中国人民大学
RENMIN UNIVERSITY OF CHINA

高瓴人工智能学院
Gaoling School of Artificial Intelligence

Equivariant Diffusion for Molecule Generation in 3D

Emiel Hoogeboom*, **Victor Garcia Satorras***, **Clement Vignac***, **Max Welling**

UvA–Bosch Delta Lab, University of Amsterdam, Netherlands

EPFL, Lausanne, Switzerland

(ICML 2022)

Fanmeng Wang

2023-5-24

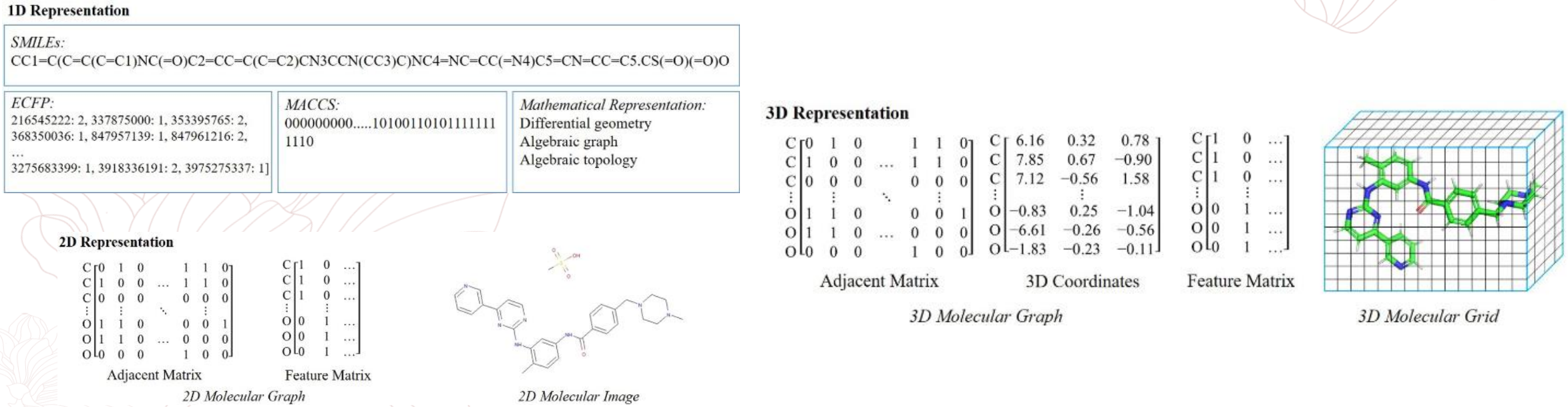
Outline

- Introduction
- Background: Diffusion Models & Equivariance
- Method: Equivariant Diffusion Models (EDMs)
- Experiment
- Conclusion

- Introduction
- Background: Diffusion Models & Equivariance
- Method: Equivariant Diffusion Models (EDMs)
- Experiment
- Conclusion

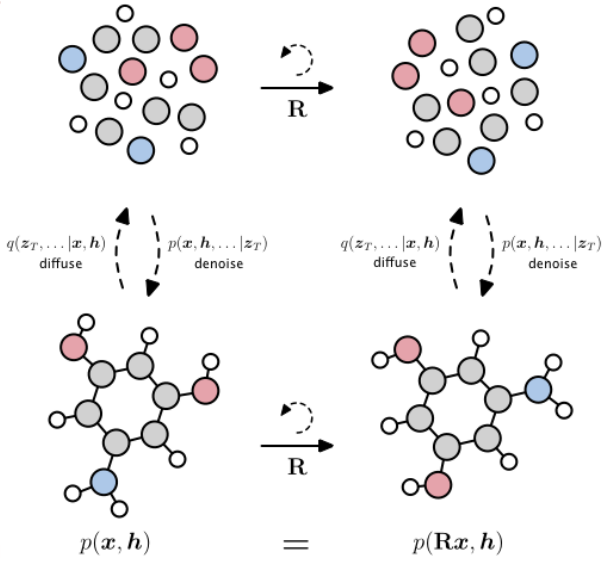
Introduction

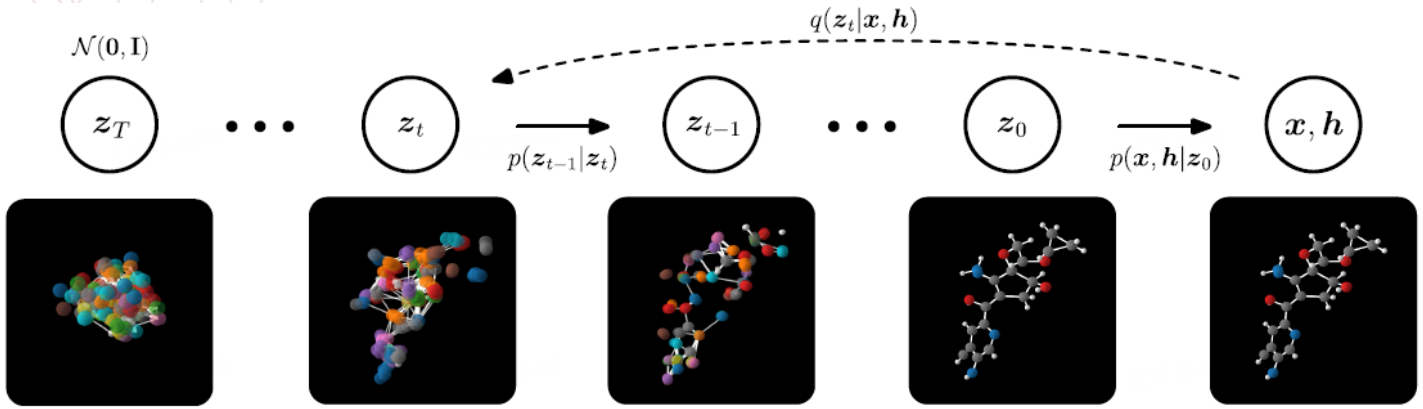
- **Molecular generation** aims to generate molecules with certain physicochemical and pharmacological properties.
- Considering the properties of molecules are mainly determined by the 3D structure of molecules, more and more works are trying to **generate molecules in 3D**.



