Representation Learning on Graphs with Jumping Knowledge Networks

Keyulu Xu, Chengtao Li et al.

Presenter: Shagun Sodhani

Mila

Graph Representation Learning

- A common trend is to encode a node in terms of its neighbour’s representations.

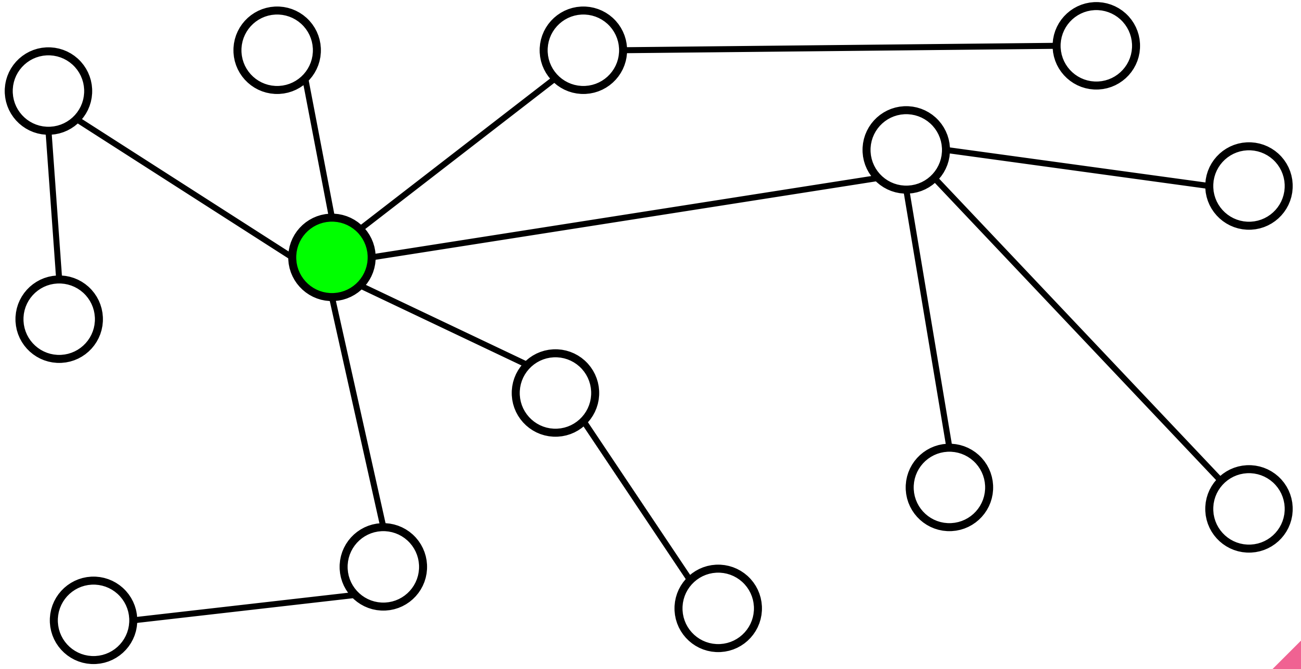
Graph Representation Learning

- A common trend is to encode a node in terms of its neighbour's representations.
- This is followed by a neighbourhood aggregation procedure.

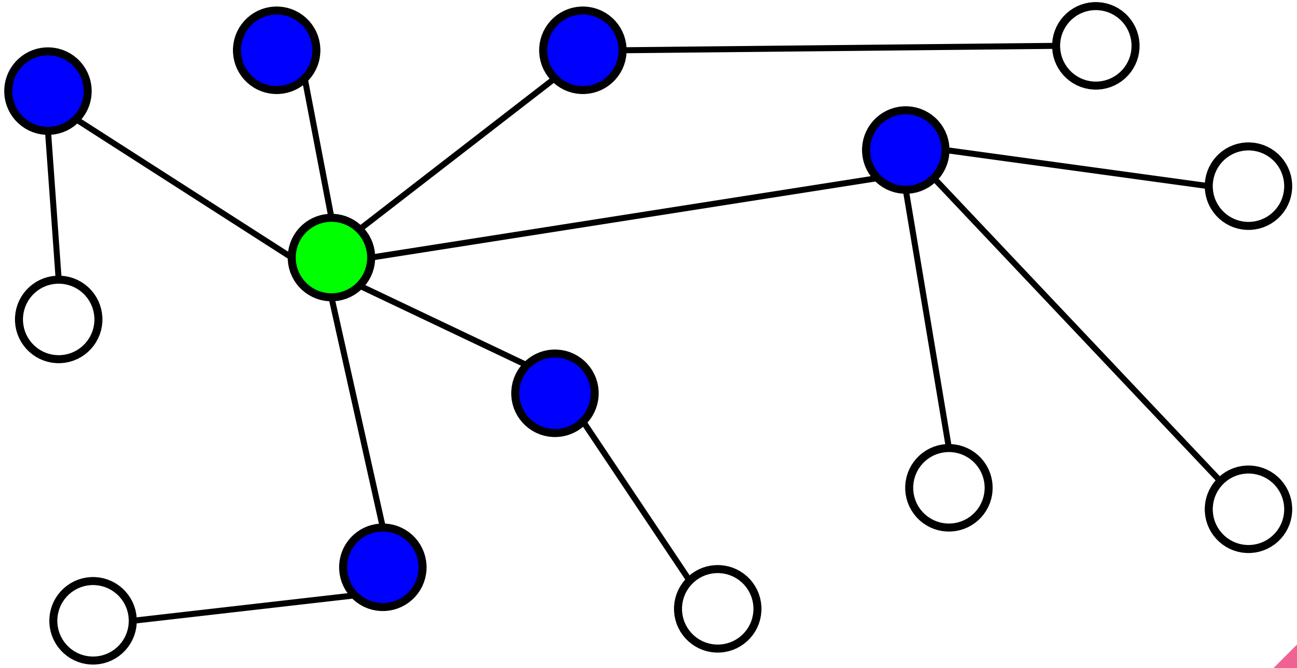
Graph Representation Learning

- A common trend is to encode a node in terms of its neighbour’s representations.
- This is followed by a neighbourhood aggregation procedure.
- Aggregation operation allows for capturing higher-level features in the graph.

