A presentation and review of the publication:

Neural Trees for Learning on Graphs

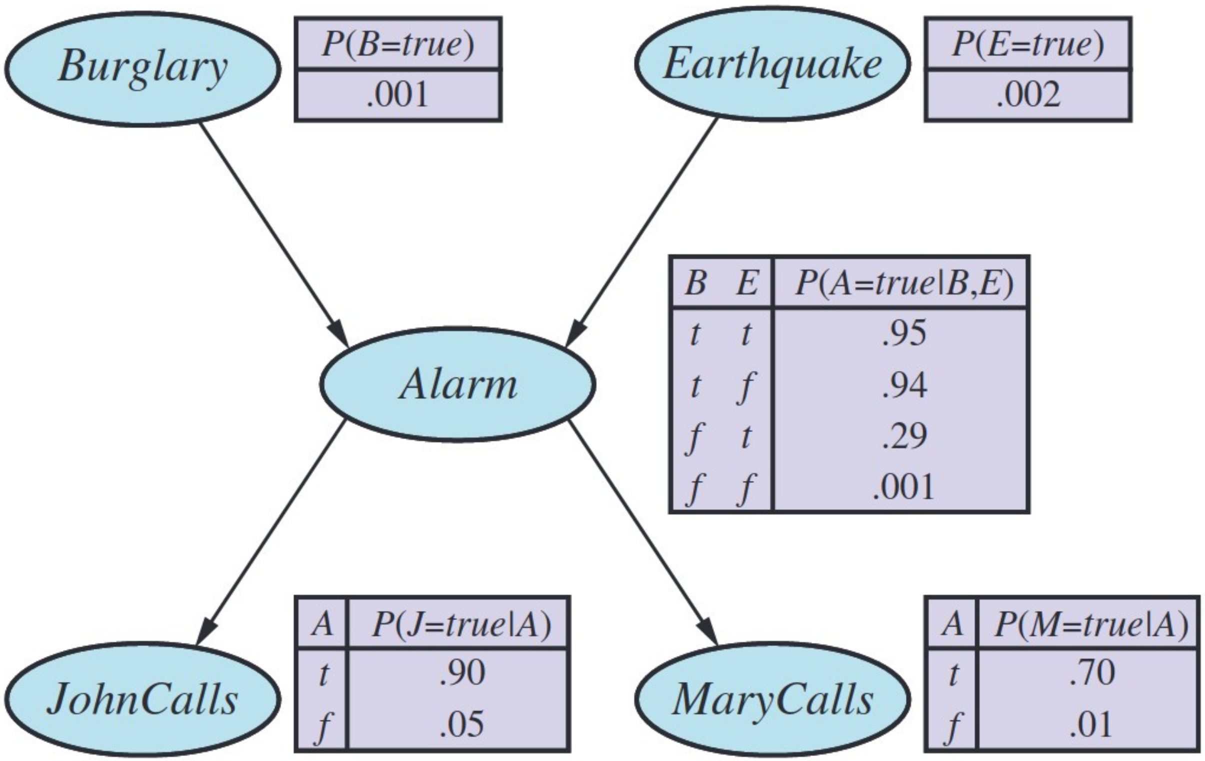
By Rajat Talak, Siyi Hu, Lisa Peng, and Luca Carlone

Presenter: Niklas Vest
Advisor: Evin Pınar Örnek

Technische Universität München + Johns Hopkins Whiting School of Engineering

Motivation

- Standard GNNs cannot capture basic graph properties
- At least *Probabilistic Graphical Model*?



Source: S. Russel, P. Norvig, 2020 (Artificial Intelligence - A modern approach)

Contributions

- *G-compatibility*
- *Neural Tree* architecture

Reasoning

- Approximating *G-compatible* function = Approximating PGM
- *Neural Tree* architecture can approximate* G-compatible function

Definitions (1) – Graphs for Semi-Supervised Learning

$$\mathcal{G} = (\mathcal{V}, \mathcal{E})$$

$$\boldsymbol{X} = (\boldsymbol{x}_v)_{v \in \mathcal{V}}$$

$\{z_v \in \mathcal{L} \mid v \in \mathcal{A}\}$ for a subset $\mathcal{A} \subset \mathcal{V}$

Goal: Predict labels for all $v \in \mathcal{V} \backslash \mathcal{A}$

Definitions (2) – GNNs and Node Classification

$$h_v^t = \mathrm{AGG}_t \left(h_v^{t-1}, \left\{ \left(h_u^{t-1}, \kappa_{u,v} \right) \mid u \in \mathcal{N}_{\mathcal{G}} \left(v \right) \right\} \right)$$

$$y_v = \mathrm{READ}(h_v^T)$$

Definitions (3) – Graph terminology

- **Clique**
  A set of a graph's nodes that induce a complete subgraph.
- **Tree Decomposition**
  *A tree-shaped representation of a Graph. The nodes are **sets of** the graph's vertices, called **bags**.*
  - *For a vertex of the graph, the bags that contain this vertex must induce a subtree of the decomposition.*
  - *If two vertices share an edge, at least one bag must contain both vertices.*
- **Treewidth**
  The largest bag of any of a graph's tree decompositions minus one.

