Updates and Progress

Yogesh Verma
Doctoral Candidate
Aalto University

Updates

- Paper read: 32/1280
  - GraphDF: A Discrete Flow Model for Molecular Graph Generation
  - GraphNVP
  - GeoDiff
  - Categorical Normalizing flows
  - Dual use of artifcial-intelligence-powered drug discovery
  - Energy-Inspired Molecular Conformation Optimization [Reading]
  - Chemical-Reaction-Aware Molecule Representation Learning [Reading]

Updates

- Paper read: 32/1280
  - GraphDF: A Discrete Flow Model for Molecular Graph Generation
  - GraphNVP
  - GeoDiff
  - Categorical Normalizing flows
  - Dual use of artifcial-intelligence-powered drug discovery
  - Energy-Inspired Molecular Conformation Optimization [Reading]
  - Chemical-Reaction-Aware Molecule Representation Learning [Reading]
- CNF for generating valid molecules, coding, debugging.....
- Reversible SDEs for graphs
- Ideas in Stack: Molecular surface by 3D Zerneike Descriptors

Nearby Papers

- Categorical Normalizing flows
- GraphNVP
- GraphDF: A Discrete Flow Model for Molecular Graph Generation

Categorical Normalizing flows

- Normalizing flows for categorical data by encoding of categorical data in continuous space as a variational inference problem, we jointly optimize the continuous representation and the model likelihood.
- **Decoder:** Simplify the decoder by factorizing the decoder over latent variables: $p(\mathbf{x}|\mathbf{z}) = \Pi_i p(\mathbf{x}_i|\mathbf{z}_i)$. It means that we enforce independence between the categorical variables $\mathbf{x}_i$ given their learned continuous encoding $\mathbf{z}_i$
- **Encoder:** Consider is to encode each category by a logistic distribution with a learned mean and scaling. Encoder becomes $q(\mathbf{z}|\mathbf{x}) = \Pi_{i=1}^{S} g(\mathbf{z}_i|\mu(x_i), \sigma(x_i))$ and decoder likelihood can be calculated as

$$p(\mathbf{x}_i|\mathbf{z}_i) = \frac{\tilde{p}(\mathbf{x}_i)q(\mathbf{z}_i|\mathbf{x}_i)}{\sum_{\hat{x}} \tilde{p}(\hat{\mathbf{x}}_i)q(\mathbf{z}_i|\hat{\mathbf{x}}_i)} \tag{1}$$

3 step generation

- Given a graph $\mathcal{G}$, we model each node and edge as a separate categorical variable where the categories correspond to their discrete attributes. To represent the graph structure, i.e. between which pairs of nodes does or does not exist an edge, we add an extra category to the edges representing the missing or virtual edges.

3 step generation

- Given a graph $\mathcal{G}$, we model each node and edge as a separate categorical variable where the categories correspond to their discrete attributes. To represent the graph structure, i.e. between which pairs of nodes does or does not exist an edge, we add an extra category to the edges representing the missing or virtual edges.
- First step, we encode the nodes into continuous latent space, using Categorical Normalizing Flows. On those, we apply a group of coupling layers, which additionally use the adjacency matrix and the edge attributes.

3 step generation

- Given a graph $\mathcal{G}$, we model each node and edge as a separate categorical variable where the categories correspond to their discrete attributes. To represent the graph structure, i.e. between which pairs of nodes does or does not exist an edge, we add an extra category to the edges representing the missing or virtual edges.
- First step, we encode the nodes into continuous latent space, using Categorical Normalizing Flows. On those, we apply a group of coupling layers, which additionally use the adjacency matrix and the edge attributes.
- The second step incorporates the encoding edge attributes (not virtual edges) into latent space where coupling layers transform both the node and edge attribute variables.

