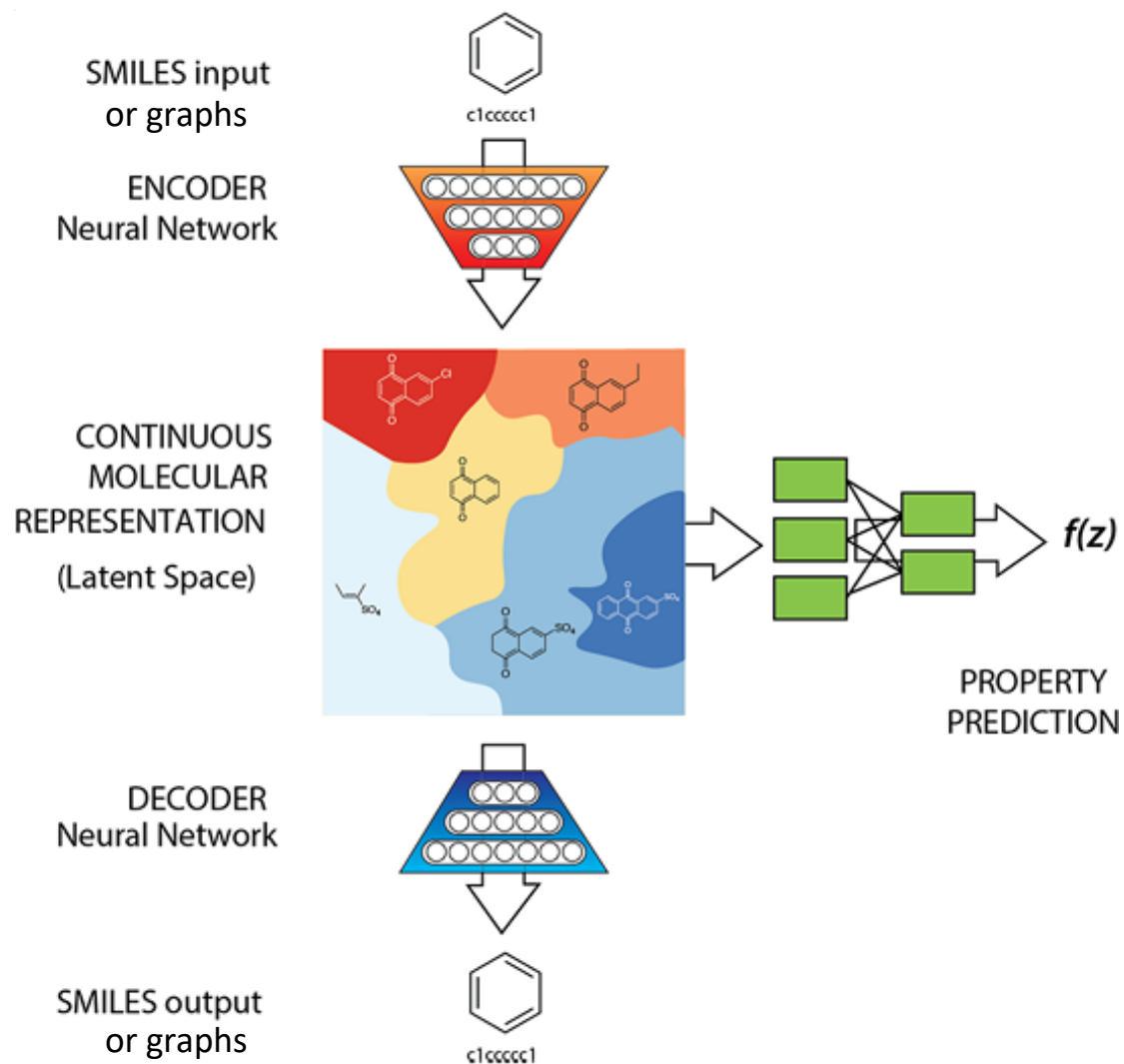
ML-based approaches

Molecule Generation with Joint VAEs



$$\ln p(\mathbf{x}, y) = \ln p(y|\mathbf{x}) + \ln p(\mathbf{x})$$

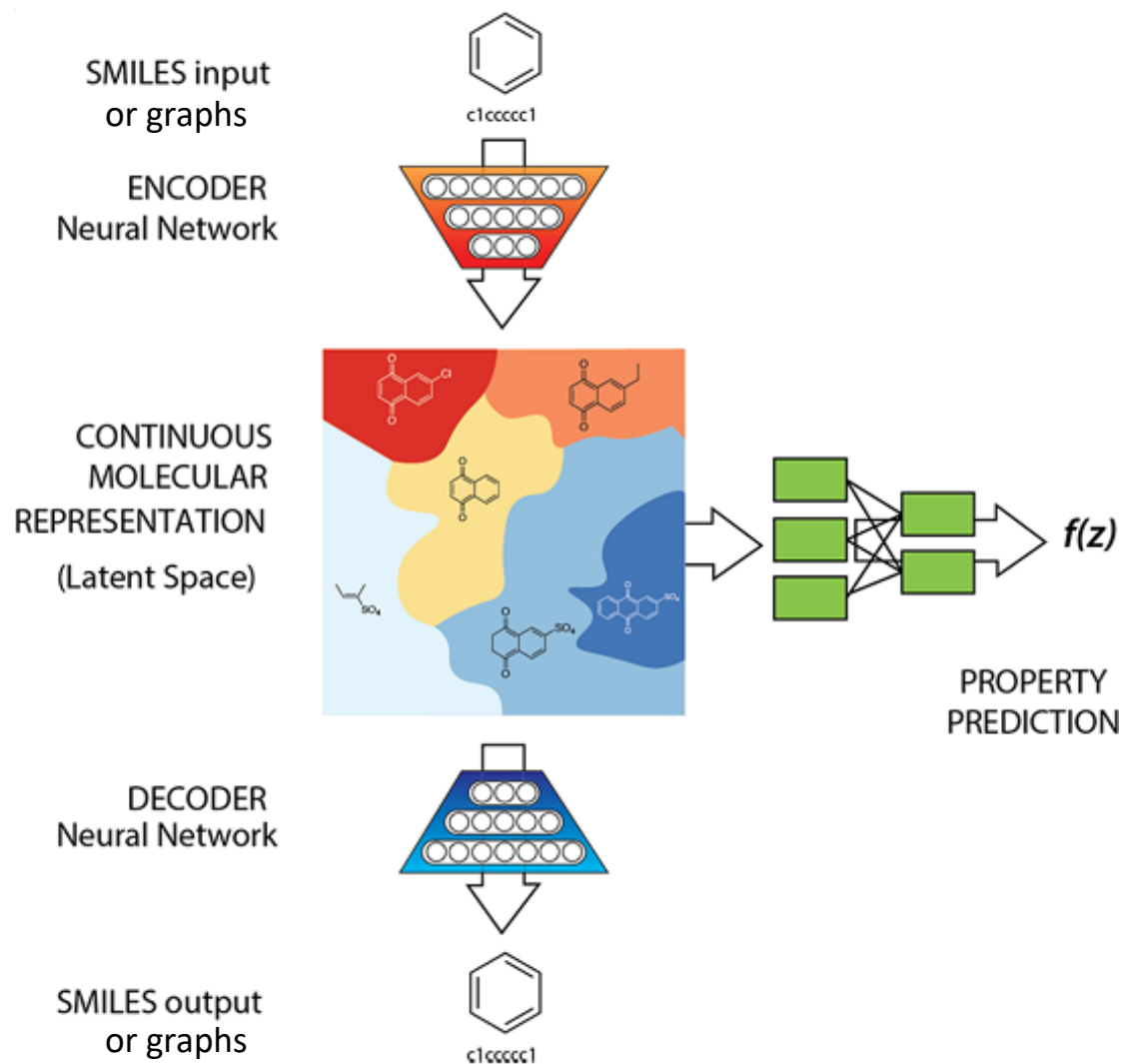
Molecule Generation with Joint VAEs



$$\ln p(\mathbf{x}, y) = \ln p(y|\mathbf{x}) + \ln p(\mathbf{x})$$

(V)AE

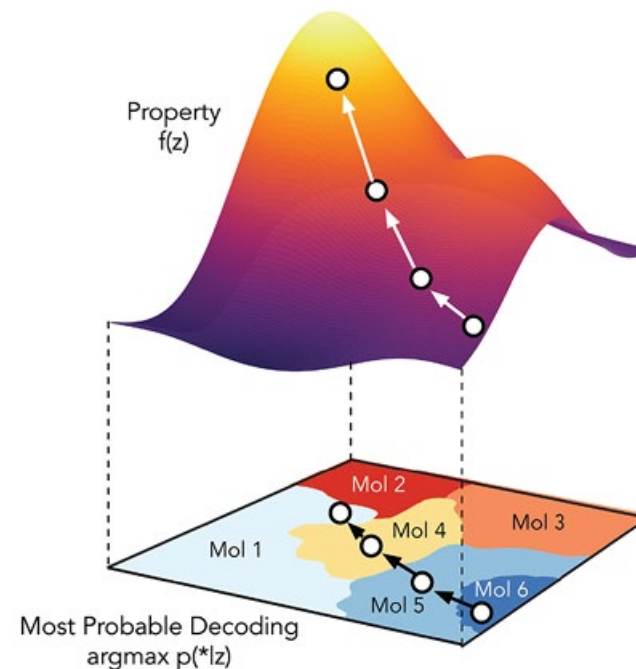
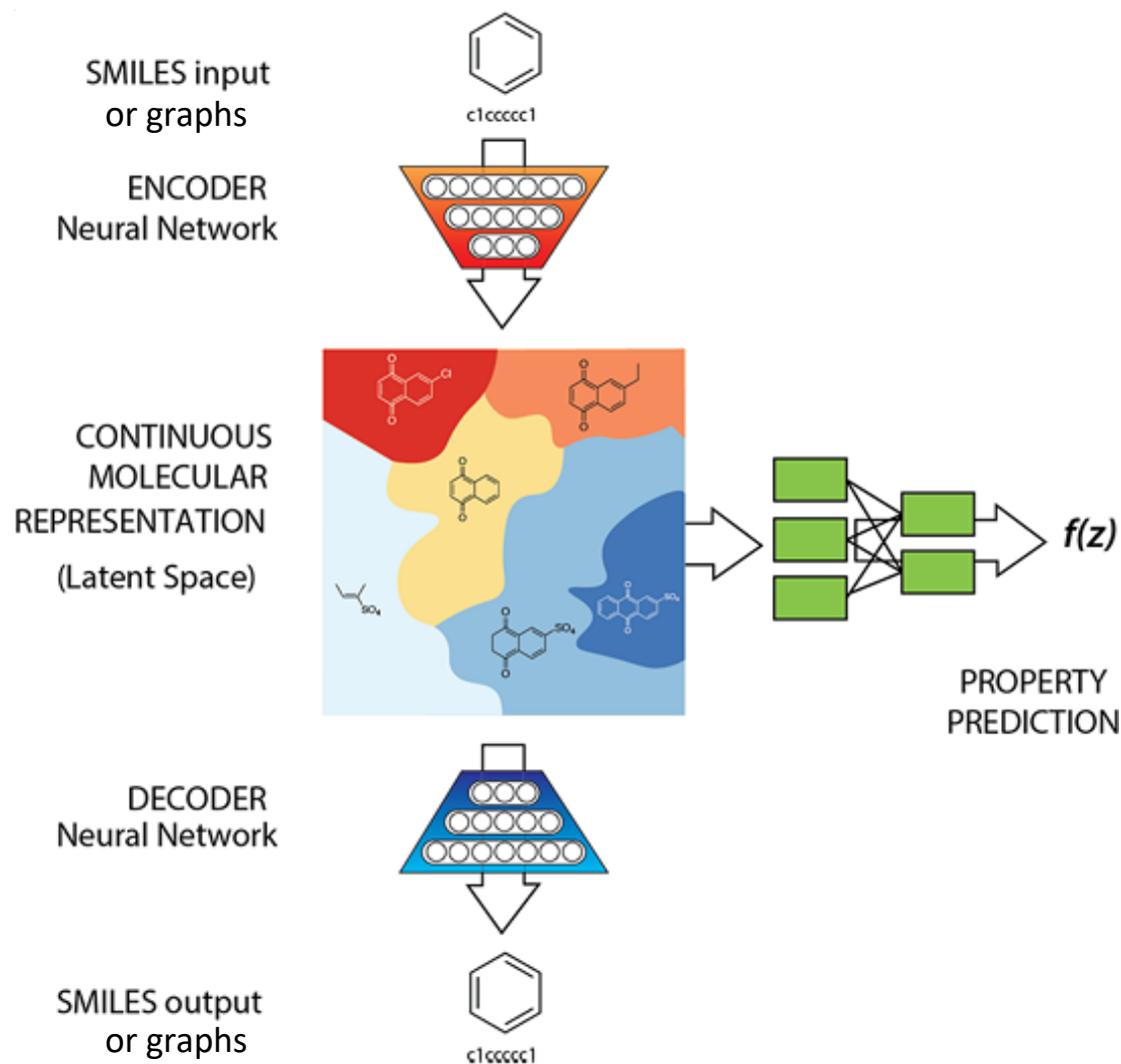
Molecule Generation with Joint VAEs