Introduction

- In this paper, we introduce an **E(3) Equivariant Diffusion Models** (EDMs)
  - Denoises a diffusion process with an equivariant network to directly generates molecules in 3D
  - Operates on both **continuous** (atom coordinates) and **categorical** features (atom types) at the same time
  - Outperforms previous molecule generation models in log-likelihood and molecule stability



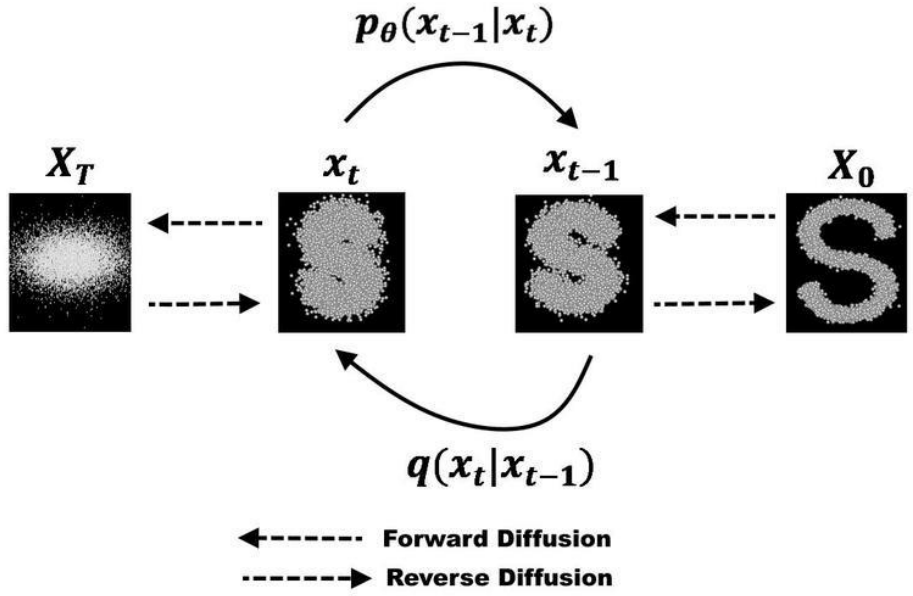


- Introduction
- Background: Diffusion Models & Equivariance
- Method: Equivariant Diffusion Models (EDMs)
- Experiment
- Conclusion

Background: Diffusion Models

■ Basic compositions

- Forward Diffusion Process： $X_0 \rightarrow X_T$ gradually convert the **original distribution** to **Standard Gaussian distribution**(or other prior distribution) by adding Gaussian noise.
- Reverse Diffusion Process： $X_T \rightarrow X_0$ gradually recover the **original distribution** from the **Standard Gaussian distribution** by denoising neural network.



Background: Diffusion Models

■ Key Points

- The Forward Diffusion Process is a **Markov chain** with fixed parameters that gradually convert the **original distribution** to the **Standard Gaussian distribution**.

$$q(x_{1:T}|x_0) = \frac{q(x_{0:T})}{q(x_0)} = \prod_{t=1}^{T} \frac{q(x_{0:t})}{q(x_{0:t-1})} = \prod_{t=1}^{T} q(x_t|x_{0:t-1}) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$

$$q(x_t|x_0) = \boldsymbol{N}(x_t; \sqrt{\overline{\alpha_t}}x_0, \sqrt{1-\overline{\alpha_t}}\,\boldsymbol{I})$$

- The Reverse Diffusion Process is also a **Markov chain**, trying to recover the **original distribution** from **Standard Gaussian distribution** by **denoising neural network**.

$$p_\theta(x_{0:T}) = p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$$

$$p_\theta(x_{t-1}|x_t) = N(x_{t-1}; \mu_\theta(x_t,t), \sum_\theta(x_t,t))$$

$$\mu_\theta(x_t,t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} z_\theta(x_t,t)\right)$$

In DDPM, we just use $\frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t}\beta_t$ to replace it

Noise predicted by denoising network

Background: Diffusion Models

- We can use a surrogate **variational lower bound** to train this model:

$$L = \mathbb{E}_q[-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1) + \sum_t D_{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) + D_{KL}(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T)]$$

then we can further get

$$L_t^{simple} = E_{x_0,z}[||z_t - z_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}z_t, t)||^2]$$

so we just need to minimize $||\bar{z}_t - z_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\bar{z}_t, t)||$ to train this model

- The key of diffusion model is to **design a neural network to predict noise** based on the current x_t.

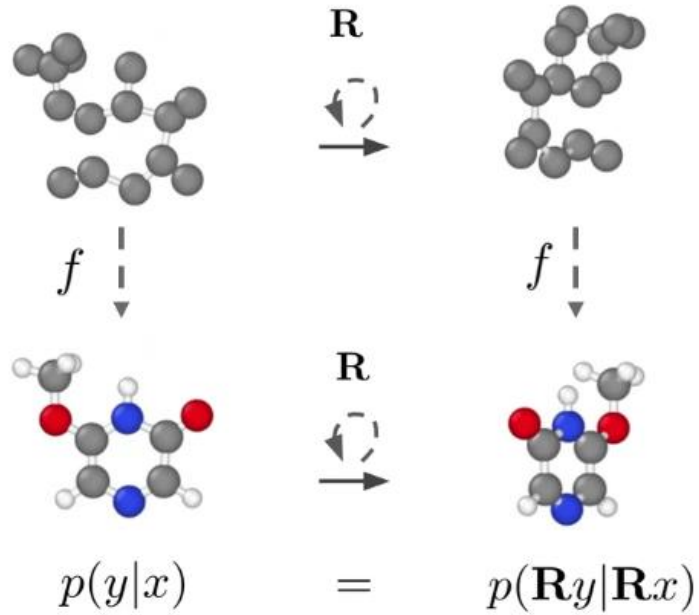
Background: Equivariance

- Molecules live in the physical 3D space, and as such are subject to geometric **symmetries** such as **translations, rotations, and possibly reflections**.
- A function is **equivariant** if

$$\mathbf{R}f(\boldsymbol{x}) = f(\mathbf{R}\boldsymbol{x})$$

- A conditional distribution is **equivariant** if

$$p(y|x) = p(\mathbf{R}y|\mathbf{R}x)$$



invariant distribution + **equivariant** invertible function = **invariant** distribution