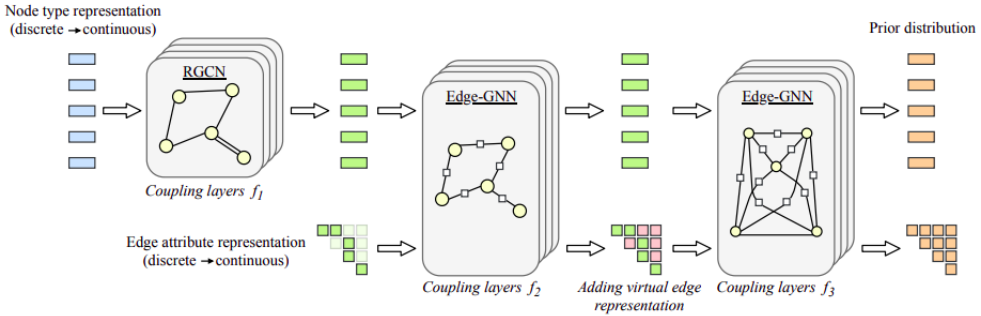
3 step generation

- Given a graph $\mathcal{G}$, we model each node and edge as a separate categorical variable where the categories correspond to their discrete attributes. To represent the graph structure, i.e. between which pairs of nodes does or does not exist an edge, we add an extra category to the edges representing the missing or virtual edges.
- First step, we encode the nodes into continuous latent space, using Categorical Normalizing Flows. On those, we apply a group of coupling layers, which additionally use the adjacency matrix and the edge attributes.
- The second step incorporates the encoding edge attributes (not virtual edges) into latent space where coupling layers transform both the node and edge attribute variables.
- Finally, we encode the virtual edges using Categorical Normalizing Flows

3 step generation

- Given a graph $\mathcal{G}$, we model each node and edge as a separate categorical variable where the categories correspond to their discrete attributes. To represent the graph structure, i.e. between which pairs of nodes does or does not exist an edge, we add an extra category to the edges representing the missing or virtual edges.
- First step, we encode the nodes into continuous latent space, using Categorical Normalizing Flows. On those, we apply a group of coupling layers, which additionally use the adjacency matrix and the edge attributes.
- The second step incorporates the encoding edge attributes (not virtual edges) into latent space where coupling layers transform both the node and edge attribute variables.
- Finally, we encode the virtual edges using Categorical Normalizing Flows
- During sampling, we first determine the general graph structure. Next, we reconstruct the edge attributes. Finally, we apply determine the node type.

Categorical Normalizing flows



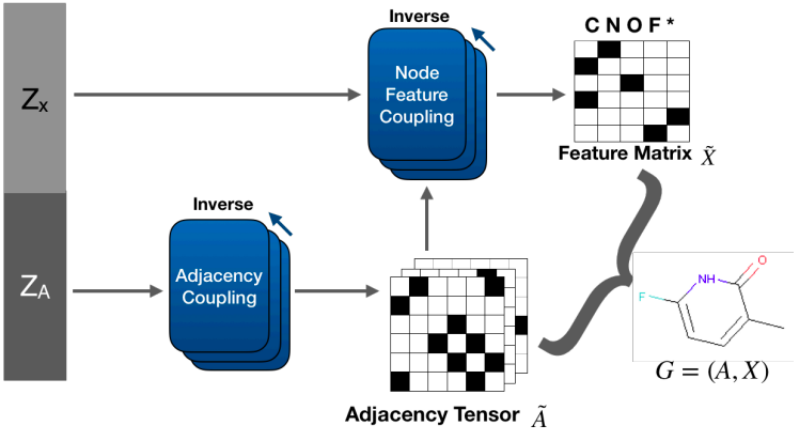
GraphNVP

- Decompose graph generation in two steps (i) Adjacency tensor (ii) Node Attributes
- Based on real NVP propose two types of reversible affine coupling layers; adjacency coupling layers and node feature coupling layers that transform the adjacency tensor and the feature matrix into latent representations.

GraphNVP

- Decompose graph generation in two steps (i) Adjacency tensor (ii) Node Attributes
- Based on real NVP propose two types of reversible affine coupling layers; adjacency coupling layers and node feature coupling layers that transform the adjacency tensor and the feature matrix into latent representations.
- 2-step generation process
  - Draw $z = concat(z_A, z_X)$ from prior, apply sequence of inverted adjacency coupling layers on z_A. Construct a discrete adjacency tensor by node-wise and edge-wise argmax.
  - Generate a feature matrix given z_X and the generated adjacency matrix into inverted node feature coupling layers to get final node feature matrix. Take node-wise argmax to get discrete feature matrix.

