Semi-Supervised Classification with Graph Convolutional Network

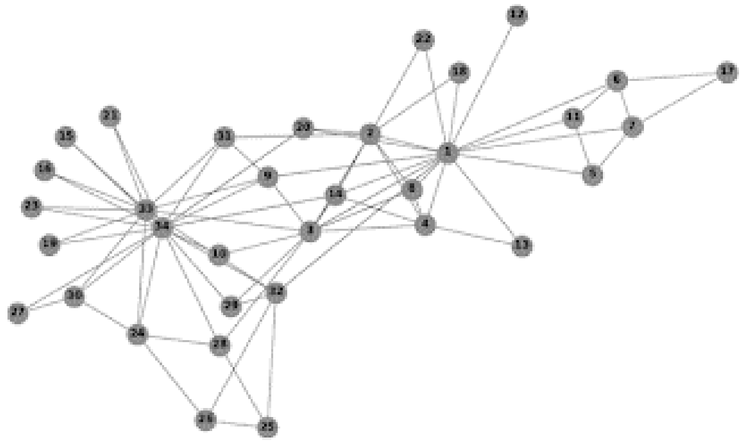
Thomas N. Kipf, Max Welling
ICLR 2017

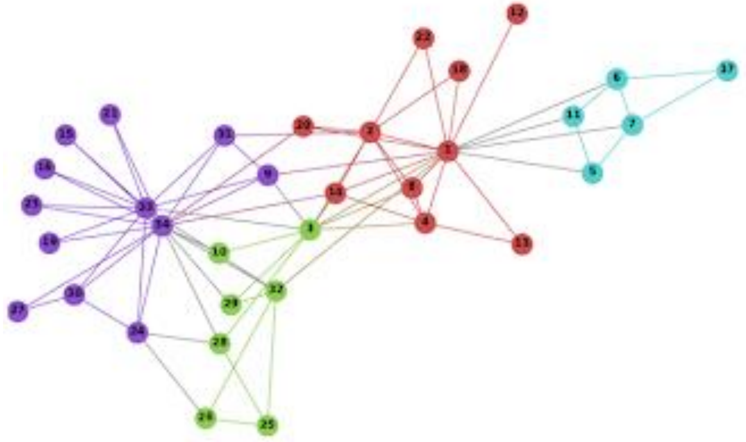
Steven Xia

Introduction

- Graphs contains rich set of information in both nodes and edges
- Example tasks is node classification
- Apply Convolution Filter to graph

$$H^{(l+1)} = f\left(H^{l}, A\right)$$





Approximating Spectral Convolution

- Laplacian and Normalization

$$L = D - A$$

$$L_{sym} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

- Spectral Convolution

$$g_\theta \bigstar x = U g_\theta U^T x$$

Eigenvectors of Laplacian

- First Order Approximation

$$g_\theta x \approx \theta_0 x + \theta_1 (I - L_{sym}) x \approx \theta (I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}) x$$

Convolutional Layer

- Further Approximation

$$g_\theta \star x \approx \theta(I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}})\, x$$

$$\tilde{A} = A + I \quad \text{and} \quad \widetilde{D}_{ii} = \sum_j \tilde{A}_{ij}$$

- Matrix Representation

$$Z = \sigma\,(\widetilde{D}^{-\frac{1}{2}} \tilde{A} \widetilde{D}^{-\frac{1}{2}}\; X\, \Theta)$$