$$\ln p(\mathbf{x}, y) = \ln p(y|\mathbf{x}) + \ln p(\mathbf{x})$$

encoder
+
predictor

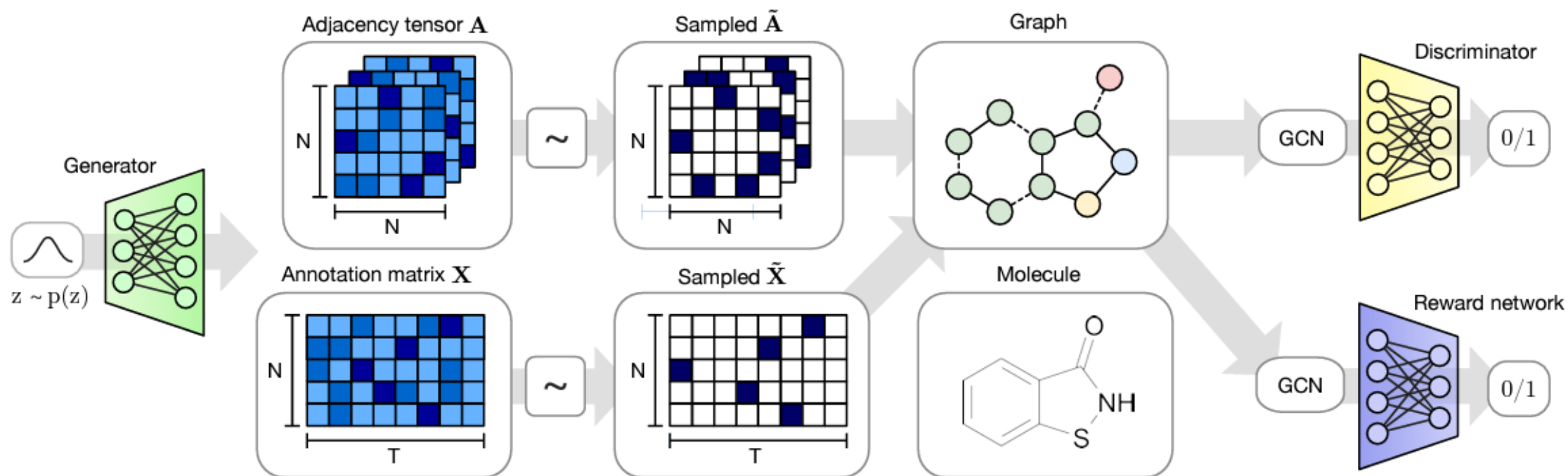
Molecule Generation with Joint VAEs



Optimization through Gradient Descent

$$\ln p(\mathbf{x}, \mathbf{y}) = \ln p(\mathbf{y}|\mathbf{x}) + \ln p(\mathbf{x})$$

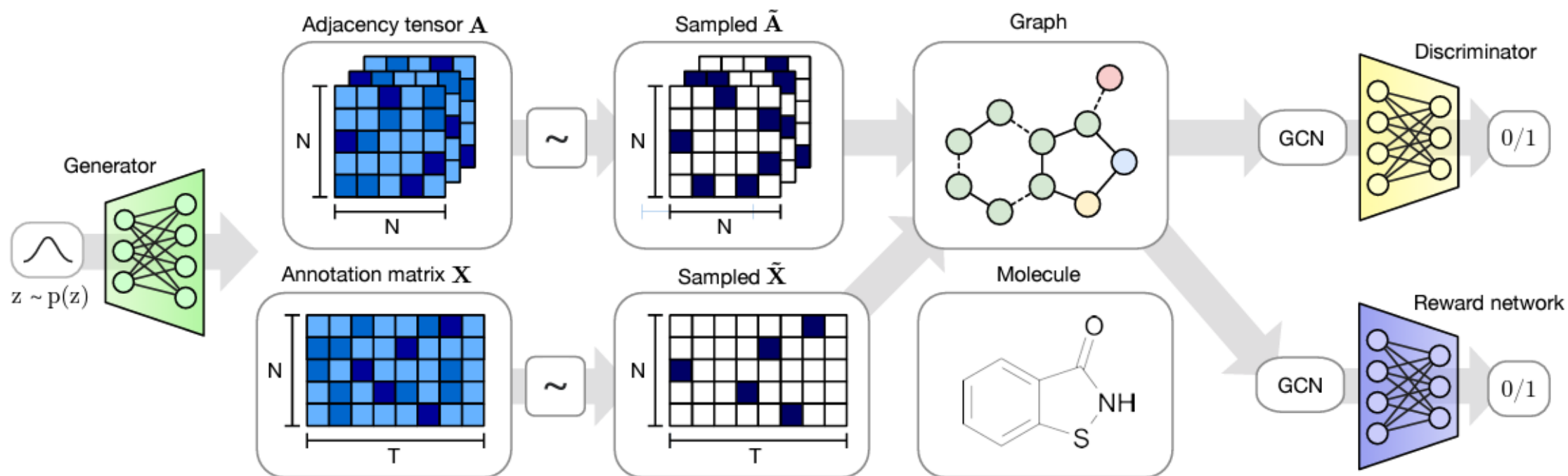
Molecule Generation with GANs



Objective: adversarial loss + RL

$$L(\theta) = \lambda \cdot L_{WGAN}(\theta) + (1 - \lambda) \cdot L_{RL}(\theta)$$

