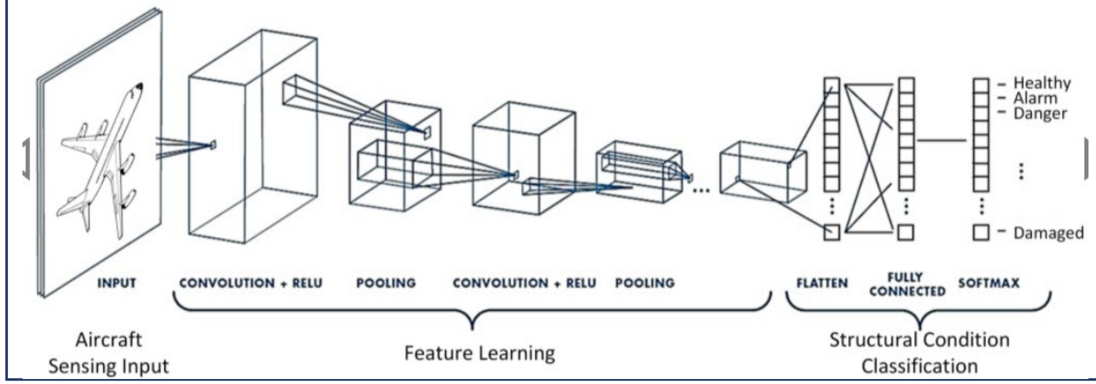
# Graph Attention Networks

Lab Intern 이준모

# Contents

# Introduction – (1)

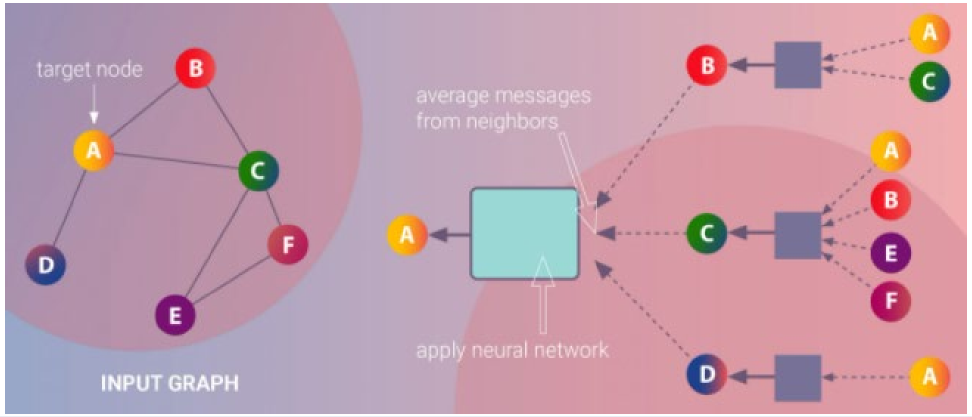
## Convolutional Neural Networks (CNNs)




- Good at image classification, machine translation
- But, only in Grid Structure
- Cannot apply to 3D mesh, social network

Tabian, I.; Fu, H.; Sharif Khodaei, Z. A Convolutional Neural Network for Impact Detection and Characterization of Complex Composite Structures. Sensors 2019, 19, 4933.

## Graph Neural Networks (GNN)




- Appeared in Gori et al. (2005)
- Introduced as Generalized version of RNN
- Can directly deal with a more general graphs

https://perfectial.com/blog/graph-neural-networks-and-graph-convolutional-networks/

# Introduction – (2)

| Spectral Representation | Spatial Representation |
|---|---|
| • Convolution operation defined in Fourier domain<br>• Can not be directly applied to a graph with different structure<br>• Ex) Spectral Graph CNN (Bruna et al. ICLR 2015) | • Define convolutions directly on the graph, on groups of spatially close neighbors<br>• Impressive performance across several large-scale datasets<br>• Ex) GraphSAGE (Hamilton et al. NIPS 2017) |

