Graph Neural Networks: Somewhere between Spectral Graph Processing and Message Passing Algorithms

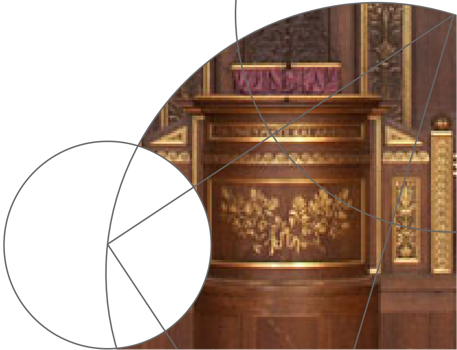
Summer School on Geometric Deep Learning

Raghavendra Selvan

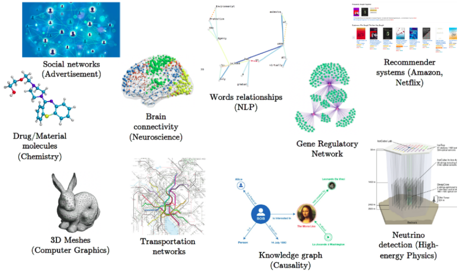
Assistant Professor
Dept. of Computer Science (ML Section)
Dept. of Neuroscience (Kiehn Lab)
University of Copenhagen

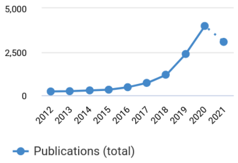
raghav@di.ku.dk
`https://raghavian.github.io`
@raghavian

August 18, 2021



Motivation


Social networks (Advertisement)
Drug/Material molecules (Chemistry)
Brain connectivity (Neuroscience)
Words relationships (NLP)
Gene Regulatory Network
Recommender systems (Amazon, Netflix)
3D Meshes (Computer Graphics)
Transportation networks
Knowledge graph (Causality)
Neutrino detection (High-energy Physics)
[1]



- Geometric deep learning $\equiv$ Non-Euclidean DL
- Heterogenous graph-structured data are non-Euclidean data
- Graph-based learning has taken off in the past 3-4 years
- Connections to other domains such as probabilistic graphical models, spectral theory

(From dimensions.ai)

[1] https://graphdeeplearning.github.io/project/spatial-convnets/

Graph Neural Networks (GNNs): Overview

Let's start in known waters

Laplacian

Laplacian on different domains means (almost) the same

- **Laplacian**: Second order differential operator
  (Tells you how smoothly the function is changing in its domain)
- In Euclidean space: $\Delta f = \nabla \cdot \nabla f$
- Generalized to Reimannian manifolds $\rightarrow$ Laplace-Beltrami operator
- When specified for graphs (discrete grids) $\rightarrow$ Graph Laplacian

Graph Laplacian

Consider graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$ with vertex set $\mathcal{V} : |\mathcal{V}| = N$, $\mathcal{E}$ set of edges and $\mathbf{A} \in \mathbb{R}^{N \times N}$: adjacency matrix, the graph Laplcian is simply given as:

$$\Delta = \mathbf{D} - \mathbf{A} \tag{1}$$

$\mathbf{D} \in \mathbb{R}^{N \times N}$ is the degree matrix with $D_{ii} = \sum_j A_{ij}$

Laplacian eigenfunctions form an orthonormal Fourier basis

The Symmetric Normalized Graph Laplacian can be factorized as:

$$\mathbf{L} = \mathbf{I}_N - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^T \qquad (2)$$

where, $\mathbf{\Phi} = (\phi_0, \phi_1 \dots \phi_{N-1}) \in \mathbb{R}^{N\times N}$ are the orthogonal eigenfunctions forming the Fourier basis, with corresponding eigenvalues in the diagonal matrix $\mathbf{\Lambda} = \mathrm{diag}(\lambda_0, \lambda_1, \dots, \lambda_{N-1})$

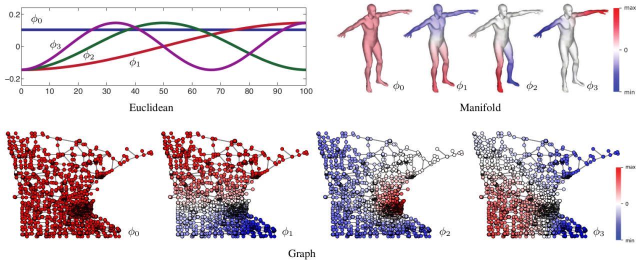


Figure from Geometric deep learning: Going beyond Euclidean data. Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, Pierre Vandergheynst 2016