GraphNVP


Z_X
Z_A
Inverse
Node Feature Coupling
Inverse
Adjacency Coupling
C N O F *
Feature Matrix $\tilde{X}$
Adjacency Tensor $\tilde{A}$
$G = (A, X)$

GraphDF

- Molecular graphs by a sequential decision process by starting from an empty molecular graph G_0

GraphDF

- Molecular graphs by a sequential decision process by starting from an empty molecular graph G_0
- Generative model f_θ first generates a new node with type a_i based on a sampled latent variable z_i. Afterwards, the edges between the new node and all nodes in G_{i-1} are generated sequentially by f_θ. The above process is an autoregressive function of previously generated elements and repeats until no new edge is added at step i.

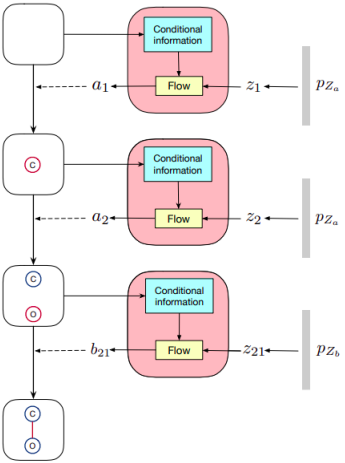
GraphDF

- Molecular graphs by a sequential decision process by starting from an empty molecular graph G_0
- Generative model f_θ first generates a new node with type a_i based on a sampled latent variable z_i. Afterwards, the edges between the new node and all nodes in G_{i-1} are generated sequentially by f_θ. The above process is an autoregressive function of previously generated elements and repeats until no new edge is added at step i.
- z_i is sampled from a multinomial distribution p_{Z_a} with trainable parameters $(\alpha_0, ..., \alpha_{k-1})$ as

$$p_{Z_a}(z_i = s) = \frac{\exp(\alpha_s)}{\sum_{t=0}^{k-1} \exp(\alpha_t)} \tag{2}$$

- Discrete flow model to reversibly map discrete latent variables to new nodes and edges

GraphDF


Conditional information
Flow
a_1
z_1
p_{Z_a}
C
Conditional information
Flow
a_2
z_2
p_{Z_a}
C
O
Conditional information
Flow
b_{21}
z_{21}
p_{Z_b}
C
O

Flowing Molecular Graphs (Better MoFlow) (Ours)

- Aim: Learn realistic molecular distributions and generating valid molecules
- We represent atom configurations by modelling local neighborhoods with coupled PDE CNFs in graph domain, Attack validity by accurate local densities

Flowing Molecular Graphs (Better MoFlow) (Ours)

- Aim: Learn realistic molecular distributions and generating valid molecules
- We represent atom configurations by modelling local neighborhoods with coupled PDE CNFs in graph domain, Attack validity by accurate local densities
- Given a molecule with a graph representation (connectivity C) $\mathcal{G} = (V, E)$, one can define a probability distribution at each node over the atomic vocabulary (Vocabulary of all the unique atoms spanning the whole molecule data-set) conditional over each node(atom) neighbours.

$$P(X) = \Pi_i^{N(V)} p(x_i | N_{J(i)}) \tag{3}$$

- Building on CNFs, we present flows on graphs specially applied in the molecular regime where we model the continuous time dynamics of random variables on graphs with respect to some conditionals over connectivity of the graph applied to graph structured data

Flowing Molecular Graphs (Better MoFlow)

- Given a set of vertices $\mathbf{X}$ for a graph (molecule), the goal is to learn the joint distribution $P(X)$ given by Eq.1 of the set of nodes.
- For continuous time dynamics of the set of variables X, by following Eq. 3,4 we formulate an ODE system as follows

$$\begin{bmatrix} \dot{\log}(\mathbf{p}(\mathbf{x_1}(\mathbf{t})|\mathbf{N_{J(1)}})) \\ \dot{\log}(\mathbf{p}(\mathbf{x_2}(\mathbf{t})|\mathbf{N_{J(2)}})) \\ \vdots \\ \dot{\log}(\mathbf{p}(\mathbf{x_n}(\mathbf{t})|\mathbf{N_{J(n)}})) \end{bmatrix} = \begin{bmatrix} -Tr(\frac{\partial \mathbf{f}(\mathbf{X}(\mathbf{t}),\mathbf{N_{J(1)}})}{\partial \mathbf{x_1}(\mathbf{t})}) \\ -Tr(\frac{\partial \mathbf{f}(\mathbf{X}(\mathbf{t}),\mathbf{N_{J(2)}})}{\partial \mathbf{x_2}(\mathbf{t})}) \\ \vdots \\ -Tr(\frac{\partial \mathbf{f}(\mathbf{X}(\mathbf{t}),\mathbf{N_{J(n)}})}{\partial \mathbf{x_n}(\mathbf{t})}) \end{bmatrix} \tag{4}$$

