DiGress: Discrete denoising diffusion for graph generation

Clément Vignac

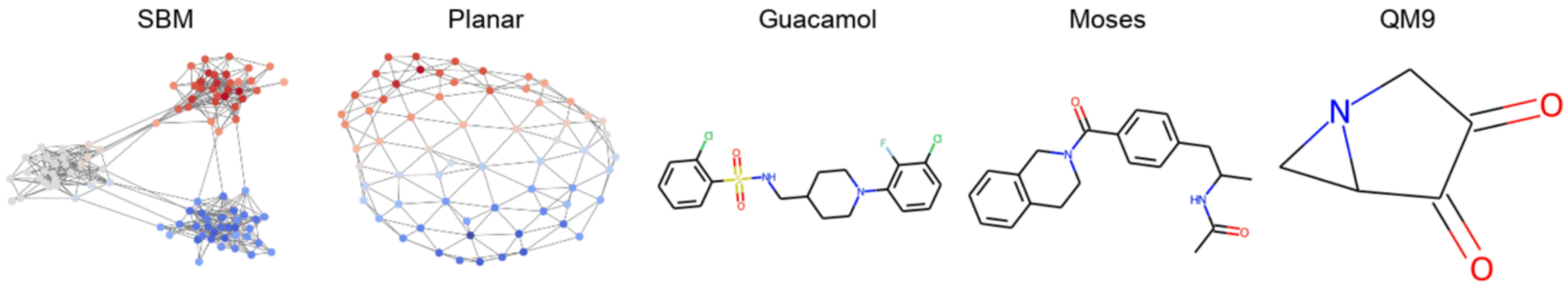
Joint work with Igor Krawczuk (co-1st author), Antoine Siraudin, Bohan Wang,
Volkan Cevher & Pascal Frossard (EPFL)

CIRM, 09/11/2022

EPFL

GOAL

Generate graphs with categorical node and edge attributes that *look like* graphs in a training set



Outline

1. Key properties of diffusion models
2. DiGress: a discrete diffusion model for graphs
3. Improvements over the vanilla model
4. Results
5. What’s next?

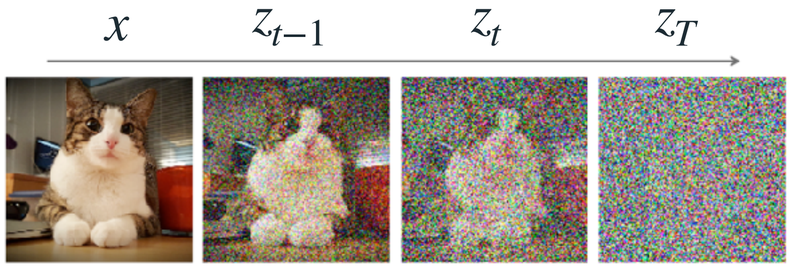
I Denoising diffusion models

DALLE-2

Stable diffusion

+ 3d point clouds, text, audio…

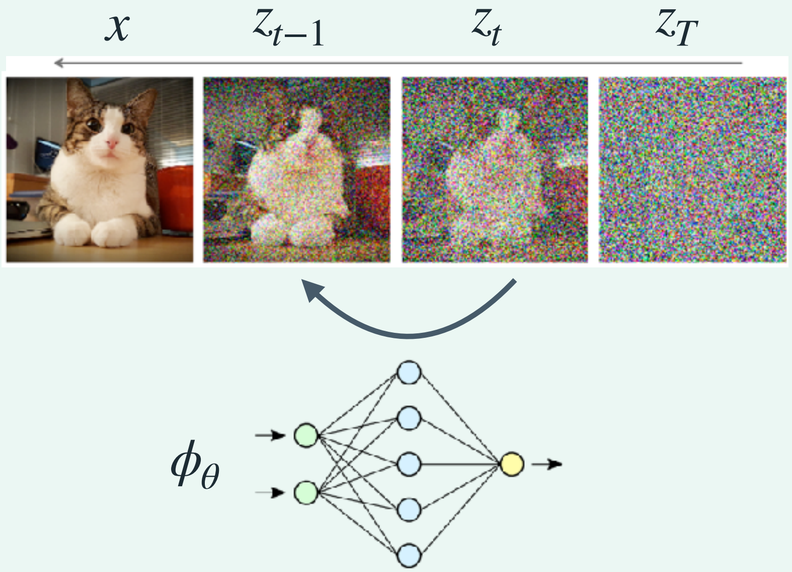
2 equivalent perspectives: score-matching and denoising diffusion



$$q(z_1, \ldots, z_T | x) = q(z_1 | x) \prod_{t=2}^{T} q(z_t | z_{t-1})$$

Noise model

Generate **diffusion trajectories** by recursively adding noise to a data point.



Denoising network

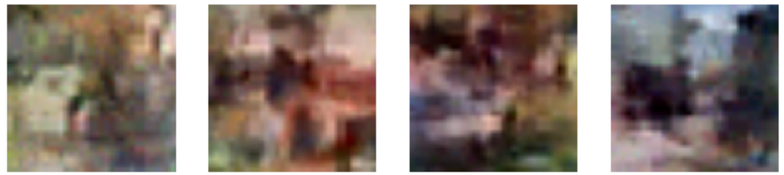
Train a network to predict the **inverse** diffusion iterations.

1. Sample noise from a prior distribution → 2. Apply the denoising network

Iterate

2. Apply the denoising network → 3. Sample from this new distribution

to predict the distribution **at previous step**.



(Sohl-Dickstein et al. 15)

LDSM (*Ho et al. 20)*

What is missing?

Sohl-Dickstein et al.15, Deep Unsupervised Learning using Nonequilibrium Thermodynamics
Ho et al. 20, Denoising diffusion probabilistic models

Algorithm : Training

Input: Data point x
Sample $t \sim \mathcal{U}(1, ..., T)$
Sample $z_t \sim \text{noise}(x, t)$
$\hat{p}_x \leftarrow f_\theta(z_t, t)$
$\text{optimizer.step}(\text{loss}(x, \hat{p}_x))$

Algorithm : Sampling

Sample z_t from a prior distribution
for $t = T$ **to** 1 **do**
  $\hat{p}_x \leftarrow f_\theta(z_t, t)$
  $\hat{p}_{t-1} \leftarrow \int_x q(z_{t-1}|z_t, x) d\hat{p}_x$
  $z_{t-1} \sim \hat{p}_{t-1}$
end
return z_0

Differences with the naive model

- The model is not trained on trajectories
- $\hat{p}_{t-1}$ is deduced from $\hat{p}_x$ or $\hat{p}_\epsilon$

Algorithm : Training

Input: Data point x
Sample $t \sim \mathcal{U}(1, ..., T)$
Sample $z_t \sim \text{noise}(x, t)$
$\hat{p}_x \leftarrow f_\theta(z_t, t)$
$\text{optimizer.step}(\text{loss}(x, \hat{p}_x))$

- z^{t-1} (= the target of early diffusion models) is **very noisy**
- There are two sources of noise:
  - The data sample $x \sim p_{\text{data}}$
  - The trajectory sampled from x
- If we can predict x directly we remove some stochasticity

Differences with the naive model

- The model is not trained on trajectories
- $\hat{p}_{t-1}$ is deduced from $\hat{p}_x$ or $\hat{p}_\epsilon$

Property 1

$q(z^t \mid x)$ should have a closed-form formula

Avoid adding noise recursively during training to generate trajectories.