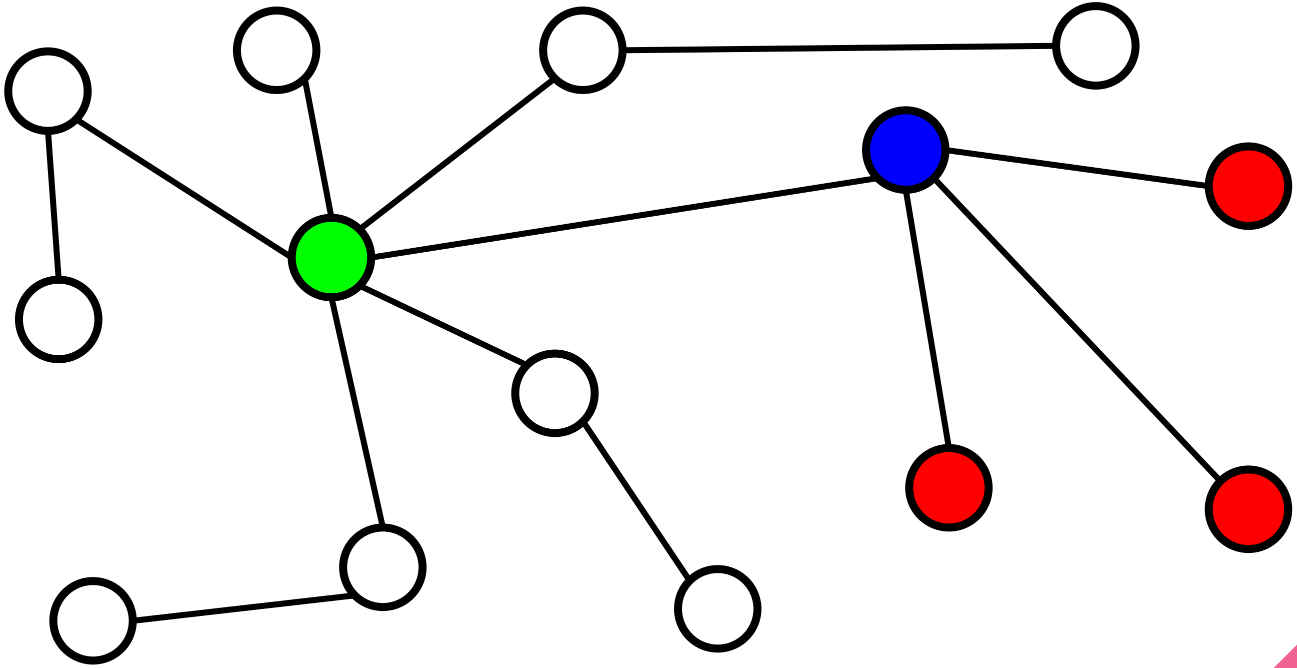
Graph Representation Learning



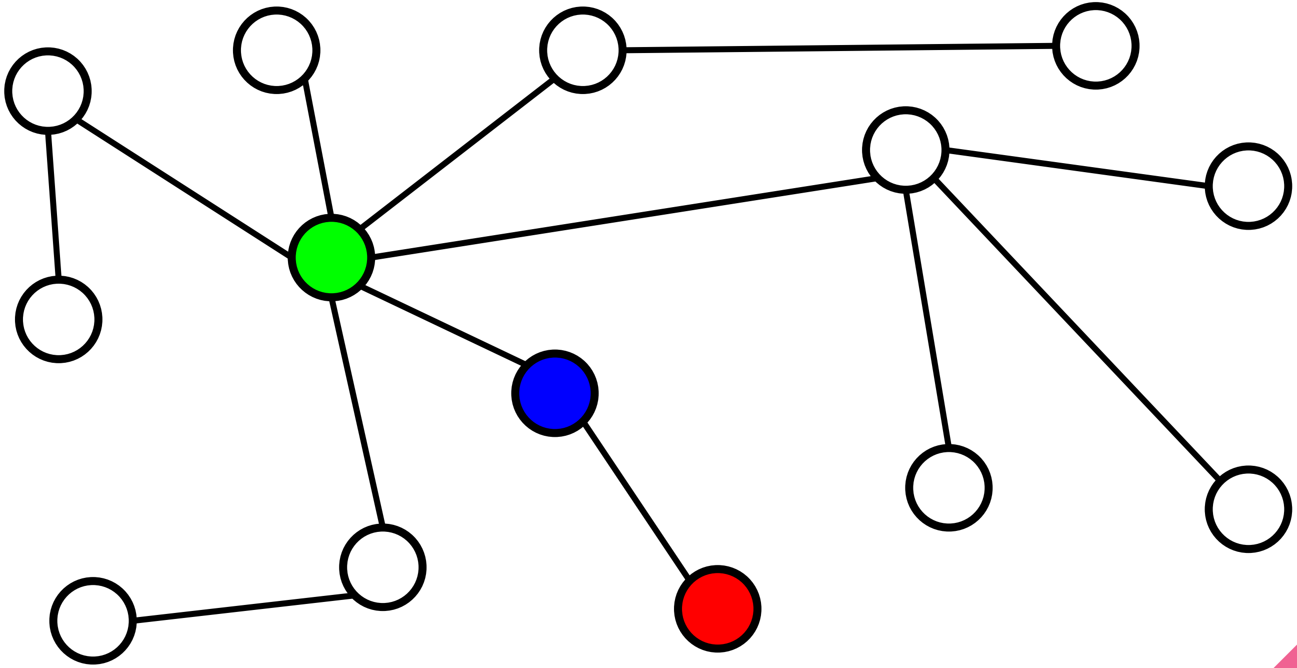
Graph Representation Learning



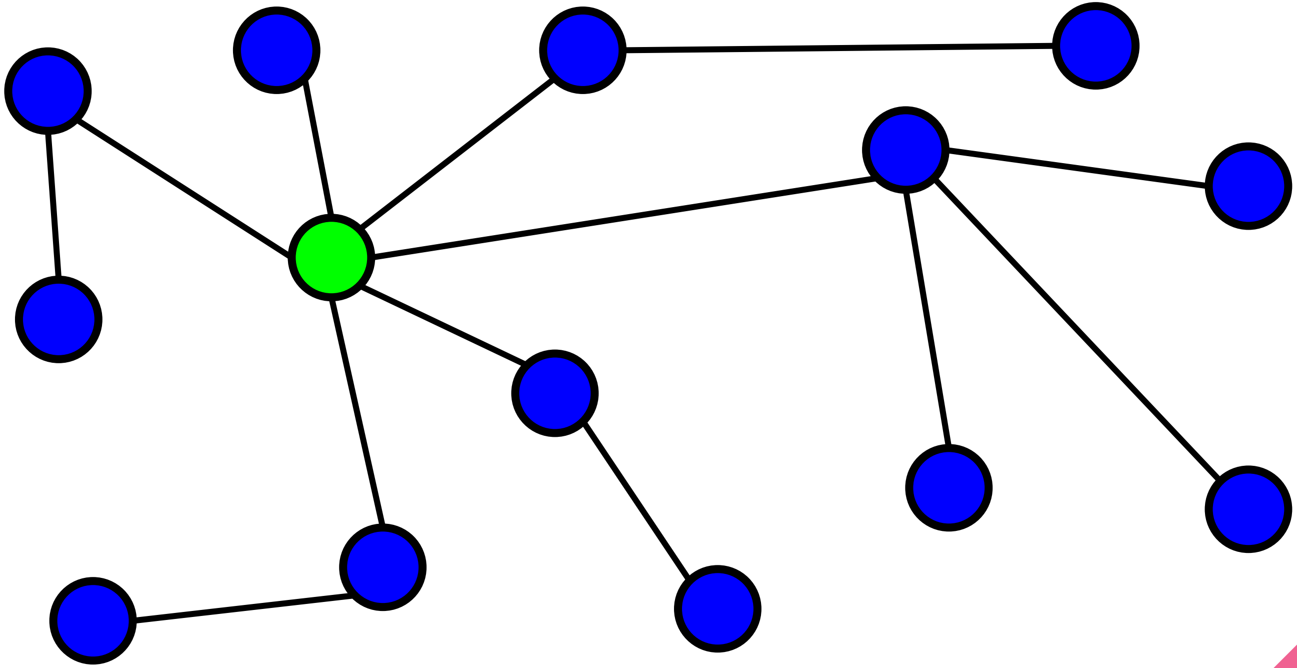
Graph Representation Learning



Graph Representation Learning



Graph Representation Learning



Theoretically...

- An aggregation process of *k* iterations makes use of the subtree structures of height *k* rooted at every node.

Theoretically...

- An aggregation process of *k* iterations makes use of the subtree structures of height *k* rooted at every node.
- Such schemes can simultaneously learn the topology as well as the distribution of node features in the neighborhood.

Practically...