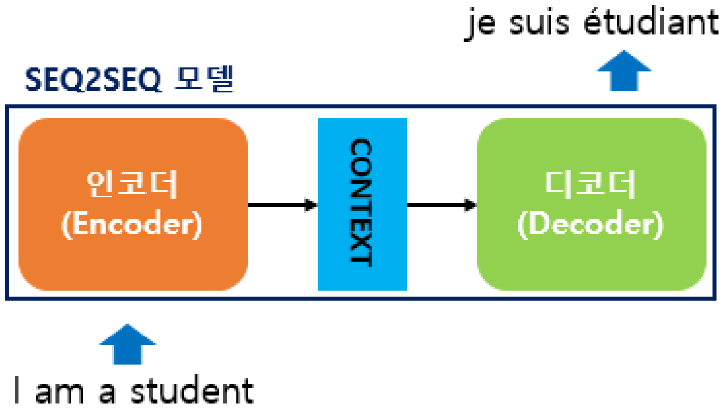
↓

## Graph Attention Networks(GAT)

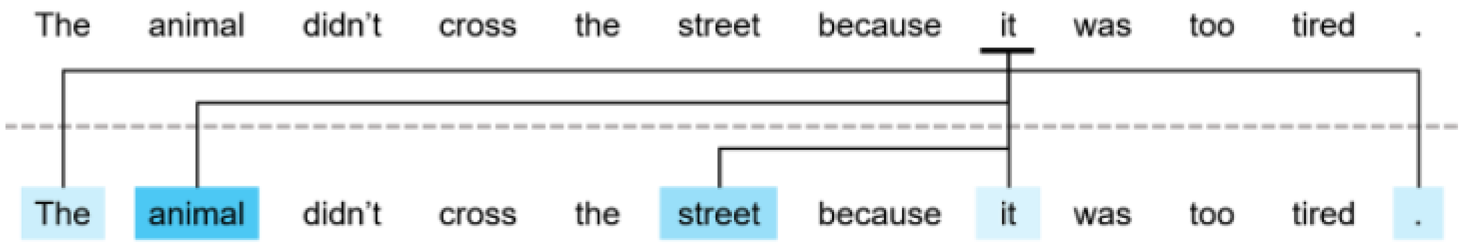
- Inspired by self-attention structure of transformer (Vaswani et al. NIPS 2017)
- Apply to Node Classification
- Compute the hidden representations of each node in the graph by a self-attention strategy

Veličković, Petar, et al. "Graph attention networks." arXiv preprint arXiv:1710.10903 (2017).

# Self-Attention




Seq2Seq Model in NLP




Self-attention mechanism in NLP

- Loss of Information (by fixed vector)
- Vanishing gradient

- Pronoun is depend on context
- Numerically indicate which of the input elements should be considered important

## Characteristics of Self-Attention in Graph

- Efficient operation by parallel computing
- Can be Applied to graph nodes having different degrees by specifying arbitrary weights to neighbors
- Directly applicable to inductive learning problems

Reference from : https://wikidocs.net/24996, https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

# GAT Architecture – (1)

## Describing a Single graph attentional layer

- Sole layer utilized throughout all of the GAT structure
- Closely follows the work of Bahdanau et al. (2015)

## Notation Definition

| Input to layer | Output to layer | Linear Transformation | Self-Attention Transformation |
|---|---|---|---|
| Set of Node Features<br>$\mathbf{h} = \{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_N\}$<br>$\vec{h}_i \in \mathbb{R}^F$<br>$N$ : number of Nodes<br>$F$ : number of Node's Features | Set of New Node Features<br>$\mathbf{h}' = \{\vec{h}'_1, \vec{h}'_2, \ldots, \vec{h}'_N\}$<br>$\vec{h}'_i \in \mathbb{R}^{F'}$<br>$N$ : number of Nodes<br>$F'$ : number of Output Node's Features | Transform input feature into higher-level features<br>$\mathbf{W} \in \mathbb{R}^{F' \times F}$<br>$W$ : weight matrix | Define Self-Attention Transformation after Linear Transformation<br>$a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \rightarrow \mathbb{R}$ |

Reference from : https://thejb.ai/gat/

# GAT Architecture – (2)

## Attention Coefficients

- Defined as $e_{ij} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$ : The importance of node $j$'s features to node $i$

## Masked Attention

- Compute $e_{ij}$ for nodes $j \in \mathcal{N}_i$, where $\mathcal{N}_i$ is some neighborhood of node $i$ (include i itself)

