Just Jump: Dynamic Neighborhood Aggregation in Graph Neural Networks

tu dortmund university

Matthias Fey

SFB 876 Providing Information by Resource-Constrained Data Analysis

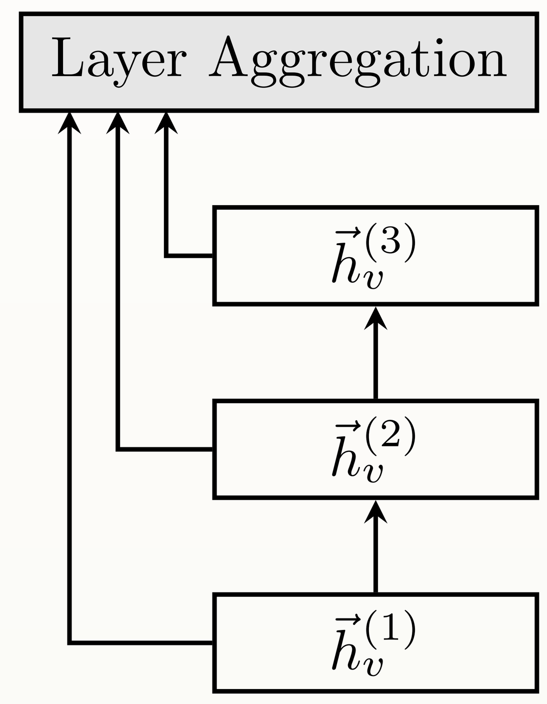
Motivation and Related Work

Graph Neural Networks (GNNs) iteratively update node features by aggregating localized information:

$$\vec{h}_v^{(t)} = f_{\mathbf{\Theta}}^{(t)} \left(\vec{h}_v^{(t-1)}, \left\{ \vec{h}_w^{(t-1)} : w \in \mathcal{N}(v) \right\} \right)$$

Empirically observed: gradually **decreasing performance** when **deeply stacking** those layers

Theoretically explained: varying speed of expansion due to **structure-dependent influence radii**



Jumping Knowledge (JK) enables deeper GNNs by layer-wise jump connections:

$$\vec{h}_v^{(final)} = g \left(\vec{h}_v^{(1)}, \ldots, \vec{h}_v^{(T)} \right)$$

(*e.g.*, concatenation, max-pooling, attention)

✓ Model **adapts the neighborhood size** for each node as needed

Dynamic Neighborhood Aggregation (DNA)

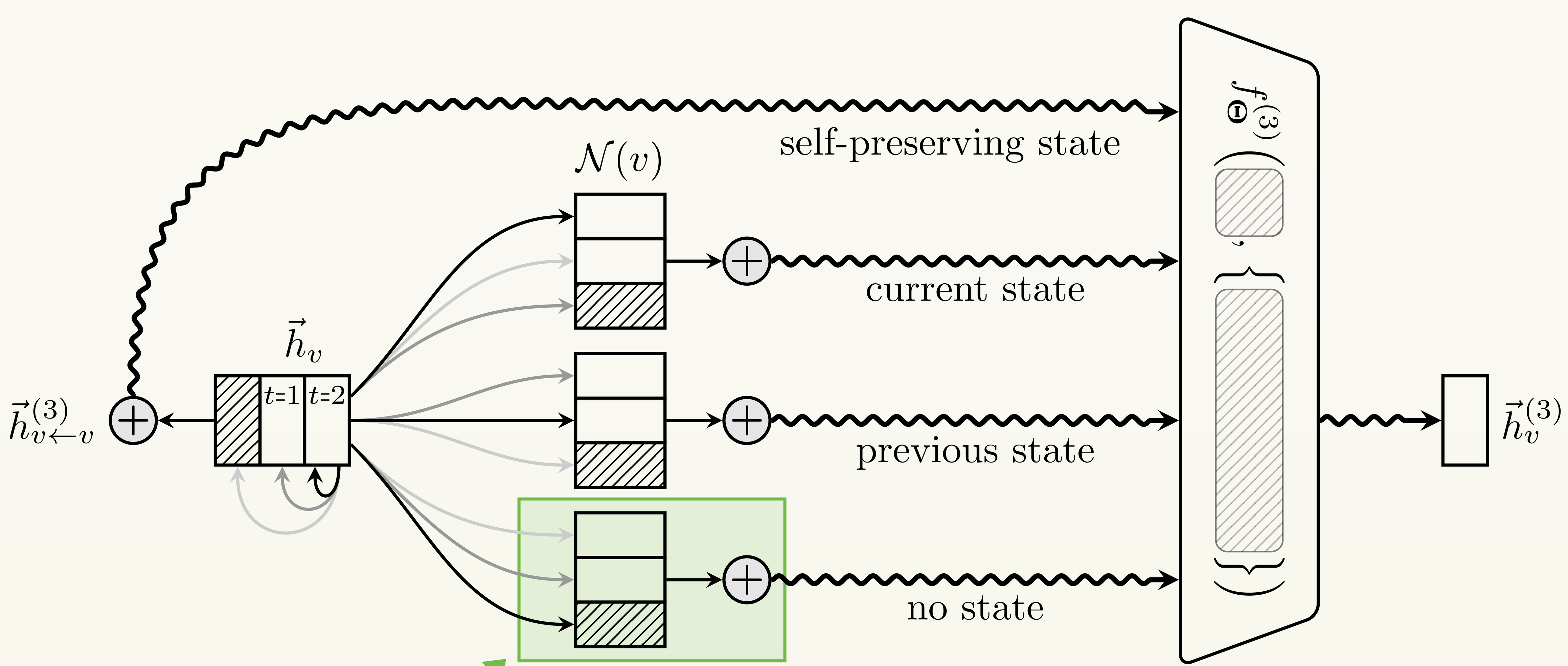
JK does **not prevent "washed out" representations** in later layers

