# DiGress: Discrete Denoising Diffusion for Graph Generation

C. Vignac, I. Krawczuk, A. Siraudin et Al.

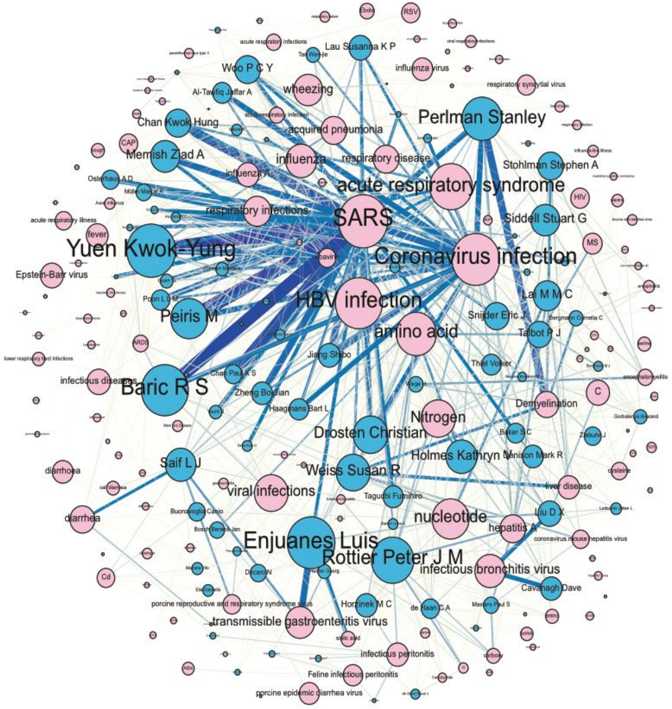
*Presentation by*
*Lucas Morin*

# DiGress: Discrete Denoising Diffusion for **Graph** Generation

# Graph structure

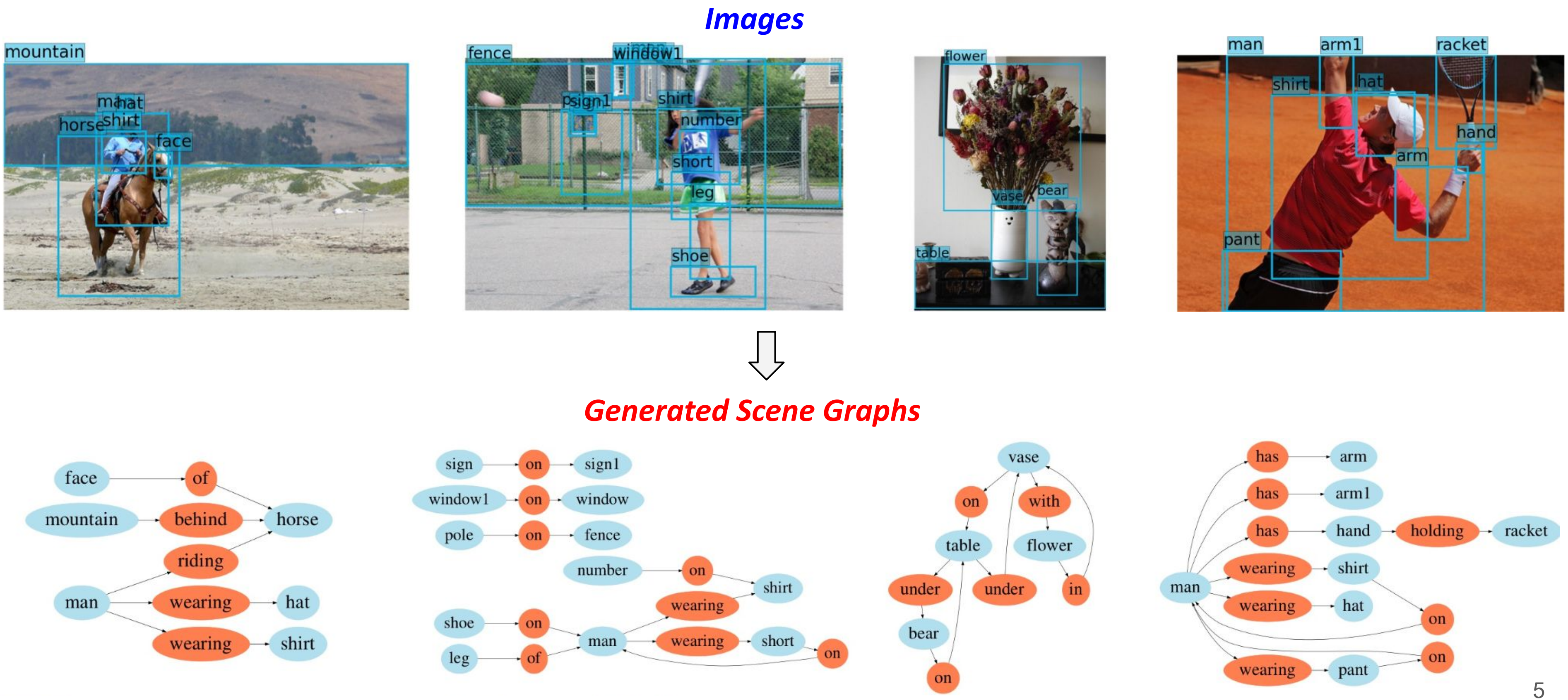
**Social Network**:
Can we predict how information spread?

**Knowledge Graph**:
Can we extract knowledge from a set of documents?



# DiGress: Discrete Denoising Diffusion for **Graph Generation**

# Graph Generation: Scene Graph Generation






*Reference: Scene Graph Generation by Iterative Message Passing*

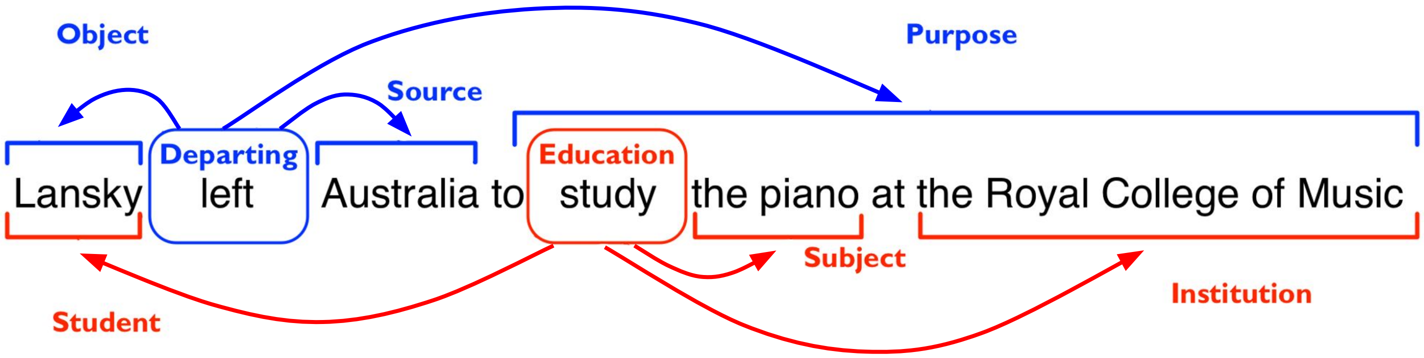
# Graph Generation: Semantic Role Labelling

