Graph Attention Networks

Peter Veličković

Guillem Cucurull

Arantxa Casanova

Adriana Romero

Pietro Liò

Yoshua Bengio

Presented by: Yasmin Salehi

COMP 766 – Graph Representation Learning

Prof. William L. Hamilton

Introduction and Motivation

- Research on graph analysis
  - Node classification
  - Link prediction
  - Clustering
- Expressive power in representing non-Euclidean data
  - Examples: 3D meshes, social networks, telecommunication networks, brain connectomes, etc.

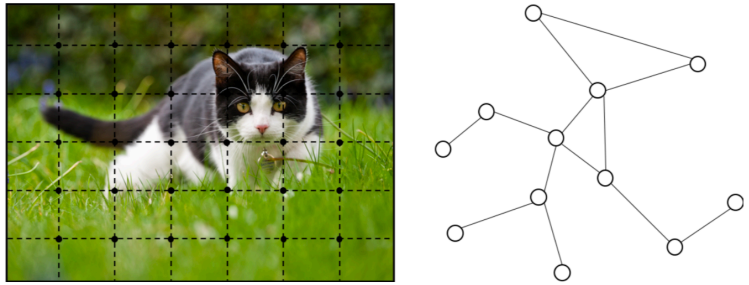
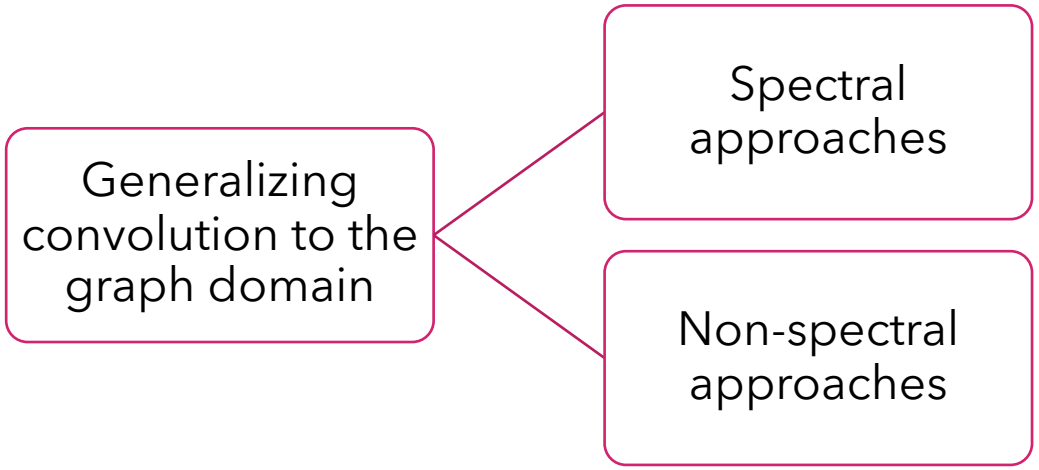


Fig. 1. Left: image in Euclidean space. Right: graph in non-Euclidean space [2]

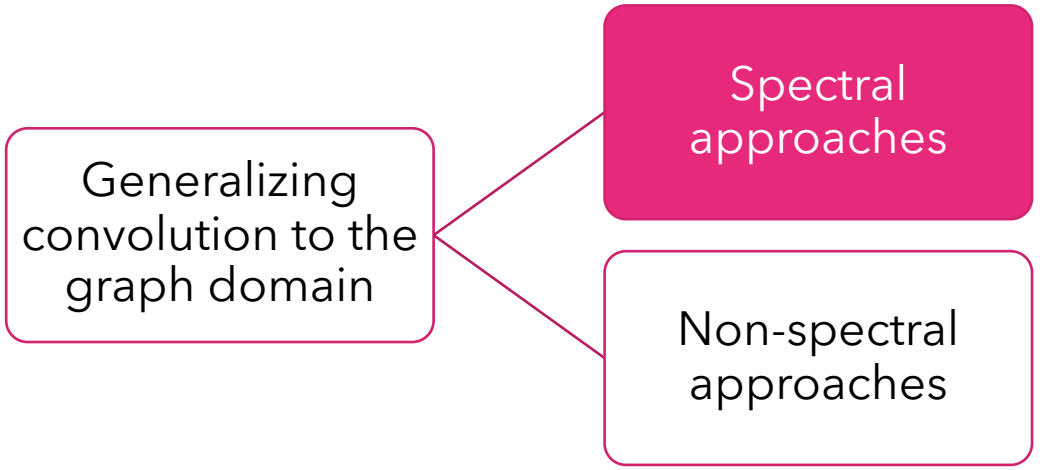
Graph Neural Networks (GNNs)

