Updates and Progress

Yogesh Verma
Doctoral Candidate
Aalto University

Updates

- Paper read: 39/1280
  - Denoising Diffusion GANs
  - Scalable Gradients for Stochastic Differential Equations
  - Geometric Transformers for protein interface contact prediction
  - Reliable Categorical Variational Inference with Mixture of Discrete Normalizing Flows [Reading]
  - GRAND: Graph Neural Diffusion [Reading]

Updates

- Paper read: 39/1280
  - Denoising Diffusion GANs
  - Scalable Gradients for Stochastic Differential Equations
  - Geometric Transformers for protein interface contact prediction
  - Reliable Categorical Variational Inference with Mixture of Discrete Normalizing Flows [Reading]
  - GRAND: Graph Neural Diffusion [Reading]
- CNF for generating valid molecules, coding, debugging.....
- Reversible SDEs for graphs
- Ideas in Stack: Molecular surface by 3D Zerneike Descriptors

Proposed Method

- 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

Proposed Method

- 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.

$$P(G) = \prod_{\mathbf{v} \in V} p\left(\mathbf{v} \mid \mathrm{N}(\mathbf{v})\right) \tag{1}$$

- 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

Proposed Method

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

$$\frac{\partial \mathbf{v}}{\partial t} = \mathrm{f}(\mathbf{v}, \mathrm{N}(\mathbf{v}), t) \tag{2}$$

Proposed Method

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

$$\frac{\partial \mathbf{v}}{\partial t} = \mathrm{f}(\mathbf{v}, \mathrm{N}(\mathbf{v}), t) \tag{2}$$

- Then the change in log probability follows

$$\frac{\partial \log p(\mathbf{v}(t))}{\partial t} = -tr(\frac{\partial \mathrm{f}(\mathbf{v}, \mathrm{N}(\mathbf{v}), t)}{\partial \mathbf{v}(t)}) \tag{3}$$

f and Optimization

- Options for f
  - NN

$$\mathrm{f} = \mathrm{NN}(\mathbf{v}, N(\mathbf{v})) \tag{4}$$

  - RBF+NN: Transform the neighborhood into a distribution ($\sum_{\mathbf{v}_j \in \mathrm{N}(\mathbf{v}_i)} \phi(\mathbf{v}_i, \mathbf{v}_j)$), and then input the distribution into a NN

f and Optimization

- Options for f
  - NN

$$\mathrm{f} = \mathrm{NN}(\mathbf{v}, N(\mathbf{v})) \tag{4}$$

  - RBF+NN: Transform the neighborhood into a distribution ($\sum_{\mathbf{v}_j \in \mathrm{N}(\mathbf{v}_i)} \phi(\mathbf{v}_i, \mathbf{v}_j)$), and then input the distribution into a NN
- Fit the flow-based model to the samples by maximum likelihood estimation (by using label smoothing)

Alternatives to Argmax?

- Currently, using `argmax` at the end to transforming the final flow probability into Dirac-peaks giving the atom assignment.

Some Preliminary checks and results

Model	Reconstruction	Validity	Novelty	Diversity
NN	**100%**	**94∼98%**	**100%**	**100%**
RBF+NN	to do	to do	to do	to do
MoFloW	100%	50.3%	100%	99.99%
MRNN	100%	65.3%	99.8%	99.89%
GraphDF	100%	89.03%	99.8%	99.16%

Table: Random generation performance on ZINC250k dataset

Model	Reconstruction	Validity	Novelty	Diversity
NN	**100%**	**96.2%**	**100%**	**100%**
RBF+NN	to do	to do	to do	to do
GraphNVP	100%	83.1%	58.2%	99.2%
MoFlow	100%	88.96%	96.4%	98.53%
GraphDF	100%	82.67%	98.1%	97.62%

Table: Random generation performance on QM9 dataset

Alternatives to Argmax?

- Currently, using `argmax` at the end to transforming the final flow probability into Dirac-peaks giving the atom assignment.

Alternatives to Argmax?

- Currently, using `argmax` at the end to transforming the final flow probability into Dirac-peaks giving the atom assignment.
- Sampling
  - Sample from final flow probability many times such that low probability assigned atoms may also occur

Alternatives to Argmax?