Definitions (4) – Graph functions

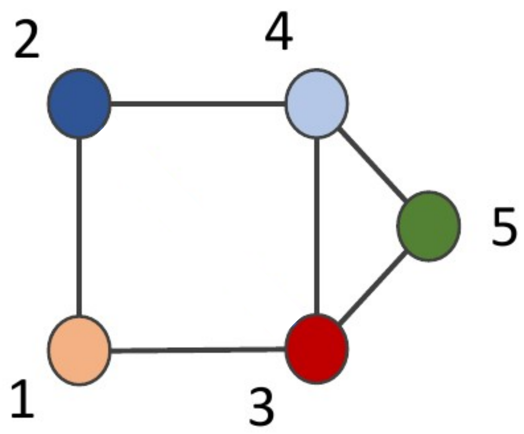
G-Invariant function. A function $f : \mathbb{R}^m \to \mathbb{R}^n$ is called graph invariant if it satisfies $f(\varphi(g) \cdot x) = f(x)$ for all $x \in \mathbb{R}^n$ and $g \in G$ and a homomorphism $\varphi : G \to S_m$

G-Equivariant function. A function $f : \mathbb{R}^m \to \mathbb{R}^n$ is called graph equivariant if it satisfies $f(\varphi(g) \cdot x) = \psi(g) \cdot f(x)$ for all $g \in G$ and a homomorphism $\psi : G \to S_n$.

G-Compatible function. A function $f : (\times_{v \in V} \mathbb{X}_v, G) \to \mathbb{R}$ is compatible with a graph G if it can be factorized as

$$f(\mathbf{X}) = \sum_{c \in C(G)} \theta_c(\mathbf{x}_C)$$

Standard GNNs vs Neural Tree architecture (1)

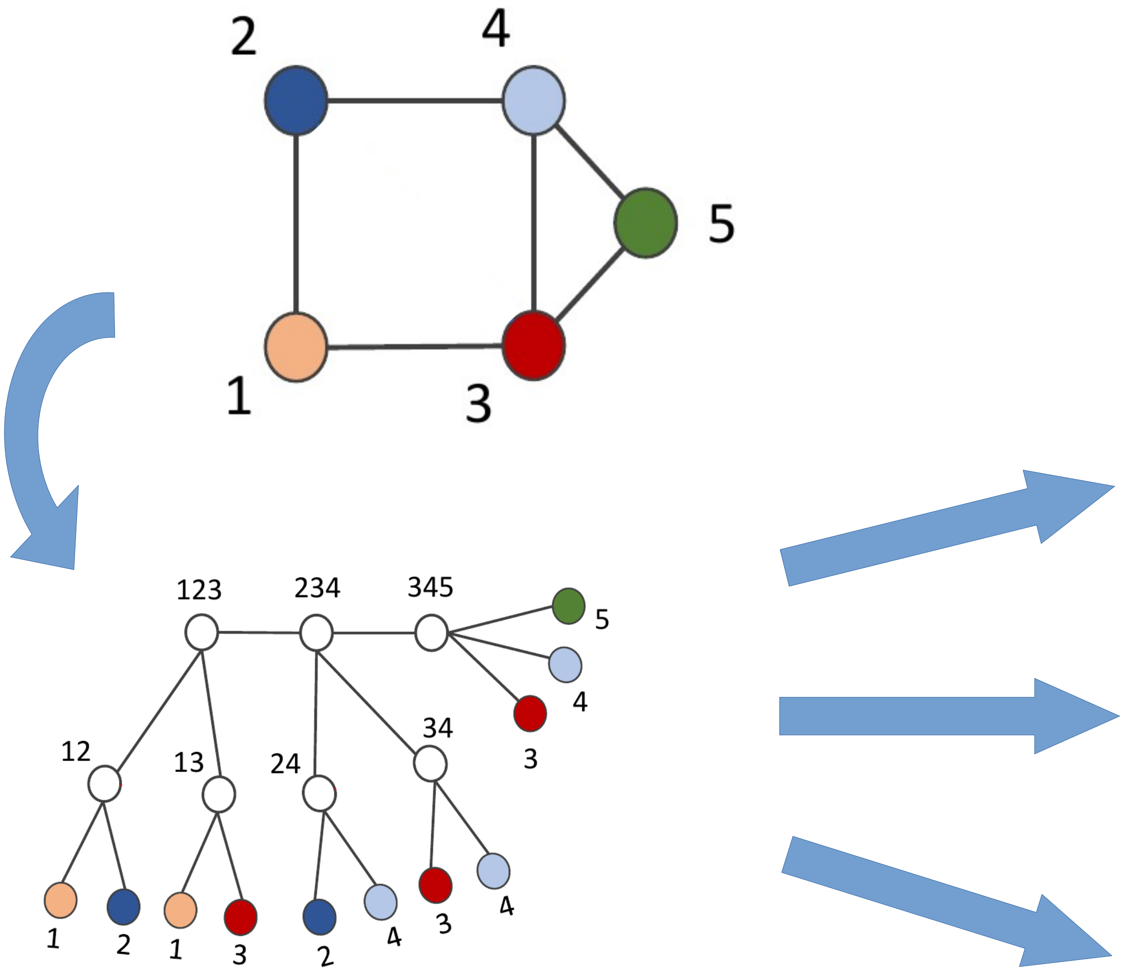


GCN

GAT

GraphSAGE

Standard GNNs vs Neural Tree architecture (2)

