Graph Attention Networks

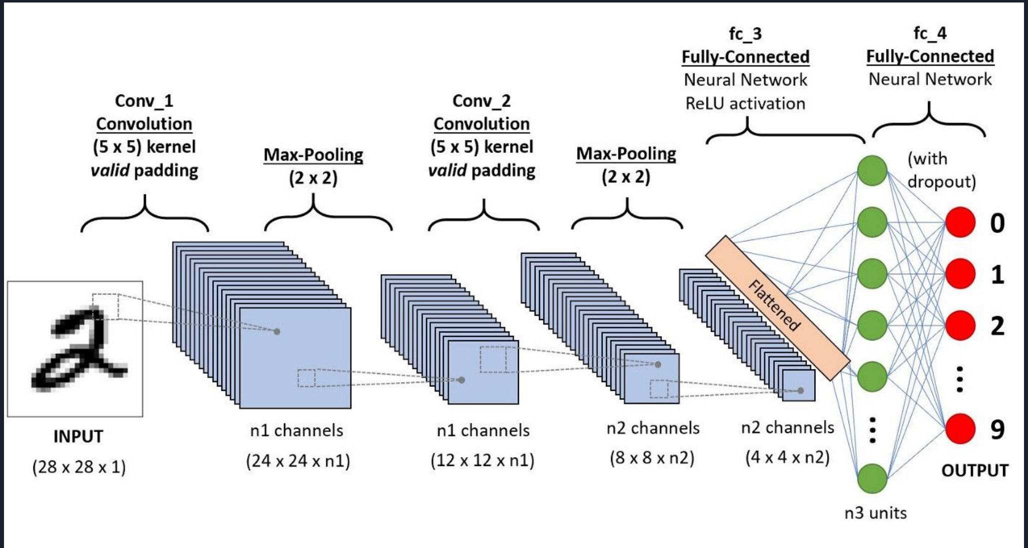
P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio

Presented by Benjamín Farías V.

Contents

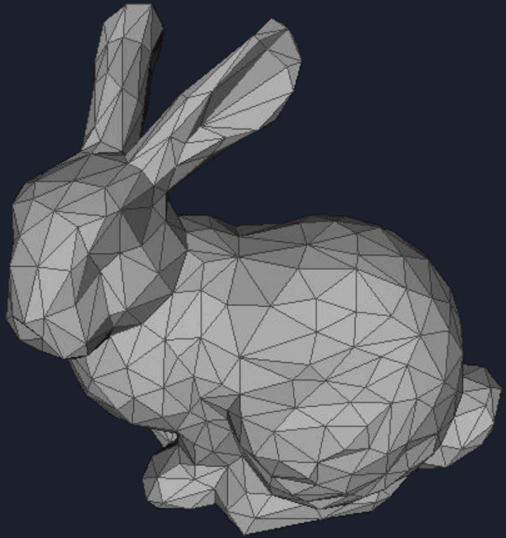
1. **Context**
2. Related Work
3. Model
4. Experiments
5. Conclusions
6. Personal Criticism

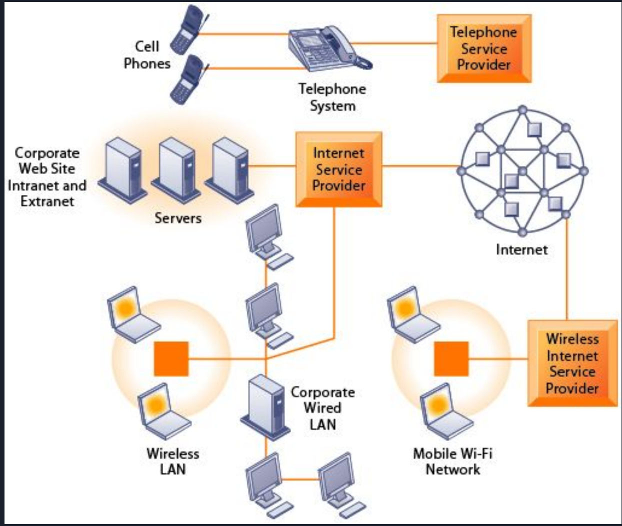
Context - CNNs



- Successful for **grid-like structured data** (ex. image classification)
- Efficiently reuse **local filters** with learnable parameters

Context - Graph Applications

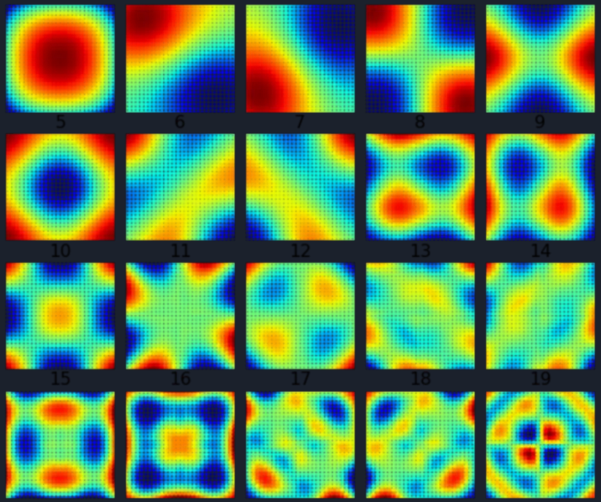




- Many tasks have an **irregular domain** (ex. 3D meshes, networks)
- Can be represented with **graphs**
- **Let's generalize convolutions to the graph domain!**