- Introduction
- Background: Diffusion Models & Equivariance
- Method: Equivariant Diffusion Models (EDMs)
- Experiment
- Conclusion

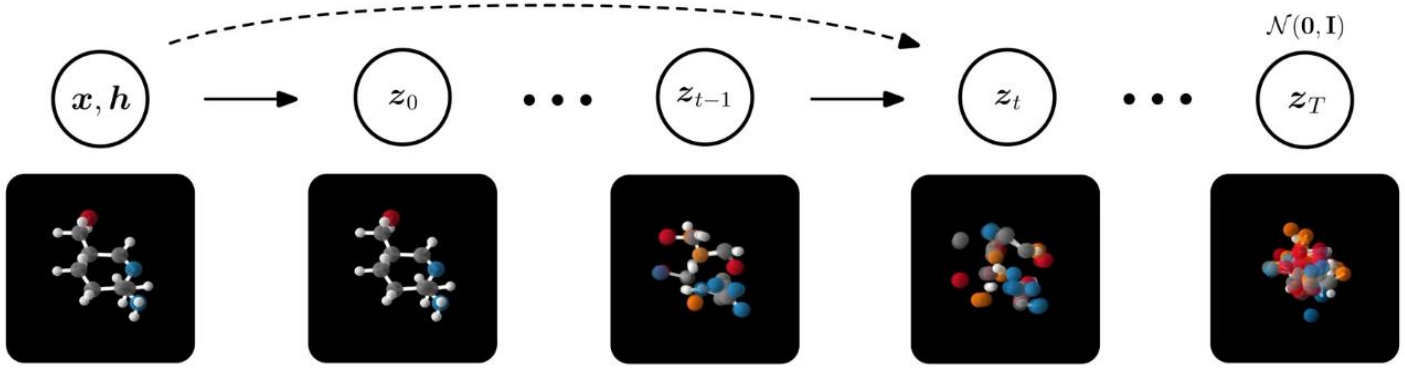
Method: EDMs

■ Method design

- We define a 3D molecule as **a point cloud** with coordinates $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M) \in \mathbb{R}^{M \times 3}$ and corresponding features $\boldsymbol{h} = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_M) \in \mathbb{R}^{M \times \mathrm{nf}}$
- **Diffusion Process**
  - Adds Gaussian noise over time steps t = 0,…,T

$$q(\boldsymbol{z}_t|\boldsymbol{x}, \boldsymbol{h}) = \mathcal{N}_{xh}(\boldsymbol{z}_t|\alpha_t[\boldsymbol{x}, \boldsymbol{h}], \sigma_t^2\mathbf{I})$$

$$\mathcal{N}_x(z_t^{(x)}|\alpha_t \boldsymbol{x}, \sigma_t^2\mathbf{I}) \cdot \mathcal{N}(z_t^{(h)}|\alpha_t \boldsymbol{h}, \sigma_t^2\mathbf{I})$$



Method: EDMs

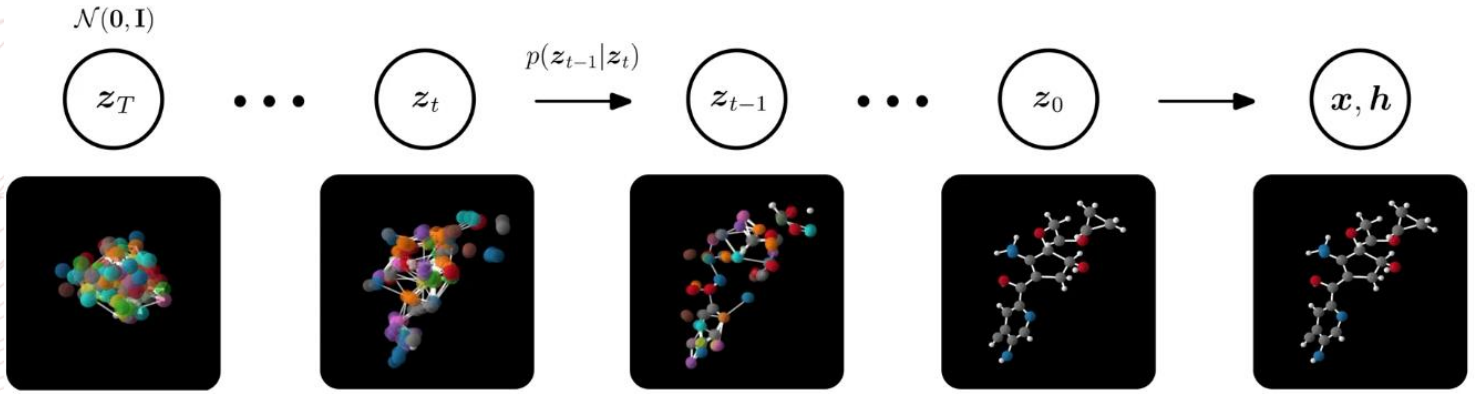
■ Method design

➢ Reverse Denoising / Generating Process

- Denoise with distribution

$$p(\boldsymbol{z}_s|\boldsymbol{z}_t) = \mathcal{N}_{xh}(\boldsymbol{z}_s|\boldsymbol{\mu}_{t\to s}([\hat{\boldsymbol{x}}, \hat{\boldsymbol{h}}], \boldsymbol{z}_t), \sigma^2_{t\to s}\mathbf{I})$$

$$[\hat{\boldsymbol{x}}, \hat{\boldsymbol{h}}] = \boldsymbol{z}_t/\alpha_t - \hat{\boldsymbol{\epsilon}}_t \cdot \sigma_t/\alpha_t$$



Method: EDMs

■ Method design

➢ Optimization

$$\mathcal{L}_t = -\mathrm{KL}(q(z_s|x, z_t)||p(z_s|z_t))$$

$$\mathcal{L}_t = \mathbb{E}_{\epsilon_t \sim \mathcal{N}_{xh}(0,\,\mathbf{I})}\left[\frac{1}{2}\, w(t)\, \left||\epsilon_t - \hat{\epsilon}_t\right||^2\right]$$

$$\begin{aligned}\hat{\epsilon}_t &= \phi(z_t, t) \\ &= \phi(z_t^{(x)}, z_t^{(h)}, t) \\ &= \mathrm{EGNN}(z_t^{(x)}, [z_t^{(h)}, t/T]) - [z_t^{(x)}, \mathbf{0}]\end{aligned}$$