Here: Allow jumps **directly while aggregating** information

✓ each node can dynamically craft its own receptive field, *e.g.*, aggregate local and global information from different branches

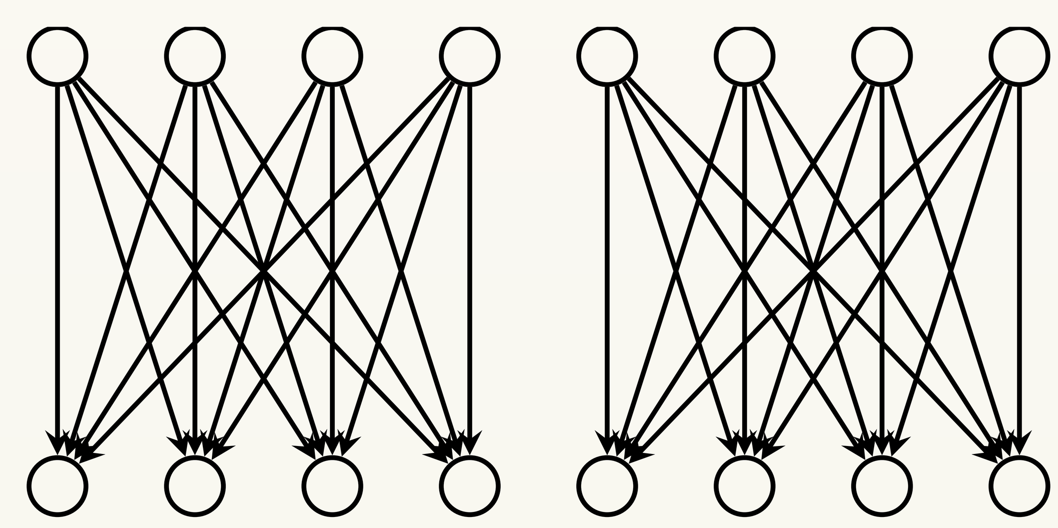
$$\vec{h}_v^{(t)} = f_{\mathbf{\Theta}}^{(t)} \left(\vec{h}_{v \leftarrow v}^{(t)}, \left\{ \vec{h}_{v \leftarrow w}^{(t)} : w \in \mathcal{N}(v) \right\} \right)$$

$$\vec{h}_{v \leftarrow w}^{(t)} = \mathrm{Attention} \left(\mathbf{\Theta}_Q^{(t)} \vec{h}_v^{(t-1)}, \left[\vec{h}_w^{(1)}, \ldots, \vec{h}_w^{(T)} \right]^\top \mathbf{\Theta}_K^{(t)} \right)$$



(Multi-headed) dot-product attention with adjusted $\mathrm{softmax}(\vec{x})_i = \dfrac{\exp(x_i)}{1 + \sum_j \exp(x_j)}$

Regularization via **grouped linear projections:**



$g = 2$ (Number of groups)

