Final Project Team 10 - KDD 2020 Graph Structure Learning for Robust Graph Neural Networks

tags: GNN

姜林寬 109065510 鄭宏彬 109062657
paper link : https://arxiv.org/abs/2005.10203 (https://arxiv.org/abs/2005.10203)

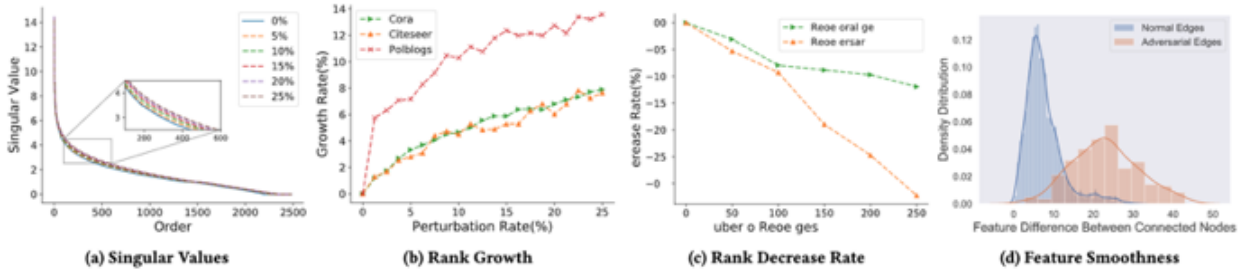
Introduction

在防禦 Graph Adversarial Attack 的研究中，主要有兩個研究方向，一個是加入 Attention 機制，設法降低被擾動邊的 Attention 以降低攻擊影響，另一種是使用 Graph Structure Learning (GSL) 去學習 Graph 正常結構，找出不合理的邊，這篇論文的研究方向是後者，這篇論文的主要貢獻是提出了 Pro-GNN 架構，他可以在各種不同的 Adversarial Attack 下學習出合理的圖架構，邊還原邊訓練 GCN 來得到一個較不受攻擊影響效果的網路。

Perturbation Properties

作者觀察到被攻擊的圖與正常的圖有以下差異，這裡用metattack為例：

- metattack enlarges the singular values of the adjacency matrix.
- metattack quickly increases the rank of adjacency matrix.
- removing adversarial edges reduces the rank faster than removing normal edges.
- metattack tends to connect nodes with large feature difference.

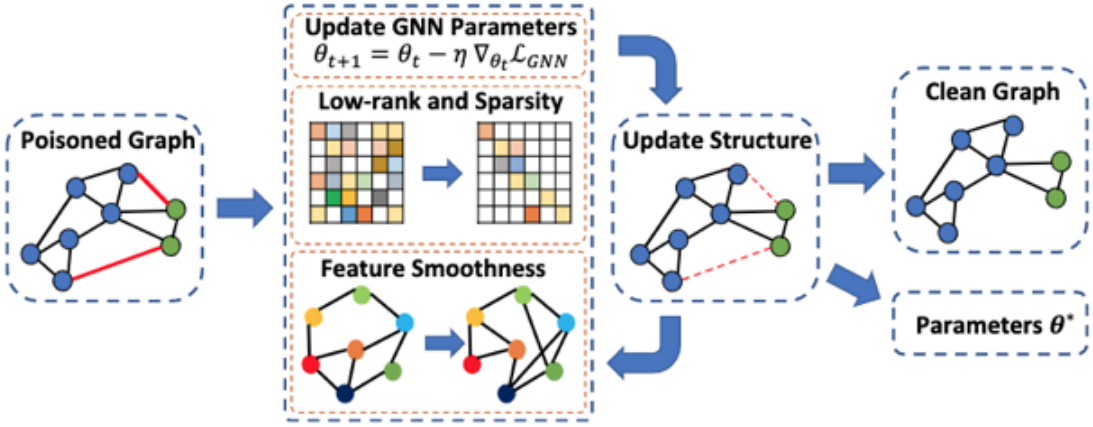


(a) Singular Values (b) Rank Growth (c) Rank Decrease Rate (d) Feature Smoothness

Framework

Pro-GNN Architecture

以下是作者提出的 Pro-GNN 架構，對於一張被擾動的 Graph，先根據前面發現的一些擾動圖會有的特徵算梯度，對目前的 Adjacency matrix 更新，再算新結構的 GNN parameters，最後會得到一張乾淨的圖和已經訓練好的GNN模型，這種方式訓練 GNN 雖然在原本就沒受攻擊的 Graph 時犧牲了一些準確率，但換到的是對於受攻擊過的 Graph 有更好的魯棒性。



Formula

以下公式分成了三部分：

- ||A – S||^2 是要限制每一階段還原的Graph不會跟原來的圖差太多，S代表還原後的adjacency matrix，A代表一開始擾動後的 adjacency matrix
- 加入 ℓ1 norm 是為了讓還原的圖維持 sparse 的特性
- 加入 nuclear norm 是為了讓還原的圖維持維持 low rank 的特性

$$\arg\min_{\mathbf{S}\in\mathcal{S}} \mathcal{L}_0 = \|\mathbf{A}-\mathbf{S}\|_F^2 + \alpha\|\mathbf{S}\|_1 + \beta\|\mathbf{S}\|_*, \; s.t., \mathbf{S} = \mathbf{S}^\top, \quad (4)$$

