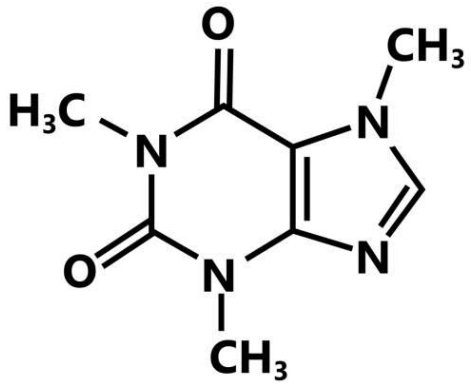
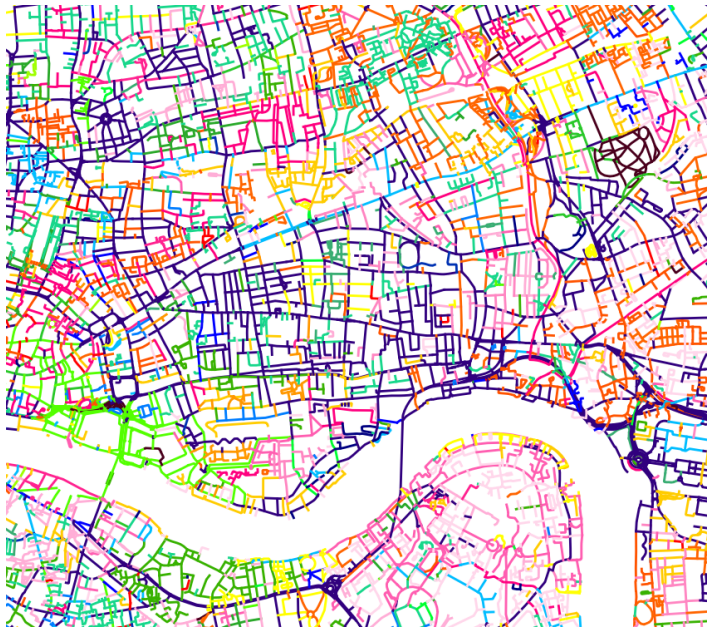
Graph Neural Networks – basics architectures

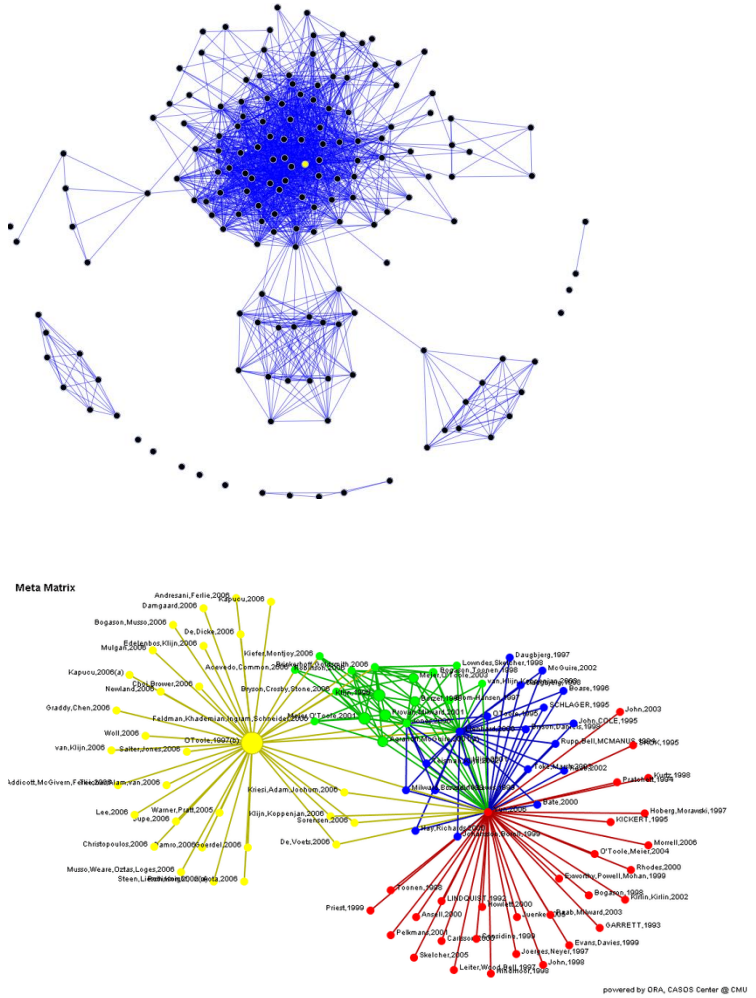
Giorgio Piatti

SiDNN – 08.03.2022

Motivation

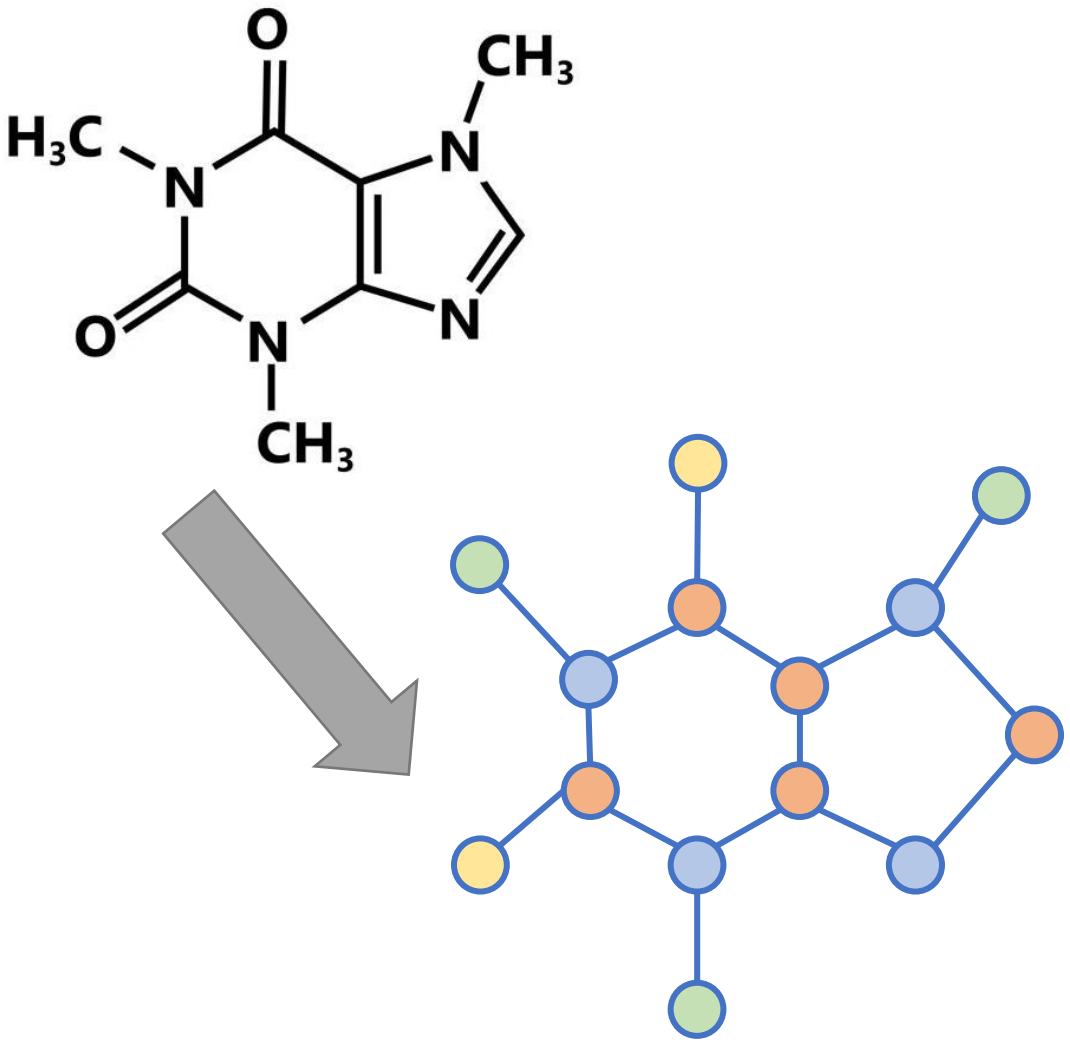

O
CH3
H3C
N
N
O
N
N
CH3




Meta Matrix
powered by ORA, CASOS Center @ CMU

Motivation

- Atoms as nodes
- Bonds as edges
- Features:
  - Atom type
  - Charge
  - Bond type



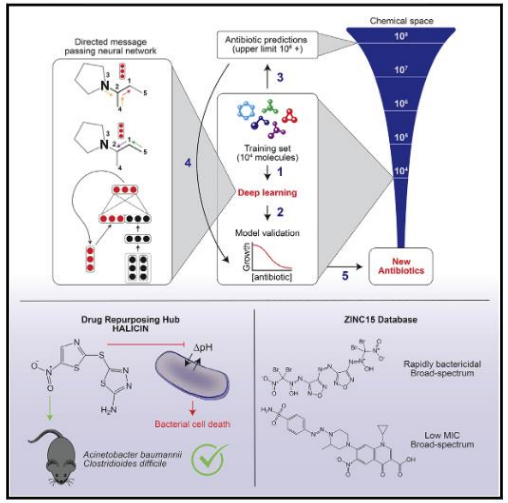
Motivation

Article

Cell

A Deep Learning Approach to Antibiotic Discovery

Graphical Abstract



Authors

Jonathan M. Stokes, Kevin Yang, Kyle Swanson, ..., Tommi S. Jaakkola, Regina Barzilay, James J. Collins

Correspondence

regina@csail.mit.edu (R.B.), jimjc@mit.edu (J.J.C.)

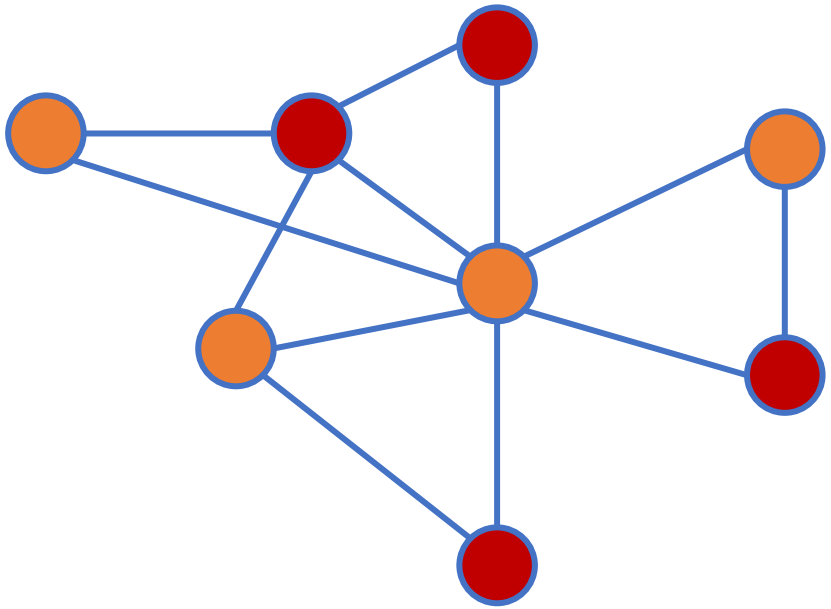
In Brief

A trained deep neural network predicts antibiotic activity in molecules that are structurally different from known antibiotics, among which Halicin exhibits efficacy against broad-spectrum bacterial infections in mice.