## Normalized Attention Coefficients

- Defined as $\alpha_{ij} = \mathrm{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}$ ∵ to make coefficients easily comparable
- In this experiments, a is a single-layer feedforward neural network

Attention Coefficients

$$\alpha_{ij} = \frac{\exp\left(\mathrm{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\mathrm{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$

$\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$ : a weight vector

LeakyReLU : activation function

where $\cdot^T$ represents transposition and $\|$ is the concatenation operation

Reference from : https://thejb.ai/gat/

# GAT Architecture – (3)

## Output Layers

- Defined as $\vec{h}'_i = \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}\mathbf{W}\vec{h}_j\right)$ $\sigma$ : Activation function (Use ELU)

## Multi-head Attention

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

K independent attention mechanisms + Concatenation

∵ to stabilize the learning process of self-attention

Output will consist of $KF'$ features for each node

## Final(Prediction) Layers

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

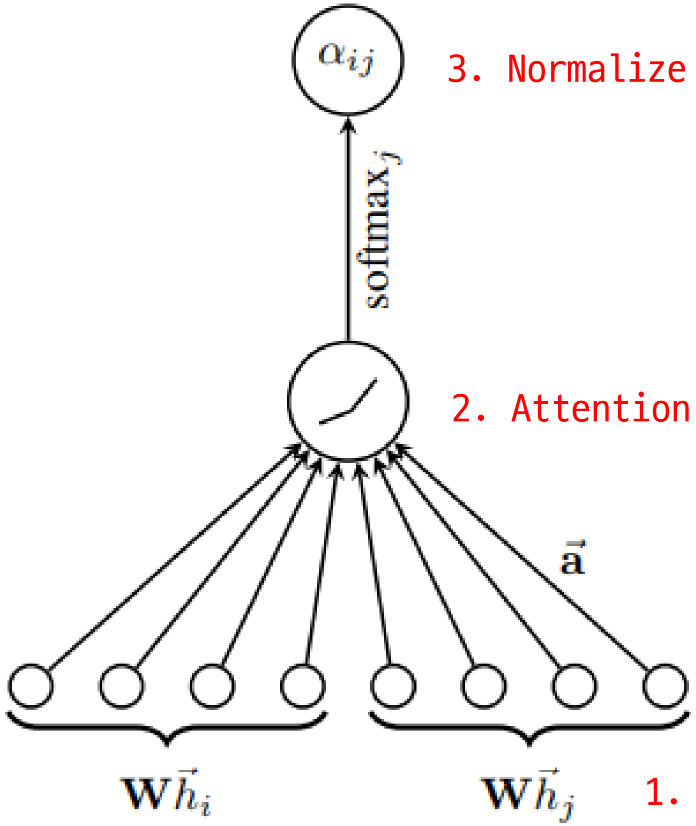
Average attention values instead of concatenating

$\sigma$ : Activation function (Use Softmax or Logistic)