***Text***

Lansky left Australia to study the piano at the Royal College of Music

⇩

***Semantic Graph***





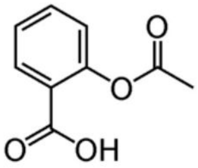

*Reference: http://sanjaymeena.io/tech/semantic_role_labeling/*

# Graph Generation: De Novo Molecular Generation

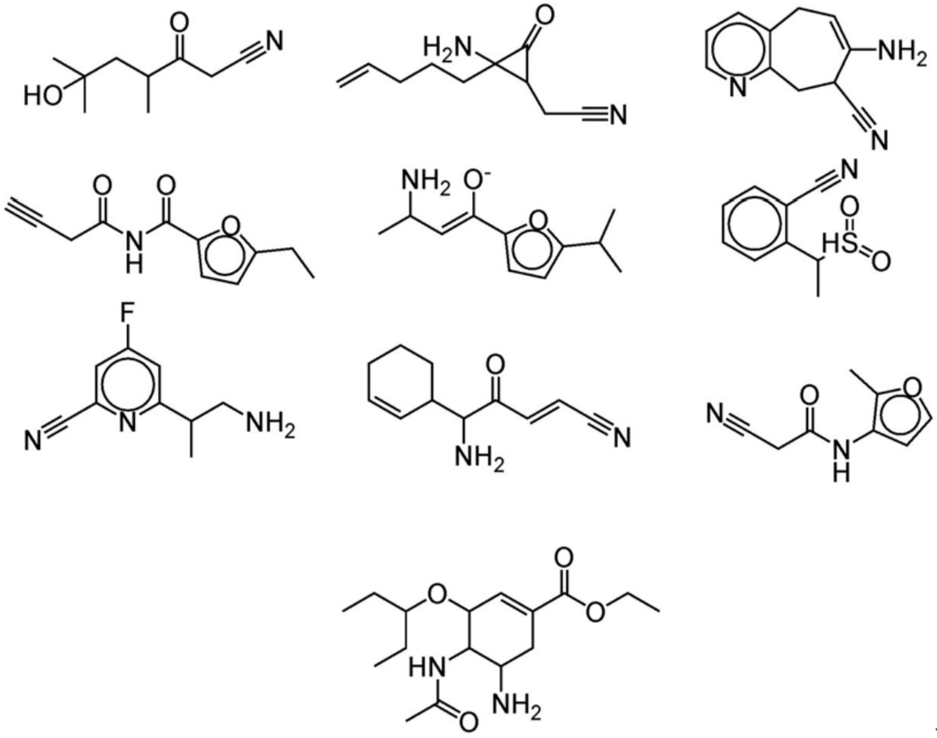
***Molecular Properties***

**Aspirin properties**

| | |
|---|---|
| Molecular Weight: | MW= 180 |
| Lipophilicity: | LogP = 1.30 |
| Hydrogen Bond Donor: | HBD = 1 |
| Hydrogen Bond Acceptor: | HBA = 3 |
| Polar Surface Area: | TPSA = 63.6 |



***Novel molecules***





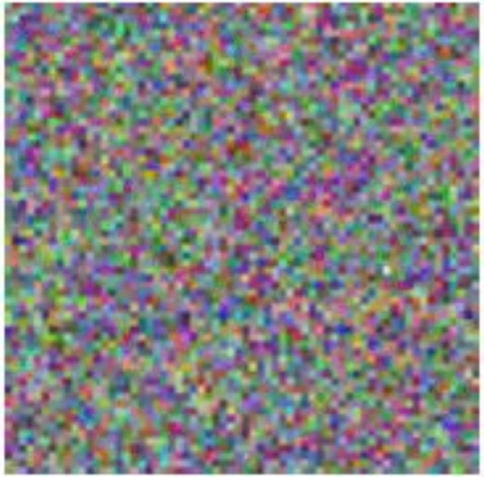
*Reference: Molecular generative model based on conditional variational autoencoder for de novo molecular design*

# DiGress: Discrete **Denoising Diffusion** for Graph Generation

# Diffusion Probabilistic Models

Image Generation: Generate **new** and **diverse** images **similar** to the training images.

**Noise**



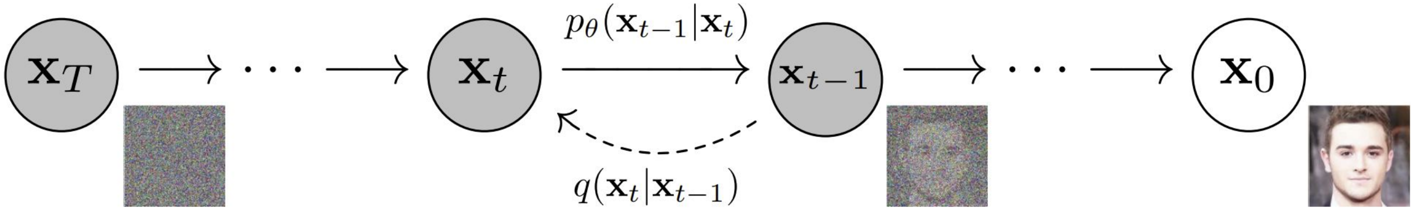

**Generated image**



*Reference: https://keras.io/examples/generative/ddim/*

# Gaussian Diffusion Models

**Markov Chain Model**




$p(x_T) = \mathcal{N}(x_T; 0, I)$

**Forward Process (Noise)**

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \mu_t, \sigma_t^2)$$

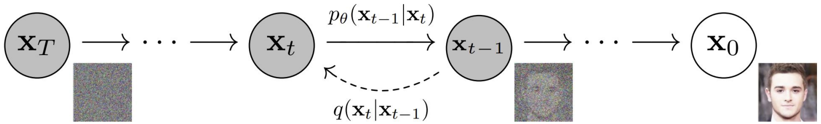
**Backward Process (Learned Model)**

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

(approximation of $q(x_{t-1}|x_t)$)

*Reference: Denoising Diffusion Probabilistic Models*

# Gaussian Diffusion Models




## Forward Process

Isotropic Gaussian    Mean    Variance

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \boxed{\sqrt{1-\beta_t}} x_{t-1}, \boxed{\beta_t} I)$$

$\beta_t$ is the diffusion rate

$$\beta_1, \ldots, \beta_T \in [0, 1]$$

Bring the mean of each new Gaussian closer to 0. This scaling keeps pixels values in bound.