Convolution on graphs in Fourier domain

Consider a signal operating on the nodes, $\mathbf{x} \in \mathbb{R}^N$ and a filter g_Θ parameterized by $\Theta \in \mathbb{R}^N$, the graph convolution is given as:

$$g_\Theta \star \mathbf{x} = (\mathbf{\Phi} g_\Theta)(\mathbf{\Phi}^T \mathbf{x}) \tag{3}$$

Recollect that: $\mathbf{L} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^T$ and $\mathbf{\Phi}^T\mathbf{x}$ is the Fourier transform of $\mathbf{x}$

Turns out that $g_\Theta = \mathbf{\Lambda}$ and not very useful

- Non-localized, non-parametric (all free parameters)
- Computationally expensive

So, use a polynomial parameterization for localized filters with:

$$g_\Theta(\mathbf{\Lambda}) = \sum_{k=0}^{K-1} \theta_k \mathbf{\Lambda}^k \tag{4}$$

Approximating Convolution on graphs with truncated Chebyshev polynomials

Recursive formulation for fast filtering:

$$g_\Theta(\mathbf{\Lambda}) = \sum_{k=0}^{K-1} \theta_k \mathbf{\Lambda}^k \approx \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\mathbf{\Lambda}}) \qquad (5)$$

with $\tilde{\mathbf{\Lambda}} = \frac{2}{\lambda_{\text{max}}}\mathbf{\Lambda} - \mathbf{I}_N \in [-1, 1]$ and Chebyshev coefficients $\Theta \in \mathbb{R}^K$
Incorporating the truncated approximation yields the popular spectral graph convolution method: **ChebyNet**

$$g_\Theta \star \mathbf{x} = (\mathbf{\Phi} g_\Theta)(\mathbf{\Phi}^T \mathbf{x}) \approx \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\mathbf{L}})\mathbf{x} \qquad (6)$$

with $\tilde{\mathbf{L}} = \frac{2}{\lambda_{\text{max}}}\mathbf{L} - \mathbf{I}_N$
Note that the convolution is K-localized and only depends the Laplacian.

Chebyshev polynomial $T_k(y)$ of order k is given by the recurrence:
$T_k(y) = 2yT_{k-1}(y) - T_{k-2}(y)$ with $T_0 = 1$, $T_1 = y$

Defferrard, Michaël, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. Advances in neural information processing systems 29 (2016): 3844-3852.

Further approximation yields the class of neighbourhood aggregating GNNs

With $K = 2, \lambda_{max} = 2$, the graph convolution operation becomes:

$$g_\Theta \star \mathbf{x} \approx \theta_0 \mathbf{x} + \theta_1 \tilde{\mathbf{L}}\mathbf{x} \tag{7}$$

$$= \theta_0 \mathbf{x} + \theta_1 (\mathbf{L} - \mathbf{I}_N)\mathbf{x} \tag{8}$$

$$= \theta(\mathbf{I}_N - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})\mathbf{x} \tag{9}$$

$$g_\Theta \star \mathbf{x} \approx \theta(\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}\mathbf{x}) \tag{10}$$

with $\theta_0 = -\theta_1$.
More generally, for input $\mathbf{X} \in \mathbb{R}^{N \times C}$ and weight matrix $\mathbf{\Theta} \in \mathbb{R}^{C \times F}$:

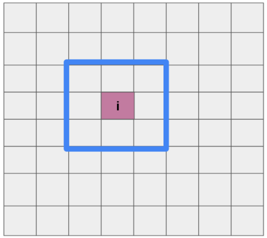
$$\mathbf{H} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}\mathbf{X}\mathbf{\Theta} \tag{11}$$

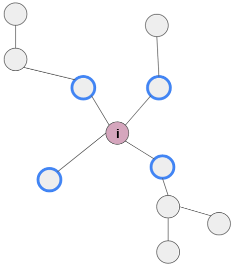
Stacking multiple of these layers with non-linearities yields the class of node GNNs[5]!

[5] Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." arXiv preprint arXiv:1609.02907 (2016).

