Topic 7 - DiGress: Discrete Graph Denoising Diffusion Model

Jilixin Tang

Note: this presentation was created as part of the WS 24/25 Hands-on Graph Neural Networks Seminar at Saarland University

03/02/2025

Outline

- Establishing **motivation** for AI-generated molecules
- What is a **diffusion** model?
- How to **design** and **train** DiGress?
- DiGress's **performance**
- Discussion

Motivation for AI-Generated Molecules

Why use AI not rule-based algorithms instead?

- Drug discovery and material science require efficient exploration of large chemical spaces.
- Traditional methods constrained by rules (e.g., Lipinski's Rule of Five).
- Generative models leverage graph structures (nodes = atoms, edges = bonds).

A Real-life Example

Bloomberg
https://www.bloomberg.com › news › articles › german-...

German Power Slips Below Zero as Negative-Price ...

2 Jan 2025 — **German power prices dropped below zero on the first trading day of the year**, an increasingly frequent phenomenon in Europe as renewables ...

Negative Electricity Prices in Germany

In 2024, Germany experienced with almost 460 hours of negative prices [1].
Why?

- Inflexible **electricity generation** that led to surplus energy on days of low-demand(e.g., holidays).
- Limited **energy storage infrastructure** to save surplus energy for later use.

A Visual Overview of Hydrogen Energy Storage

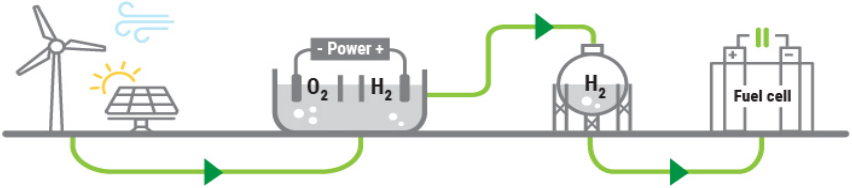


Figure: Top: Hydrogen-based energy storage system with **catalysts** aiding electrolysis and fuel cells [2]. Bottom: Industrial hydrogen storage facility [3].

The Open Catalyst Project

Open Catalyst Project is led by Meta and Carnegie Mellon University and aims to tackle the problem of renewable energy storage.

The Role of AI in Catalyst Discovery

- One solution is the conversion of renewable energy to other fuels, such as hydrogen. Hydrogen is used as a medium to store energy generated from wind and solar as an alternative to batteries.
- An open challenge is finding **low-cost catalysts** to drive these reactions at high rates. However, designing and testing new catalysts in a traditional chemical laboratory is fairly expensive and time-consuming.
- The use of AI or machine learning may **accelerate** catalyst discovery by approximating these calculations and experiments [4].

What is a Diffusion Model?

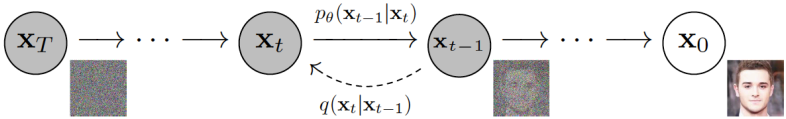


Figure: The Markov chain of forward (reverse) diffusion process, where noise is gradually added and removed to generate a clean sample [5].

High-Level Understanding of Diffusion Models

- Diffusion models are often characterized as the **Markov chains** of the **forward diffusion** process which gradually blurring the image; the **reverse diffusion** iterations that recover the clean sample step by step.
- Information is removed and recovered on a **global level**, allowing diffusion models to capture broader patterns compared to **auto-regressive models**, which learn local dependencies sequentially.

DiGress Model uses Discrete Noise for Graph Diffusion

What is Discrete Noise?

- Unlike adding continuous values from a Gaussian distribution for images, **discrete** noise refers to inserting **categorical** noise.
- Nodes and edges belong to **predefined categories**, noise is added by gradually altering the types of the original nodes and edges.

```
atom_one_hot_encodings = torch.tensor([
                         [1, 0, 0, 0, 0],  # Hydrogen
                         [0, 1, 0, 0, 0],  # Carbon
                         [0, 0, 1, 0, 0],  # Nitrogen
                         [0, 0, 0, 1, 0],  # Oxygen
                         [0, 0, 0, 0, 1]   # Fluorine
                         ])

bond_one_hot_encodings = torch.tensor([
                         [1, 0, 0, 0],  # SINGLE
                         [0, 1, 0, 0],  # DOUBLE
                         [0, 0, 1, 0],  # TRIPLE
                         [0, 0, 0, 1]   # AROMATIC
                         ])
```

Figure: Code snippet from the DiGress Colab notebook.

