Generative AI for Drug Discovery

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

Molecule generation

Problem

Goal: Generate novel molecules

Constraints: Molecules that have certain desirable properties

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

Problem

Goal: Generate novel molecules

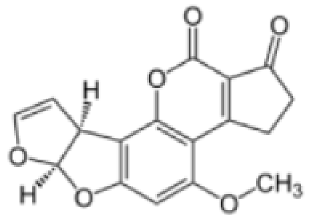
Constraints: Molecules that have certain desirable properties

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

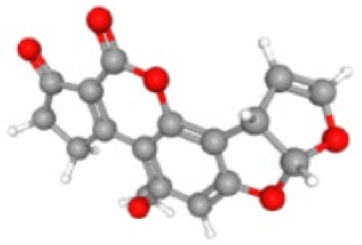
Representation of molecules:

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

SMILES



Molecular graph

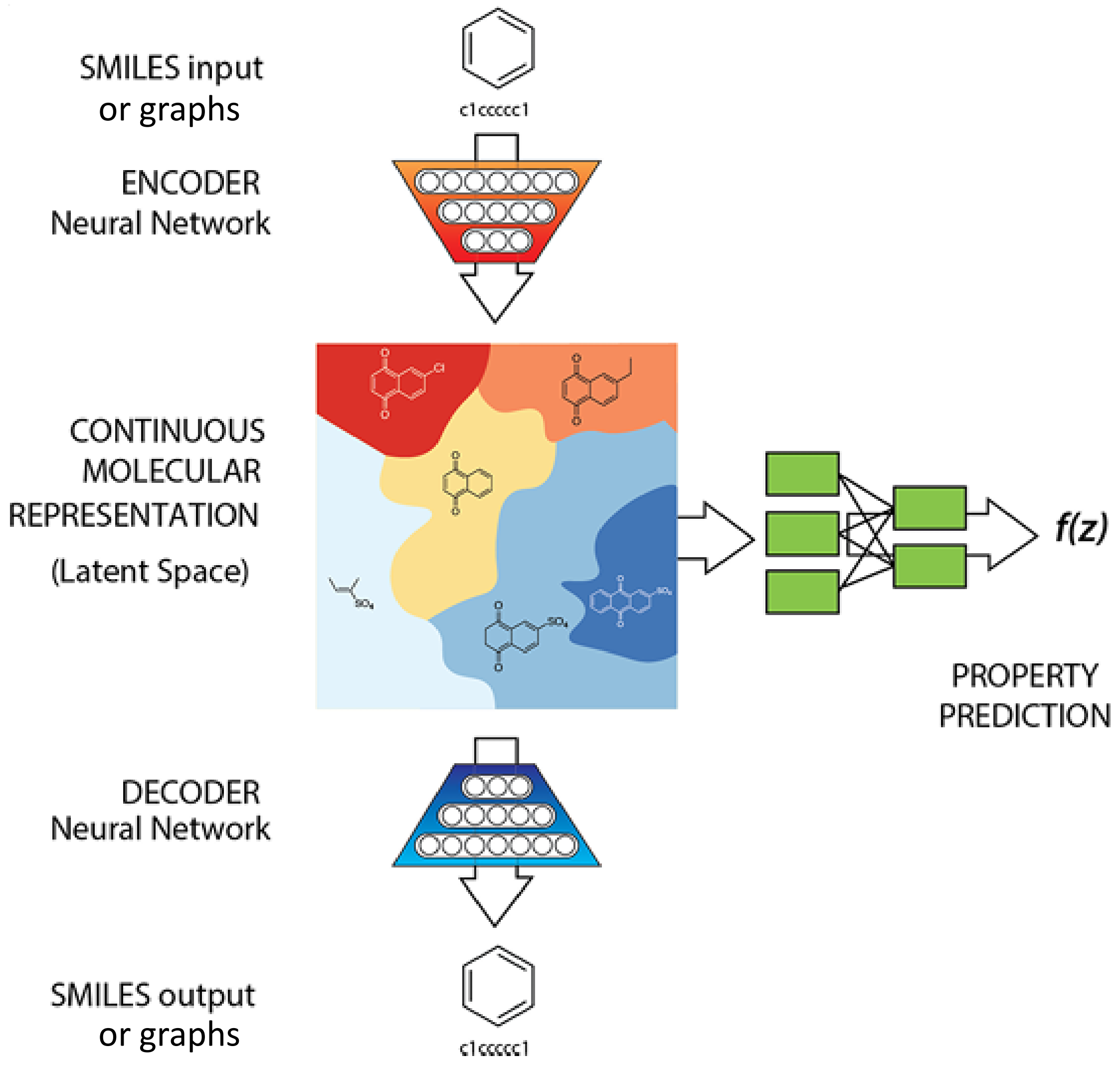


Molecular graph

+

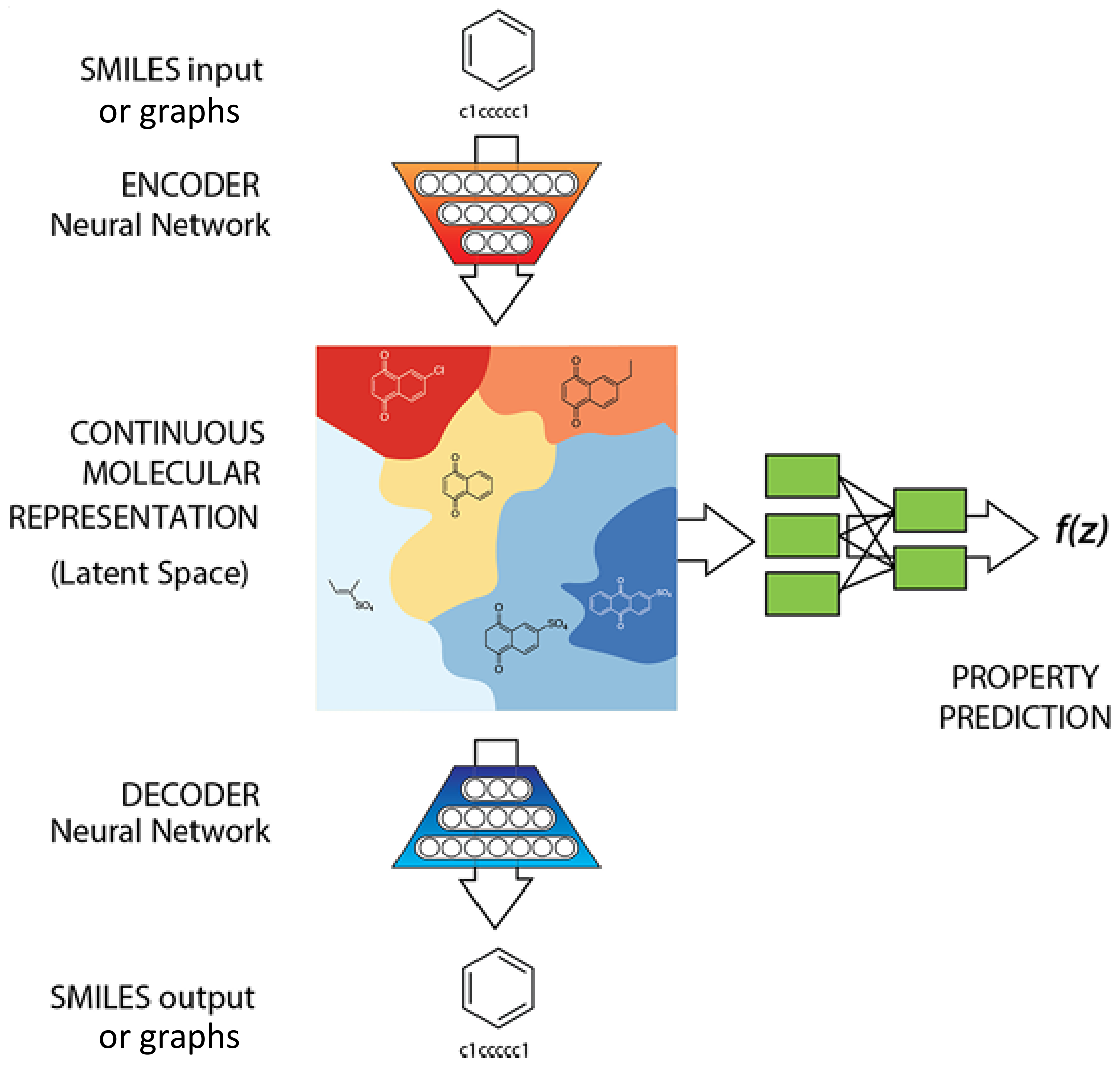
3D positions

Molecule Generation with Joint VAEs



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

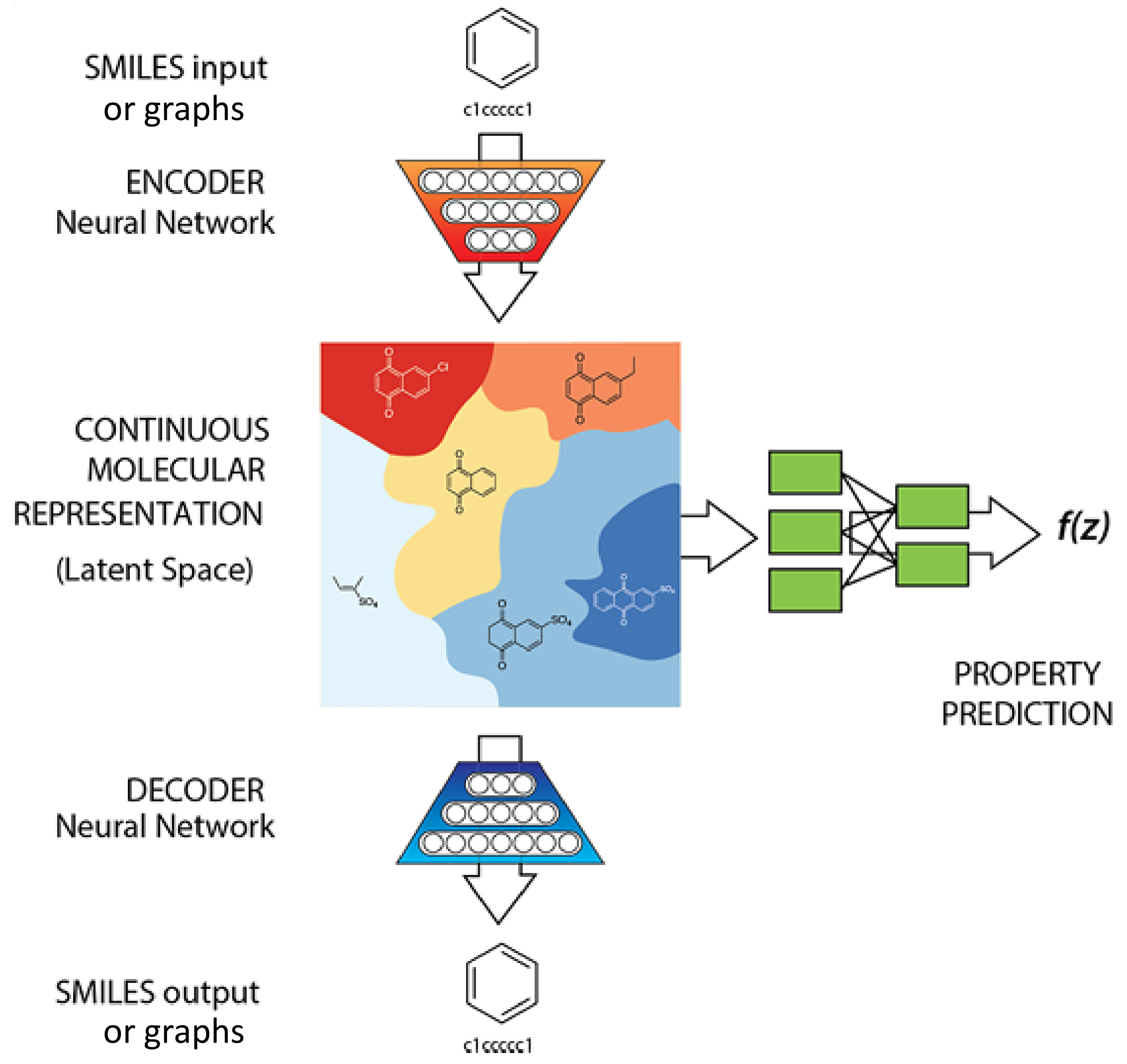
Molecule Generation with Joint VAEs



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

(V)AE

Molecule Generation with Joint VAEs



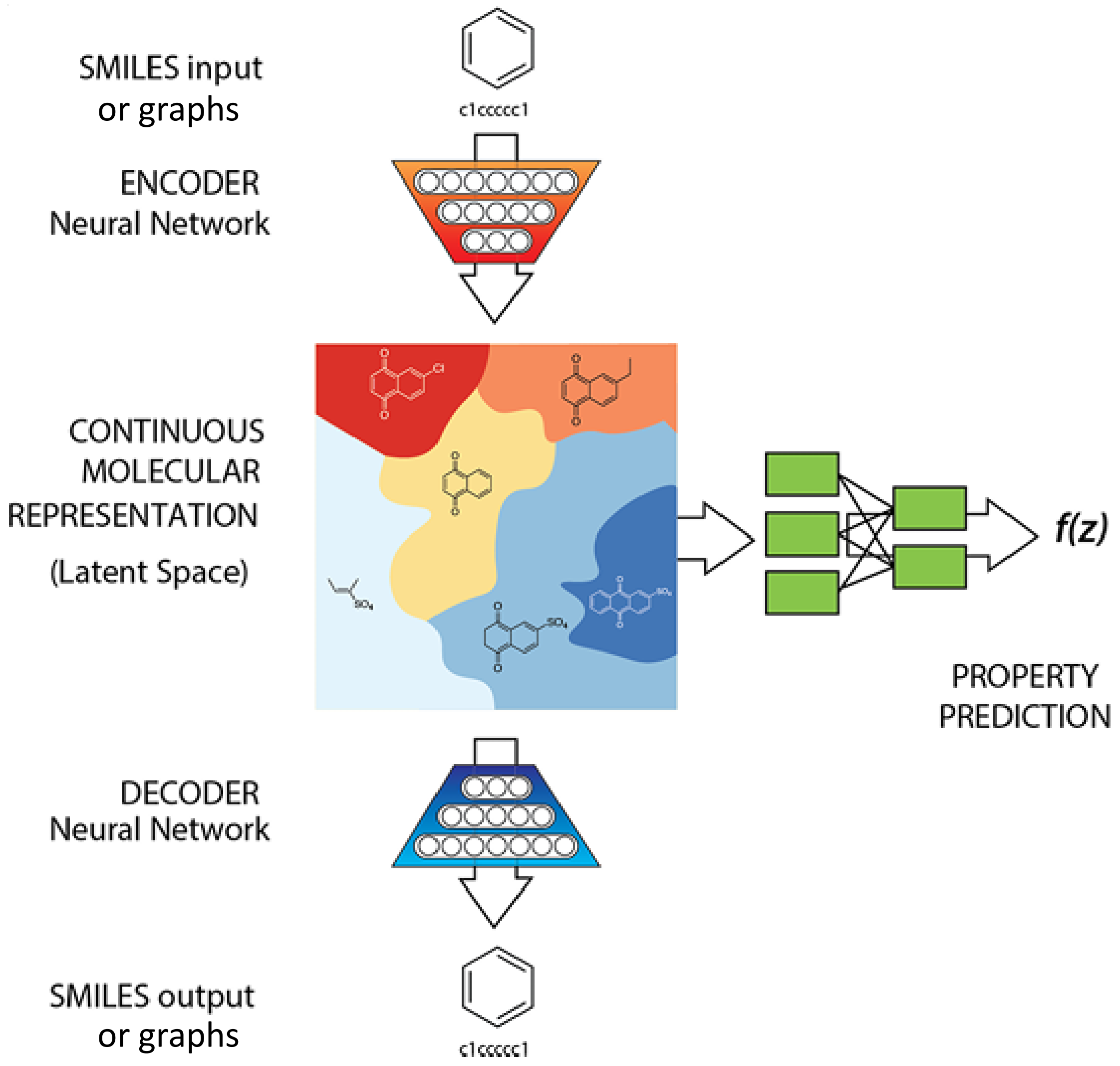
$$\ln p(\mathbf{x}, y) = \ln p(y|\mathbf{x}) + \ln p(\mathbf{x})$$