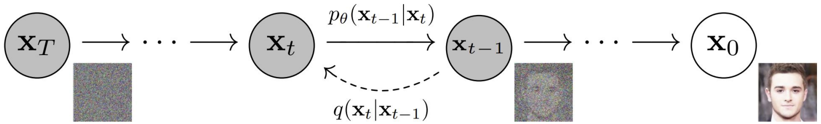
Corrupt the image by shifting pixels values.

$$\lim_{T\to\infty} q(x_T|x_0) = \mathcal{N}(0, I)$$

**Independent of the input image**

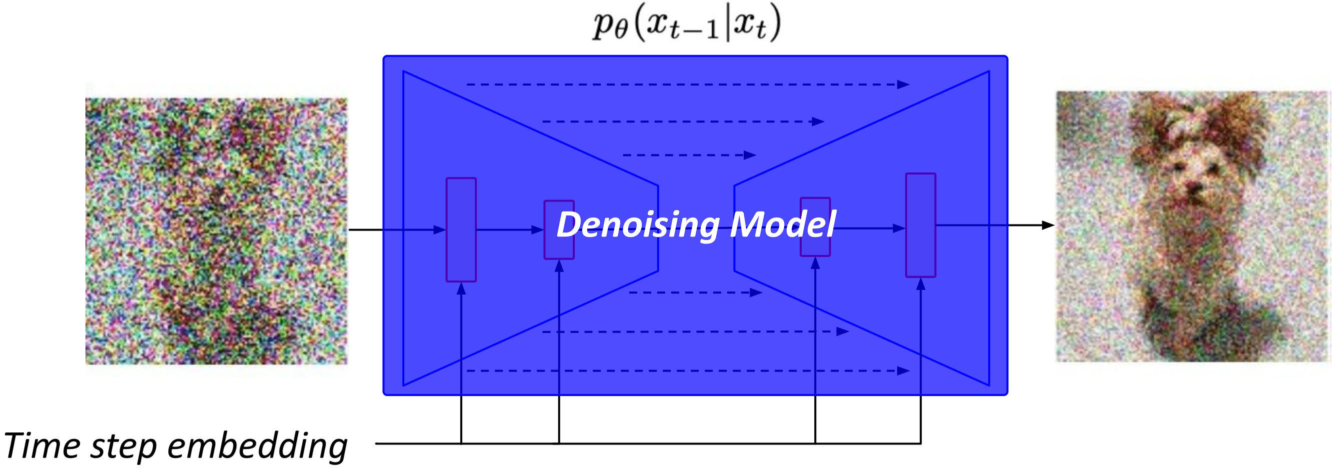
*Reference: Denoising Diffusion Probabilistic Models*

# Gaussian Diffusion Models



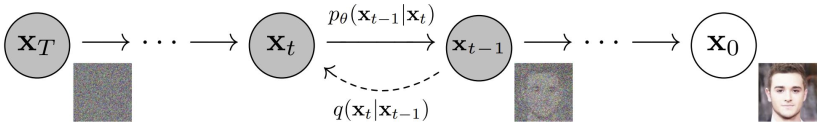

**Backward Learned Process**

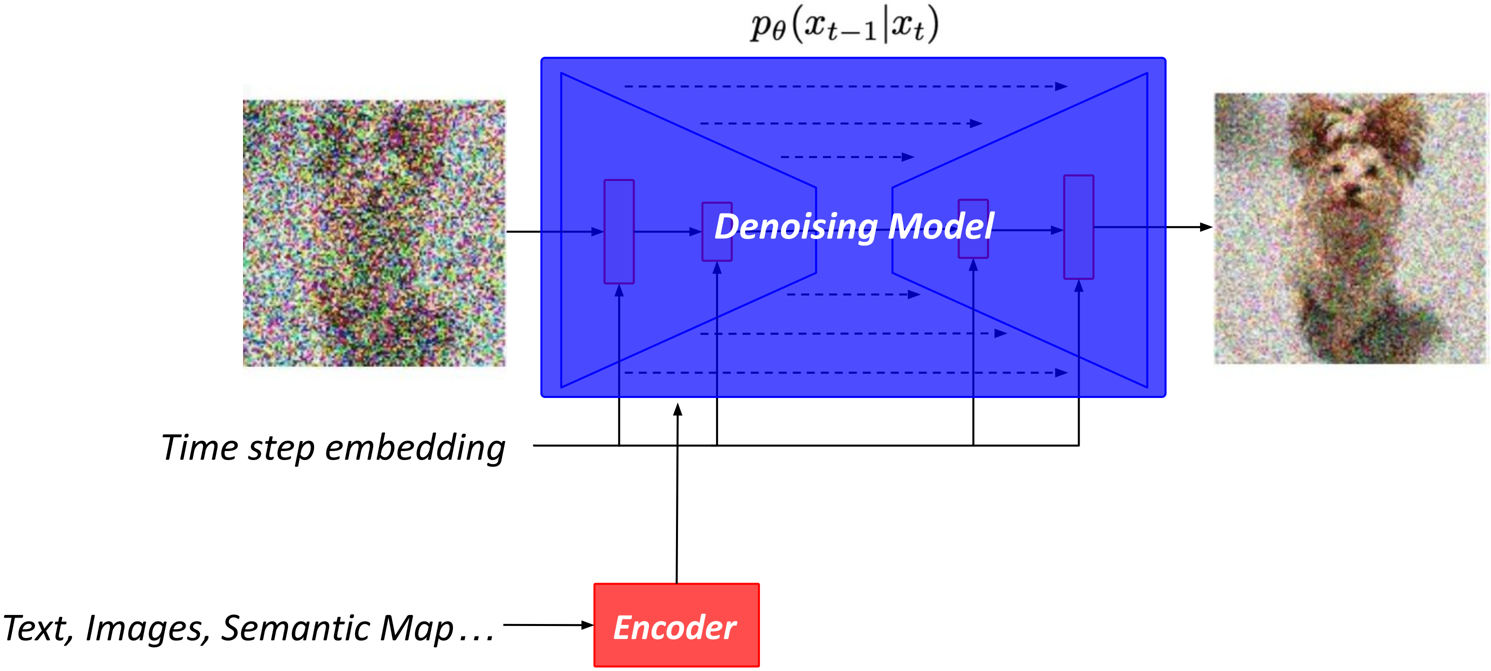
$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$