Reference from : https://thejb.ai/gat/

# GAT Architecture – (4)

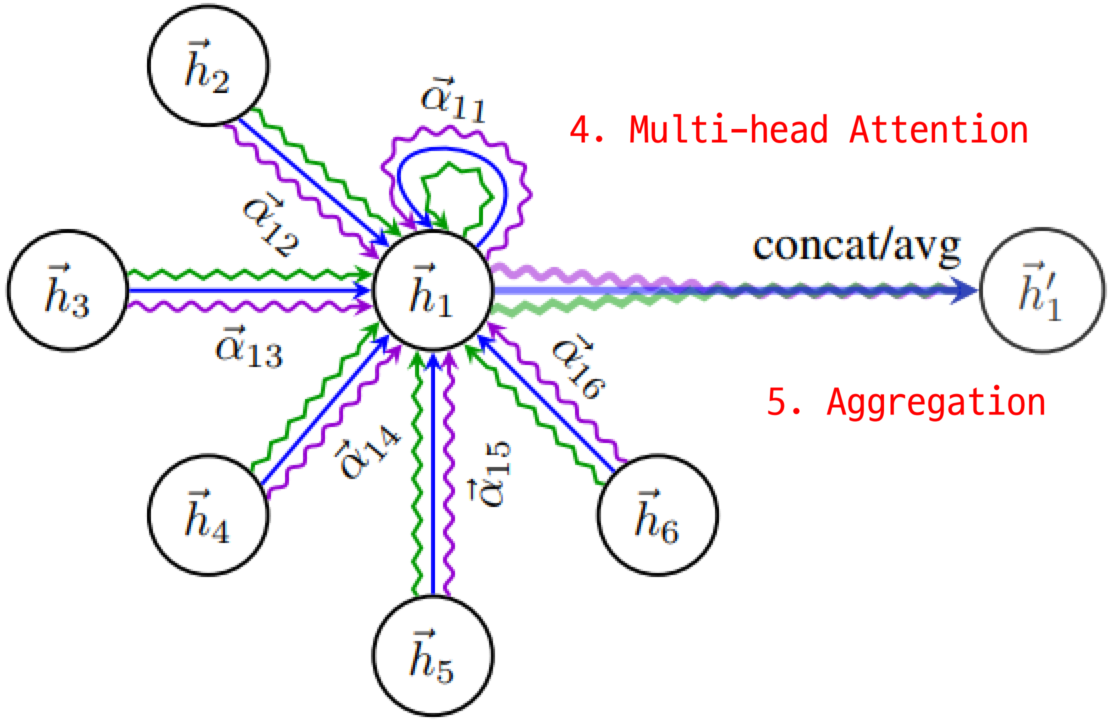
## Summary




3. Normalize Coefficients

2. Attention Coefficients

1. Linear Transformation




4. Multi-head Attention

5. Aggregation

Veličković, Petar, et al. "Graph attention networks." arXiv preprint arXiv:1710.10903 (2017).

# Comparison to related work

## vs GCN

- GAT allows for assigning different importances to nodes of a same neighborhood

-> Improve Model Capacity

- Analyzing the learned attentional weights lead to benefits in interpretability

## Attention Mechanism

- This is applied in a shared manner to all edges in graph

-> Learning can procced without access to the global graph structure

  - The graph is not required to be undirected
  - Directly applicable to inductive learning
  - Effective Computation and be flexible to graph variations

## Layer

- GAT Layer leverages sparse matrix operations -> reduce storage complexity -> larger graph
- Framework supports mm for rank-2 tensor -> limits the batching capabilities -> Future Work!

Reference from : https://greeksharifa.github.io/machine_learning/2021/05/29/GAT/

# Experiment

## Transductive Learning

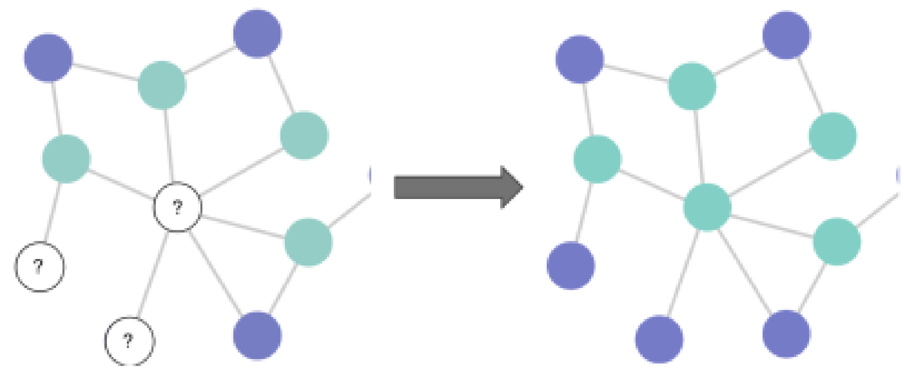
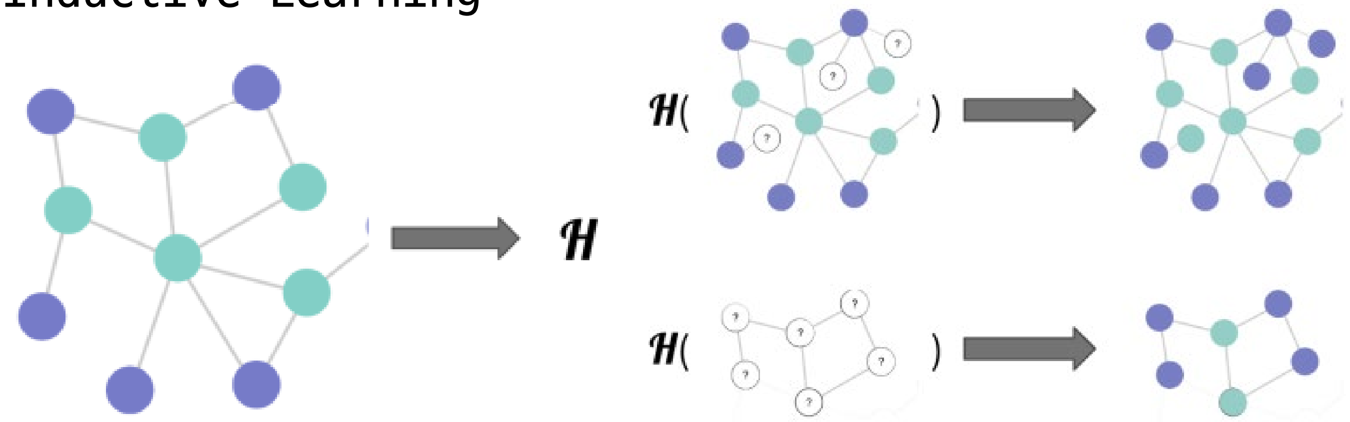


