Graph Neural Networks

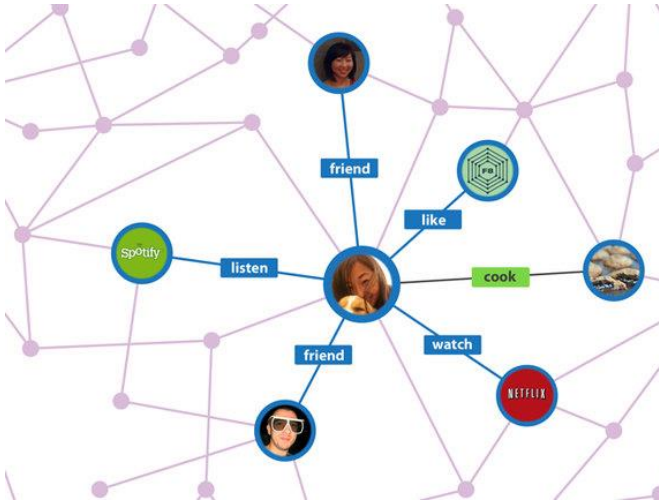
Jian Tang

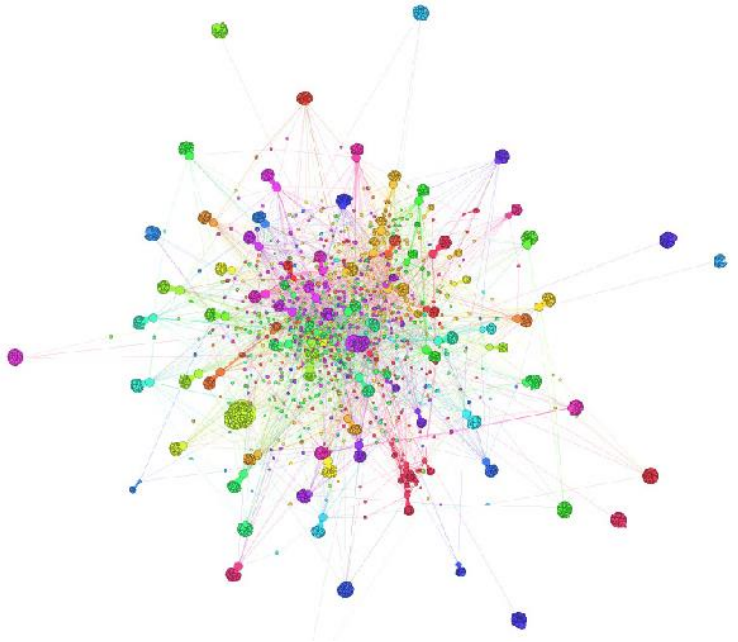
HEC Montreal

Mila-Quebec AI Institute

Email: jian.tang@hec.ca

Social Networks

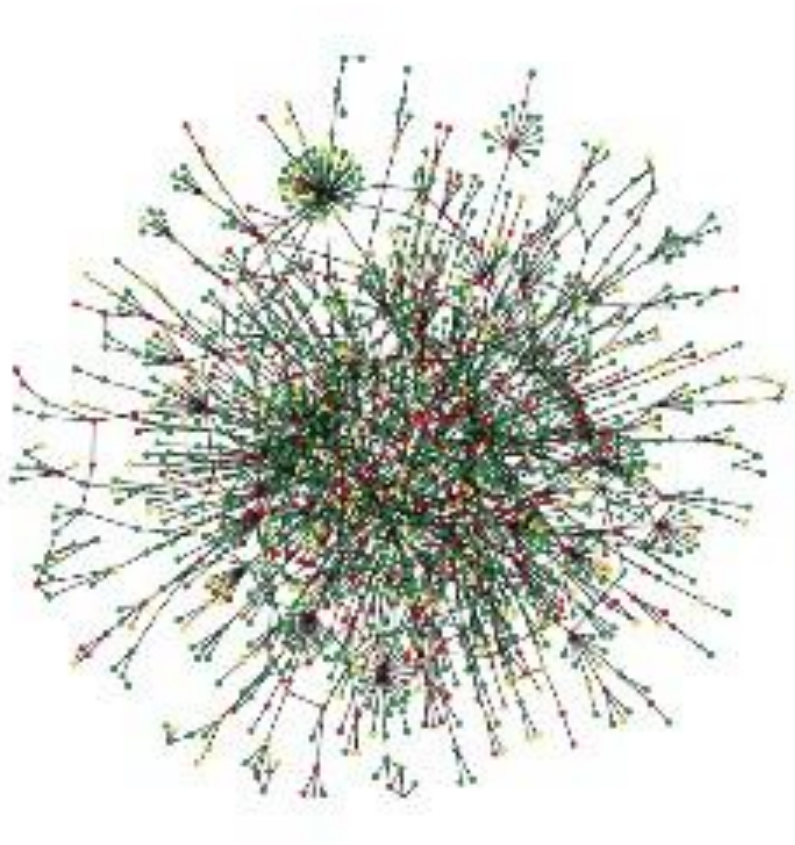


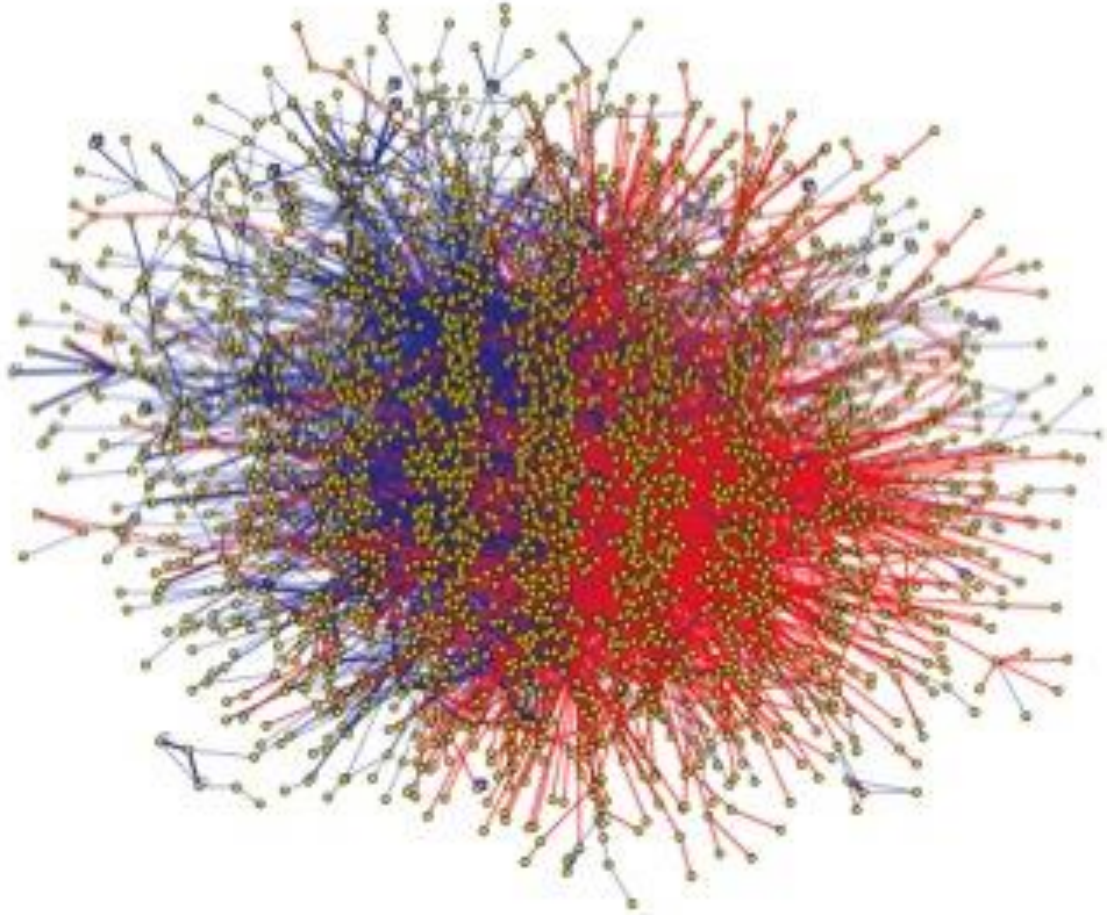


Facebook **Twitter**

Graph from Albert-László Barabási' s SIGIR09 keynote

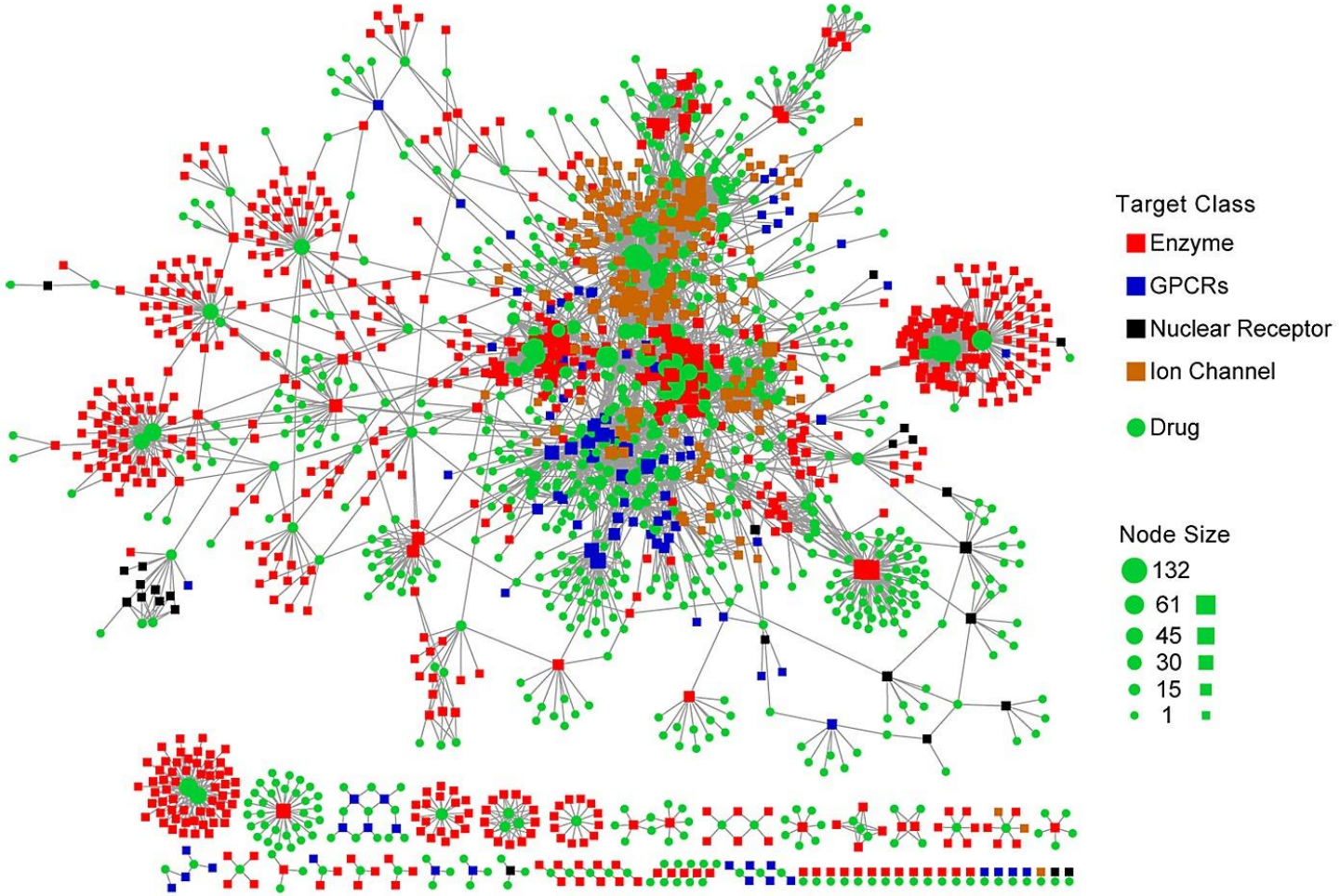
Protein-Protein Interaction Graph





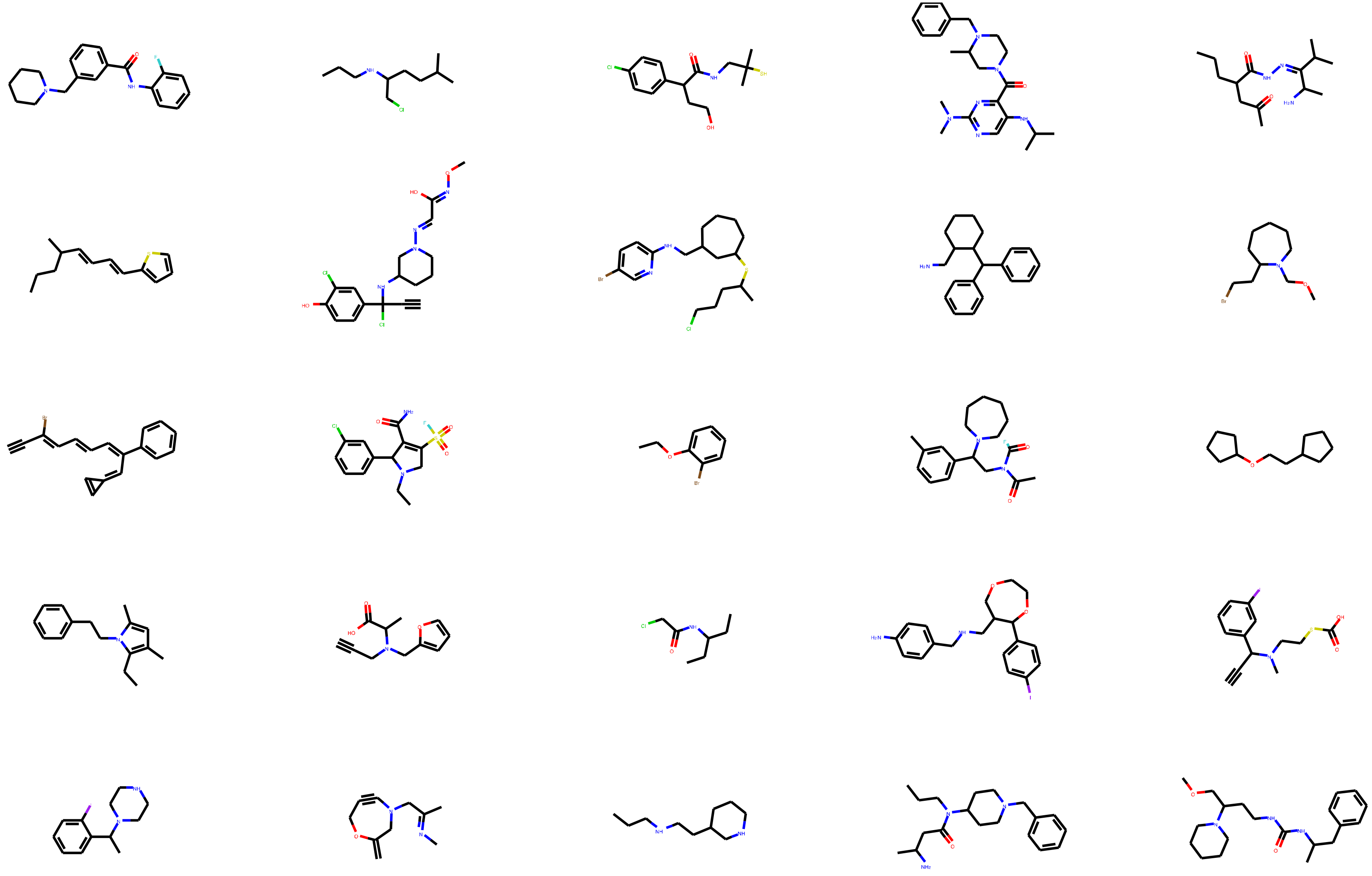
https://www.ebi.ac.uk/training/online/course/network-analysis-protein-interaction-data-introduction/protein-protein-interaction-networks

Drug-Protein Interaction Graph



https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1002503