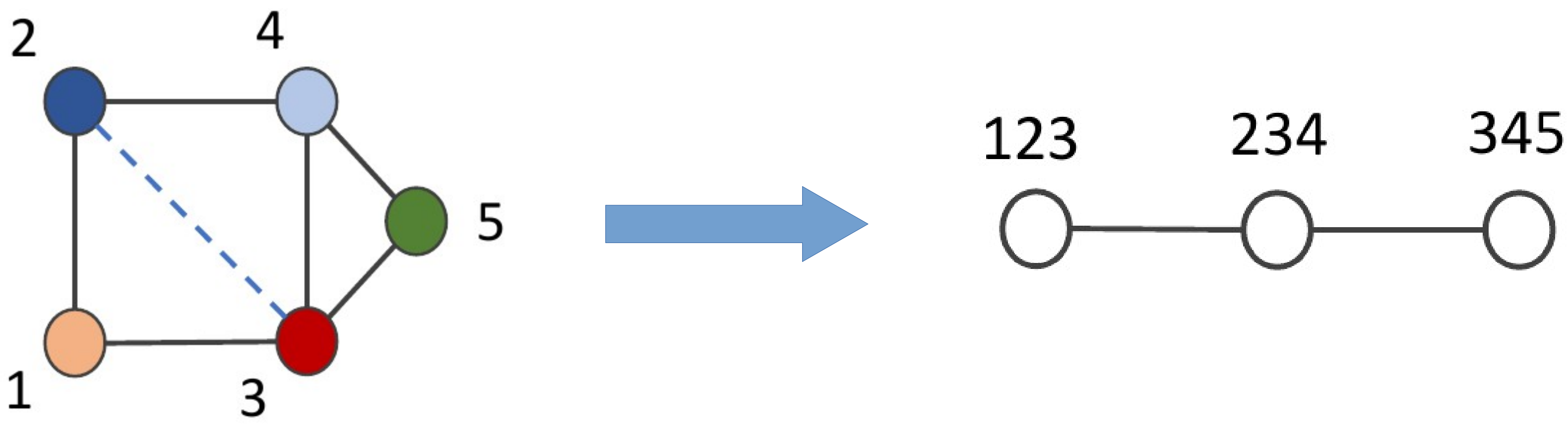


GCN

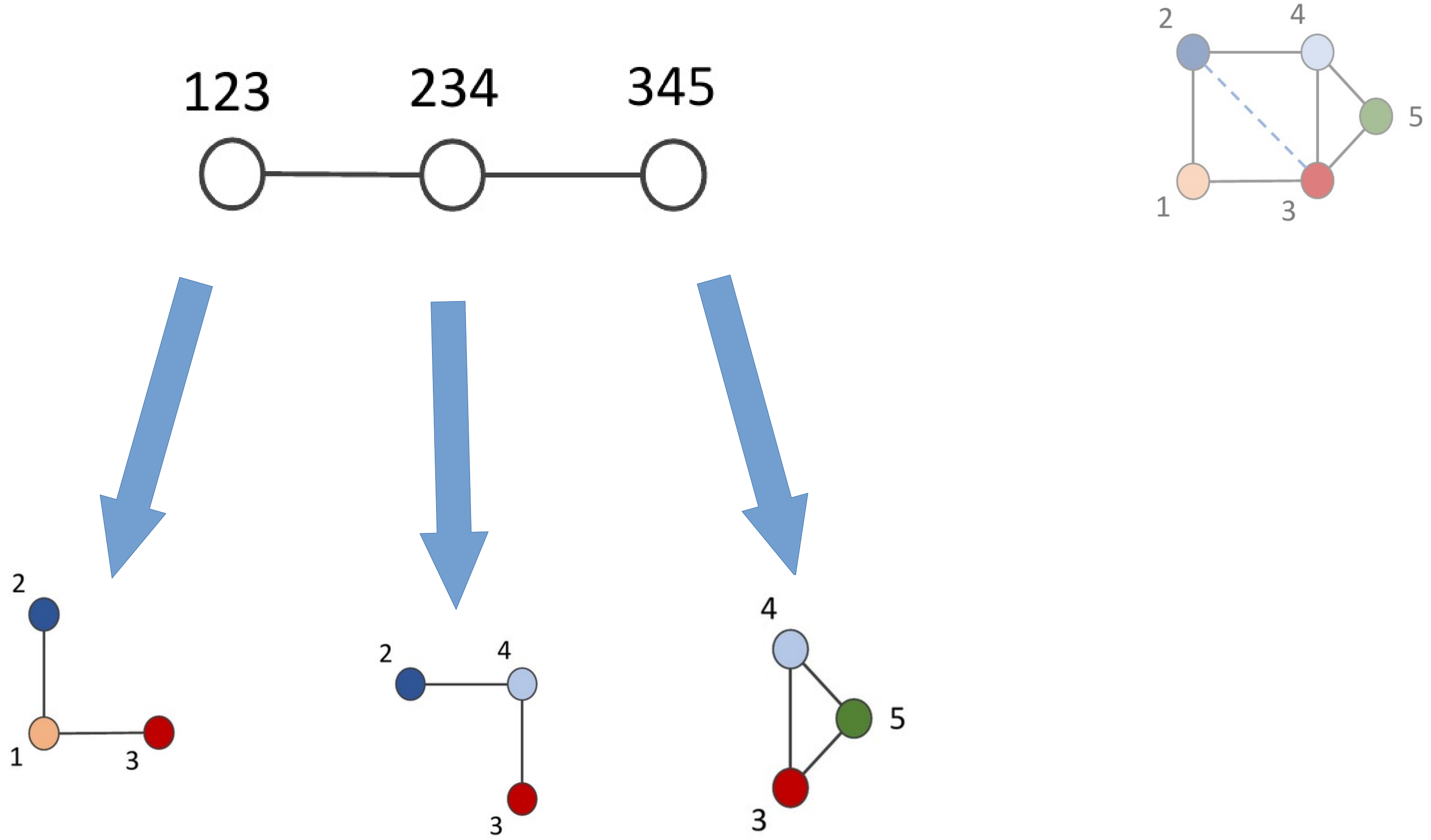
GAT