*Reference: Improved Denoising Diffusion Probabilistic Models*

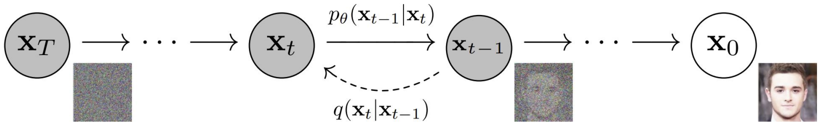
# Guided Gaussian Diffusion Models



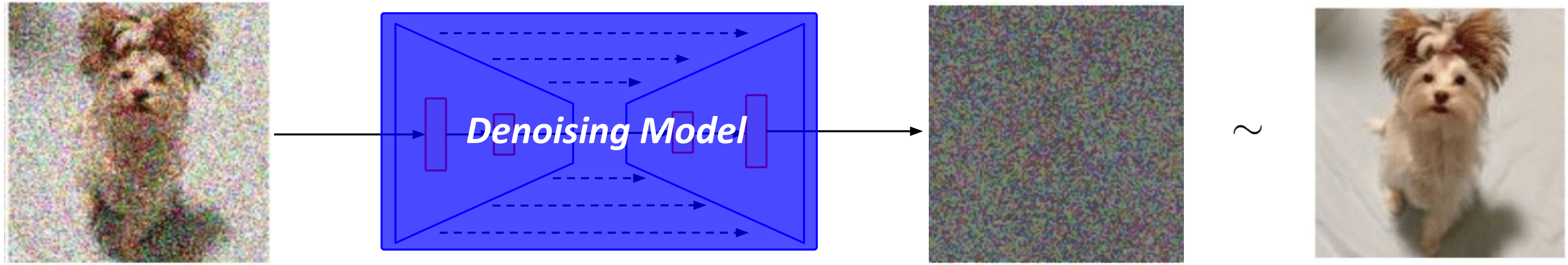

*Reference: High-Resolution Image Synthesis with Latent Diffusion Models*

# Gaussian Diffusion Models




**How to train the denoising model?**

Noisy image Noise Original Image




**Sampling at arbitrary step in closed form**

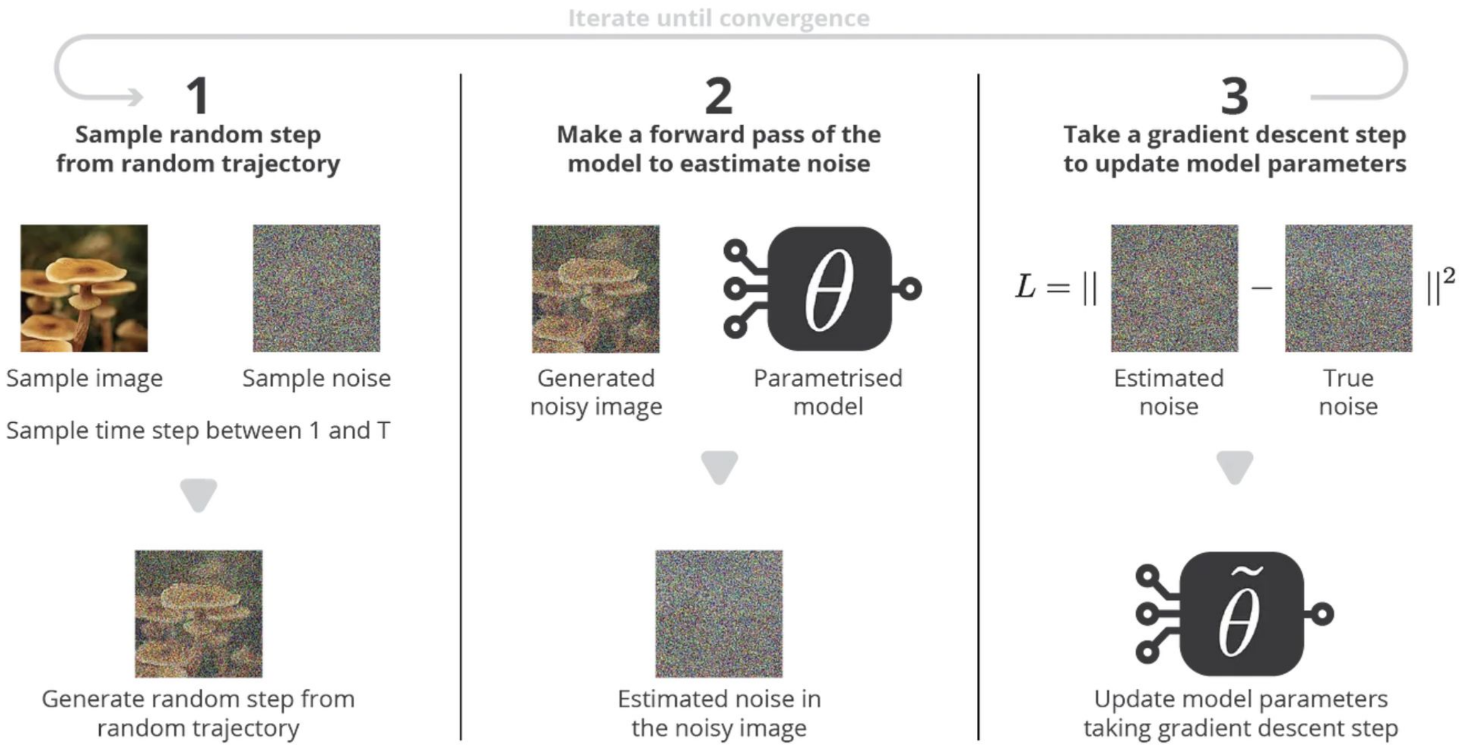
$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \quad \Rightarrow \quad q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I)$$

$$\alpha_t = (1-\beta_t) \qquad \bar{\alpha}_t = \prod_{t=1}^{t} \alpha_t = \prod_{t=1}^{t}(1-\beta_t)$$

**Parallel training**

*Reference: Improved Denoising Diffusion Probabilistic Models*

# Gaussian Diffusion Models




*Reference: Improved Denoising Diffusion Probabilistic Models*

# Gaussian Diffusion Models

**Objective function**

$$L_t^{\text{simple}} = \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\boldsymbol{\epsilon}_t}\left[\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t, t)\|^2\right]$$

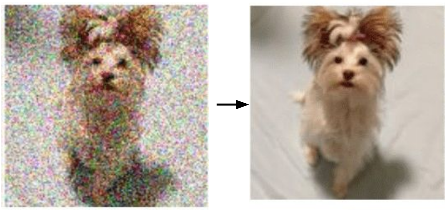
MSE between the true and the predicted Gaussian noise.

**Key advantages**

The model is not trained of trajectories, which reduce one source of noise in the training process.
The training is performed in parallel.