Molecule Generation with GANs

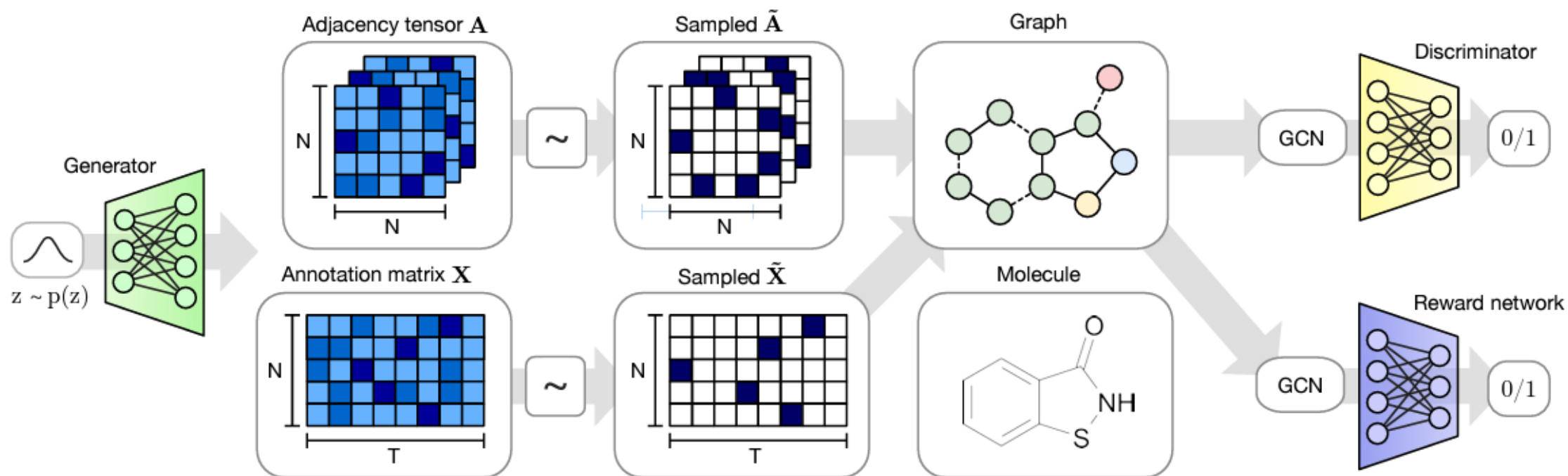


Objective: adversarial loss + RL

$$L(\theta) = \lambda \cdot L_{WGAN}(\theta) + (1 - \lambda) \cdot L_{RL}(\theta)$$

generation

Molecule Generation with GANs



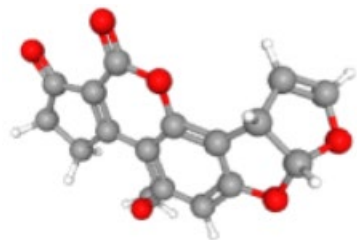
Objective: adversarial loss + RL

$$L(\theta) = \lambda \cdot L_{WGAN}(\theta) + (1 - \lambda) \cdot L_{RL}(\theta)$$