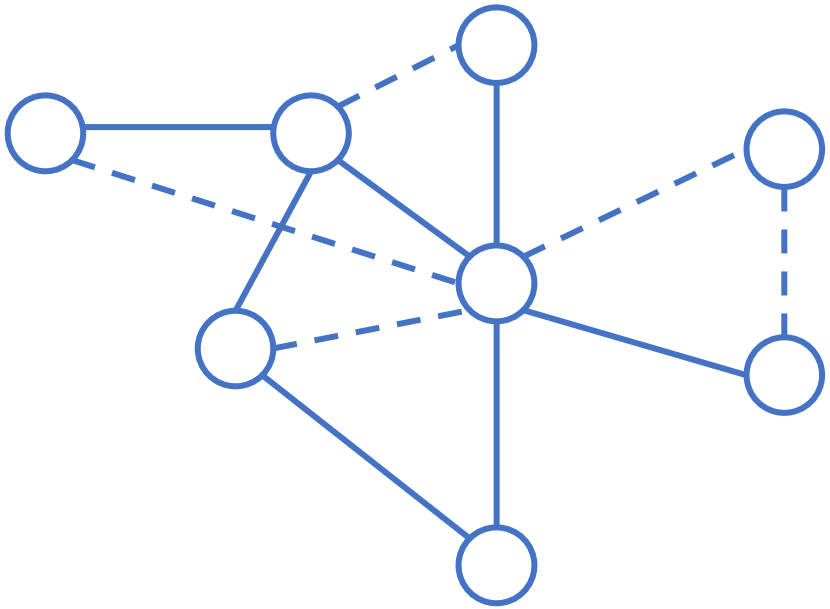
https://doi.org/10.1016/j.cell.2020.01.021

Learning problems

Node classification

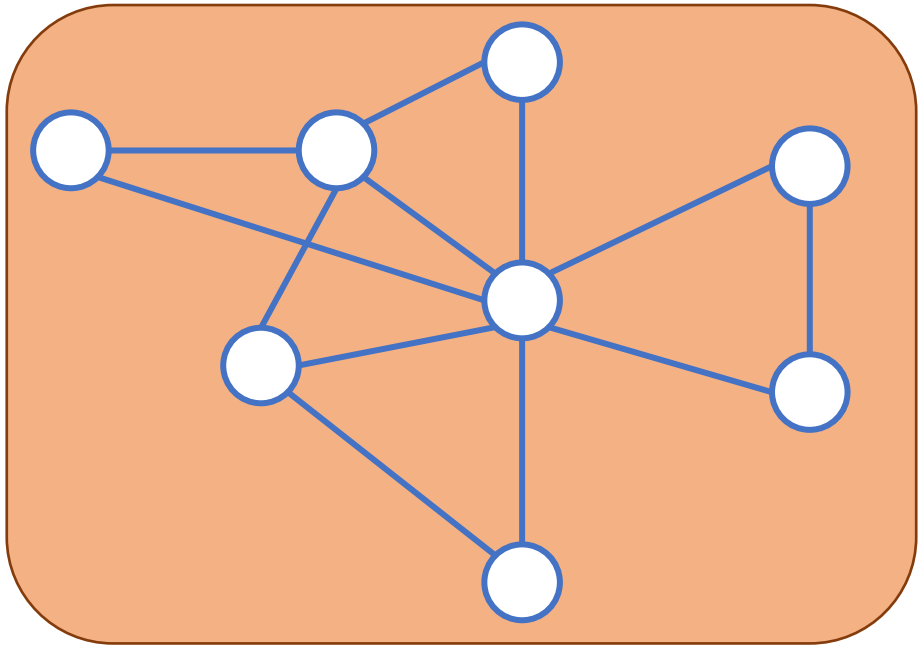


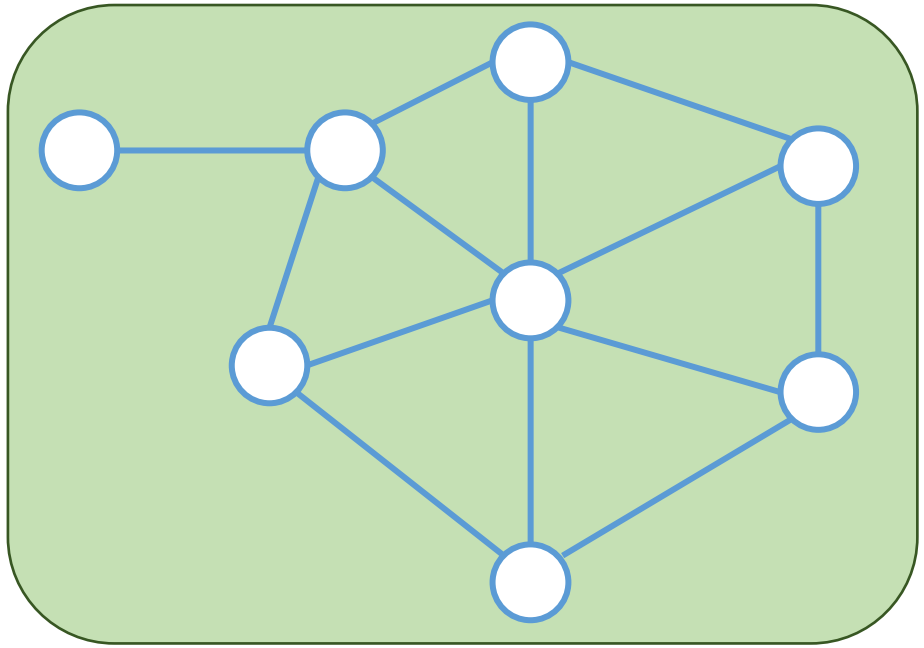
Relation prediction



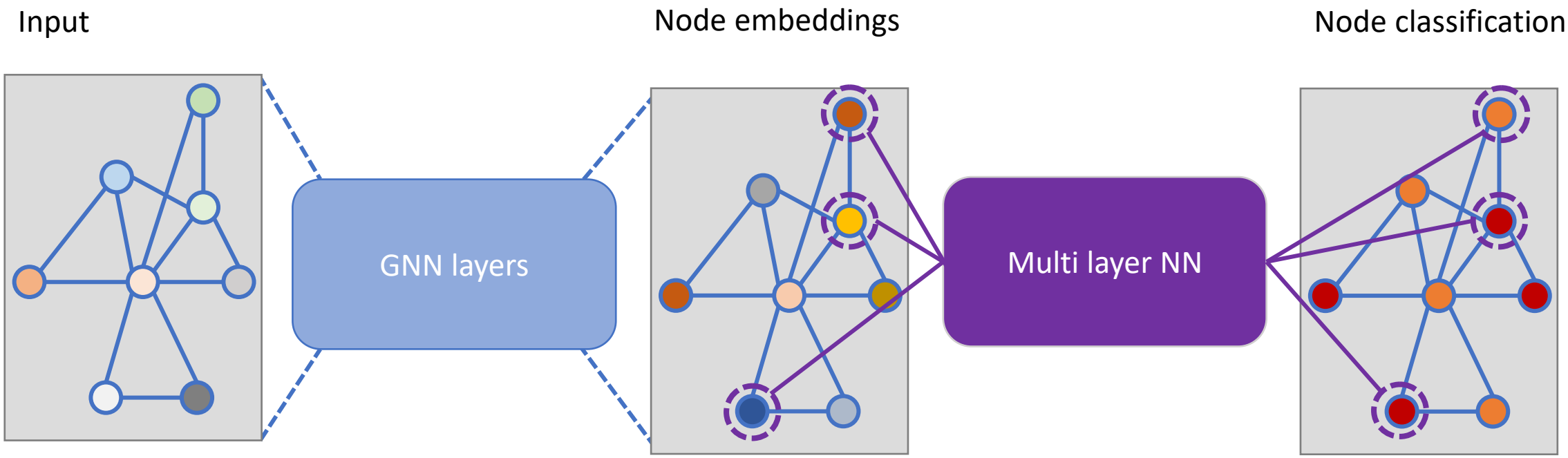
Learning problems

Graph classification





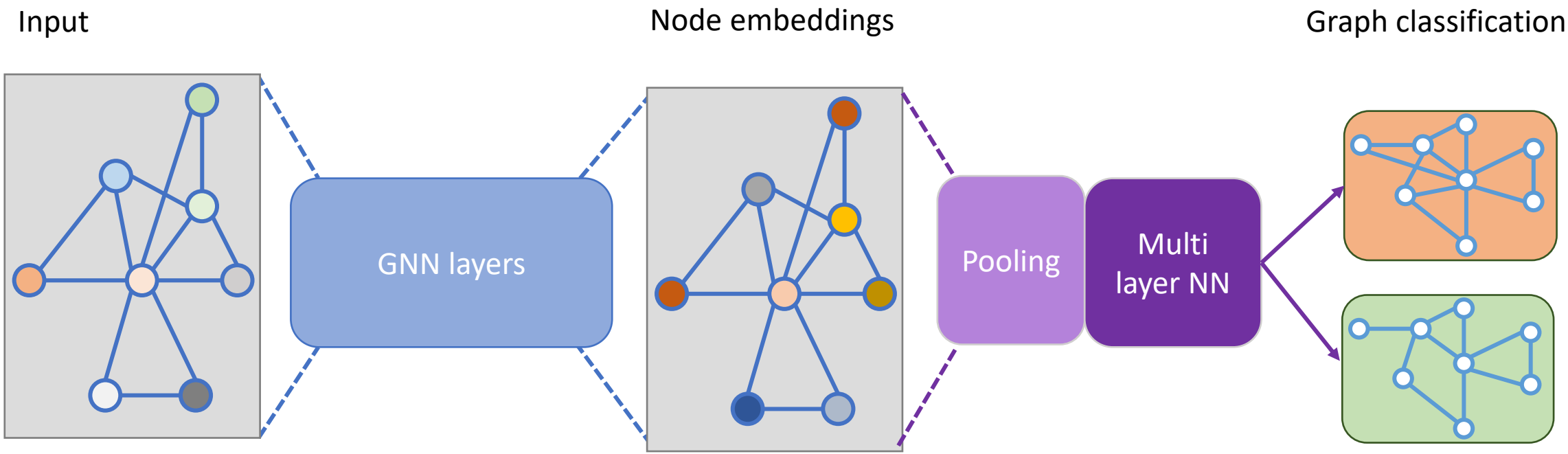
GNN architecture – node classification



A GNN is an optimizable transformation on all attributes of the graph (nodes, edges)

Loss: cross-entropy

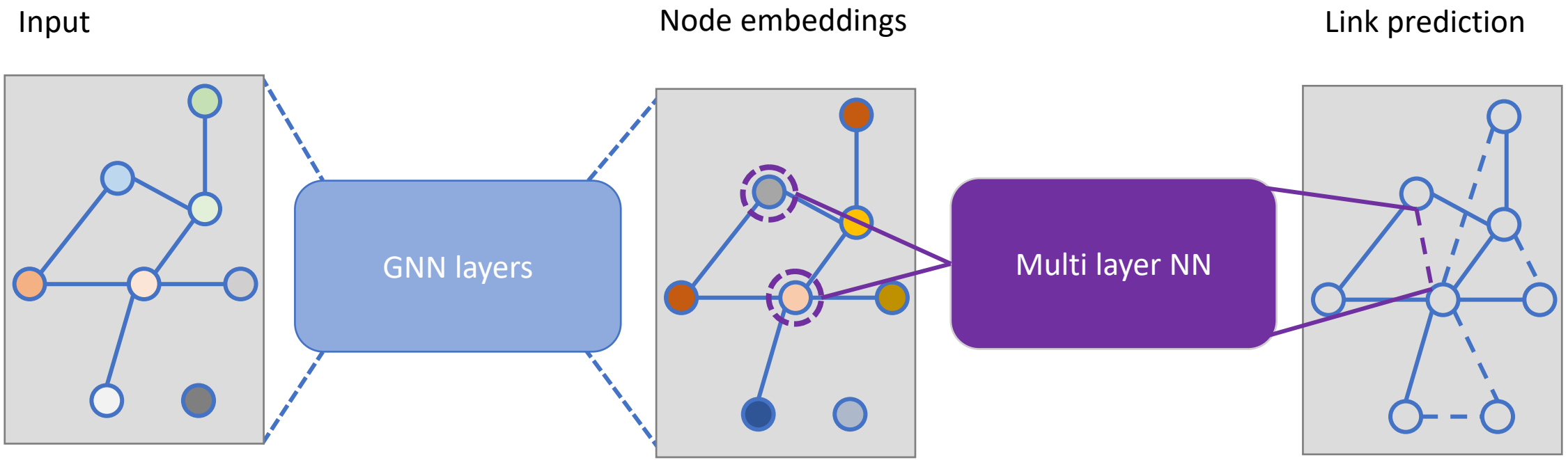
GNN architecture – graph classification



A GNN is an optimizable transformation on all attributes of the graph (nodes, edges)