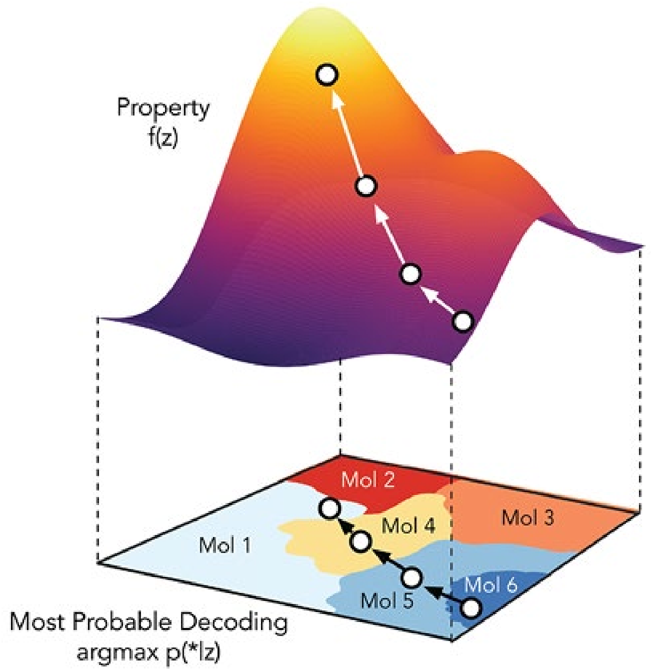
encoder

+

predictor

Molecule Generation with Joint VAEs

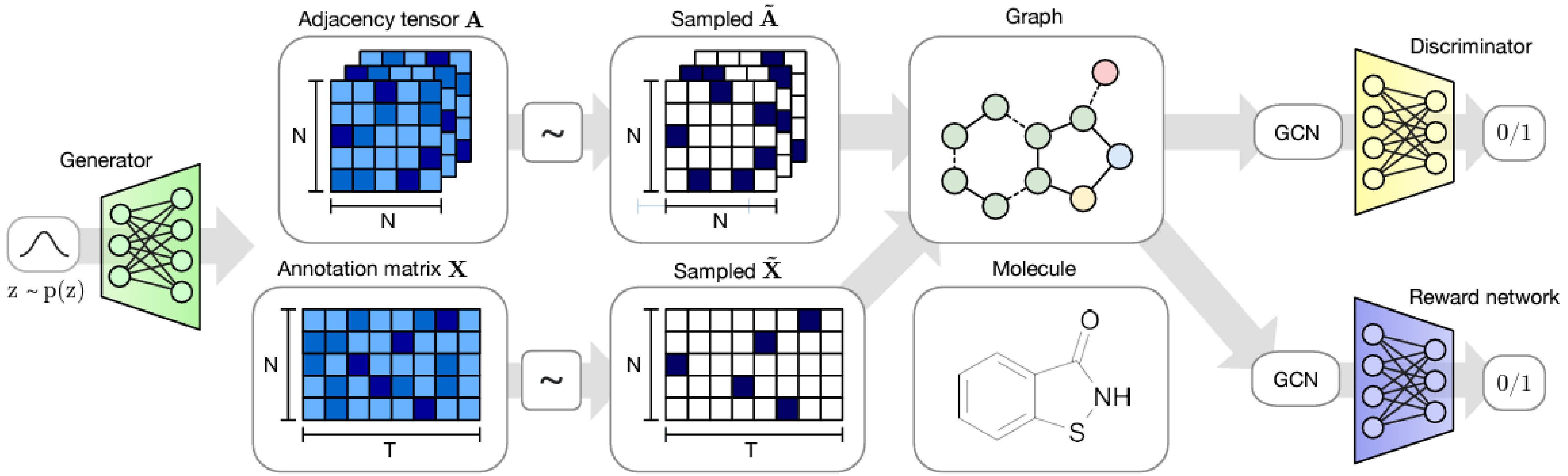




Optimization through Gradient Descent

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

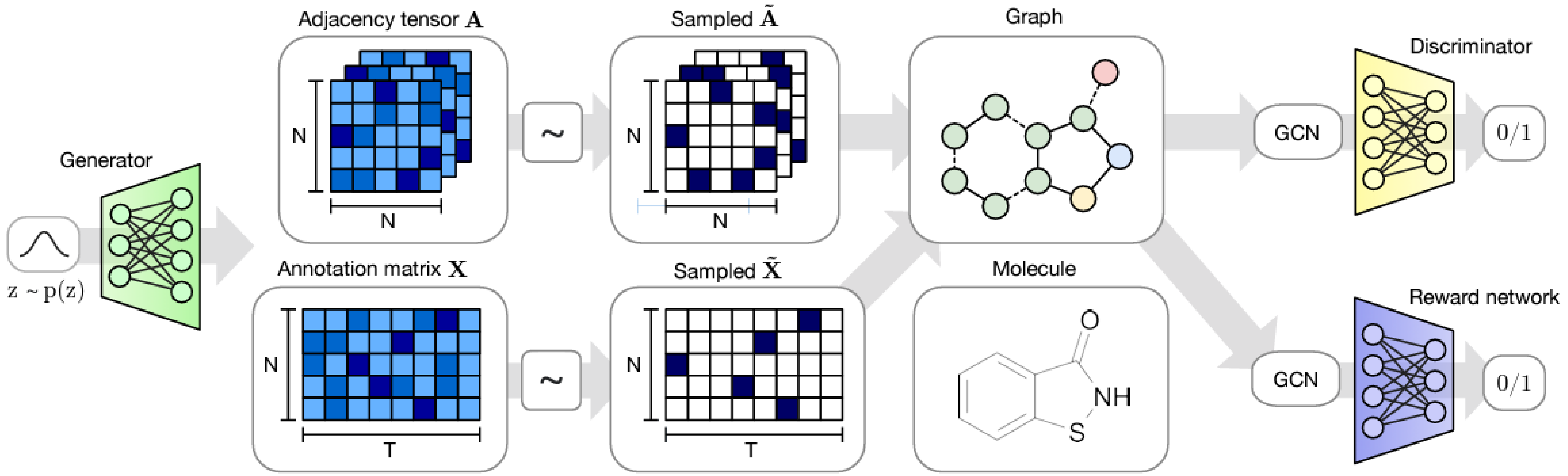
Molecule Generation with GANs


Adjacency tensor $\mathbf{A}$
Sampled $\tilde{\mathbf{A}}$
Graph
Discriminator
Generator
N
N
N
N
~
~
GCN
GCN
0/1
0/1
$z \sim p(z)$
Annotation matrix $\mathbf{X}$
Sampled $\tilde{\mathbf{X}}$
Molecule
Reward network
T
T
O
NH
S

Objective: adversarial loss + RL

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

Molecule Generation with GANs

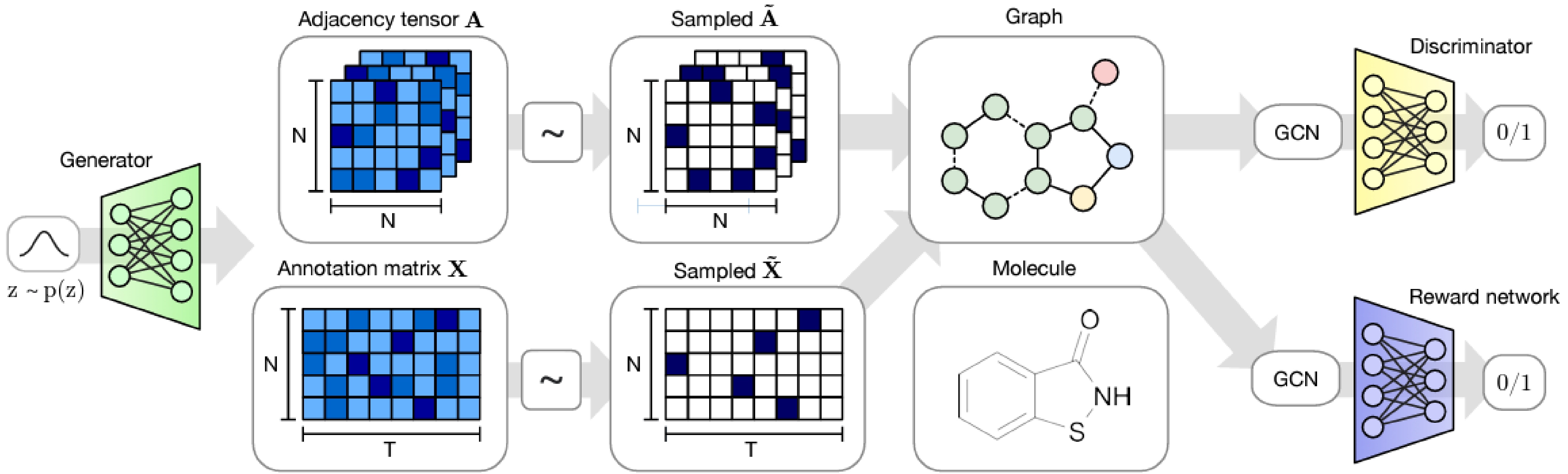


Objective: adversarial loss + RL

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

generation

Molecule Generation with GANs


Generator
$z \sim p(z)$
Adjacency tensor $\mathbf{A}$
N
N
Annotation matrix $\mathbf{X}$
N
T
~
~
Sampled $\tilde{\mathbf{A}}$
N
N
Sampled $\tilde{\mathbf{X}}$
N
T
Graph
Molecule
O
NH
S
GCN
Discriminator
0/1
GCN
Reward network
0/1