generation

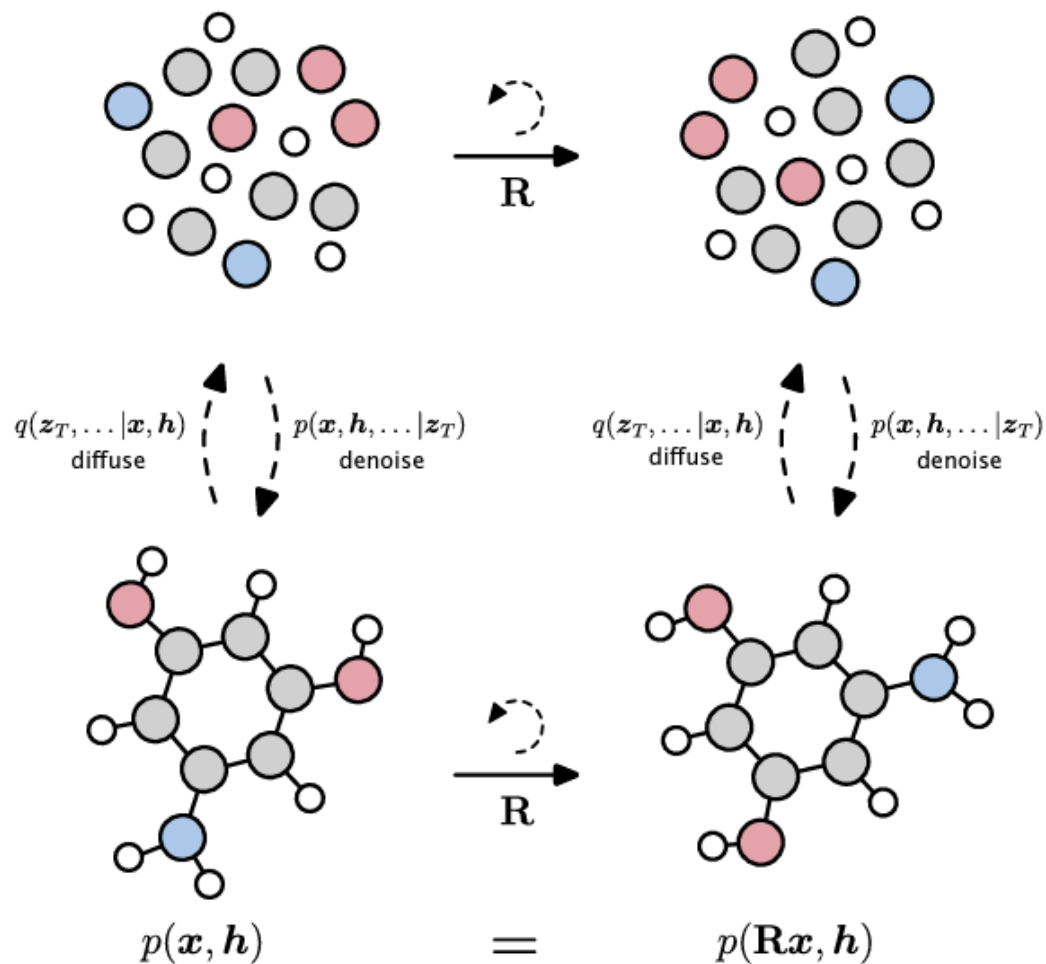
properties

Molecule Generation with Diffusion Models

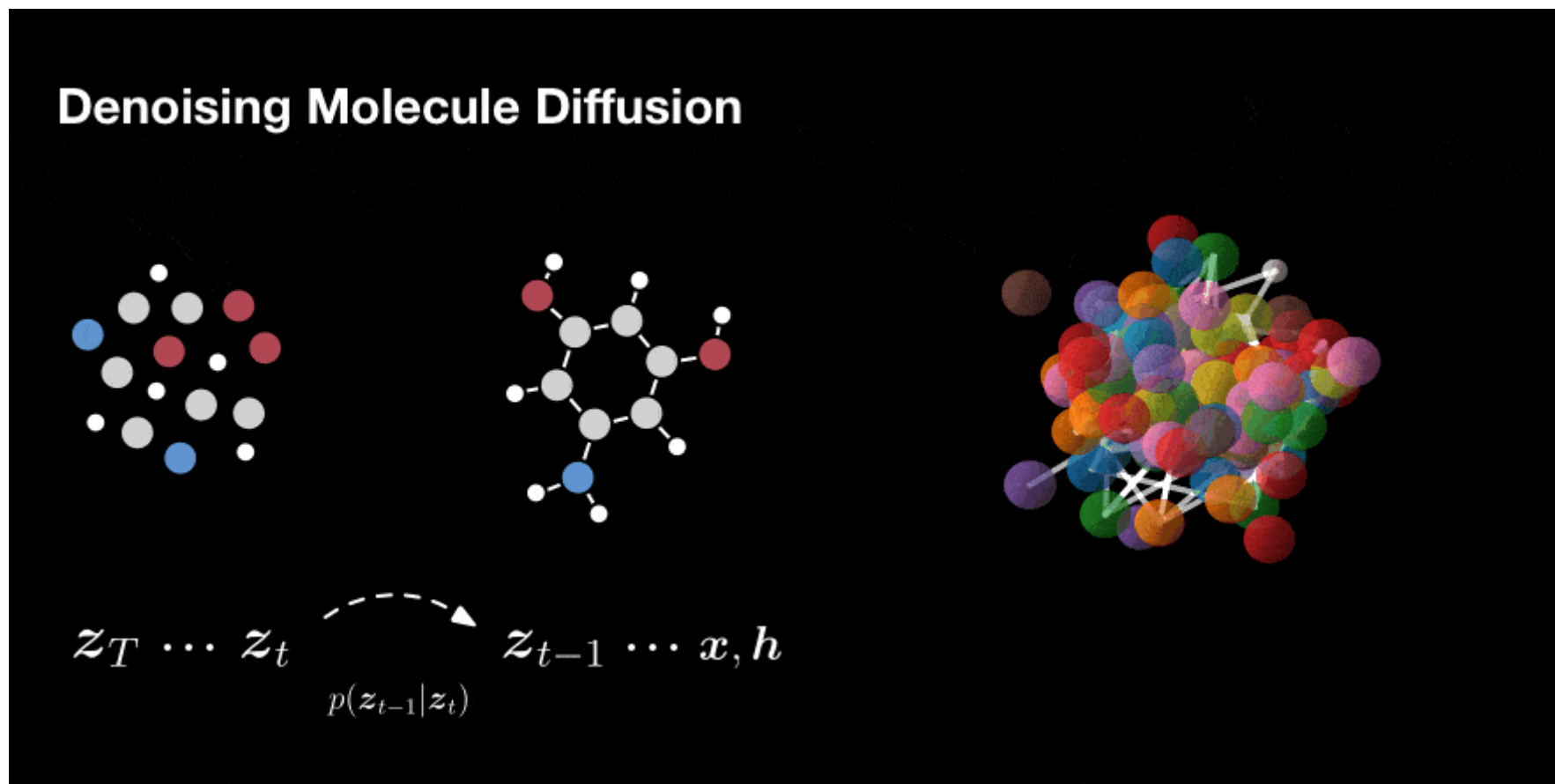


Molecular graph
+
3D positions

Equivariance is important



Molecule Generation with Diffusion Models



Summary

Model	Representation	Objective	Constraints	EVIDence Tractability
VAEs	SMILES Graphs	ELBO	Property predictor	✘
GANs	Graphs	Adversarial loss	RL loss	✘
Diffusion models	Graphs + 3D	ELBO	Property predictor	✘

Jointformer: A shared model for generating and predicting

Molecule generation with joint models

We want to **generate molecules** with specific **properties!**

A possible solution: training a joint model

$$\ln p(\mathbf{x}, y) = \ln p(y|\mathbf{x}) + \ln p(\mathbf{x})$$

How to do that?

Molecule generation with joint models

We want to **generate molecules** with specific **properties!**

A possible solution: training a joint model

$$\ln p(\mathbf{x}, y) = \ln p(y|\mathbf{x}) + \ln p(\mathbf{x})$$

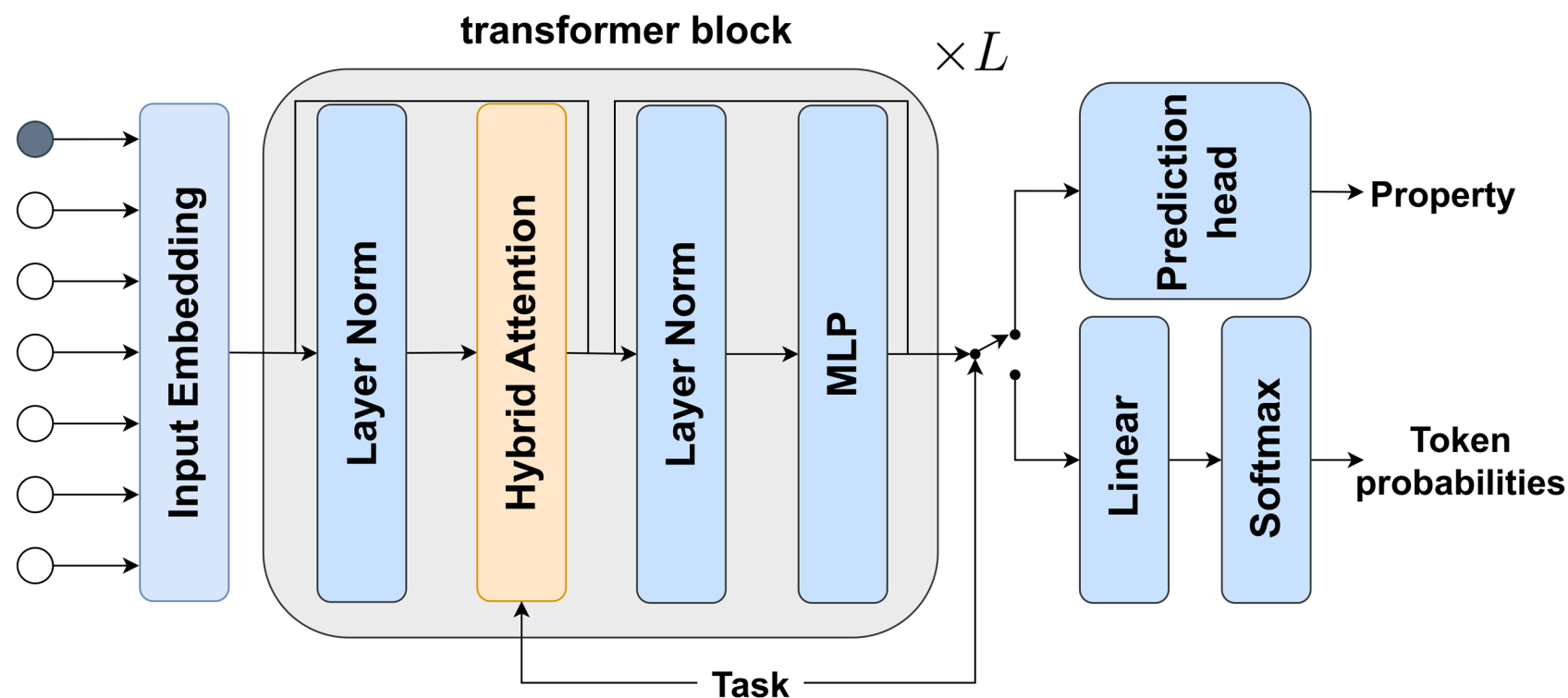
How to do that?

We need to ensure that we can generate molecules AND predict properties!

Ideally, we would like to have a single model that has it all!

Jointformer for molecules

We propose to use a single, shared Jointformer:



$$\begin{aligned}
 p_{\theta}(\mathbf{x}) &:= f_{\theta, \psi}(\mathbf{x}, \emptyset, \langle \text{GEN} \rangle), \\
 p_{\theta, \psi}(y | \mathbf{x}) &:= f_{\theta, \psi}(\mathbf{x}, \emptyset, \langle \text{PRED} \rangle), \\
 \prod_{m \in M} p_{\theta}(x_m | \mathbf{x}_{-m}) &:= f_{\theta, \psi}(\mathbf{x}, M, \langle \text{REC} \rangle),
 \end{aligned}$$

Parameters are shared between $p(y|\mathbf{x})$ and $p(\mathbf{x})$ and the difference is changing the masking from **causal=True** to **causal=False**.