- GCNs with more than 2 layers do not perform as well as the 2-layer GCNs on many datasets.

Practically...

- GCNs with more than 2 layers do not perform as well as the 2-layer GCNs on many datasets.
- Not even with residual connections.

Paper's Contributions

- Studies properties and limitations of neighborhood aggregation schemes.

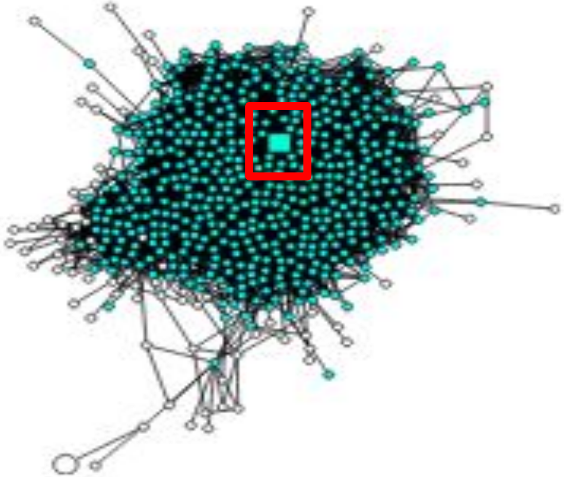
Paper’s Contributions

- Studies properties and limitations of neighborhood aggregation schemes.

- Propose architecture that enables adaptive *structure-aware* representations.

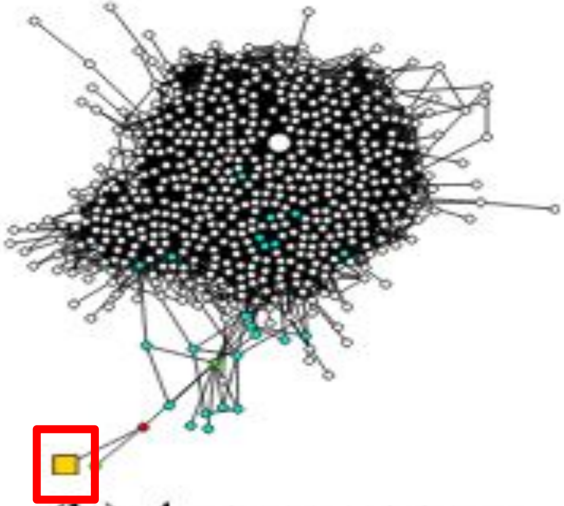
Preliminaries

- *Influence Distribution* of a node - Effective range of nodes that a given node’s distribution depends upon.