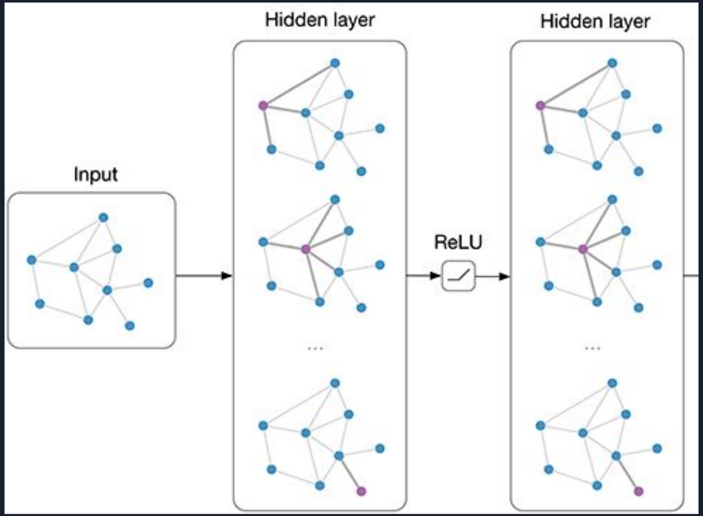
Context - Graph Convolutions

Spectral Approaches



Work with a spectral representation of the graph

Spatial Approaches

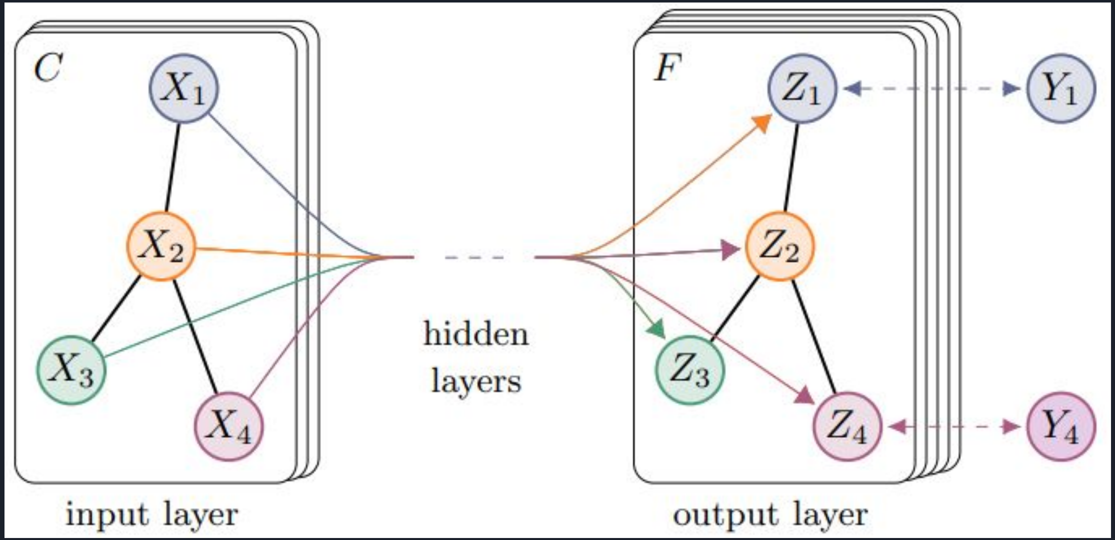


Define convolutions directly on the graph, using neighborhood

Contents

