# Updates and Progress

Yogesh Verma
Doctoral Candidate
Aalto University

# Updates

- Paper read: 65/1280
  - Neural Sheaf Diffusion
  - Neural Controlled Differential Equations for Irregular Time Series
  - Neural SDEs as Infinite-Dimensional GANs
  - Variational Neural Cellular Automata
  - Generative Coarse-Graining of Molecular Conformations [Reading]
  - Critical points in Quantum Generative Models [Reading]

# Updates

- Paper read: 65/1280
  - Neural Sheaf Diffusion
  - Neural Controlled Differential Equations for Irregular Time Series
  - Neural SDEs as Infinite-Dimensional GANs
  - Variational Neural Cellular Automata
  - Generative Coarse-Graining of Molecular Conformations [Reading]
  - Critical points in Quantum Generative Models [Reading]
- Reversible SDEs for graphs
- TO DO: Meeting with External Supervisor (CS Doctoral Support Program)

# ICML'22

- Path-aware and structure-preserving generation of synthetically accessible molecules
- LIMO: Latent Inceptionism for Targeted Molecule Generation
- Generating 3D Molecules for Target Protein Binding
- Equivariant Diffusion for Molecule Generation in 3D
- 3D Infomax improves GNNs for Molecular Property Prediction
- EquiBind: Geometric Deep Learning for Drug Binding Structure Prediction
- Generative Coarse-Graining of Molecular Conformations
- Molecular Graph Representation Learning via Heterogeneous Motif Graph Construction
- 3DLinker: An E(3) Equivariant Variational Autoencoder for Molecular Linker Design
- Pocket2Mol: Efficient Molecular Sampling Based on 3D Protein Pockets
- Antibody-Antigen Interface Design via Hierarchical Structure Refinement
- Biological Sequence Design with GFlowNets
- Maximum Likelihood Training for Score-based Diffusion ODEs by High Order Denoising Score Matching
- Diffusion bridges vector quantized variational autoencoders
- Nonlinear Feature Diffusion on Hypergraphs
- Learning to Solve PDE-constrained Inverse Problems with Graph Networks
- Robust SDE-based variational formulations for solving linear PDEs via deep learning
- Equivariant Quantum Graph Circuits
- Quantum-Inspired Algorithms from Randomized Numerical Linear Algebra

# Reversible SPDE on Graphs for Conformer Generation

# Before we go,

- Learn the mapping from 2D molecular graph to 3D conformers of that molecule
- Assumptions: Sparsifying on graph structure, locality
- Inputs: Coordinate embeddings ($\mathbf{x}_i$,$\mathbf{X}$)
- Output: Coordinate embeddings
- Method: Reversible SDEs

# SDE

An Ito SDE can be written as:

$$d\mathbf{X}(t) = \mathbf{f}(\mathbf{X}(t))dt + \mathbf{g}(\mathbf{X}(t))d\mathbf{w} \tag{1}$$

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

# SDE

An Ito SDE can be written as:

$$d\mathbf{X}(t) = \mathbf{f}(\mathbf{X}(t))dt + \mathbf{g}(\mathbf{X}(t))d\mathbf{w} \tag{1}$$

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

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

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 $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{X})$ where $\mathbf{X}$ are the node ($\mathbf{V}$) features and $\mathbf{E}$ are the edges determining the connections.

# SDE on Graphs

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

$$d\mathbf{X}(t) = \mathbf{f}(\mathbf{X}(t))dt + \mathbf{g}(\mathbf{X}(t))d\mathbf{w} \tag{3}$$

# SDE on Graphs

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

$$d\mathbf{X}(t) = \mathbf{f}(\mathbf{X}(t))dt + \mathbf{g}(\mathbf{X}(t))d\mathbf{w} \tag{3}$$

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

$$d\mathbf{X}(t) = [\mathbf{f}(\mathbf{X}(t)) - \mathbf{g}^2 \nabla_{\mathbf{X}(t)} \log p_t(\mathbf{X}(t))]d\tilde{t} + \mathbf{g}(\mathbf{X}(t))d\tilde{\mathbf{w}} \tag{4}$$