- Here, we use **EGNN** to predict noise based on current state z_t
- The l-th layer in EGNN can be defines as:

$$\mathbf{m}_{ij} = \phi_e\left(\boldsymbol{h}_i^l, \boldsymbol{h}_j^l, d_{ij}^2, a_{ij}\right), \boldsymbol{h}_i^{l+1} = \phi_h(\boldsymbol{h}_i^l, \sum_{j \neq i} \tilde{e}_{ij}\mathbf{m}_{ij})$$

$$\boldsymbol{x}_i^{l+1} = \boldsymbol{x}_i^l + \sum_{j \neq i} \frac{\boldsymbol{x}_i^l - \boldsymbol{x}_j^l}{d_{ij}+1} \phi_x\left(\boldsymbol{h}_i^l, \boldsymbol{h}_j^l, d_{ij}^2, a_{ij}\right),$$

Method: EDMs

■ Method design

➢ Related algorithms

Algorithm 1 Optimizing EDM

Input: Data point x, neural network ϕ
Sample $t \sim \mathcal{U}(0, \ldots, T)$, $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
Subtract center of gravity from $\epsilon^{(x)}$ in $\epsilon = [\epsilon^{(x)}, \epsilon^{(h)}]$
Compute $z_t = \alpha_t[\boldsymbol{x}, \boldsymbol{h}] + \sigma_t \epsilon$
Minimize $||\epsilon - \phi(\boldsymbol{z}_t, t)||^2$

Algorithm 2 Sampling from EDM

Sample $z_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for t in $T, T-1, \ldots, 1$ where $s = t - 1$ **do**
  Sample $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  Subtract center of gravity from $\epsilon^{(x)}$ in $\epsilon = [\epsilon^{(x)}, \epsilon^{(h)}]$
  $z_s = \frac{1}{\alpha_{t|s}} z_t - \frac{\sigma^2_{t|s}}{\alpha_{t|s}\sigma_t} \cdot \phi(\boldsymbol{z}_t, t) + \sigma_{t \to s} \cdot \epsilon$
end for
Sample $\boldsymbol{x}, \boldsymbol{h} \sim p(\boldsymbol{x}, \boldsymbol{h} | \boldsymbol{z}_0)$

To make sure initial density invariant

Method: EDMs

■ Method design

➢ Number of atoms

- We compute the categorical distribution p(M) of molecule sizes on the training set.
- To sample molecules from the model p(x;h;M), we first sample M from p(M) and then sample x, h from p(x,h | M)

➢ Extension: Conditional generation

- Given the desired property c, we just takes it as additional input for noise prediction

$$\hat{\epsilon}_t = \phi(z_t, t) \longrightarrow \hat{\epsilon}_t = \phi(z_t, [t, c])$$

- Introduction
- Background: Diffusion Models & Equivariance
- Method: Equivariant Diffusion Models (EDMs)
- **Experiment**
- Conclusion

Experiment

■ Molecule Generation

Table 1. Neg. log-likelihood $-\log p(\mathbf{x}, \mathbf{h}, M)$, atom stability and molecule stability with standard deviations across 3 runs on QM9, each drawing 10000 samples from the model.

# Metrics	NLL	Atom stable (%)	Mol stable (%)
E-NF	-59.7	85.0	4.9
G-Schnet	N.A	95.7	68.1
GDM	-94.7	97.0	63.2
GDM-aug	-92.5	97.6	71.6
EDM (ours)	**-110.7**$\pm$1.5	**98.7**$\pm$0.1	**82.0**$\pm$0.4
Data		99.0	95.2

Table 2. Validity and uniqueness over 10000 molecules with standard deviation across 3 runs. Results marked (*) are not directly comparable, as they do not use 3D coordinates to derive bonds. H: model hydrogens explicitly

Method	H	Valid (%)	Valid and Unique (%)
Graph VAE (*)		55.7	42.3
GTVAE (*)		74.6	16.8
Set2GraphVAE (*)		59.9$\pm$1.7	56.2$\pm$1.4
EDM (ours)		**97.5**$\pm$0.2	**94.3**$\pm$0.2
E-NF	✓	40.2	39.4
G-Schnet	✓	85.5	80.3
GDM-aug	✓	90.4	89.5
EDM (ours)	✓	**91.9**$\pm$0.5	**90.7**$\pm$0.6
Data	✓	97.7	97.7

- **atom stability** (the proportion of atoms that have the right valency)
- **molecule stability** (the proportion of generated molecules for which all atoms are stable)
- **validity** (whether can be recognised by RDKit)

EDM is able to generate a high rate of **valid** and **unique** molecules with high **stability**

Experiment

■ Conditional Molecule Generation

properties α, gap, homo, lumo, μ and C_v

Table 3. Mean Absolute Error for molecular property prediction by a EGNN classifier ϕ_c on a QM9 subset, EDM generated samples and two different baselines "Naive (U-bounds)" and "# Atoms".

Task	α	$\Delta\varepsilon$	$\varepsilon_{\text{HOMO}}$	$\varepsilon_{\text{LUMO}}$	μ	C_v
Units	Bohr^3	meV	meV	meV	D	$\frac{\text{cal}}{\text{mol}}$K
Naive (U-bound)	9.01	1470	645	1457	1.616	6.857
#Atoms	3.86	866	426	813	1.053	1.971
EDM	2.76	655	356	584	1.111	1.101
QM9 (L-bound)	0.10	64	39	36	0.043	0.040

- EDM outperforms both "Naive (U-bound)" and "#Atoms" baselines in all properties (except μ).
- There is still room for improvement by looking at the gap between "EDMs" and "QM9 (L-bound)".

- Introduction
- Background: Diffusion Models & Equivariance
- Method: Equivariant Diffusion Models (EDMs)
- **Experiment**
- Conclusion

Conclusion

- We presented EDMs, an E(3) equivariant diffusion model for molecule generation in 3D.
- EDMs can generate a high rate of valid and unique molecules with high stability.
- Besides, EDMs can alao generate molecules with desired properties.

A key problem:

- This work just directly lift discrete attributes into continuous space without using some discrete diffusion model.