Molecules

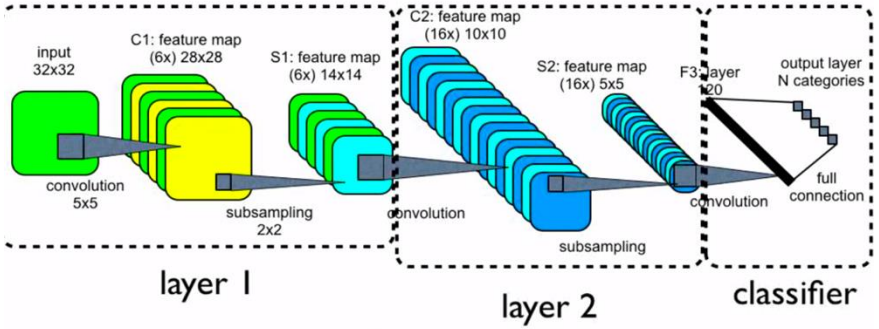


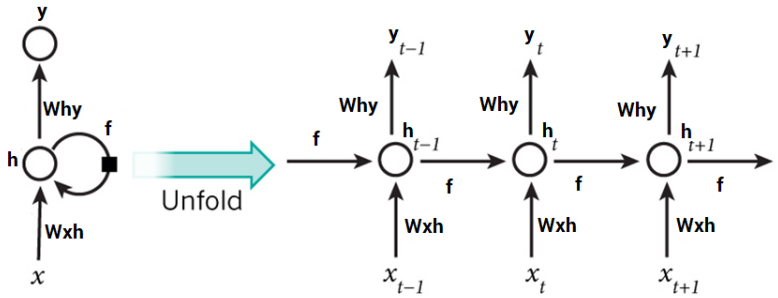
Various Applications on Graphs

- Predicting whether a user is a democratic or republican in Facebook?
- Recommending friends in social networks
- Predicting how information diffuses over social networks
- Predicting the roles of proteins in a protein-protein interaction graphs
- Predicting the chemical properties of molecules
- …
- **Most of these applications require good feature representation of graphs!!**

Challenges of Graph Representation Learning

- Existing deep neural networks are designed for data with regular-structure
  - images, text, and speech





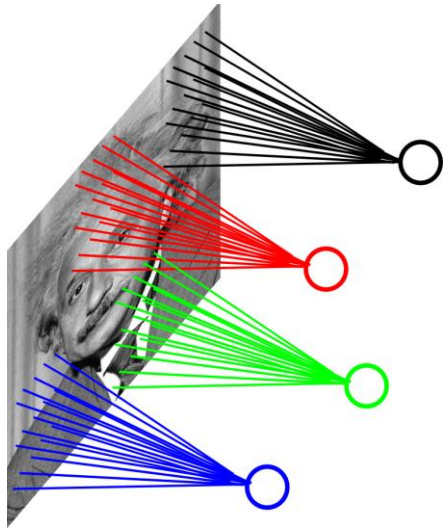
- Graphs are very complex
  - Arbitrary structures
  - Large-scale: more than millions of nodes and billions of edges
  - Heterogeneous: directed/undirected, binary/weighted/typed

Problem Definition

- Given a graph $G = (V, E)$, V is the set of nodes, E is the set of edges. Two types of information are given as input:
  - A feature description $\boldsymbol{x}_i \in R^D$ for each node v_i. The whole node features can be summarized into a $N \times D$ feature matrix **X**.
  - The graph structure, usually in the form of adjacency matrix **A**. A_{ij} is the weight between node i and j.
- Goal: produce node representations, denoted as H (an $N \times F$ feature matrix, F is the dimension of each node representation).