# SDE on Graphs

- Assuming a 1-neighbourhood where local effects are strong, we can factorize or decompose $\mathbf{f}(\mathbf{X}(t))$ as contribution from local regions ($\mathbf{x}_i(t)$ is the node features of $i$ node at time t, i = 1, . . . , M)

$$\mathbf{f}(\mathbf{X}(t)) = \mathtt{Agg}_{\mathbf{i}\in V}(\mathbf{f}(\mathbf{x}_i(t), \mathbf{x}_{\mathcal{N}(i)}(t))) \tag{5}$$

# SDE on Graphs

- Assuming a 1-neighbourhood where local effects are strong, we can factorize or decompose $\mathbf{f}(\mathbf{X}(t))$ as contribution from local regions ($\mathbf{x}_i(t)$ is the node features of $i$ node at time t, i = 1, . . . , M)

$$\mathbf{f}(\mathbf{X}(t)) = \mathtt{Agg}_{\mathbf{i}\in V}(\mathbf{f}(\mathbf{x}_i(t), \mathbf{x}_{\mathcal{N}(i)}(t))) \tag{5}$$

- By using chain rule of differentiation and factorization we can write

$$\nabla_{\mathbf{X}(t)} \log p_t(\mathbf{X}(t)) = \frac{\partial \log p_t(\mathbf{X}(t))}{\partial \mathbf{X}(t)} := \sum_{i\in V} \sum_{j\in\{i\}\cup\mathcal{N}(i)} \frac{\partial \log p_t(\mathbf{x}_i(t)|\mathbf{x}_{\mathcal{N}(i)}(t)))}{\partial \mathbf{x}_j(t)} \tag{6}$$

# SDE on Graphs

- One can decompose $\mathbf{g}(\mathbf{X}(t))$ similarly as,

$$\mathbf{g}^2 \nabla_{\mathbf{X}(t)} \log p_t(\mathbf{X}(t)) = \sum_{i \in V} \mathbf{g}^2 \sum_{j \in \{i\} \cup \mathcal{N}(i)} \nabla_{\mathbf{x}_j(t)} p_t(\mathbf{x}_i(t) | \mathbf{x}_{\mathcal{N}(i)}(t)) \tag{7}$$

# SDE on Graphs

- One can decompose $\mathbf{g}(\mathbf{X}(t))$ similarly as,

$$\mathbf{g}^2 \nabla_{\mathbf{X}(t)} \log p_t(\mathbf{X}(t)) = \sum_{i \in V} \mathbf{g}^2 \sum_{j \in \{i\} \cup \mathcal{N}(i)} \nabla_{\mathbf{x}_j(t)} p_t(\mathbf{x}_i(t) | \mathbf{x}_{\mathcal{N}(i)}(t)) \tag{7}$$

- Now one can use above equations in Eq.9 and decompose it for each $\mathbf{x} \in X$ as (Different noise for each)

$$d\mathbf{x}_i(t) = [\mathbf{f}(\mathbf{x}_i(t), \mathbf{x}_{\mathcal{N}(i)}(t) - \mathbf{g}^2 \sum_{j \in \{i\} \cup \mathcal{N}(i)} \nabla_{\mathbf{x}_j(t)} p_t(\mathbf{x}_i(t) | \mathbf{x}_{\mathcal{N}(i)}(t))] d\tilde{t} + \mathbf{g}(\mathbf{x}_i(t), \mathbf{x}_{\mathcal{N}(i)}(t)) d\tilde{\mathbf{w}}$$

# Solving Reverse SDE

- Solving reverse SDE, we need the score function and the final distribution $p_T$. (Note that notation is not adapted, and x can be extended to previous notation)

# Solving Reverse SDE

- Solving reverse SDE, we need the score function and the final distribution $p_T$. (Note that notation is not adapted, and x can be extended to previous notation)
- One can estimate $\nabla_x \log p_t(x)$, as time-dependent-score-based-model as $s_\theta(x, t) \approx \nabla_x \log p_t(x)$