Equivariant Diffusion for Molecule Generation in 3D

Emiel Hoogeboom [*1] Victor Garcia Satorras [*1] Clément Vignac [*2] Max Welling [1]

ICML 2022

DIGRESS: DISCRETE DENOISING DIFFUSION FOR GRAPH GENERATION

Clément Vignac*
LTS4, EPFL
Lausanne, Switzerland

Igor Krawczuk*
LIONS, EPFL
Lausanne, Switzerland

Antoine Siraudin
LTS4, EPFL
Lausanne, Switzerland

Bohan Wang
LTS4, EPFL
Lausanne, Switzerland

Volkan Cevher
LIONS, EPFL
Lausanne, Switzerland

Pascal Frossard
LTS4, EPFL
Lausanne, Switzerland

ICLR 2023

Thank You for listening!

Fanmeng Wang

2023-5-24

中国人民大学
RENMIN UNIVERSITY OF CHINA

高瓴人工智能学院
Gaoling School of Artificial Intelligence

DiGress: Discrete Denoising diffusion for graph generation

Clement Vignac*, Igor Krawczuk*, Antoine Siraudin, Bohan Wang,
Volkan Cevher, Pascal Frossard

LTS4, EPFL
Lausanne, Switzerland

(ICLR 2023)

Fanmeng Wang

2022-11-18

Outline

- Introduction
- DiGress: Discrete Denoising diffusion for graph generation
- Improving DiGress: Some algorithmic improvements to DiGress
- Experiments
- Conclusion

- Introduction
- DiGress: Discrete Denoising diffusion for graph generation
- Improving DiGress: Some algorithmic improvements to DiGress
- Experiments
- Conclusion

Introduction

Diffusion Models

- A powerful class of generative models
- They have been used sucessfully in various settings, outperforming all other methods on image and video data.

Graph Generation

- A task that has applications as diverse as molecule design, traffic modeling and code completion.
- It remains challenging due to the unordered nature and sparsity properties of graph structures.

Diffusion Models + Graph Generation

Introduction

■ Existing diffusion models for graph generation

- These models embed the graphs in a **continuous space** and add **Gaussian noise** to the node features and graph adjacency matrix
- This destroys the **graph's sparsity** and creates **fully connected noisy graphs** for which structural information (such as connectivity or cycle counts) is not defined.

1	0	1
0	1	1
1	1	1

add Gaussian noise →

1.1	0.1	1.1
0.1	1.3	0.9
1.1	0.9	1.2

- As a result, **continuous diffusion** can make it hard for the denoising network to **capture the structural properties** of the data.

Continuous diffusion vs **Discrete diffusion**

Introduction

- **DiGress:A discrete denoising diffusion model for generating graphs with categorical node and edge attributes**
  - **The diffusion process** is a Markov process that consists of successive graphs edits (edge addition or deletion, node or edge category edit) that can occur independently on each node or edge.
  - In order to **reverse this diffusion process**, we train a **graph transformer network** to predict the clean graph from a noisy input.
  - The resulting architecture is **permutation equivariant** and admits an evidence lower bound for likelihood estimation.

Introduction

- **Improving DiGress: Some algorithmic improvements to DiGress**
  - We use a noise model that **preserves the marginal distribution** of node and edge types during diffusion rather than uniform noise.
  - We **augment the input** of our denoising network with **structural and spectral features** at each diffusion step.
  - We introduce **a novel guidance procedure** for **conditioning graph generation** on graph-level properties, which is a key requirement for many applications

Experiments show that DiGress is **state-of-the-art** on several benchmarks, it outperforms existing one-shot generation methods and is able to scale to larger datasets, reaching the performance of autogressive models on molecule generation

- Introduction
- DiGress: Discrete Denoising diffusion for graph generation
- Improving DiGress: Some algorithmic improvements to DiGress
- Experiments
- Conclusion

Methodology: DiGress

Diffusion models

- They are defined by two components: **the diffusion process** achieved by a noise model and **the reverse diffusion process** achieved by a denoising neural network.
- **The diffusion process** achieved by a noise model q takes as input a data point x and progressively corrupts it to create noisy data points $(z^1, \ldots, z^t, \ldots, z^T)$, which together form **a trajectory of increasingly noisy data**. Besides, it is a Markov process that

$$q(z^1, \ldots, z^t, \ldots, z^T|x) = q(z^1|x) \prod_{t=2}^{T} q(z^t|z^{t-1})$$

- **The reverse diffusion process** achieved by a denoising network $\emptyset_\theta$ learns to **reverse these trajectories**: it takes as input a noisy state z^t and predicts $\hat{z}^{t-1} = \emptyset_\theta(z^t)$

Methodology: DiGress

■ Discrete Diffusion models (Austin et al. 2021)

- A data point x that belongs to one of d classes and $\boldsymbol{x} \in R^d$ is its one-hot encoding.
- The noise is now represented by **transition matrices** $(\boldsymbol{Q^1}, \ldots, \boldsymbol{Q^t}, \ldots, \boldsymbol{Q^T})$ such that $[\boldsymbol{Q^t}]_{ij}$ represents the probability of jumping from state i to state j:

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

and

$$q(z^t|x) = \boldsymbol{x}\overline{\boldsymbol{Q^t}}$$

where

$$\overline{\boldsymbol{Q^t}} = \boldsymbol{Q^1}\,\boldsymbol{Q^2}\ldots\boldsymbol{Q^t}$$

Continuous diffusion model

$$q(z^t|z^{t-1}) = \boldsymbol{N}(z^t; \sqrt{1-\beta_t}z^{t-1}, \beta_t\boldsymbol{I})$$

Methodology: DiGress

- **Discrete Diffusion models (Austin et al. 2021)**
  - Then the **posterior distribution** $q(z^{t-1}|z^t, x)$ can also be computed in closed-form **using Bayes** rule:

$$q(z^{t-1}|z^t, x) \propto z^t \, (Q^t)' \odot x \, \bar{Q}^{t-1}$$

where $\odot$ denotes a pointwise product and Q' is the transpose of Q.

