# GLOBAL SELF-ATTENTION GRAPH CONVOLUTION NETWORKS

## CS 535

Chen Wang, Chengyuan Deng

Advisor: : Sungjin Ahn

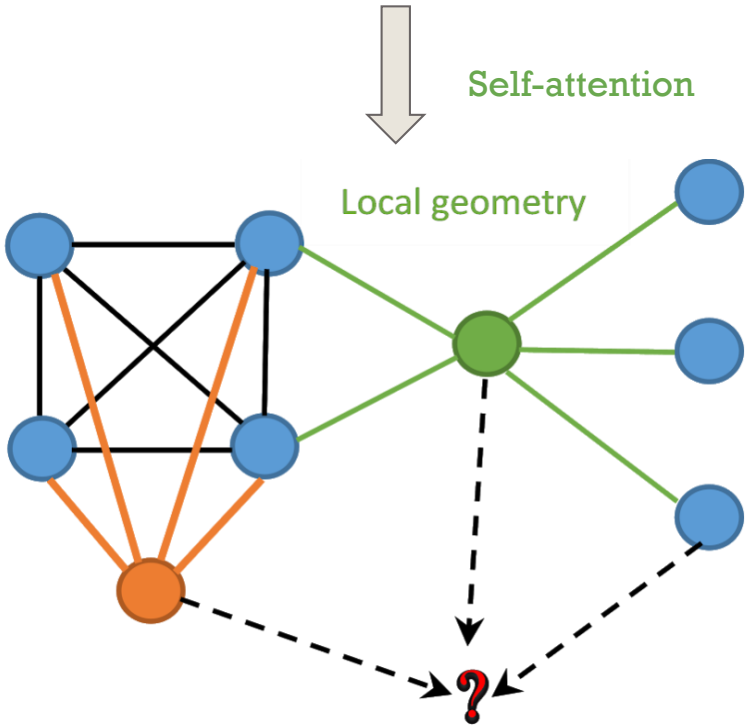
# OUTLINES

- **Introduction**
- **Background & Related Work**
- **Method**
- **Experiments**
- **References**

# INTRODUCTION

- Topic

  Leverage Global Information in self-attention Graph convolution networks

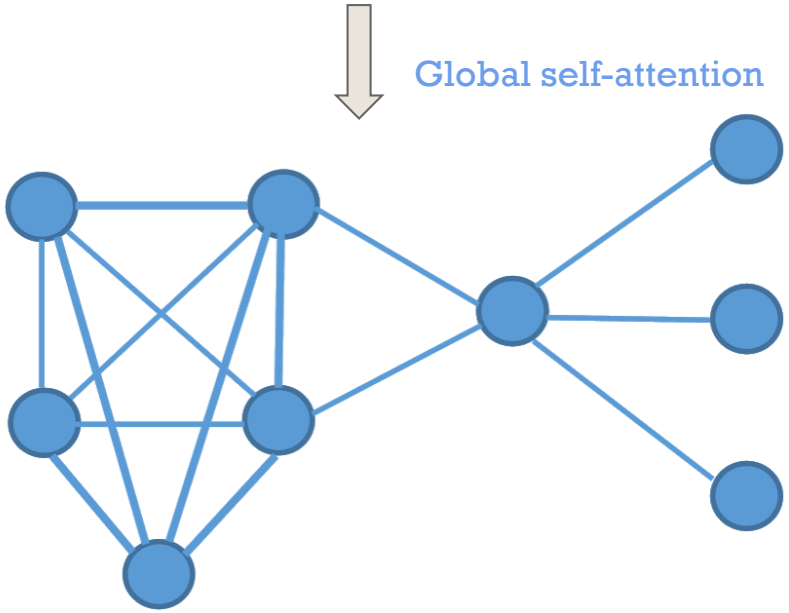


# INTRODUCTION

## ➢ Motivation

SA-GAN(self-attention GAN)(Han Zhang et all, 2018) proposed a novel global self-attention mechanism into convolutional GANs.

The self-attention layer takes features globally from previous layer as input, and calculate attention accordingly.