Loss: cross-entropy

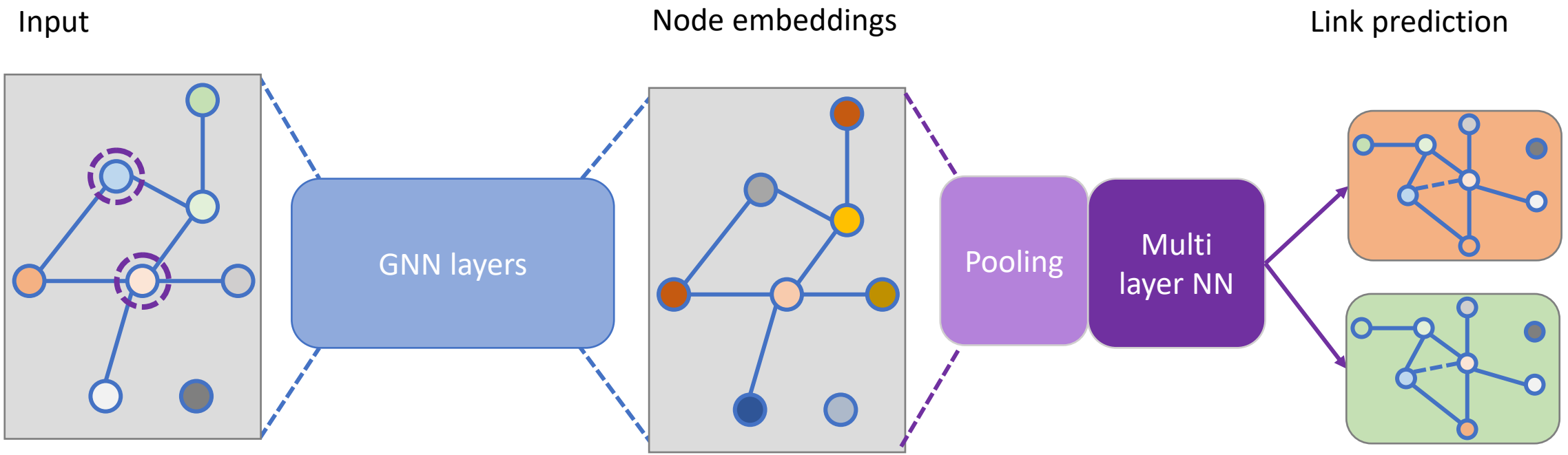
GNN architecture – link prediction



A GNN is an optimizable transformation on all attributes of the graph (nodes, edges)

Loss form: $\mathcal{L}(h_u, h_v, e_{uv})$

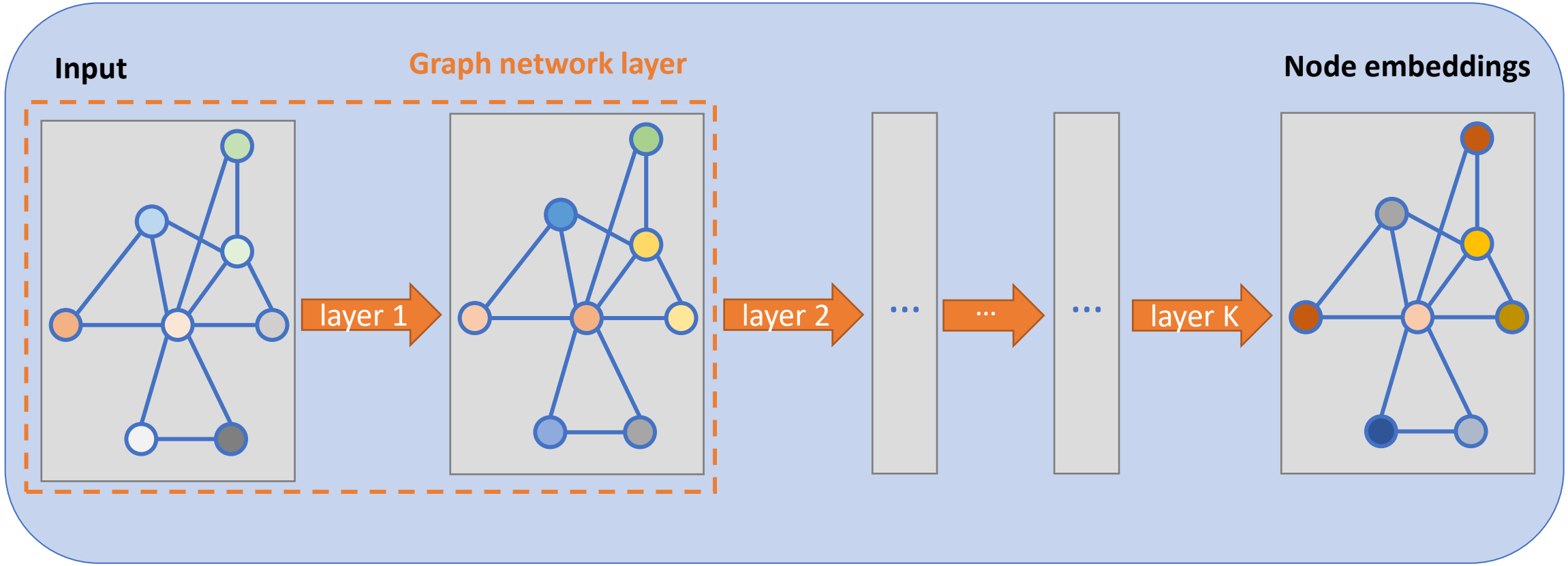
GNN architecture – link prediction



A GNN is an optimizable transformation on all attributes of the graph (nodes, edges)

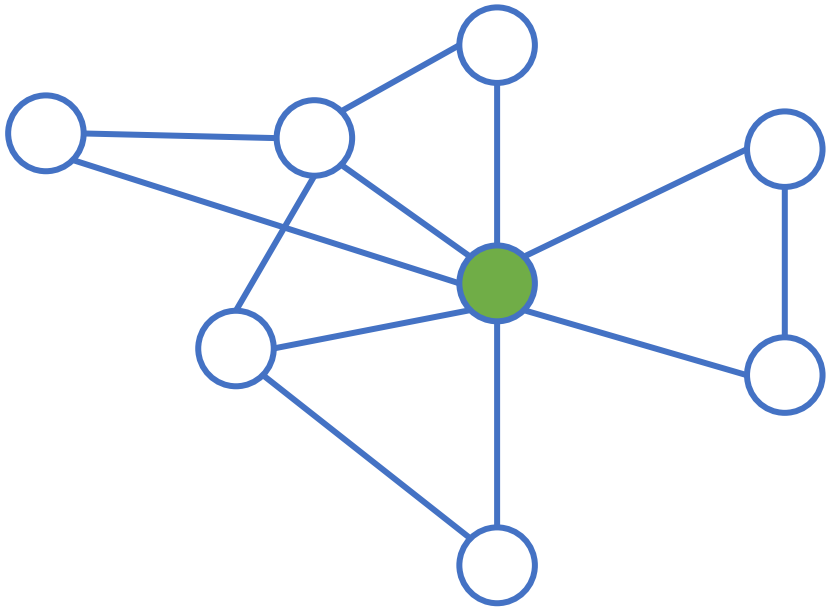
Loss form: $\mathcal{L}(h_u, h_v, e_{uv})$

GNN layers


Input
Graph network layer
Node embeddings
layer 1
layer 2
...
...
...
layer K

GNN layers

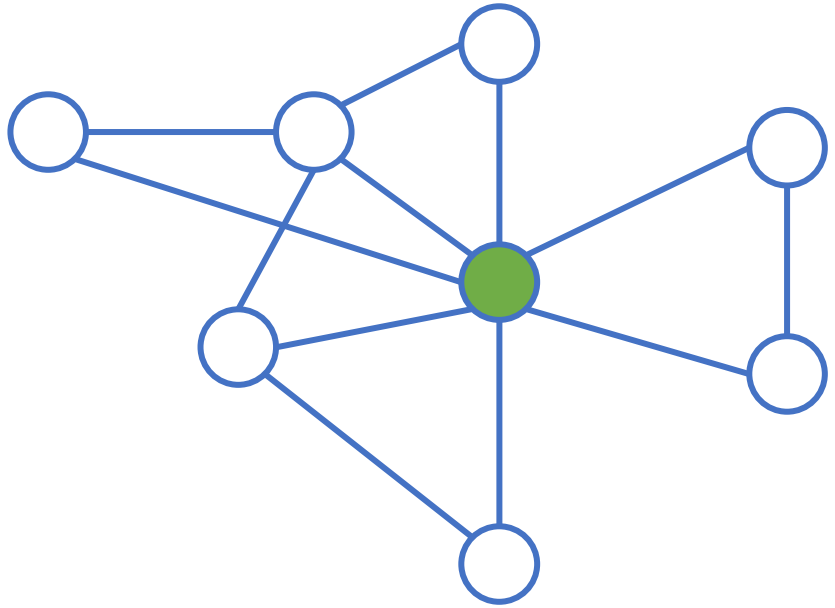
Input



Graph G=(V,E)
Feature vector $x_u \in \mathbb{R}^d$ for all $u \in V$

$h_u = f(x_u)$

Node embeddings



Vector embedding $h_u \in \mathbb{R}^d$

Key ideas

Special class of graphs: image grids

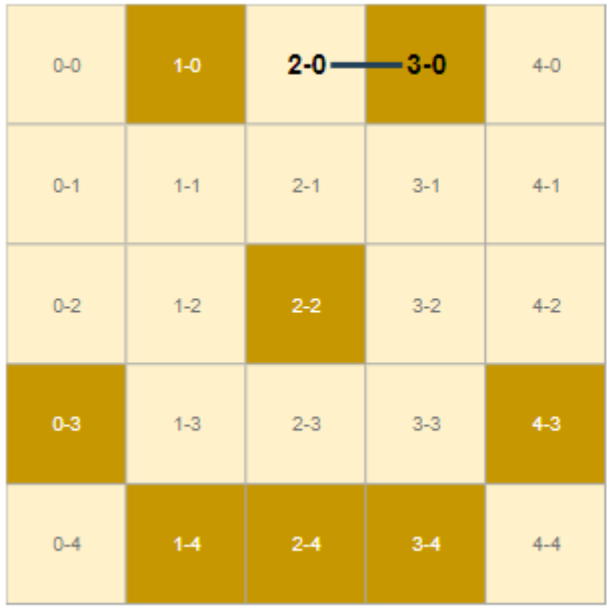
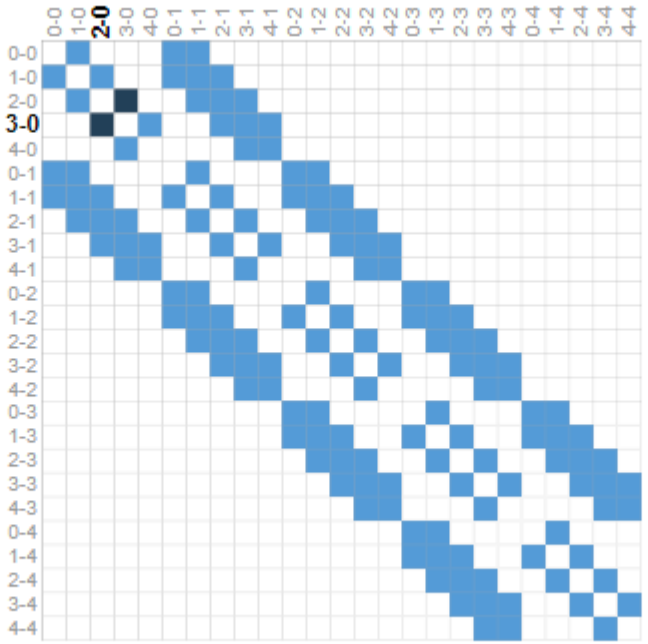
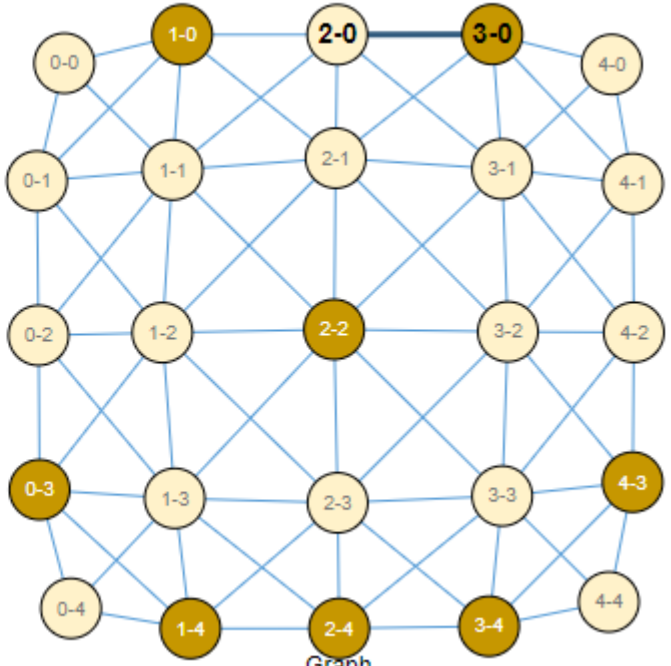