(a) 4 steps at core

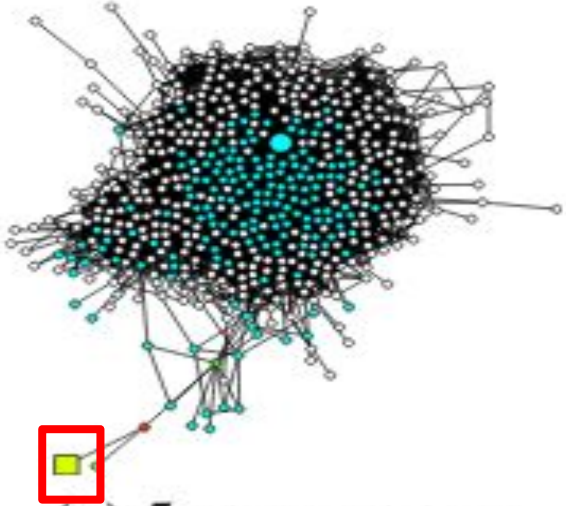
Figure 1. Expansion of a random walk (and hence influence distribution) starting at (square) nodes in subgraphs with different structures. Different subgraph structures result in very different neighborhood sizes.



(b) 4 steps at tree

Figure 1. Expansion of a random walk (and hence influence distribution) starting at (square) nodes in subgraphs with different structures. Different subgraph structures result in very different neighborhood sizes.

(b) 4 steps at tree



(c) 5 steps at tree

Figure 1. Expansion of a random walk (and hence influence distribution) starting at (square) nodes in subgraphs with different structures. Different subgraph structures result in very different neighborhood sizes.

What we know?

- Subgraph structure can drastically affect the result of neighbourhood aggregation.

What we know?

- Subgraph structure can drastically affect the result of neighbourhood aggregation.

- Different nodes would have different influence distribution depending on the topology.

What if

- We run the same number of neighbourhood aggregation steps on each node?

What if

- We run the same number of neighbourhood aggregation steps on each node?

- Can we dynamically adjust the influence radius of each node - depending on the topology and the task?

What if

- We run the same number of neighbourhood aggregation steps on each node?
- Can we dynamically adjust the influence radius of each node - depending on the topology and the task?
- Jumping Knowledge Network allows the node representation to “jump” to last layer.

Preliminaries

- $G = (V, E)$ - Undirected Graph

Preliminaries

- $G = (V, E)$ - Undirected Graph
- $\mathbf{x}_v$ - feature vector associated with node v

Preliminaries

- $G = (V, E)$ - Undirected Graph
- $\mathbf{x}_v$ - feature vector associated with node v
- h^l_v - feature vector learnt by the l^{th} layer for node v

Preliminaries

$$h_v^{(l)} = \sigma\left(W_l \cdot \text{AGGREGATE}\left(\{h_u^{(l-1)}, \forall u \in \widetilde{N}(v)\}\right)\right)$$

Preliminaries

$$h_v^{(l)} = \sigma\left(\boxed{W_l} \cdot \mathrm{AGGREGATE}\left(\{h_u^{(l-1)}, \forall u \in \widetilde{N}(v)\}\right)\right)$$

Weight matrix for the lth layer

Preliminaries

$$h_v^{(l)} = \sigma\left(W_l \cdot \text{AGGREGATE}\left(\{h_u^{(l-1)}, \forall u \in \boxed{\widetilde{N}(v)}\}\right)\right)$$

All neighbours of v (including v)

Influence Distribution

- Sensitivity of node x to node y : How much does a change in input representation of y affect representation of x in the last layer?

Influence Distribution

- Sensitivity of node *x* to node *y* : How much does a change in input representation of *y* affect representation of *x* in the last layer?

- Aka influence of node *y* on node *x*.

Influence Distribution

$$I_x(y) = e^T \left[\frac{\partial h_x^{(k)}}{\partial h_y^{(0)}} \right] e \Big/ \left(\sum_{z \in V} e^T \left[\frac{\partial h_x^{(k)}}{\partial h_z^{(0)}} \right] e \right)$$

Influence Distribution

$$I_x(y) = e^T \left[\frac{\partial h_x^{(k)}}{\partial h_y^{(0)}} \right] \boxed{e} \Bigg/ \left(\sum_{z \in V} e^T \left[\frac{\partial h_x^{(k)}}{\partial h_z^{(0)}} \right] e \right)$$

All-ones vector

Influence Distribution

$$I_x(y) = e^T \boxed{\left[\frac{\partial h_x^{(k)}}{\partial h_y^{(0)}}\right]} e \Big/ \left(\sum_{z \in V} e^T \left[\frac{\partial h_x^{(k)}}{\partial h_z^{(0)}}\right] e \right)$$

Jacobian Matrix

Influence Distribution

$$I_x(y) = \boxed{e^T \left[\frac{\partial h_x^{(k)}}{\partial h_y^{(0)}} \right] e} \Big/ \left(\sum_{z \in V} e^T \left[\frac{\partial h_x^{(k)}}{\partial h_z^{(0)}} \right] e \right)$$

$\boxed{\text{I(x, y)}}$

Influence Distribution

$$I_x(y) = e^T \left[\frac{\partial h_x^{(k)}}{\partial h_y^{(0)}} \right] e \Big/ \boxed{\left(\sum_{z \in V} e^T \left[\frac{\partial h_x^{(k)}}{\partial h_z^{(0)}} \right] e \right)}$$

I(x, z)

Influence Distribution

$$I_x(y) = \boxed{I(x,y)} / \sum_z I(x,z)$$

$$\boxed{e^T \left[\frac{\partial h_x^{(k)}}{\partial h_y^{(0)}} \right] e}$$

Wait. Why are we doing this?

- Influence distribution of common aggregation techniques are closely related to random walk distributions.