- Graph Neural Networks (GNNs): generalization of recursive neural networks
- Motivation:
  - Key characteristics of CNNs that apply to graph data:
    - Local connection
    - Shared weights
    - Multi-layer structures
  - Limitations of shallow embedding techniques
    - Not allowing parameter sharing → inefficient
    - Failing to leverage node attributes
    - Inherently transductive

Literature Review



Literature Review



- Signal: $\mathbf{x} \in \mathbb{R}^N$
- Filter: $\mathbf{g}_\theta = \mathrm{diag}(\theta) \quad \theta \in \mathbb{R}^N$
- Convolution operation:

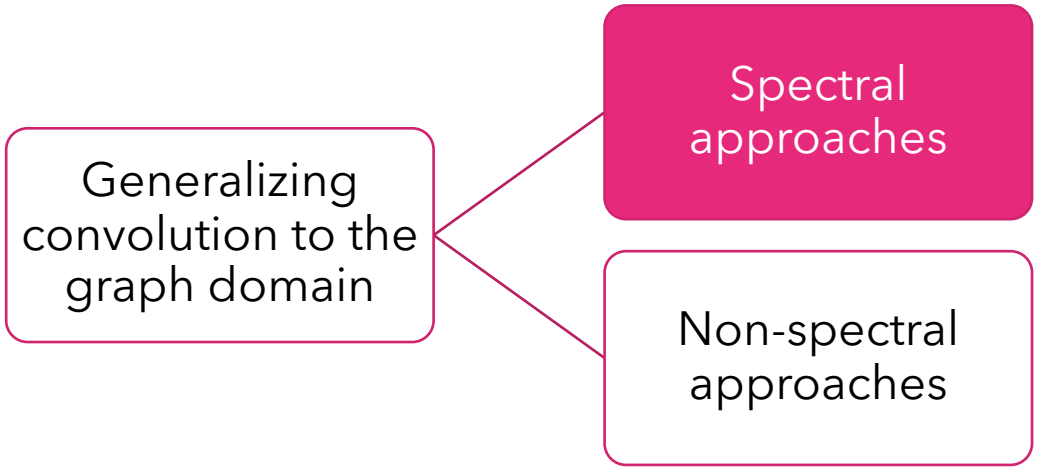
$$\mathbf{g}_\theta \star \mathbf{x} = \mathbf{U}\mathbf{g}_\theta(\mathbf{\Lambda})\mathbf{U}^T\mathbf{x}$$

- **U**: matrix of eigenvectors of the normalized **graph Laplacian**

$$\mathbf{L} = \mathbf{I}_N - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$$

- **A**: adjacency matrix, **D**: degree matrix, $\boldsymbol{\Lambda}$: diagonal matrix holding the eigenvalues

Literature Review



- Signal: $\mathbf{x} \in \mathbb{R}^N$
- Filter: $\mathbf{g}_\theta = \mathrm{diag}(\theta) \quad \theta \in \mathbb{R}^N$
- Convolution operation:

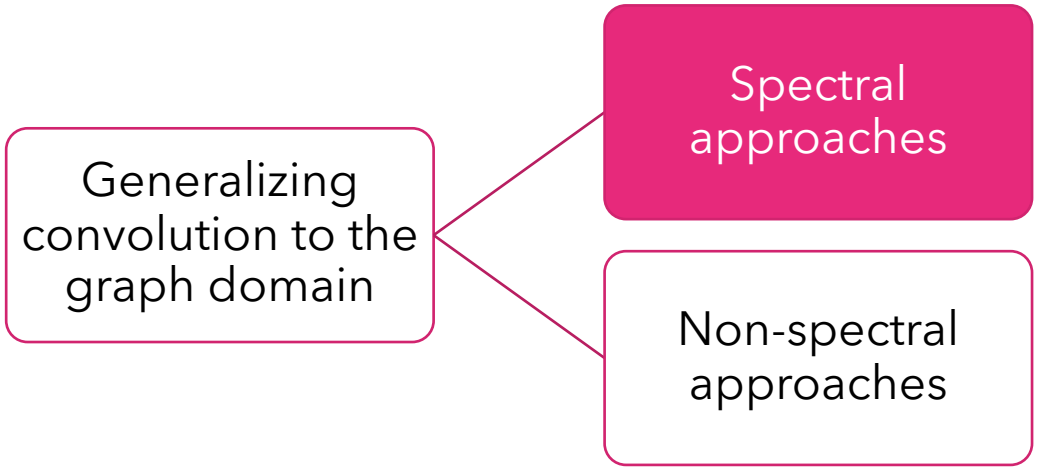
$$\mathbf{g}_\theta \star \mathbf{x} = \mathbf{U}\mathbf{g}_\theta(\mathbf{\Lambda})\mathbf{U}^T\mathbf{x}$$