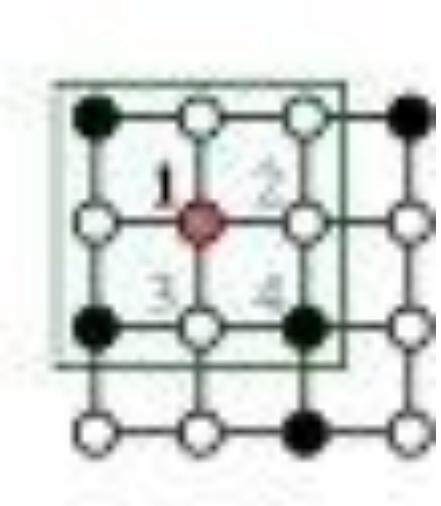
Recap: Convolutional Neural Networks for Learning Image Representations

- Convolutional Filters
  - Local feature detectors
  - A feature is learned in each local receptive field by a convolutional filter

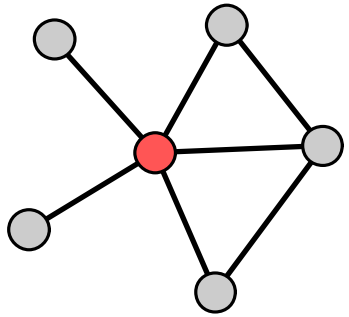


Local Receptive Field on Graphs

- How should we define local receptive fields on graphs?
  - local subgraphs
- However, there are no orders between the neighbors
  - In images, the neighbors of a node can follow specific order



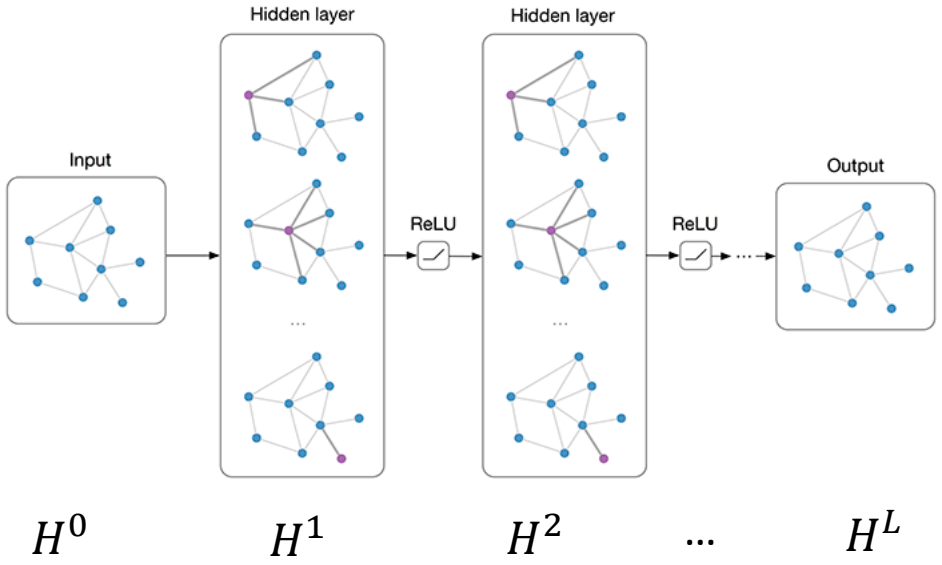
Image



Graph

The Framework of Graph Neural Networks

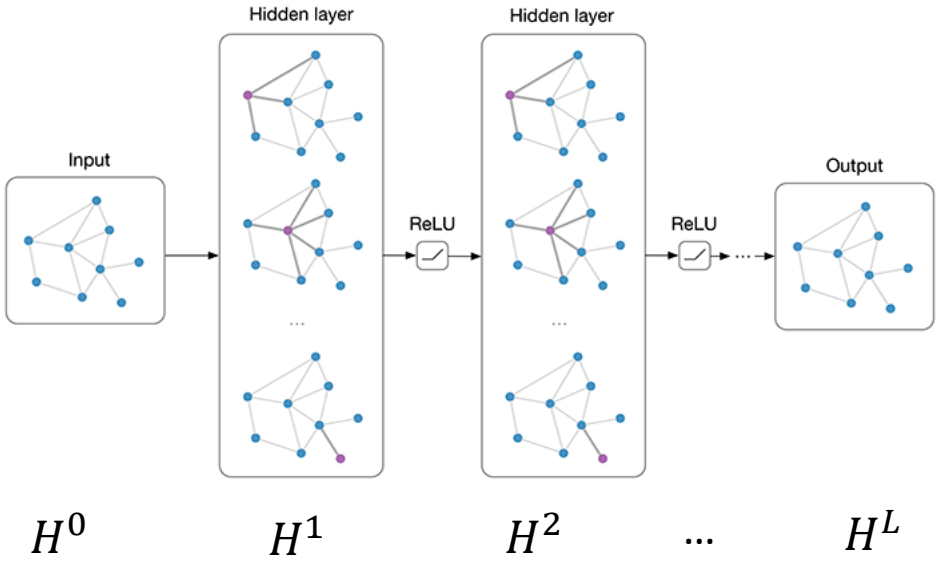
- Multi-layer Graph Neural Networks
  - H^0 = X, the initial node feature matrix
  - Iteratively update the node representations
- The k_{th} layer of graph neural network will update node representations from H^k to H^{k+1}
- The final node representations: H^L
  - Used for some supervised tasks
  - E.g., node classification



Supervised Training

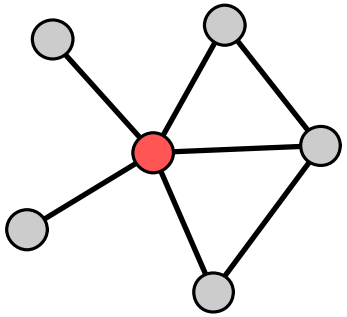
- Train a classifier f based on the final node representations H^L
- The overall objective function:

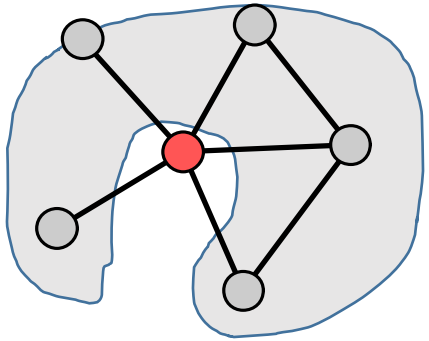
$$O = \sum_{i \in Labeled} loss\left(f\left(H_i^L\right), y_i\right)$$



How to Update the Node Representations?