# BACKGROUND

## ➢ Graph Convolution Networks

- GCNs introduce a scalable approach based on approximation of spectral convolutions on graphs.

- Learn a function of features on Graph (V, E) with following inputs:
  - Feature descriptions of each node
  - Representation of graph structure (e.g. adjacency matrix)

# BACKGROUND

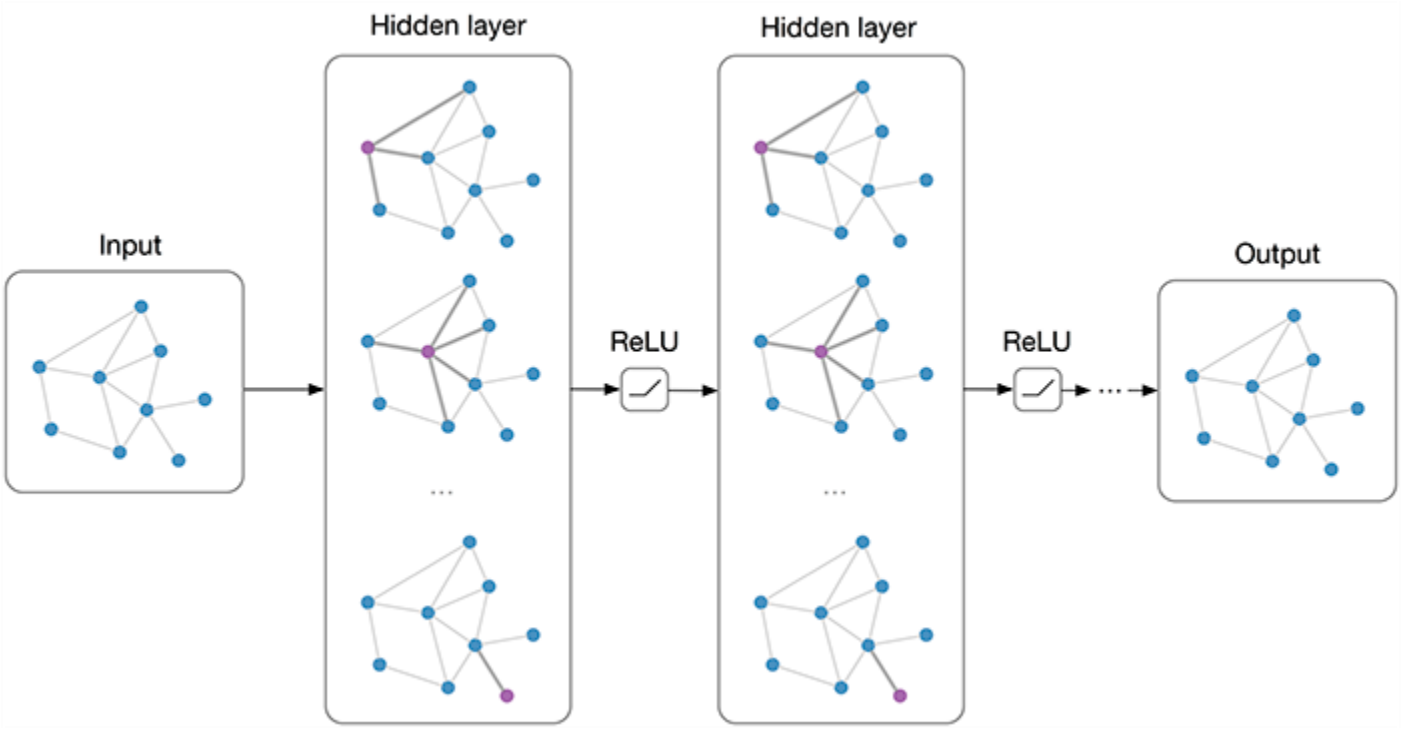
## ➢ Graph Convolution Networks

- The network layers are connected by convolutional projection of input graph with adjacency matrix A

- The output of each layer could be regarded as a non-linear function of previous layer output and adjacency matrix