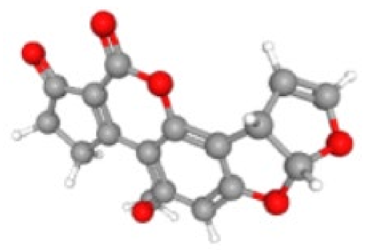
Objective: adversarial loss + RL

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

properties

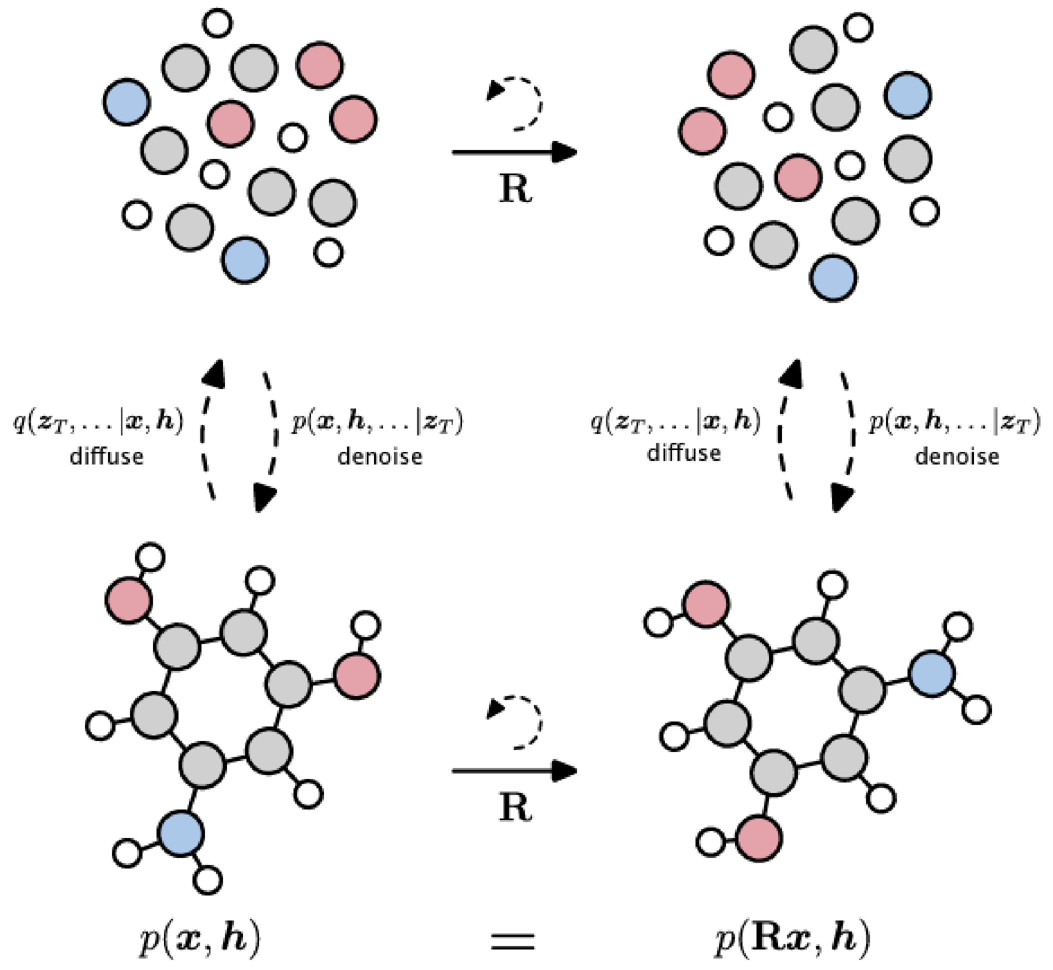
generation

Molecule Generation with Diffusion Models



Molecular graph
+
3D positions

Equivariance is important

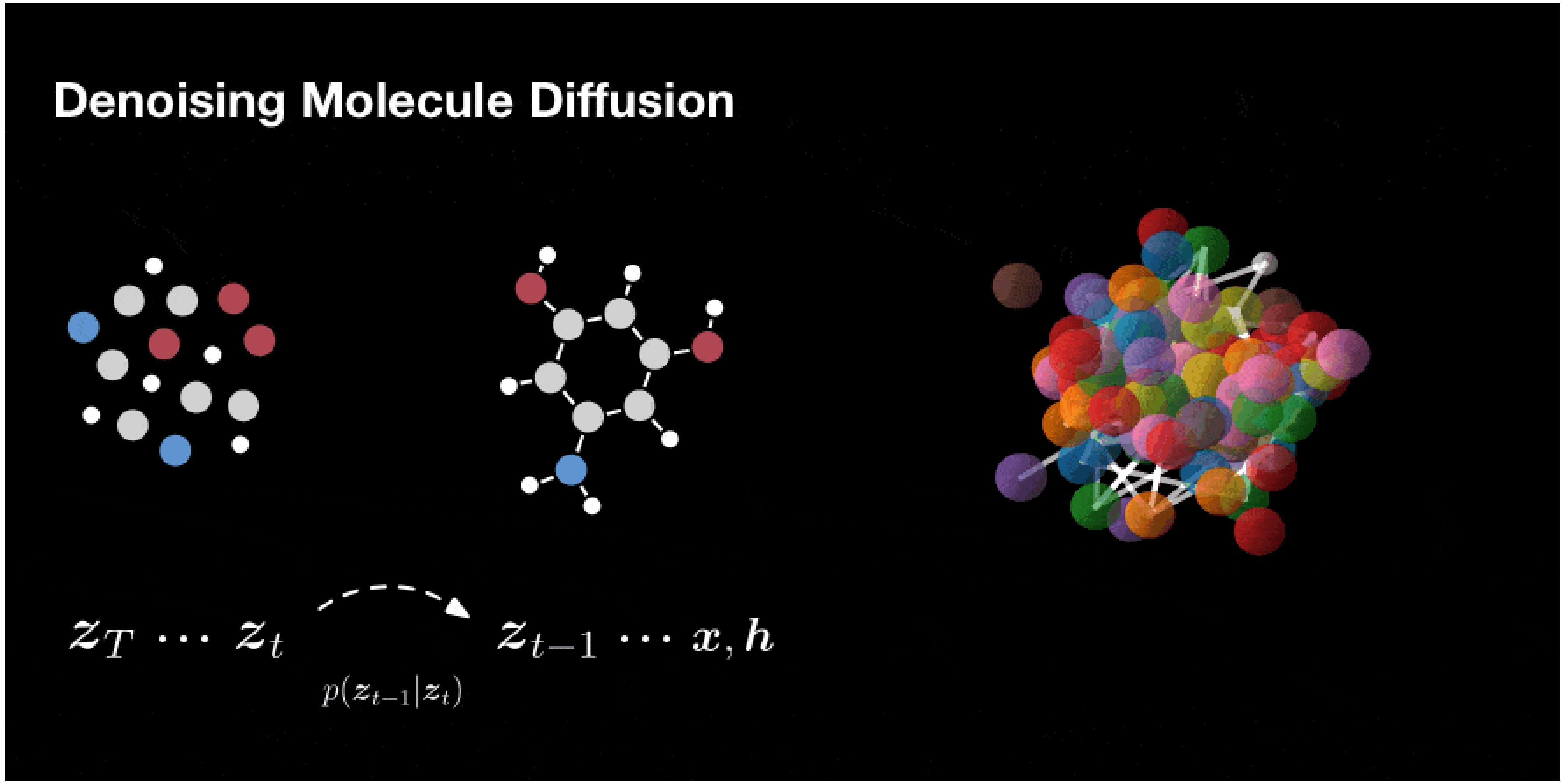


$\mathbf{R}$
$q(\boldsymbol{z}_T, \ldots | \boldsymbol{x}, \boldsymbol{h})$
diffuse
$p(\boldsymbol{x}, \boldsymbol{h}, \ldots | \boldsymbol{z}_T)$
denoise
$q(\boldsymbol{z}_T, \ldots | \boldsymbol{x}, \boldsymbol{h})$
diffuse
$p(\boldsymbol{x}, \boldsymbol{h}, \ldots | \boldsymbol{z}_T)$
denoise
$\mathbf{R}$
$p(\boldsymbol{x}, \boldsymbol{h}) = p(\mathbf{R}\boldsymbol{x}, \boldsymbol{h})$

Hoogeboom et al., ICML 2022

"Equivariant Diffusion for Molecule Generation in 3D"

Molecule Generation with Diffusion Models



Hoogeboom et al., ICML 2022
"Equivariant Diffusion for Molecule Generation in 3D"

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

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

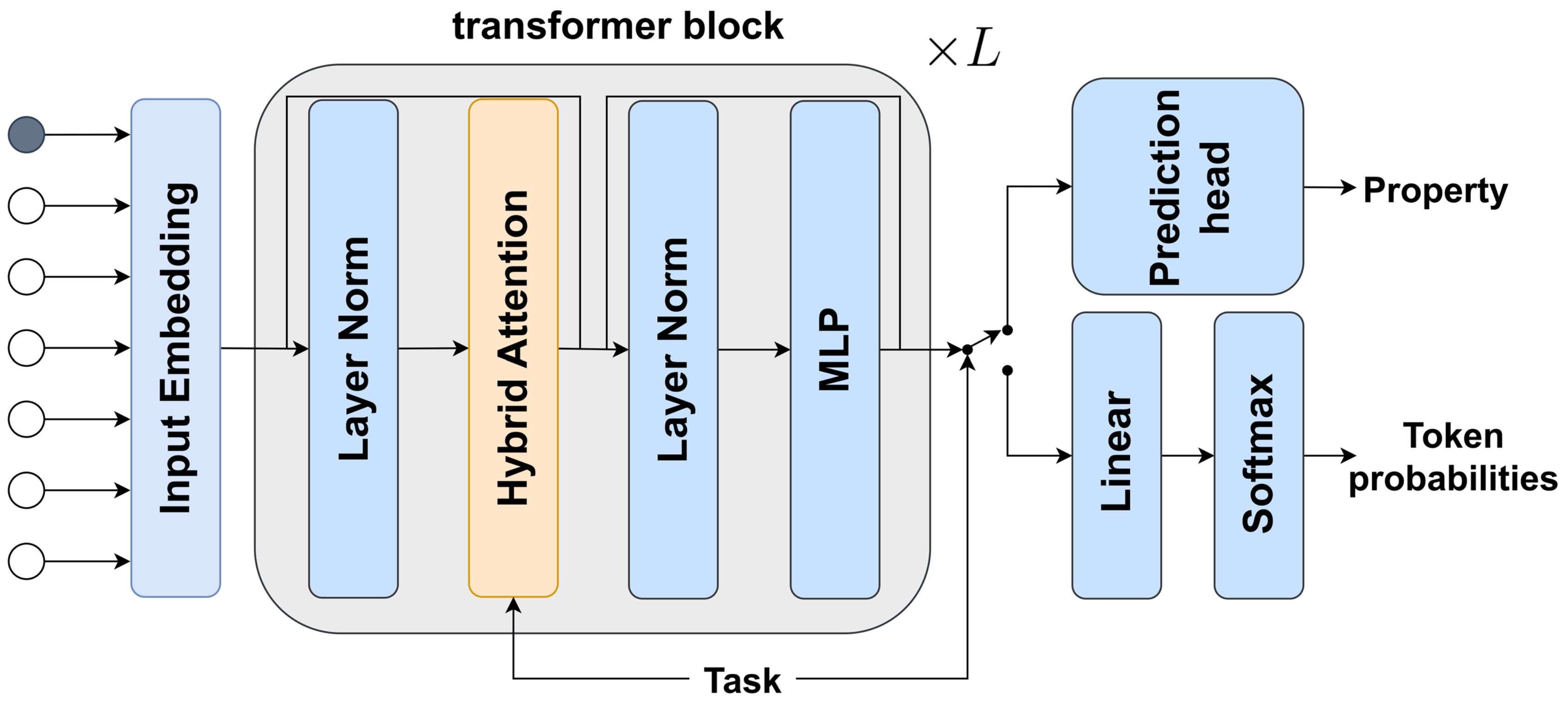
How to do that?

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:



$$
\begin{aligned}
p_\theta(\mathbf{x}) &:= f_{\theta,\psi}(\mathbf{x}, \emptyset, <\mathrm{GEN}>), \\
p_{\theta,\psi}(y \mid \mathbf{x}) &:= f_{\theta,\psi}(\mathbf{x}, \emptyset, <\mathrm{PRED}>), \\
\prod_{m \in M} p_\theta(x_m \mid \mathbf{x}_{-m}) &:= f_{\theta,\psi}(\mathbf{x}, M, <\mathrm{REC}>),
\end{aligned}
$$

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***.