Note: Chebyshev polynomial $T_k(y)$ of order k is given by the recurrence: $T_k(y) = 2yT_{k-1}(y) - T_{k-2}(y)$ with $T_0 = 1$, $T_1 = y$

GNNs when seen from a node's point of view





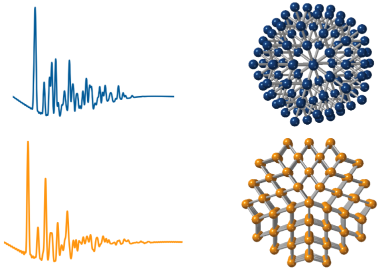
- For a node $i \in \mathcal{V}$ with neighbours $\mathcal{N}_i$ the GNN operation in layer-m is given as:

$$\mathbf{h}_i^{(m)} = \sum_j g_m(\mathbf{h}_{i,j}^{(m-1)}; \Theta_m) j \in \mathcal{N}_i \tag{12}$$

  with $\mathbf{h}_i^{(0)} = \mathbf{x}_i$ and $g_m(\cdot; \Theta_m)$ is an MLP
- Aggregation of transformed neighbouring node features
- M-layered GNN provides information from M-hops away!
- A form of learnable message passing (more on this shortly!)

GNNs for Characterising Atomic Structure of Mono-Metallic Nanoparticles

Consider the task of predicting the atomic structure of nanoparticles based on some desired structure property.

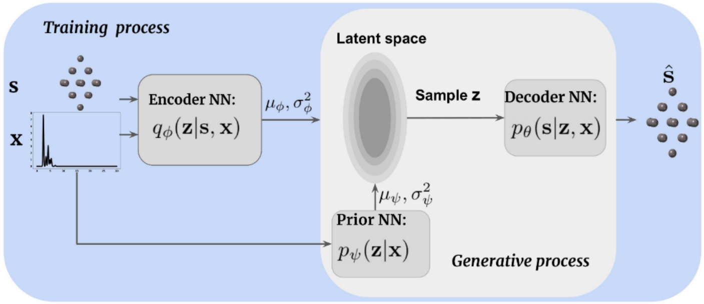


- Treat the atomic structure as a graph with atoms as nodes
- Using property-structure pairs formulate a conditional generative model
- Use GNNs in the encoder:

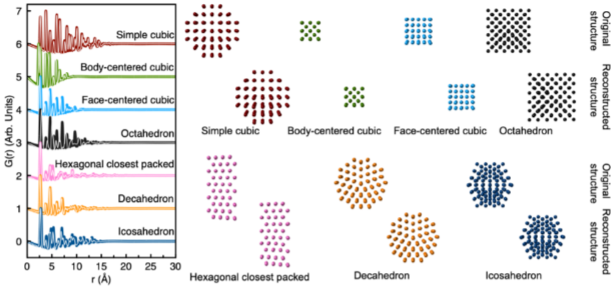
$$\mathbf{H}^{(m)} = \sigma(f^{(m-1)}(\mathbf{H}^{(m-1)}, \mathbf{A}; \Theta_{m-1})) \tag{13}$$

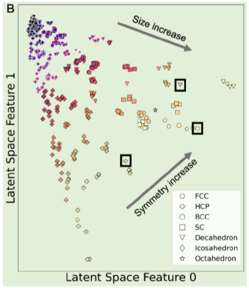
with $\mathbf{H}^{(0)} = \mathbf{X}$ and $\mathbf{H}^{(M)} = \mathbf{Z}$

High level overview of the conditional generative model


Training process
$\mathbf{s}$
$\mathbf{x}$
Encoder NN:
$q_\phi(\mathbf{z}|\mathbf{s},\mathbf{x})$
μ_ϕ, σ_ϕ^2
Latent space
Sample $\mathbf{z}$
Decoder NN:
$p_\theta(\mathbf{s}|\mathbf{z},\mathbf{x})$
$\hat{\mathbf{s}}$
μ_ψ, σ_ψ^2
Prior NN:
$p_\psi(\mathbf{z}|\mathbf{x})$
Generative process

Results on structure prediction of nanoparticles based on properties





Relational GNNs

- Neighbourhood aggregation is good for predictions on nodes or graph level
- Explicit modeling of relations or edges in tasks such as link prediction

$$\begin{aligned}
\text{Node Embedding:} \quad \mathbf{h}_j^{(1)} &= g_n(\mathbf{x}_j) \\
\text{Node-to-Edge mapping:} \quad \mathbf{h}_{(i,j)}^{(1)} &= g_{n2e}([\mathbf{h}_i^{(1)}, \mathbf{h}_j^{(1)}]) \\
\text{Edge-to-Node mapping:} \quad \mathbf{h}_j^{(2)} &= g_{e2n}(\sum_i^{N_j} \mathbf{h}_{(i,j)}^{(1)}]) \\
\text{Node-to-Edge mapping:} \quad \mathbf{h}_{(i,j)}^{(2)} &= g_{n2e}([\mathbf{h}_i^{(2)}, \mathbf{h}_j^{(2)}])
\end{aligned}$$

where $g(\cdot)$ are MLPs.