- **U**: matrix of eigenvectors of the normalized **graph Laplacian**

$$\mathbf{L} = \mathbf{I}_N - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$$

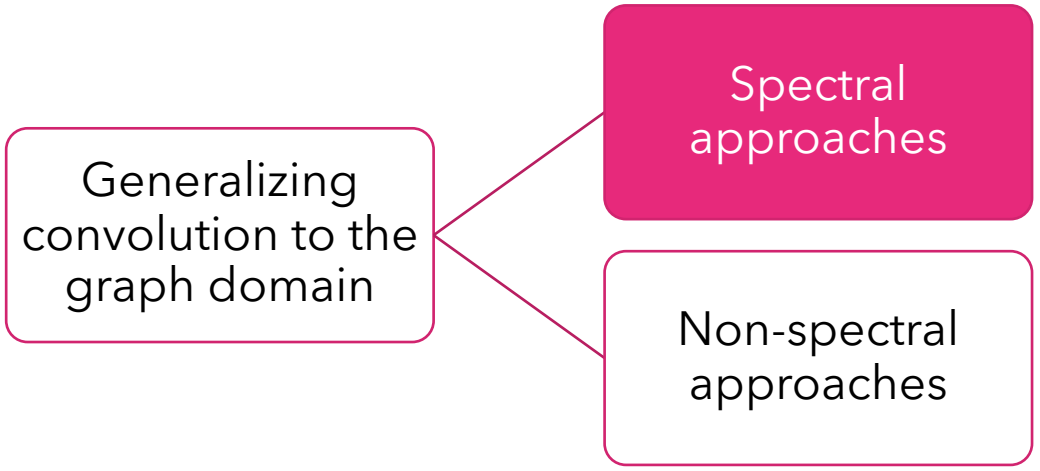
- **Results in**: intense computations and non-spatially localized filters

Literature Review



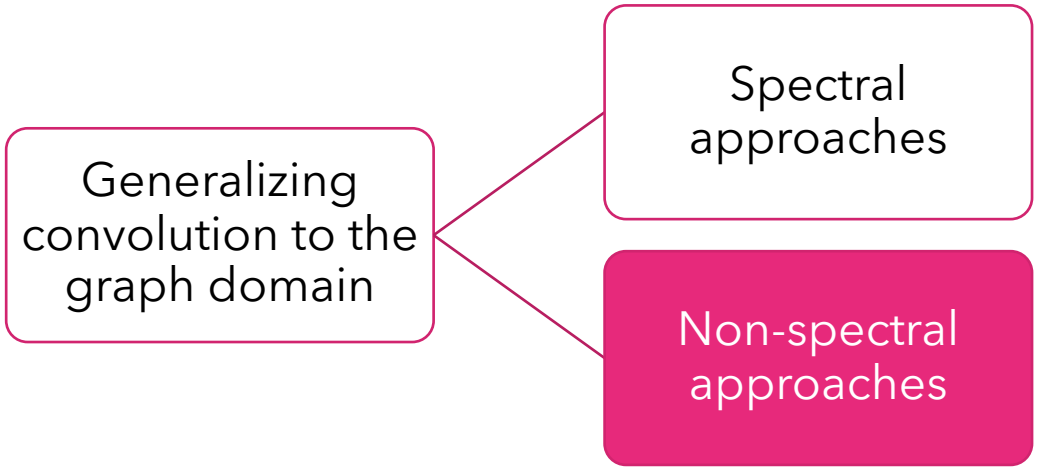
- **Solutions**:
- Introducing a parameterization of the spectral filters with smooth coefficients
- Approximate the filters by means of a Chebyshev expansion of the graph Laplacian
- Simplifying the previous method by restricting the filters to operate in a 1-step neighborhood around each node