Again. Why are we doing this?

- Random walk distributions have useful properties.

Again. Why are we doing this?

- Random walk distributions have useful properties.

- It becomes more spread out as time increases.

Again. Why are we doing this?

- Random walk distributions have useful properties.

- It becomes more spread out as time increases.

- Converges to a limit distribution if graph is non-bipartite.

Again. Why are we doing this?

- Random walk distributions have useful properties.
- It becomes more spread out as time increases.
- Converges to a limit distribution if graph is non-bipartite.
- Rate of convergence depends on the structure of the subgraph.

Connection with Random Walk

- Under simplifying assumptions, we could show that influence distribution of a node in a k-layer GCN model is equivalent to k-step random walk distribution.

Connection with Random Walk

- Under simplifying assumptions, we could show that influence distribution of a node in a k-layer GCN model is equivalent to k-step random walk distribution.

- Can be verified empirically.

Connection with Random Walk

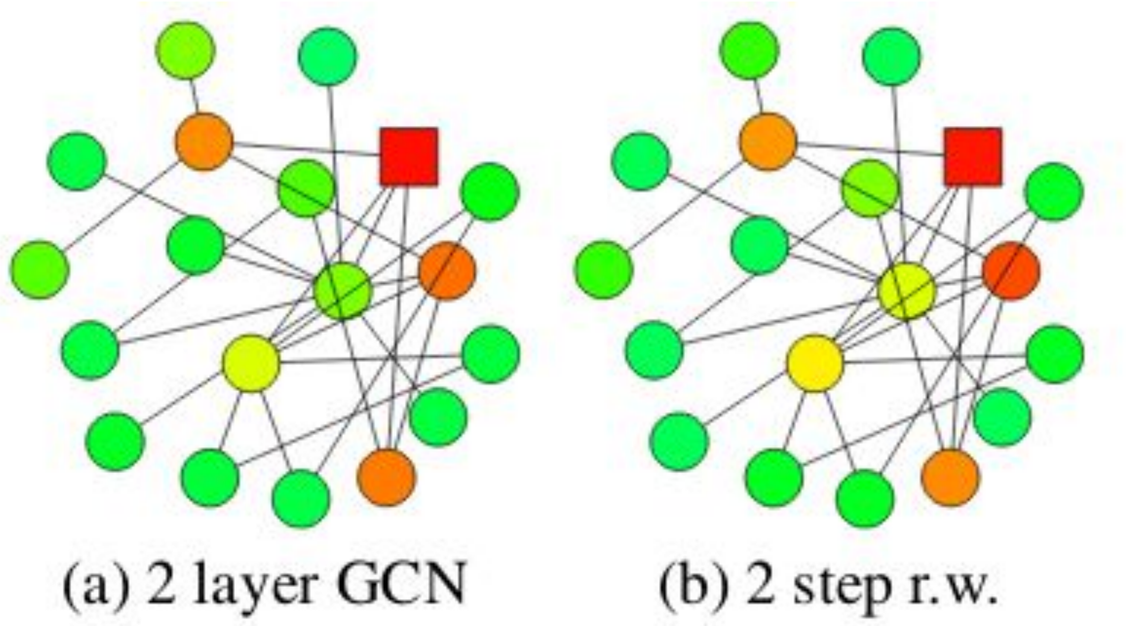


Figure 2. Influence distributions of GCNs and random walk distributions starting at the square node

Connection with Random Walk

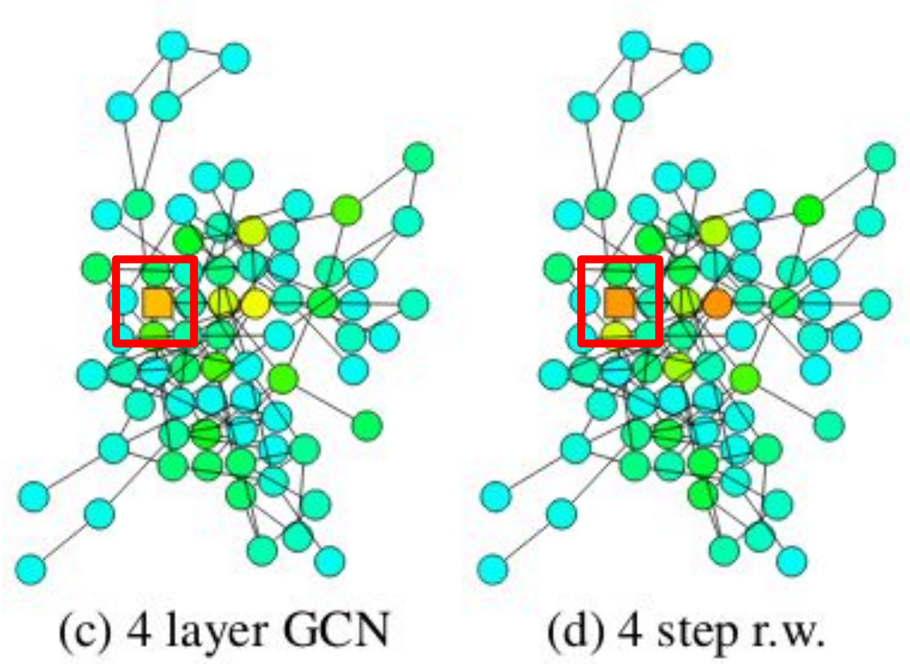
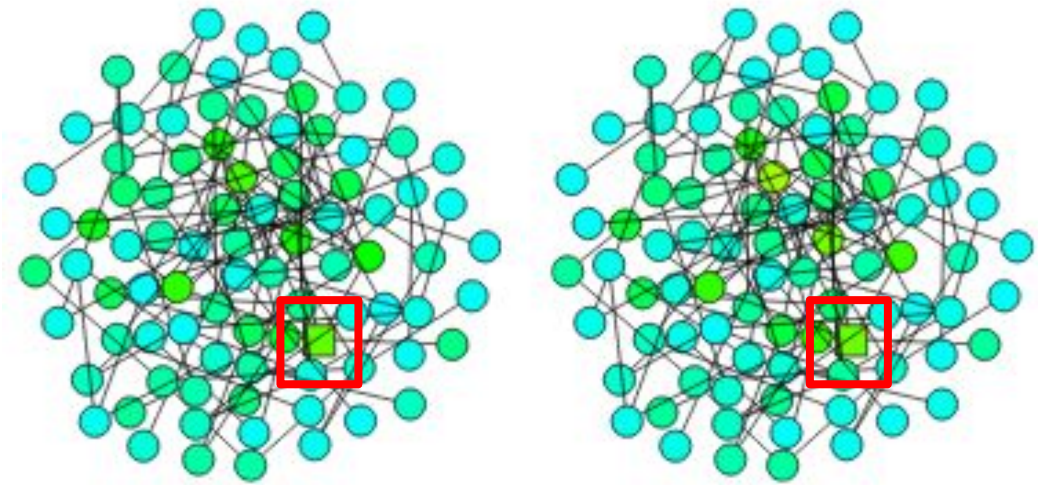