# BACKGROUND

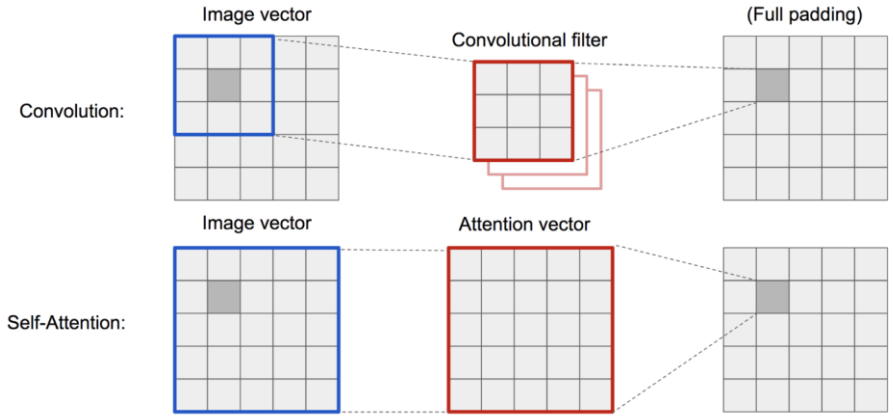
- Graph Convolution Networks




https://tkipf.github.io/graph-convolutional-networks/

# RELATED WORK

- Self-attention in SA-GAN




Each element in image vector has attention score at corresponding position in attention vector

The final output of self-attention layer adds back the input feature map

# METHOD

- Formalization of the idea:
  - GCN layers:
    The output of each layer is:

$$H^{(l+1)} = f(H^{(l)}, A)$$
$$= \sigma\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right)$$

where D serves as normalization of A and W as the weight matrix

$$D = \mathrm{diag}\left(\sum_{j=1}^{N}\hat{A}_j\right)$$

# METHOD

- Formalization of the idea:
  - self-attention layers:

    The attention score is calculated by:

$$\beta_{j,i} = \mathbf{softmax}(s_{i,j}), \text{where } s_{i,j} = (W_{1,i}X_i)^{\mathrm{T}}(W_{2,j}X_j)$$

    The output of self-attention layer is:

$$o_j = W(\sum_{i=1}^{N} \beta_{j,i} W_h X_i)$$

    The final output:

$$\boldsymbol{y_i} = \gamma \boldsymbol{o_i} + \boldsymbol{x_i}$$

# METHOD

- Semi-supervised Node Classification
  - Forward model takes the simple form:

$$Z = f(X, A) = \mathrm{softmax}\Big(\hat{A}\ \mathrm{ReLU}\Big(\hat{A}XW^{(0)}\Big)\ W^{(1)}\Big)$$

where $W^{(0)} \in \mathbb{R}^{C\times H}$ is input-to-hidden weight and $W^{(1)} \in \mathbb{R}^{H\times F}$ is hidden-to-output weight

  - Evaluate Cross-entropy Error:

$$\mathcal{L} = -\sum_{l\in\mathcal{Y}_L}\sum_{f=1}^{F} Y_{lf}\ln Z_{lf}$$

where $\mathcal{Y}_L$ is the set of node indices that have labels.

# EXPERIMENTS

## Experiments on two tasks:

- Semi-supervised Node Classification
- Supervised Graph Classification

## Graph Network Setup:

- Graph Convolutional Network following the framework of (Kipf & Welling, 2016)
- Performance with/without the Global Self-attention layer

# EXPERIMENTS

High-level Summary:

- Global Self-Attention GCNs outperform plain GCNs on test data in general
- Global Self-Attention GCNs perform especially well when provided reliable node features
- Global Self-Attention GCNs are significantly better on generalization

# EXPERIMENTS: Node Classification

Datasets:

- Karate:
  - 34 nodes, no given feature
  - generate 2-d feature with a 3-layer GCN
- ENZYMES:
  - 126 nodes, 18-dim given feature
- Protein:
  - 620 nodes, 29-dim given feature

# EXPERIMENTS: Node Classification

| | Performance | **Karate** | **ENZYMES** | **Protein** |
|---|---|---|---|---|
| Plain GCN | Train Accuracy | 0.7500 | 1.0 | 1.0 |
| | Test Accuracy | 0.1667 | 0.7903 | 0.4061 |
| GSA GCN | Train Accuracy | 1.0000 | 1.0 | 0.5 |
| | Test Accuracy | **0.2667** | **0.8145** | **0.7087** |

N.B.: They are simple datasets and plain GCN sometimes can do as good as GSA-GCN -- results are produced by setting the learning rate very small.