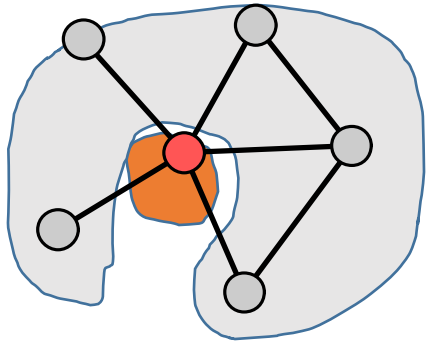
- In each layer of graph neural networks, for each node
  - **AGGREGATE** the information from the neighbors
  - **COMBINE** information from neighbors with its own information





AGGREGATE

$$a_v^k = \mathbf{AGGREGATE^k}(\{h_u^{k-1} : u \in N(v)\})$$



COMBINE

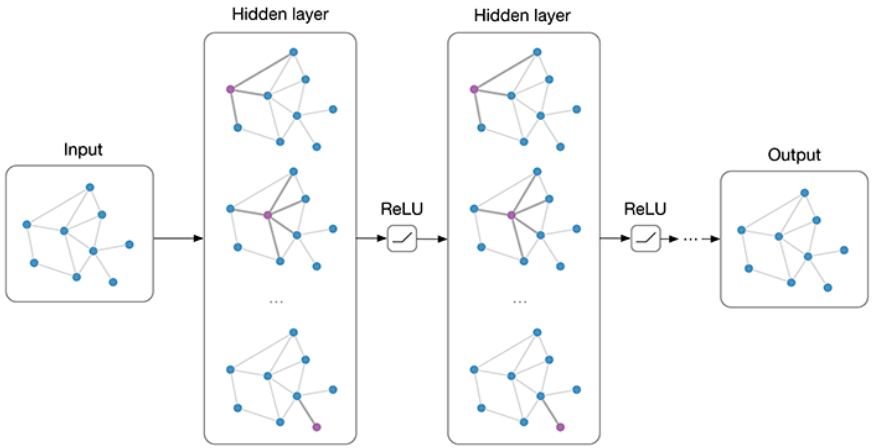
$$h_v^k = \mathbf{COMBINE^k}(h_v^{k-1}, a_v^k)$$

Graph Convolutional Networks

Kipf et al. 2017. Semi-supervised Classification with Graph Convolutional Networks.

Graph Convolutional Networks (GCNs, Kipf et al. 2013)

- A: the adjacency matrix
- Add self link: $\hat{A} = A + I$



$$\mathrm{h}_{\mathrm{i}}^{k} = \sigma\Big(\sum_{j \in \{N(i) \cup i\}} \frac{\hat{a}_{ij}}{\sqrt{d_i d_j}} \mathrm{W}^{\mathrm{k}} \mathrm{h}_{\mathrm{j}}^{k-1} \Big)$$

$$\mathrm{h}_{\mathrm{i}}^{k} = \sigma\Big(\sum_{j \in N(i)} \frac{a_{ij}}{\sqrt{d_i d_j}} \mathrm{W}^{\mathrm{k}} \mathrm{h}_{\mathrm{j}}^{k-1} + \frac{1}{d_i} \mathrm{W}^{\mathrm{k}} \mathrm{h}_{i}^{k-1} \Big)$$

d_i: degree of node i (in matrix $\hat{A}$)
W^{k}: transformation matrix in layer k

The Computation Graph

- Two layers of GCNs

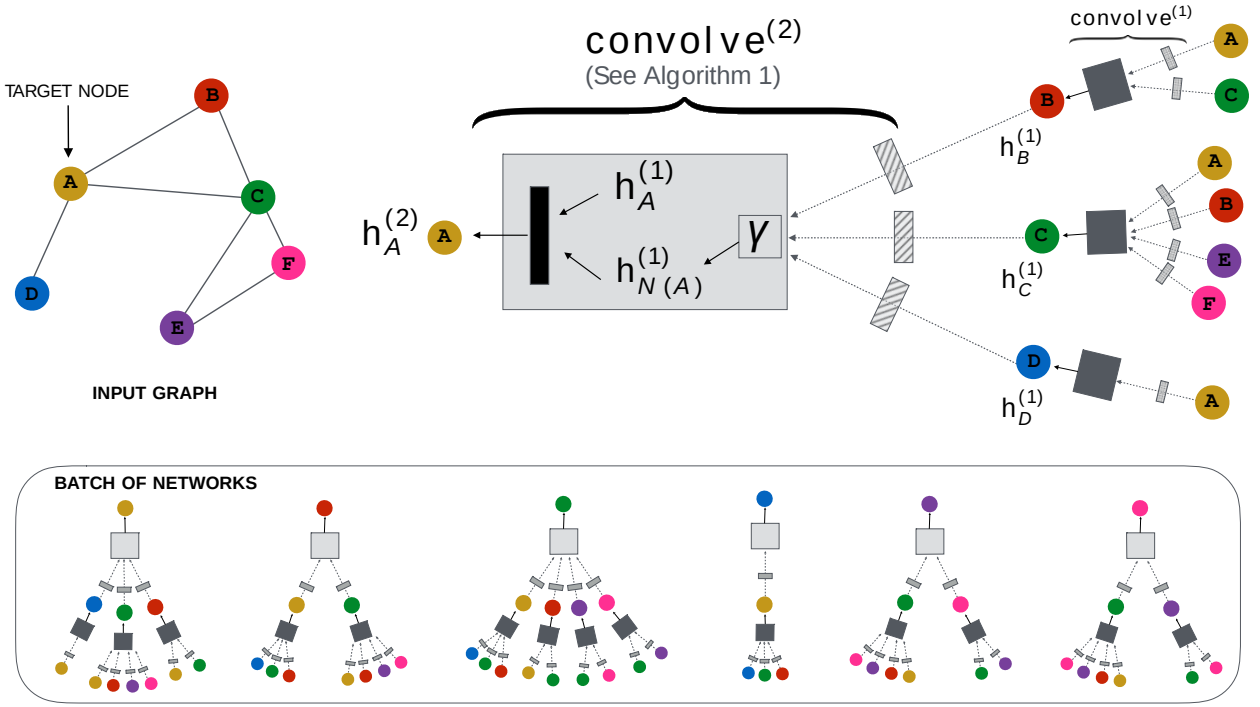


Figure from Ying et al. 2018

Can we change the weights of the edges?