Related Work - Transductive Learning

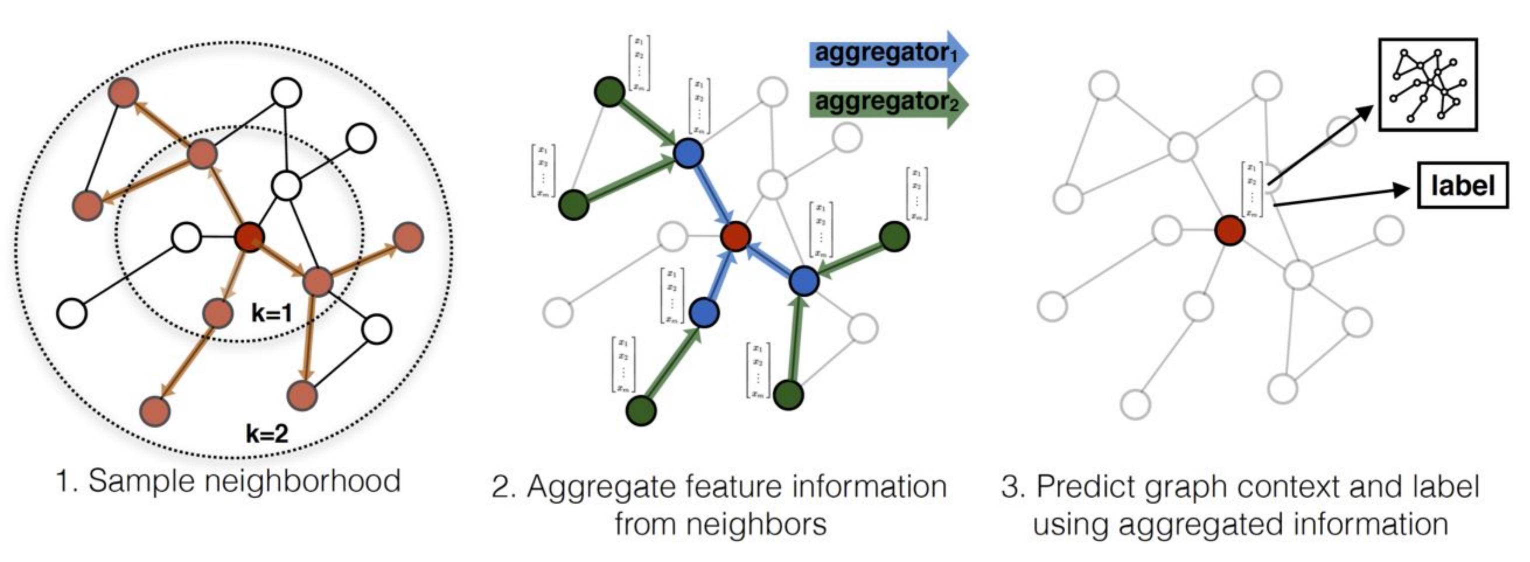
Graph Convolutional Networks (GCNs)



Localized first-order approximation of spectral graph convolutions

Related Work - Inductive Learning

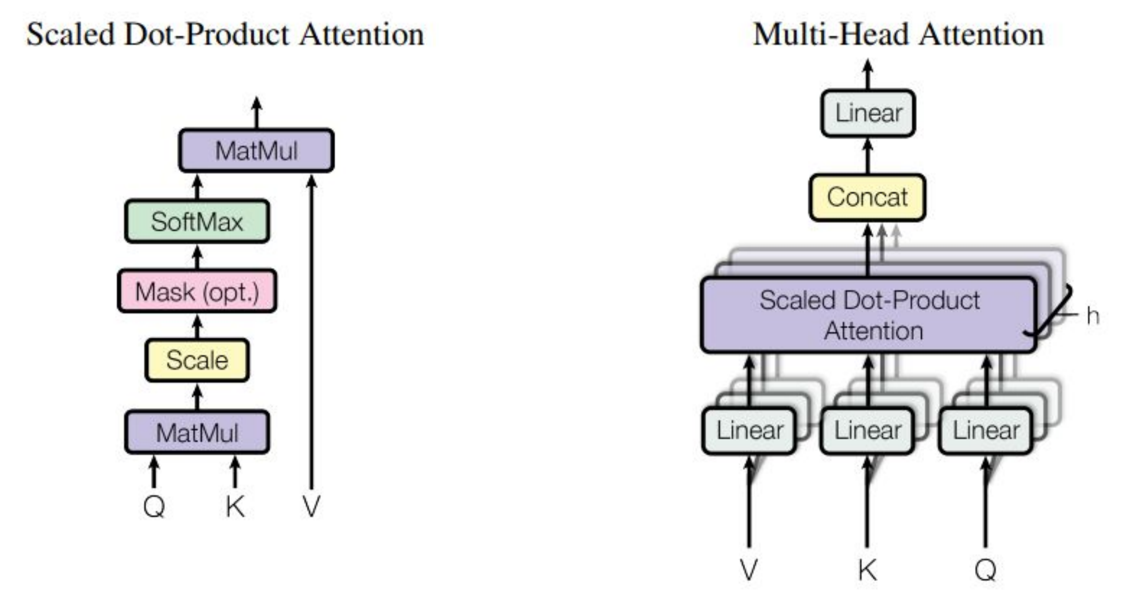
Graph Sample & Aggregate (GraphSAGE)