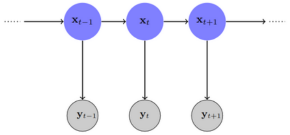
Two types of messages bearing similarities with belief propagation

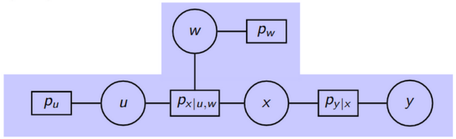
- Node to edges
- Edges to node

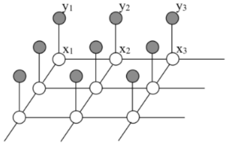
Rich classes of message passing algorithms exist for inference on PGMs

- Kalman filter for Markov chains
- Sum-product algorithm for factor graphs
- Loopy belief propagation for graphs with cycles
- Mean field approximation for Markov random fields

Can GNNs be interpreted as generalizations of these algorithms?





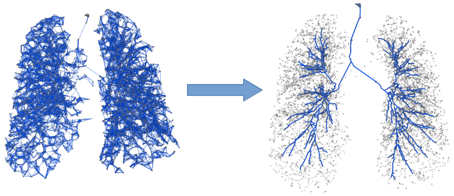


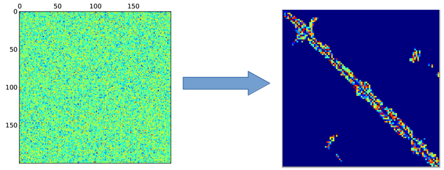
Airway extraction as Graph Refinement task

Graph Refinement Model

$$f(\cdot) : \mathcal{G}_{\text{in}} \mapsto \mathcal{G}$$

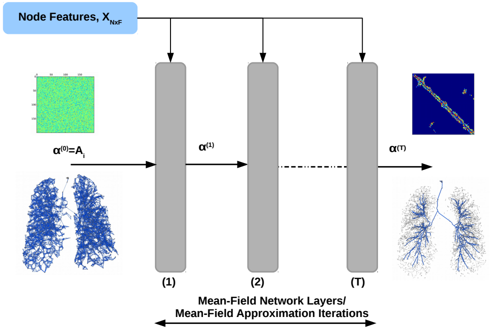
Output subgraph $\mathcal{G}$ with $\mathcal{E} \subset \mathcal{E}_{in}$; $\mathbf{A} \in \{0, 1\}^{N \times N}$



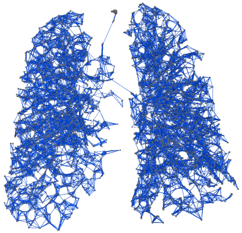


Two approches to Graph Refinement

- Mean-Field Networks (MFNs)
- Graph Neural Networks (GNNs)



Probabilistic Graphical Model for MFN



- Binary random variable $s_{ij} \in \{0,1\}$ with prob. $\alpha_{ij} \in [0,1]$
- For each node: $\mathbf{s}_i = \{s_{ij}\} : j = 1 \dots N$
- Global connectivity variable: $\mathbf{S} = [\mathbf{s}_1 \dots \mathbf{s}_N]$
- Instances of $\mathbf{S}$ are $N \times N$ adjacency matrices

Posterior density of interest: $p(\mathbf{S}|\mathbf{X}, \mathbf{A}_{\text{in}})$

$$\begin{aligned}\ln p(\mathbf{S}|\mathbf{X}, \mathbf{A}_{\text{in}}) &\propto \ln p(\mathbf{S}, \mathbf{X}, \mathbf{A}_{\text{in}}) \\ &= \sum_{i \in \mathcal{N}} \phi_i(\mathbf{s}_i) + \sum_{(i,j) \in \mathcal{E}} \phi_{ij}(\mathbf{s}_i, \mathbf{s}_j) - \ln Z,\end{aligned}$$