An Overview of DiGress I

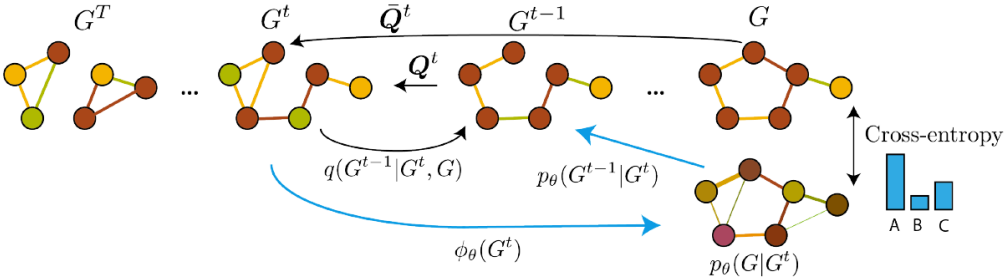


Figure: Overview of DiGress[6].

An Overview of DiGress II

Forward Diffusion: $q(G^t|G^{t-1})$ $\begin{cases} \text{Noise } \mathbf{Q}^t \\ \text{Limit Distribution } \mathbf{m} \end{cases}$

Reverse Diffusion $p_\theta(G_i^{t-1}|G_i^t)$ $\begin{cases} \text{Denoising Neural Network } \phi_\theta \\ \text{Ideal Reverse Diffusion: } q(G_i^{t-1}|G, G_i^t) \end{cases}$

Forward Diffusion: the Noise $\mathbf{Q^t}$

The noise is represented by **Markov transition matrices** $(\mathbf{Q^1}, ..., \mathbf{Q^T})$:

```
# Transition probability matrix at step 1
Q1_X = torch.tensor([
                     [0.4, 0.2, 0.2, 0.1, 0.1],
                     [0.1, 0.6, 0.1, 0.1, 0.1],
                     [0.2, 0.2, 0.5, 0.05, 0.05],
                     [0.1, 0.1, 0.1, 0.6, 0.1],
                     [0.05, 0.05, 0.05, 0.15, 0.7]
                     ])

# Transition probability matrix at step 2
Q2_X = torch.tensor([
                     [0.5, 0.3, 0.1, 0.05, 0.05],
                     [0.2, 0.5, 0.2, 0.05, 0.05],
                     [0.1, 0.1, 0.6, 0.1, 0.1],
                     [0.05, 0.2, 0.1, 0.6, 0.05],
                     [0.1, 0.05, 0.05, 0.1, 0.7]
                     ])
```

- each step has a different transition matrix $\mathbf{Q^t}$
- $[\mathbf{Q}_X^t]_{ij}$ represents the probability of a node transitioning from type i at time $t-1$, to type j at time t
- $q(G^t|G^{t-1}) = (\mathbf{X}^{t-1}\mathbf{Q}_X^t, \mathbf{E}^{t-1}\mathbf{Q}_E^t)$
- As the process is Markovian: $\bar{\mathbf{Q}}^t = \mathbf{Q^1}\mathbf{Q^2}...\mathbf{Q^t}$
- $q(G^t|G) = (\mathbf{X}\bar{\mathbf{Q}}_X^t, \mathbf{E}\bar{\mathbf{Q}}_E^t)$
- See illustration on the board

Reverse Diffusion: Denoising Neural Network ϕ_θ

a graph transformer parametrised by θ, that takes a noisy graph $\mathbf{G}^t = (\mathbf{X}^t, \mathbf{E}^t)$ as input and make predictions for its clean state $\mathbf{G}$. The model ϕ_θ is trained by optimising the cross entropy between the network predictions and the true graph:

$$\text{cross-entropy}\left(x_i, \widehat{p}_i^X\right) = -\sum_k x_i^{(k)} \log \widehat{p}_i^{X(k)} \quad (1)$$