GraphSAGE

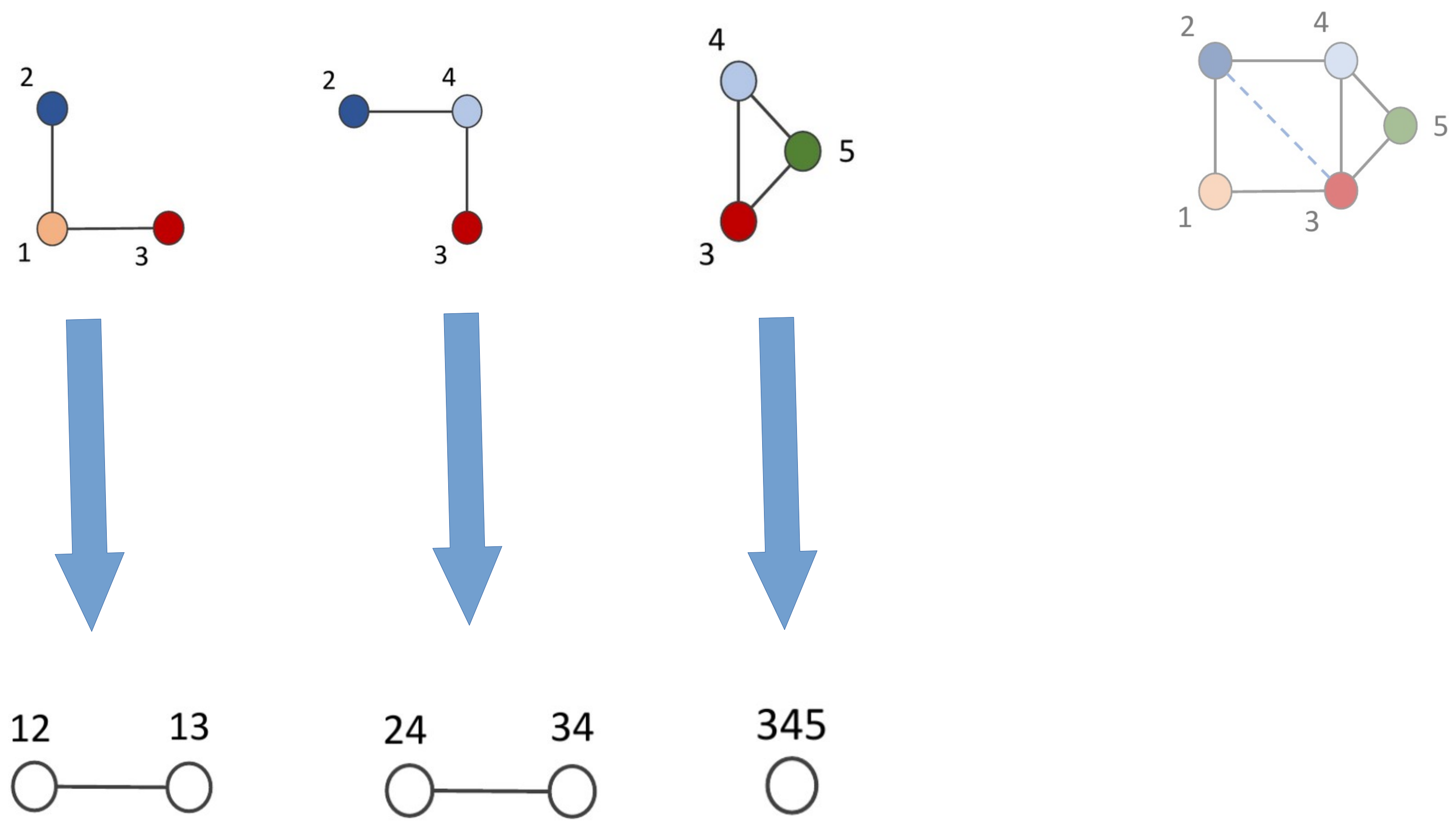
H-Tree (1) – Tree Decomposition of the Graph