Image Pixels



Adjacency Matrix

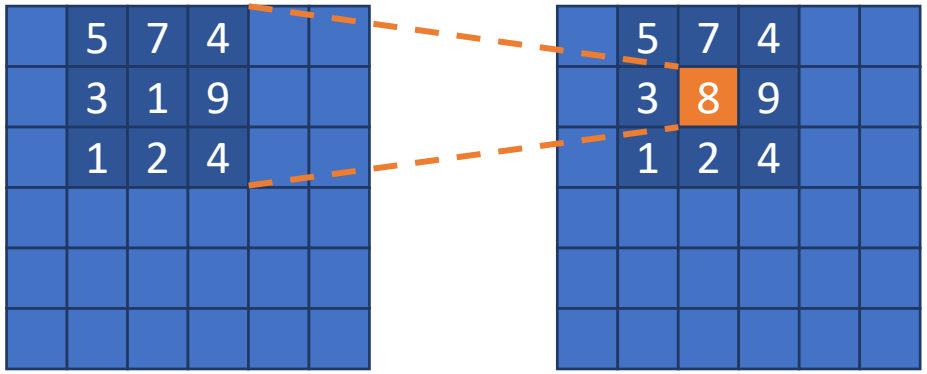


Graph

Key ideas

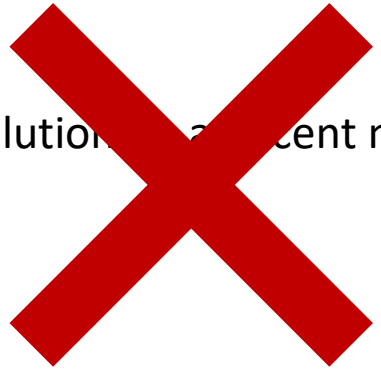
Images

Convolution on grid



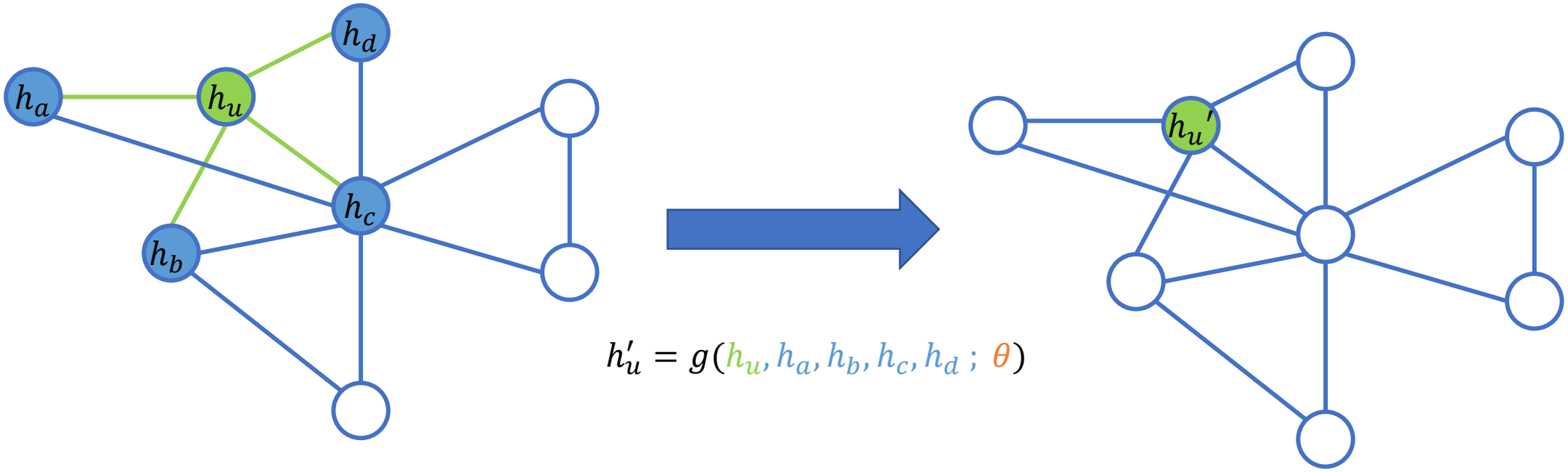
Graphs

Convolution[illegible] adjacent matrix ?



Sensitive to node reordering
& graph size

Key ideas



$$h'_u = g(h_u, h_a, h_b, h_c, h_d\ ;\ \theta)$$

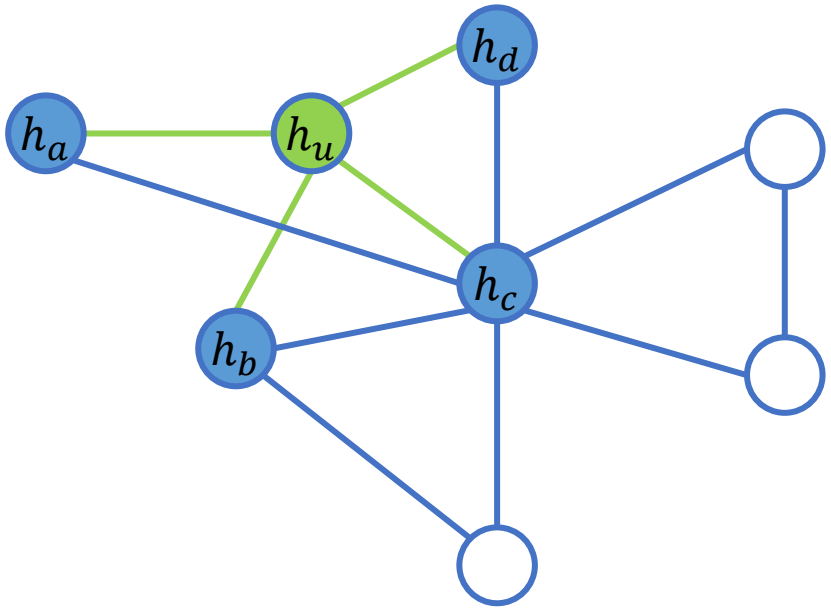
Key ideas

Desirable properties of g:

- Fixed number of parameters
- Leverage local information
- Permutation invariant

Message Passing Neural Network (MPNN)

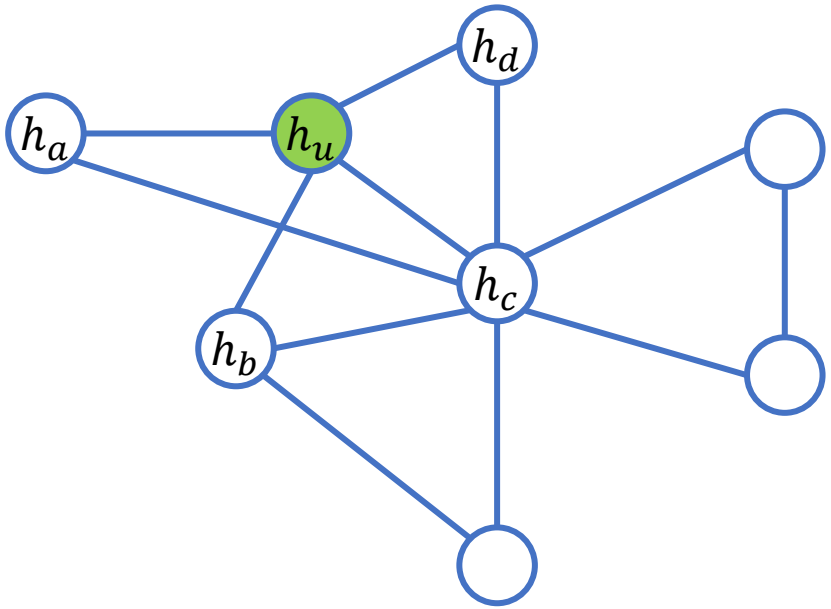
Message Passing Neural Network (MPNN)



$$m_{N(u)}^{(k)} = AGGREGATE^{(k)}\left(\left\{h_v^{(k)}, \forall\, v \in N(u)\right\}\right)$$

$h_u^{(k)} = embedding\ of\ u\ at\ k^{th} - iteration$

Message Passing Neural Network (MPNN)

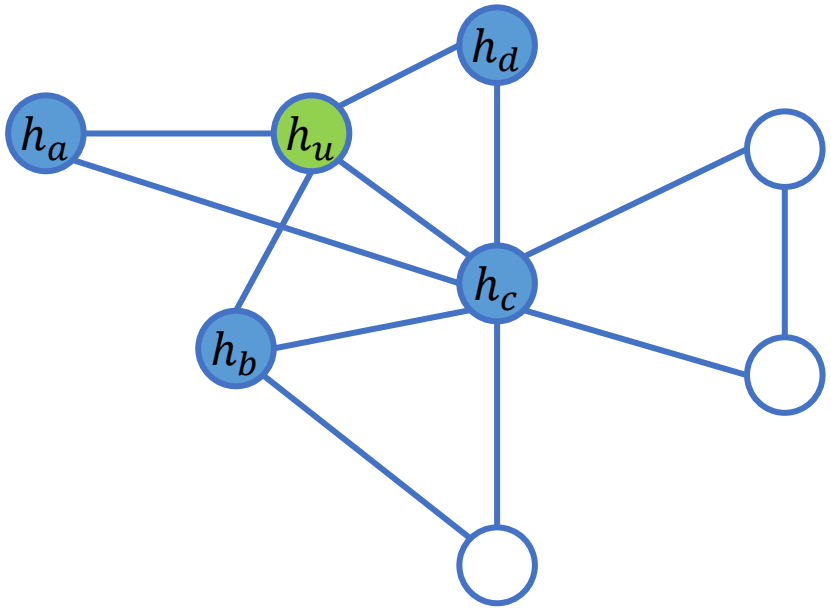


$$m_{N(u)}^{(k)} = AGGREGATE^{(k)}\left(\left\{h_v^{(k)}, \forall\, v \in N(u)\right\}\right)$$

$$h_u^{(k+1)} \leftarrow UPDATE^{(k)}\left(h_u^{(k)}, m_{N(u)}^{(k)}\right)$$

$h_u^{(k)} = embedding\ of\ u\ at\ k^{th} - iteration$

Basic GNN



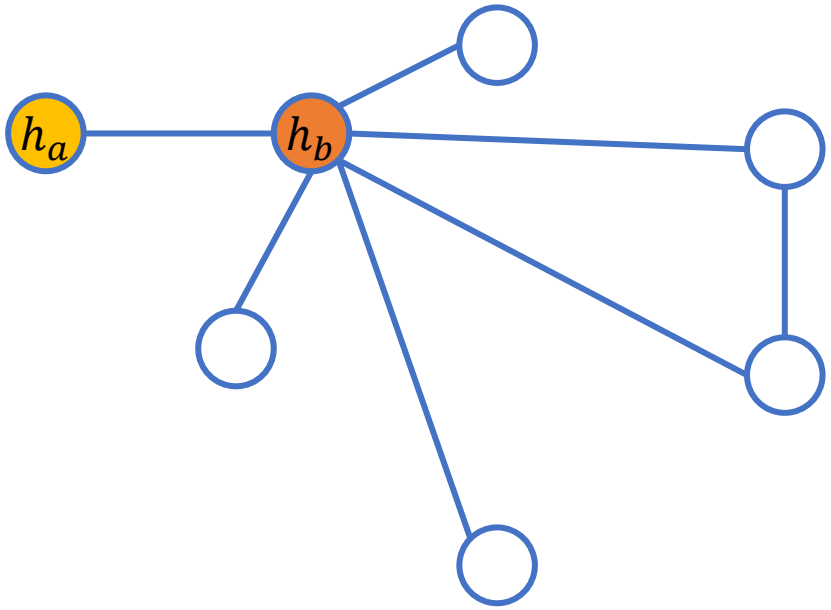
$$m_{N(u)} = AGGREGATE(\{h_v, \forall v \in N(u)\}) = \sum_{v \in N(u)} h_v$$