- In GCNs, the influence from node i to node j is determined by the weight of the edge, degree of node i and j, i.e. $\frac{a_{ij}}{\sqrt{d_i d_j}}$
  - i.e., determined by the graph structure
- However,
  - The edges could be very noisy
  - May not be optimal for specific tasks

Graph Attention Networks

Veličković et al. 2017. Graph Attention Networks

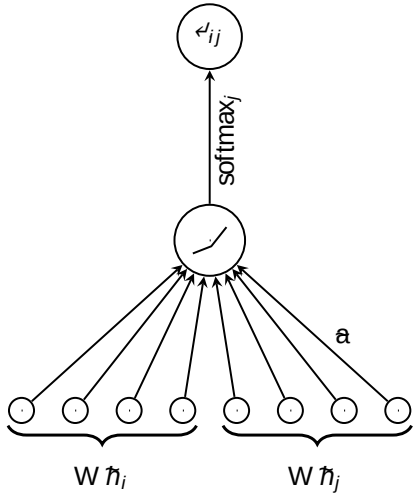
Graph Attention Networks (GATs)

- We can use ATTENTION mechanism to learn the weights between the edges
  - Query: current node
  - Memory: neighbors (including node itself).
- The attention between node *i* and *j*:

$$e_{ij} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$$

$$\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}.$$

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T [\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T [\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$



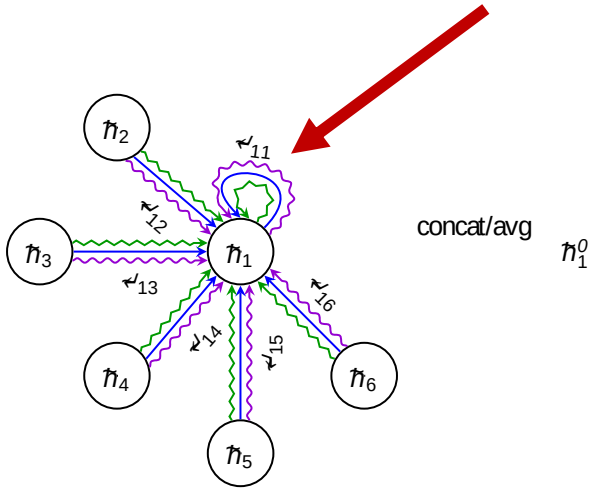
||: vector concatenation

Graph Attention Networks (GATs)

- Aggregate the information from the neighbors with attention

$$\hbar_i^0 = \sigma\Big(\sum_{j \in N_i} \alpha_{ij} W \hbar_j^1\Big)$$

- Note that each node can attend to the node itself

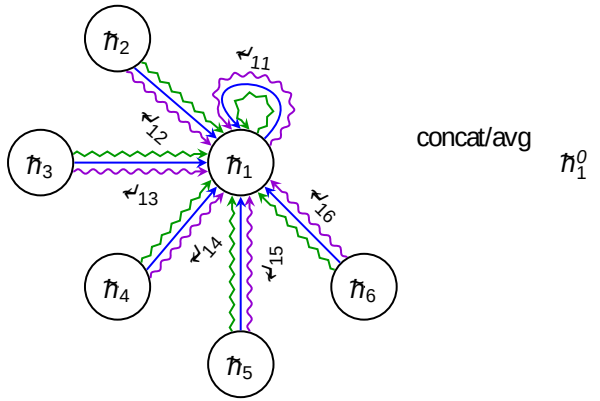


Multi-head Attention

- Following the multi-head attention in the Transformer model, multi-head can be used
- The new node representation can be the concatenation or average of the outputs of different attention heads

$$\vec{h}_i^0 = \Big\Vert_{k=1}^{K} \sigma\Big(\sum_{j \in N_i} \alpha_{ij}^k W^k \vec{h}_j\Big)$$

$$\vec{h}_i^0 = \sigma\Big(\frac{1}{K}\sum_{k=1}^{K}\sum_{j \in N_i} \alpha_{ij}^k W^k \vec{h}_j\Big)$$



A Practical Issue

- Some nodes may have too many neighbors
- Randomly sample a fixed number of neighbors in each iteration of SGD (Hamilton et al. 2017).

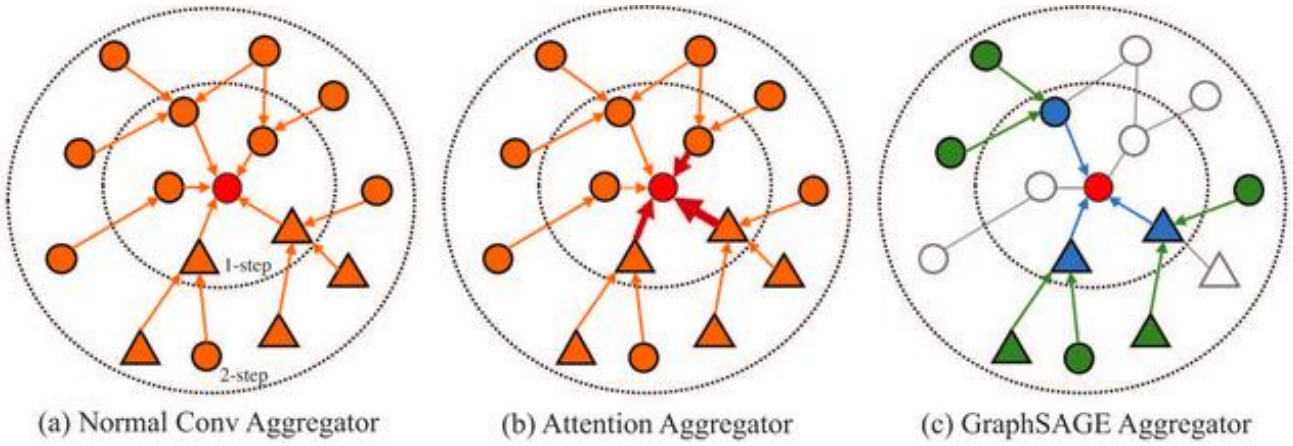


Image from (Wang et al. 2019)

Neural Message Passing Networks

Gilmer et al. 2017. Neural Message Passing for Quantum Chemistry.

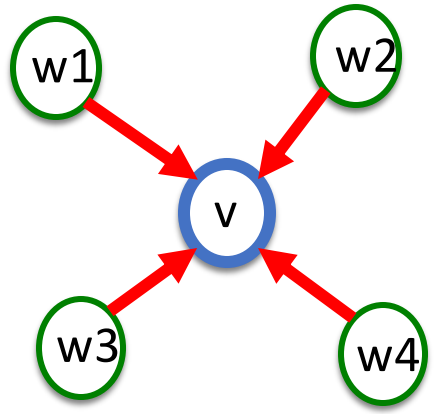
Neural Message Passing Networks (MPNNs, Gilmer et al. 2017)

- All existing graph neural networks can be formulated as the general framework of neural message passing
  - Iteratively pass neural messages (vectors) between nodes
- Different Functions
  - Message Function
  - Node Update Function

Message Passing Phase

AGGREGATE: $$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw})$$

COMBINE: $$h_v^{t+1} = U_t(h_v^t, m_v^{t+1})$$

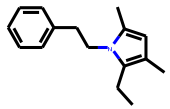
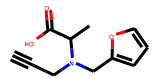
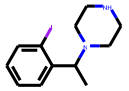
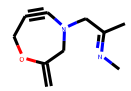


M_t : message function

U_t : vertex update function

What if we want to learn the representations of entire graphs?