Node and Pairwise Potentials for MFA

Node Potential: For each node $i \in \mathcal{V}$

$$\phi_i(\mathbf{s}_i) = \sum_{v=0}^{D} \beta_v \mathbb{I}\left[\sum_j s_{ij} = v\right] + \mathbf{a}^T \mathbf{x}_i \sum_j s_{ij}, \quad (5)$$

Pairwise Potential: For each edge,$(i, j) \in \mathcal{E}_{\text{in}}$

$$\phi_{ij}(\mathbf{s}_i, \mathbf{s}_j) = \lambda(1 - 2|s_{ij} - s_{ji}|) + (2s_{ij}s_{ji} - 1)\left[\boldsymbol{\eta}^T|\mathbf{x}_i - \mathbf{x}_j| + \boldsymbol{\nu}^T(\mathbf{x}_i\mathbf{x}_j)\right]. \quad (6)$$

$$\text{Parameters} = [\beta, \mathbf{a}, \lambda, \boldsymbol{\eta}, \boldsymbol{\nu}]$$

Approximate posterior density with a simpler one

Mean-Field Factorisation: $q(\mathbf{S}) \in \mathcal{Q}$

$$q(\mathbf{S}) = \prod_{i=1}^{N} \prod_{j=1}^{N} q_{ij}(s_{ij}), \tag{14}$$

Implication: Node connectivities are independent.

Variational Inference to approximate $p(\mathbf{S}|\mathbf{X}, \mathbf{A}_{\text{in}})$

$$p(\mathbf{S}|\mathbf{X}, \mathbf{A}_{\text{in}}) \approx q(\mathbf{S}) \tag{15}$$

Minimize KL Divergence $\equiv$ Maximize Evidence Lower Bound (ELBO)

$$\text{ELBO}(q) = -\text{KLD}(q(\mathbf{S}) || p(\mathbf{S}|\mathbf{X}, \mathbf{A}_{\text{in}})\Big] + \ln Z \tag{16}$$

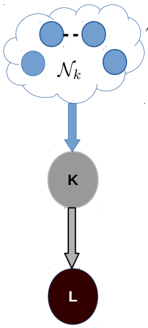
Maximising ELBO wrt $q_{ij}(s_{ij})$ yields MFA Iterations

MFA Iterations

$$\alpha_{kl}^{(t+1)} = q_{kl}^{(t+1)}(s_{kl} == 1) = \frac{1}{1+\exp^{-\gamma_{kl}}}$$

$\forall\ k = \{1 \ldots N\},\ l \in \mathcal{N}_k$

$\boldsymbol{\alpha}$: Global connectivity prediction



$$\gamma_{kl} = \prod_{j \in \mathcal{N}_k \backslash l} (1 - \alpha_{kj}^{(t)}) \Big\{ \sum_{m \in \mathcal{N}_k \backslash l} \frac{\alpha_{km}^{(t)}}{(1-\alpha_{km}^{(t)})} \Big[(\beta_2 - \beta_1) - \beta_2 \sum_{n \in \mathcal{N}_k \backslash l,m} \frac{\alpha_{kn}^{(t)}}{(1-\alpha_{kn}^{(t)})} \Big] + (\beta_1 - \beta_0) \Big\} + \mathbf{a}^T \mathbf{x}_i + (4\alpha_{lk}^{(t)} - 2)\lambda + 2\alpha_{lk}^{(t)} \big(\boldsymbol{\eta}^T |\mathbf{x}_i - \mathbf{x}_j| + \boldsymbol{\nu}^T (\mathbf{x}_i \mathbf{x}_j) \big)$$