✓ Attention heads only have **local influence** on other heads

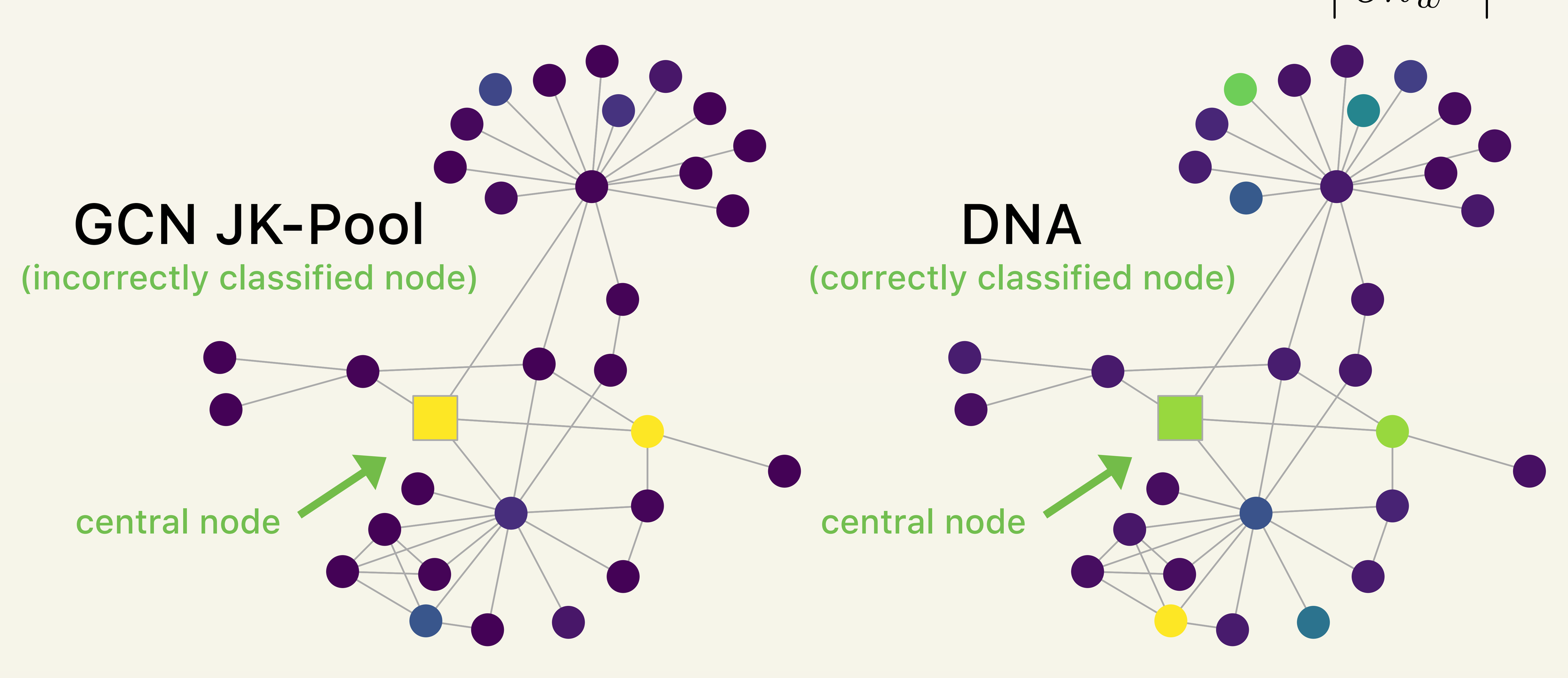
✓ Avoids overfitting while maintaining **large feature dimensionality**

Quantitative and Qualitative Evaluation on Transductive Benchmark Datasets

	Model	Cora	CiteSeer	PubMed	Cora Full
GCN	JK-None	83.20 ± 0.98	**73.87 ± 0.81**	86.93 ± 0.25	62.55 ± 0.60
GCN	JK-Concat	83.99 ± 0.72	73.77 ± 0.89	87.52 ± 0.25	**65.62 ± 0.49**
GCN	JK-Pool	**84.36 ± 0.62**	73.86 ± 0.97	**87.61 ± 0.27**	65.14 ± 0.81
GCN	JK-LSTM	80.46 ± 0.88	72.92 ± 0.69	87.38 ± 0.29	55.39 ± 0.40
DNA	$g = 1$	83.88 ± 0.50	73.37 ± 0.83	87.80 ± 0.25	63.72 ± 0.44
DNA	$g = 8$	85.86 ± 0.45	74.19 ± 0.66	**88.04 ± 0.17**	66.50 ± 0.42
DNA	$g = 16$	**86.15 ± 0.57**	**74.50 ± 0.62**	88.04 ± 0.22	**66.64 ± 0.47**

	Model	Coauthor CS	Coauthor Physics	Amazon Computers	Amazon Photo
GCN	JK-None	92.90 ± 0.14	95.90 ± 0.16	89.32 ± 0.20	93.11 ± 0.27
GCN	JK-Concat	95.44 ± 0.32	96.71 ± 0.15	90.27 ± 0.28	**94.74 ± 0.29**
GCN	JK-Pool	**95.47 ± 0.21**	**96.74 ± 0.17**	90.30 ± 0.37	94.64 ± 0.24
GCN	JK-LSTM	94.40 ± 0.28	96.55 ± 0.08	90.06 ± 0.23	94.54 ± 0.30
DNA	$g = 1$	94.02 ± 0.17	96.49 ± 0.10	90.52 ± 0.40	94.89 ± 0.26
DNA	$g = 8$	94.46 ± 0.15	**96.58 ± 0.09**	**90.99 ± 0.40**	94.96 ± 0.24
DNA	$g = 16$	94.64 ± 0.15	96.53 ± 0.10	90.81 ± 0.38	**95.00 ± 0.19**

Visualization of the **influence score** $I_v(w) = \mathbf{1}^\top \left| \dfrac{\partial \vec{h}_v^{(T)}}{\partial \vec{h}_w^{(0)}} \right| \mathbf{1}$:



DNA can aggregate **localized information** even from nodes **far away!**

Code is integrated as part of /rusty1s/pytorch_geometric