Literature Review



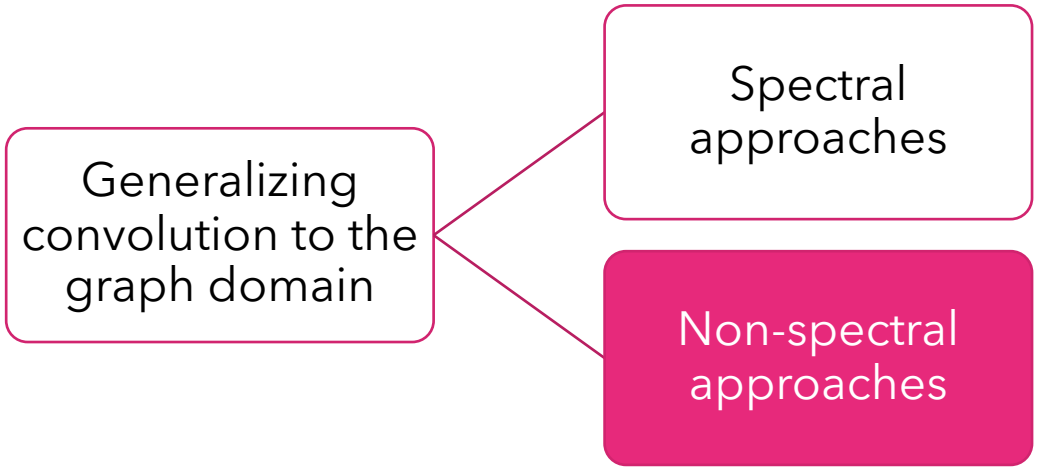
- **Limitations**:
- Learnt filters depend on the Laplacian eigenbasis
  - Depends on the graph structure
  - Cannot be generalized to new graphs

Literature Review



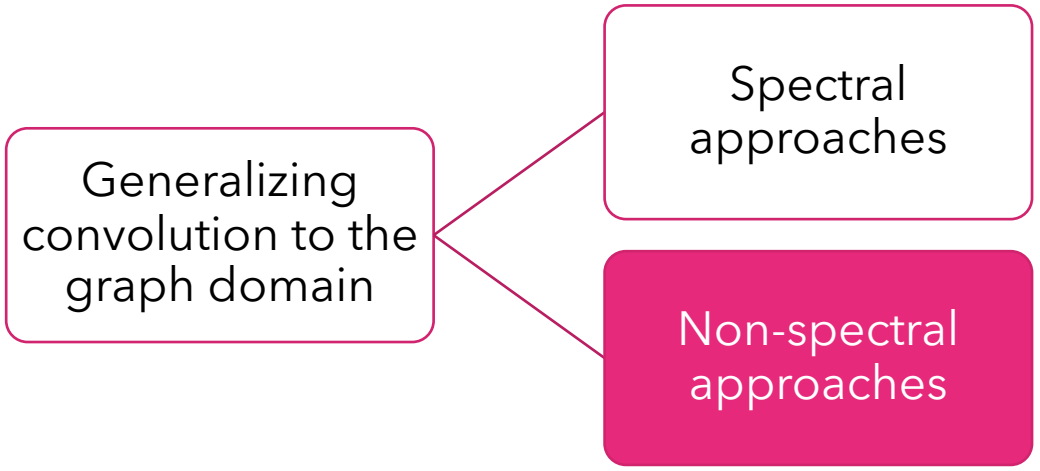
- Convolution is directly defined on the graph.
- Challenge:
  - Operator that works with different sized neighborhoods and maintains the weight-sharing property of CNNs

Literature Review



- A few of the most recent approaches:
  - MoNet: provides unified generalization of CNN architectures to graphs
  - GraphSAGE: has yielded impressive performance over inductive benchmarks

Literature Review



- **Limitations**:
  - MoNet: uses node's structural properties for similarity computation
    - requires knowing the graph structure upfront
  - GraphSAGE: samples a fixed-size neighborhood of each node
    - Prevents it from using the entirety of its neighborhood

Graph Attention Networks (GATs)

- GATs: novel neural network architecture that operates on graph structured data
- Use masked self-attentional layers
  - Computationally efficient $O(|V|FF' + |E|F')$
  - Allows assigning different importances to different nodes
  - Does not require the global graph structure upfront
    - Does not require undirected graphs
    - Enables inductive learning
  - Allows inputs to have variable sizes
  - Works with the entirety of nodes' neighborhoods

Graph Attention Networks (GATs)

- Building block: Graph attentional layer

Input: a set of node features

$$\mathbf{h} = \{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_N\}, \vec{h}_i \in \mathbb{R}^F$$

$$\vec{h}'_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W} \vec{h}_j\right)$$