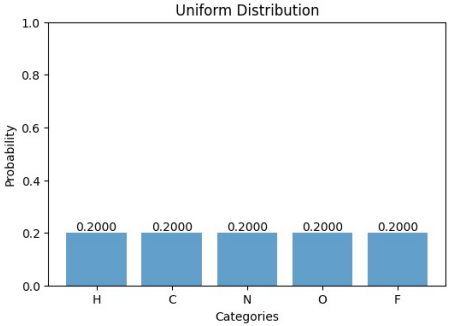
Network Prediction of Node

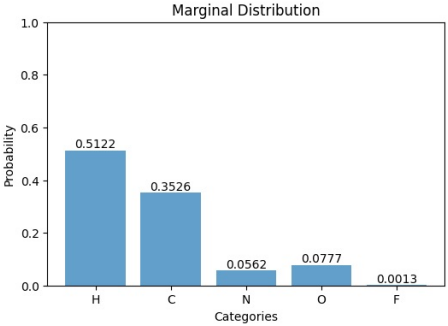
$$l(\widehat{p}^G, G) = \sum_{1 \le i \le n} \text{cross-entropy}\left(x_i, \widehat{p}_i^X\right) + \lambda \sum_{1 \le i,j \le n} \text{cross-entropy}\left(e_{ij}, \widehat{p}_{ij}^E\right) \quad (2)$$

Network Prediction of Edge

Forward Diffusion: the Limit Distribution I

The limit distribution of the noise model is the distribution that the **forward noise model** $q(x^T|x) = \mathbf{x}\bar{\mathbf{Q}}^T$ converges to, when $T \to \infty$. It is also the distribution from which we can directly sample noisy graphs.





Forward Diffusion: the Limit Distribution II

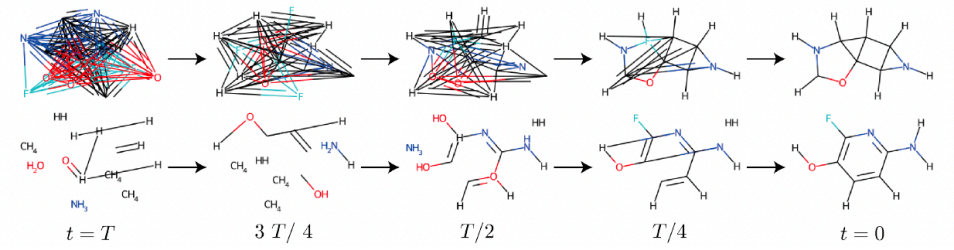


Figure: Models trained on uniform distribution(top) and marginal distribution(bottom)[6]

Forward Diffusion: the Optimal Noise Model (if time allows)

It has been argued in DiGress that the **The optimal noise model** is:

Identity Matrix

$$\mathbf{Q}_X^t = \alpha^t \mathbf{I} + (1-\alpha)^t \mathbf{1}_\sigma \mathbf{m}_X^{'} \quad (3)$$

Marginal Matrix

- The **noise schedule** is governed by α^t, a parameter that transitions from 1 to 0 as t increases.
- When α^t is close to 1 (**early steps**), $\mathbf{Q}^t$ is closer to the identity matrix, representing no added noise.
- When α^t is close to 0 (**later steps**), $\mathbf{Q}^t$ becomes more heavily weighted by the marginal distribution of node/edge types $\mathbf{m}^{'}$.

The Training Algorithm

Step 0: Prepare a batch of clean graphs as input.
Step 1: Define the noise model.
Step 2: Get discrete noisy graphs using the cumulative transition matrices.
Step 3: Define the graph transformer, which takes a noisy graph as input and makes predictions of its clean state.
Step 4: Train the graph transformer by optimizing the cross entropy between the true graph and the predictions.