Sample and aggregate fixed-size neighborhoods to generate embeddings

Related Work - Attention Mechanisms

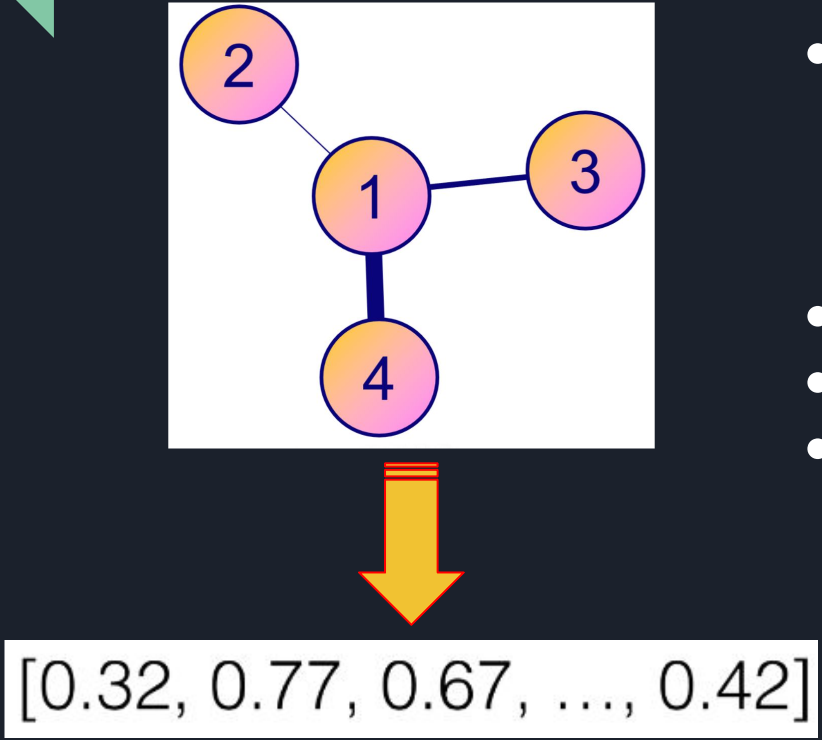
Transformer & Self-Attention



Self-Attention is sufficient for constructing a powerful model

Contents

Model - Graph Attention Network (GAT)



- Compute node **embeddings** using a **self-attention** mechanism

- Highly **parallelizable** across node pairs
- Flexible for different graph structures
- Directly applicable to **inductive** learning

Model - GAT Layer

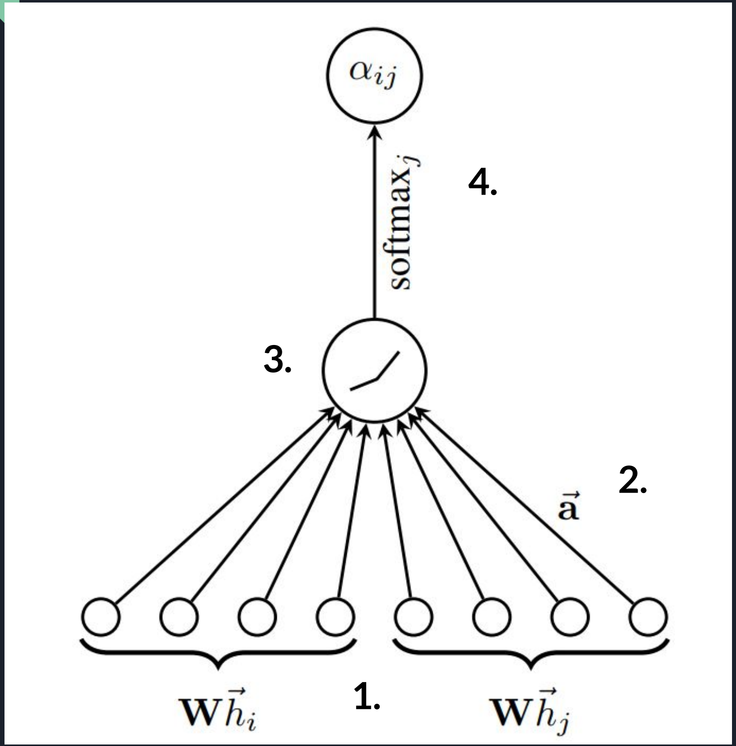
- Building blocks used to construct the **GAT**
- **INPUT:** Set of **node features** for every node in the graph

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

- **OUTPUT:** New set of **node features** that is more **rich**

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

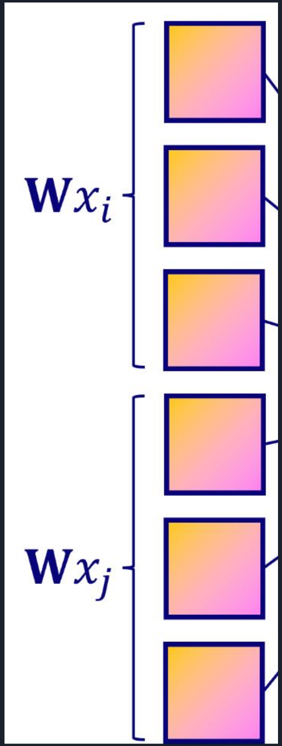
Model - GAT Layer



Self-Attention Mechanism

1. Shared **Linear Transformation**
2. **Single Layer** Feedforward Step
3. **Nonlinear** Activation Function
4. **Normalization**