- Learn the representations of molecular graphs
  - For predicting the chemical properties of molecules

- Add a readout function, which is applied to the node representations in the last layer:

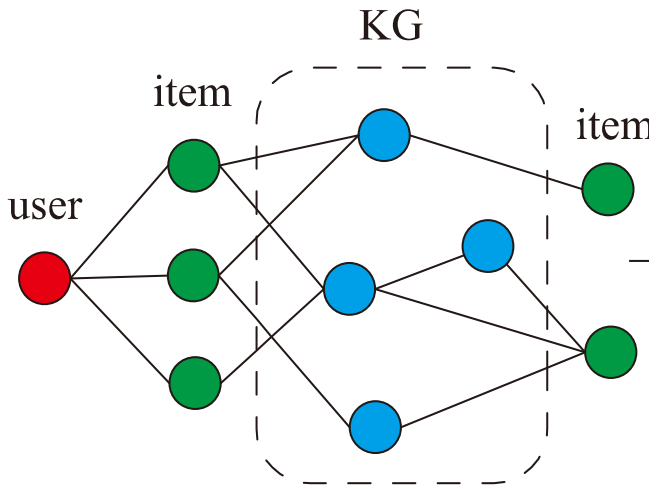
$$\hat{y} = R(\{h_v^T \mid v \in G\})$$

R : readout function

- $\hat{y}$ is the representation of entire graph
- R can be some simple functions such summation or average

Applications: Recommendation

- Predict the most relevant items given users
  - User-item and item-item graphs



Qu et al. An End-to-End Neighborhood-based Interaction Model for Knowledge-enhanced Recommendation.

Applications: Natural Language Understanding

- Semantic Role Labeling
  - Encoding Sentences with Graph Convolutional Networks

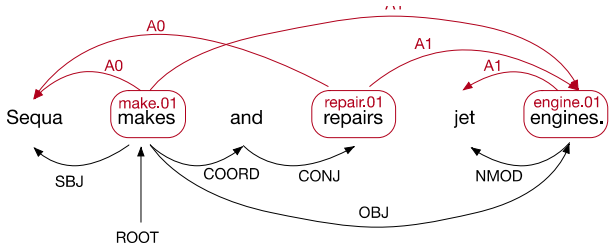
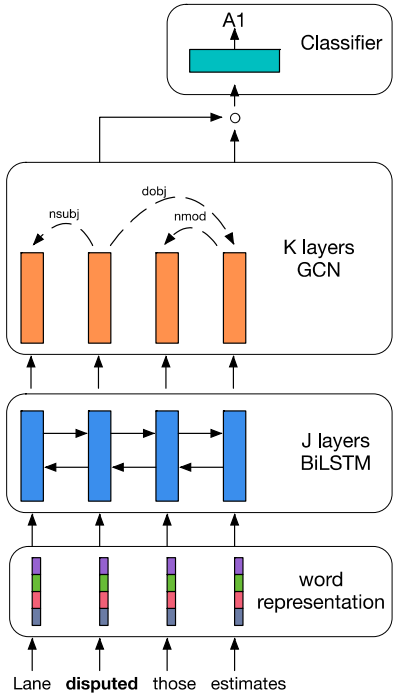


Figure 1: An example sentence annotated with semantic (top) and syntactic dependencies (bottom).



Applications: Drug Discovery

- Drug repurposing
  - Protein-drug-disease graph
- Molecule properties prediction

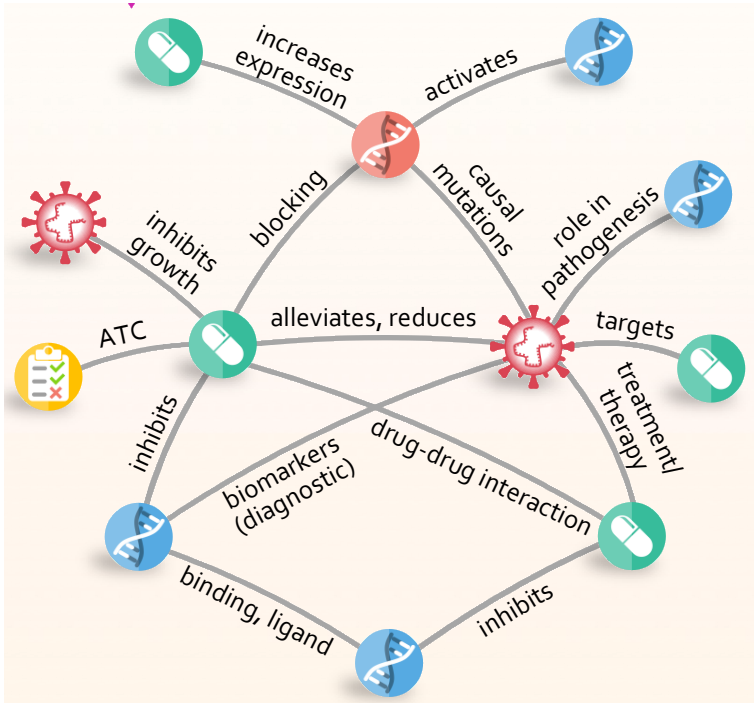
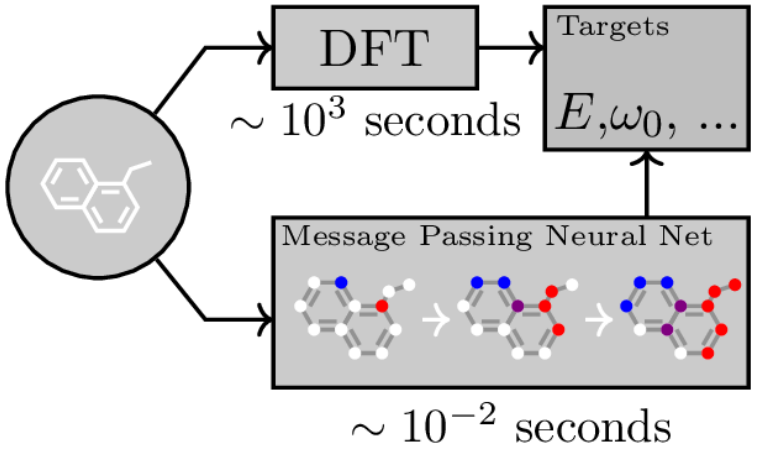