$$h_u \leftarrow UPDATE(h_u, m_{N(u)}) = \sigma(W_{self} h_u + W_{neigh} m_{N(u)})$$

$$h_u \leftarrow \sigma\left(W_{self}\, h_u + W_{neigh}(h_a + h_b + h_c + h_d)\right)$$

$$\theta = \{W_{self}, W_{neigh}\}$$

Neighborhood normalization

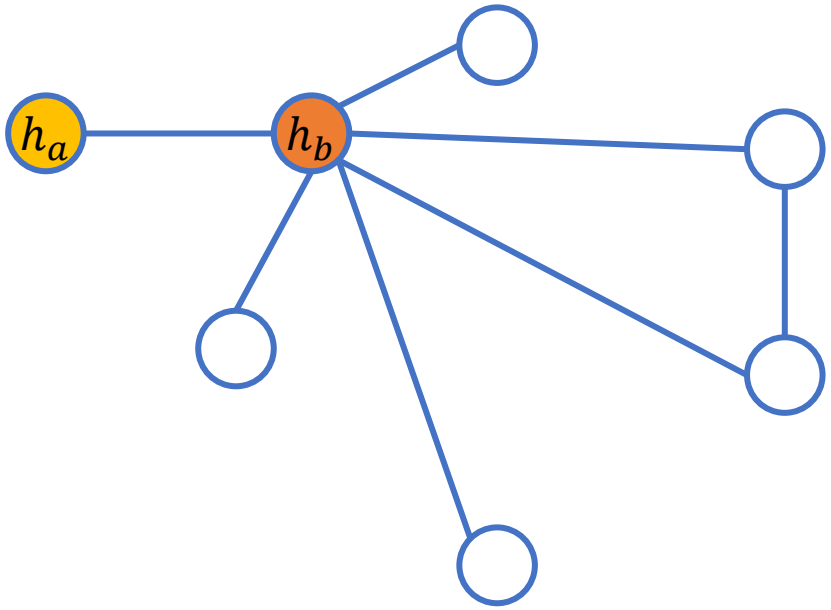


Assume

$$\left\| \sum_{v \in N(b)} h_v \right\| \gg \left\| \sum_{v \in N(a)} h_v \right\|$$

Numerical instabilities & difficulties for optimization

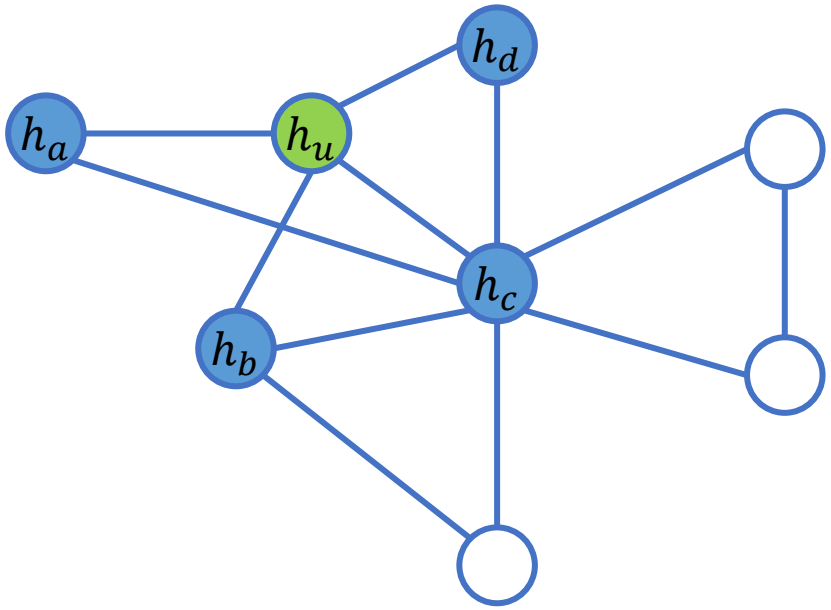
Neighborhood normalization



$$m_{N(u)} = \frac{1}{|N(u)|} \sum_{v \in N(u)} h_v$$

Downsides:
loss of information about structural info

Graph Convolution Network (GCN)



$$m_{N(u)} = AGGREGATE\left(\left\{h_v \ , \forall\, v \in N(u) \cup \{u\}\right\}\right)$$

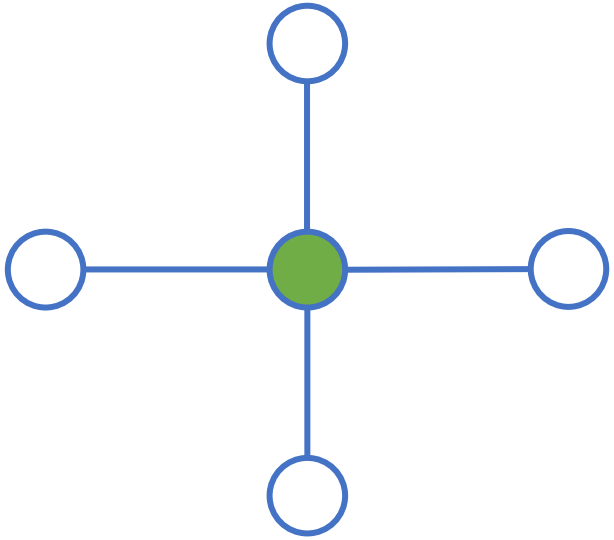
$$= \sum_{v \in N(u) \cup \{u\}} \frac{h_v}{\sqrt{|N(u)|\,|N(v)|}}$$

$$h_u \leftarrow UPDATE\left(h_u\,, m_{N(u)}\right) = \sigma\left(W m_{N(u)}\right)$$

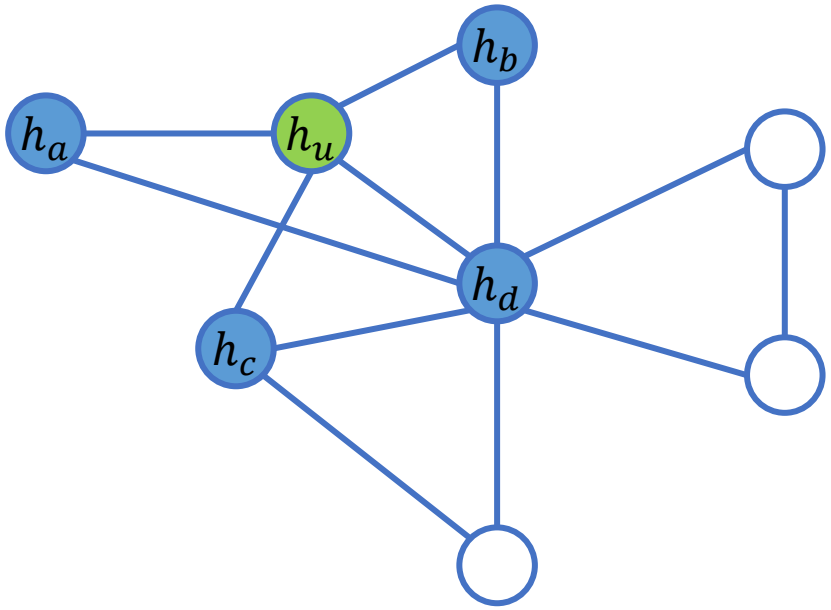
[T. N. Kipf et al, 2017]

MPNN – limitations

Sum aggregate can distinguish,
mean cannot



Graph Isomorphism Network (GIN)



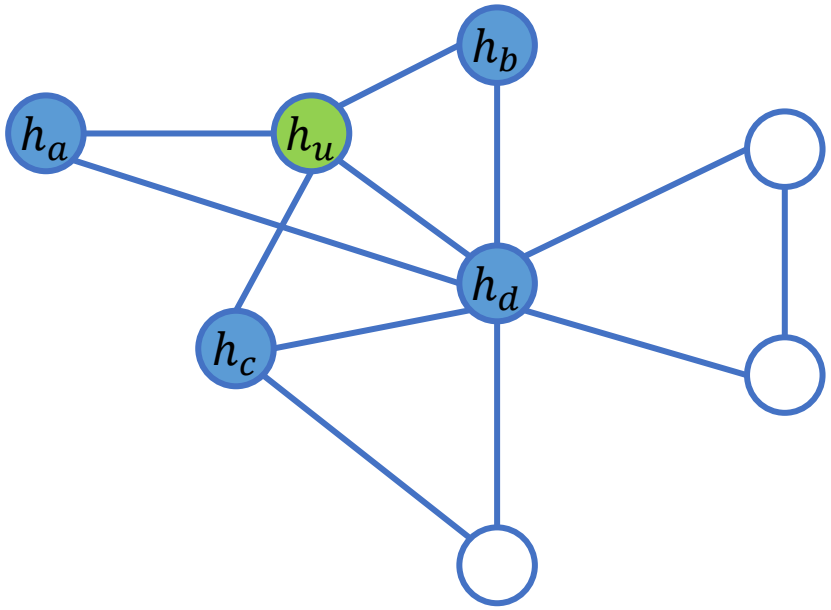
$$h_u \leftarrow UPDATE\left(h_u, AGGREGATE\left(\{h_v, \forall v \in N(u)\}\right)\right)$$

How to learn any set function?

$$g(X) = \varphi\left(\sum_{x \in X} f(x)\right), \qquad s.t. \sum_{x \in X} f(x) \ unique$$

[K. Xu et al, 2019]

Graph Isomorphism Network (GIN)

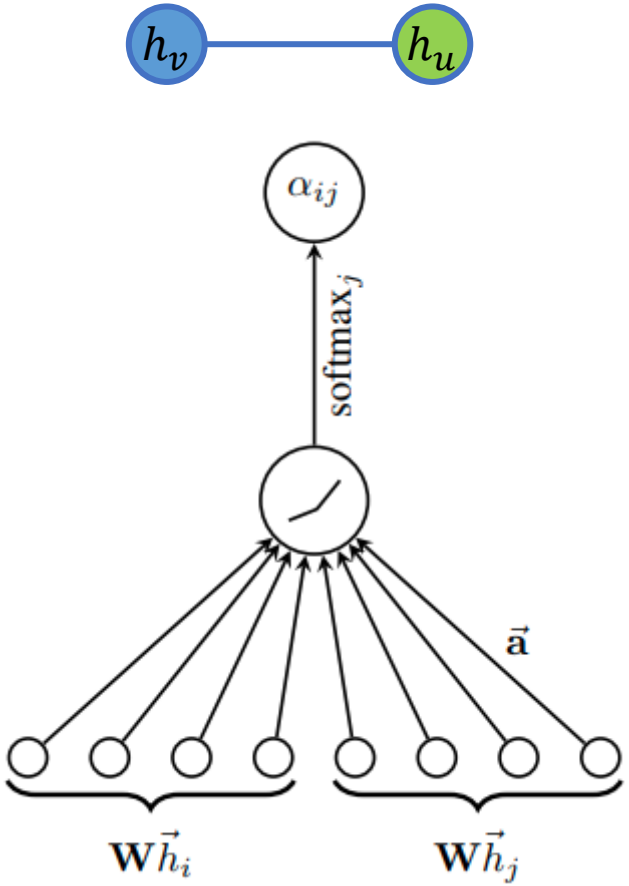


$$h_u^{(k)} = MLP^{(k)}\left((1+\epsilon)h_u^{(k-1)} + \sum_{v \in N(u)} h_v^{(k-1)}\right)$$

MLP=multi-layer perceptron NN

[K. Xu et al, 2019]

Graph Attention Network (GAT)



[P. Veličković, 2018]

How relevant a neighboring node is in relation to the center node?

$$m_{N(u)} = AGGREGATE\left(\{h_v \ , \forall\, v \in N(u)\}\right) = \sum_{v \in N(u)} \alpha_{u,v} h_v$$