Model - GAT Layer



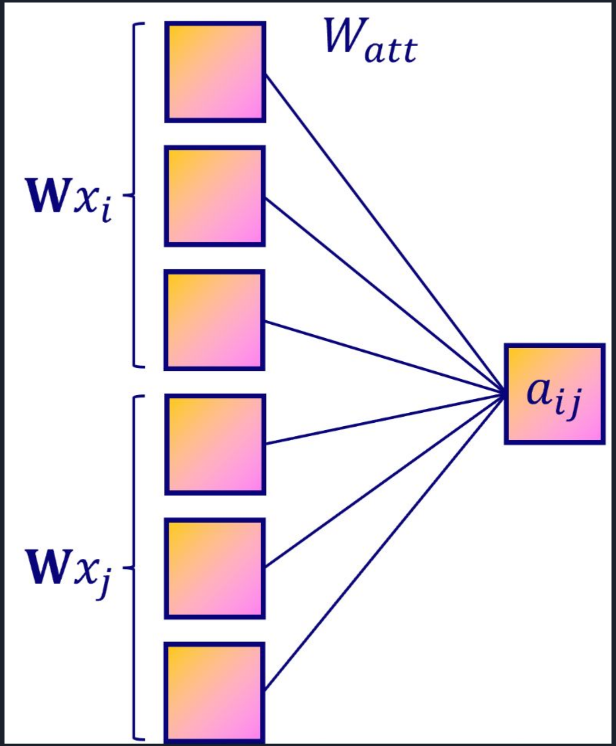
Self-Attention Mechanism

1. Shared **Linear Transformation**

- A **learnable weight matrix** is applied to every node
- Node pairs are then **concatenated**

$$[\mathbf{W}x_i \,\|\, \mathbf{W}x_j]$$

Model - GAT Layer



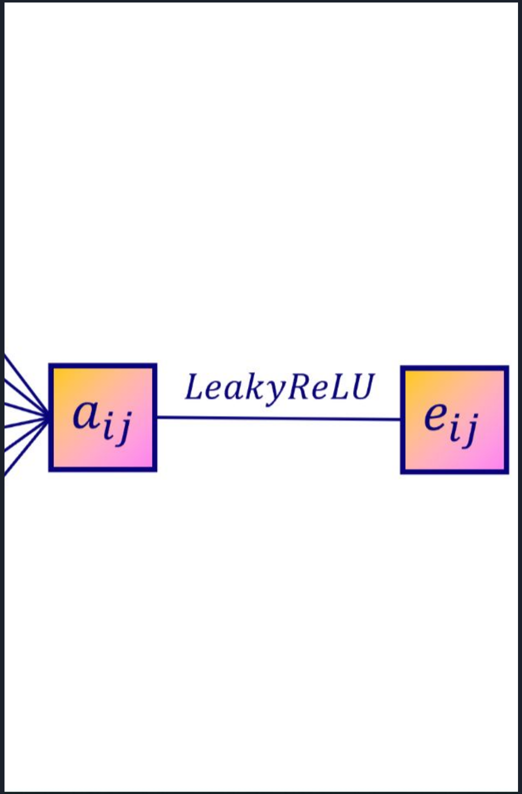
Self-Attention Mechanism

2. **Single Layer** Feedforward Step

- **Learnable** linear transformation
- **Mixes information** from both nodes

$$a_{ij} = W_{att}^{t}[\mathbf{W}x_i \parallel \mathbf{W}x_j]$$

Model - GAT Layer



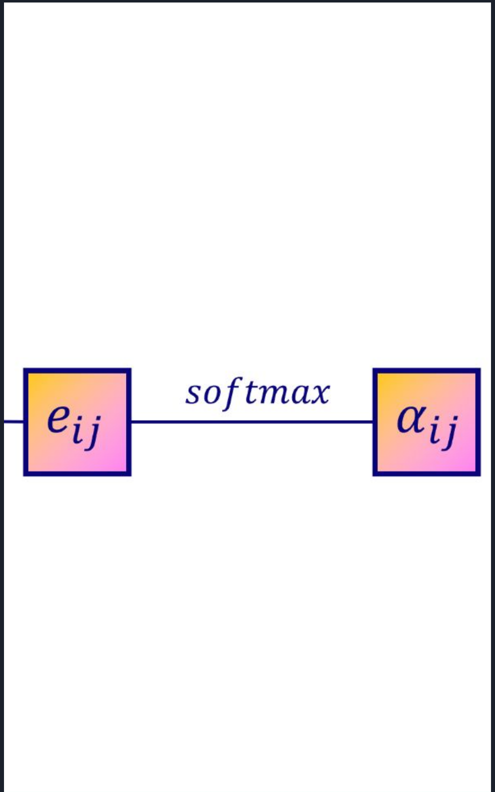
Self-Attention Mechanism

3. **Nonlinear** Activation Function

- The **LeakyReLU** function is chosen
- Allows for more flexibility in activation

$$e_{ij} = LeakyReLU(a_{ij})$$

Model - GAT Layer



Self-Attention Mechanism

4. **Normalization**

- The **Softmax function** is applied
- Normalize across all neighbors

$$\alpha_{ij} = softmax_j(e_{ij}) = \frac{exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} exp(e_{ik})}$$