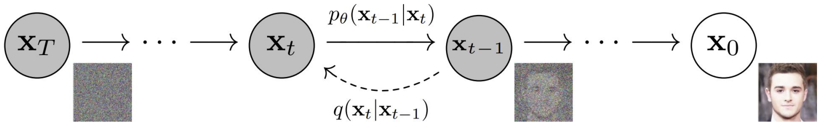
**Inference**

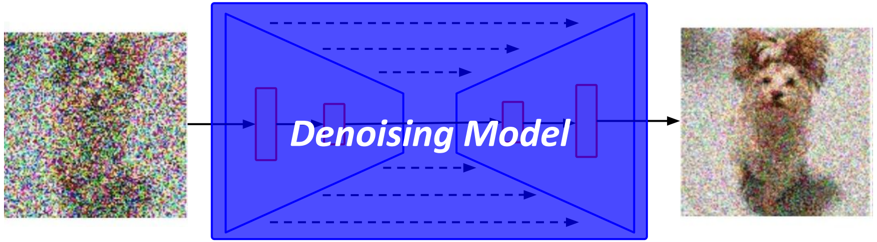
**Training**



*Reference: Improved Denoising Diffusion Probabilistic Models*

# Diffusion Models




**Conditions for efficient training**

**Property 1**

$q(x_t|x_0)$ should have a closed-form formula

**Property 2**

$q(x_{t-1}|x_t, x_0)$ should be tractable

**Property 3**

$\lim_{T\to\infty} q(x_T|x_0)$ should not depend on $x_0$

=> Gaussian Noise (95% of the articles) or Discrete Diffusion

*Reference: DiGress: Discrete Denoising diffusion for graph generation*

# Diffusion Models Summary

**Advantages**

- Competitive generative models against VAE, GAN and Flow-based models:
    - **High** sample generation **quality.**
    - **Diverse** sample **generation.**
- Its iterative nature is that we perform **supervised training at each timestep**.

**Drawbacks**

- **Low sampling rate.**
  (T = 1000 time steps for the first article. More recent works achieve T = 4 time steps.)

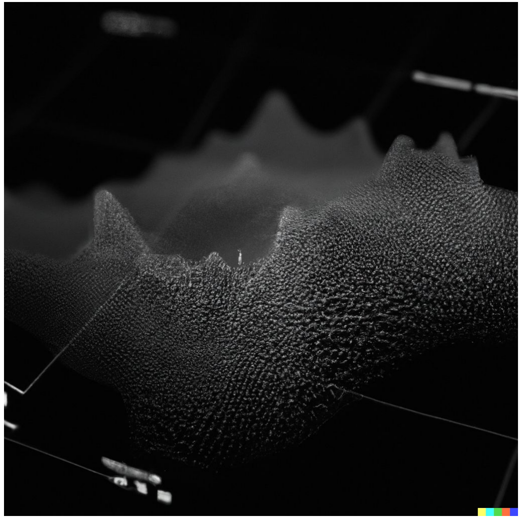


*Reference: Denoising Diffusion Probabilistic Models*

# Diffusion Probabilistic Models: Examples

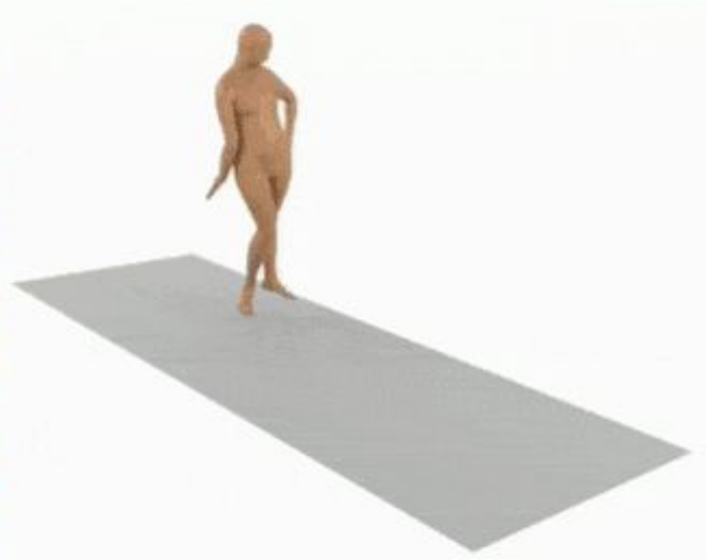
***Image generation***

Text conditioning: "*A diffusion probabilistic model*"



*GLIDE (DALL-E 2)*

***Human Motion***

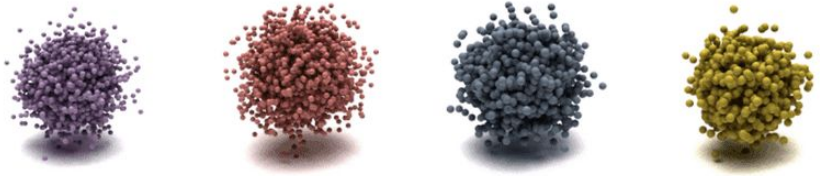


Human Motion Diffusion Model

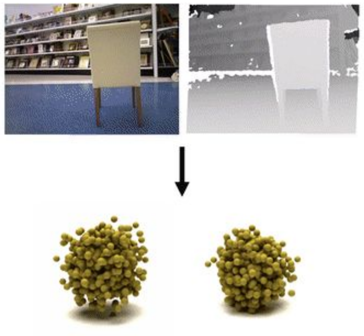
# Diffusion Probabilistic Models: Examples

***3D Point Cloud Generation and Completion***

Shape Generation



Shape Completion



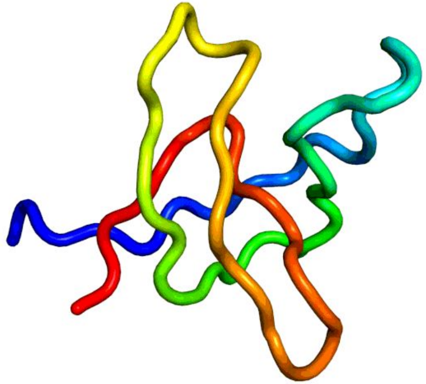
***Novel View Synthesis***



*Reference: 3D Shape Generation and Completion through Point-Voxel Diffusion, Novel View Synthesis with Diffusion Models*

# Diffusion Probabilistic Models: Examples

**Motif-scaffolding problem**



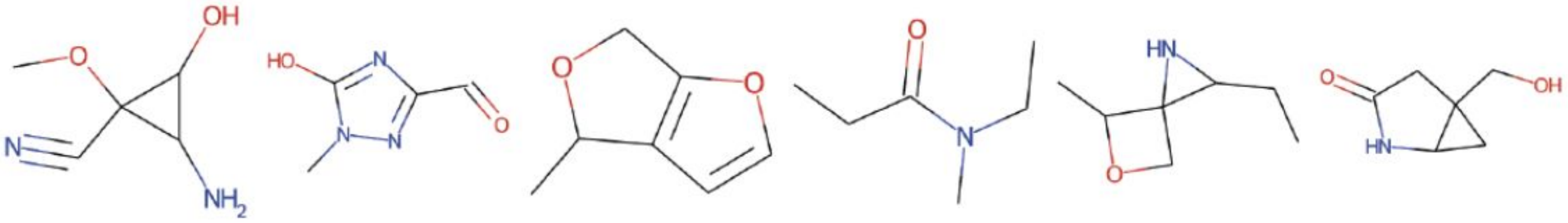
***Graph Generation***

*Reference: Diffusion probabilistic modeling of protein backbones in 3D for the motif-scaffolding problem*