**Figure 1: Node classification in transductive settings. At the training time, the learning algorithm has access to all the nodes and edges, including those nodes for which labels are to be predicted (denoted by question marks).**

Predict by feature of nodes without Model

Example : kNN Method

## Inductive Learning



(a) A model $\mathcal{H}$ is learned over some graph

(b) The model is then by applied to new nodes and edges

**Figure 2: Node classification in inductive settings. Once learned, the model can be applied to new unseen nodes (denoted by question marks). There may or may not exist edges between such new nodes and the nodes used for training.**

Make a generalized model and predict by using it

Example : Typical Supervised Learning

Mishra, Pushkar, et al. "Node masking: Making graph neural networks generalize and scale better." *arXiv preprint arXiv:2001.07524* (2020).

# Experiment

Table 1: Summary of the datasets used in our experiments.

| | 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) |

| | | | | |
|---|---|---|---|---|
| Nodes | Publications | Publications | Publications | Protein |
| Edges | Citation Network | Citation Network | Citation Network | Interaction |
| Features | Unique Words | Unique Words | Unique Words | Positional gene sets/motif gene set/ immunological signatures |
| Classes | Label | Label | Label | Protein roles |

Reference from : https://linqs.soe.ucsc.edu/data

# Evaluation

| Transductive Learning | Inductive Learning |
|---|---|
| • Two-layer GAT model<br>• First Layer : K = 8 attention heads computing F’= 8 features with ELU<br>• Second Layer : a single attention head computes C classes by a softmax<br>• $L_2$ Regularization & Dropout | • Three-layer GAT model<br>• First Two Layer : K = 4 attention heads computing F’= 256 features with ELU<br>• Final Layer : K = 6 attention heads computing 121 classes by a logistic sigmoid<br>• No $L_2$ Regularization & Dropout |

Table 2: Summary of results in terms of classification accuracies, for Cora, Citeseer and Pubmed. GCN-64* corresponds to the best GCN result computing 64 hidden features (using ReLU or ELU).

*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% |

Table 3: Summary of results in terms of micro-averaged $F_1$ scores, for the PPI dataset. GraphSAGE* corresponds to the best GraphSAGE result we were able to obtain by just modifying its architecture. Const-GAT corresponds to a model with the same architecture as GAT, but with a constant attention mechanism (assigning same importance to each neighbor; GCN-like inductive operator).

