DiGress: Discrete Denoising diffusion for graph generation

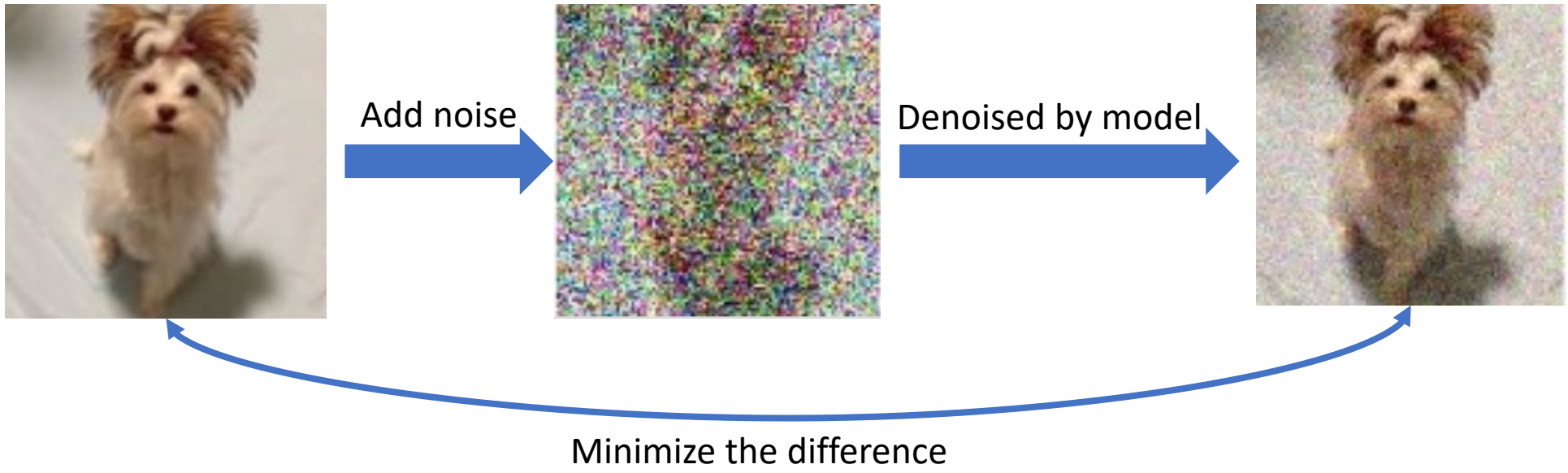
Tinglin Huang

Diffusion Model (1)

- Diffusion model is a powerful class of generative models
  - Generate new sample by recursively denoising a random sampled noise

Diffusion Model (2)

- To learn how to generate the sample with denoising, one intuitive way is to train the model to recover the sample after adding noise



Diffusion Model (3)

- However, such a method suffers from several limitations
  - How to balance the weight of noise we add?
    - Model is hard to learn if the weight is too large; Otherwise, the task is too simple
  - There is a bias between training and inference
    - Training: denoise a sample conditional on the sample distribution $p(x)$
    - Inference: denoise a randomly sampled noise
- Diffusion model addresses these issues **by denoising a trajectory of noised samples**

Diffusion Model (4)

- Main idea
  - Add a sequence of noises to the data with a decreasing weight
    - **Advantage**: when $t \to \infty$, the noised sample doesn't depend on $p(x)$
    - x: original instance, $n \sim N(0,1)$: random noise, α: weight

$$x^t = \alpha^t x^{t-1} + (1 - \alpha^t)n$$

$\alpha^0 = 1$ $\alpha^\infty = 0$

  - Train the model by recovering sample at step t to step $t-1$
    - **Advantage**: offer a more stable and smooth training

Diffusion Model (5)

- Let's formulate the process
  - Sample an instance x
  - Create a sequence of noisy instances $z^1, \ldots, z^T$
    - Markovian structure
    - n^t: sampled noise at step t, α^t: weight at step t

$$z^{t+1} = \alpha^t z^t + (1 - \alpha^t) n^t$$

  - Train the model by learning the distribution $p(z^{t-1}|z^t)$
    - Recovering the sample step by step
    - Predict t-step noised instance based on $t-1$-step noised instance
- Note: $p(z^{t-1}|z^t)$ denotes the denoising distribution, and $q(z^t|x)$ denotes the add-noise distribution