where each $\mathbf{x_i}(\mathbf{t})$ is i^{th} node, $\mathbf{N_{J(i)}}$ its neighbours and $\mathbf{x_i} \in \mathbf{X}$.

f and Optimization

- For starters, we consider f to be a deep NN taking input as the parent node and its neighbours.

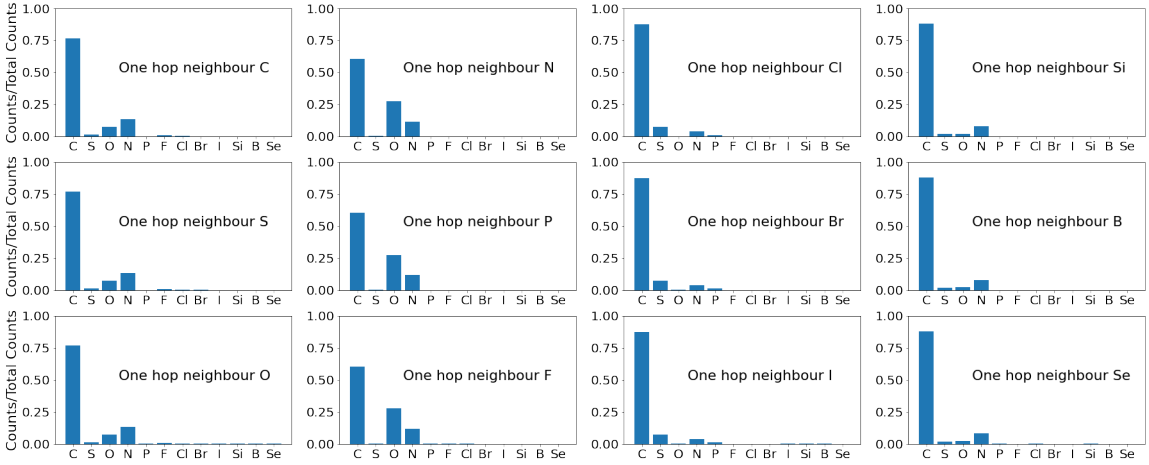
$$f = \text{NN}(X(t), N_{J(n)}) \tag{5}$$

- Minimize the $D_{KL}\ [p(x(t)|N_J, \theta)\ ||\ p(x^*|N_J)]$ or maximize likelihood (defined by Bernoulli which gives cross-entropy) to fit the flow based model, which can be formally written as

$$\mathcal{L}(\theta) = D_{KL}\ [p(x(t)|N_J, \theta)\ ||\ p(x^*|N_J)] \tag{6}$$

$$= \mathbb{E}_{p(x(t)|N_J,\theta)}\ [\log(p(x(t)|N_J, \theta)) - \log(p(x^*|N_J)] \tag{7}$$

Empirical prior for the atom neighborhood distributions



Reversible SDE on Graphs

SDE

An Ito SDE can be written as:

$$d\mathbf{X}_t = \mathbf{f}_t(\mathbf{X}_t)dt + \mathbf{g}_t(\mathbf{X}_t)d\mathbf{w} \tag{8}$$

where $\mathbf{f}_t$ is the drift coefficient, $\mathbf{g}_t$ is diffusion coefficient and $\mathbf{w}$ is standard weiner process.

SDE

An Ito SDE can be written as:

$$d\mathbf{X}_t = \mathbf{f}_t(\mathbf{X}_t)dt + \mathbf{g}_t(\mathbf{X}_t)d\mathbf{w} \tag{8}$$

where $\mathbf{f}_t$ is the drift coefficient, $\mathbf{g}_t$ is diffusion coefficient and $\mathbf{w}$ is standard weiner process. The reverse-time SDE for above can be written as (Ref)

$$d\mathbf{X}_t = [\mathbf{f}_t(\mathbf{X}_t) - \mathbf{g}_t^2 \nabla_{\mathbf{X}_t} \log p_t(\mathbf{X}_t)]d\tilde{t} + \mathbf{g}_t(\mathbf{X}_t)d\tilde{\mathbf{w}} \tag{9}$$

where $\tilde{w}$ is reverse-time standard wiener process and $d\tilde{t}$ is an infinitesimal negative time step.