Property 2

$\int q(z^{t-1} \mid z^t, x)\ d\hat{p}_x$ should be tractable

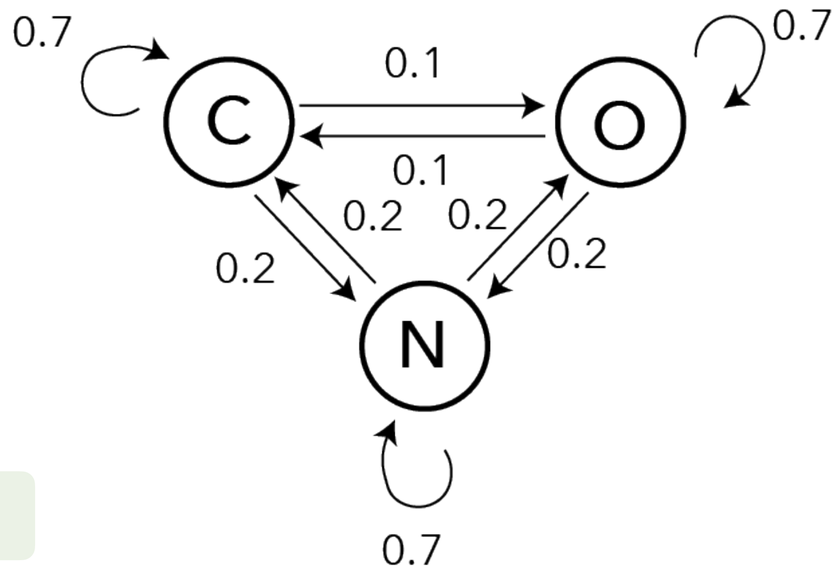
Use x as the target of the neural network instead of z^{t-1}.

Property 3

$\lim_{T \to \infty} q(z^T \mid x)$ should not depend on x

Prior distribution =
Limit noise distribution

2 main frameworks: Gaussian noise (95% of the papers) and discrete diffusion



- Noise model = Markov process over a discrete state-space

  Defined by transition matrices $(Q^1, \ldots, Q^T)$:

$$q(z^t \mid z^{t-1}) = z^{t-1} Q^t$$

- Multiple steps: $q(z^t \mid x) = x\ \bar{Q}^t \quad \text{for} \quad \bar{Q}^t = Q^1 \ldots Q^t$ — Property 1

- Bayes rule: $q(z^{t-1} \mid z^t, x) \propto z^t (Q^t)' \odot x\ \bar{Q}^{t-1}$ — Property 2

- Most common model: Uniform transitions — Property 3

$$\begin{bmatrix} 0.8 & 0.1 & 0.1 \\ 0.1 & 0.8 & 0.1 \\ 0.1 & 0.1 & 0.8 \end{bmatrix} \qquad Q^t = \alpha^t I + \beta^t 1_d 1_d'/d \qquad \lim_{t \to \infty} Q^t x = 1_d/d$$

Applications:

Text, image, audio…

Austin et al. 21, Structured Denoising Diffusion Models in Discrete State-Spaces

II DiGress: Discrete Graph Denoising Diffusion

Challenges of learning on graphs

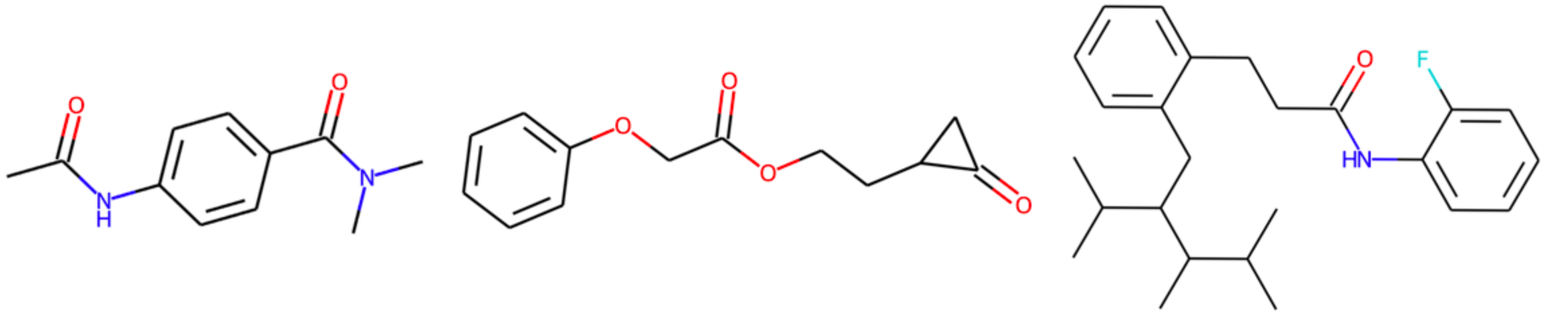
- Varying graph sizes
- Limited representation power

Challenges of graph generation

- Unsupervised learning task that might require graph matching
- Sparsity
- Evaluation

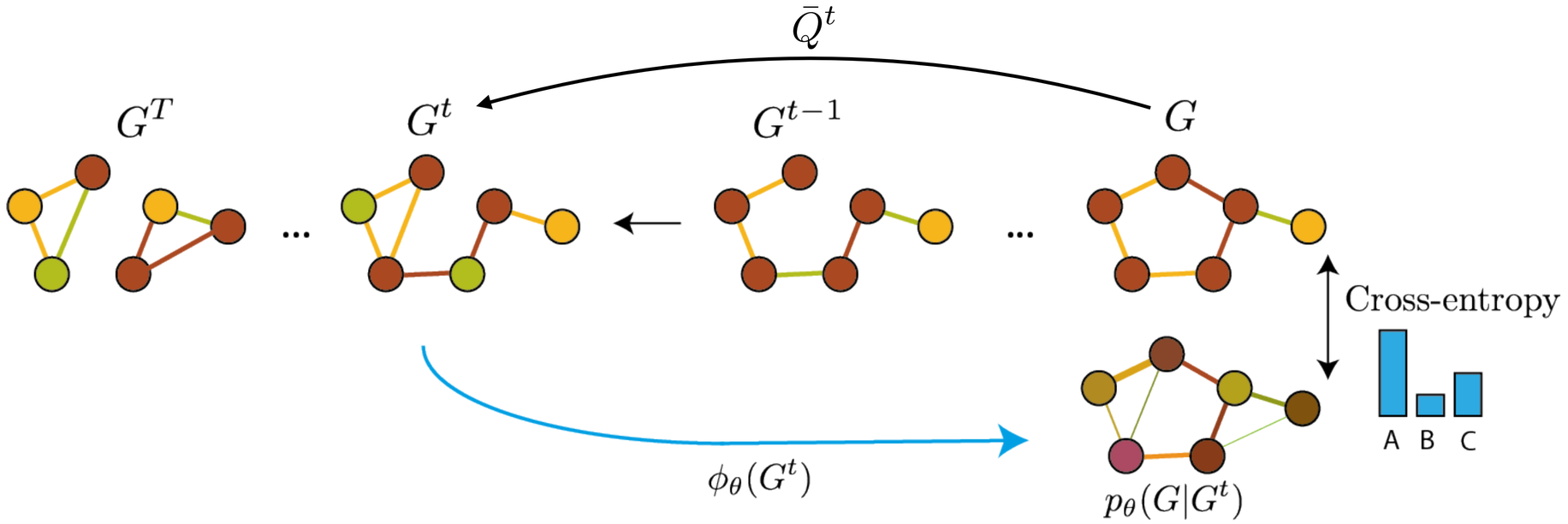
Score-based Generative Modeling of Graphs via the System of SDEs