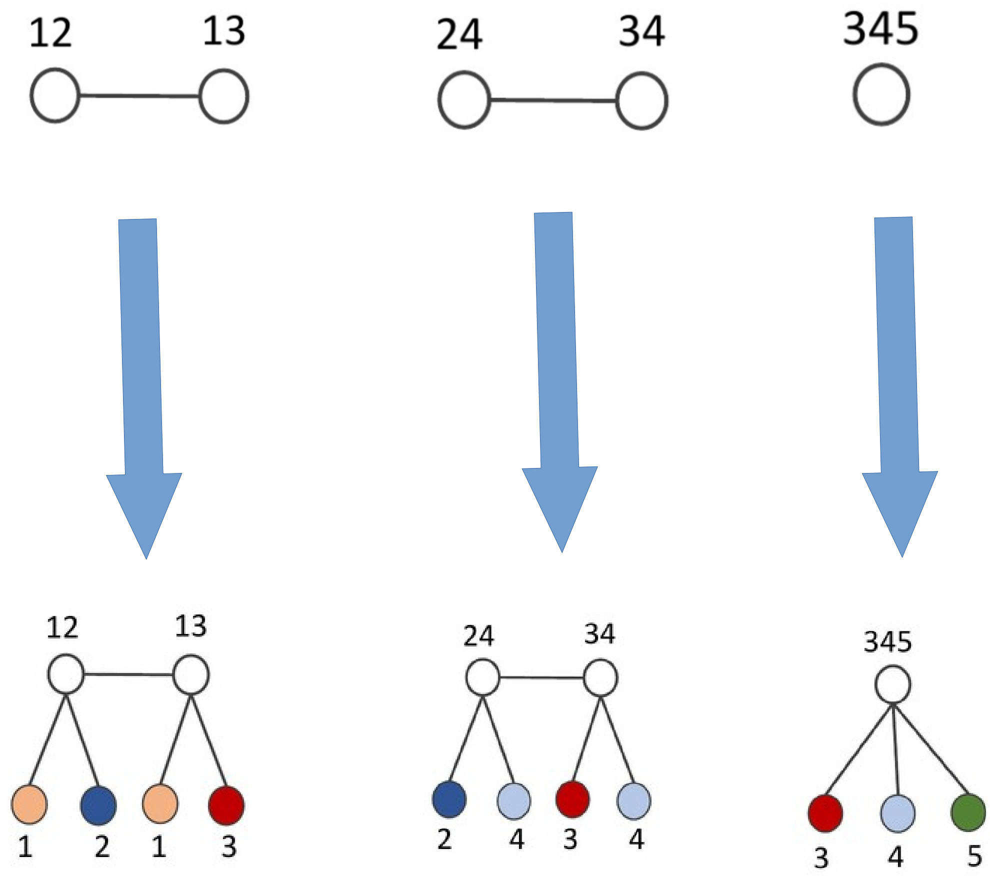
H-Tree (2) – Identify subgraphs induced by bags