Figure 2. Influence distributions of GCNs and random walk distributions starting at the square node

Connection with Random Walk



(e) 6 layer GCN

(f) 6 step r.w.

Figure 2. Influence distributions of GCNs and random walk distributions starting at the square node

So far

- Common aggregation strategies induces fixed radius size.

So far

- Common aggregation strategies induces fixed radius size.

- Too large radius would lead to too much averaging and the node would capture only global information.

So far

- Common aggregation strategies induces fixed radius size.
- Too large radius would lead to too much averaging and the node would capture only global information.
- Too small radius would lead to insufficient information aggregation.

Jumping Knowledge Networks

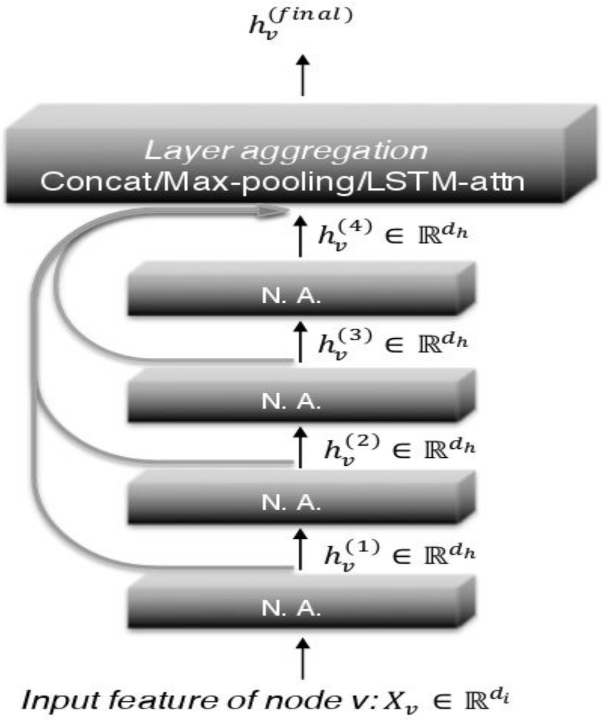


Figure 4. A 4-layer Jumping Knowledge Network (JK-Net). N.A. stands for neighborhood aggregation.

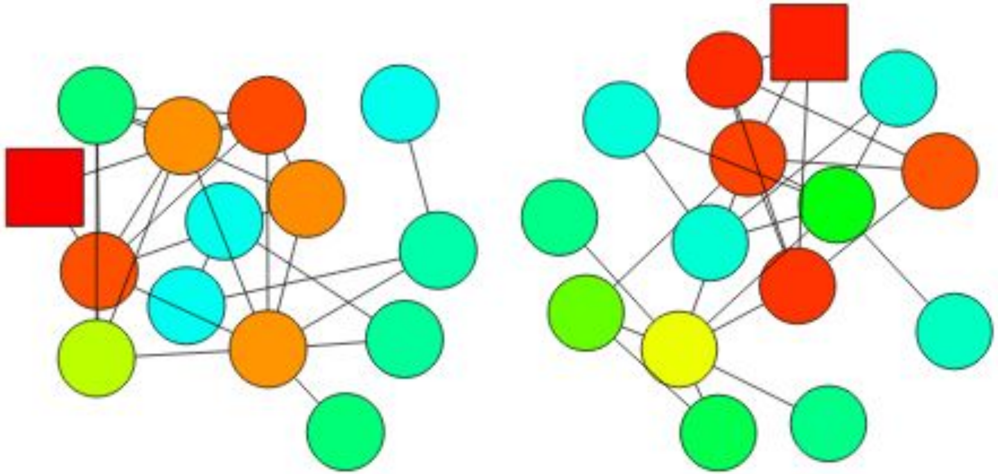
Jumping Knowledge Networks

Proposition 1. *Assume that paths of the same length in the computation graph are activated with the same probability. The influence score $I(x, y)$ for any $x, y \in V$ under a k-layer JK-Net with layer-wise max-pooling is equivalent in expectation to a mixture of $0, .., k$-step random walk distributions on $\widetilde{G}$ at y starting at x, the coefficients of which depend on the values of the layer features $h_x^{(l)}$.*

Jumping Knowledge Networks

Proposition 1. *Assume that paths of the same length in the computation graph are activated with the same probability. The influence score $I(x, y)$ for any $x, y \in V$ under a k-layer JK-Net with layer-wise max-pooling is equivalent in expectation to a mixture of $0, .., k$-step random walk distributions on $\tilde{G}$ at y starting at x, the coefficients of which depend on the values of the layer features $h_x^{(l)}$.*