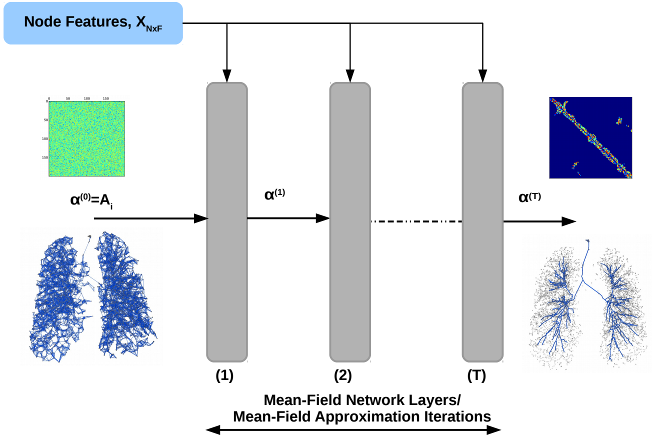
$$\Big\Downarrow$$

$$\alpha_{kl}^{(t+1)} = \sigma(\gamma_{kl}) = \frac{1}{1+\exp^{-\gamma_{kl}}}$$

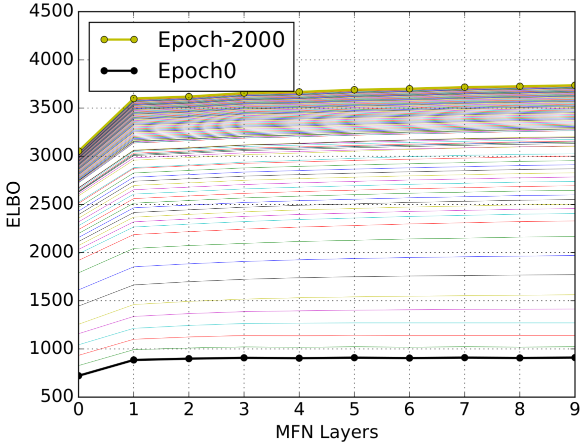
Note: MFA iterations resemble feed-forward operations in neural nets

MFA as Mean-Field Networks

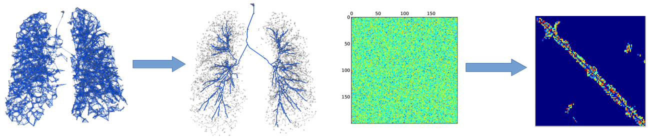
- $T-$iterations as a $T-$layered network
- Gradient descent to learn model parameters: $\mathcal{L}(\boldsymbol{\alpha}, \mathbf{A}_r)$



Increasing ELBO $\implies$ Better approximation



GNN based Graph Refinement



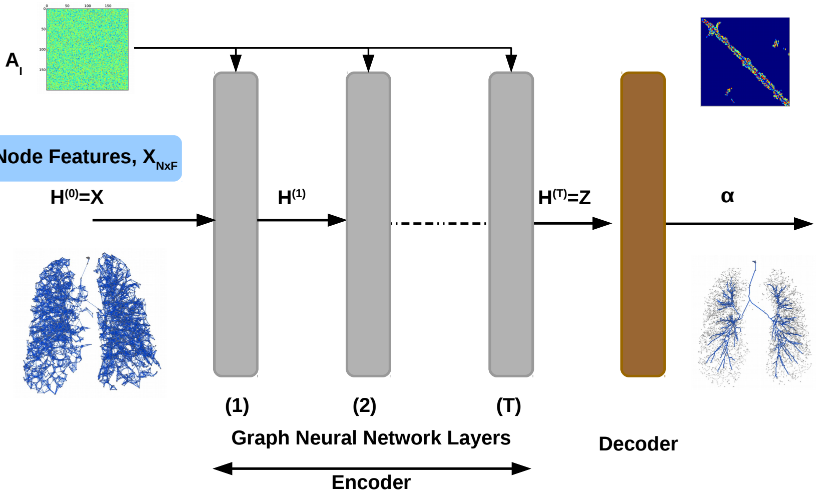
- Graph refinement task: $f(\cdot) : \mathcal{G}_{\text{in}} \mapsto \mathcal{G}$
- GNN based encoder-decoder pair
- Encoder comprises stacks of GNNs
- Learnable Message passing between nodes
- Joint training of encoder-decoder pair to learn useful embeddings
- Simple decoder predicts graph connectivity

GNN Model for Graph Refinement

Consider node j with neighbours $\mathcal{N}_j$,

$$\begin{aligned}
\text{Node Embedding:} \quad \mathbf{h}_j^1 &= g_n(\mathbf{x}_j) && (8)\\
\text{N2E mapping:} \quad \mathbf{h}_{(i,j)}^1 &= g_{n2e}([\mathbf{h}_i^1, \mathbf{h}_j^1]) && (9)\\
\text{E2N mapping:} \quad \mathbf{h}_j^2 &= g_{e2n}(\sum \mathbf{h}_{(i,j)}^1]) \quad \forall i \in \mathcal{N}_j && (10)\\
\text{N2E mapping:} \quad \mathbf{h}_{(i,j)}^2 &= g_{n2e}([\mathbf{h}_i^2, \mathbf{h}_j^2]) && (11)\\
\textbf{Decoder:} \quad \alpha_{ij} &= \sigma(g_{dec}(\mathbf{h}_{(i,j)}^2)) && (12)
\end{aligned}$$