Training of Transformers

Standard training procedure:

Pre-training: Training with the masked loss over tokens with **masking** ($\mathbf{m} \sim p(\mathbf{m})$):

$$L(\theta) = \sum_n \left(\sum_d \ln p(x_{n,d} | \mathbf{m} \odot \mathbf{x}_{n,-d}; \theta) \right)$$

Fine-tuning: Training a predictor $p(y|\mathbf{x})$ (e.g., properties) or a decoder-transformer (causal=True) $p(\mathbf{x})$ using the likelihood function:

$$L(\theta) = \sum_n \ln p(y_n | \mathbf{x}_n; \theta, \phi) \quad \text{OR} \quad L(\theta) = \sum_n \ln p(\mathbf{x}_n; \theta)$$

Training of Transformers

Standard training procedure:

Pre-training: Training with the masked loss over tokens with **masking** ($\mathbf{m} \sim p(\mathbf{m})$):

$$L(\theta) = \sum_n \left(\sum_d \ln p(x_{n,d} | \mathbf{m} \odot \mathbf{x}_{n,-d}; \theta) \right)$$

Fine-tuning: Training a predictor $p(y|\mathbf{x})$ (e.g., properties) or a decoder-transformer (causal=True) $p(\mathbf{x})$ using the likelihood function:

$$L(\theta) = \sum_n \ln p(y_n | \mathbf{x}_n; \theta, \phi) \quad \text{OR} \quad L(\theta) = \sum_n \ln p(\mathbf{x}_n; \theta)$$

EITHER predictive OR generative

Training of Transformers

Standard training procedure:

Pre-training: Training with the masked loss over tokens with **masking** ($\mathbf{m} \sim p(\mathbf{m})$):

$$L(\theta) = \sum_n \left(\sum_d \ln p(x_{n,d} | \mathbf{m} \odot \mathbf{x}_{n,-d}; \theta) \right)$$

Fine-tuning: Training a predictor $p(y|\mathbf{x})$ (e.g., properties) (causal=True) using the penalized likelihood function with $\ln p(\mathbf{x})$:

$$L(\theta) = \sum_n \ln p(y_n | \mathbf{x}_n; \theta, \phi) + \lambda \sum_n \ln p(\mathbf{x}_n; \theta)$$

Strongly predictive but very poor generative

Training of Transformers

Standard training procedure:

Pre-training: Training $p(\mathbf{x})$ using the masked loss over tokens with **masking** ($\mathbf{m} \sim p(\mathbf{m})$) as a penalty:

$$L(\theta) = \sum_n \left(\sum_d \ln p(x_{n,d} | \mathbf{m} \odot \mathbf{x}_{n,-d}; \theta) + \ln p(\mathbf{x}_n; \theta) \right)$$

Fine-tuning: Training a predictor $p(y|\mathbf{x})$ (e.g., properties) or a decoder-transformer (causal=True) $p(\mathbf{x})$ using the likelihood function:

$$L(\theta) = \sum_n \ln p(y_n | \mathbf{x}_n; \theta, \phi) \quad \text{OR} \quad L(\theta) = \sum_n \ln p(\mathbf{x}_n; \theta)$$

Training of Transformers

Standard training procedure:

Pre-training: Training $p(\mathbf{x})$ using the masked loss over tokens with **masking** ($\mathbf{m} \sim p(\mathbf{m})$) as a penalty:

$$L(\theta) = \sum_n \left(\sum_d \ln p(x_{n,d} | \mathbf{m} \odot \mathbf{x}_{n,-d}; \theta) + \ln p(\mathbf{x}_n; \theta) \right)$$

Fine-tuning: Training a predictor $p(y|\mathbf{x})$ (e.g., properties) or a decoder-transformer (causal=True) $p(\mathbf{x})$ using the likelihood function:

$$L(\theta) = \sum_n \ln p(y_n | \mathbf{x}_n; \theta, \phi) \quad \text{OR} \quad L(\theta) = \sum_n \ln p(\mathbf{x}_n; \theta)$$

EITHER predictive OR generative

Training of Jointformers

We propose the following **modified training procedure**:

Pre-training: Training $p(\mathbf{x})$ using causal=True and the masked over tokens (causal=False) with **masking** ($\mathbf{m} \sim p(\mathbf{m})$):

$$L(\theta) = \sum_n \left(\sum_d \ln p(x_{n,d} | \mathbf{m} \odot \mathbf{x}_{n,-d}; \theta) + \ln p(\mathbf{x}_n; \theta) \right)$$

Fine-tuning: Training a predictor $p(y|\mathbf{x})$ (causal=False) and a decoder-transformer (causal=True) $p(\mathbf{x})$:

$$L(\theta) = \sum_n (\ln p(y_n | \mathbf{x}_n; \theta, \phi) + \ln p(\mathbf{x}_n; \theta))$$

Training of Jointformers

We propose the following **modified training procedure**:

Pre-training: Training $p(\mathbf{x})$ using causal=True and the masked over tokens (causal=False) with **masking** ($\mathbf{m} \sim p(\mathbf{m})$):

$$L(\theta) = \sum_n \left(\sum_d \ln p(x_{n,d} | \mathbf{m} \odot \mathbf{x}_{n,-d}; \theta) + \ln p(\mathbf{x}_n; \theta) \right)$$

Enforcing good representation learning!

Fine-tuning: Training a predictor $p(y|\mathbf{x})$ (causal=False) and a decoder-transformer (causal=True) $p(\mathbf{x})$:

$$L(\theta) = \sum_n (\ln p(y_n | \mathbf{x}_n; \theta, \phi) + \ln p(\mathbf{x}_n; \theta))$$

Ensuring both generative and predictive

Generative & Predictive capabilities of Jointformers

The performance of our Jointformer:

Table 2. Ablation study demonstrating the benefits of the pre-training and training objectives and the hybrid attention on the joint generative and predictive performance of JOINTFORMER. We report the mean and standard deviation across seven GuacaMol and three MoleculeNet tasks. T. - transformer.

Model	Pre-training loss	Attention	Training loss	Guacamol		MoleculeNet
				FCD (\uparrow)	RMSE (\downarrow)	RMSE (\downarrow)
GENERATIVE T.			Generative (Eq. 4)	0.87 ± 0.00	N/A	N/A
PREDICTIVE T.			Predictive (Eq. 5)	0.02 ± 0.06	0.044 ± 0.013	0.720 ± 0.141
JOINT T.	Generative (Eq. 4)	Causal	Joint (Eq. 2)	0.85 ± 0.00	0.059 ± 0.020	0.740 ± 0.172
JOINT T., WEIGHTED			Joint, weighted (Eq. 3)	0.71 ± 0.02	0.044 ± 0.013	0.710 ± 0.167
JOINTFORMER	Reconstructive-generative (Eq. 13)	Hybrid	Joint (Eq. 2)	0.84 ± 0.01	0.039 ± 0.009	0.716 ± 0.182

It is important to add the masked loss to pre-training!

Generative & Predictive capabilities of Jointformers

The performance of our Jointformer:

Table 2. Ablation study demonstrating the benefits of the pre-training and training objectives and the hybrid attention on the joint generative and predictive performance of JOINTFORMER. We report the mean and standard deviation across seven GuacaMol and three MoleculeNet tasks. T. - transformer.

Model	Pre-training loss	Attention	Training loss	Guacamol		MoleculeNet
				FCD (\uparrow)	RMSE (\downarrow)	RMSE (\downarrow)
GENERATIVE T.			Generative (Eq. 4)	0.87 ± 0.00	N/A	N/A
PREDICTIVE T.			Predictive (Eq. 5)	0.02 ± 0.06	0.044 ± 0.013	0.720 ± 0.141
JOINT T.	Generative (Eq. 4)	Causal	Joint (Eq. 2)	0.85 ± 0.00	0.059 ± 0.020	0.740 ± 0.172
JOINT T., WEIGHTED			Joint, weighted (Eq. 3)	0.71 ± 0.02	0.044 ± 0.013	0.710 ± 0.167
JOINTFORMER	Reconstructive-generative (Eq. 13)	Hybrid	Joint (Eq. 2)	0.84 ± 0.01	0.039 ± 0.009	0.716 ± 0.182

It seems possible to train a powerful predictor with joint likelihood!

But “no-free-lunch”: the generative performance drops a bit.

Targeted Virtual Screening Experiment

Various predictors vs. Jointformer as a foundation model:

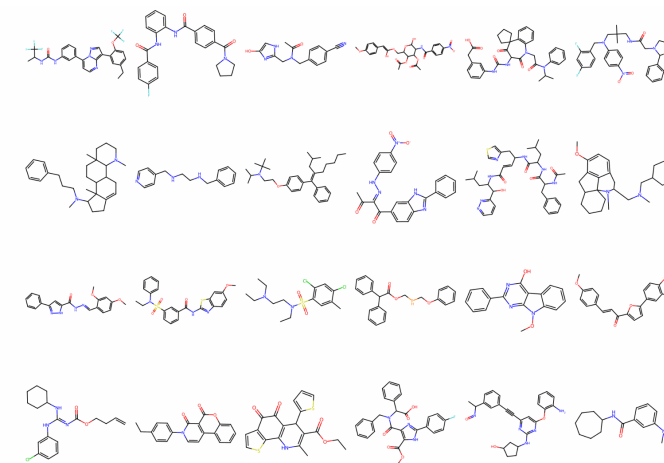
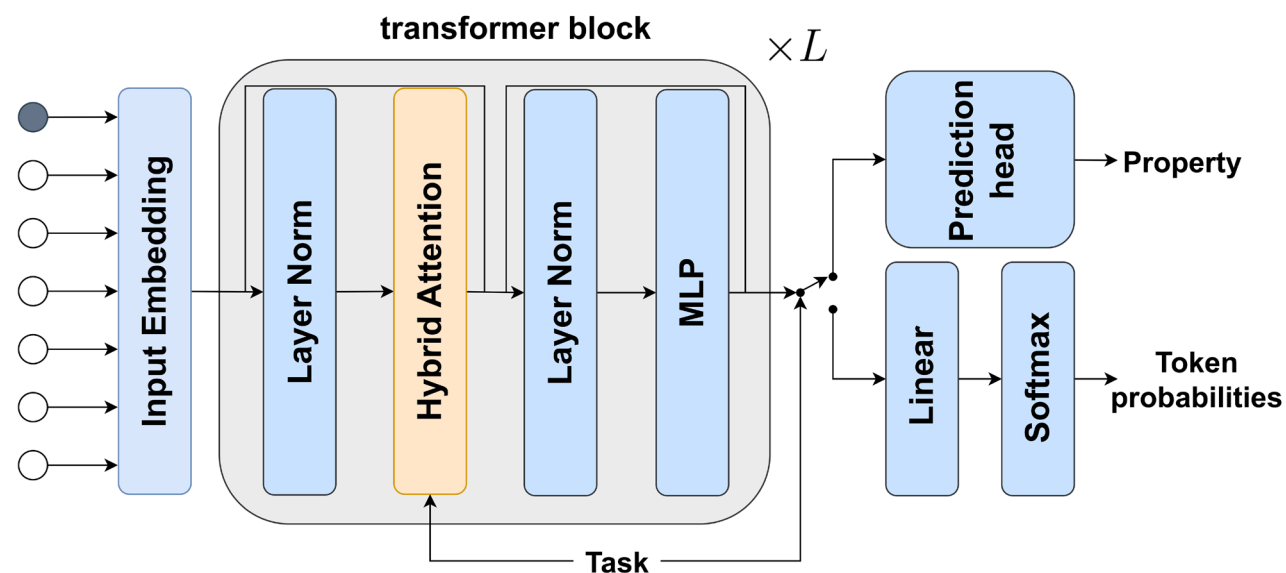
PREDICTOR	CPM	PRECISION	RECALL	F1	ACCURACY
CLASSIFICATION TREE	MOL2VEC	0.60	0.61	0.58	0.58
	JOINTFORMER	0.71	0.61	0.48	0.50
K-NN	MOL2VEC	0.72	0.73	0.69	0.69
	JOINTFORMER	0.66	0.65	0.58	0.58
LOGISTIC REGRESSION	MOL2VEC	0.42	0.49	0.28	0.35
	JOINTFORMER	0.64	0.58	0.46	0.48
RANDOM FOREST	MOL2VEC	0.64	0.65	0.64	0.67
	JOINTFORMER	0.71	0.63	0.50	0.52
SVM	MOL2VEC	0.70	0.60	0.45	0.48
	JOINTFORMER	0.56	0.56	0.52	0.52
MLP	MOL2VEC	0.53	0.53	0.45	0.46
	JOINTFORMER	0.57	0.56	0.47	0.48
JOINTFORMER (OURS)	N/A	0.73	0.75	0.72	0.73

Jointformers

We can learn a joint transformer by **maximizing the joint log-likelihood function**, ...

... but we need a **penalty term** to have a strong predictive performance.

... and we can have a **single model** (i.e., generating + predicting)!

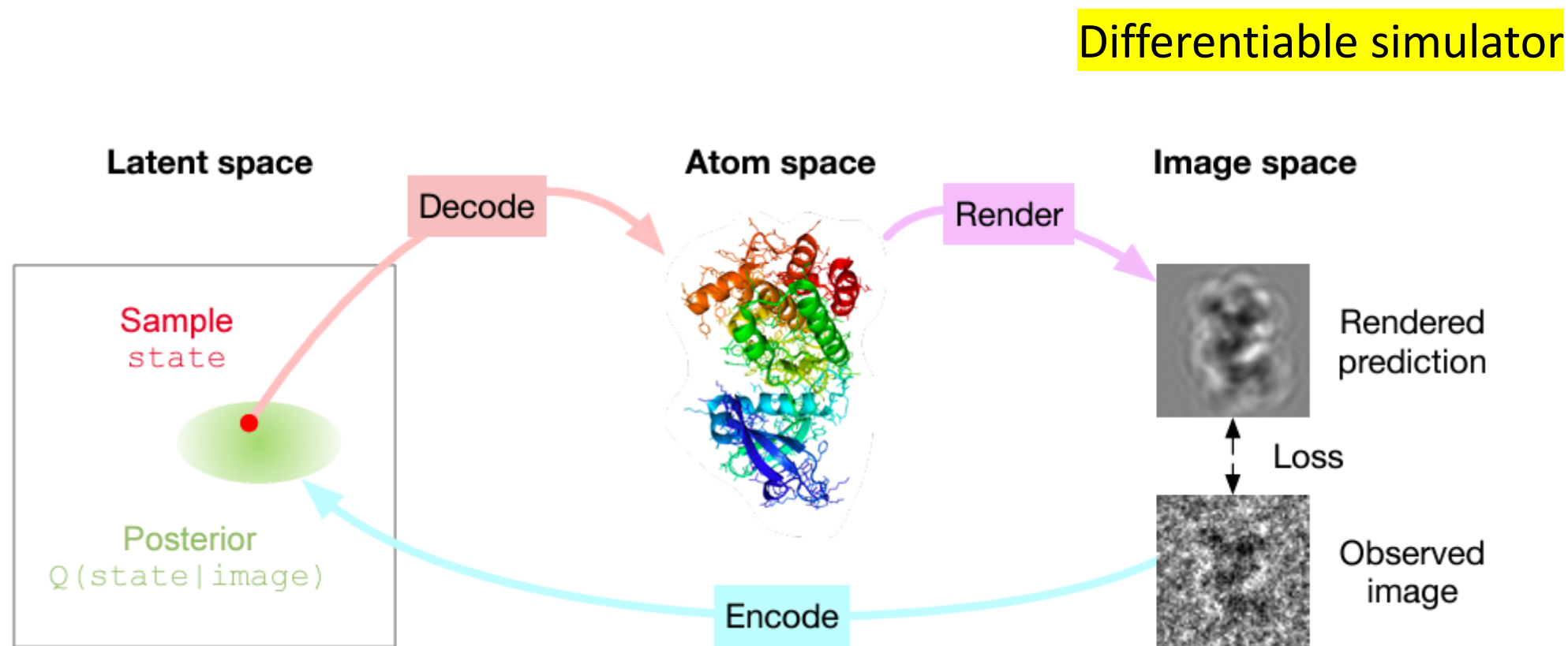


Summary

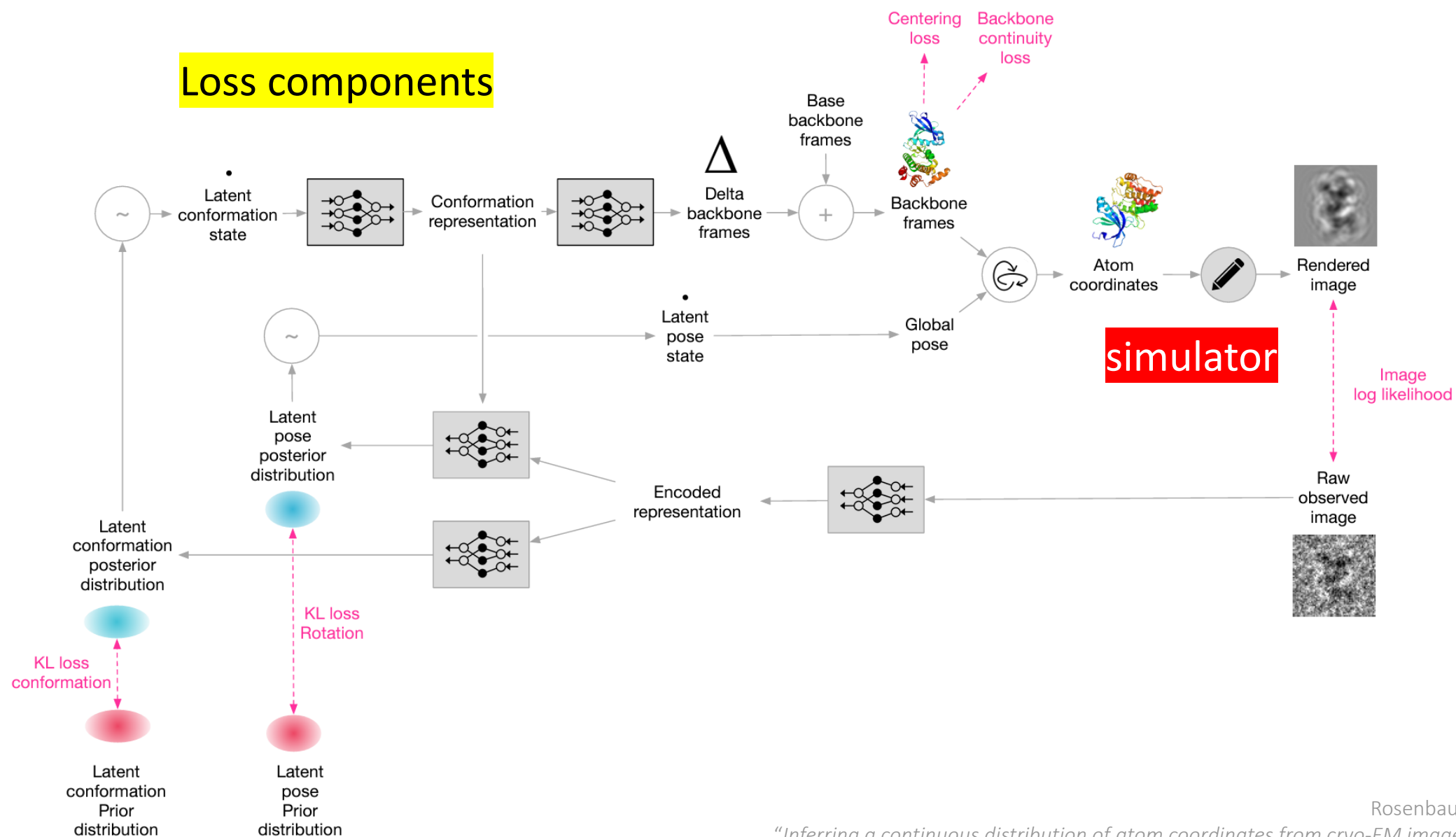
Model	Representation	Objective	Constraints	Evidence Tractability
VAEs	SMILES Graphs	ELBO	Property predictor	✗
GANs	Graphs	Adversarial loss	RL loss	✗
Diffusion models	Graphs + 3D	ELBO	Property predictor	✗
Jointformer	SMILES	Joint Likelihood	-	✓