Feature Maps ($N \, X \, F$) → Z

Input signal ($N \, X \, C$) → X

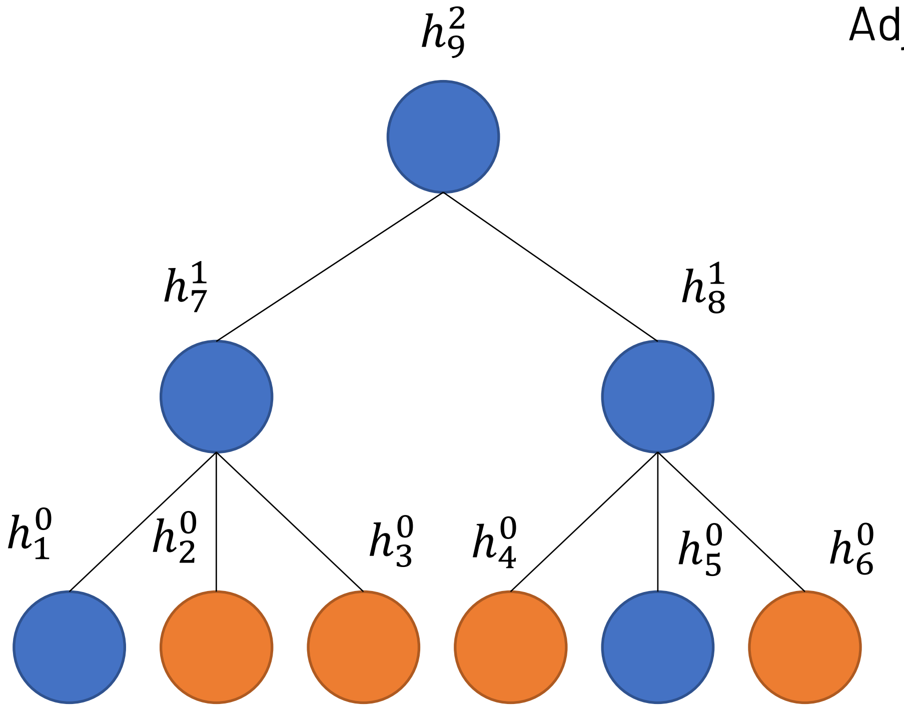
Non-linear Activation → σ

Filter/Kernal (C X F) → Θ

Spatial Interpretation

$$Z = \sigma\left(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}} X\,\Theta\right)$$

Symmetric Normalized
Self Loop
Adjacency Matrix



Normalization
Constant

$$h_i^{\,l+1} = \sigma\Big(\sum_j \frac{1}{\sqrt{\widetilde{D}_{ii}\widetilde{D}_{jj}}} h_j^{\,l} W^l\Big)$$

Sum over all
neighbors + itself

Example: Two Layer GCN Model

- Model

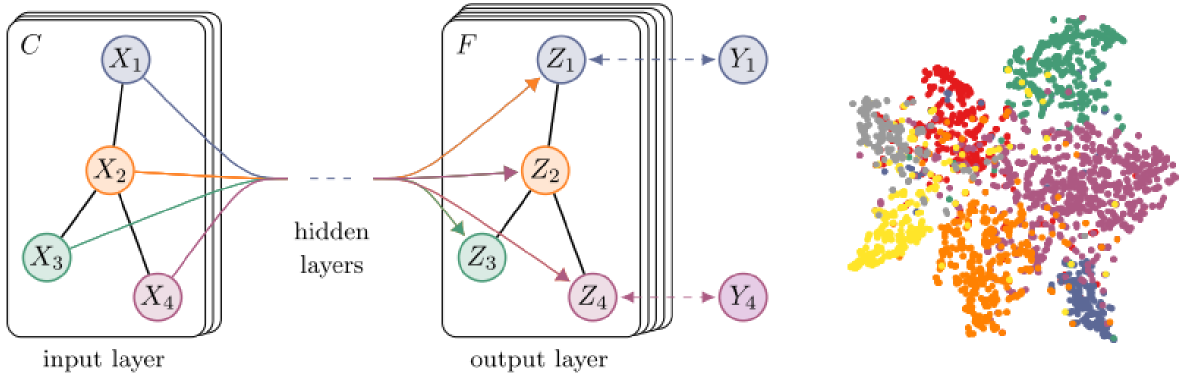
$$\hat{A} = \widetilde{D}^{-\frac{1}{2}} \tilde{A} \widetilde{D}^{-\frac{1}{2}}$$ ← Precomputed

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

↑ Sparse Dense Matrix Multiplication: *O*(*E*)

- Loss Function

$$loss = -\sum_{l \in y} \sum_{f} Y_{lf} \log(Z_{lf})$$



(a) Graph Convolutional Network

(b) Hidden layer activations

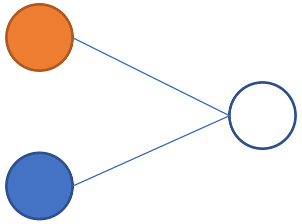
Evaluation

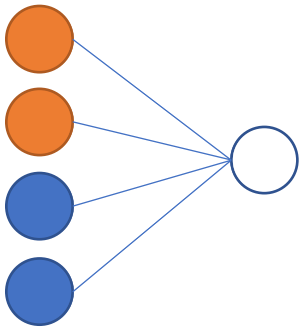
Method	Citeseer	Cora	Pubmed	NELL
ManiReg [3]	60.1	59.5	70.7	21.8
SemiEmb [28]	59.6	59.0	71.1	26.7
LP [32]	45.3	68.0	63.0	26.5
DeepWalk [22]	43.2	67.2	65.3	58.1
ICA [18]	69.1	75.1	73.9	23.1
Planetoid* [29]	64.7 (26s)	75.7 (13s)	77.2 (25s)	61.9 (185s)
GCN (this paper)	**70.3** (7s)	**81.5** (4s)	**79.0** (38s)	**66.0** (48s)