Diffusion Model (6)

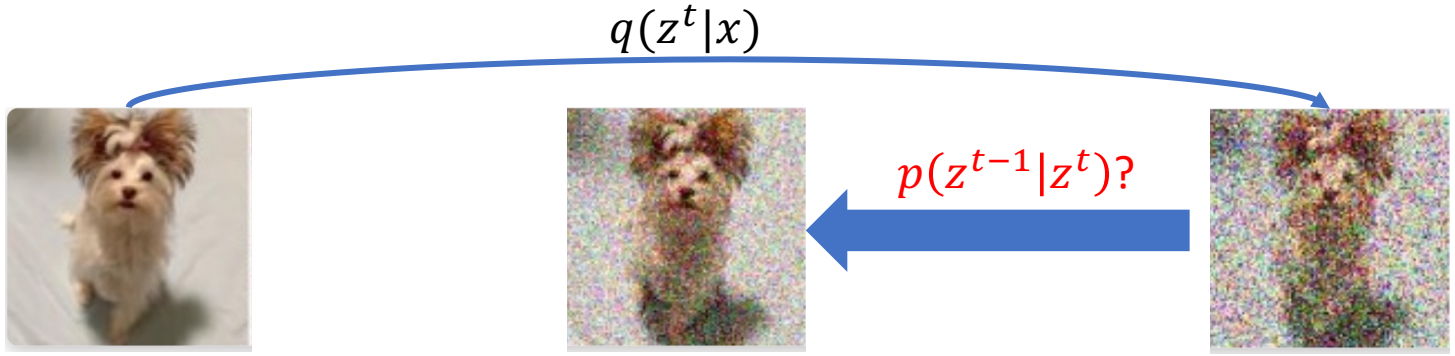
- Three properties required for diffusion model
  - 1) The generation of the noised sample $q(z^t|x)$ should have a close-form formula that only depends on x, α^t, n^t
    - So that we can directly generate z^t based on x
    - Enable a parallel training

Past:

Now:

Diffusion Model (7)

- Three properties required for diffusion model
  - 2) $p(z^{t-1}|z^t)$ should also have a close-form formula that depends on x
    - It can be used for loss function with the target of x



  - 3) $q(z^{\infty}|x)$ should not depend on x
    - We can generate the sample from the noise distribution (e.g., Gaussian distribution) during inference

Discrete diffusion model (1)

- 1) Consider the data point x that belongs to one of d classes
  - x, z^t would be one-hot vectors
  - For example: Atom type (C, O, H, N) in molecule; Word in sequence
- 2) Use **transition matrix** Q as the noise instead of Gaussian distribution
  - $Q \in \mathbb{R}^{N \times N}$
    - N: number of types
    - Q_{ij}: the probability of jumping from type i to type j
  - Similar to the graph adjacent matrix
  - The process of add noise: randomly change the type according to probability

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

Discrete diffusion model (2)