Model - GAT Layer

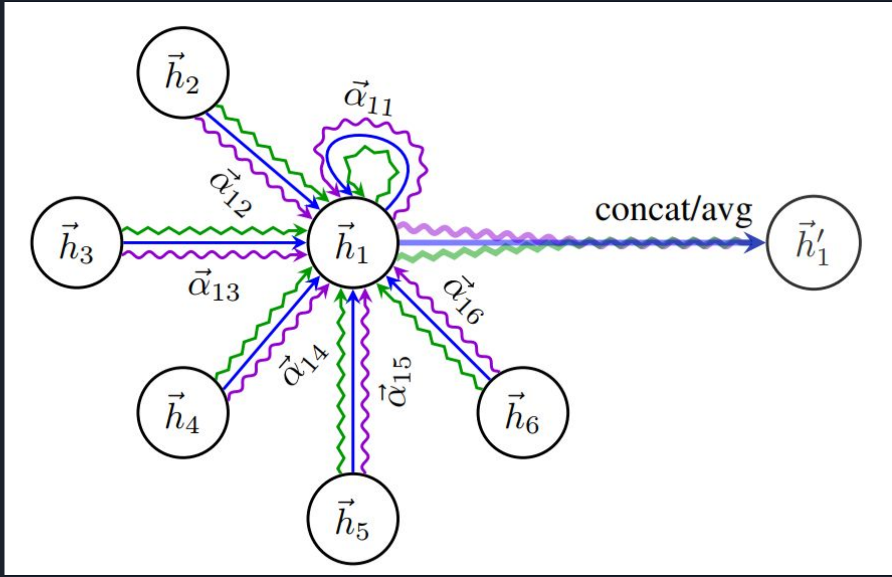
$$\vec{h}'_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W} \vec{h}_j\right)$$

Self-Attention Mechanism (Output)

- Compute **linear combination** of node features
- Use **attention coefficients** previously calculated **(with neighborhood)**
- Apply **nonlinearity** to obtain **output**

- **Self-Attention is not very stable!**

Model - GAT Layer



Multi-Head Attention

- Use **K independent** attention mechanisms
- **Aggregate** features obtained from **all** attention heads

Model - GAT Layer

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

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

Multi-Head Attention

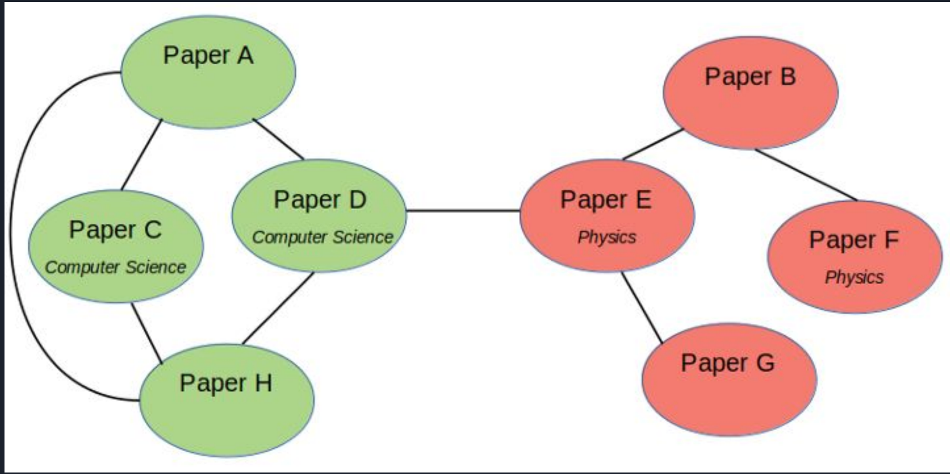
- For **hidden** layers, **concatenate**
- For **prediction** layers, **average**

Model - Advantages of GAT

- **Efficiency:** Time complexity on par with **GCNs**, can be parallelized.
- **Capacity:** Can assign **different importances** to nodes of the same neighborhood.
- **Flexibility:** Shared attention mechanism allows for **inductive** learning and graph **generalization**.
- **Representation:** Uses information from entire neighborhood, unlike **GraphSAGE**.