- Currently, using `argmax` at the end to transforming the final flow probability into Dirac-peaks giving the atom assignment.
- Sampling
  - Sample from final flow probability many times such that low probability assigned atoms may also occur
  - Leads to many invalid molecules

Alternatives to Argmax?

- Currently, using `argmax` at the end to transforming the final flow probability into Dirac-peaks giving the atom assignment.
- Sampling
  - Sample from final flow probability many times such that low probability assigned atoms may also occur
  - Leads to many invalid molecules
- Conditional sampling based on overall likelihood
  - Likelihood based on Validity may lead to more valid molecules when sampling

Reversible SDE on Graphs

SDE

An Ito SDE can be written as:

$$d\mathbf{X}_t = \mathbf{f}_t(\mathbf{X}_t)dt + \mathbf{g}_t(\mathbf{X}_t)d\mathbf{w} \tag{5}$$

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:

$$d\mathbf{X}_t = \mathbf{f}_t(\mathbf{X}_t)dt + \mathbf{g}_t(\mathbf{X}_t)d\mathbf{w} \tag{5}$$

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)

$$d\mathbf{X}_t = [\mathbf{f}_t(\mathbf{X}_t) - \mathbf{g}_t^2 \nabla_{\mathbf{X}_t} \log p_t(\mathbf{X}_t)]d\tilde{t} + \mathbf{g}_t(\mathbf{X}_t)d\tilde{\mathbf{w}} \tag{6}$$

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 $\{\mathbf{G}_t = (\mathbf{X}_t)\}_{t=0}^{T}$ [assuming $\mathbf{E}$ remain same during time] with continuous time variable $t \in [0, T]$ where $G_0 \sim p_{data}$ and $G_T \sim p_T$

$$d\mathbf{G}_t = \mathbf{f}_t(\mathbf{G}_t)dt + \mathbf{g}_t(\mathbf{G}_t)d\mathbf{w} \tag{7}$$

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 $\{\mathbf{G}_t = (\mathbf{X}_t)\}_{t=0}^{T}$ [assuming $\mathbf{E}$ remain same during time] with continuous time variable $t \in [0, T]$ where $G_0 \sim p_{data}$ and $G_T \sim p_T$

$$d\mathbf{G}_t = \mathbf{f}_t(\mathbf{G}_t)dt + \mathbf{g}_t(\mathbf{G}_t)d\mathbf{w} \tag{7}$$

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

$$d\mathbf{G}_t = [\mathbf{f}_t(\mathbf{G}_t) - \mathbf{g}_t^2 \nabla_{\mathbf{G}_t} \log p_t(\mathbf{G}_t)]d\tilde{t} + \mathbf{g}_t(\mathbf{G}_t)d\tilde{\mathbf{w}} \tag{8}$$

SDE on Graphs

- The reverse process can be defined as

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

SDE on Graphs

- The reverse process can be defined as

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

- Assuming a 1-neighbourhood where local effects are strong, we can factorize or decompose $\mathbf{f}_t(\mathbf{G}_t)$ as contribution from local regions

$$\mathbf{f}_t(\mathbf{G}_t) = \mathtt{Agg}_{\mathbf{x} \in X}(\mathbf{f}_t(\mathbf{x}, N(\mathbf{x}))) \tag{10}$$

SDE on Graphs

- By using chain rule of differentiation we can write

$$\nabla_{\mathbf{G}_t} \log p_t(\mathbf{G}_t) = \frac{\partial \log p_t(\mathbf{G}_t)}{\partial \mathbf{G}_t} = \sum_{\mathbf{x} \in X} \frac{\partial \log p_t(\mathbf{G}_t)}{\partial \mathbf{x}_t} \frac{\partial \mathbf{x}_t}{\partial \mathbf{G}_t} = \sum_{\mathbf{x} \in X} \nabla_{\mathbf{x}_t} \log p_t(\mathbf{G}_t) \cdot \frac{\partial \mathbf{x}_t}{\partial \mathbf{G}_t} \quad (11)$$

- $\frac{\partial \mathbf{x}_t}{\partial \mathbf{G}_t}$ similar to gradient of node w.r.t graph (maybe a connection to laplacian)