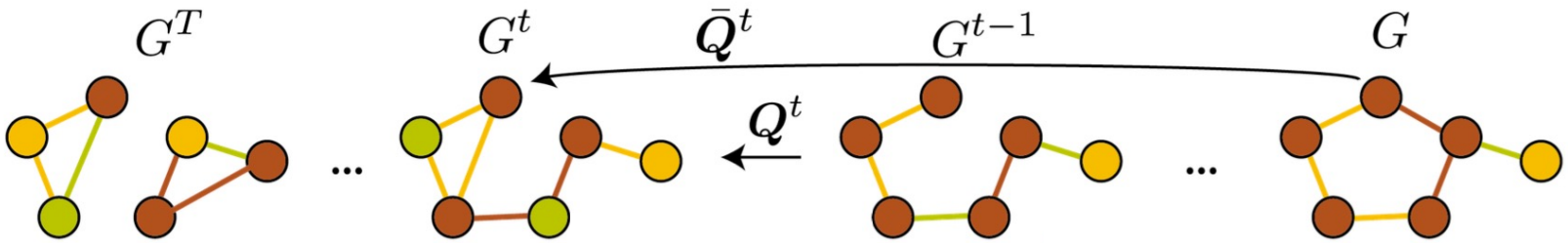
- It can satisfy the three requirements for diffusion model
  - 1) The noise sample can be built by apply noise recursively
    - $q(z^t|x) = x\bar{Q}^t = xQ^1Q^2 \dots Q^t$
  - 2) $p(z^{t-1}|z^t, x)$ can also be computed in closed-form
    - $p(z^{t-1}|z^t, x) \propto z^t(Q^t)' \odot x\bar{Q}^{t-1}$
      - $(\cdot)'$ denotes the transpose of Q
  - 3) When $t \to \infty$, $q(z^t|x)$ converges to a noise distribution independently of x
    - Uniform distribution as noise distribution
      - $Q^t = \alpha^t I + (1 - \alpha^t)1_N 1_N'/N$

[1] Structured denoising diffusion models in discrete state-spaces
[2] Diffsound: Discrete diffusion model for text-to-sound generation
[3] Beyond in-place corruption: Insertion and deletion in denoising probabilistic models
[4] Argmax flows and multinomial diffusion: Learning categorical distributions

DiGress (1)

- Notation
  - a: number of node type, b: number of edge type, n: number of nodes
  - $X \in \mathbb{R}^{n \times a}$: One-hot encoding for all the nodes x_i
  - $E \in \mathbb{R}^{n \times n \times b}$: One-hot encoding for all the edges e_{ij}
  - $G^t = (X^t, E^t)$: Graph at step t
- Contribution
  - Adapt the discrete diffusion model to the graph generation problem
  - Propose a noise model
  - Extend the framework to a conditional generation method

DiGress (2)



- Innovation: diffuse separately on each node and edge feature
  - $Q_X^t \in \mathbb{R}^{a \times a}$ for node and $Q_E^t \in \mathbb{R}^{b \times b}$ for edge
  - Adding noise to graph == Adding noise to node and edge
    - Step by step adding noise

$$q(G^t|G^{t-1}) = (X^t Q_X^t, E^t Q_E^t)$$

    - Obtain the noise directly from G (Property 1)

$$q(G^t|G) = (X^t \bar{Q}_X^t, E^t \bar{Q}_E^t)$$

$$\bar{Q}_X^t = Q_X^1 \dots Q_X^t \qquad \bar{Q}_E^t = Q_E^1 \dots Q_E^t$$

DiGress: Training objective (1)

- Recall: training with denoising, i.e., predict $p(z^{t-1}|z^t)$
- DiGress formulates it as the prediction over nodes and edges

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

- For node, we can have

$$p\left(x_i^{t-1}\middle|G^t\right) = \sum_x p\left(x_i^{t-1}\middle|x_i = x, G^t\right) p(x_i = x|G^t)$$

- For edge

$$p\left(e_{ij}^{t-1}\middle|G^t\right) = \sum_e p\left(e_{ij}^{t-1}\middle|e_{ij} = x, G^t\right) p(e_{ij} = x|G^t)$$

DiGress: Training objective (2)

- Recall: $p(x^{t-1}|x^0, x^t) \propto x^t (Q^t)' \odot x^0 \bar{Q}^{t-1}$ (Property 2)
- We can have

$$p\left(x_i^{t-1}\middle|G^t\right) = \sum_x p\left(x_i^{t-1}\middle|x_i^0 = x, G^t\right) p(x_i = x|G^t)$$

$$= \sum_x p\left(x_i^{t-1}\middle|x_i^0 = x, x_i^t\right) p(x_i^0 = x|G^t)$$

DiGress assumes node and edge are independent

It is unknown:
The distribution of x_i^0 conditional on t-step noise samples

This distribution is deterministic:
Can be replaced with above equation $p(x^{t-1}|x^0, x^t)$

DiGress: Training objective (3)