Jo et al. 2022



- Add **Gaussian noise** to a graph adjacency matrix, and the node and edge features
- Use a graph attention network to denoise this process

- **Scales** significantly better than previous work
- Does not respect the **discrete nature** of graphs
- The noisy graphs are **complete**: structural information (connectivity, cycles) is not defined



Corrupt *independently* on each node and edge

Absence of edge = a particular edge type

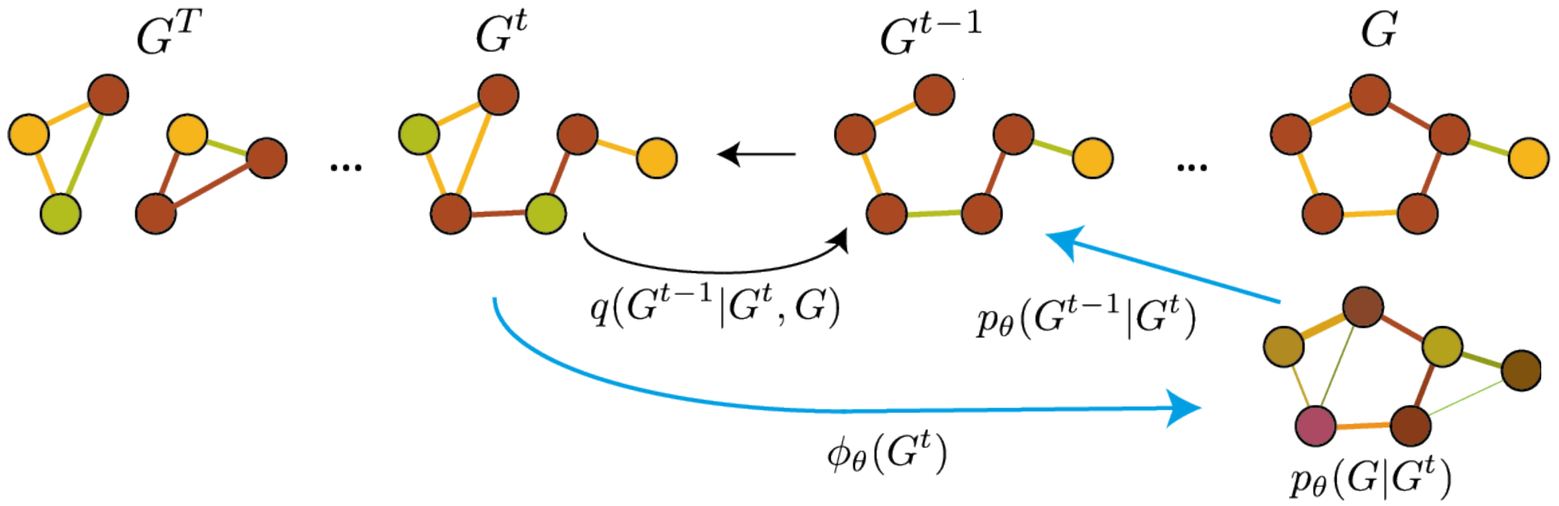
$\mathcal{X}$: space of node types

$\mathcal{E}$: space of edge types

Algorithm : Training DiGress

Input: A graph $G = (\boldsymbol{X}, \boldsymbol{E})$
Sample $t \sim \mathcal{U}(1, ..., T)$
Sample $G^t \sim \boldsymbol{X}\bar{\boldsymbol{Q}}^t_X \times \mathbf{E}\bar{\boldsymbol{Q}}^t_E$
$\hat{p}^X, \hat{p}^E \leftarrow \phi_\theta(G_t, t)$
$loss \leftarrow l_{CE}(\hat{p}^X, \boldsymbol{X}) + \lambda\, l_{CE}(\hat{p}^E, \mathbf{E})$
optimizer. step($loss$)

Graph generation = sequence of node and edge classification tasks



Model:

$$p_\theta(G^{t-1}\,|\,G^t) = \prod_i p_\theta(x_i^{t-1}\,|\,G^t) \prod_{i,j} p_\theta(e_{ij}^{t-1}\,|\,G^t)$$

$$p_\theta(x_i^{t-1}\,|\,G^t) = \int_{x_i} p_\theta(x_i^{t-1}\,|\,x_i, G^t)\; dp_\theta(x_i\,|\,G^t) = \sum_{x\in\mathcal{X}} q(x_i^{t-1}\,|\,x_i = x, x_i^t)\; \hat{p}_i^X(x)$$

Algorithm : Sampling from DiGress

Sample n from the training data distribution
Sample $G^T \sim q_X(n) \times q_E(n)$
for $t = T$ **to** 1 **do**
  $\hat{p}^X, \hat{p}^E \leftarrow \phi_\theta(G^t, t)$
  Sample $G^{t-1} \sim \prod_i p_\theta(x_i^{t-1}|G^t) \times \prod_{ij} p_\theta(e_{ij}^{t-1}|G^t)$
end
return G^0

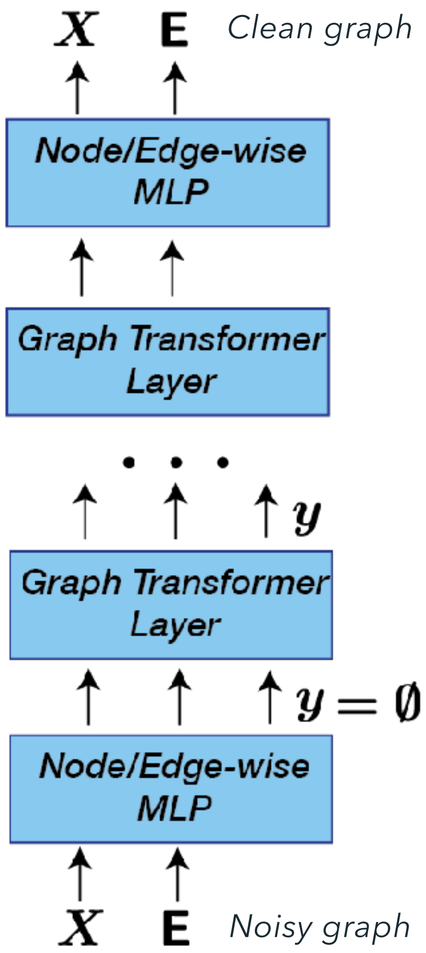
Graph generation = sequence of node and edge classification tasks