# EXPERIMENTS: Graph Classification

Datasets:

- ENZYMES:
  - 600 126-node graphs
  - 18-dim given feature

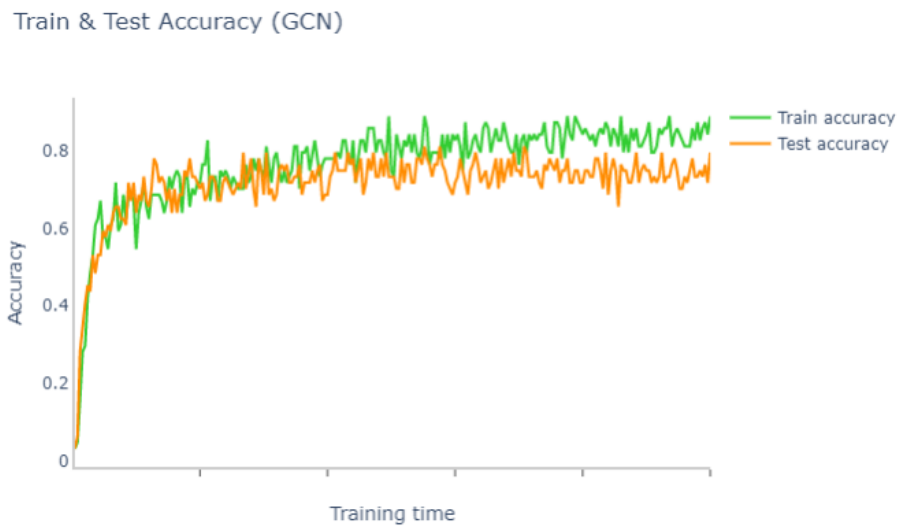
- COIL-RAG:
  - 3900 11-node graphs
  - 29-dim given feature

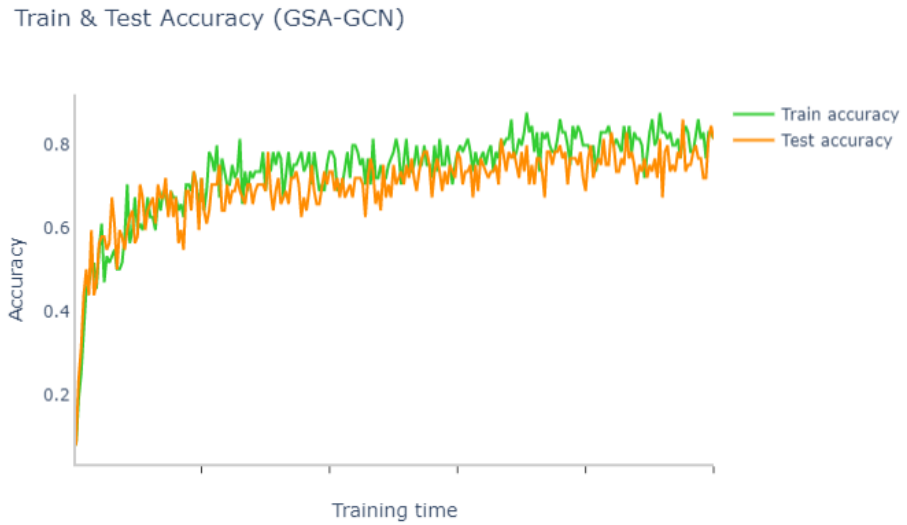
N.B.: Padding with 0s to keep the number of dimensions same