- For training
  - 1) We generate noised sample G^t based on G^0
  - 2) Train the model by predicting $p(G^{t-1}|G^t)$
    - Predicting $p\left(x_i^{t-1}\middle|G^t\right)$ and $p\left(e_{ij}^{t-1}\middle|G^t\right)$ in DiGress

$$p\left(x_i^{t-1}\middle|G^t\right) = \sum_x p\left(x_i^{t-1}\middle|x_i^0 = x, x_i^t\right) p(x_i^0 = x|G^t)$$

$$p\left(e_{ij}^{t-1}\middle|G^t\right) = \sum_e p\left(e_{ij}^{t-1}\middle|e_{ij}^0 = e, e_{ij}^t\right) p(e_{ij}^0 = e|G^t)$$

Deterministic

Target: predict these two terms

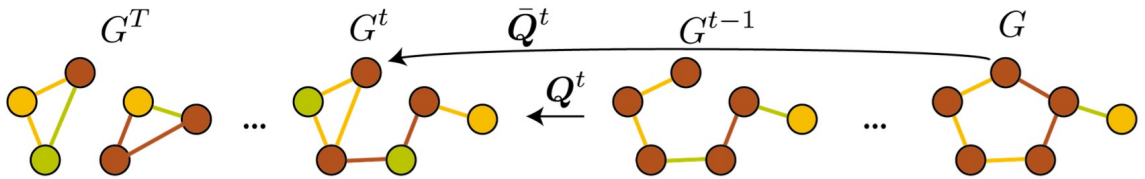
==

Predict the original graph based on noised graph

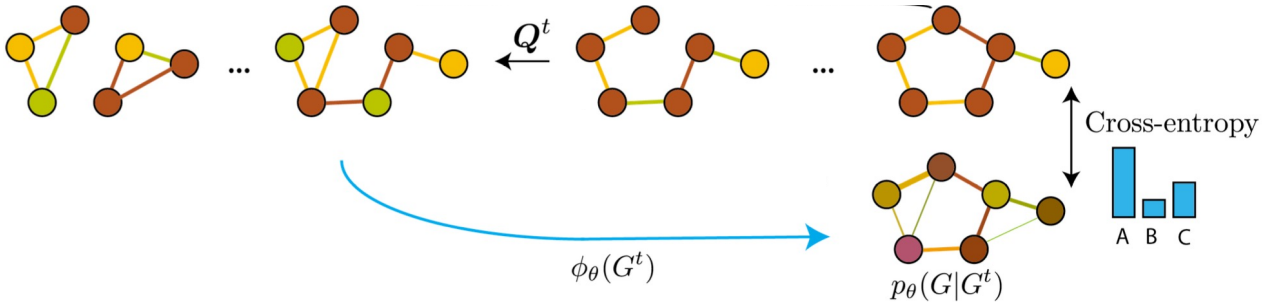
DiGress: Training objective (4)

- For training
  - 1) We generate noised sample G^t based on G^0
  - 2) Train the model by predicting the original node and edge type with G^t
    - Cross-entropy loss