02 DIGRESS: DENOISING NETWORK

X: node features (n, a)

E: edge features (n, n, b)

y: graph-level features (c)



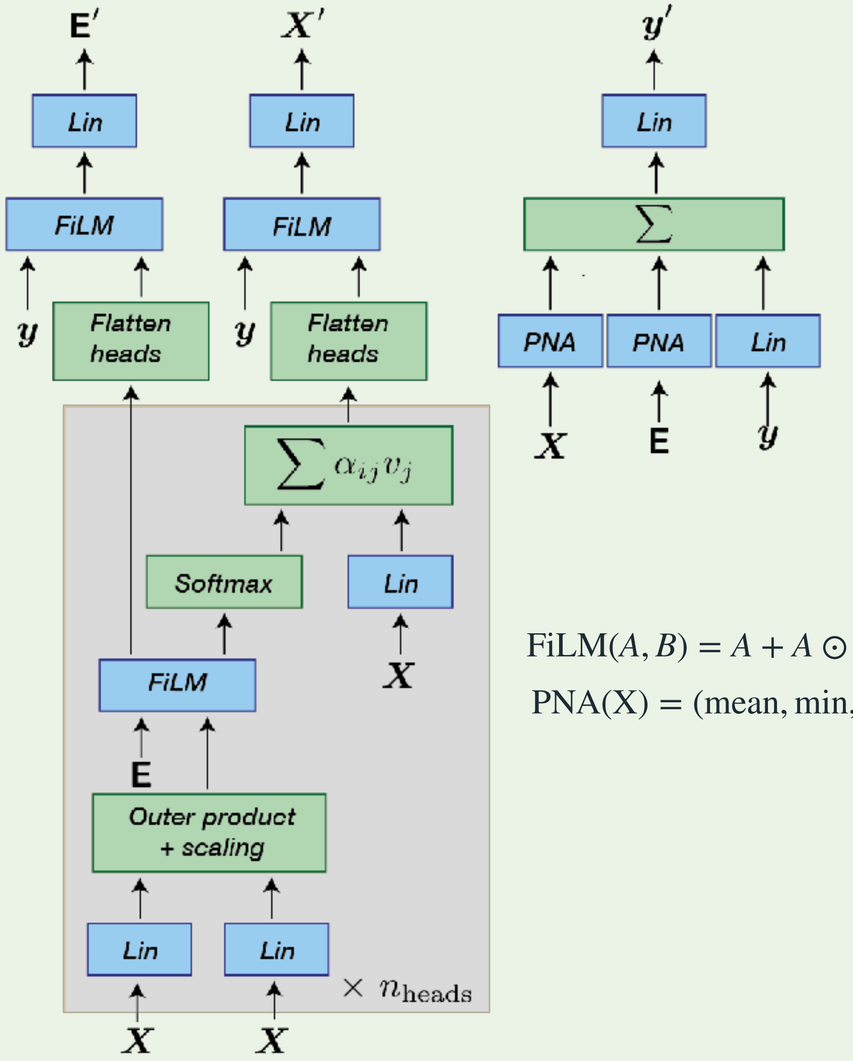
Motivation for graph transformers:

- Compute embeddings for the pairs for edge prediction
- Make the attention mechanism stable

Extends

Dwivedi & Bresson 20, A Generalization of Transformer Networks to Graphs

Self-attention layer



$$\mathrm{FiLM}(A, B) = A + A \odot (BW_1) + BW_2$$

$$\mathrm{PNA(X)} = (\mathrm{mean}, \mathrm{min}, \mathrm{max}, \mathrm{std})(X)$$

DiGress is permutation equivariant

It is made of permutation equivariant blocks

Its loss is permutation invariant

$$l(\hat{p}^G, G) = \sum_i \mathrm{CE}(x_i, \hat{p}_i^X) + \sum_{ij} \mathrm{CE}(e_{ij}, \hat{p}_{ij}^E)$$

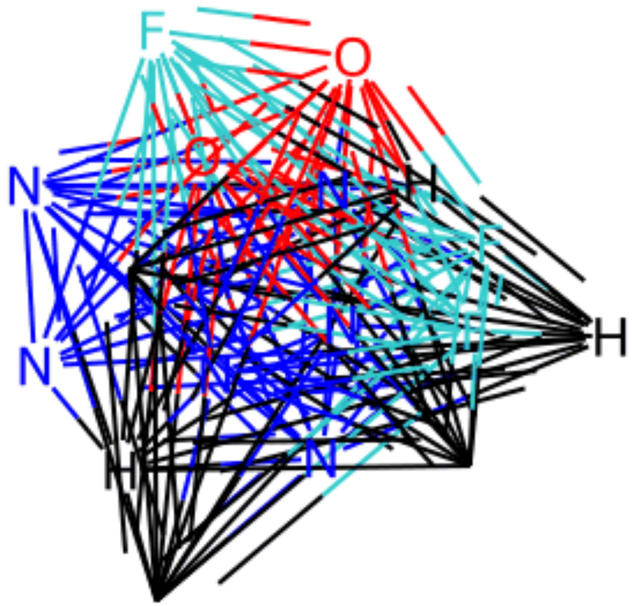
Do not depend on i, j

The model is exchangeable

All permutations of a generated graph **are equally likely**

Needed for likelihood computation

First results



Frame 0

Have we addressed the specificities of learning on graphs?

- Varying sizes 👍
- Equivariance properties 👍
- Sparsity 🤔
- Limited expressive power 🤔

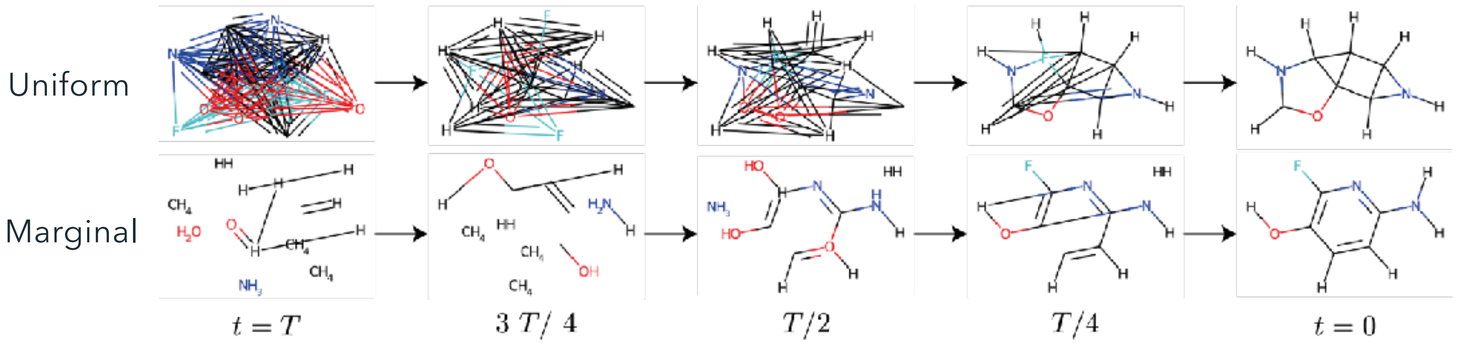
III Improving DiGress with marginal transitions and structural features