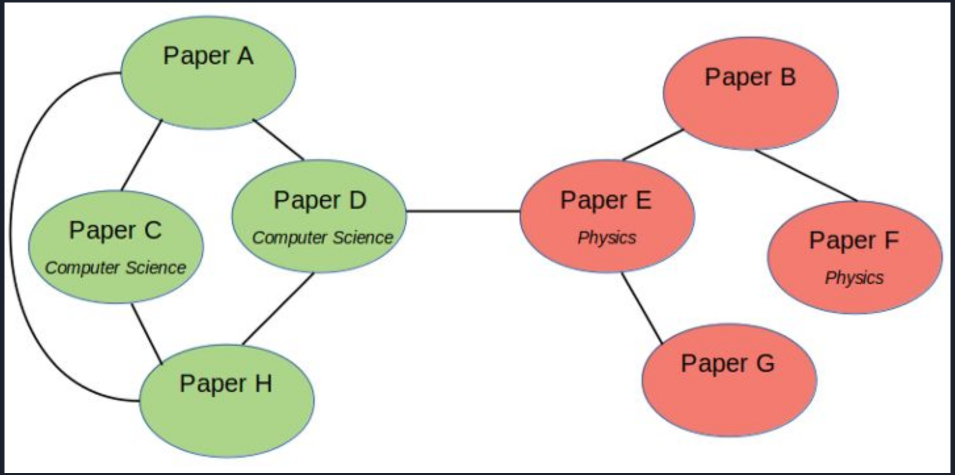
Contents

Experiments - Transductive Learning



- Node **classification** on **Citation Network** datasets (ex. Cora)
- Initial node features come from a **bag-of-words** representation
- **Baseline:** GCN, Chebyshev, MoNet and older approaches

Experiments - Transductive Learning



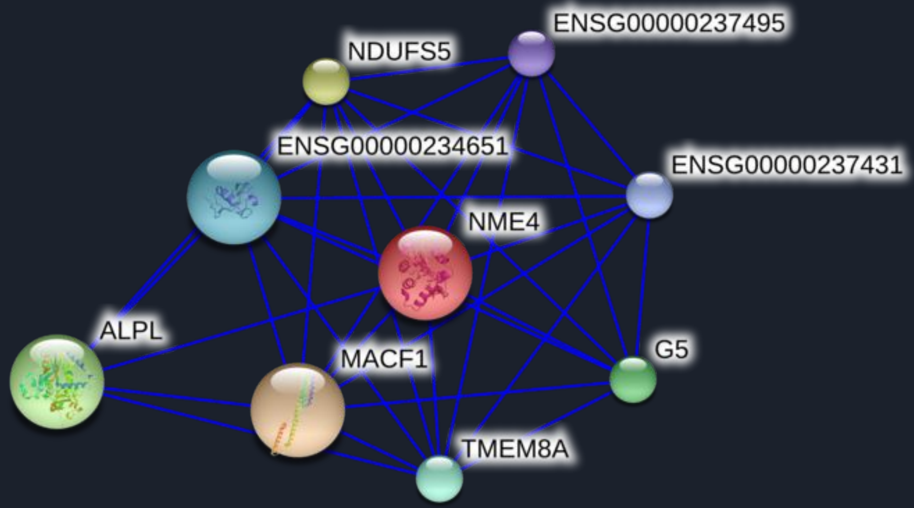
- **Model:** Two-Layer GAT
- First layer has **8 attention heads** and **ELU** activation
- Second layer has a **single attention head** and **Softmax** activation
- **Regularization (L2) and Dropout** applied during training

Experiments - Transductive Learning

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)	$\mathbf{83.0} \pm 0.7\%$	$\mathbf{72.5} \pm 0.7\%$	$\mathbf{79.0} \pm 0.3\%$

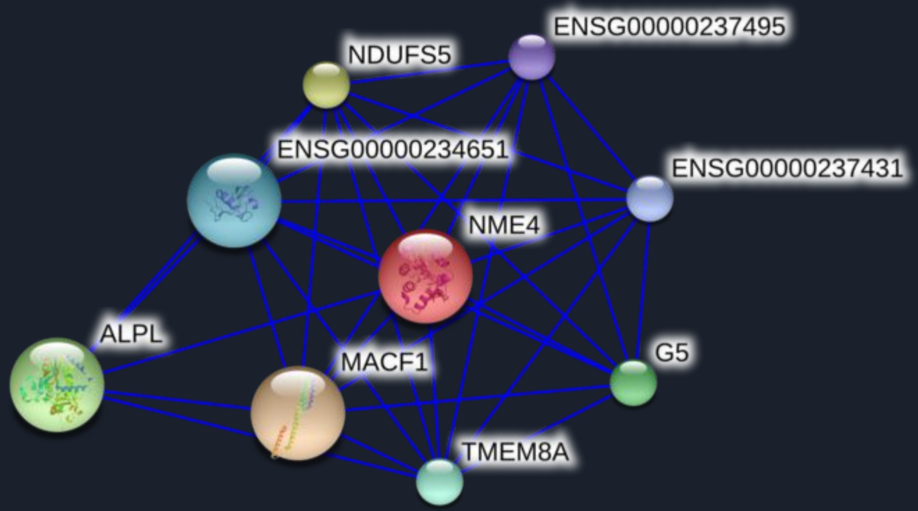
- **Metric:** Mean Classification Accuracy after 100 runs
- Improvement over **GCNs** by around **1.5%**
- State of the art performance!