1) Sample noises

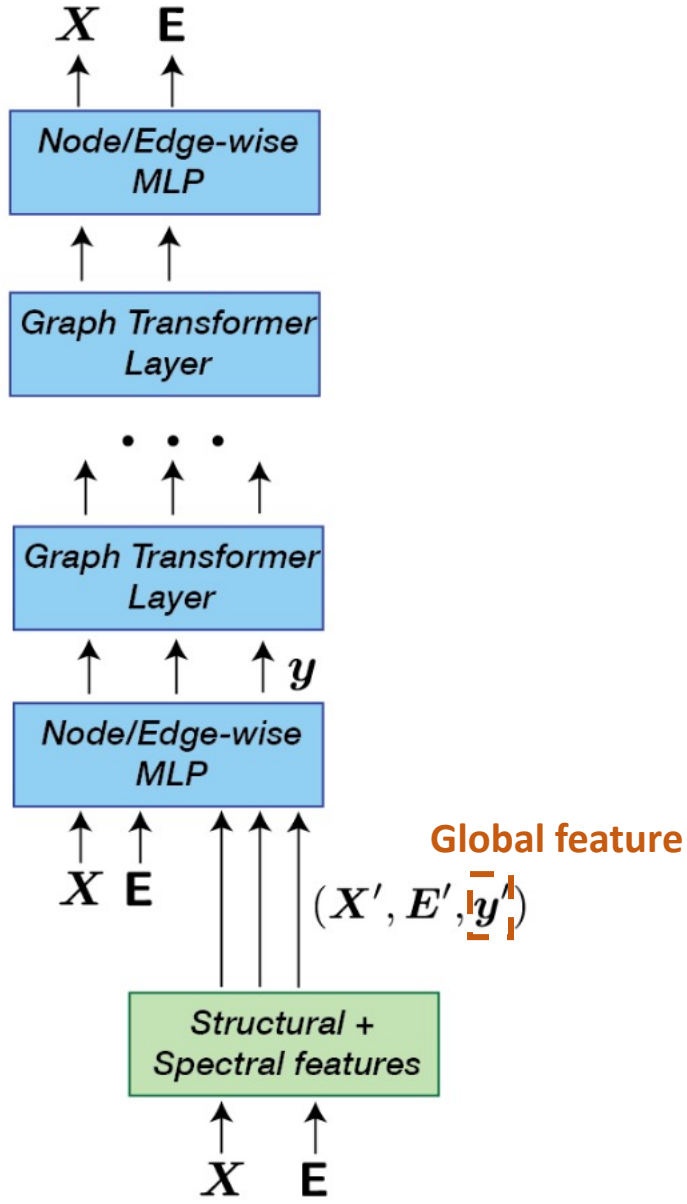


2) Training



DiGress: Training objective (5)

- Denoising network
  - Graph transformer network[1] equipped with FiLM[2]
  - Output the updated node and edge based on the input node and edge feature
  - Time step t is normalized and incorporated as a global feature inside y



[1] A generalization of transformer networks to graphs
[2] FiLM: Visual reasoning with a general conditioning layer

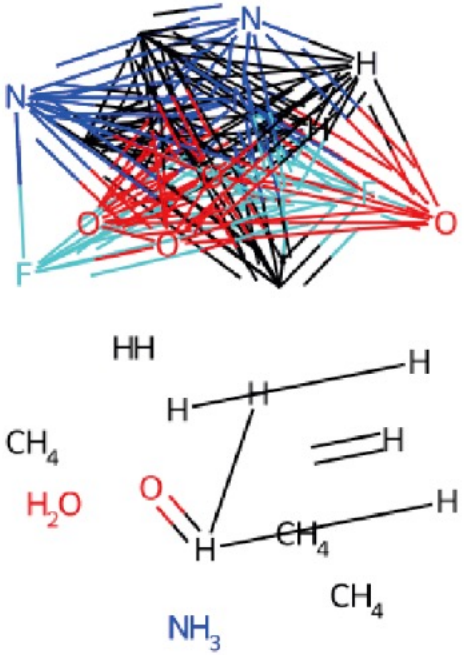
DiGress: Noise Model

- Uniform distribution is not applicable to graph
  - $Q^t = \alpha^t I + (1 - \alpha^t) 1_N 1_N' / N$
  - Graphs are usually sparse, so edge type is far from uniform
- DiGress: Transition matrix is based on the **frequency** of node/edge types in training set
  - The prior distribution that is close to the true data distribution makes training easier
  - $m_X \in \mathbb{R}^{a \times 1}, m_E \in \mathbb{R}^{b \times 1}$: frequency vector

t-step transition matrix:

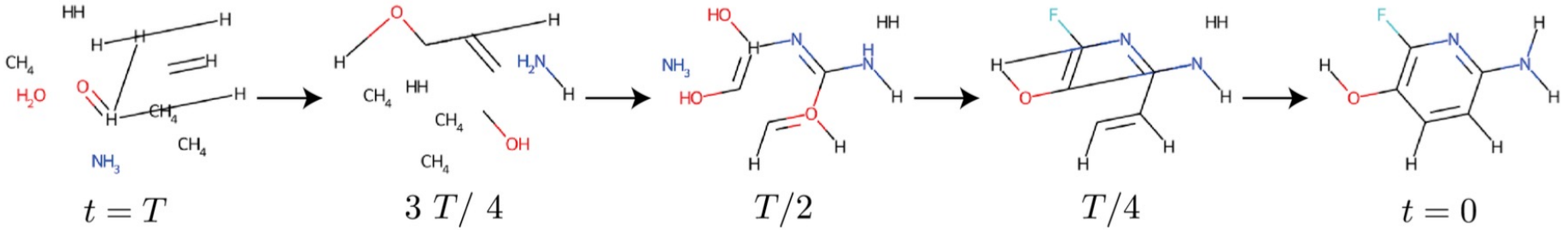
$$Q_X^t = \alpha^t I + (1 - \alpha^t) 1_a m_X'$$

$$Q_E^t = \alpha^t I + (1 - \alpha^t) 1_b m_E'$$



$t = T$

DiGress: Graph Generation



- Sample a random graph G^T based on m_X, m_E
- For $t = T$ for 1 do
  - Predict the node distribution and edge distribution of G^0
    - $p(x^0|G^t), p(e^0|G^t) = \text{Transformer}(G^t)$
  - Generate posterior node/edge distribution of the graph at $t-1$ step
    - $p\left(x_i^{t-1}\middle|G^t\right) = \sum_x p\left(x_i^{t-1}\middle|x_i^0 = x, x_i^t\right) p(x_i^0 = x|G^t)$
    - $p\left(e_{ij}^{t-1}\middle|G^t\right) = \sum_e p\left(e_{ij}^{t-1}\middle|e_{ij}^0 = e, e_{ij}^t\right) p(e_{ij}^0 = e|G^t)$
  - Sample graph at $t-1$ step
    - $G^{t-1} \sim \prod_{1\le i\le n} p(x_i^{t-1}|G^t) \prod_{1\le i,j\le n} p(e_{ij}^{t-1}|G^t)$

DiGress: Conditional Generation

- How to generate the graph conditional on a target property y_G
  - E.g., molecular property
- Classifier guidance algorithm[1]
  - Additionally train a classifier $g_\eta(G^t) = \hat{y}$
  - This regressor guides the unconditional diffusion model by modulating the predicted distribution at each sampling step

$$p(G^{t-1}|G^t) \Rightarrow p(G^{t-1}|G^t)p(y_G|G^{t-1})$$

[1] Deep unsupervised learning using nonequilibrium thermodynamics

Experiment: (1)

- Unconditional generation on non-molecule graph
  - Dataset:
    - 200 graphs sampled from SBM (200 nodes per graph)
    - 200 graphs sampled from planar graphs (64 nodes per graph)
  - Train the model on training set and compare the distribution between test set and the generated graphs
    - Degree distribution
    - Clustering coefficient
    - Orbit count

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%**

Experiment: (2)

- Unconditional generation on QM9 dataset
  - 100k molecules for training, 20k for validation and 13k for evaluating likelihood on a test set
  - Metric
    - NLL: Negative log-likelihood
    - Valid: Validity (measured by RDKit sanitization)
    - Unique: Uniqueness

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**

Experiment: (3)

- Conditional generation on QM9
  - Test set construction
    - Sample 100 molecules from the test set and retrieve their dipole moment μ and the highest occupied molecular orbit (HOMO)
  - Generate 10 molecules with DiGress conditional on μ, HOMO
  - Evaluation
    - For each generated molecule, use RdKit to produce conformer
    - Then use Psi4 to estimate the values of μ and HOMO
    - Report the MSE between the targets and the estimated values

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

Experiment: (4)

- Conditional generation on molecule dataset MOSES and GuacaMol
  - DiGress does not yet match the performance of SMILES and fragment-based methods

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

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

DiGress

- Adapt the unconditional/conditional discrete diffusion model to graph generation problem
  - Separate the diffusion on each node and edge feature
- Propose an effective noise model for graph
  - Based on the frequency of node and graph types
- Use a Transformer model as the denoising model
- Evaluate the performance on 5 benchmark datasets