Annotation	Description		Propagation model	Citeseer	Cora	Pubmed
K nodes away →	Chebyshev filter (Eq. 5)	$K=3$	$\sum_{k=0}^{K} T_k(\tilde{L})X\Theta_k$	69.8	79.5	74.4
		$K=2$		69.6	81.2	73.8
	1st-order model (Eq. 6)		$X\Theta_0 + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}X\Theta_1$	68.3	80.0	77.5
	Single parameter (Eq. 7)		$(I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}})X\Theta$	69.3	79.2	77.4
Vanishing Gradient →	**Renormalization trick** (Eq. 8)		$\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X\Theta$	**70.3**	**81.5**	**79.0**
No self loop →	1st-order term only		$D^{-\frac{1}{2}}AD^{-\frac{1}{2}}X\Theta$	68.7	80.5	77.8
	Multi-layer perceptron		$X\Theta$	46.5	55.1	71.4

Limitation and Discussion

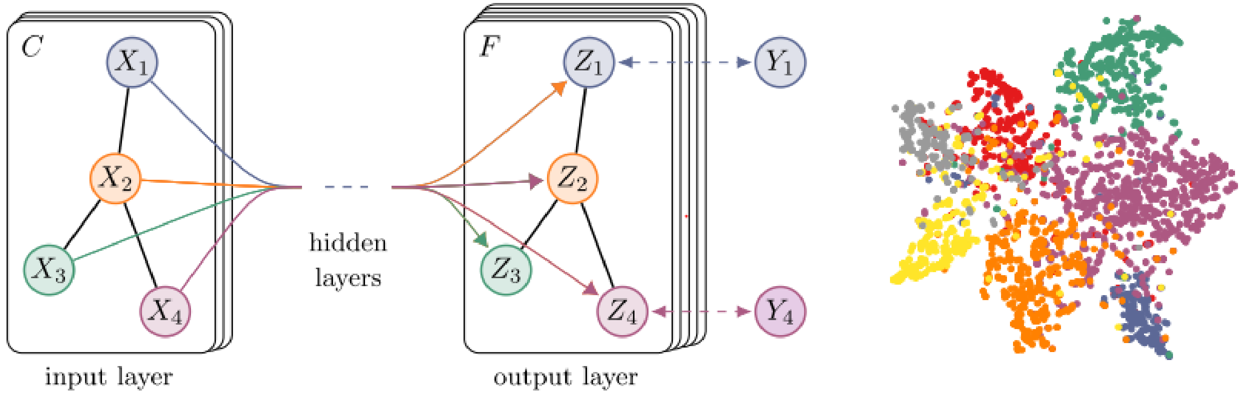
- Spatial Convolution with 1st order approximation in current framework does not support **edge features** and **directed graphs**
- Memory and Computation cost can grow
    - Sparse Dense matrix multiplication relies on sparsely connected graphs
    - Memory grows linearly with size of dataset
- GCN can be viewed as an "average" of neighboring nodes, propagating thru the graph via message passing





Summary

- Fast Approximation of Spectral Convolutions on Graph to provide local spectral filter that is fast to compute
- Stack multiple layers to build a neural network model
- Allows for Semi-Supervised Node Classification via Loss function and differentiable functions