Izdebski, Weglarz-Tomczak, Szczurek, Tomczak, 2023
"Joint Molecule Generation and Property Prediction with Jointformer"

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:

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

Devlin et al. , NAACL-HLT, 2019

"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"

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:

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

EITHER predictive OR generative

Devlin et al. , NAACL-HLT, 2019
"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"

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

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., 2018, “Improving language understanding by generative pre-training”

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:

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

Dong et al., NeurIPS, 2019
"Unified language model pre-training for natural language understanding and generation"

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:

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

EITHER predictive OR generative

Dong et al., NeurIPS, 2019

“Unified language model pre-training for natural language understanding and generation”

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:

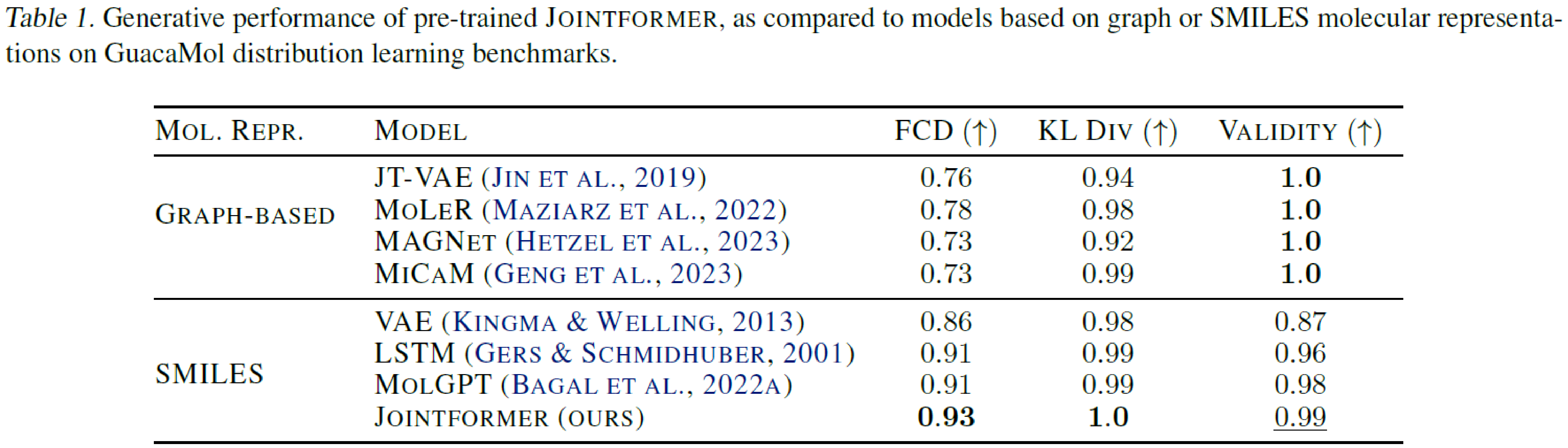


Table 1. Generative performance of pre-trained JOINTFORMER, as compared to models based on graph or SMILES molecular representations on GuacaMol distribution learning benchmarks.

MOL. REPR.	MODEL	FCD (↑)	KL DIV (↑)	VALIDITY (↑)
GRAPH-BASED	JT-VAE (JIN ET AL., 2019)	0.76	0.94	**1.0**
	MOLER (MAZIARZ ET AL., 2022)	0.78	0.98	**1.0**
	MAGNET (HETZEL ET AL., 2023)	0.73	0.92	**1.0**
	MICAM (GENG ET AL., 2023)	0.73	0.99	**1.0**
SMILES	VAE (KINGMA & WELLING, 2013)	0.86	0.98	0.87
	LSTM (GERS & SCHMIDHUBER, 2001)	0.91	0.99	0.96
	MOLGPT (BAGAL ET AL., 2022A)	0.91	0.99	0.98
	JOINTFORMER (OURS)	**0.93**	**1.0**	0.99

Pre-trained without labels! But with the masked loss and the generative loss.

Generative & Predictive capabilities of Jointformers

The performance of our Jointformer:

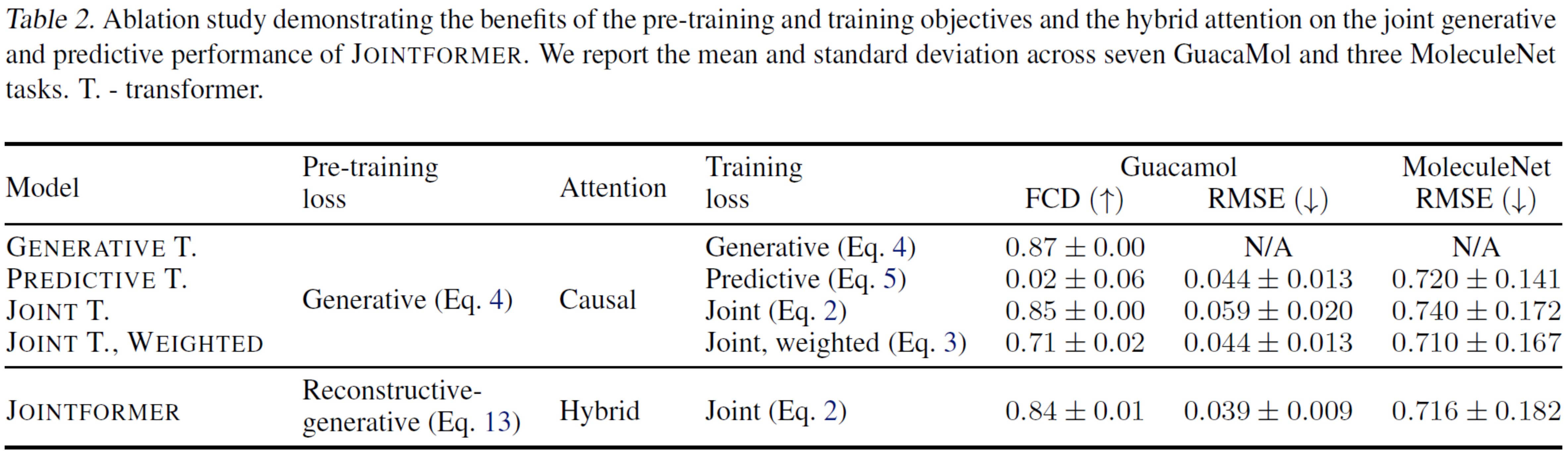


Table 2. Ablation study demonstrating the benefits of the pre-training and training objectives and the hybrid attention on the joint generative and predictive performance of JOINTFORMER. We report the mean and standard deviation across seven GuacaMol and three MoleculeNet tasks. T. - transformer.

Model	Pre-training loss	Attention	Training loss	Guacamol FCD (↑)	Guacamol RMSE (↓)	MoleculeNet RMSE (↓)
GENERATIVE T.	Generative (Eq. 4)	Causal	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.			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:

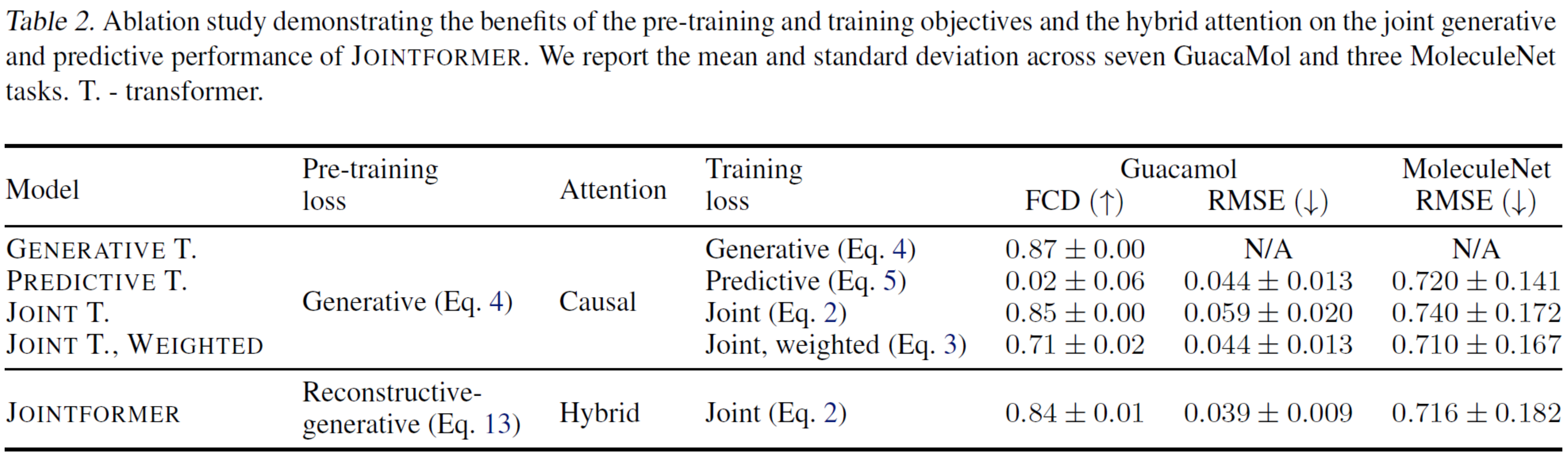


Table 2. Ablation study demonstrating the benefits of the pre-training and training objectives and the hybrid attention on the joint generative and predictive performance of JOINTFORMER. We report the mean and standard deviation across seven GuacaMol and three MoleculeNet tasks. T. - transformer.

Model	Pre-training loss	Attention	Training loss	Guacamol FCD (↑)	Guacamol RMSE (↓)	MoleculeNet RMSE (↓)
GENERATIVE T.	Generative (Eq. 4)	Causal	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.			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!

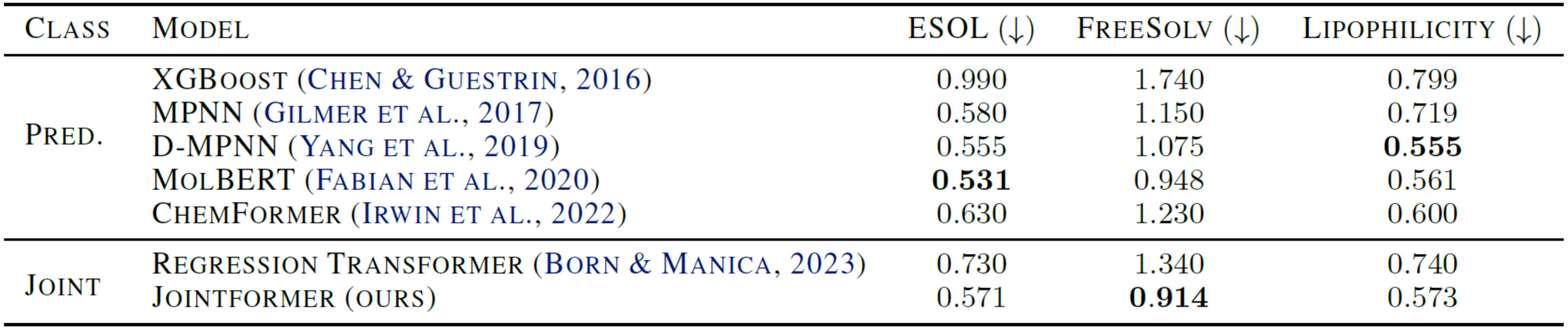
But “no-free-lunch”: the generative performance drops a bit.