Summarising GNN Model



Experiments

- **Baseline:** a) Region growing on probability images b) Bayesian smoothing merged with region growing for evaluation c) 3D U-net
- Pretraining dataset used to tune hyperparameters
- Eight-fold cross validation
- **Error measures**:
  - Average centerline distance: $d_{err} = (d_{FP} + d_{FN})/2$
  - $d_{FP} \equiv$ Specificity
  - $d_{FN} \equiv$ Sensitivity
  - Percentage of tree length (TL)
  - False positive rate (FPR)

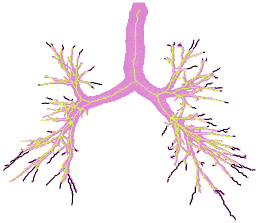
Performance comparison

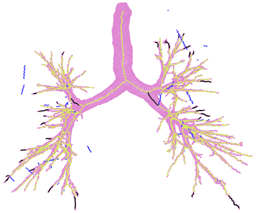
Table 1
Performance comparison of five methods: Region growing on probability images (Vox+RG), Bayesian smoothing merged with Vox+RG (BS+RG), UNet, MFN and GNN models. Dice similarity, centerline distances (d_{FP}, d_{FN}, d_{err}), fraction of tree length detected (TL) and false positive rate (FPR) are reported based on 8−fold cross validation. Significant improvements when compared to other methods are shown in boldface. Additionally, we also report the running time to train each of the models in a single fold. Note that the MFN and GNN models require additional preprocessing that is performed only once when preparing the graphs..

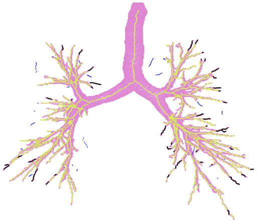
	Dice(%)	d_{FP}(mm)	d_{FN}(mm)	d_{err} (mm)	TL(%)	FPR(%)	Time (m)
Vox+RG	–	2.937 ± 1.005	6.762 ± 2.1042	4.847 ± 2.527	73.2 ± 9.9	4.9 ± 3.9	90
BS+RG	–	2.827 ± 1.266	4.601 ± 2.002	3.714 ± 1.896	73.6 ± 6.1	7.9 ± 6.1	105
UNet	–	3.540 ± 1.316	3.525 ± 1.201	3.532 ± 1.259	75.6 ± 8.7	6.5 ± 3.3	5700
MFN	86.5 ± 2.5	3.608 ± 1.360	3.116 ± 0.632	3.362 ± 1.297	74.5 ± 6.7	8.6 ± 5.4	60 + 35
GNN	84.8 ± 3.3	**2.216 ± 0.464**	**2.878 ± 0.505**	**2.547 ± 0.587**	**81.9 ± 7.3**	7.8 ± 4.6	60 + 12

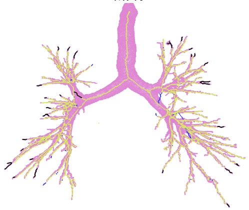
- $d_{FP} \equiv$ Specificity
- $d_{FN} \equiv$ Sensitivity
- Average centerline distance: d_{err}
- Percentage of tree length (TL)
- False positive rate (FPR)

Visualisation of extracted airways


Vox+RG

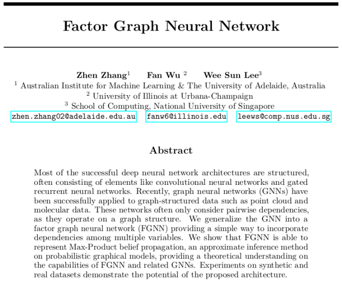

MFN


BS+RG


GNN

Legend: Reference (pink), True Positive (Yellow), False Negative (Black), False Positive (Blue)