Experiments - Inductive Learning



- Multi-Label **classification** on a **Protein Interaction Network** dataset
- Initial node features represent **genetic information** for a protein
- **Baseline:** GraphSAGE

Experiments - Inductive Learning



- **Model:** Three-Layer GAT
- First two layers have **4 attention heads** and **ELU** activation
- Third layer has **6 attention heads** and **Logistic Sigmoid** activation
- **Skip Connections** across the middle layer

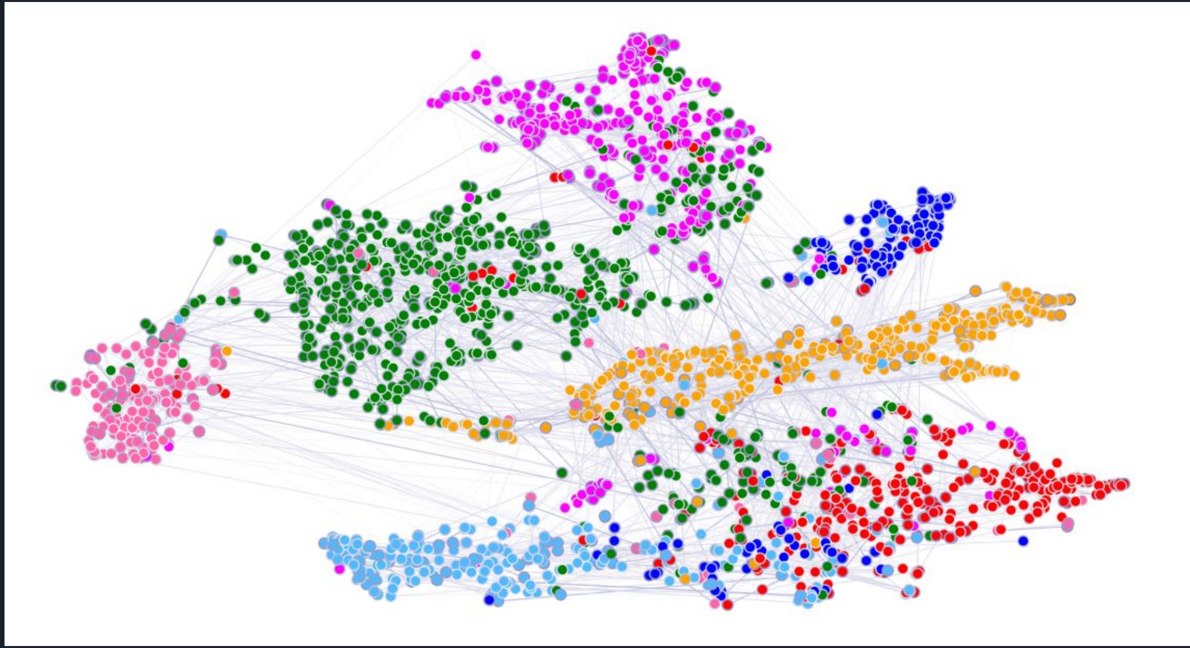
Experiments - Inductive Learning

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)	$\mathbf{0.973} \pm 0.002$

- **Metric:** Micro-Averaged F1 Score after 10 runs
- Improvement over **GraphSAGE** by around **20.5%**
- State of the art performance!

Experiments - Feature Visualization



- Discernible **clustering** in the **t-SNE** projected 2D space (on Cora)

Contents

Conclusions

- The **GAT** model is efficient, flexible and has good generalization potential
- Experiments show **state of the art** performance on node classification
- **Future Work:** Interpretability, graph classification, edge features

Contents

Personal Criticism

Good:

- Nice application of attention mechanisms in the context of graph learning
- The model can generalize well to graphs never seen in training

Bad:

- Training can be slow on massive datasets, due to using all neighbors
- Node outliers are not well represented by the learned attention

Bibliography

[1] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y. (2017). *Graph Attention Networks.*

[2] Kipf, T., Welling, M. (2016). *Semi-Supervised Classification with Graph Convolutional Networks.*

[3] Hamilton, W., Ying, R., Leskovec, J. (2017). *Inductive Representation Learning on Large Graphs.*

[4] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L., Polosukhin, I. (2017). *Attention Is All You Need.*