Izdebski, Weglarz-Tomczak, Szczurek, Tomczak, 2024
“Joint Molecule Generation and Property Prediction with Jointformer”

Generative & Predictive capabilities of Jointformers

The performance of our Jointformer:

Table 4. Predictive performance of purely predictive and joint models across three molecular property prediction tasks from the MoleculeNet benchmark. JOINTFORMER outperforms other joint models across all tasks and achieves the best performance on the FreeSolv task, as measured by RMSE.



CLASS	MODEL	ESOL (↓)	FREESOLV (↓)	LIPOPHILICITY (↓)
PRED.	XGBOOST (CHEN & GUESTRIN, 2016)	0.990	1.740	0.799
	MPNN (GILMER ET AL., 2017)	0.580	1.150	0.719
	D-MPNN (YANG ET AL., 2019)	0.555	1.075	**0.555**
	MOLBERT (FABIAN ET AL., 2020)	**0.531**	0.948	0.561
	CHEMFORMER (IRWIN ET AL., 2022)	0.630	1.230	0.600
JOINT	REGRESSION TRANSFORMER (BORN & MANICA, 2023)	0.730	1.340	0.740
	JOINTFORMER (OURS)	0.571	**0.914**	0.573

Our Jointformer can generate, all other methods can't!
Overall, we are always in top-3!
We beat our direct competitor (no likelihood-based training).

Izdebski, Weglarz-Tomczak, Szczurek, Tomczak, 2024
"Joint Molecule Generation and Property Prediction with Jointformer"

Generative & Predictive capabilities of Jointformers

The performance of our Jointformer:

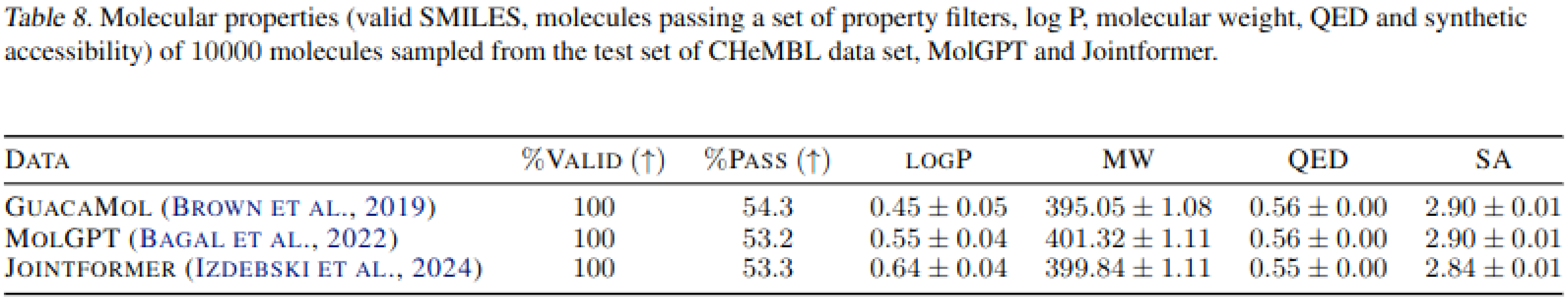


Table 8. Molecular properties (valid SMILES, molecules passing a set of property filters, log P, molecular weight, QED and synthetic accessibility) of 10000 molecules sampled from the test set of CHeMBL data set, MolGPT and Jointformer.

DATA	%VALID (↑)	%PASS (↑)	LOGP	MW	QED	SA
GUACAMOL (BROWN ET AL., 2019)	100	54.3	0.45 ± 0.05	395.05 ± 1.08	0.56 ± 0.00	2.90 ± 0.01
MOLGPT (BAGAL ET AL., 2022)	100	53.2	0.55 ± 0.04	401.32 ± 1.11	0.56 ± 0.00	2.90 ± 0.01
JOINTFORMER (IZDEBSKI ET AL., 2024)	100	53.3	0.64 ± 0.04	399.84 ± 1.11	0.55 ± 0.00	2.84 ± 0.01

Molecular properties of our approach are in line with test data (GuacaMol).

Generative & Predictive capabilities of Jointformers

The performance of our Jointformer:

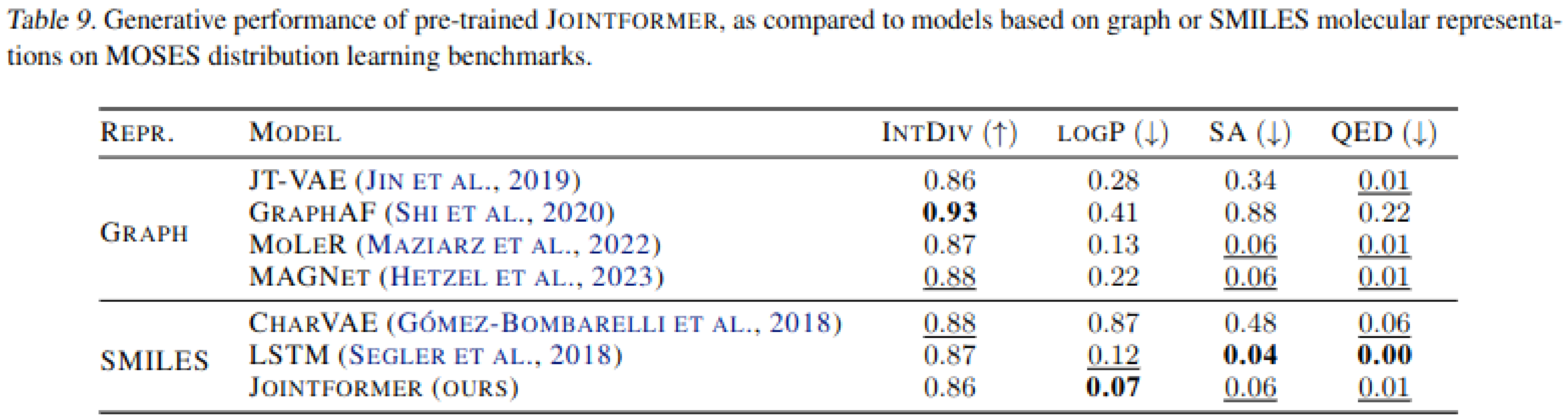


Table 9. Generative performance of pre-trained JOINTFORMER, as compared to models based on graph or SMILES molecular representations on MOSES distribution learning benchmarks.

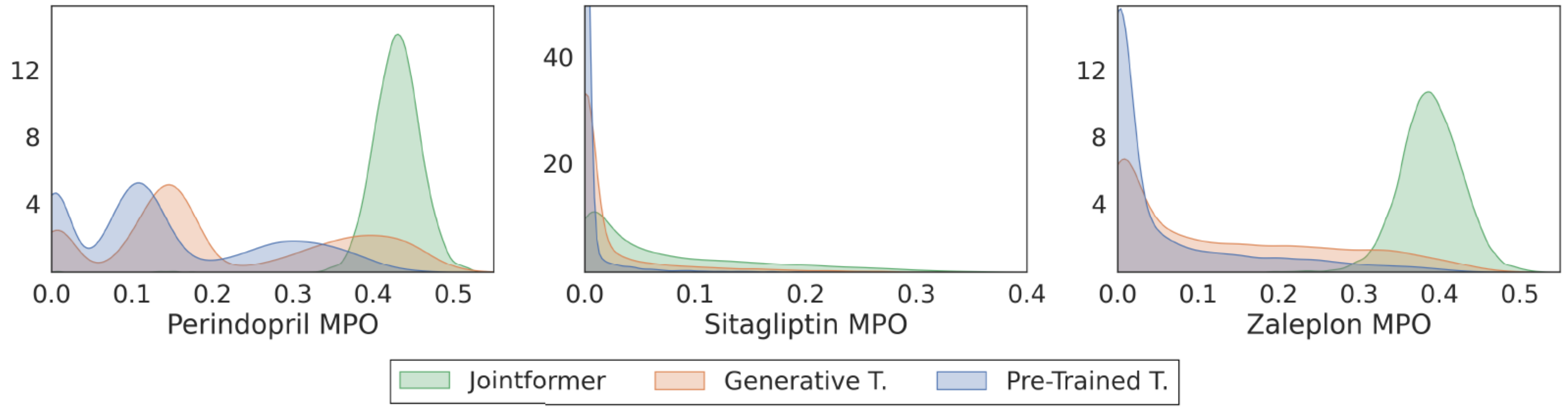
REPR.	MODEL	INTDIV (↑)	LOGP (↓)	SA (↓)	QED (↓)
GRAPH	JT-VAE (JIN ET AL., 2019)	0.86	0.28	0.34	0.01
	GRAPHAF (SHI ET AL., 2020)	**0.93**	0.41	0.88	0.22
	MOLER (MAZIARZ ET AL., 2022)	0.87	0.13	0.06	0.01
	MAGNET (HETZEL ET AL., 2023)	0.88	0.22	0.06	0.01
SMILES	CHARVAE (GÓMEZ-BOMBARELLI ET AL., 2018)	0.88	0.87	0.48	0.06
	LSTM (SEGLER ET AL., 2018)	0.87	0.12	**0.04**	**0.00**
	JOINTFORMER (OURS)	0.86	**0.07**	0.06	0.01

Jointformer achieves on par performance to SOTA purely generative models.