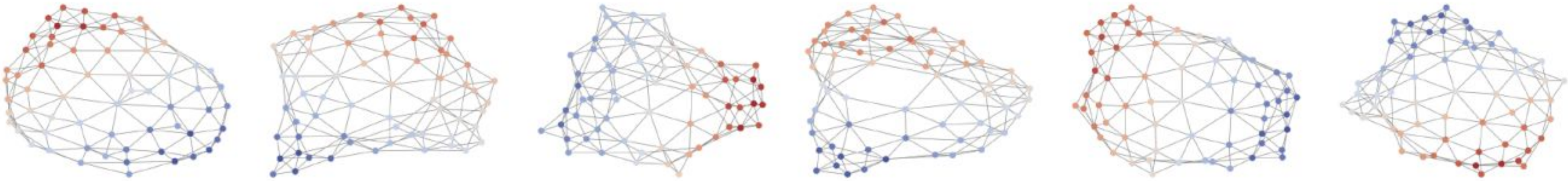
# DiGress: **Discrete** Denoising Diffusion for Graph Generation

# Why do we need Discrete Diffusion for Graph Generation?

**Molecular Graphs**




**Planar Graphs**

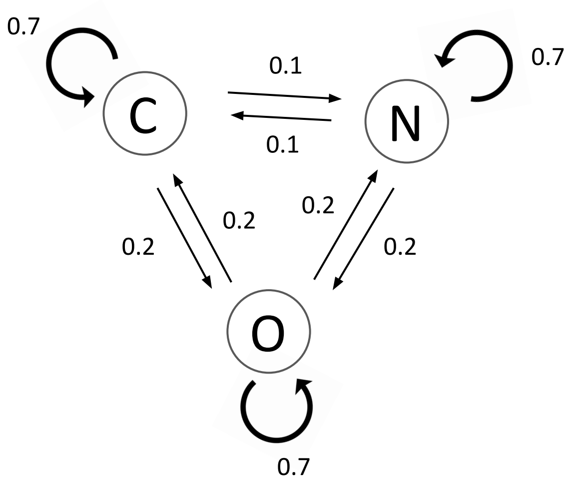


Gaussian noise model **destroys sparsity** as well as graph theoretic notions such as **connectivity** or **cycle counts**.

*Reference: DiGress: Discrete Denoising diffusion for graph generation*

# Discrete Diffusion Models: continuous to discrete state-space

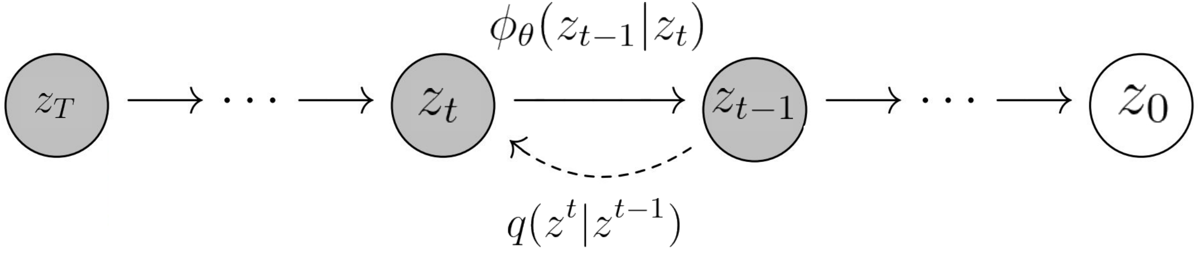
Example of **States** and **Transition Probabilities** for an atom




*Reference: DiGress: Discrete Denoising diffusion for graph generation*

# Discrete Diffusion Models

**Markov Chain Model over a discrete state-space**




**Forward Process (Noise)**

$q(z^t|z^{t-1}) = z^{t-1}Q^t$ Transition matrices $(Q^1, \ldots, Q^T)$

**Backward Process (Learned Model)**

$\phi_\theta(z_{t-1}|z_t)$

*Reference: DiGress: Discrete Denoising diffusion for graph generation*

# Discrete Diffusion Models

**Conditions for efficient training**

**Property 1**

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

$$q(z^t|x) = x\bar{Q}^t$$
$$\bar{Q}^t = Q^1 ... Q^t$$

**Property 2**

$q(z^{t-1}|z^t, x)$ should be tractable

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

**Property 3**

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

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

$$\begin{pmatrix} 0.7 & 0.1 & 0.1 \\ 0.1 & 0.7 & 0.1 \\ 0.1 & 0.1 & 0.7 \end{pmatrix}$$

$$\lim_{T\to\infty} x\bar{Q}^T = 1_d/d$$

*Reference: DiGress: Discrete Denoising diffusion for graph generation*

# DiGress: Discrete Denoising Diffusion for Graph Generation

**Key contributions**

1. First Discrete Diffusion model for Graph Generation.
   => Demonstrates that Discrete Diffusion is superior than Gaussian Noise from Graph Generation.

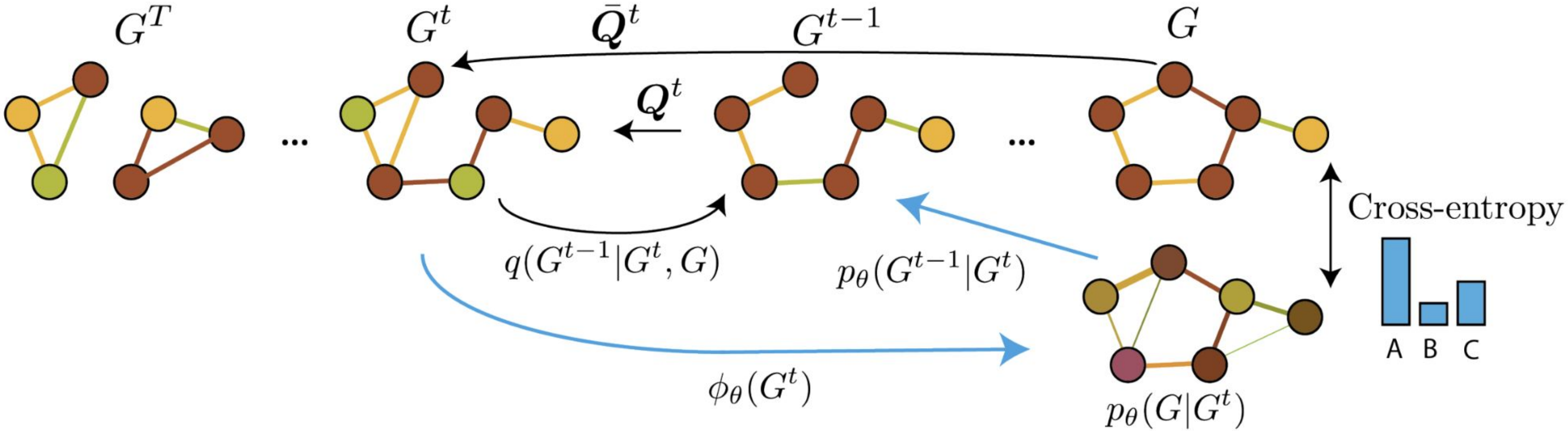
2. High rate generation, realistic generation, high diversity and novelty on molecular and non-molecular graph generation datasets.