Jumping Knowledge Networks



(a) tree-like (b) tree-like

Figure 5. A 6-layer JK-Net learns to adapt to different subgraph structures

Jumping Knowledge Networks

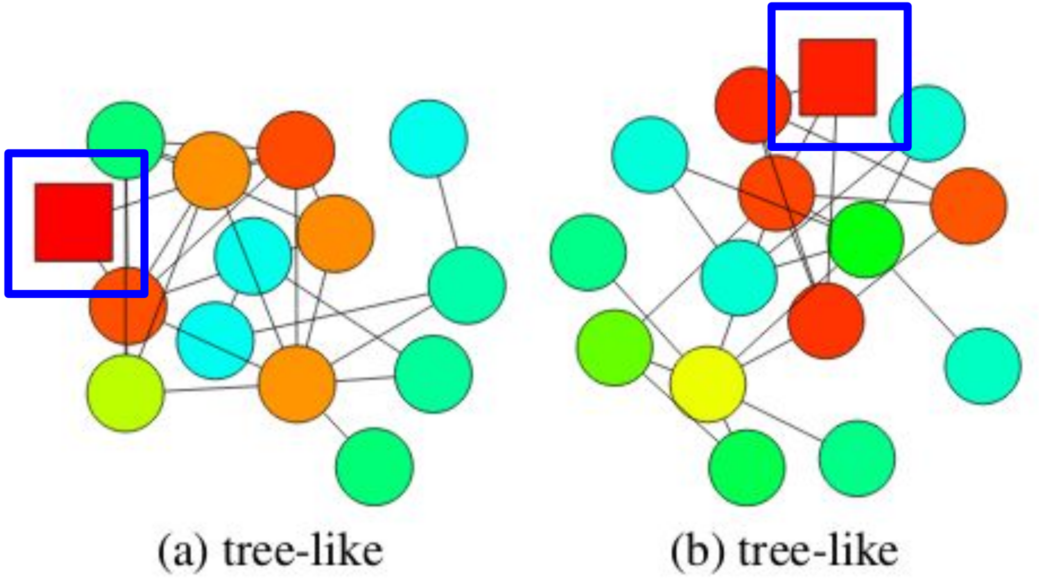
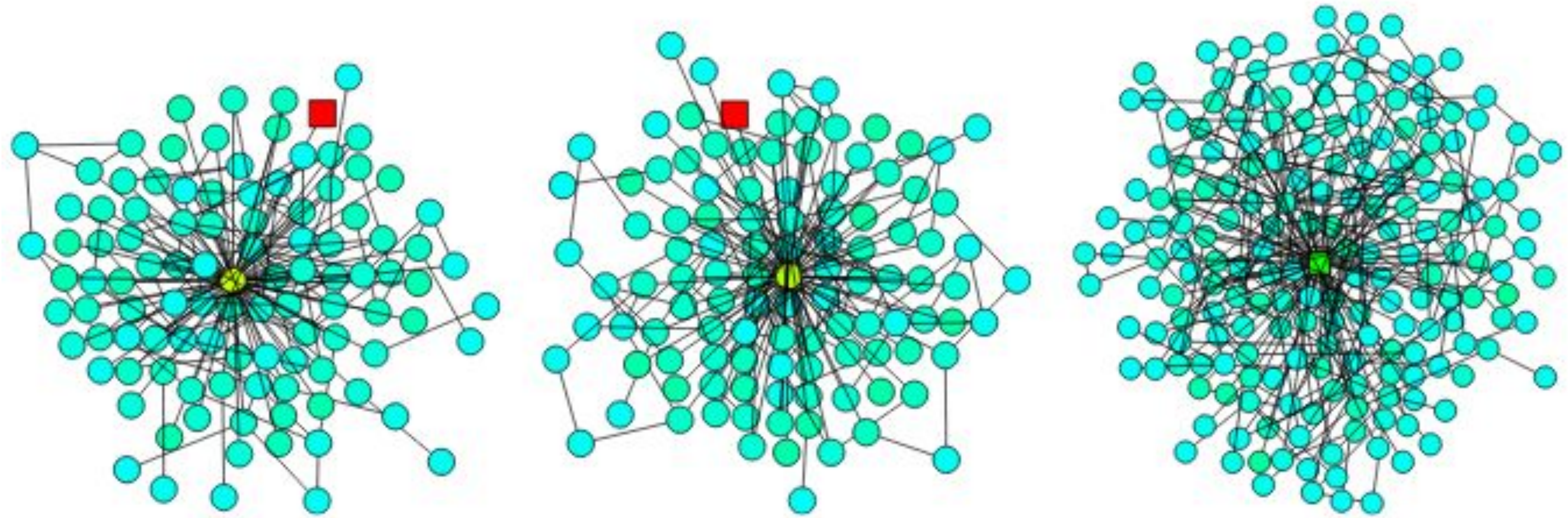