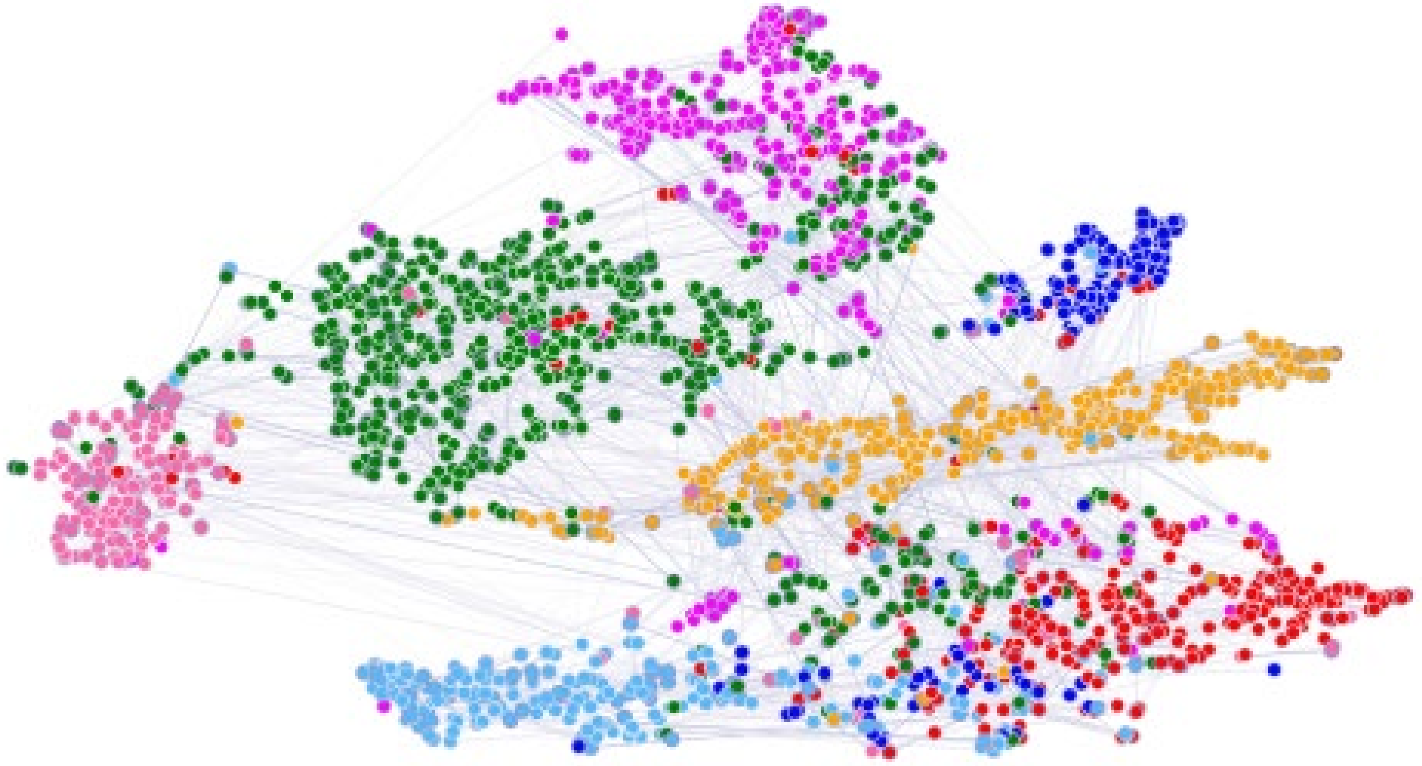
*Inductive*

| 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 |

Veličković, Petar, et al. "Graph attention networks." arXiv preprint arXiv:1710.10903 (2017).

# Feature Representations

t-SNE plot of the computed feature representations of GAT (first hidden layer)



Good Interpretability!

Color = class of Node

Edge thickness = aggregated normalized attention coefficients

Veličković, Petar, et al. "Graph attention networks." arXiv preprint arXiv:1710.10903 (2017).

# Conclusion & Future Work

## Conclusion

New Convolution-style Neural Networks on graph-structured data by masked self-attentional layers

1. Computationally efficient
2. Allowing for assigning different importances to different nodes within a neighborhood
3. Inductive Learning

## Future Work

1. Overcoming the Practical Problem ( Handling larger batch sizes )
2. Use attention mechanism to analyze on the model interpretability
3. Extend the method to perform graph classification
4. Extend the model to incorporate edge features

Veličković, Petar, et al. "Graph attention networks." arXiv preprint arXiv:1710.10903 (2017).

# Any Questions?