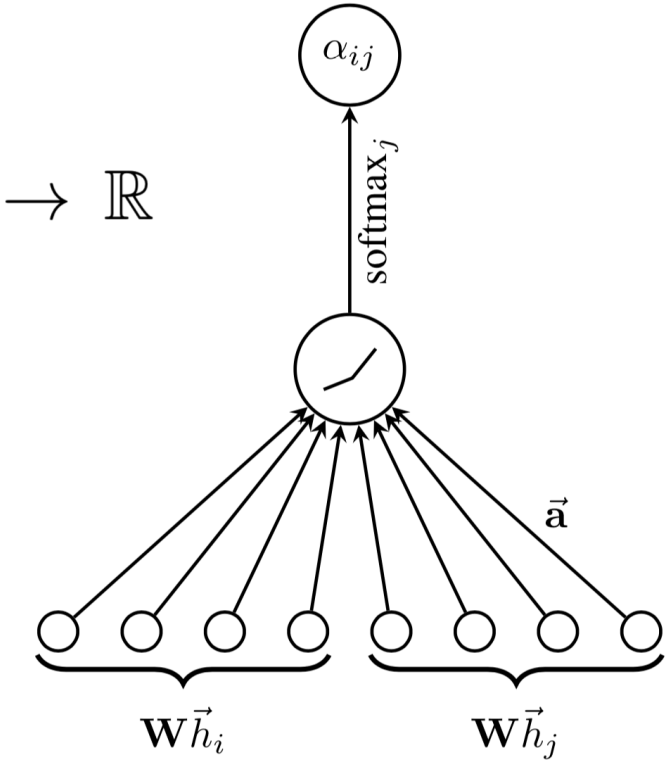
Output: a new set of node features

$$\mathbf{h}' = \{\vec{h}'_1, \vec{h}'_2, \ldots, \vec{h}'_N\}, \vec{h}'_i \in \mathbb{R}^{F'}$$

- Weight matrix: $\mathbf{W} \in \mathbb{R}^{F' \times F}$
- Attention coefficients: α_{ij}

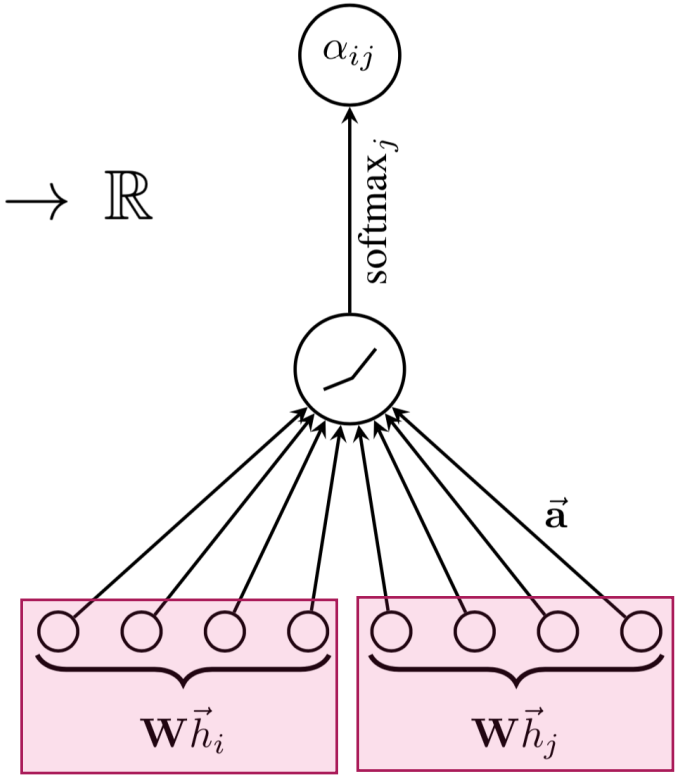
Graph Attention Networks (GATs)

- $e_{ij} = \text{LeakyReLU}\left(\vec{\mathbf{a}}^T [\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)$
- Shared attention mechanism: $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \rightarrow \mathbb{R}$
  - A single-layer feedforward neural network parametrized by a weight vector $\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$
- $\alpha_{ij} = \text{softmax}_j(e_{ij}) = \dfrac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}$