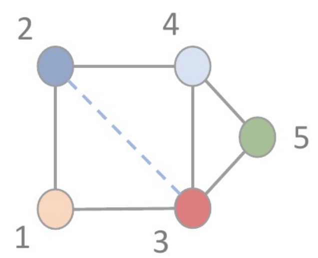


H-Tree (3) – Apply recursively

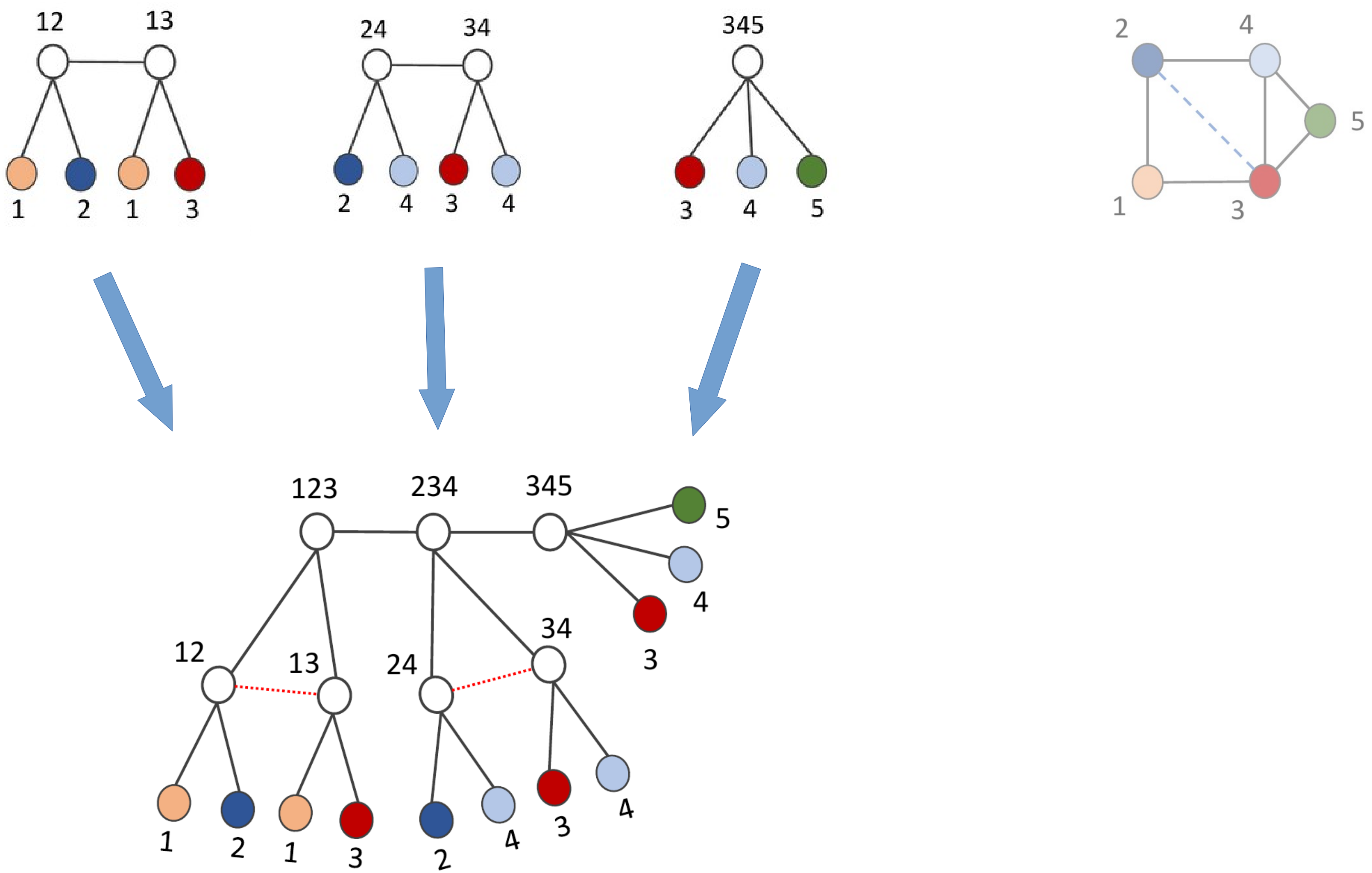


H-Tree (4) – Merge results (recursively)

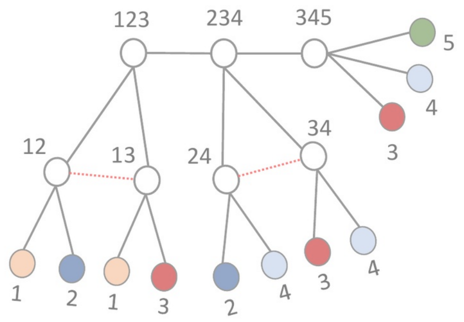




H-Tree (5) – Final H-Tree



Neural Tree Architecture



$$h_v^t = \text{AGG}_t\left(h_v^{t-1}, \left\{\left(h_u^{t-1}, \kappa_{u,v}\right) \mid u \in \mathcal{N}_{\mathcal{G}}(v)\right\}\right)$$

$$\boldsymbol{h}_u^t = \text{AGG}_t\left(\boldsymbol{h}_u^{t-1}, \left\{\left(\boldsymbol{h}_w^{t-1}, \kappa_{w,u}\right) \mid w \in \mathcal{N}_{\mathcal{J}_\mathcal{G}}(u)\right\}\right)$$

$$y_v = \text{READ}(h_v^T)$$

$$y_v = \text{COMB}\left(\left\{\boldsymbol{h}_l^T \mid l \text{ leaf node in } \mathcal{J}_\mathcal{G} \text{ s.t. } \kappa(l) = v\right\}\right)$$

Parameter Bounds (1)

$$\boldsymbol{h}_u^t = \mathrm{AGG}_t\left(\boldsymbol{h}_u^{t-1}, \{(\boldsymbol{h}_w^{t-1}, \kappa_{w,u}) \mid w \in \mathcal{N}_{\mathcal{J}_\mathcal{G}}(u)\}\right)$$

$$= \mathtt{ReLU}\left(\sum_{k=1}^{N_u} a_{u,t}^k \langle \boldsymbol{w}_{u,t}^k, \boldsymbol{h}_{\bar{\mathcal{N}}(u)}^{t-1}\rangle + b_{u,t}^k\right)$$

$$N = \mathcal{O}\left(n \times (tw\left[\mathcal{J}_\mathcal{G}\right] + 1)^{2tw[\mathcal{J}_\mathcal{G}]+3} \times \epsilon^{-(tw[\mathcal{J}_\mathcal{G}]+1)}\right)$$

Parameter Bounds (2)

$$h_u^t = \mathrm{AGG}_t\left(h_u^{t-1}, \{(h_w^{t-1}, \kappa_{w,u}) \mid w \in \mathcal{N}_{\mathcal{J}_\mathcal{G}}(u)\}\right)$$