3. First one-shot model that can be trained on really large training sets. (MOSES and GuacaMol)

*Reference: DiGress: Discrete Denoising diffusion for graph generation*

# DiGress: Training method

Graph Generation: sequence of node and edge classification tasks

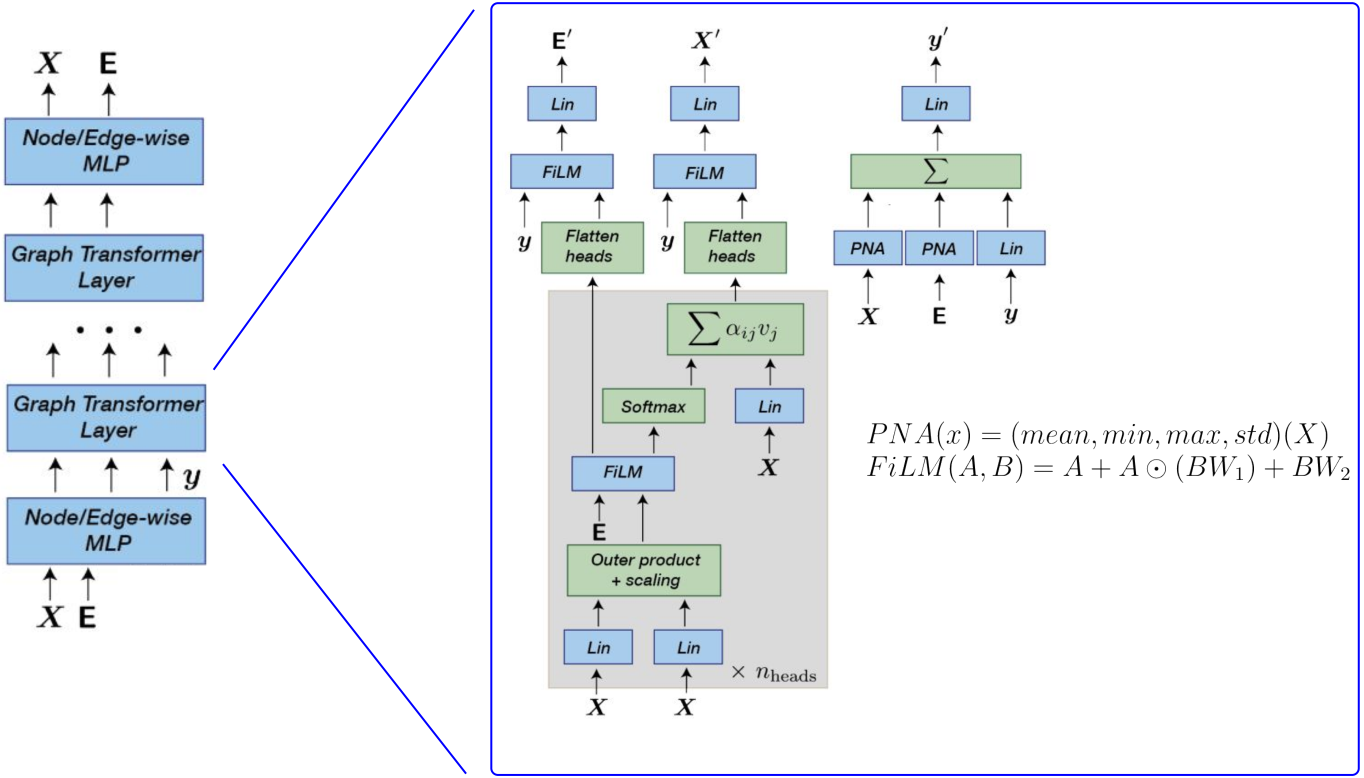





*Reference: DiGress: Discrete Denoising diffusion for graph generation*

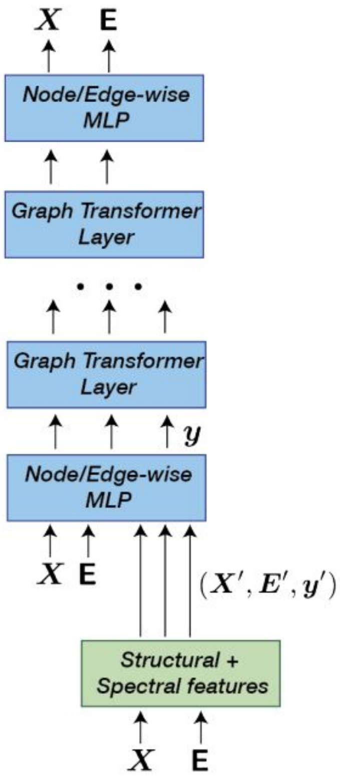
# DiGress: Denoising Model

X : Nodes features
E : Edges features
y : Graph-level features




*Reference: DiGress: Discrete Denoising diffusion for graph generation*

# DiGress: Structural features




**Graph theoretic:**
Cycles and spectral features
(Cycle count, number of connected components, estimation of the biggest connected component)

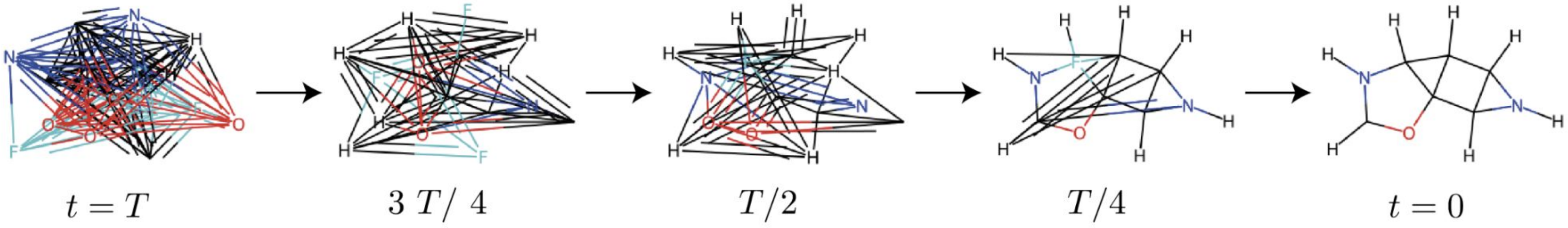
**Domain specific:**
Molecular features
(Valency of each atom, current Molecular Weight)



