# *Literature review : Graph Neural Network

- Assume that $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ has $M$ nodes with each node having $N$ dimensions.

## 1. Graph Convolutional Network (GCN).

- First attempt to generalize the Convolutional Neural Network for graph structured data is presented in the Semi-supervised classification with graph convolutional networks with a simple propagation rule:
  - $H^{l+1} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^l W^l\right)$
- Where :
  - $\tilde{A}$ is the adjacency matrix of the undirected graph $\mathcal{G}$ with added self-connection $(\tilde{A} = A + I_N)$.
  - $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$.
  - $W^l$ is a learnable weight matrix.

## 2. Most commonly seen framework - MPNN

- The Message Passing Neural Network (MPNN) framework is first introduced in the Neural Message Passing for Quantum Chemistry paper for a graph classification problem. The authors recognizes the general framework of GNNs in the previous work that learn a message passing algorithm and aggregation procedure. The authors also summarize the general framework as followed:
- Given a Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, The MPNN Framework is defined by :

- The message passing operation : $m_{ij} = \phi_e(h_i^l, h_j^l, e_{ij})$.
- The node operation : $h_i^{l+1} = \phi_h\left(h_i^l, \sum_{j \in N_i} m_{ij}\right)$.
- The read-out function : $\hat{y} = R(\{h_v^T | v \in \mathcal{V}\})$.

- In this paper, the read-out function is defined as a function applied on the final hidden states of all the nodes **Regardless of their correlation to each other.** The read-out function used particularly in the paper was :
  - $R = \sum_{v \in \mathcal{V}} \sigma\left(i(h_v^T, h_v^0)\right) \cdot j(h_v^T)$.
  - $i$ *and* $j$ *are neural networks*.

## 3. E(n) Equivariant Graph Neural Network (EGNN)

- The authors of the E(n) Equivariant Graph Neural Networks introduces an MPNN architecture that is equivariant to rotation, reflection, translation and permutation in n-dimensional space. The proposed method is computationally efficient and not restricted to 3 dimensional space like the previous works.
- E(n) Equivariance to :
  - Translation : $y + g = \phi(x + g)$.
  - Rotation/Reflection : $Qy = \phi(Qx)$.
  - Permutation : $P(y) = \phi(P(x))$.

- The Equivariant Graph Convolutional Layer (EGCL) proposes an **additional coordinates embeddings** aside from the node hidden representation embeddings in plain MPNN framework :
  - Message passing operation : $m_{ij} = \phi_e\left(h_i^l, h_j^l, ||x_i - x_j||^2, e_{ij}\right)$.

- Coordinates update : $x_i^{l+1} = x_i^l \ + \ C\sum_{j\neq i}(x_i - x_j)\phi_x(m_{ij}) \quad (C = 1/(M-1))$.
- Node operation : $h_i^{l+1} = \phi_h\left(h_i^l, \ \sum_{j\neq i} m_{ij}\right)$

- Extension of E(n) EGNN for initialization of initial velocity : If the velocity is not initially 0, the coordinate embeddings update rule is modified as followed :
  - Velocity update : $v_i^{l+1} = \phi_v(h_i^l)v_i^{init} + C\sum_{j\neq i}(x_i - x_j)\phi_x(m_{ij})$.
  - Coordinate update : $x_i^{l+1} = x_i^l + v_i^{l+1}$.
- We can think of the **second term in the velocity update as the momentum** and the **first term as the current velocity**. The velocity of the current state is dependent on the initial velocity and the node representation embeddings.

## 4. Graph Attention Neural network (GAT)

- Introduced in Graph Attention Networks paper, the idea is to compute hidden node representation by attending over the neighborhood following a self-attention mechanism.

- Given a set of input representation of nodes $\{h_1, h_2, ..., h_N\}, \ \ h_i \in \mathbb{R}^F$. The GAT layer produces a set of output representation $\{h_1^{'}, h_2^{'}, ..., h_N^{'}\}, \ \ h_i^{'} \in \mathbb{R}^{F^{'}}$.
  - Unscaled Attention coefficients : $e_{ij} = LeakyReLU(W_a(Wh_i||Wh_j))$.
  - Attention coefficients : $a_{ij} = softmax_j(e_{ij}) = \frac{exp(e_{ij})}{\sum_{k\in N_i} exp(e_{ik})}$.

- $W \in \mathbb{R}^{F \times F'}$ is a common matrix used to project every node from $\mathbb{R}^F$ to $\mathbb{R}^{F'}$.
- $||$ Denotes the concatenation operation.
- $W_a \in \mathbb{R}^{2F'}$ is a weight vector used to calculate the unscaled attention coefficient.

  - $\text{Updated node embeddings}: h_i' = \sigma\left(\sum_{j \in N_i} a_{ij} W h_j\right)$ (Single-headed attention).

- Multi-headed attention mechanism : Given $K$ independent attention mechanism, the features can be aggregated by :
  - $\text{Concatenation}: h_i' = \Big|\Big|_{k=1}^{K} \sigma\left(\sum_{j \in N_i} a_{ij}^k W^k h_j\right)$.
  - $\text{Averaging}: h_i' = \sigma\left(\frac{1}{K} \sum_{k=1}^{K} \sum_{j \in N_i} a_{ij}^k W^k h_j\right)$.

## 5. Graph Neural Ordinary Differential Equations (GDE)

- Introduced in the Graph Neural Ordinary Differential Equations paper. The authors proposed a general framework for node classification using a continuous GNN architecture by formulating the GNN as an ODE. The authors also further generalized the GDE framework for spatio-temporal graph data. However, the scope of the research will not look further into that.
- The general GDE framework is defined as the following Cauchy Problem:

$$\begin{cases} \dot{H}(t) = F_{\mathcal{G}}(t, H(t), \Theta(t)) \quad (t \in \mathcal{T} \subset \mathbb{R}) \\ H_0 = X_e \end{cases}$$

- Symbolically, the output of the GDE is obtained by the following formulation:

$$Y = X_e + \int_{\mathcal{T}} \dot{H}(t, H(t), \Theta(t))dt$$

- Variations of GDEs framework : Different works have proposed different forms of the ODE defined in $\dot{H}(t, H(t), \Theta(t))$. Some of them are :

$$\begin{cases} \text{Graph Neural Diffusion} : \dot{H} = (A - I)(H(t)) \\ \text{Graph Convolution Net} : \dot{H} = \mathcal{C}_{\mathcal{G}}^{\mathcal{N}} \circ \mathcal{C}_{\mathcal{G}}^{\mathcal{N}-1} \circ \ldots \circ \mathcal{C}_{\mathcal{G}}^{1} \mathcal{H}(t) \end{cases}$$

## References


- Satorras, V. G., Hoogeboom, E., & Welling, M. (2021). E(n) Equivariant Graph Neural Networks. https://doi.org/10.48550/arXiv.2102.09844
- Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017). Neural Message Passing for Quantum Chemistry. ArXiv:1704.01212 [Cs]. http://arxiv.org/abs/1704.01212
- Kipf, T. N., & Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. ArXiv:1609.02907 [Cs, Stat]. http://arxiv.org/abs/1609.02907
- Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph Attention Networks. ArXiv:1710.10903 [Cs, Stat]. http://arxiv.org/abs/1710.10903
- Poli, M., Massaroli, S., Park, J., Yamashita, A., Asama, H., & Park, J. (2021). Graph Neural Ordinary Differential Equations. ArXiv:1911.07532 [Cs, Stat]. http://arxiv.org/abs/1911.07532
- Introduction to Graph Neural Network (ML TechTalks) : Link
- Graph Convolutional Networks (GCN) : Link