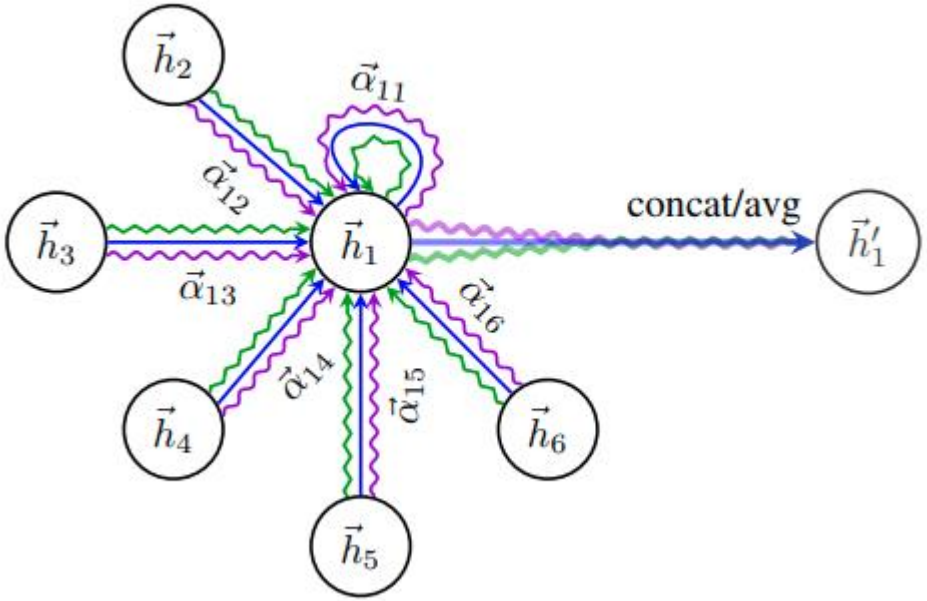
Bilinear attention model

$$\alpha_{u,v} = \frac{\exp(a^T[Wh_u \oplus Wh_v])}{\sum_{v' \in N(u)} \exp(a^T[Wh_u \oplus Wh_{v'}])}$$

Attention via MLP

$$\alpha_{u,v} = \frac{\exp(MLP(h_u, h_v\ ;\ \theta))}{\sum_{v' \in N(u)} \exp(MLP(h_u, h_{v'};\ \theta))}$$

Graph Attention Network (GAT)



How relevant a neighboring node is in relation to the center node?

Multiple attention "heads"

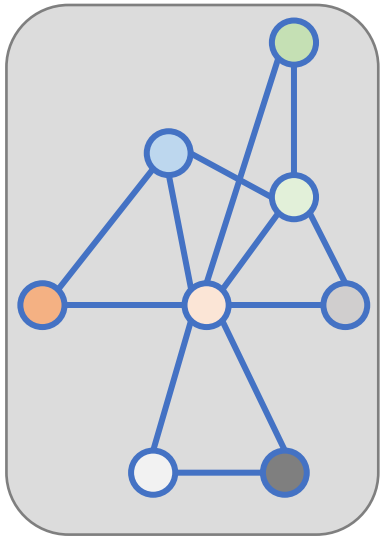
$$m_{N(u)} = AGGREGATE(\{h_v \ , \forall\, v \in N(u)\}) = [a_1 \oplus a_2 \oplus \cdots \oplus a_K]$$

$$a_k = W_i \sum_{v \in N(u)} \alpha_{u,v,k} h_v$$

GAT - results

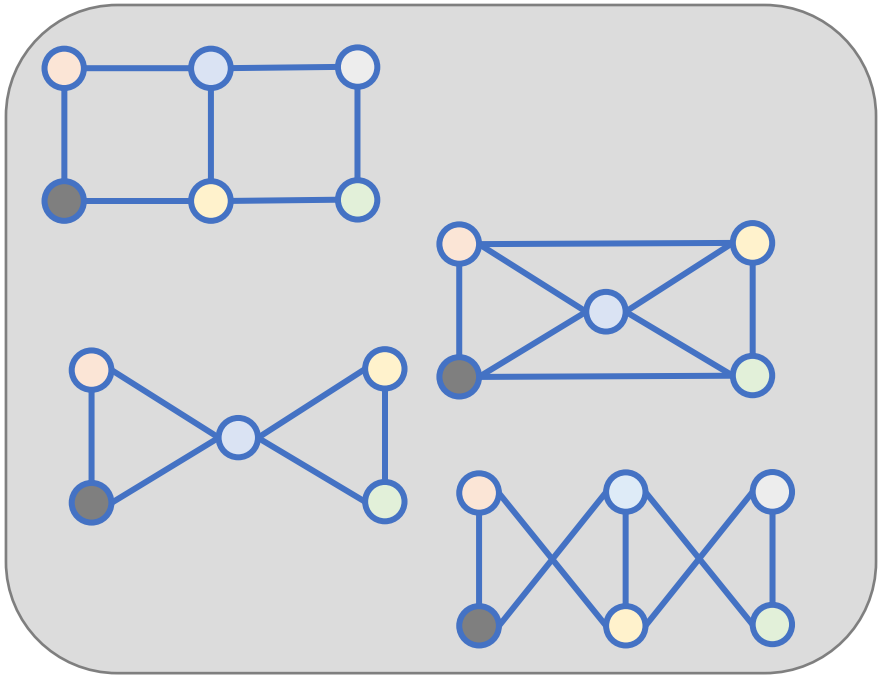
Transductive setting

1 graph, some labels are masked



Inductive setting

Multiple graphs, some for training, some for test



GAT - results

	Cora	Citeseer	Pubmed	PPI
Task	Transductive	Transductive	Transductive	Inductive
# Nodes	2708 (1 graph)	3327 (1 graph)	19717 (1 graph)	56944 (24 graphs)
# Edges	5429	4732	44338	818716
# Features/Node	1433	3703	500	50
# Classes	7	6	3	121 (multilabel)
# Training Nodes	140	120	60	44906 (20 graphs)
# Validation Nodes	500	500	500	6514 (2 graphs)
# Test Nodes	1000	1000	1000	5524 (2 graphs)

old

Transductive

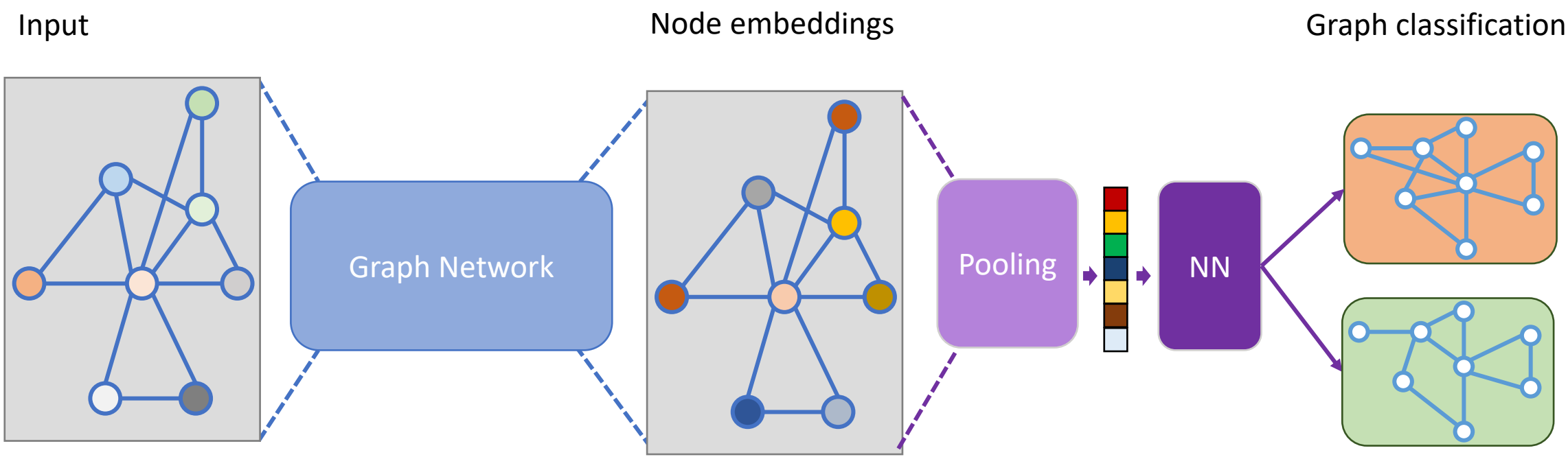
Method	**Cora**	**Citeseer**	**Pubmed**
MLP	55.1%	46.5%	71.4%
ManiReg (Belkin et al., 2006)	59.5%	60.1%	70.7%
SemiEmb (Weston et al., 2012)	59.0%	59.6%	71.7%
LP (Zhu et al., 2003)	68.0%	45.3%	63.0%
DeepWalk (Perozzi et al., 2014)	67.2%	43.2%	65.3%
ICA (Lu & Getoor, 2003)	75.1%	69.1%	73.9%
Planetoid (Yang et al., 2016)	75.7%	64.7%	77.2%
Chebyshev (Defferrard et al., 2016)	81.2%	69.8%	74.4%
GCN (Kipf & Welling, 2017)	81.5%	70.3%	**79.0%**
MoNet (Monti et al., 2016)	81.7 ± 0.5%	—	78.8 ± 0.3%
GCN-64*	81.4 ± 0.5%	70.9 ± 0.5%	**79.0** ± 0.3%
GAT (ours)	**83.0** ± 0.7%	**72.5** ± 0.7%	**79.0** ± 0.3%

old

Inductive

Method	**PPI**
Random	0.396
MLP	0.422
GraphSAGE-GCN (Hamilton et al., 2017)	0.500
GraphSAGE-mean (Hamilton et al., 2017)	0.598
GraphSAGE-LSTM (Hamilton et al., 2017)	0.612
GraphSAGE-pool (Hamilton et al., 2017)	0.600
GraphSAGE*	0.768
Const-GAT (ours)	0.934 ± 0.006
GAT (ours)	**0.973** ± 0.002