(a) Graph Convolutional Network (b) Hidden layer activations

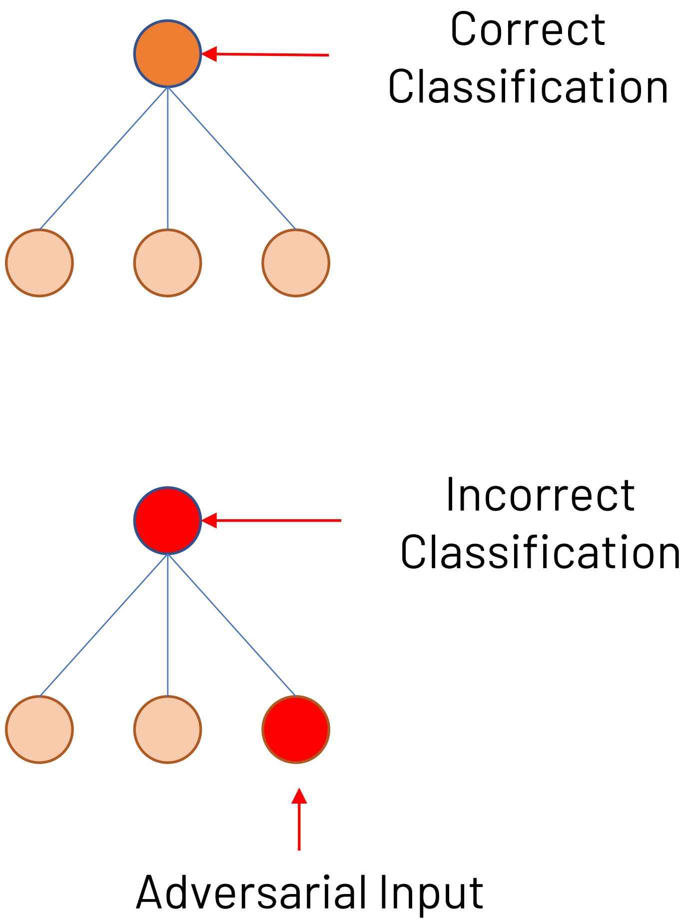
Robust Graph Convolutional Networks Against Adversarial Attacks

Dingyuan Zhu, Ziwei Zhang, Peng Cui, Wenwu Zhu
KDD 2019

Steven Xia

Introduction

- Graphic Convolutional Network (GCN) are vulnerable to adversarial attacks
  - Changing Node Links or Attributes
- How to design a Robust GCN?
  - Limit the effect of adversarial inputs
- Adopt Gaussian Distribution as hidden representation
  - Variance Matrix for attention-mechanism



Gaussian-based Graph Convolution Layer

- Latent Representation is a gaussian distribution

$$h_i^l = N(u_i^l, diag(\sigma_i^l))$$

- Weighted sum of gaussian vector is also gaussian

$$h_{ne(i)}^l \sim N(\sum_{j \in ne(i)} \frac{1}{\sqrt{\widetilde{D}_{ii}\widetilde{D}_{jj}}} u_j^l, diag\left(\sum_{j \in ne(i)} \frac{1}{\widetilde{D}_{ii}\widetilde{D}_{jj}} \sigma_j^l\right))$$

- Apply “Attention” based on variance, larger variance = more uncertainty

Attention Weights

$$\alpha_j^l = \exp(-\gamma \sigma_j^l)$$

$$h_{ne(i)}^l \sim N(\sum_{j \in ne(i)} \frac{u_j^l \odot \alpha_j^l}{\sqrt{\widetilde{D}_{ii}\widetilde{D}_{jj}}}, diag\left(\sum_{j \in ne(i)} \frac{\sigma_j^l \odot \alpha_j^l \odot \alpha_j^l}{\widetilde{D}_{ii}\widetilde{D}_{jj}}\right))$$

Gaussian-based Graph Convolution Layer

- Propagate Mean and Variance Directly by applying weight and non-linearity

$$H^{l+1} = \rho\,(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}H^{l}W^{l})$$

$$M^{l+1} = \rho(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}(M^{l}\odot \mathcal{A}^{l})W_{\mu}{}^{l})$$

$$\Sigma^{l+1} = \rho(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}(\Sigma^{l}\odot \mathcal{A}^{l} \odot \mathcal{A}^{l})W_{\sigma}{}^{l})$$

- Sample From distribution to obtain final layer output with regularize KL divergence for 1st hidden layer

$$\mathcal{L} = \sum_{i=1}^{N} KL\,\left(N\left(u_i^1, diag\left(\sigma_i^1\right)\right)\;||\,N(0, I)\right)$$