Generalization of message passing algorithms

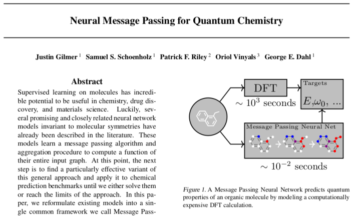


Factor Graph Neural Network

Zhen Zhang[1] **Fan Wu**[2] **Wee Sun Lee**[3]
[1] Australian Institute for Machine Learning & The University of Adelaide, Australia
[2] University of Illinois at Urbana-Champaign
[3] School of Computing, National University of Singapore
zhen.zhang02@adelaide.edu.au fanw6@illinois.edu leews@comp.nus.edu.sg

Abstract

Most of the successful deep neural network architectures are structured, often consisting of elements like convolutional neural networks and gated recurrent neural networks. Recently, graph neural networks (GNNs) have been successfully applied to graph-structured data such as point cloud and molecular data. These networks often only consider pairwise dependencies, as they operate on a graph structure. We generalize the GNN into a factor graph neural network (FGNN) providing a simple way to incorporate dependencies among multiple variables. We show that FGNN is able to represent Max-Product belief propagation, an approximate inference method on probabilistic graphical models, providing a theoretical understanding on the capabilities of FGNN and related GNNs. Experiments on synthetic and real datasets demonstrate the potential of the proposed architecture.



Neural Message Passing for Quantum Chemistry

Justin Gilmer[1] **Samuel S. Schoenholz**[1] **Patrick F. Riley**[2] **Oriol Vinyals**[3] **George E. Dahl**[1]

Abstract

Supervised learning on molecules has incredible potential to be useful in chemistry, drug discovery, and materials science. Luckily, several promising and closely related neural network models invariant to molecular symmetries have already been described in the literature. These models learn a message passing algorithm and aggregation procedure to compute a function of their entire input graph. At this point, the next step is to find a particularly effective variant of this general approach and apply it to chemical prediction benchmarks until we either solve them or reach the limits of the approach. In this paper, we reformulate existing models into a single common framework we call Message Pass-

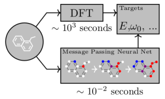
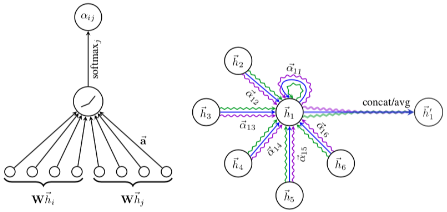


Figure 1. A Message Passing Neural Network predicts quantum properties of an organic molecule by modeling a computationally expensive DFT calculation.

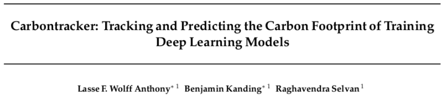
Summary

- Convolutions on graphs can be approximated in spectral domain
- ChebyNet uses a polynomial of spectral filter approximated with Chebyshev polynomials
- First order approximation to ChebyNet yields spatial graph convolutions
- Relational GNNs can be tied to message passing algorithms
- Weighting nodes in a neighbourhood differently yields attention-type models



Thanks!

- Packages for Graph-based deep learning
- DGL[8], Pytorch Geometric[9], JGraph[10]
- raghav@di.ku.dk



Carbontracker: Tracking and Predicting the Carbon Footprint of Training Deep Learning Models

Lasse F. Wolff Anthony* [1] Benjamin Kanding* [1] Raghavendra Selvan [1]

pip install carbontracker

[8] `https://docs.dgl.ai/en/latest/index.html`
[9] `https://pytorch-geometric.readthedocs.io/en/latest/`
[10] `https://github.com/deepmind/jraph`

References

- Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. IEEE Trans. Neural Networks, 20(1):61–80, 2009.
- Defferrard, M., Bresson, X., and Vandergheynst, P. Con- volutional neural networks on graphs with fast localized spectral filtering. In Advances in Neural Information Processing Systems (NIPS), 2016.
- Kipf, T. N. and Welling, M. Semi-supervised classifica- tion with graph convolutional networks. In International Conference on Learning Representations (ICLR), 2017.
- Monti, F., Boscaini, D., and Masci, J. Geometric deep learning on graphs and manifolds using mixture model CNNs. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.
- Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In International Conference on Learning Representations (ICLR), 2018.
- Selvan, R., Kipf, T., Welling, M., Juarez, A. G. U., Pedersen, J. H., Petersen, J., de Bruijne, M. (2020). Graph refinement based airway extraction using mean-field networks and graph neural networks. Medical Image Analysis, 64, 101751.
- Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y., Bresson, X. (2020). Benchmarking graph neural networks. arXiv preprint arXiv:2003.00982.