I. Choice of the noise model

Rather than **uniform transitions** over node and edge classes, choose a noise model that **preserves the marginal distribution of node and edge types**.

$$Q_X^t = \alpha^t I + \beta^t \ 1_a m_X'$$
$$Q_E^t = \alpha^t I + \beta^t \ 1_b m_E'$$

$$\begin{bmatrix} 0.8 & 0.1 & 0.1 \\ 0.2 & 0.7 & 0.1 \\ 0.2 & 0.1 & 0.7 \end{bmatrix}$$



II. Structural and spectral features

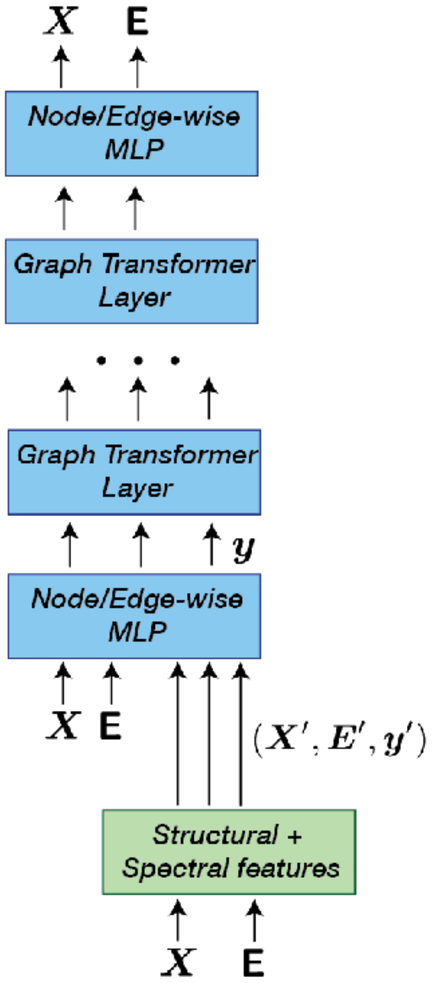
Message-passing neural networks have a **limited representation power** and cannot **detect substructures** such as cycles.

Solution 1: use a more expressive network

Very costly in practice

Solution 2: add features that the network cannot compute by itself

- Cycle counts
- Spectral features (# 0 eigenvalues, first non zero eigenvalues, first eigenvectors)



Xu et al. 18, How powerful are graph neural networks?
Bouritsas et al. 22, Improving graph neural network expressivity via subgraph isomorphism counting
Beaini et al. 21, Directional graph networks

Algorithms

Algorithm 1: Training DiGress

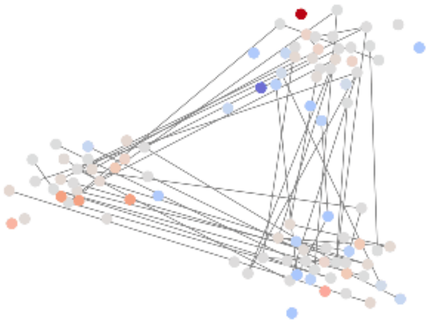
Input: A graph $G = (\boldsymbol{X}, \boldsymbol{E})$
Sample $t \sim \mathcal{U}(1, ..., T)$
Sample $G^t \sim \boldsymbol{X}\bar{\boldsymbol{Q}}^t_X \times \mathbf{E}\bar{\boldsymbol{Q}}^t_E$ ▷ Add noise
$z \leftarrow f(G^t, t)$ ▷ Structural and spectral features
$\hat{p}^X, \hat{p}^E \leftarrow \phi_\theta(G_t, z)$ ▷ Forward pass
$loss \leftarrow l_{CE}(\hat{p}^X, \boldsymbol{X}) + \lambda\, l_{CE}(\hat{p}^E, \mathbf{E})$
optimizer.step($loss$)

Algorithm 2: Sampling from DiGress

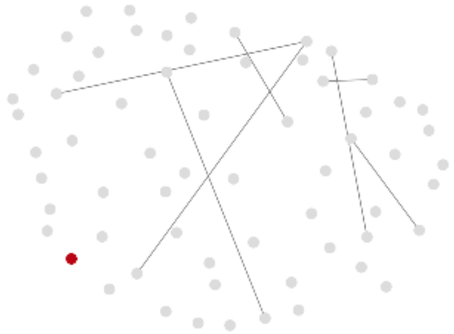
Sample n from the training data distribution
Sample $G^T \sim q_X(n) \times q_E(n)$ ▷ Random graph
for $t = T$ **to** 1 **do**
  $z \leftarrow f(G^t, t)$ ▷ Structural and spectral features
  $\hat{p}^X, \hat{p}^E \leftarrow \phi_\theta(G^t, z)$ ▷ Forward pass
  Sample $G^{t-1} \sim \prod_i p_\theta(x_i^{t-1}|G^t) \times \prod_{ij} p_\theta(e_{ij}^{t-1}|G^t)$ ▷ Reverse process
end
return G^0

Results

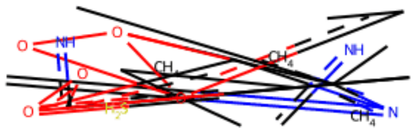
SBM



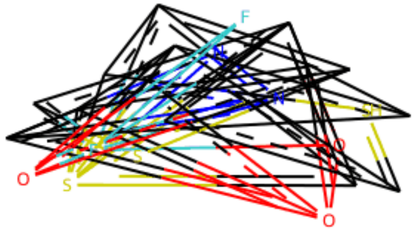
Planar



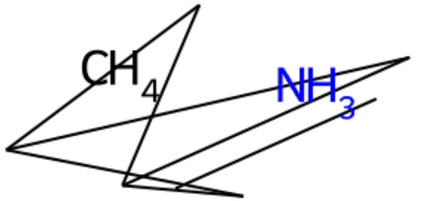
Guacamol



Moses



QM9



H_2O

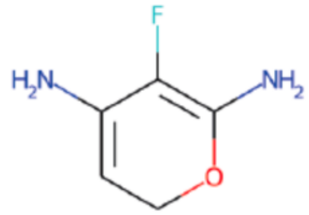
Qm9

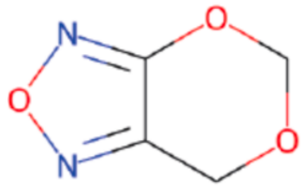
Implicit hydrogens

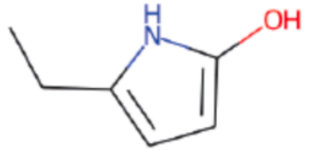
Method	NLL	Valid	Unique	Training time (h)
Dataset	–	99.3	100	–
Set2GraphVAE	–	59.9	93.8	–
SPECTRE	–	87.3	35.7	–
GraphNVP	–	83.1	99.2	–
GDSS	–	95.7	**98.5**	–
ConGress (ours)	–	98.9±.1	96.8±.2	7.2
DiGress (ours)	23.2±.0	**99.0**±.1	96.2±.1	**1.0**