Clean Dataset Results

Graph Convolutional Network

Attention based GCN

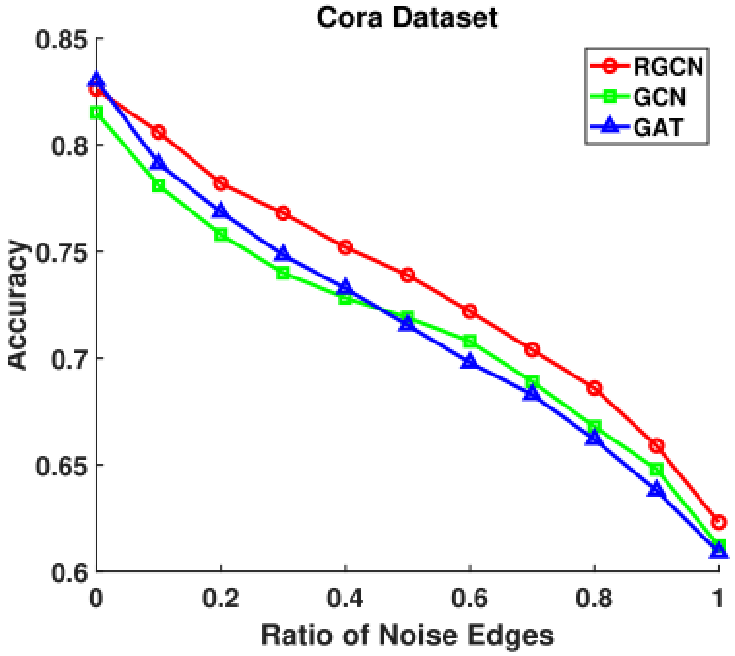
This paper

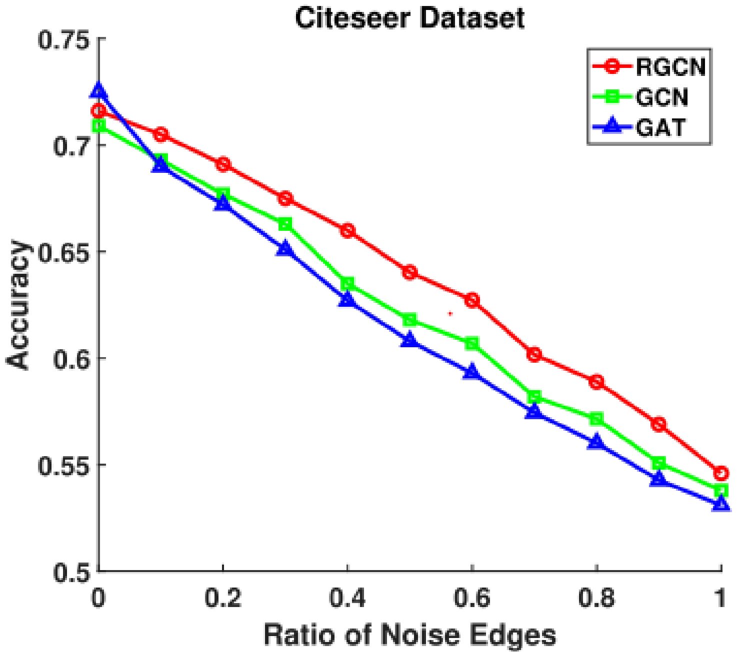
	Cora	Citeseer	Pubmed
GCN	81.5 ± 0.5	70.9 ± 0.5	79.0 ± 0.3
GAT	$\mathbf{83.0 \pm 0.7}$	$\mathbf{72.5 \pm 0.7}$	79.0 ± 0.3
RGCN	82.8 ± 0.6	71.2 ± 0.5	$\mathbf{79.1 \pm 0.3}$

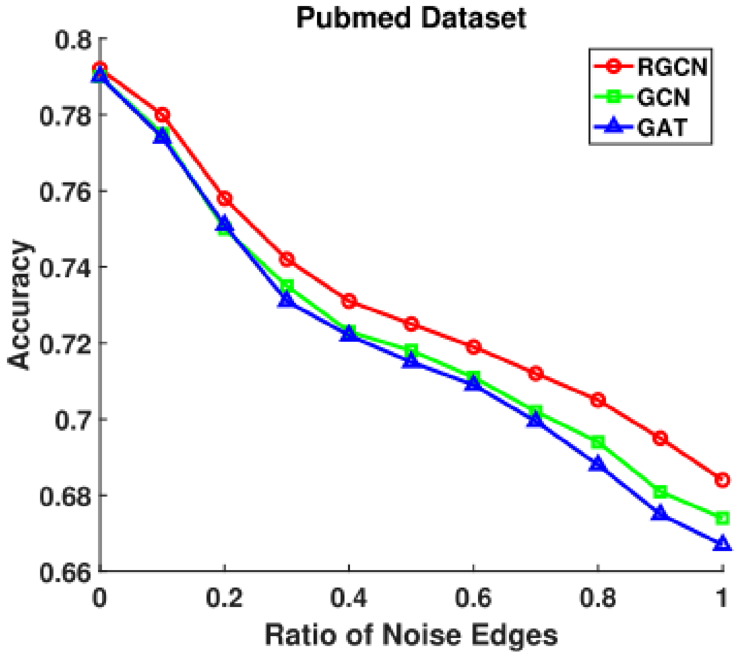
- Similar Effectiveness compared to other approaches under adversarial-free setting