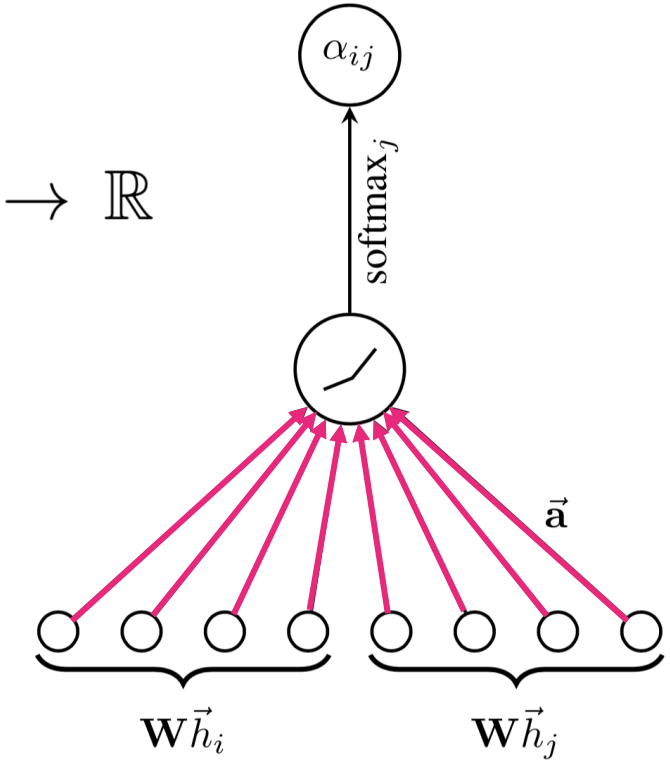
Graph Attention Networks (GATs)

- $e_{ij} = \text{LeakyReLU}\left(\vec{\mathbf{a}}^T [\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)$
- Shared attention mechanism: $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \rightarrow \mathbb{R}$
  - A single-layer feedforward neural network parametrized by a weight vector $\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$
- $\alpha_{ij} = \text{softmax}_j(e_{ij}) = \dfrac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}$



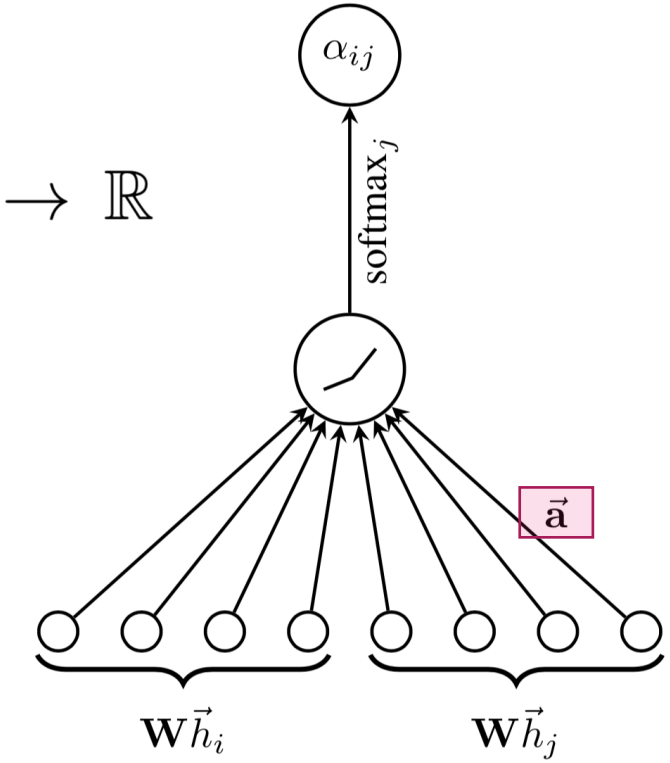
Graph Attention Networks (GATs)

- $e_{ij} = \text{LeakyReLU}\left(\vec{\mathbf{a}}^T [\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)$
- Shared attention mechanism: $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \rightarrow \mathbb{R}$
  - A single-layer feedforward neural network parametrized by a weight vector $\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$
- $\alpha_{ij} = \text{softmax}_j(e_{ij}) = \dfrac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}$