Graph pooling

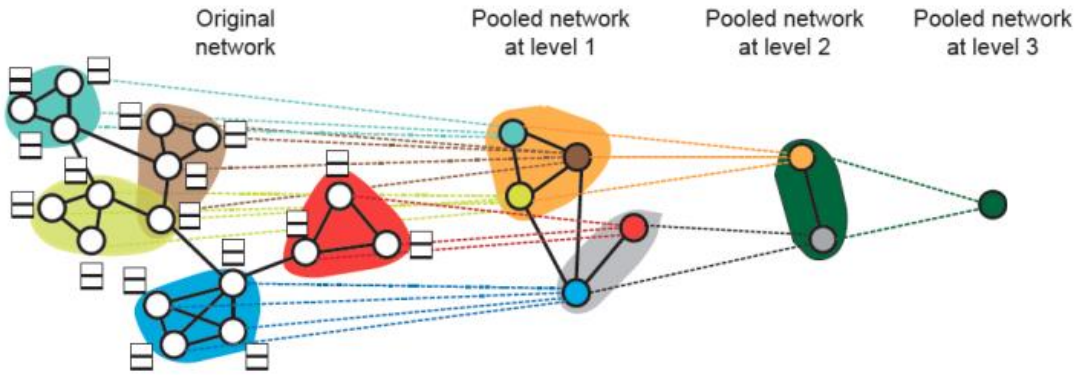


Graph pooling

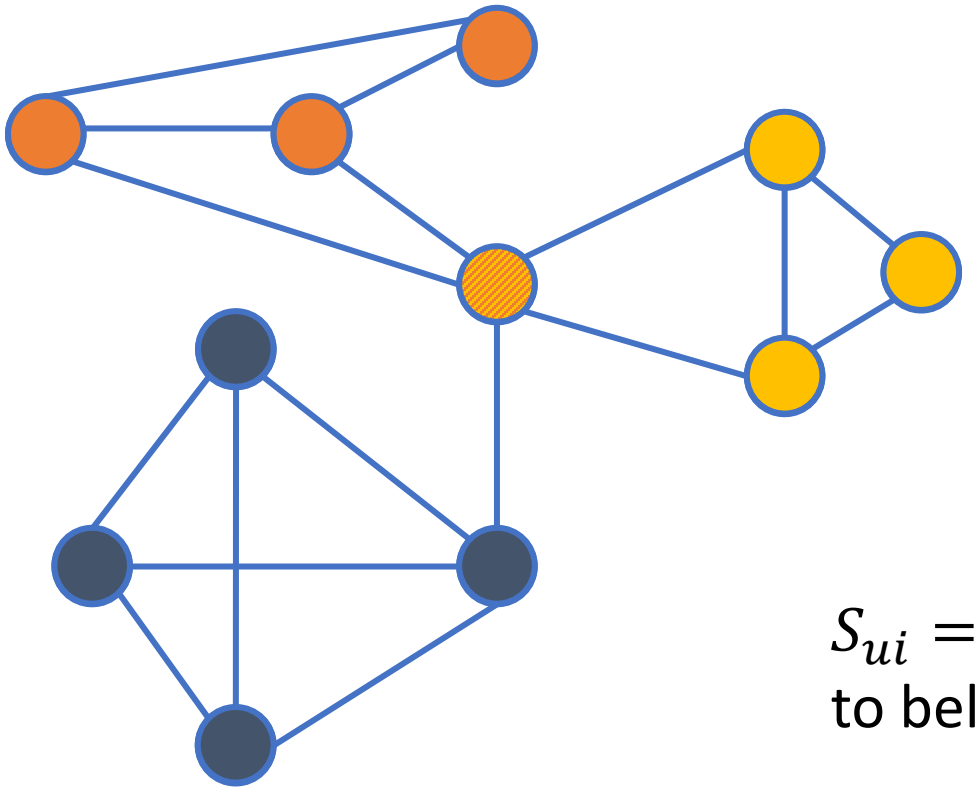
Set pooling approaches

- $z_G = \frac{\sum_{v \in V} z_v}{|V|}$
- $z_G = \sum_{v \in V} z_v$

Graph coarsening approaches

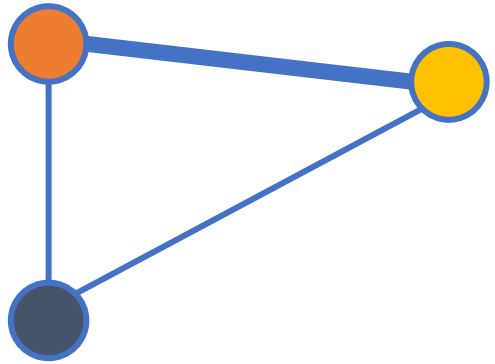


Graph coarsening



$G: A, X$

S_{ui} = prob. of node u to belong to cluster i

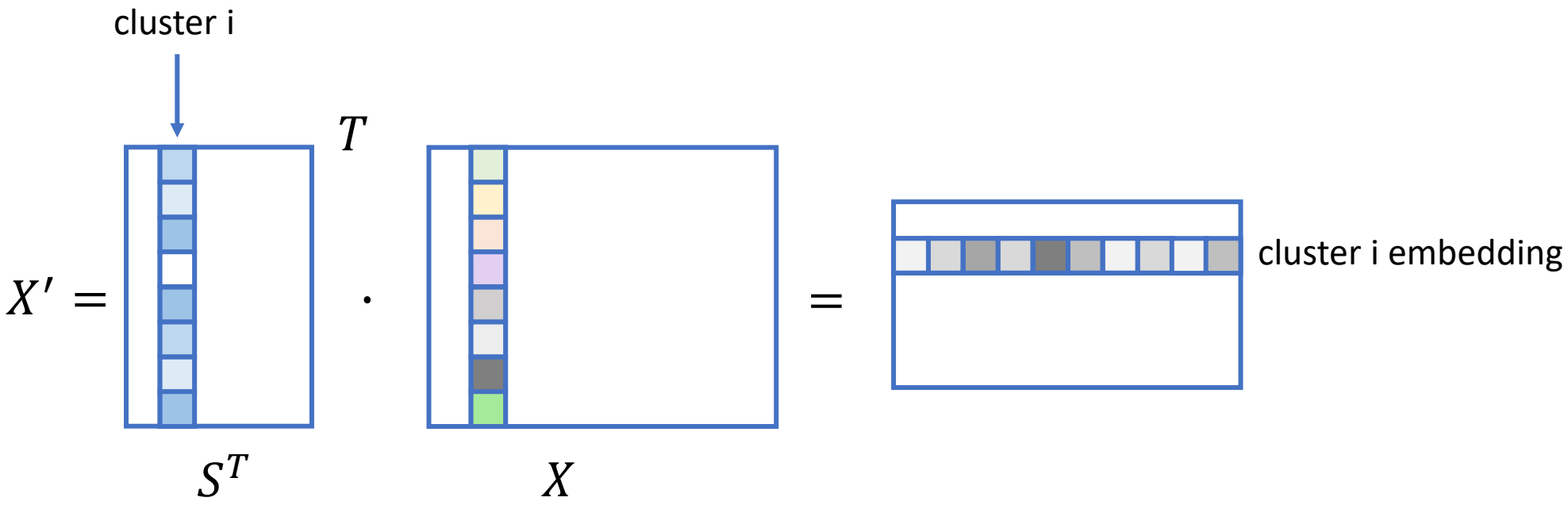


$G': A', X'$

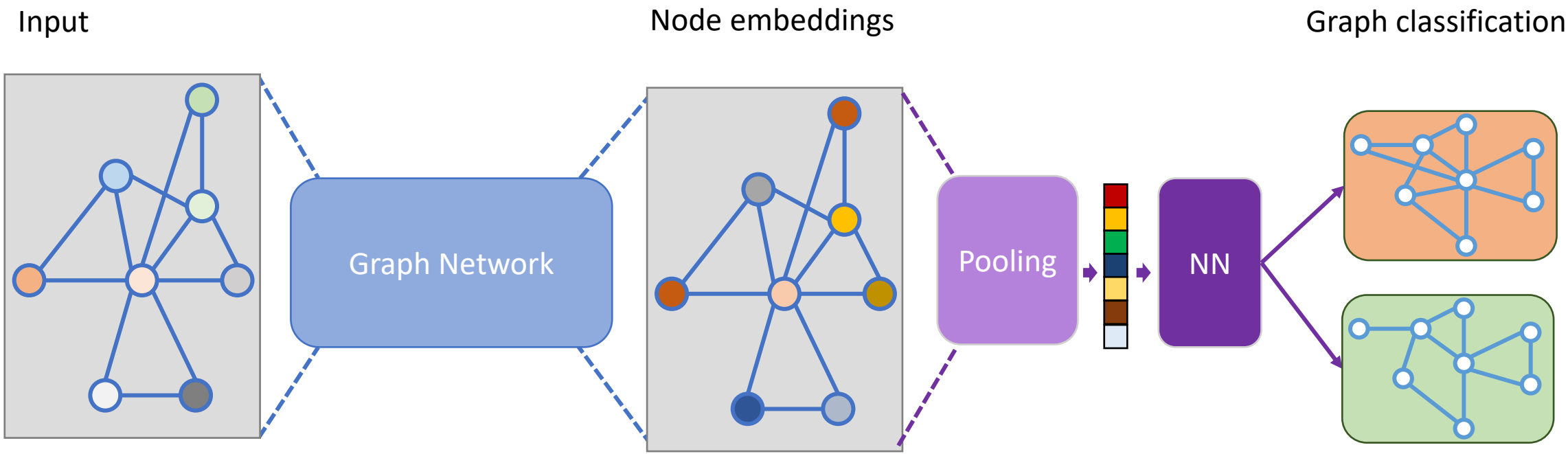
$A' = S^T AS \in \mathbb{R}^{c \times c}$

$X' = S^T X \in \mathbb{R}^{c \times d}$

Graph coarsening

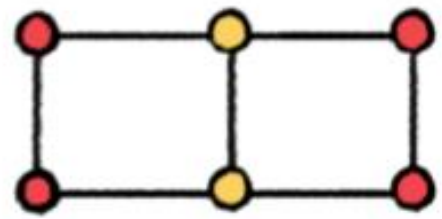


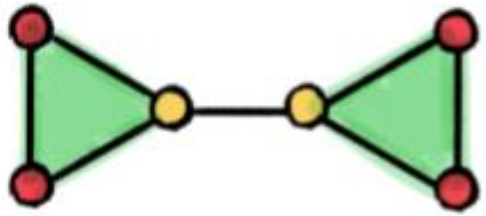
Graph pooling

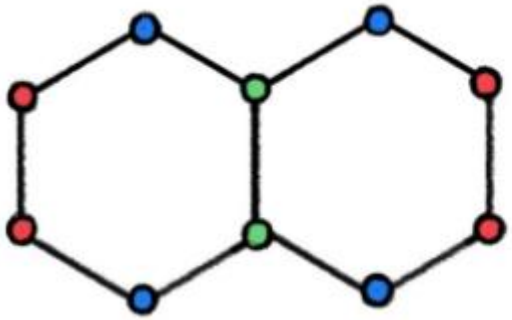


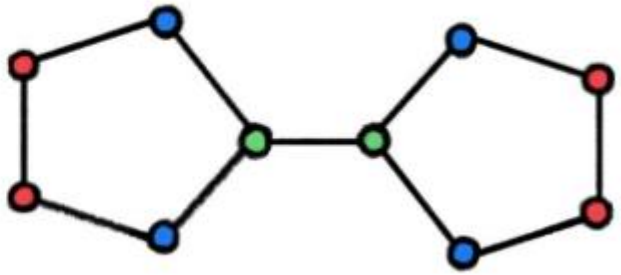
Outlook: problems

Problems – Graph isomorphism (WL-test)