Training ML models with simulators: Examples

Utilizing differentiable simulator in a VAE framework

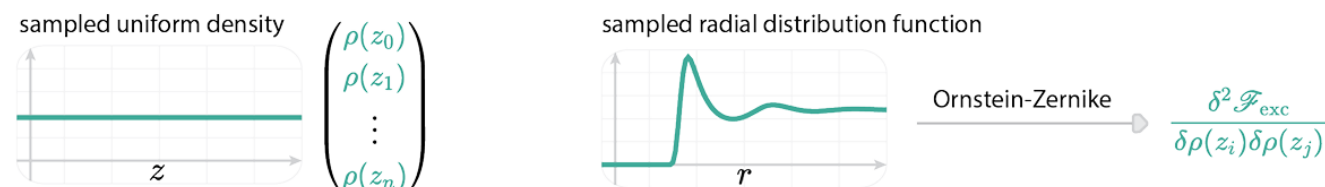


Utilizing differentiable simulator in a VAE framework

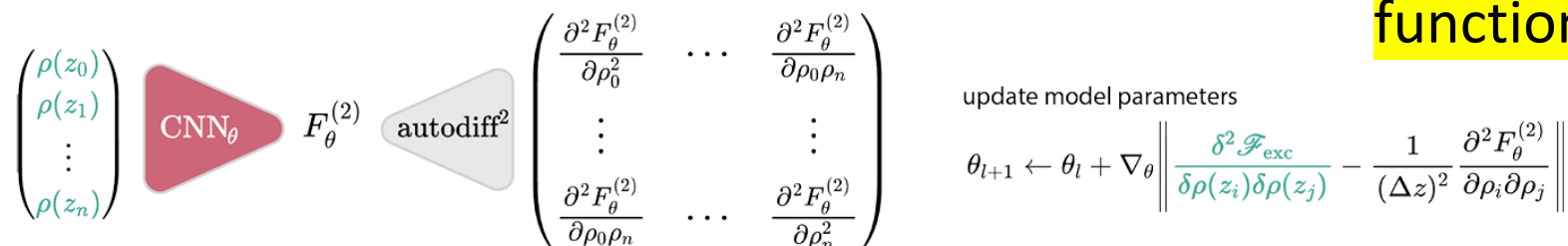


Training neural approximate free energy functional

1. Generate training data from MC simulations



2. Train the neural network with pair-correlation matching



Autodiff from classical density functional theory (cDFT)

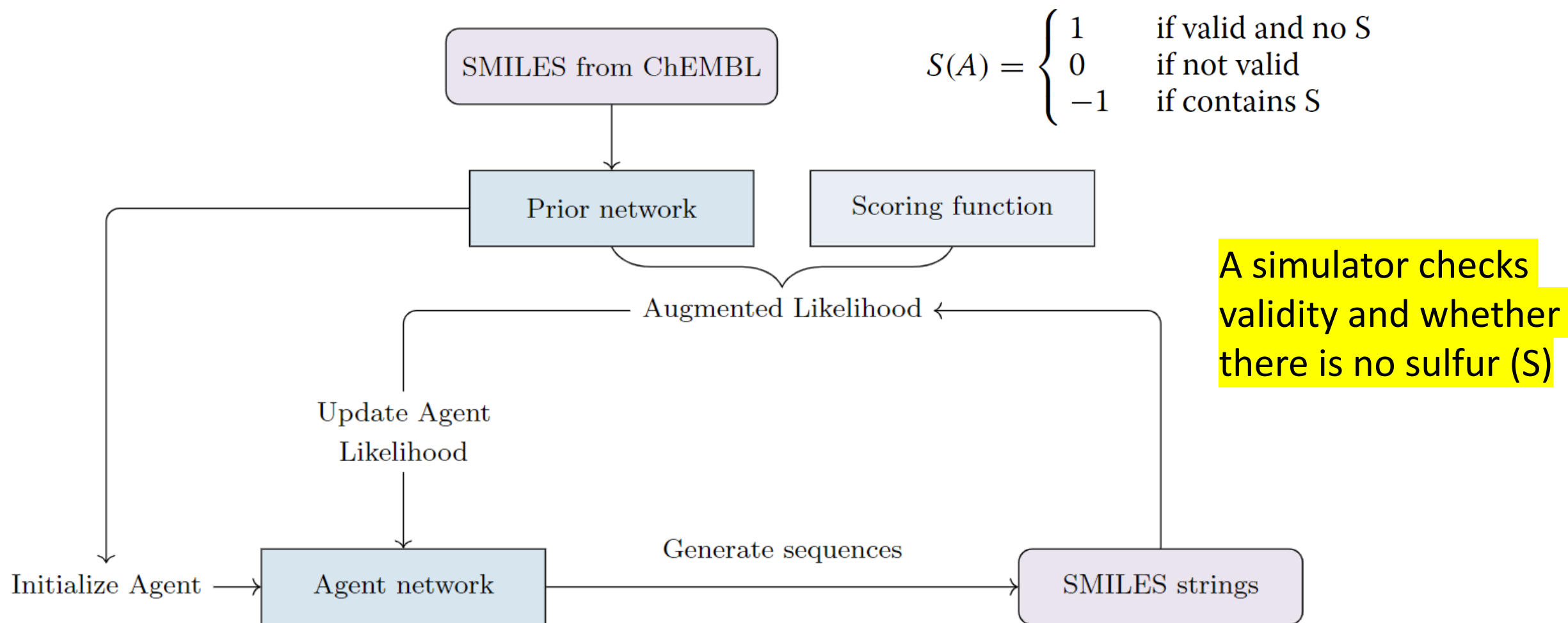
3. Classical DFT at fixed $\beta\mu$ and $\beta V^{\text{ext}}(z)$



FIG. 1: **1.** Bulk densities in planar geometry $\rho(z_i) = \rho_b$ and radial distribution functions $g(r)$ are sampled from Monte Carlo simulations of homogeneous bulk systems of Lennard-Jones particles. Each $g(r)$ is converted to the second functional derivative of the excess free-energy $\delta^2 \mathcal{F}_{\text{exc}} / \delta\rho(z_i)\delta\rho(z_j)$ by employing the Ornstein-Zernike equation. **2.** Through automatic differentiation (autodiff²), the neural functional $F_\theta^{(2)}$ is optimized to fit the Hessian of the model output with respect to input density profiles to $\delta^2 \mathcal{F}_{\text{exc}} / \delta\rho(z_i)\delta\rho(z_j)$. **3.** The optimized model can then be applied in cDFT to obtain non-uniform equilibrium density profiles through automatic differentiation (autodiff) and the free energy $F_\theta^{(2)}$ for a system of Lennard-Jones particles subjected to arbitrary external potentials.