SDE on Graphs

A graph $\mathbf{G}$ can represented by $\mathbf{G} = (\mathbf{X}, \mathbf{E})$ where $\mathbf{X}$ are the node features and $\mathbf{E}$ are the edges determining the connections.

SDE on Graphs

A graph $\mathbf{G}$ can represented by $\mathbf{G} = (\mathbf{X}, \mathbf{E})$ where $\mathbf{X}$ are the node features and $\mathbf{E}$ are the edges determining the connections.
The forward diffusion process can be represented as $\{\mathbf{G}_t = (\mathbf{X}_t)\}_{t=0}^{T}$ with continuous time variable $t \in [0, T]$ where $G_0 \sim p_{data}$ and $G_T \sim p_T$

$$d\mathbf{G}_t = \mathbf{f}_t(\mathbf{G}_t)dt + \mathbf{g}_t(\mathbf{G}_t)d\mathbf{w} \tag{10}$$

SDE on Graphs

A graph $\mathbf{G}$ can represented by $\mathbf{G} = (\mathbf{X}, \mathbf{E})$ where $\mathbf{X}$ are the node features and $\mathbf{E}$ are the edges determining the connections.
The forward diffusion process can be represented as $\{\mathbf{G}_t = (\mathbf{X}_t)\}_{t=0}^{T}$ with continuous time variable $t \in [0, T]$ where $G_0 \sim p_{data}$ and $G_T \sim p_T$

$$d\mathbf{G}_t = \mathbf{f}_t(\mathbf{G}_t)dt + \mathbf{g}_t(\mathbf{G}_t)d\mathbf{w} \tag{10}$$

Following the same analogy, the reverse process can be defined as

$$d\mathbf{G}_t = [\mathbf{f}_t(\mathbf{G}_t) - \mathbf{g}_t^2 \nabla_{\mathbf{G}_t} \log p_t(\mathbf{G}_t)]d\tilde{t} + \mathbf{g}_t(\mathbf{G}_t)d\tilde{\mathbf{w}} \tag{11}$$

Note: It requires computation of $\nabla_{\mathbf{G}_t} \log p_t(\mathbf{G}_t)$ which is expensive to compute.

System of SDEs

We propose reverse-time diffusion process equivalent to Eq.(5) modelled on X_t

$$d\mathbf{X}_t^i = [\mathbf{f}_{i,t}(\mathbf{X}_t^i, \mathbf{N}_{\mathbf{J(i)}}) - \mathbf{g}_{i,t}^2 \nabla_{\mathbf{X}_t^i} \log p_t(\mathbf{X}_t^i, \mathbf{N}_{\mathbf{J(i)}})]d\tilde{t} + \mathbf{g}_{i,t}(\mathbf{X}_t^i, \mathbf{N}_{\mathbf{J(i)}})d\tilde{\mathbf{w}} \tag{12}$$

where $\mathbf{g}_t = (\forall_i\ \mathbf{g}_{i,t})$, $\mathbf{f}_t(\mathbf{G}_t) = (\forall_i\ \mathbf{f}_{i,t}(\mathbf{X}_t^i, \mathbf{N}_{\mathbf{J(i)}}))$ and $\mathbf{N}_{\mathbf{J(i)}}$ are the 1-hop neighbours of i^{th}*node*. Each SDE in Eq.6 describes diffusion process for each component of $\mathbf{X}$ coupled through its neighbouring connectivity on graph.