Conditional sampling from Jointformers

Distribution of properties sampled conditionally:

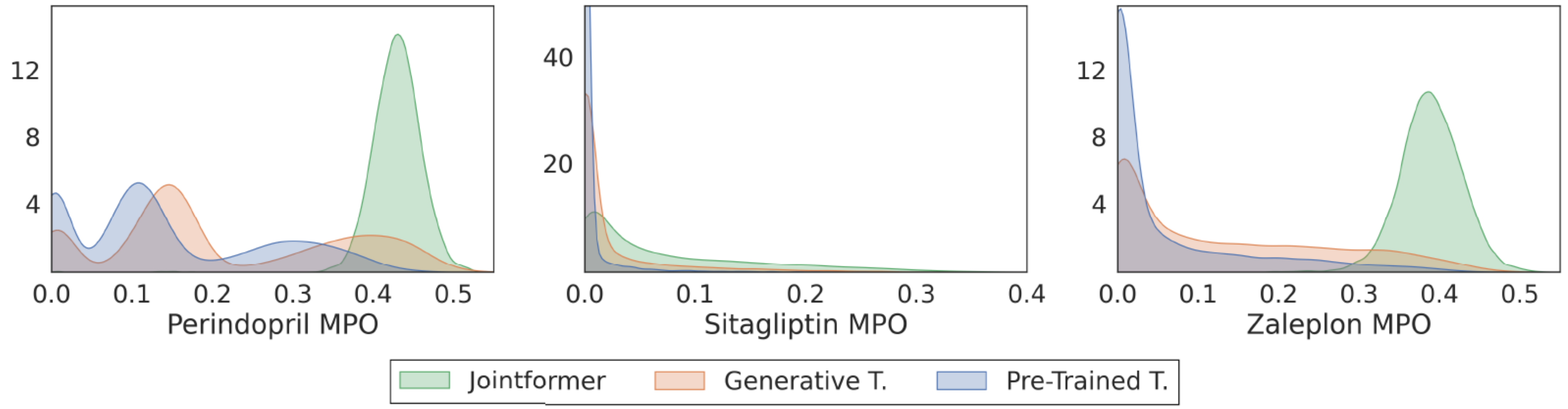
x-axis: property value, y-axis: counts



Conditional sampling from Jointformers

Distribution of properties sampled conditionally:

x-axis: property value, y-axis: counts

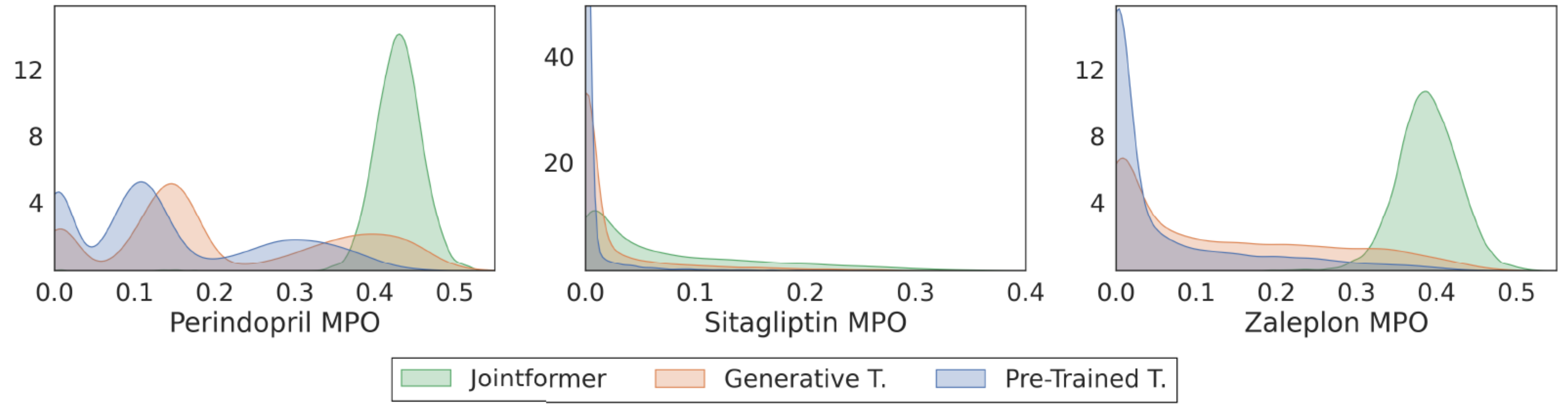


Pre-trained Transformer learns the data distribution perfectly

Conditional sampling from Jointformers

Distribution of properties sampled conditionally:

x-axis: property value, y-axis: counts

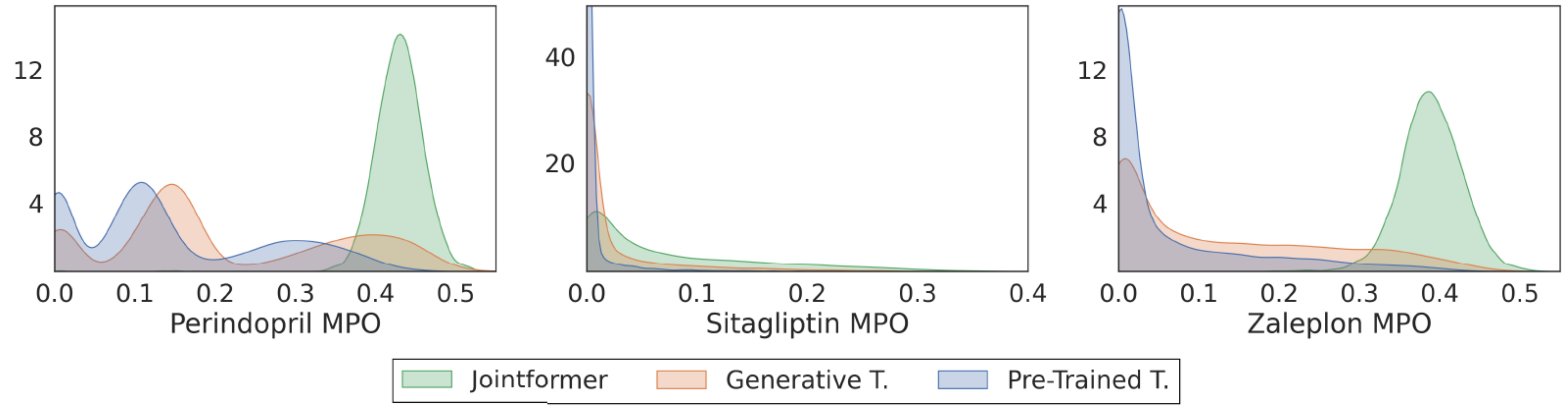


Fine-tuned Transformer shifts the distribution!

Conditional sampling from Jointformers

Distribution of properties sampled conditionally:

x-axis: property value, y-axis: counts



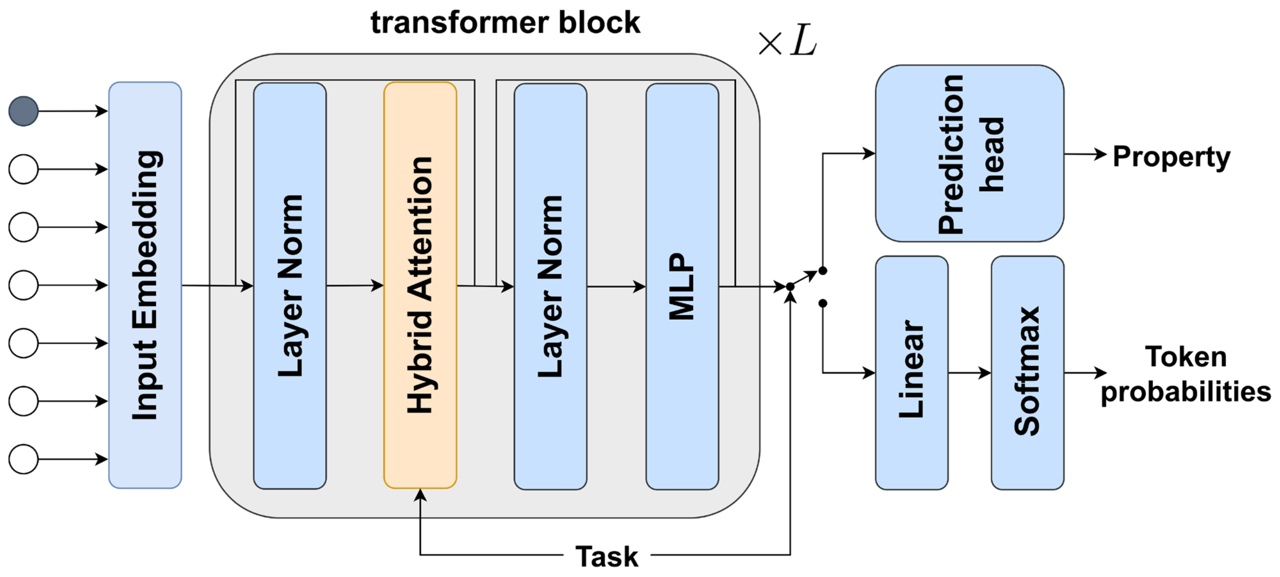
BUT our Jointformer samples **BEST** molecules!

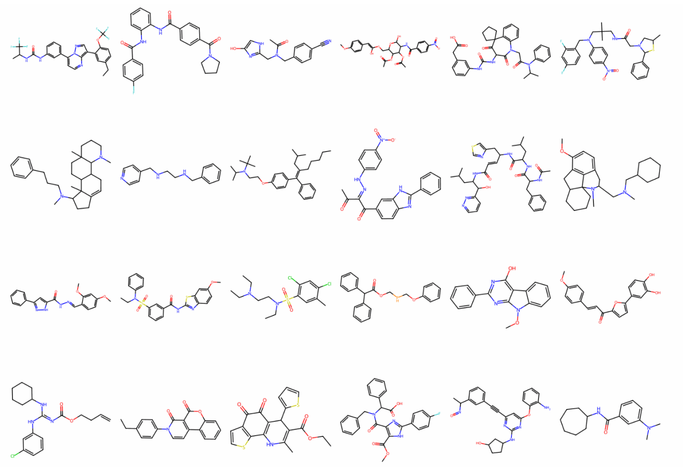
Jointformers

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)!