SDE on Graphs

- By using chain rule of differentiation we can write

$$\nabla_{\mathbf{G}_t} \log p_t(\mathbf{G}_t) = \frac{\partial \log p_t(\mathbf{G}_t)}{\partial \mathbf{G}_t} = \sum_{\mathbf{x} \in X} \frac{\partial \log p_t(\mathbf{G}_t)}{\partial \mathbf{x}_t} \frac{\partial \mathbf{x}_t}{\partial \mathbf{G}_t} = \sum_{\mathbf{x} \in X} \nabla_{\mathbf{x}_t} \log p_t(\mathbf{G}_t) \cdot \frac{\partial \mathbf{x}_t}{\partial \mathbf{G}_t} \quad (11)$$

- $\frac{\partial \mathbf{x}_t}{\partial \mathbf{G}_t}$ similar to gradient of node w.r.t graph (maybe a connection to laplacian)
- One can decompose $\mathbf{g}_t(\mathbf{G}_t)$ similarly as,

$$\mathbf{g}_t^2 \nabla_{\mathbf{G}_t} \log p_t(\mathbf{G}_t) = \sum_{\mathbf{x} \in X} \mathbf{g}_t^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{G}_t) \cdot \frac{\partial \mathbf{x}_t}{\partial \mathbf{G}_t} \quad (12)$$

SDE on Graphs

- By using chain rule of differentiation we can write

$$\nabla_{\mathbf{G}_t} \log p_t(\mathbf{G}_t) = \frac{\partial \log p_t(\mathbf{G}_t)}{\partial \mathbf{G}_t} = \sum_{\mathbf{x} \in X} \frac{\partial \log p_t(\mathbf{G}_t)}{\partial \mathbf{x}_t} \frac{\partial \mathbf{x}_t}{\partial \mathbf{G}_t} = \sum_{\mathbf{x} \in X} \nabla_{\mathbf{x}_t} \log p_t(\mathbf{G}_t) \cdot \frac{\partial \mathbf{x}_t}{\partial \mathbf{G}_t} \quad (11)$$

- $\frac{\partial \mathbf{x}_t}{\partial \mathbf{G}_t}$ similar to gradient of node w.r.t graph (maybe a connection to laplacian)
- One can decompose $\mathbf{g}_t(\mathbf{G}_t)$ similarly as,

$$\mathbf{g}_t^2 \nabla_{\mathbf{G}_t} \log p_t(\mathbf{G}_t) = \sum_{\mathbf{x} \in X} \mathbf{g}_t^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{G}_t) \cdot \frac{\partial \mathbf{x}_t}{\partial \mathbf{G}_t} \quad (12)$$

- Now once can use above equations in Eq.9 and decompose it for each $\mathbf{x} \in X$ as

$$d\mathbf{x}_t = [\mathbf{f}_t(\mathbf{x}, N(\mathbf{x})) - \mathbf{g}_t^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{G}_t) \cdot \frac{\partial \mathbf{x}_t}{\partial \mathbf{G}_t}] d\tilde{t} + \mathbf{g}_t(\mathbf{x}, N(\mathbf{x})) d\tilde{\mathbf{w}} \quad (13)$$

SDE on Graphs

- Now, as we have assumed a factorized of $\mathbf{f}_t(\mathbf{G}_t)$ and $\mathbf{g}_t(\mathbf{G}_t)$ pertaining towards local contributions (local neighbours (L.N.)) as a result $p_t(\mathbf{G}_t)$ will factorize, we can write for a single node as

$$\frac{\partial \log p_t(\mathbf{G}_t)}{\partial \mathbf{x}_t} = \frac{\partial \log p_t(\mathbf{x}_{1,t}, \mathbf{x}_{2,t}, ...\mathbf{x}_{n,t})}{\partial \mathbf{x}_{1,t}} = \frac{\partial \log p_t(\mathbf{x}_{1,t}, \overbrace{\mathbf{x}_{2,t}, ..., \mathbf{x}_{k,t}}^{\text{L.N. (1-hop)}})}{\partial \mathbf{x}_{1,t}} \tag{14}$$