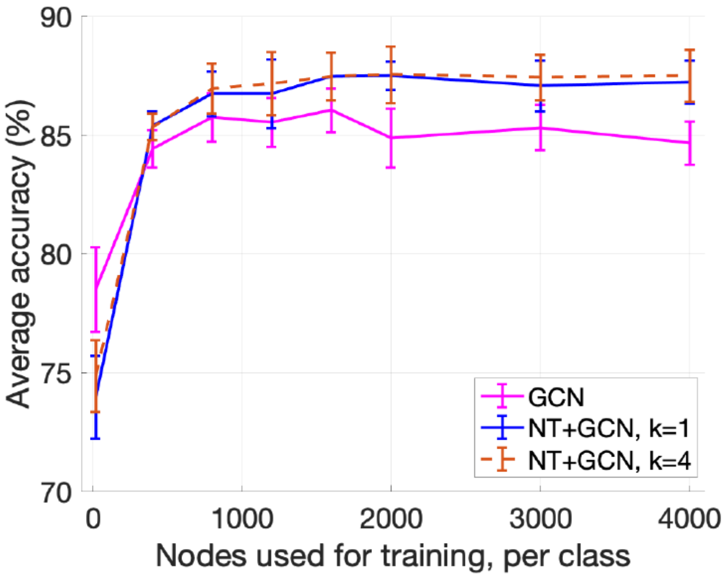
$$= \mathtt{ReLU}\left(\sum_{k=1}^{N_u} a_{u,t}^k \langle w_{u,t}^k, h_{\bar{\mathcal{N}}(u)}^{t-1}\rangle + b_{u,t}^k\right)$$

$$N = \mathcal{O}\left(n \times (tw\,[\mathcal{J}_\mathcal{G}] + 1)^{2tw[\mathcal{J}_\mathcal{G}]+3} \times \epsilon^{-(tw[\mathcal{J}_\mathcal{G}]+1)}\right)$$

- Number of parameters required for ϵ-approximation is
  - **Linear** in the # of nodes
  - **Exponential** in `tw[ ]`
- Low-average-tree-width data = less training data required

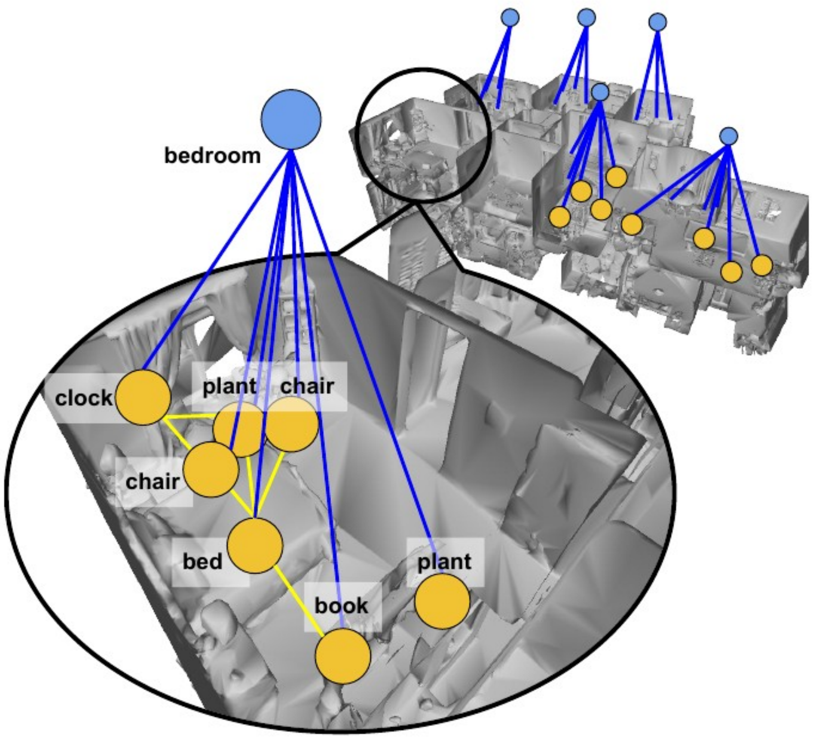
Experiment: Citation Networks

- **Datasets:** PubMed, CiteSeer, Cora
- Node = Document, Edge = Citation, Label = Topic
- CiteSeer: 380.000+ nodes, 1.751.000+ edges
- Citation networks have high treewidth
- *Bounded treewidth graph sub-sampling*



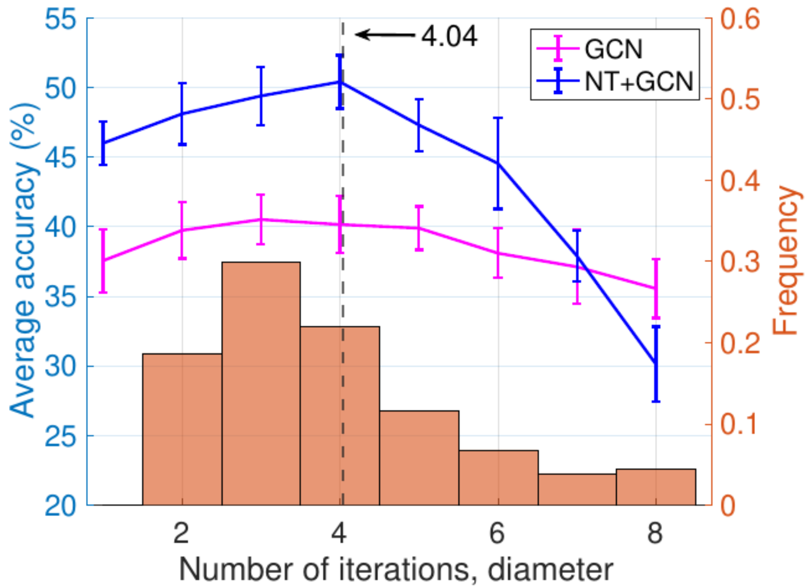
Experiment: 3D scene graphs (1)