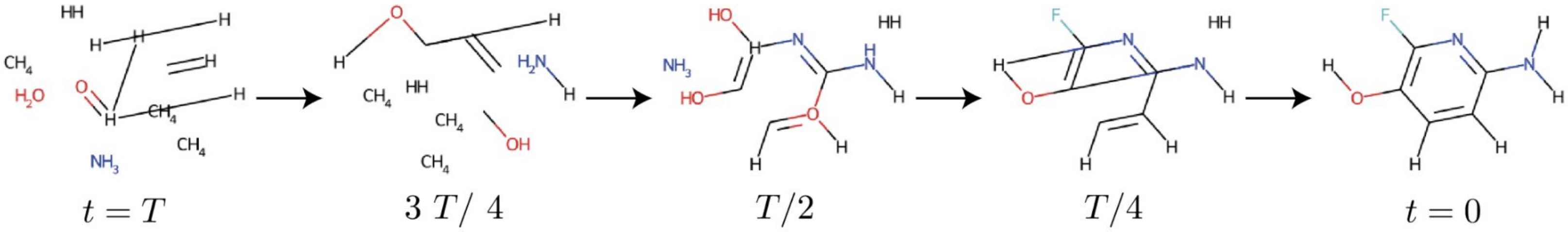
*Reference: DiGress: Discrete Denoising diffusion for graph generation*

# DiGress: Choice of the Noise Model

Uniform transitions




Preservation of marginal distribution of node and edge types

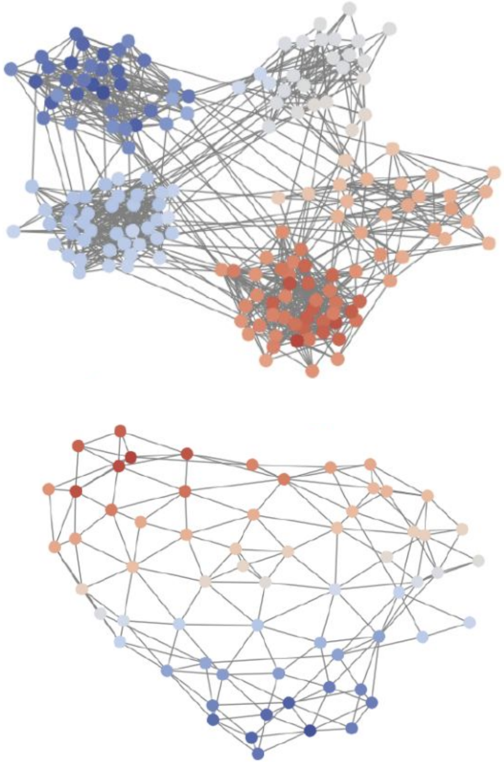





*Reference: DiGress: Discrete Denoising diffusion for graph generation*

# DiGress: Abstract 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%** |



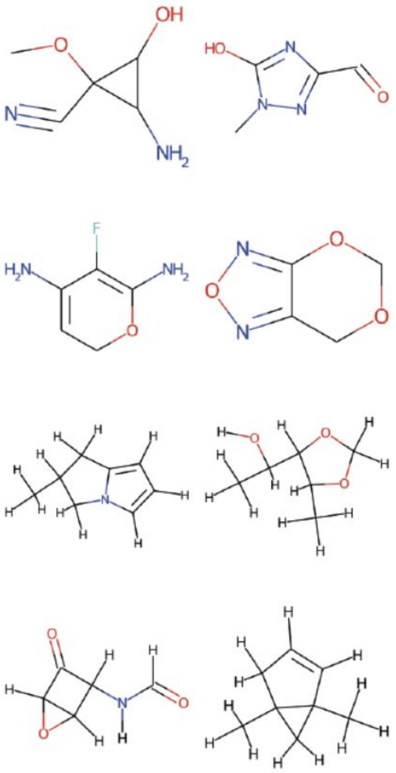


*Reference: DiGress: Discrete Denoising diffusion for graph generation*

# DiGress: Qm9

| 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) | 69.6±1.5 | **99.0**±.1 | 96.2±.1 | **1.0** |

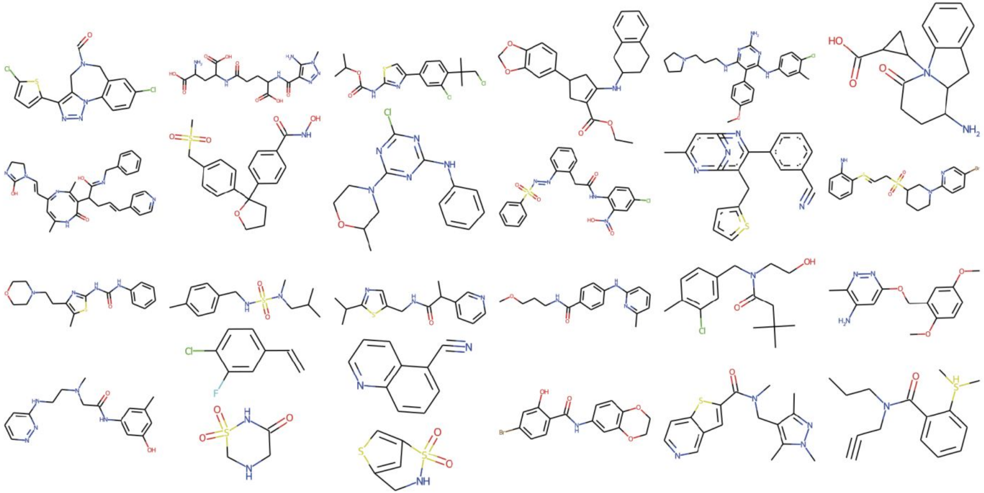
| Model | Valid↑ | Unique↑ | Atom stable↑ | Mol stable↑ |
|---|---|---|---|---|
| Dataset | 97.8 | 100 | 98.5 | 87.0 |
| ConGress | 86.7±1.8 | **98.4**±0.1 | 97.2±0.2 | 69.5±1.6 |
| DiGress (uniform) | 89.8±1.2 | 97.8±0.2 | 97.3±0.1 | 70.5±2.1 |
| DiGress (marginal) | 92.3±2.5 | 97.9±0.2 | **97.3**±0.8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





*Reference: DiGress: Discrete Denoising diffusion for graph generation*

# Diffusion Results:MOSES (1.9M molecules)



| 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 |



*Reference: DiGress: Discrete Denoising diffusion for graph generation*

# DiGress: Conditional Generation

Train a regressor which predicts a graph-level property of interest from a noisy Graph

$q$ Conditional noising process

$\dot{q}$ Unconditional noising process

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

Push the generation towards a graph that have the graph-level property of interest

| Target | $\mu$ | HOMO | $\mu$ & HOMO |
|---|---|---|---|
| Uncondit. | $1.71 \pm .04$ | $0.93 \pm .01$ | $1.34 \pm .01$ |
| Guidance | $0.81 \pm .04$ | $0.56 \pm .01$ | $0.87 \pm .03$ |



*Reference: DiGress: Discrete Denoising diffusion for graph generation*

# DiGress: Conclusion

**Key contributions**

1. First Discrete Diffusion model for Graph Generation.

2. High rate generation, realistic generation, high diversity and novelty on molecular and non-molecular graph generation datasets.

3. First one-shot model that can be trained on really large training sets. (MOSES and GuacaMol)

**Limitations**

1. Evaluation setup.

2. Poor results on conditional generation.