  - The limit distribution of the noise model depends on the **transition matrices**. The simplest and most common one is a **uniform transition** parametrized by

$$Q^t = \alpha^t I + (1 - \alpha^t) 1_d 1_d' / d$$

with α^t transitioning from 1 to 0, When $lim_{t\to\infty} \alpha^t = 0$, $q(z^t|x)$ converges to a **uniform distribution** independently of x

Methodology: DiGress

- **Discrete Diffusion models (Austin et al. 2021)**

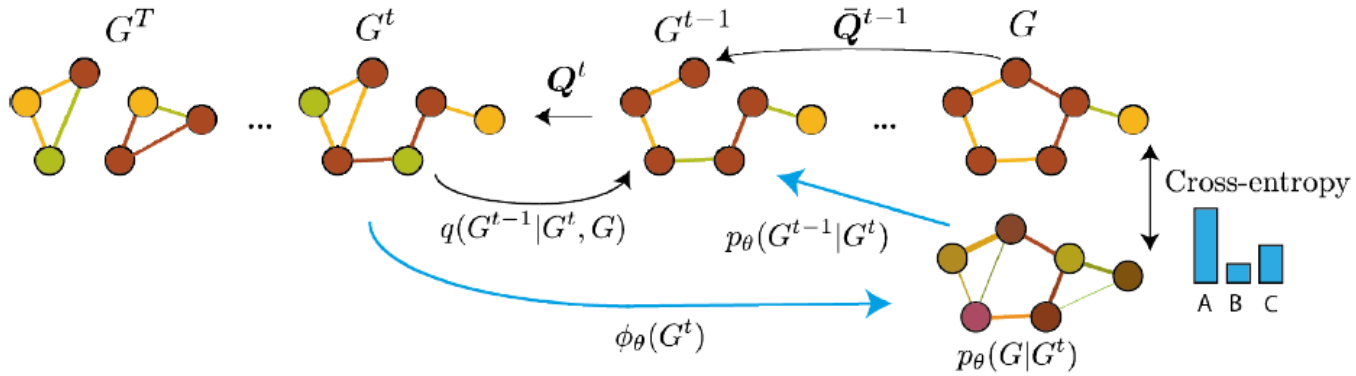


Figure 1: Overview of the model. The noise model is defined by Markov transition matrices Q^t whose product is $\bar{Q}^t$. The denoising neural networks ϕ_θ learns to predict the clean output from G^t. The predicted distribution is combined with $q(G^{t-1}, G \mid G^t)$ in order to sample G^{t-1}.

- However, while this framework has been applied successfully to several data modalities, **graphs have unique challenges** that need to be considered: they have **varying sizes**, **permutation equivariance properties**, and to this date no known tractable universal approximator.
- Therefore, we **propose a new discrete diffusion model (DiGress)** that addresses the specific challenges of graph generation.

Methodology: DiGress

■ DiGress: a new discrete diffusion model for graph generation

We consider graphs with **categorical node and edge attributes**.

➢ **Basic Definitions**

- a is the number of node types and b is the number of edge types.
- x_i denotes the attribute of node i and $\boldsymbol{x_i} \in \boldsymbol{R^a}$ is its one-hot encoding. These vectors are grouped in a matrix $\boldsymbol{X} \in \boldsymbol{R^{n \times a}}$.
- Similarly, a tensor $\boldsymbol{E} \in \boldsymbol{R^{n \times n \times b}}$ groups the one-hot encoding $\boldsymbol{e_{ij}}$ of each edge. Besides, we simply treat **the absence of edge** as a particular edge type.

➢ We diffuse **separately on each node and edge feature**. In other words, the **state-space** that we consider is not the one of graphs, but only the $\boldsymbol{a}$ **node types and** $\boldsymbol{b}$ **edge types**.

Methodology: DiGress

■ DiGress: a new discrete diffusion model for graph generation

We consider graphs with **categorical node and edge attributes**.

➢ Diffusion process

- For any node (resp. edge), the **transition probabilities** are defined by the matrices:

$$[Q_X^t]_{ij} = q(x^t = j \mid x^{t-1} = i) \quad [Q_E^t]_{ij} = q(e^t = j \mid e^{t-1} = i)$$

- Therefore, adding noise to form $G^t = (\boldsymbol{X^t}, \boldsymbol{E^t})$ means **sampling each node and edge type** from a categorical distribution defined by the following probabilities:

$$q(G^t|G^{t-1}) = (\boldsymbol{X}^{t-1}\boldsymbol{Q}_X^t, \mathbf{E}^{t-1}\boldsymbol{Q}_E^t) \quad \text{and} \quad q(G^t|G) = (\boldsymbol{X}\bar{\boldsymbol{Q}}_X^t, \mathbf{E}\bar{\boldsymbol{Q}}_E^t)$$

$$\overline{\boldsymbol{Q^t}} = \boldsymbol{Q^1}\,\boldsymbol{Q^2}\ldots\boldsymbol{Q^t}$$

Methodology: DiGress

- **DiGress: a new discrete diffusion model for graph generation**

We consider graphs with **categorical node and edge attributes**.

➢ **Reverse diffusion process achieved by denoising neural network $\emptyset_\theta$**

- The **denoising neural network** $\emptyset_\theta$ takes as input a noisy graph $G^t = (\boldsymbol{X^t}, \boldsymbol{E^t})$ and outputs tensors $\boldsymbol{X'}$ and $\boldsymbol{E'}$ (the predicted distribution of clean graphs).
- Internally, our layers also manipulate **graph-level features** y, as they are able to store information more efficiently.
- We use the **graph transformer network** proposed by (Dwivedi & Bresson 2021) to predict the clean graph G from a noisy graph G^t.
- 注意，与continuous diffusion model不同的是，这里的**denoising neural network** $\emptyset_\theta$ 以 noisy graph G^t 为输入，被用来预测真实的图G，而不是预测噪声Q^t

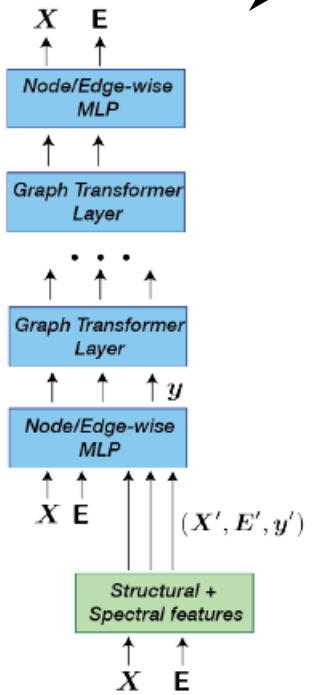


Figure 5: Architecture of the denoising network.