SDE on Graphs

- Now, as we have assumed a factorized of $\mathbf{f}_t(\mathbf{G}_t)$ and $\mathbf{g}_t(\mathbf{G}_t)$ pertaining towards local contributions (local neighbours (L.N.)) as a result $p_t(\mathbf{G}_t)$ will factorize, we can write for a single node as

$$\frac{\partial \log p_t(\mathbf{G}_t)}{\partial \mathbf{x}_t} = \frac{\partial \log p_t(\mathbf{x}_{1,t}, \mathbf{x}_{2,t}, ...\mathbf{x}_{n,t})}{\partial \mathbf{x}_{1,t}} = \frac{\partial \log p_t(\mathbf{x}_{1,t}, \overbrace{\mathbf{x}_{2,t}, ..., \mathbf{x}_{k,t}}^{\text{L.N. (1-hop)}})}{\partial \mathbf{x}_{1,t}} \tag{14}$$

- It will modify the equation as

$$d\mathbf{x}_t = [\mathbf{f}_t(\mathbf{x}, N(\mathbf{x})) - \mathbf{g}_t^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t, \overbrace{\mathbf{x}_{k,t}, ..., \mathbf{x}_{s,t}}^{\text{L.N. (1-hop)}}) \cdot \frac{\partial \mathbf{x}_t}{\partial \mathbf{G}_t}] d\tilde{t} + \mathbf{g}_t(\mathbf{x}, N(\mathbf{x})) d\tilde{\mathbf{w}} \tag{15}$$

Optimize

- Use score matching [Hyvärinen 2005] to compute the scores and evaluate the reverse SDE

$$\psi(\mathbf{X}_t, \theta) = \sum_{\mathbf{x} \in X} \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t, \overbrace{\mathbf{x}_{k,t}, ..., \mathbf{x}_{s,t}}^{\text{L.N. (1-hop)}}) \cdot \frac{\partial \mathbf{x}_t}{\partial \mathbf{G}_t} \tag{16}$$

Optimize

- Use score matching [Hyvärinen 2005] to compute the scores and evaluate the reverse SDE

$$\psi(\mathbf{X}_t, \theta) = \sum_{\mathbf{x} \in X} \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t, \overbrace{\mathbf{x}_{k,t}, ..., \mathbf{x}_{s,t}}^{\text{L.N. (1-hop)}}) \cdot \frac{\partial \mathbf{x}_t}{\partial \mathbf{G}_t} \tag{16}$$

$$J(\theta) = \int_{\mathbf{X}_t} p_D(\mathbf{X}) \sum_{\mathbf{x} \in X} \left[\partial_i \psi(\mathbf{x}_t, \theta) + \frac{1}{2} \psi(\mathbf{x}_t, \theta)^2 \right] \tag{17}$$

Optimize

- Use score matching [Hyvärinen 2005] to compute the scores and evaluate the reverse SDE

$$\psi(\mathbf{X}_t, \theta) = \sum_{\mathbf{x} \in X} \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t, \overbrace{\mathbf{x}_{k,t}, ..., \mathbf{x}_{s,t}}^{\text{L.N. (1-hop)}}) \cdot \frac{\partial \mathbf{x}_t}{\partial \mathbf{G}_t} \tag{16}$$

$$J(\theta) = \int_{\mathbf{X}_t} p_D(\mathbf{X}) \sum_{\mathbf{x} \in X} \left[\partial_i \psi(\mathbf{x}_t, \theta) + \frac{1}{2} \psi(\mathbf{x}_t, \theta)^2 \right] \tag{17}$$

$$\partial_i \psi(\mathbf{x}_t, \theta) = \sum_{\mathbf{x} \in X} \left(\frac{\partial^2 \log p_t(\mathbf{x}_t, \mathbf{x}_{k,t}, ..., \mathbf{x}_{s,t})}{\partial^2 \mathbf{x}_t} \cdot \frac{\partial \mathbf{x}_t}{\partial \mathbf{G}_t} + \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t, \mathbf{x}_{k,t}, ..., \mathbf{x}_{s,t}) \cdot \frac{\partial^2 \mathbf{x}_t}{\partial \mathbf{G}_t \partial \mathbf{x}_t} \right)$$

- Use the $\nabla_\theta J(\theta)$ to compute gradients and optimize

THANK YOU
FEEDBACK?