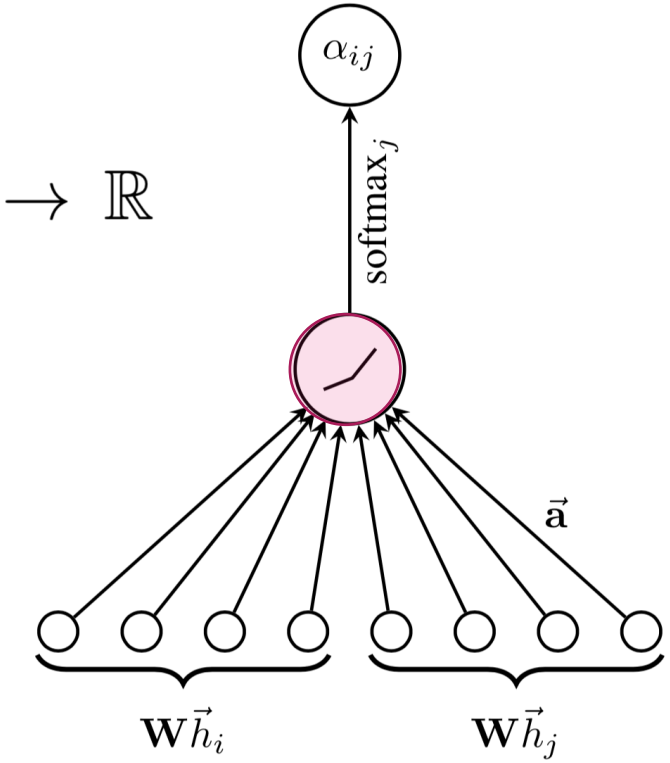
Graph Attention Networks (GATs)

- $e_{ij} = \text{LeakyReLU}\left(\vec{\mathbf{a}}^T [\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)$
- Shared attention mechanism: $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \rightarrow \mathbb{R}$
  - A single-layer feedforward neural network parametrized by a weight vector $\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$
- $\alpha_{ij} = \text{softmax}_j(e_{ij}) = \dfrac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}$



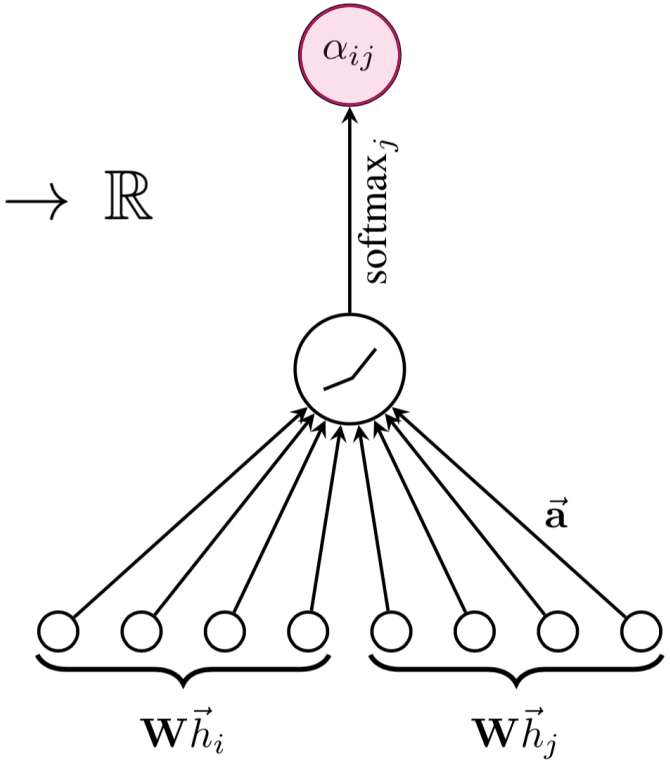
Graph Attention Networks (GATs)

- $e_{ij} = \text{LeakyReLU}\left(\vec{\mathbf{a}}^T [\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)$
- Shared attention mechanism: $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \rightarrow \mathbb{R}$
  - A single-layer feedforward neural network parametrized by a weight vector $\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$
- $\alpha_{ij} = \text{softmax}_j(e_{ij}) = \dfrac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}$



Graph Attention Networks (GATs)

- $e_{ij} = \text{LeakyReLU}\left(\vec{\mathbf{a}}^T [\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)$
- Shared attention mechanism: $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \rightarrow \mathbb{R}$
  - A single-layer feedforward neural network parametrized by a weight vector $\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$
- $$\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}$$