System of SDEs

We propose reverse-time diffusion process equivalent to Eq.(5) modelled on X_t

$$d\mathbf{X}_t^i = [\mathbf{f}_{i,t}(\mathbf{X}_t^i, \mathbf{N}_{\mathbf{J(i)}}) - \mathbf{g}_{i,t}^2 \nabla_{\mathbf{X}_t^i} \log p_t(\mathbf{X}_t^i, \mathbf{N}_{\mathbf{J(i)}})]d\tilde{t} + \mathbf{g}_{i,t}(\mathbf{X}_t^i, \mathbf{N}_{\mathbf{J(i)}})d\tilde{\mathbf{w}} \tag{12}$$

where $\mathbf{g}_t = (\forall_i\ \mathbf{g}_{i,t})$, $\mathbf{f}_t(\mathbf{G}_t) = (\forall_i\ \mathbf{f}_{i,t}(\mathbf{X}_t^i, \mathbf{N}_{\mathbf{J(i)}}))$ and $\mathbf{N}_{\mathbf{J(i)}}$ are the 1-hop neighbours of i^{th}*node*. Each SDE in Eq.6 describes diffusion process for each component of $\mathbf{X}$ coupled through its neighbouring connectivity on graph.

Target: How to estimate $\nabla_{\mathbf{X}_t^i} \log p_t(\mathbf{X}_t^i, \mathbf{N}_{\mathbf{J(i)}})$ (score)?

Options for $\nabla_{\mathbf{X}_t^i} \log p_t(\mathbf{X}_t^i, \mathbf{N}_{\mathbf{J(i)}})$

Many options...

- NN to approximate it $s_\theta(\mathbf{X}_t^i, \mathbf{N}_{\mathbf{J(i)}}) \approx \nabla_{\mathbf{X}_t^i} \log p_t(\mathbf{X}_t^i, \mathbf{N}_{\mathbf{J(i)}})$

$$\mathbb{E}_{\mathbf{G}_t} \| s_\theta(\mathbf{X}_t^i, \mathbf{N}_{\mathbf{J(i)}}) - \nabla_{\mathbf{X}_t^i} \log p_t(\mathbf{X}_t^i, \mathbf{N}_{\mathbf{J(i)}}) \|^2 \tag{13}$$

Options for $\nabla_{\mathbf{X}_t^i} \log p_t(\mathbf{X}_t^i, \mathbf{N}_{\mathbf{J(i)}})$

Many options...

- NN to approximate it $s_\theta(\mathbf{X}_t^i, \mathbf{N}_{\mathbf{J(i)}}) \approx \nabla_{\mathbf{X}_t^i} \log p_t(\mathbf{X}_t^i, \mathbf{N}_{\mathbf{J(i)}})$

$$\mathbb{E}_{\mathbf{G}_t} \| s_\theta(\mathbf{X}_t^i, \mathbf{N}_{\mathbf{J(i)}}) - \nabla_{\mathbf{X}_t^i} \log p_t(\mathbf{X}_t^i, \mathbf{N}_{\mathbf{J(i)}}) \|^2 \tag{13}$$

- Approximate as a contribution from local structure as

$$\nabla_{\mathbf{X}_t^i} \log p_t(\mathbf{X}_t^i, \mathbf{N}_{\mathbf{J(i)}}) \approx s_\phi(\mathbf{X}_t^i) \cdot \sum_{j=0}^{N_J} s_\theta(\mathbf{X}_t^i, \mathbf{X}_t^j) \tag{14}$$

$$\mathbb{E}_{\mathbf{G}_t} \| s_\phi(\mathbf{X}_t^i) \cdot \sum_{j=0}^{N_J} s_\theta(\mathbf{X}_t^i, \mathbf{X}_t^j) - \nabla_{\mathbf{X}_t^i} \log p_t(\mathbf{X}_t^i, \mathbf{N}_{\mathbf{J(i)}}) \|^2 \tag{15}$$

Options for $\nabla_{\mathbf{X}_t^i} \log p_t(\mathbf{X}_t^i, \mathbf{N}_{\mathbf{J(i)}})$

- Use a $\mathcal{GP}$ formulation

$$\nabla_{\mathbf{X}_t^i} \log p_t(\mathbf{X}_t^i, \mathbf{N}_{\mathbf{J(i)}}) \sim \mathcal{GP}(\mu_\theta(\mathbf{X}_t^i, \mathbf{N}_{\mathbf{J(i)}}), \mathcal{K}(\phi_w(\mathbf{X}_t^i, \mathbf{N}_{\mathbf{J(i)}}), \phi_w(\mathbf{X}_t^j, \mathbf{N}_{\mathbf{J(j)}}))) \tag{16}$$

THANK YOU
FEEDBACK?