# Solving Reverse SDE

- Solving reverse SDE, we need the score function and the final distribution $p_T$. (Note that notation is not adapted, and x can be extended to previous notation)
- One can estimate $\nabla_x \log p_t(x)$, as time-dependent-score-based-model as $s_\theta(x, t) \approx \nabla_x \log p_t(x)$
- Training objective can be a continuous weighted combination of Fisher divergences, given by

$$\mathrm{E}_{t\in\mathcal{U}(0,T)}\mathrm{E}_{p_t(x)}\left[\lambda(t)||\nabla_x \log p_t(x) - s_\theta(x, t)||_2^2\right] \tag{8}$$

Other methods:

- Variational lower bound on the log-likelihood ELBO (GeoDiff,Equivariant Diffusion for 3D molecules )

# How to do?

- Model the score when training with score matching
- Sampling with score-based MCMC like Langevin MCMC, HMC (Similar to multiple noise levels for greater accuracy in low density regions as well in Song et al. 2020)

# Factorized score matching

- Since, there occurs a factorization in the probability due to local neighbourhood dependence. This also leads to factorized score matching as we now only need to approximate $\sum_{j \in \{i\} \cup \mathcal{N}(i)} \nabla_{\mathbf{x}_j(t)} p_t(\mathbf{x}_i(t) | \mathbf{x}_{\mathcal{N}(i)}(t))$ for node $i$

# Factorized score matching

- Since, there occurs a factorization in the probability due to local neighbourhood dependence. This also leads to factorized score matching as we now only need to approximate $\sum_{j \in \{i\} \cup \mathcal{N}(i)} \nabla_{\mathbf{x}_j(t)} p_t(\mathbf{x}_i(t) | \mathbf{x}_{\mathcal{N}(i)}(t))$ for node $i$
- Plus points:
  - Less dimensionality of the score conditional $\rightarrow$ easy to accurately approximate in contrast to currently used techniques on images

# Factorized score matching

- Since, there occurs a factorization in the probability due to local neighbourhood dependence. This also leads to factorized score matching as we now only need to approximate $\sum_{j \in \{i\} \cup \mathcal{N}(i)} \nabla_{\mathbf{x}_j(t)} p_t(\mathbf{x}_i(t) | \mathbf{x}_{\mathcal{N}(i)}(t))$ for node $i$
- Plus points:
  - Less dimensionality of the score conditional $\rightarrow$ easy to accurately approximate in contrast to currently used techniques on images
  - Langevin MCMC sampling giving structure constrained sampling (can also be extended to HMC sampling)

# Modular Adjoints over SDE

- One can extend the modular joints from previous project to modular joints over SDE

# Modular Adjoints over SDE

- One can extend the modular joints from previous project to modular joints over SDE
- Leads to Augmented Diffusion, Drift for each node as we decompose $f$ and $g$

# Conditional Generation

- The above can be extended to conditional generation, where we have a property scalar or vector $y$ associated
- In practice, this means learning to predict the conditional score function $\nabla_{X(t)} p_t(X(t)|y)$

# Conditional Generation

- The above can be extended to conditional generation, where we have a property scalar or vector $y$ associated
- In practice, this means learning to predict the conditional score function $\nabla_{X(t)} p_t(X(t)|y)$
- By using Bayes rule we have:

$$p(X(t) \mid y) = \frac{p(y|X(t))p(X(t))}{p(y)} \tag{9}$$

$$\nabla_{X(t)} p(X(t) \mid y) = \nabla_{X(t)} p(y|X(t)) + \nabla_{X(t)} p(X(t)) \tag{10}$$

- We can further decompose it using prescribed previous decomposition.

# Conditional Generation

- $p(y|X(t))$ is exactly what classifiers and other discriminative models try to fit. But in our case it would be a regressive one as property vector $y$ is a continuous one. (Energy, solubility, etc.)
- One can also consider Brownian bridges for property generation as well

# Conditional Generation

- $p(y|X(t))$ is exactly what classifiers and other discriminative models try to fit. But in our case it would be a regressive one as property vector $y$ is a continuous one. (Energy, solubility, etc.)
- One can also consider Brownian bridges for property generation as well
- Usually, scaling the conditioning term by a factor leads to better results

$$\nabla_{X(t)} p_\gamma(X(t) \mid y) = \gamma \nabla_{X(t)} p(y|X(t)) + \nabla_{X(t)} p(X(t)) \tag{11}$$

- Combining the generative model with a regressive model, would lead to it

THANK YOU
FEEDBACK?