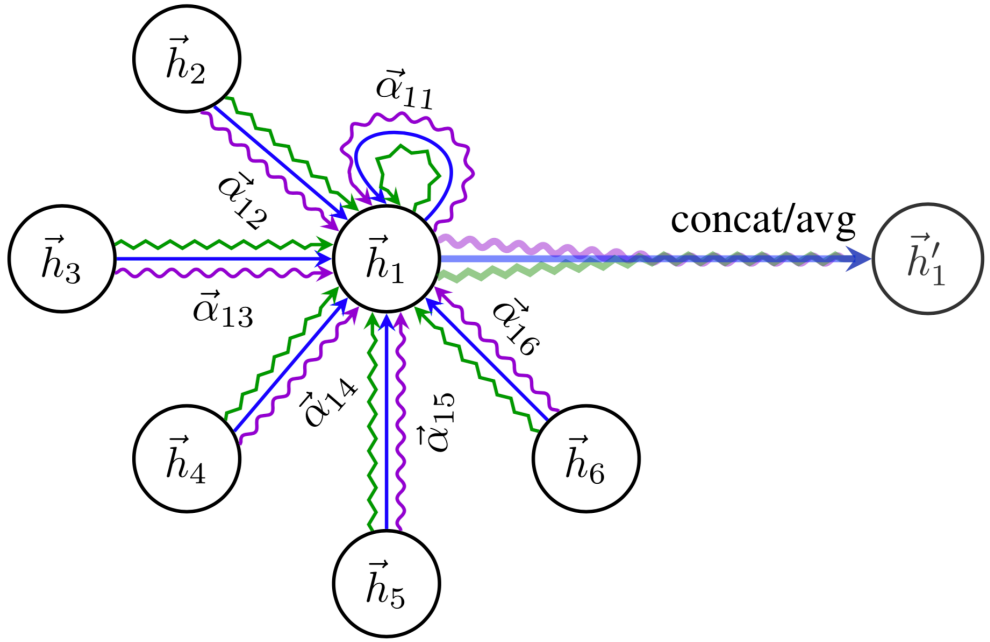
Graph Attention Networks (GATs)

- Multi-head attention
  - Feature concatenation

$$\vec{h}'_i = \Big\Vert_{k=1}^{K} \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right)$$

  - Feature averaging (for the final (prediction) layer of the network)

$$\vec{h}'_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right)$$



Experiments - Datasets

- Transductive
  - Citation networks
    - Nodes: documents
    - Edges: citations
    - Undirected
  - Features:
    - bag of words
  - 20 nodes/class for training
  - The training algorithm has access to all the nodes' feature vectors

	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)

Experiments - Datasets

- Inductive
  - PPI networks
  - Features:
    - Positional gene sets
    - Motif gene sets
    - Immunological signatures
  - Testing graphs remain completely unobserved during training

	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)

Experimental Setup

Transductive

- Used a 2-layer GAT model
- First layer: 8 attention heads, followed by an exponential linear unit (ELU)
- Second layer: single attention head used for classification
- Followed by a SoftMax activation
- L2 regularization applied (for coping with small dataset)
- Dropout applied to both layers

Inductive

- Used a 3-layer GAT model
- First 2 layers: 4 attention heads, followed by an exponential linear unit (ELU)
- Third layer: 6 attention heads used for multi-class classification
- Followed by a logistic sigmoid activation
- Training set was large – no regularization was applied
- No dropout was applied

Results – Transductive

- Evaluation metric: mean classification accuracy with standard deviation on test nodes after 100 runs
- GCN-64: graph convolution network that computes 64 hidden features

	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)	$81.7 \pm 0.5\%$	—	$78.8 \pm 0.3\%$
GCN-64*	$81.4 \pm 0.5\%$	$70.9 \pm 0.5\%$	$\mathbf{79.0} \pm 0.3\%$
GAT (ours)	$\mathbf{83.0} \pm 0.7\%$	$\mathbf{72.5} \pm 0.7\%$	$\mathbf{79.0} \pm 0.3\%$

Results – Inductive

- Evaluation metric: micro-averaged F1 score on the nodes of the two unseen test graphs, averaged after 10 runs
- GraphSAGE*: best GraphSAGE result obtained
- Const-GAT: GAT with constant attention mechanism
  - Assigning same importance to each neighbor

	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)	$\mathbf{0.973} \pm 0.002$

Concluding Remarks

- GATs: particular instance of MoNet
  - Uses node features for similarity computations rather than node's structural properties
  - Enables inductive learning
- Limitations:
  - Parallel computations: may be redundant
  - Needs manual tuning of neighbor distance

Thank you!

Questions

References

[1] P. Veli ̆ckovi ́c, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio,“Graph attention networks,” arXiv preprint arXiv:1710.10903, 2017.

[2] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, “Graph neural networks: A review of methods and applications,” arXiv preprint arXiv:1812.08434, 2018.

[3] W. L. Hamilton, R. Ying, and J. Leskovec, “Representation learning on graphs: Methods and applications,” arXiv preprint arXiv:1709.05584, 2017.