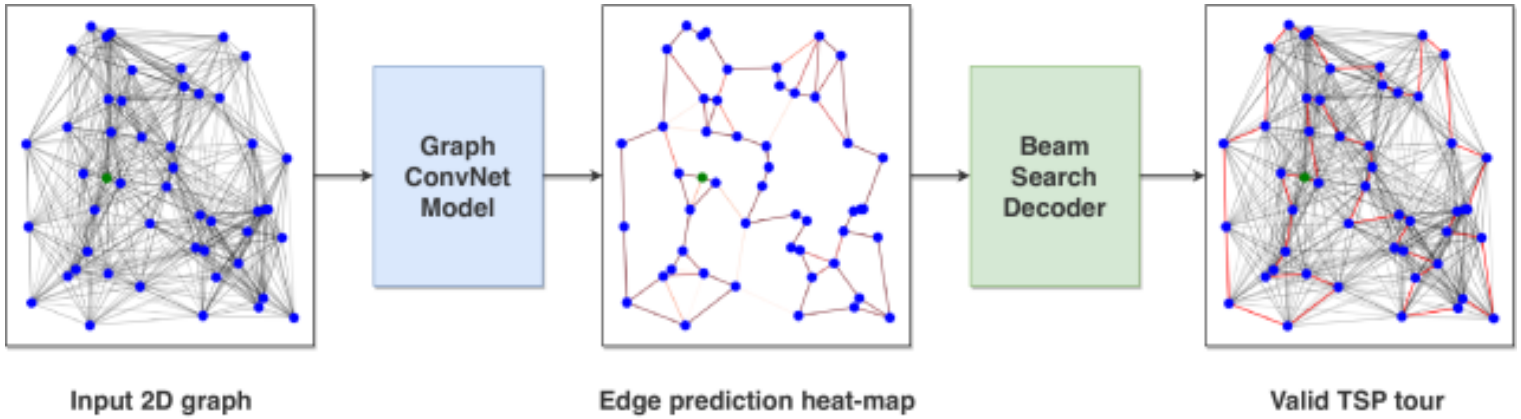
Figure from Zeng et al. 2019



Molecule properties prediction

Applications: Combinatorial Optimization

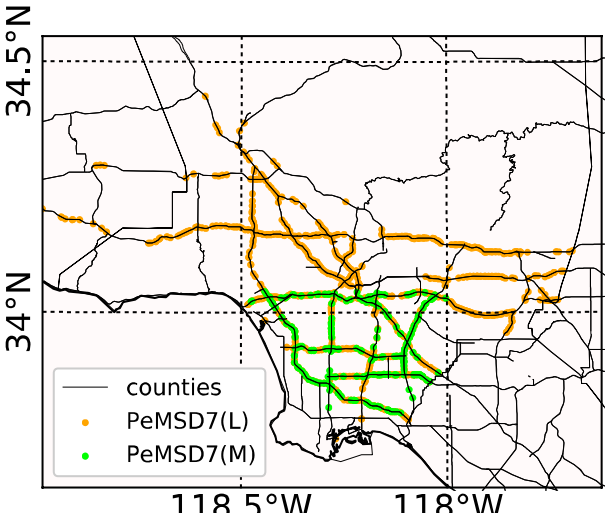
- Travelling Salesman Problem

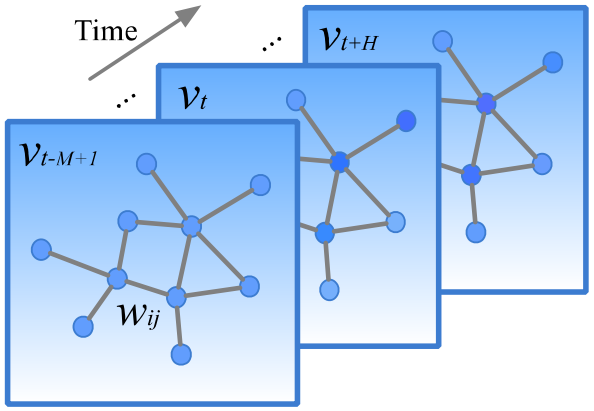


Joshi et al. An Efficient Graph Convolutional Network Technique for the Travelling Salesman Problem.

Applications: Transportation

- Traffic flow prediction
  - Road graph

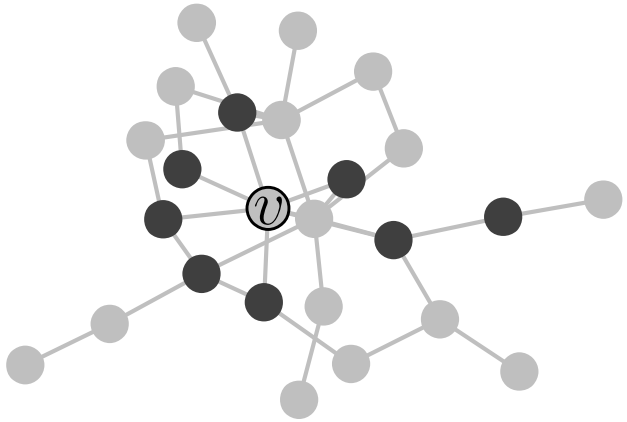




Yu et al. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. IJCAI'18.

Applications: Social Influence Prediction

- Social influence prediction
  - Predict the status of a user given the action statuses of her neighbors in social network



Qiu et al. DeepInf: Social Influence Prediction with Deep Learning.

Implementations of GNNs

- **PyTorch Geometric:** https://pytorchgeometric.readthedocs.io/en/latest/
- **Deep Graph Library:** https://www.dgl.ai/

Example: GCN (Kipf et al.) in Pytorch Geometric

- https://github.com/rusty1s/pytorch_geometric/blob/master/examples/gcn.py

```
28
29  class Net(torch.nn.Module):
30      def __init__(self):
31          super(Net, self).__init__()
32          self.conv1 = GCNConv(dataset.num_features, 16, cached=True,
33                               normalize=not args.use_gdc)
34          self.conv2 = GCNConv(16, dataset.num_classes, cached=True,
35                               normalize=not args.use_gdc)
36          # self.conv1 = ChebConv(data.num_features, 16, K=2)
37          # self.conv2 = ChebConv(16, data.num_features, K=2)
38
39      def forward(self):
40          x, edge_index, edge_weight = data.x, data.edge_index, data.edge_attr
41          x = F.relu(self.conv1(x, edge_index, edge_weight))
42          x = F.dropout(x, training=self.training)
43          x = self.conv2(x, edge_index, edge_weight)
44          return F.log_softmax(x, dim=1)
45
```

Thanks!