Methodology: DiGress

■ DiGress: a new discrete diffusion model for graph generation

We consider graphs with **categorical node and edge attributes**.

➢ Inverse denoising iterations by denoising neural network $\emptyset_\theta$

- The **denoising neural network** $\emptyset_\theta$ takes as input a noisy graph $G^t = (\boldsymbol{X^t}, \boldsymbol{E^t})$ and outputs tensors $\boldsymbol{X'}$ and $\boldsymbol{E'}$ (the predicted distribution of clean graphs).
- To **train** it, we optimize the cross-entropy l between the predicted probabilities $\hat{p}^G = (\hat{p}^X, \hat{p}^E) = (X', E')$ for each node and edge and the true graph G:

$$l(\hat{p}^G, G) = \sum_{1 \le i \le n} \text{cross-entropy}(x_i, \hat{p}_i^X) + \lambda \sum_{1 \le i,j \le n} \text{cross-entropy}(e_{ij}, \hat{p}_{ij}^E)$$

where $\lambda \in R^+$ is used to control the relative importance of nodes and edges.

Methodology: DiGress

DiGress: a new discrete diffusion model for graph generation

We consider graphs with **categorical node and edge attributes**.

Inverse denoising iterations by denoising neural network $\emptyset_\theta$

- The **denoising neural network** $\emptyset_\theta$ takes as input a noisy graph $G^t = (\boldsymbol{X^t}, \boldsymbol{E^t})$ and outputs tensors $\boldsymbol{X'}$ and $\boldsymbol{E'}$ (predicted distribution of clean graphs).
- We **model the distribution** as a product over nodes and edges:

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

and

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

$$p_\theta(e_{ij}^{t-1}|e_{ij}^t) = \textstyle\sum_{e\in\mathcal{E}} p_\theta(e_{ij}^{t-1} \mid e_{ij} = e, e_{ij}^t)\, \hat{p}_{ij}^E(e)$$

where $\hat{p}^G = (\hat{p}^X, \hat{p}^E) = (X', E')$

- Introduction
- DiGress: Discrete Denoising diffusion for graph generation
- Improving DiGress: Some algorithmic improvements to DiGress
- Experiments
- Conclusion

Methodology: Improving DiGress

- **Choice of the noise model(i.e. Markov transition matrices)**
  - The choice of the **Markov transition matrices** $(Q_t)_{t\leq T}$ defining the graph edit probabilities is arbitrary**,** and it is a priori not clear what noise model will lead to the best performance.
  - The most common noise model is a **uniform transition** over the classes

$$\boldsymbol{Q}^t = \alpha^t \boldsymbol{I} + (1 - \alpha^t)\mathbf{1}_d \mathbf{1}_d'/d$$

  - ***Hypothesis****: training is easier when the prior distribution is close to the true data distribution*
  - In this case, we choose noise model that satisfy:

$$\boldsymbol{Q}_X^t = \alpha^t \boldsymbol{I} + \beta^t \, \mathbf{1}_a \boldsymbol{m}_X' \quad \text{and} \quad \boldsymbol{Q}_E^t = \alpha^t \boldsymbol{I} + \beta^t \, \mathbf{1}_b \boldsymbol{m}_E'$$

  - We verify experimentally that these marginal transitions improves over uniform transitions.

Methodology: Improving DiGress

Structural features augmentation

- Generative models for graphs inherit the limitations of graph neural networks, and in particular their limited representation power.
- Therefore, we can **augment** standard message passing networks with features that they cannot compute by themselves.
- DiGress operates on a discrete space and takes noisy graphs as input, so we can therefore compute **various graph descriptors** at each diffusion step, and input them to the network to help it denoise the graphs.

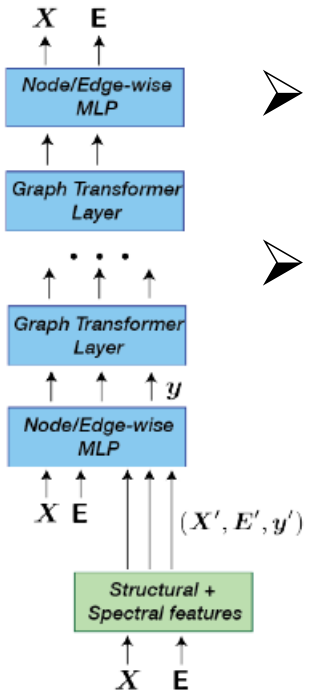


Figure 5: Architecture of the denoising network.

Methodology: Improving DiGress

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

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

$$p_\theta(e_{ij}^{t-1}|e_{ij}^t) = \textstyle\sum_{e \in \mathcal{E}} p_\theta(e_{ij}^{t-1} \mid e_{ij} = e, e_{ij}^t)\, \hat{p}_{ij}^E(e)$$

Methodology: Improving DiGress

Conditional Generation

- While good unconditional generation is a prerequisite, the ability to **condition generation** on several **graph-level properties** is key to many applications.
- One direct way to perform conditional generation is to **train the denoising network using the target properties**, but it requires to retrain the model when the conditioning properties changes.
- Therefore we build upon the **classifier guidance algorithm** (Sohl-Dickstein et al., 2015) and propose a new discrete guidance scheme.
- Our method uses a **regressor** g_η which is trained to predict target properties y of a clean graph G from a noisy graph G^t:

$$g_\eta(G^t) \approx y(G)$$

Methodology: Improving DiGress

■ Conditional Generation

➢ Conditional reverse noising process (Dhariwal & Nichol, 2021)

Denote $\dot{q}$ the noising process conditioned on y, q the unconditional noising process

We assume $$\dot{q}(G^t|G, \boldsymbol{y}) = \dot{q}(G^t|G)$$

Then $$\dot{q}(G^{t-1}|G^t, \boldsymbol{y}) \propto q(G^{t-1}|G^t)\, \dot{q}(\boldsymbol{y}|G^{t-1})$$

$$\begin{aligned}\log \dot{q}(\boldsymbol{y}|G^{t-1}) &\approx \log \dot{q}(\boldsymbol{y}|G^t) + \langle \nabla_G \log \dot{q}(\boldsymbol{y}|G^t), G^{t-1} - G^t \rangle \\ &\approx c(G^t) + \sum_{1 \le i \le n} \langle \nabla_{x_i} \log \dot{q}(\boldsymbol{y}|G^t), \boldsymbol{x}_i^{t-1} \rangle + \sum_{1 \le i,j \le n} \langle \nabla_{e_{ij}} \log \dot{q}(\boldsymbol{y}|G^t), \boldsymbol{e}_{ij}^{t-1} \rangle\end{aligned}$$

Algorithm 3: Sampling from DiGress with discrete regressor guidance.