Against Non-Targeted Attack

- Poisoning attack on model by randomly adding edges to training graph

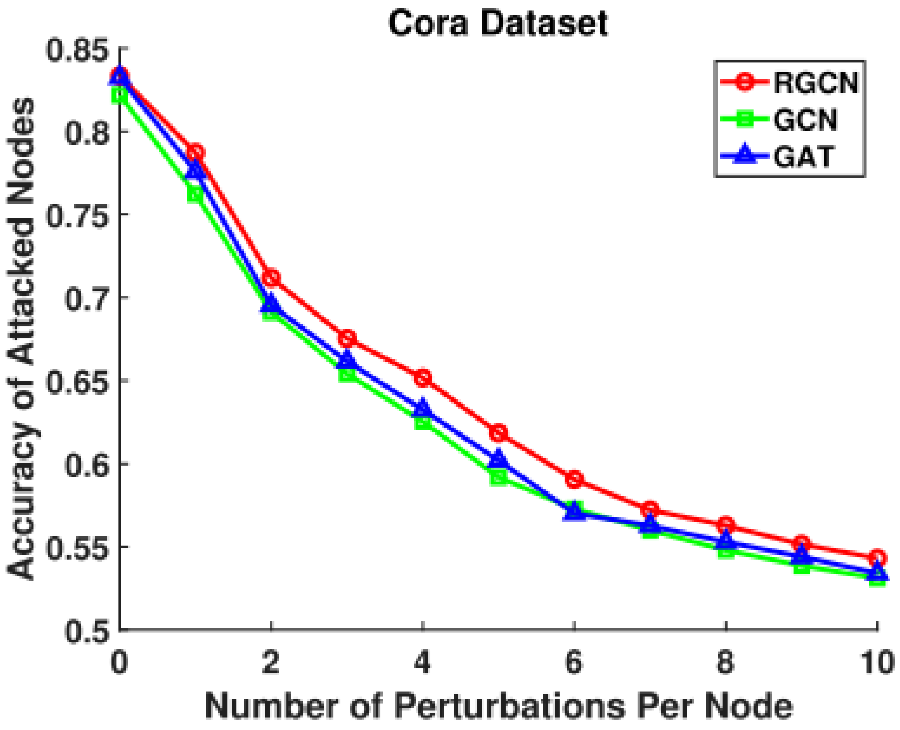




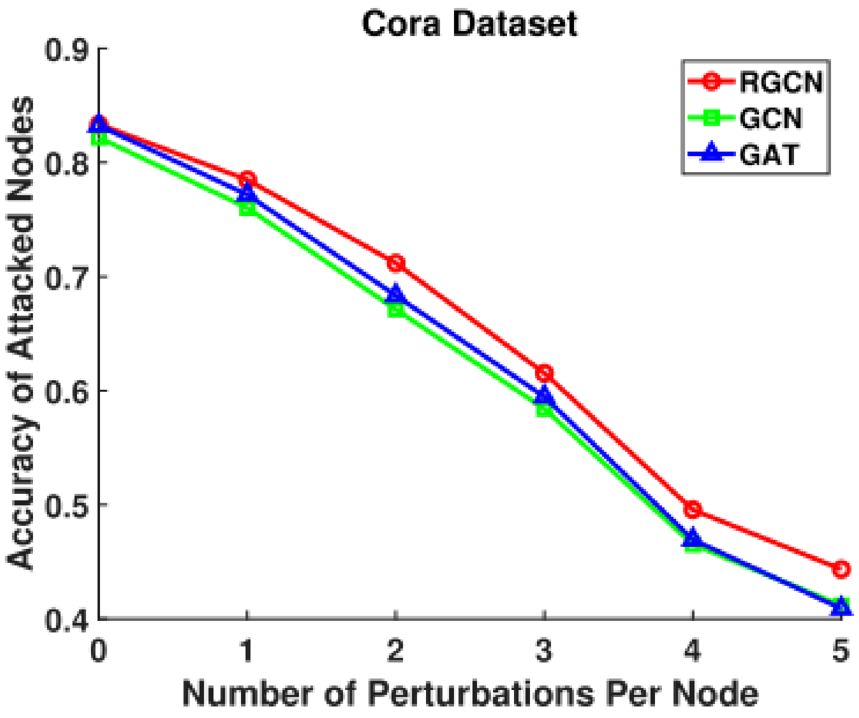


Against Targeted Attacks

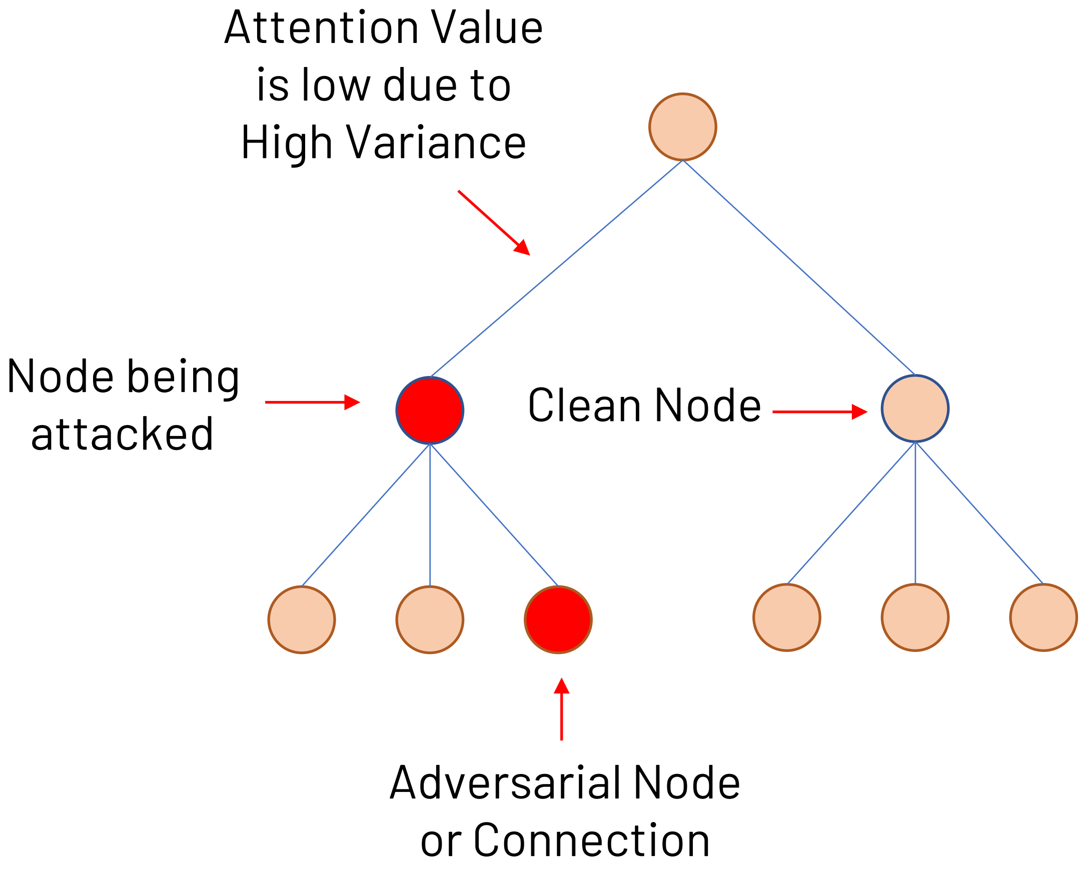
- Target **High Value Nodes** (>10 edges)
  - Evasion Attack



  - Poison Attack



Discussion: Why does Robustness Improve?



- Sampling from Distribution depends on the variance
- "Absorbs" the effect of adversarial input
- High variance stops the propagation of attacked nodes

Summary

- Represent Latent vectors using Gaussian Distribution and final output is sampled by distribution

- Compute and Propagate Mean and Variance matrix instead of hidden representations

- Improve Robustness of GCN by reducing impact of adversarial attacks using sampling and variance attention weights