Figure 5. A 6-layer JK-Net learns to adapt to different subgraph structures

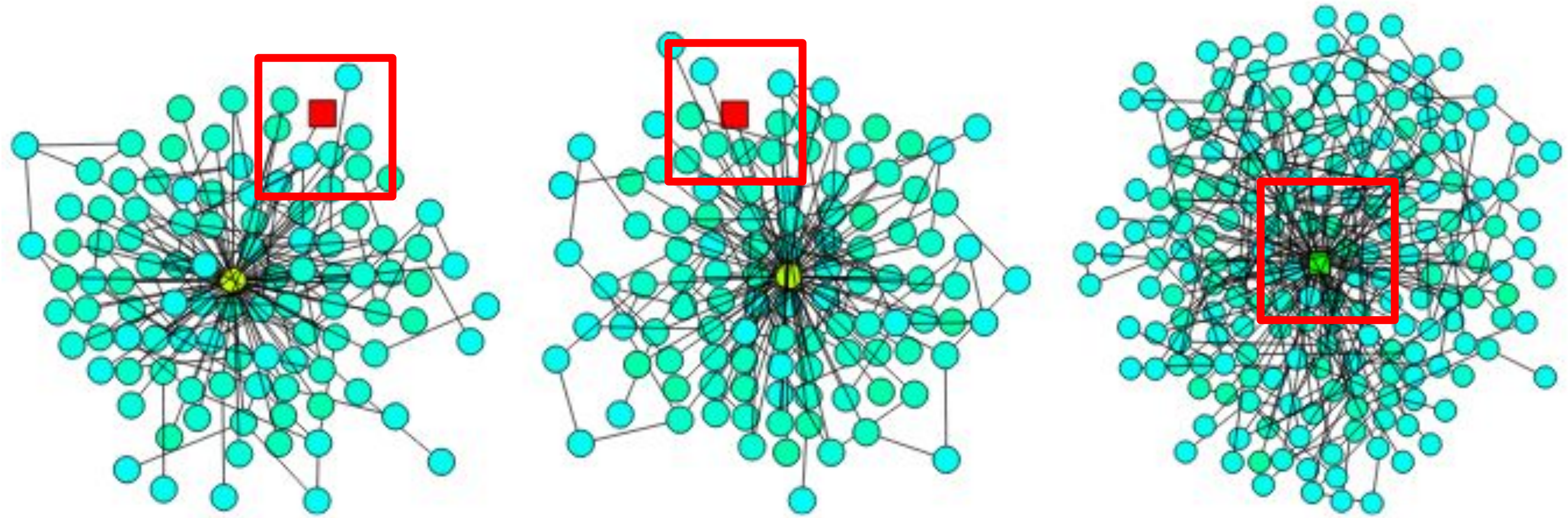
Jumping Knowledge Networks



(c) affiliate to the hub (d) affiliate to the hub (e) hub

Figure 5. A 6-layer JK-Net learns to adapt to different subgraph structures

Jumping Knowledge Networks



(c) affiliate to the hub (d) affiliate to the hub (e) hub

Figure 5. A 6-layer JK-Net learns to adapt to different subgraph structures

Jumping Knowledge Networks

Dataset	Nodes	Edges	Classes	Features
Citeseer	3,327	4,732	6	3,703
Cora	2,708	5,429	7	1,433
Reddit	232,965	avg deg 492	50	300
PPI	56,944	818,716	121	50

Table 1. Dataset statistics

Jumping Knowledge Networks

Model	Citeseer	Model	Cora
GCN (2)	77.3 (1.3)	GCN (2)	88.2 (0.7)
GAT (2)	76.2 (0.8)	GAT (3)	87.7 (0.3)
JK-MaxPool (1)	77.7 (0.5)	JK-Maxpool (6)	**89.6** (0.5)
JK-Concat (1)	**78.3** (0.8)	JK-Concat (6)	89.1 (1.1)
JK-LSTM (2)	74.7 (0.9)	JK-LSTM (1)	85.8 (1.0)

Table 2. Results of GCN-based JK-Nets on Citeseer and Cora. The baselines are GCN and GAT. The number in parentheses next to the model name indicates the best-performing number of layers among 1 to 6. Accuracy and standard deviation are computed from 3 random data splits.

Jumping Knowledge Networks

Model	Citeseer	Model	Cora
GCN (2)	77.3 (1.3)	GCN (2)	88.2 (0.7)
GAT (2)	76.2 (0.8)	GAT (3)	87.7 (0.3)
JK-MaxPool (1)	77.7 (0.5)	JK-Maxpool (6)	**89.6** (0.5)
JK-Concat (1)	**78.3** (0.8)	JK-Concat (6)	89.1 (1.1)
JK-LSTM (2)	74.7 (0.9)	JK-LSTM (1)	85.8 (1.0)

Table 2. Results of GCN-based JK-Nets on Citeseer and Cora. The baselines are GCN and GAT. The number in parentheses next to the model name indicates the best-performing number of layers among 1 to 6. Accuracy and standard deviation are computed from 3 random data splits.

Jumping Knowledge Networks

Node \ JK	GraphSAGE	Maxpool	Concat	LSTM
Mean	0.950	0.953	0.955	0.950
MaxPool	0.948	0.924	**0.965**	0.877

Table 3. Results of GraphSAGE-based JK-Nets on Reddit. The baseline is GraphSAGE. Model performance is measured in Micro-F1 score. Each column shows the results of a JK-Net variant. For all models, the number of layers is fixed to 2.

Jumping Knowledge Networks

Node \ JK	SAGE	MaxPool	Concat	LSTM
Mean (10 epochs)	0.644	0.658	0.667	**0.721**
Mean (30 epochs)	0.690	0.713	0.694	**0.818**
MaxPool (10 epochs)	0.668	0.671	0.687	0.621*

Table 4. Results of GraphSAGE-based JK-Net on the PPI data. The baseline is GraphSAGE (SAGE). Each column, excluding SAGE, represents a JK-Net with different layer aggregation. All models use 3 layers, except for those with "*", whose number of layers is set to 2 due to GPU memory constraints. 0.6 is the corresponding 2-layer GraphSAGE performance.

Jumping Knowledge Networks

Model	PPI
MLP	0.422
GAT	0.968 (0.002)
JK-Concat (2)	0.959 (0.003)
JK-LSTM (3)	0.969 (0.006)
JK-Dense-Concat (2)*	0.956 (0.004)
JK-Dense-LSTM (2)*	**0.976** (0.007)

Table 5. Micro-F1 scores of GAT-based JK-Nets on the PPI data. The baselines are GAT and MLP (Multilayer Perceptron). While the number of layers for JK-Concat and JK-LSTM are chosen from $\{2, 3\}$, the ones for JK-Dense-Concat and JK-Dense-LSTM are directly set to 2 due to GPU memory constraints.

References

- Representation Learning on Graphs with Jumping Knowledge Networks

Thank You