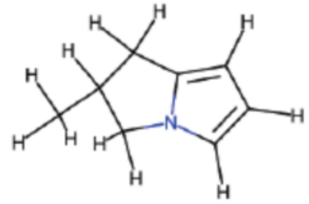
Explicit hydrogens

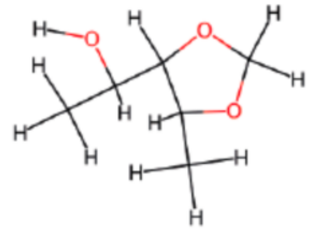
Model	Valid↑	Unique↑	Atom stable↑	Mol stable↑
Dataset	97.8	100	98.5	87.0
ConGress	86.7±1.8	**98.4**±0.1	97.2±0.2	69.5±1.6
DiGress (uniform)	89.8±1.2	97.8±0.2	97.3±0.1	70.5±2.1
DiGress (marginal)	92.3±2.5	97.9±0.2	**97.3**±0.8	66.8±11.8
DiGress (marg. + additional features)	**95.4**±1.1	97.6±0.4	**98.1**±0.3	**79.8**±5.6





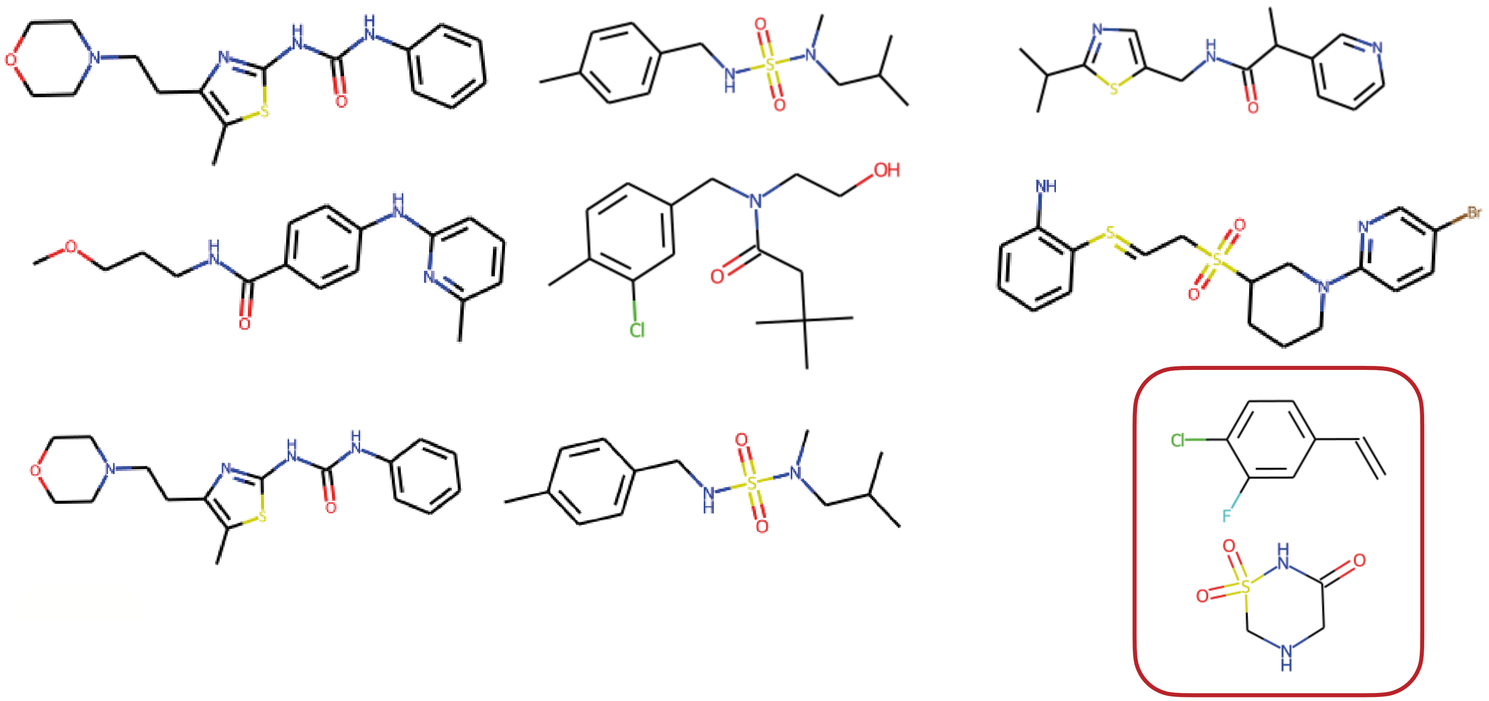






MOSES

- 1.9M molecules filtered from Zinc Clean Leads
- Results are reported on a test set of separate scaffolds

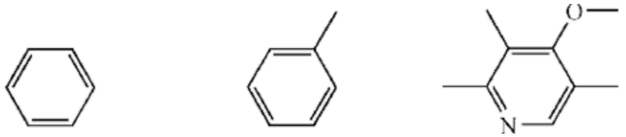


Model	Class	Val ↑	Unique↑	Novel↑	Filters↑	FCD↓	SNN↑	Scaf↑
VAE	SMILES	97.7	99.8	69.5	99.7	0.57	0.58	5.9
JT-VAE	Fragment	100	100	99.9	97.8	1.00	0.53	10
GraphINVENT	Autoreg.	96.4	99.8	–	95.0	1.22	0.54	12.7
ConGress (ours)	One-shot	83.4	99.9	96.4	94.8	1.48	0.50	16.4
DiGress (ours)	One-shot	85.7	100	95.0	97.1	1.19	0.52	14.8

Example of SMILES string:

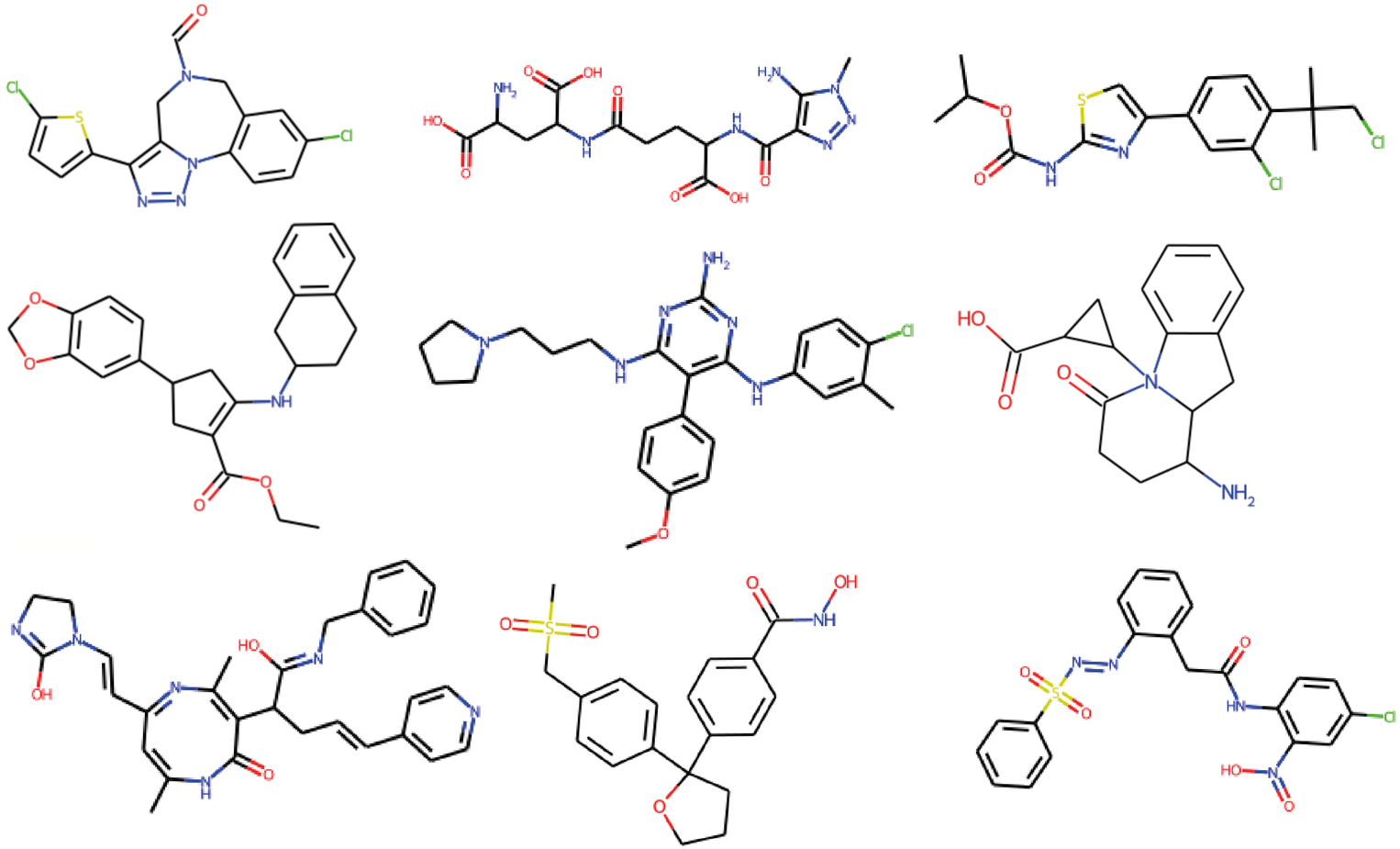
Cn1cnc2n(C)c(=O)n(C)c(=O)c12

Examples of fragments:



GuacaMol

- 1.3M molecules from ChEMBL 24
- Our model currently does not handle partial charges, which are present in the dataset



Model	Class	Valid↑	Unique↑	Novel↑	KL div↑	FCD↑
LSTM	Smiles	95.9	100	91.2	99.1	91.3
NAGVAE	One-shot	92.9	95.5	100	38.4	0.9
MCTS	One-shot	100	100	95.4	82.2	1.5
ConGress (ours)	One-shot	0.1	100	100	36.1	0.0
DiGress (ours)	One-shot	73.9	99.9	99.9	89.8	53.1

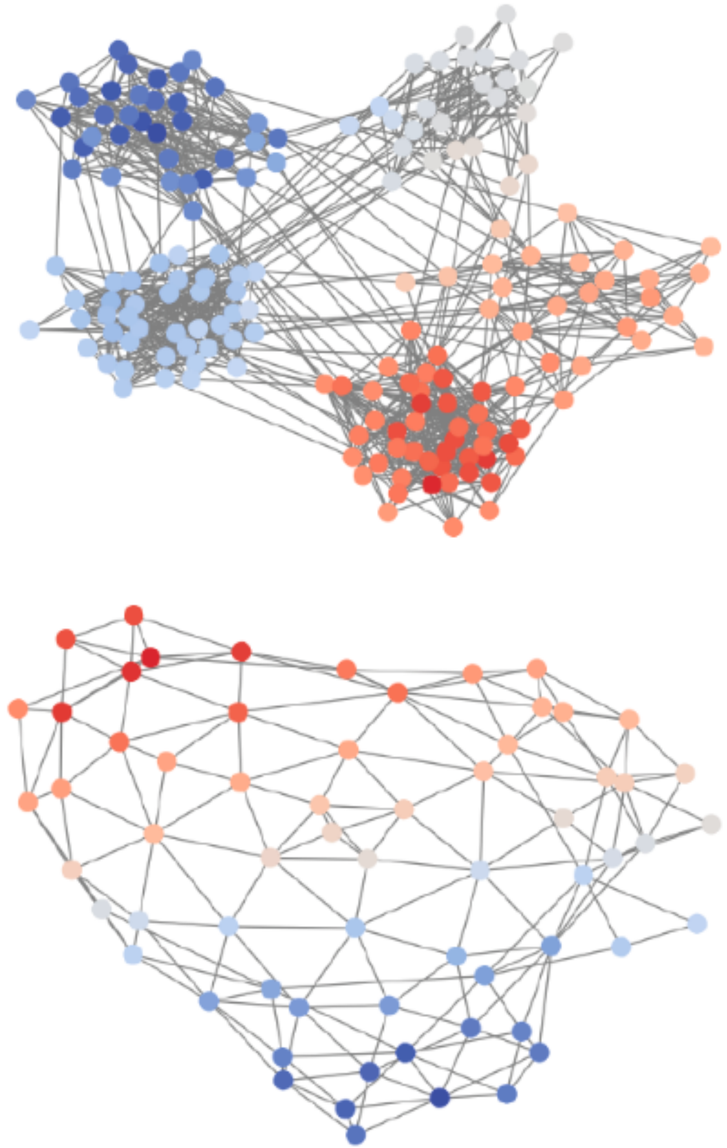
Abstract graphs

V.U.N: Valid, unique and novel graphs

Model	Deg ↓	Clus ↓	Orb↓	V.U.N. ↑
Stochastic block model				
GraphRNN	6.9	1.7	3.1	5 %
GRAN	14.1	1.7	2.1	25%
GG-GAN	4.4	2.1	2.3	25%
SPECTRE	1.9	1.6	**1.6**	53%
ConGress	34.1	3.1	4.5	0%
DiGress	**1.6**	**1.5**	1.7	**74%**
Planar graphs				
GraphRNN	24.5	9.0	2508	0%
GRAN	3.5	1.4	1.8	0%
SPECTRE	2.5	2.5	2.4	25%
ConGress	23.8	8.8	2590	0%
DiGress	**1.4**	**1.2**	**1.7**	**75%**

Could it be a graph drawn from SBM?