Input: Diffusion model ϕ_θ, graph properties regressor g, guidance scale λ, graph size n

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
- $\hat{y} \leftarrow g_\eta(G^t)$ ▷ Regressor model
- $p_\eta(\hat{y}|G^{t-1}) \propto \exp(-\lambda \langle \nabla_{G^t} ||g_\eta(G^t) - \hat{y}||^2, G^{t-1} \rangle)$ ▷ Guidance distribution
- Sample $G^{t-1} \sim p_\theta(G^{t-1}|G^t)\, p_\eta(\hat{y}|G^{t-1})$ ▷ Reverse process

end

return G^0

- Introduction
- DiGress: Discrete Denoising diffusion for graph generation
- Improving DiGress: Some algorithmic improvements to DiGress
- Experiments
- Conclusion

Experiments

General graph generation

- We use two datasets made of 200 graphs: one drawn from the **stochastic block model** (with up to 200 nodes per graphs), and another dataset of **planar graphs** (64 nodes per graph).
- We evaluate the ability of the models to correctly model various properties of these graphs. In particular, we measure if the generated graphs are statistically distinguishable from the SBM model, or if they are planar and connected.

Table 1: Unconditional generation on SBM and planar graphs. VUN: valid, unique & 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%**

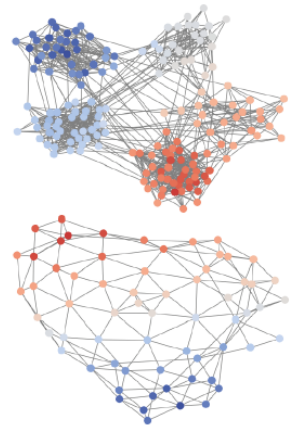


Figure 3: Samples from DiGress trained on SBM and planar graphs.

Metrics The reported metrics compare the discrepancy between the distribution of some metrics on a test set and the distribution of the same metrics on a generated graph. The metrics measured are degree distributions, clustering coefficients, and orbit counts (it measures the distribution of all substructures of size 4). We do not report raw numbers but ratios computed as follows:

$$r = \mathrm{MMD}(\mathit{generated}, \mathit{test}) \,/\, \mathrm{MMD}(\mathit{training}, \mathit{test})$$

Experiments

■ Small molecule generation

- We then evaluate our model on the standard **QM9 dataset** that contains molecules with up to 9 heavy atoms.
- We keep 100k molecules for training, 20k for validation and 13k for evaluating likelihood on a test set.
- We report the **negative log-likelihood** of our model, **validity** (measured by RDKit sanitization) and **uniqueness** over 10k molecules.

Table 2: Molecule generation on QM9. Training time is the time needed to reach 99% validity. On small graphs, DiGress achieves similar results to the continuous model but is faster to train.

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**

Experiments

Conditional generation

- We propose a **conditional generation** setting on QM9.
- We sample 100 molecules from the test set and retrieve their **dipole moment** μ and the **highest occupied molecular orbit** (HOMO).
- The pairs (**μ,HOMO**) constitute the conditioning vector that we use to generate 10 molecules.
- To evaluate the ability of a model to condition correctly, we first use RdKit to **produce conformers** of the generated graphs, and then use Psi4 to **estimate the values of μ and HOMO**.

Figure 4: Mean absolute error on conditional generation with discrete regression guidance on QM9.

Target	μ	HOMO	μ & HOMO
Uncondit.	1.71±.04	0.93±.01	1.34±.01
Guidance	0.81±.04	0.56±.01	0.87±.03

Experiments

■ Molecule generation at scale

➢ We finally evaluate our model on two much more challenging datasets made of more than a million molecules: **MOSES**, which contains small drug-like molecules, and **GuacaMol**, which contains larger molecules.

Table 3: Molecule generation on MOSES. DiGress is the first one-shot graph model that scales to this dataset. While all graph-based methods except ours have hard-coded rules to ensure high validity, DiGress outperforms GraphInvent on most other metrics.

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

Table 4: Molecule generation on GuacaMol. While SMILES seem to be the most efficient molecular representation, DiGress is the first general graph generation method that achieves correct performance on this dataset, as visible on the Frechet ChemNet Distance score (FCD).

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	85.2	100	99.9	92.9	68.0

Metrics Since MOSES and Guacamol are benchmarking tools, they come with their own set of metrics that we use to report the results. We briefly describe this metrics: Validity measures the proportion of molecules that pass basic valency checks. Uniqueness measures the proportion of molecules that have different SMILES strings (which implies that they are non-isomorphic). Novelty measures the proportion of generated molecules that are not in the training set. The filter score measures the proportion of molecules that pass the same filters that were used to build the test set. The Frechet ChemNetDistance (FCD) measures the similarity between molecules in the training set and in the test set using the embeddings learned by a neural network. SNN is the similarity to a nearest neighbor, as measured by Tanimoto distance. Scaffold similarity compares the frequencies of Bemis-Murcko scaffolds. The KL divergence compares the distribution of various physicochemical descriptors.

- Introduction
- DiGress: Discrete Denoising diffusion for graph generation
- Improving DiGress: Some algorithmic improvements to DiGress
- Experiments
- Conclusion

Conclusion

- This paper proposed **DiGress**, a new discrete denoising diffusion model for graph generation.
- DiGress learns to **progressively edit** a random graph in order to turn it into a realistic graph. It can be **conditioned** both on graph-level properties and on predefined subgraphs.
- DiGress **outperforms** existing one-shot generation methods and is able to scale to **larger datasets**, reaching the performance of autogressive models on molecule generation.

Thank You for listening!

AI4Drugs讨论班

2022-11-18