# EXPERIMENTS: Graph Classification

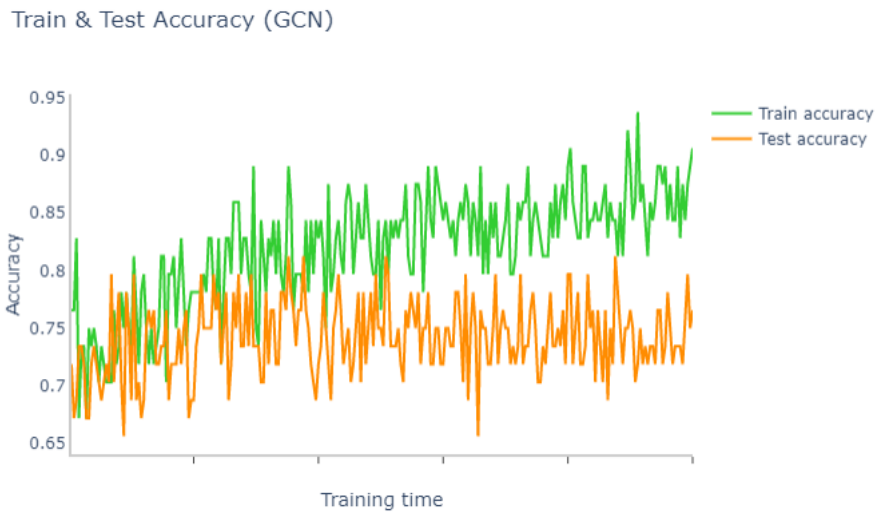
| | Performance | **ENZYMES** | **COIL-RAG** |
|---|---|---|---|
| Plain GCN | Train Accuracy | 0.625 | 0.8281 |
| | Test Accuracy | 0.0156 | 0.7032 |
| GSA GCN | Train Accuracy | 0.1875 | 0.8283 |
| | Test Accuracy | **0.0938** | **0.8435** |

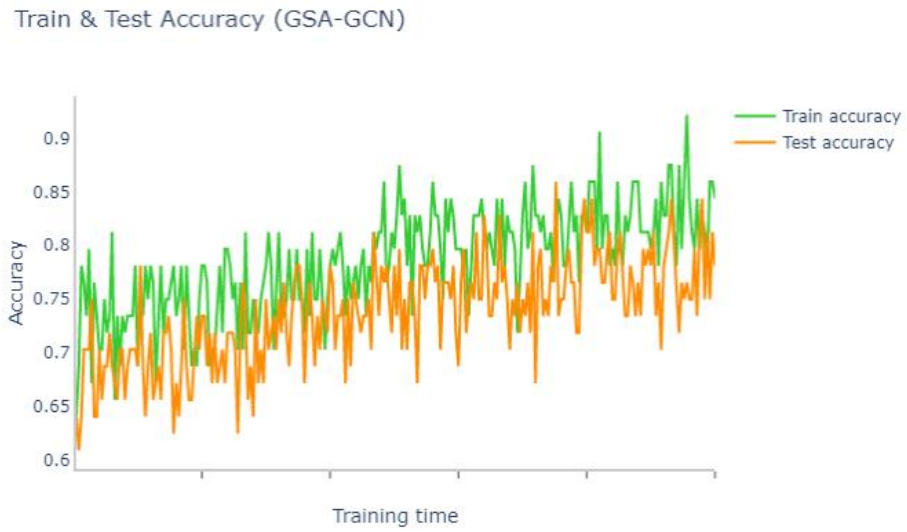
# EXPERIMENTS: Graph Classification



# EXPERIMENTS: Graph Classification



# Graph Classification: Some Discussions

- GSA-GCN generalizes significantly better, especially for the hard-to-generalize cases
- For COIL-RAG dataset, GSA-GCN consistently has test accuracy not worse than train accuracy
- If we plot the curve we can see GSA-GCN prevents overfitting

# Thank You

# REFERENCES


- Kipf, Thomas N., and Welling, Max. "Semi-supervised classification with graph convolutional networks." *arXiv preprint arXiv:1609.02907* (2016).
- Zhang, Han, et al. "Self-attention generative adversarial networks." *arXiv preprint arXiv:1805.08318* (2018).
- Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.
- Wu, Zonghan, et al. "A comprehensive survey on graph neural networks." *arXiv preprint arXiv:1901.00596* (2019).