Is it a planar graph?



To measure generation quality, compare statistics from the generated graphs with statistics from a test set

Summary

- DiGress solves graph generation as a sequence of node and edge classification task
- Diffusion models for graphs significantly outperform previous methods – opens the way to many applications
- Discrete diffusion helps scaling to large graphs
- Limitation: $O(n^2)$ complexity

V What’s next?

Discrete regression guidance (proof-of-concept)

Goal: condition the generation on a vector y representing **graph-level properties**

- Train an unconditional model ϕ_θ
- Train a regressor g_η to predict y from G^t
- At each sampling step, use the gradients of g_η to push the generation towards graph that have property y

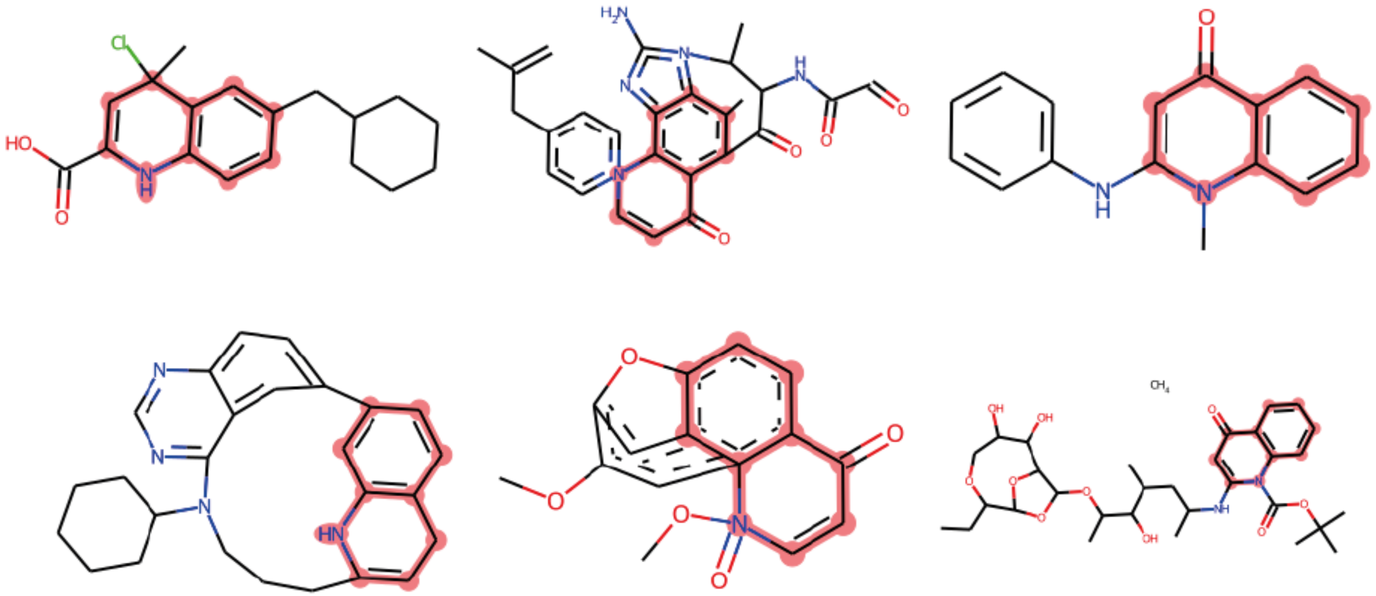
MAE between target and obtained property

Target	μ	HOMO	μ & HOMO
Uncondit.	$1.71_{\pm.04}$	$0.93_{\pm.01}$	$1.34_{\pm.01}$
Guidance	$0.81_{\pm.04}$	$0.56_{\pm.01}$	$0.87_{\pm.03}$

Guidance helps, but there is room for improvement

Sohl-Dickstein et al. 15, Deep Unsupervised Learning using Nonequilibrium Thermodynamics
Dhariwal & Nichol 21, Diffusion Models Beat GANs on Image Synthesis

Subgraph Extension



Long-scale consistency issues

A resampling strategy similar to **RePaint** could be used

Lugmayr et al 22, RePaint: Inpainting using Denoising Diffusion Probabilistic Models

Also…

- Mixed continuous-discrete generation
- Better tools for conditional generation, property optimization, fine-tuning…
- Is it possible to keep permutation equivariance without the $O(n^2)$ complexity?

Questions?

Clément Vignac
clement.vignac@epfl.ch

EPFL

Molecule generation in 3D

Unconditional generation in 3d

Example: EDM

Generate the 3d coordinates of atoms

Bonds are deduced from the atomic distances and the atom types (with a lookup table) -> not learnt

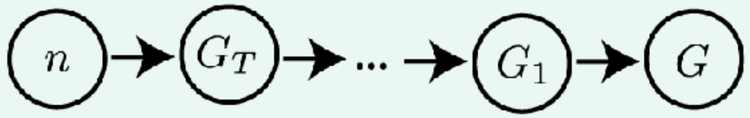
Conformer generation

Examples: Geodiff, Torsional diffusion

Take as input a graph structure, output 3d coordinates

Hoogeboom et al. 22, Equivariant diffusion for molecule generation in 3d
Xu et al. 22, GeoDiff: a Geometric Diffusion Model for Molecular Conformation Generation
Jing et al. 22, Torsional diffusion for molecular conformer generation

Likelihood computation



$$\log p_\theta(G) = \log \sum_{n \in \mathbb{N}} p(n) \int p(G^T \mid n)\; p_\theta(G^{t-1}, \ldots, G^1 \mid G^T)\; p_\theta(G \mid G^1)\; d(G^1, \ldots, G^T)$$

$$= \log p(n_G) + \log \int p(G^T \mid n_G) \prod_{t=2}^{T} p_\theta(G^{t-1} \mid G^t)\; p_\theta(G \mid G^1)\; d(G^1, \ldots, G^T)$$

$$\log p_\theta(G) \geq \log p(n_G) + \underbrace{D_{\mathrm{KL}}[q(G^T|G) \,||\, q_X(n_G) \times q_E(n_G)]}_{\text{Prior loss}} + \underbrace{\sum_{t=2}^{T} L_t(x)}_{\text{Diffusion loss}} + \underbrace{\mathbb{E}_{q(G^1|G)}[\log p_\theta(G|G^1)]}_{\text{Reconstruction loss}}$$

with $\quad L_t(G) = \mathbb{E}_{q(G^t|G)}\big[\mathrm{KL}[q(G^{t-1} \mid G^t, G) \,||\, p_\theta(G^{t-1} \mid G^t)]\big]$