REINVENT (*de novo* drug design)

Deep Reinforcement Learning for Non-diff Score functions

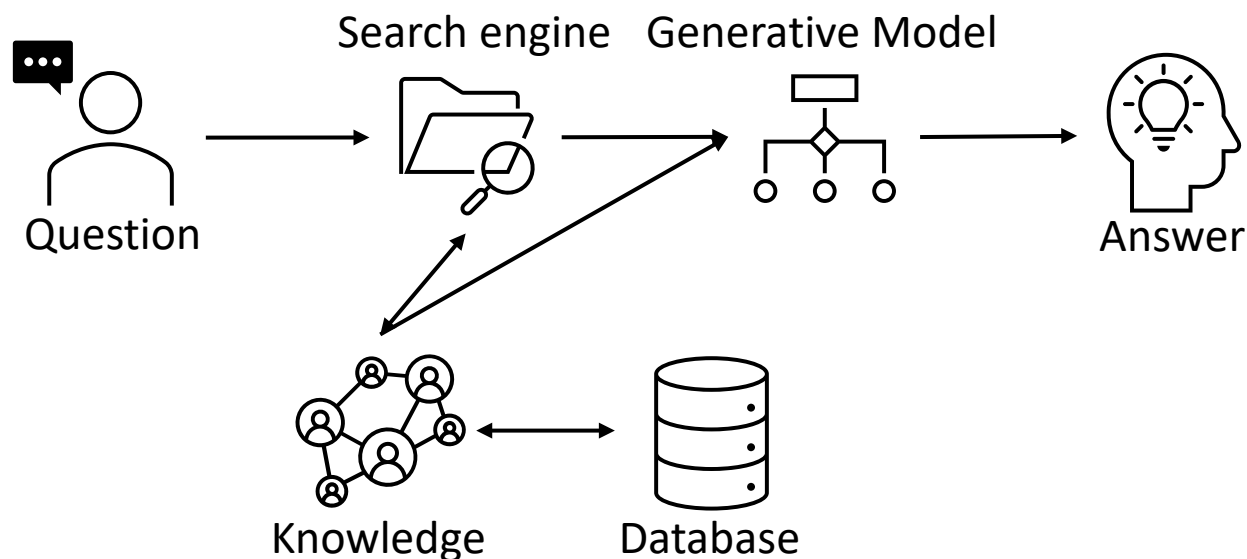


Challenges

Challenges

Trustworthiness

To what extent can we trust generated molecules?
Do ML models “understand” quantum chemistry?
Can we add *knowledge* to these models?
Do we need *something* to go beyond training data?

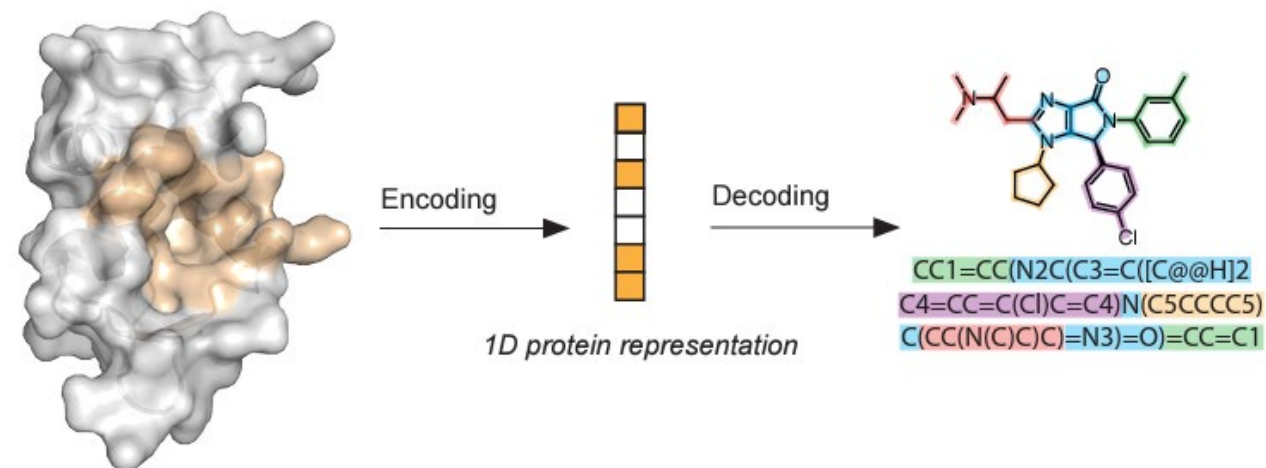


Challenges

Trustworthiness

Structure-based Molecule Generation

- Affinity prediction
- Molecular docking
- Lead optimization



Challenges

Trustworthiness

Structure-based Molecule Generation

- Affinity prediction
- Molecular docking
- Lead optimization

$$E = E_{\text{bonds}} + E_{\text{angle}} + E_{\text{dihedral}} + E_{\text{non-bonded}}$$

$$E_{\text{non-bonded}} = E_{\text{electrostatic}} + E_{\text{van der Waals}}$$

$$\Delta G_{\text{binding}} = \Delta E_{\text{VDW}} + \Delta E_{\text{el}} + \Delta E_{\text{H-bond}} + \Delta G_{\text{sol}}$$

How to further include simulators?

- Differentiable energy functions, score functions?

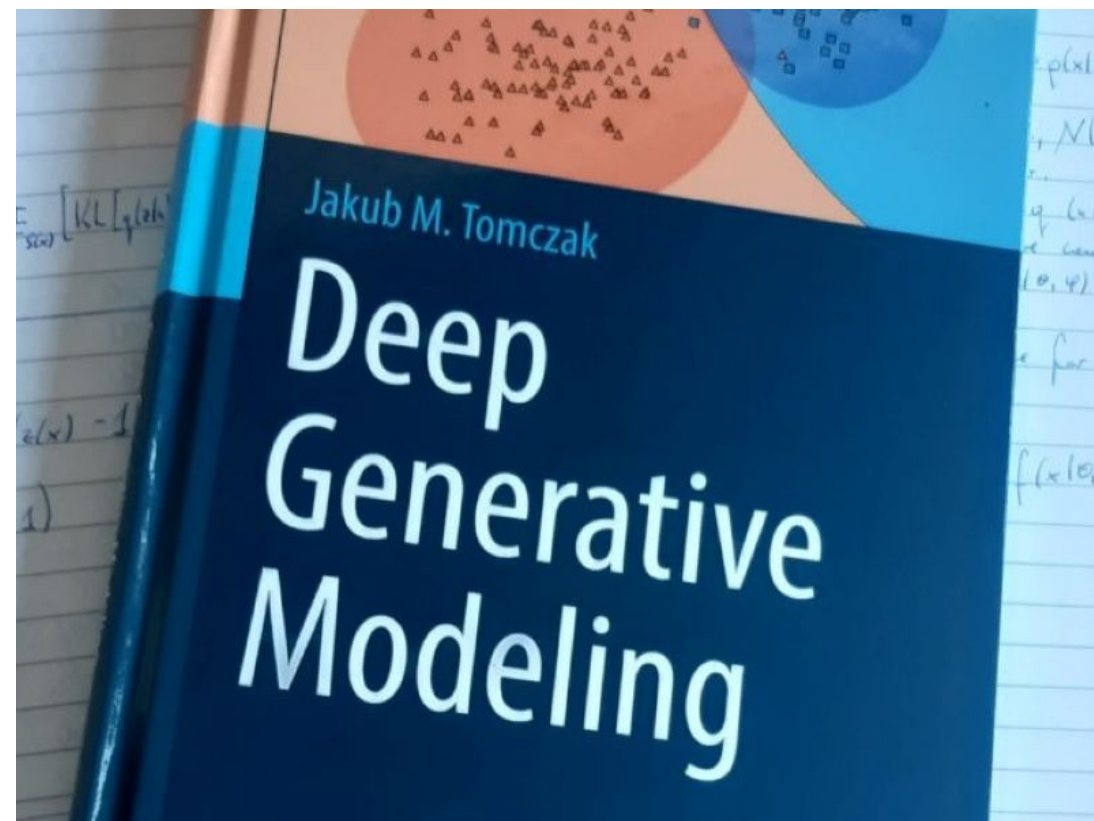
Take-aways

Take-aways

ML (especially Generative AI) has shown a great potential for molecule generation!

Many open research questions

Trustworthiness: Incorporating knowledge (quantum chemistry) into ML for molecular modeling



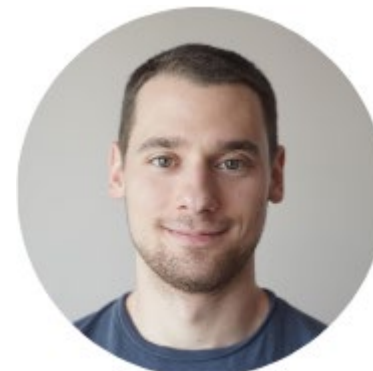
(Always remember about shameless self-promotion)

Thank you! Questions?

Contact: j.m.tomczak@tue.nl
jmk.tomczak@gmail.com

Generativ/e

Generative AI Group: <https://generativeai-tue.github.io/>



Adam Izdebski
HELMHOLTZ
MUNICH



Ewelina Węglarz-Tomczak
 NatInLab



Ewa Szczurek
HELMHOLTZ
MUNICH

Amsterdam
AI Solutions

Amsterdam AI Solutions: <https://amsterdamaisolutions.com/>