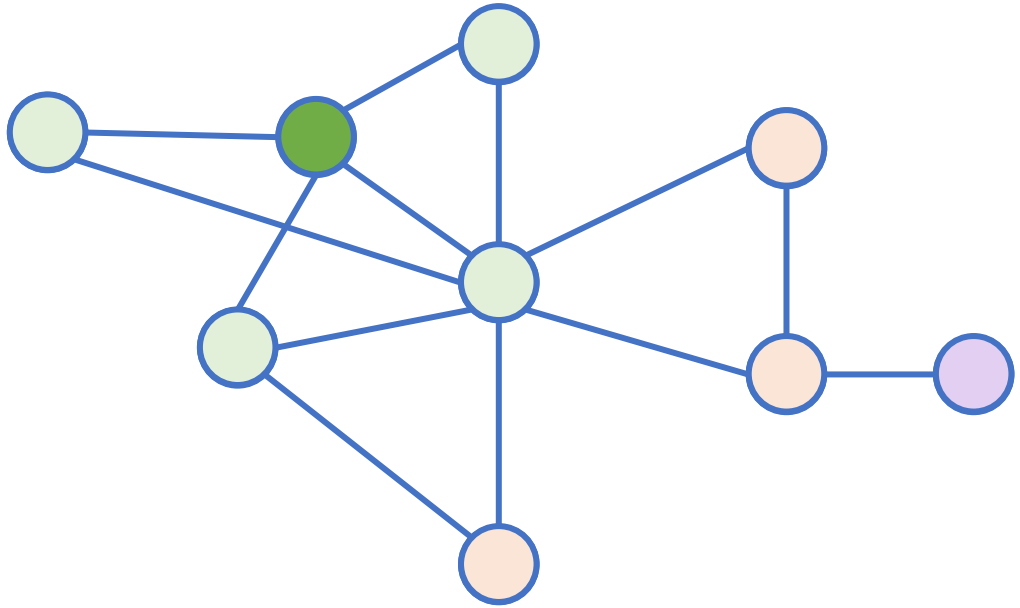








Problems – Underreaching



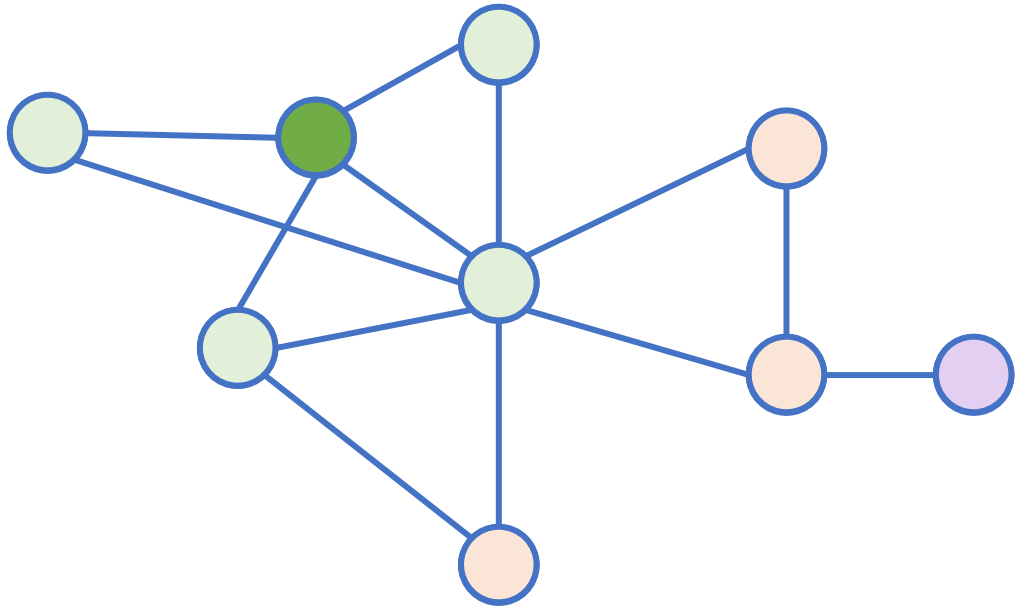
1-hop nodes

2-hop nodes

3-hop nodes

Problems – Underreaching

Issue

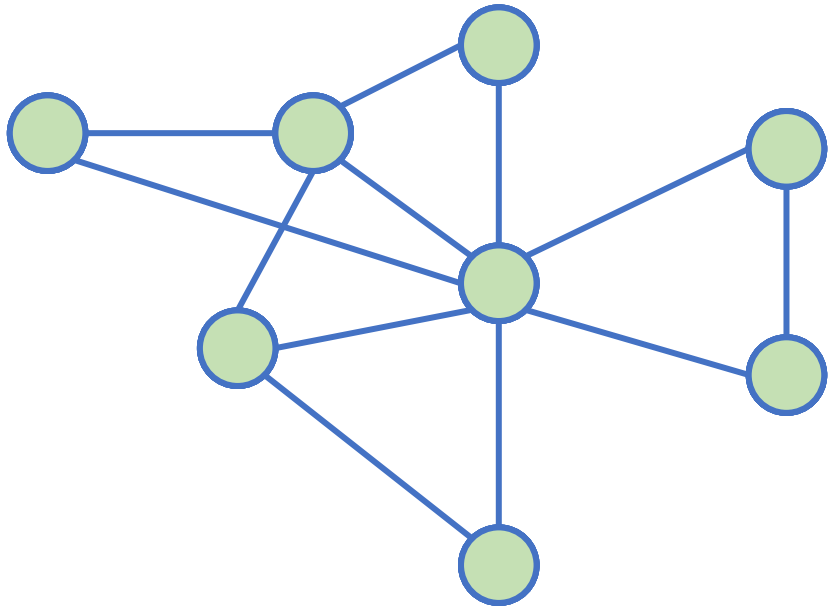


Possible solution

- Aggregate further away
- Add edges
  - Changes topology
  - May lead to oversmoothing

Problems – Oversmoothing

Issue



Averaging → same embedding for all nodes

Possible solution

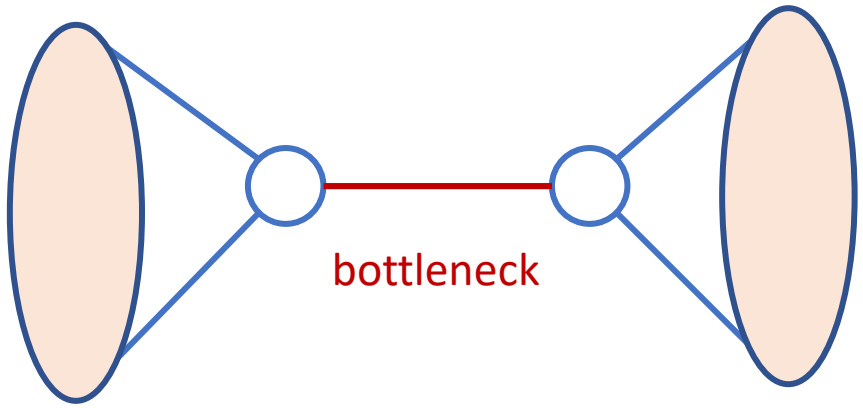
- Skip connections

$$m_{N(u)}^{(k)} = AGGREGATE^{(k)}\left(\left\{h_v^{(k)}, \forall\, v \in N(u)\right\}\right)$$

$$h_u^{(k+1)} \leftarrow \left[h_u^{(k)} \oplus UPDATE^{(k)}\left(h_u^{(k)}, m_{N(u)}^{(k)}\right)\right]$$

Problems – Oversquashing

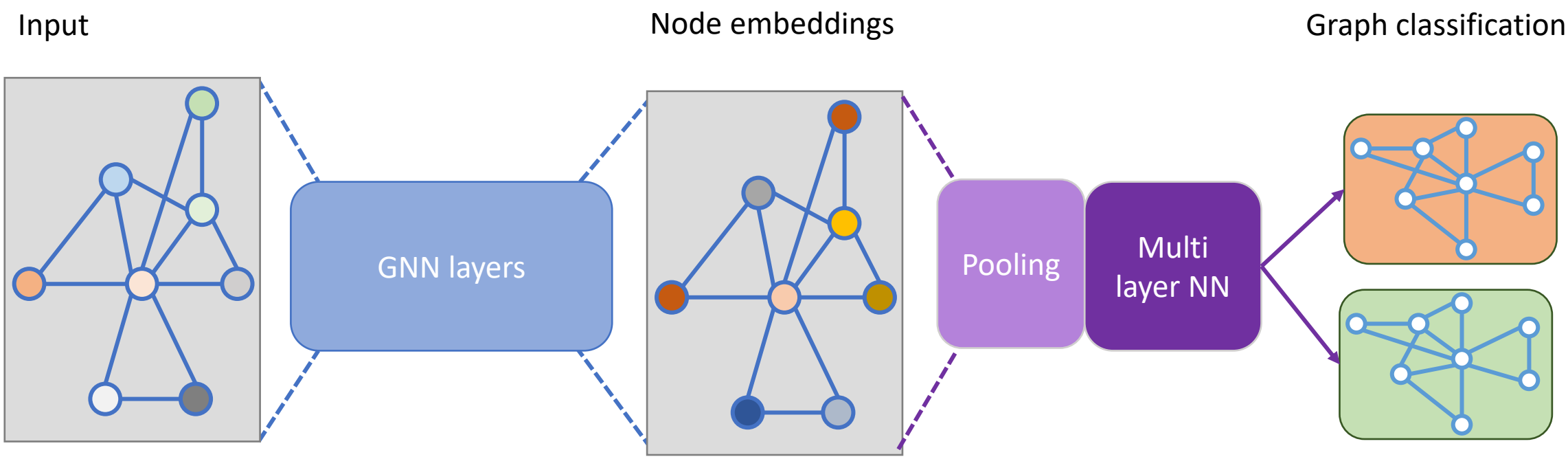
Issue



Possible solution

- Add edges
  - Changes topology
  - May lead to oversmoothing

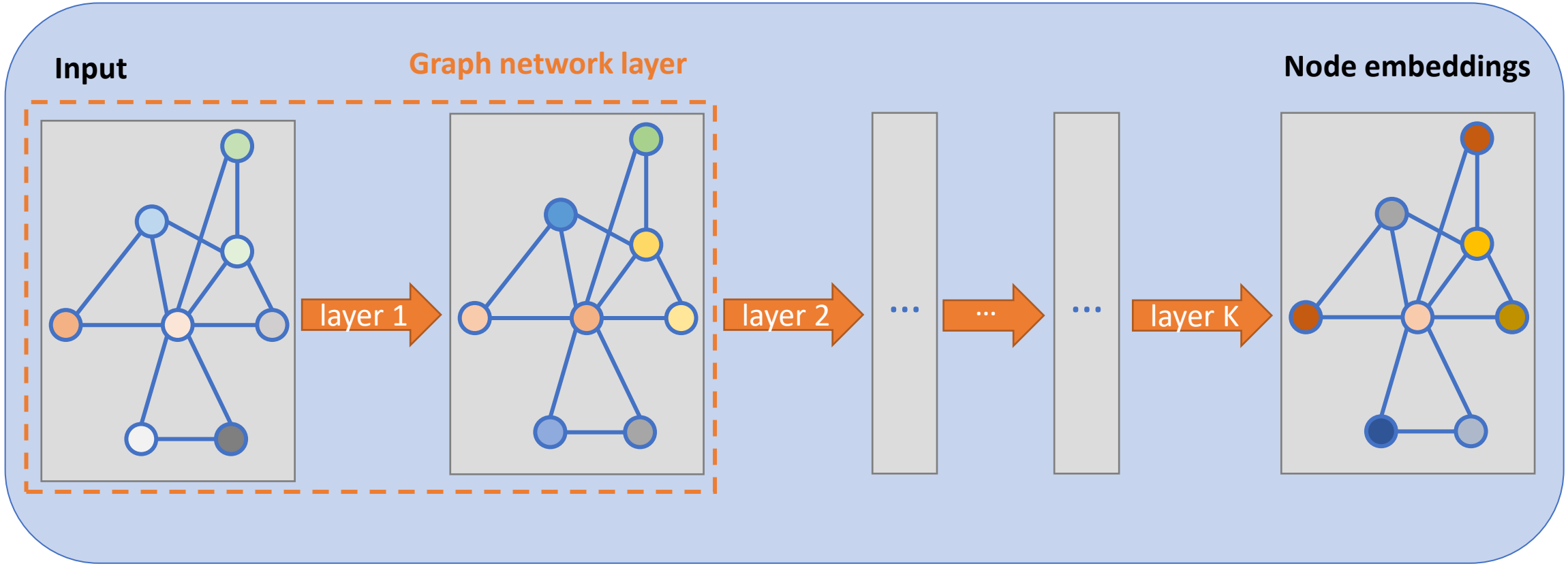
Summary



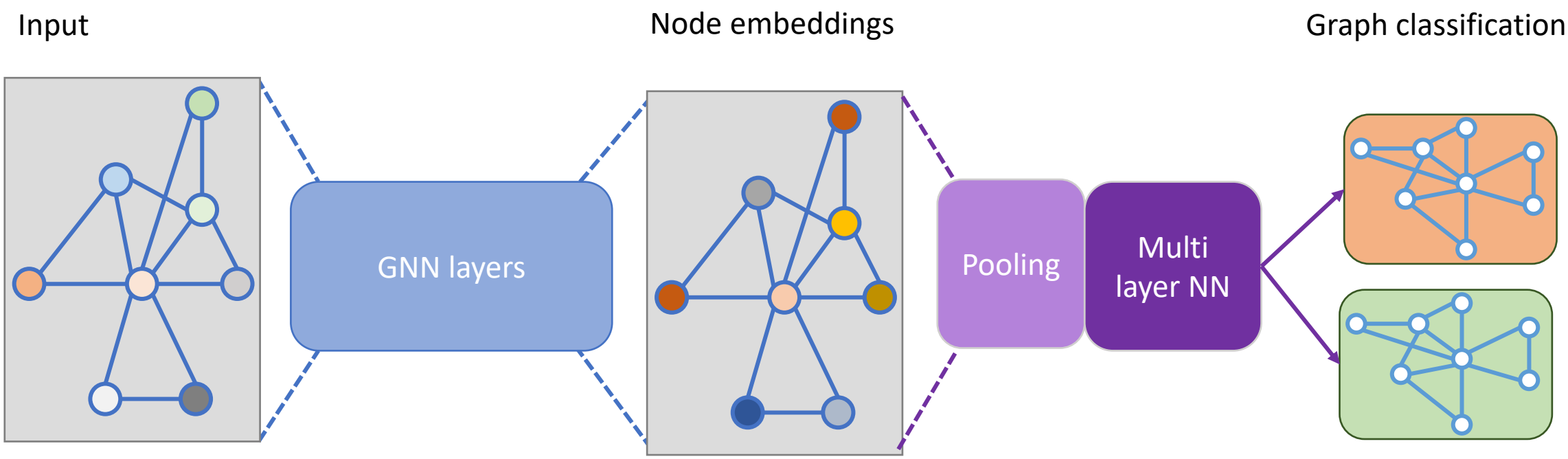
A GNN is an optimizable transformation on all attributes of the graph (nodes, edges)

Loss: cross-entropy

Summary


Input
Graph network layer
Node embeddings
layer 1
layer 2
...
...
...
layer K

Summary



A GNN is an optimizable transformation on all attributes of the graph (nodes, edges)

Loss: cross-entropy