Reverse Diffusion: the Ideal Reverse

Using Bayes' rule and Markovian properties, we have the ideal reverse of the forward noise model in closed form, conditioned on the clean state x:

Clean State Clean State

$$q(x_i^{t-1} | x, x_i^t) \propto \mathbf{x}^t (\mathbf{Q}^t)' \odot \mathbf{x}\, \mathbf{Q}^{t-1} \quad (4)$$

Transpose of Transition Matrix

as opposed to:

$$q(x_i^{t-1} | x_i^t) = x_i^t (\mathbf{Q}^t)' \quad (5)$$

since knowing only x^{t-1} leaves ambiguity about which x^t state it came from:

$$\mathbf{x_1^2} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix} (\mathbf{Q^2})' = \begin{bmatrix} 0.5 & 0.0 & 0.5 & 0.0 & 0.0 \\ 0.2 & 0.3 & 0.4 & 0 & 0.1 \\ 0.3 & 0.0 & 0.1 & 0.5 & 0.1 \\ 0.4 & 0.3 & 0.2 & 0.1 & 0 \\ 0 & 0 & 0.7 & 0.1 & 0.2 \end{bmatrix} \mathbf{x_1^2} (\mathbf{Q^2})' = \begin{bmatrix} 0.5 & 0 & 0.5 & 0 & 0 \end{bmatrix}$$

The Sampling Algorithm I: Reverse Diffusion

The generation model is the **reverse diffusion** $p_\theta(x_i^{t-1}|x_i^t)$ we discussed earlier, which is calculated combining the **denoising network prediction** and the **ideal reverse**. It marginalizes over all five types (for the QM9 dataset) of x_i:

Ideal Reverse Network Prediction

$$p_\theta(x_i^{t-1}|x_i^t) = \sum_{x \in \mathscr{X}} \begin{cases} q(x_i^{t-1}|x_i = x, x_i^t) \cdot \widehat{p}_i^X(x) & \text{if } q(x_i^t|x_i = x) > 0, \\ 0 & \text{otherwise.} \end{cases} \quad (6)$$

This is how DiGress get from **prediction to generation**: with the help of the **ideal reverse** to **formulate** a reverse diffusion process; as well as using the trained **denoising network** to **guide** this iterative process, and finally to generate new graphs.

The Sampling Algorithm II

Step 0: sample a noisy graph from the limit distribution.
Step 1: for $t = T$ to 1 do,

- graph transformer takes the noisy graph G^t as input and make predictions of its clean state $\widehat{p}^X, \widehat{p}^E$;
- using the **network-weighted reverse diffusion** $p_\theta(G_i^{t-1}|G_i^t)$ (ideal reverse weighted by the graph transformer predictions) to get $G^{t-1}, G^{t-2} \ldots G^0$.

DiGress's Performance on Molecule Generation

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)	–	$98.9_{\pm.1}$	$96.8_{\pm.2}$	7.2
DiGress (ours)	$69.6_{\pm1.5}$	$\mathbf{99.0}_{\pm.1}$	$96.2_{\pm.1}$	**1.0**

Figure: DiGress's performance on QM9 molecule generation benchmark, evaluating validity, uniqueness and training time[6].

Discussion

Questions to Consider

- Do nodes and edges have the same noise model?
- What does it mean that the author of DiGress says that the diffusion process under discussion is defined independently for each node and edge?
- Why is the denoising network essentially solving a classification problem in the training phase?
- Why is the graph transformer (i.e., the denoising network) alone not enough for sampling?

References I

F. für Energiewirtschaft e.V. (FFE), "German electricity prices on epex spot 2024," 2024, accessed: Jan 28, 2025. [Online]. Available: https://www.ffe.de/en/publications/german-electricity-prices-on-epex-spot-2024/#:~:text=The%20German%20electricity%20price%20level,year%20was%20once%20again%20surpassed.

NYISO, "The road to 2040: How green hydrogen can complement a clean energy grid," 2023, accessed: Jan 28, 2025. [Online]. Available: https://www.nyiso.com/-/the-road-to-2040-how-green-hydrogen-can-complement-a-clean-energy-grid

D. C. Dynamics, "Preparing for the hydrogen grid," 2023, accessed: Jan 28, 2025. [Online]. Available: https://www.datacenterdynamics.com/en/analysis/preparing-for-the-hydrogen-grid/

MetaAI and C. M. University, "The open catalyst project," 2025, accessed: Jan 28, 2025. [Online]. Available: https://opencatalystproject.org/index.html

J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *CoRR*, vol. abs/2006.11239, 2020. [Online]. Available: https://arxiv.org/abs/2006.11239

References II

C. Vignac, I. Krawczuk, A. Siraudin, B. Wang, V. Cevher, and P. Frossard, "Digress: Discrete denoising diffusion for graph generation," *ICLR*, 2023. [Online]. Available: https://arxiv.org/abs/2209.14734