Izdebski, Weglarz-Tomczak, Szczurek, Tomczak, 2024
"Joint Molecule Generation and Property Prediction with Jointformer"

Summary

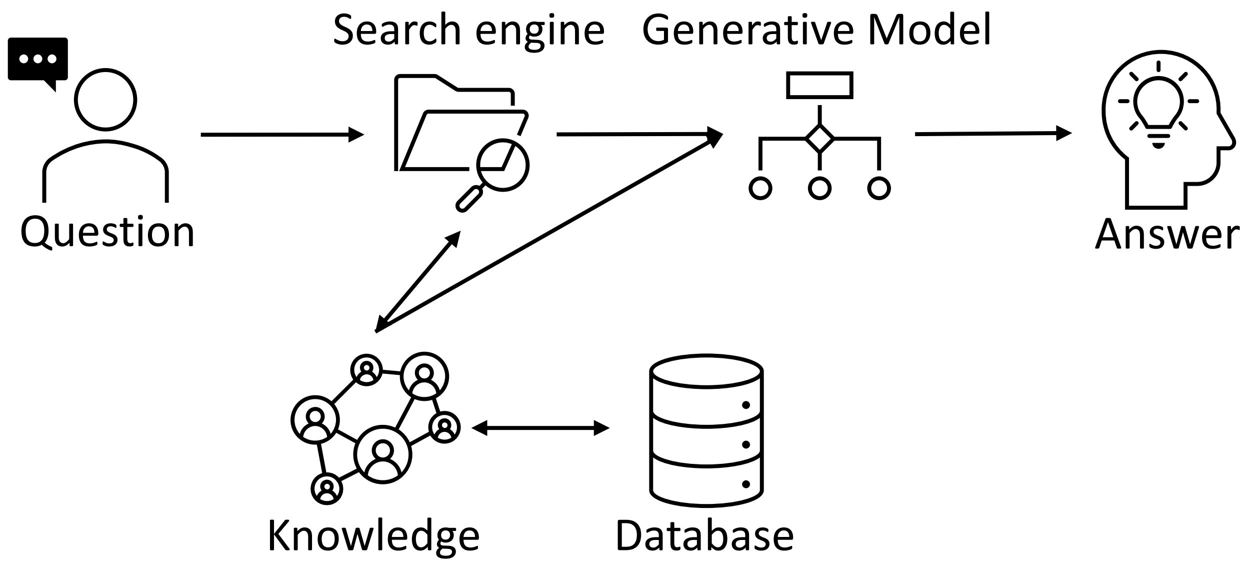
Model	Representation	Objective	Constraints	EVIdence Tractability
VAEs	SMILES Graphs	ELBO	Property predictor	✗
GANs	Graphs	Adversarial loss	RL loss	✗
Diffusion models	Graphs + 3D	ELBO	Property predictor	✗
Jointformer	SMILES	Joint Likelihood	-	✓

Challenges

Challenges

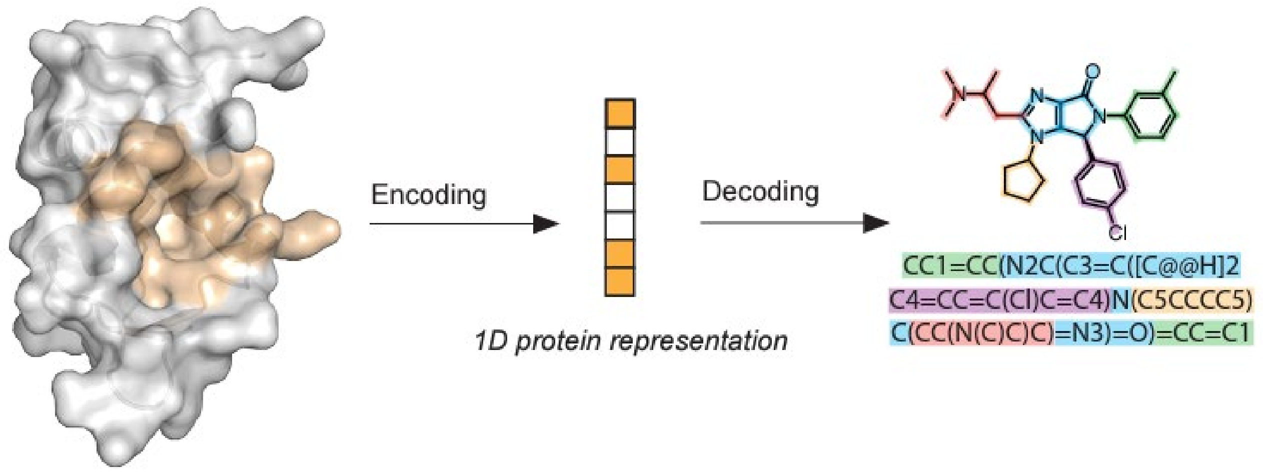
- **Trustworthiness**

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



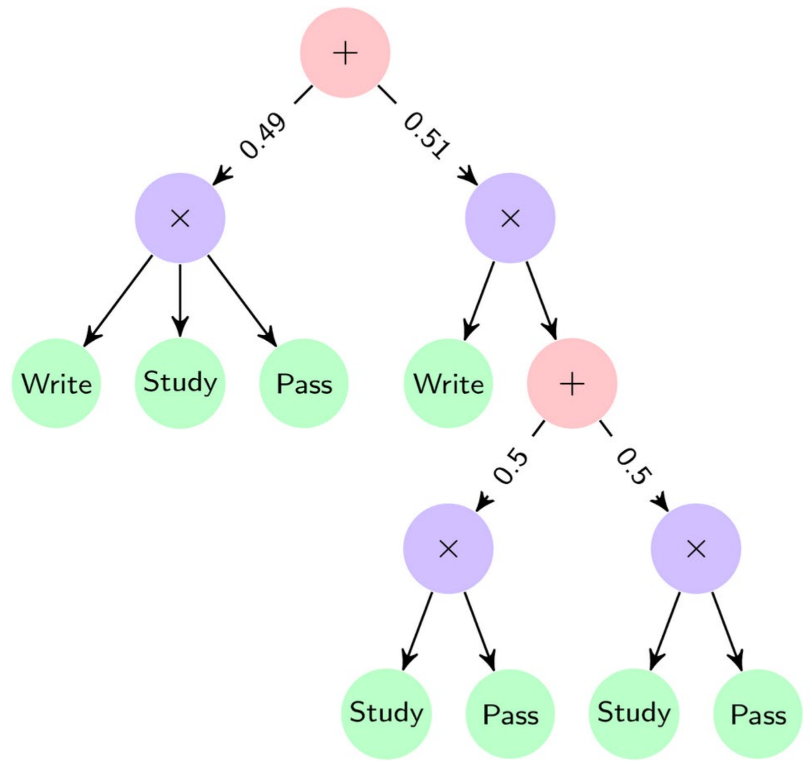
Challenges

- **Trustworthiness**
- **Structure-based Molecule Generation**
    - Affinity prediction
    - Molecular docking
    - Lead optimization


Encoding
Decoding
1D protein representation
Cl
CC1=CC(N2C(C3=C([C@@H]2
C4=CC=C(Cl)C=C4)N(C5CCCC5)
C(CC(N(C)C)C)=N3)=O)=CC=C1

Challenges

- **Trustworthiness**
- **Structure-based Molecule Generation**
  - Affinity prediction
  - Molecular docking
  - Lead optimization
- **Further tractability of generative models for molecules**
  - Developing models with MAR, COND and MAP tractability



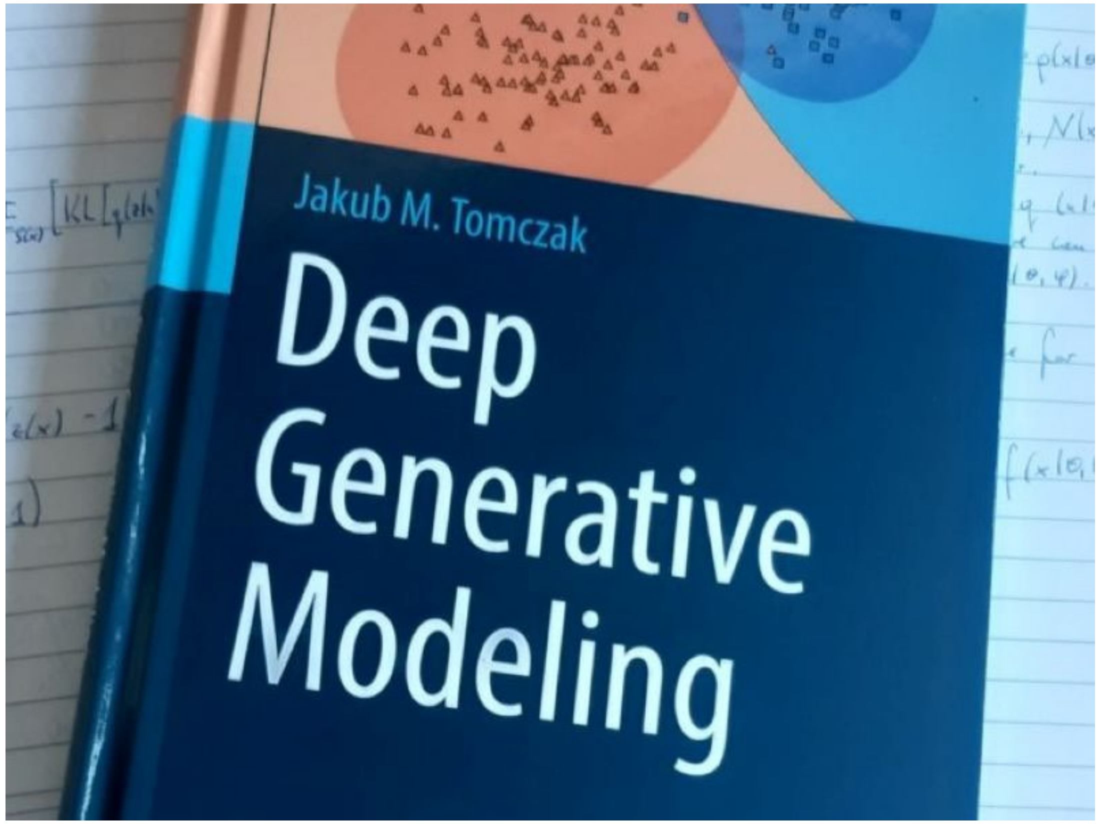
Take-aways

Take-aways

Generative AI has shown a great potential for molecule generation!

Many open research questions (including tractability!)

Trustworthiness: Incorporating knowledge (quantum chemistry) into Generative AI 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/

Amsterdam AI Solutions

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