- **Datasets:** Stanford 3D scene graph dataset
- Nodes = Objects, Edges = Spacial Adjacency, Label = Semantics
- Nodes and edges in the hundreds to low thousands
- Authors added nearly 5000 edges!



Experiment: 3D scene graphs (2)

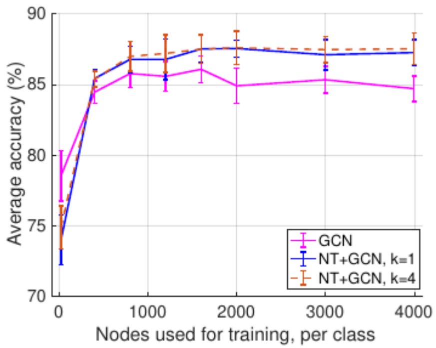
Model	Input graph	Neural Tree
GCN	40.88 ± 2.28 %	$\mathbf{50.63} \pm 2.25$ %
GraphSAGE	59.54 ± 1.35 %	$\mathbf{63.57} \pm 1.54$ %
GAT	46.56 ± 2.21 %	$\mathbf{62.16} \pm 2.03$ %
GIN	49.25 ± 1.15 %	$\mathbf{63.53} \pm 1.38$ %



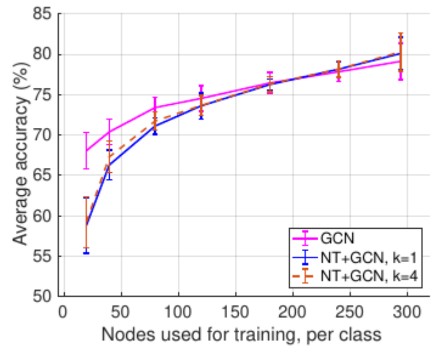
Take Home Message(s)

- Neural Tree architecture = message passing on H-Tree(G)
- Sub-sample G before constructing H-Tree in case of high treewidth
- # of parameters linear in # of nodes, exponential in treewidth

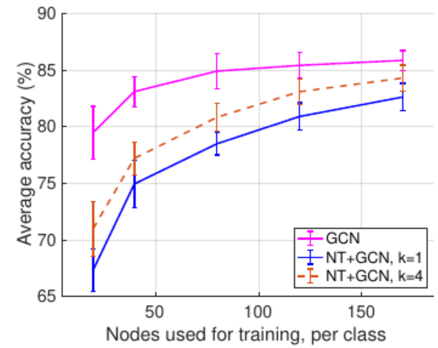
Discussion



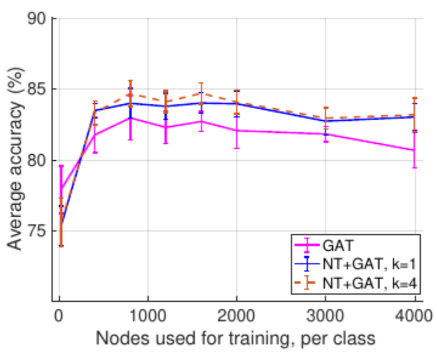
(a) NT+GCN on PubMed



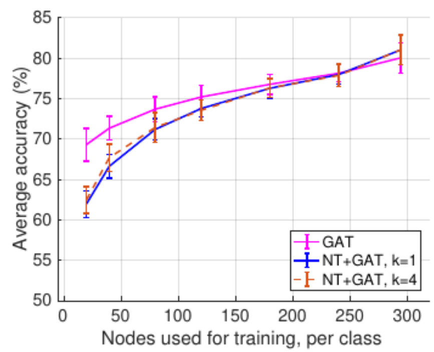
(b) NT+GCN on CiteSeer



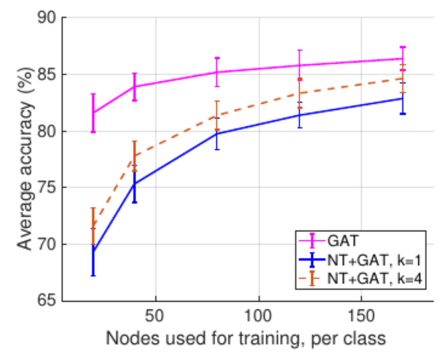
(c) NT+GCN on Cora



(d) NT+GAT on PubMed



(e) NT+GAT on CiteSeer



(f) NT+GAT on Cora

References

1. R. Talak, S. Hu, L. Peng, L. Carlone. Neural Trees for Learning on Graphs. In Neural Information Processing Systems (NeurIPS), 2021.