由於正常 edge 兩端的特徵通常不會相差太大，而攻擊者傾向在兩個 feature 相差大的 node 間加上 edge，所以把 feature difference 的特性也加入 Loss，對所有 edges 兩端點 feature xi 和 xj 取差值平方和，由於是無向圖，所以前面乘以 1/2 。

$$\mathcal{L}_s = \frac{1}{2}\sum_{i,j=1}^{N} \mathbf{S}_{ij}(\mathbf{x}_i - \mathbf{x}_j)^2,$$

對 feature difference 做 normalize ，除以 edge 兩端 nodes 的 degree 數。

$$\mathcal{L}_s = tr(\mathbf{X}^T\hat{\mathbf{L}}\mathbf{X}) = \frac{1}{2}\sum_{i,j=1}^{N} \mathbf{S}_{ij}\left(\frac{\mathbf{x}_i}{\sqrt{d_i}} - \frac{\mathbf{x}_j}{\sqrt{d_j}}\right)^2$$

把上面提到的所有Loss加起來作為 Pro-GNN 的總體 Loss ，這裡多了一個 L_GNN 是 GNN 本身的 Loss ，目的是要在還原 Graph 的同時也對 GNN 做訓練，所以把他的 Loss 也加進來，會在訓練時與代表圖結構的其他 Loss 交互作更新以降低整體 Loss 。另外，加上了四個超參數 alpha, beta, gamma, lambda 作為四個 Loss 的影響程度。

$$\arg\min_{\mathbf{S}\in\mathcal{S},\theta} \mathcal{L} = \mathcal{L}_0 + \lambda\mathcal{L}_s + \gamma\mathcal{L}_{GNN} \quad (9)$$

$$= \|\mathbf{A}-\mathbf{S}\|_F^2 + \alpha\|\mathbf{S}\|_1 + \beta\|\mathbf{S}\|_* + \gamma\mathcal{L}_{GNN}(\theta, \mathbf{S}, \mathbf{X}, \mathcal{Y}_L) + \lambda tr(\mathbf{X}^T\hat{\mathbf{L}}\mathbf{X})$$

$$s.t. \qquad \mathbf{S} = \mathbf{S}^\top,$$

Pro-GNN Algorithm

以下是 Pro-GNN 的演算法，不論收到的 Graph 是否遭受攻擊，都當成是被擾動過的 adjacency matrix A，接著 initialize S 和 GNN parameters ， S 是當前迴圈所最新還原的 Graph adjacency matrix ，第四行到第六行都是依前面提到的性質分別算出梯度對目前的 adjacency matrix S 做更新，第七行的 P 是一個 Projection function ，因為前面更新完後可能會讓 S 超出 0~1 的範圍，所以用 P 來限制範圍在 0~1，在更新 S 的時候，完全沒有訓練 GNN ， 等到結構更新完後，第八行到第十行再用 GCN train ，在 train GCN 的過程中結構是固定的，整個 while 迴圈就是不斷固定結構或 GNN parameters 其中一邊更新另外一邊交互進行。

Algorithm 1: Pro-GNN

Data: Adjacency matrix $\mathbf{A}$, Attribute matrix $\mathbf{X}$, Labels $\mathcal{Y}_L$, Hyper-parameters $\alpha, \beta, \gamma, \lambda, \tau$, Learning rate η, η'

Result: Learned adjacency $\mathbf{S}$, GNN parameters θ

1 Initialize $\mathbf{S} \leftarrow \mathbf{A}$

2 Randomly initialize θ

3 **while** *Stopping condition is not met* **do**

4 $\quad \mathbf{S} \leftarrow \mathbf{S} - \eta \nabla_S (\|\mathbf{S} - \mathbf{A}\|_F^2 + \gamma \mathcal{L}_{GNN} + \lambda \mathcal{L}_s)$

5 $\quad \mathbf{S} \leftarrow \text{prox}_{\eta\beta||.||_*}(\mathbf{S})$

6 $\quad \mathbf{S} \leftarrow \text{prox}_{\eta\alpha||.||_1}(\mathbf{S})$

7 $\quad \mathbf{S} \leftarrow P_{\mathcal{S}}(\mathbf{S})$

8 $\quad$ **for** *i=1 to* τ **do**

9 $\quad\quad g \leftarrow \frac{\partial \mathcal{L}_{GNN}(\theta, \mathbf{S}, \mathbf{X}, \mathcal{Y}_L)}{\partial \theta}$

10 $\quad\quad \theta \leftarrow \theta - \eta' g$

11 **Return** $\mathbf{S}, \theta$

Original Paper Experiment

原論文中實驗所使用的四個 benchmark dataset為 Cora、Citeseer、Pubmed 以及 Polblogs ，作者利用這些資料集來進行 Adversarial attack 及每個 state-of-the-art 的 Defense methods 之有效性驗證。

對於每個圖隨機選擇 10% 的節點進行訓練，10% 的節點進行驗證，其餘 80% 的節點進行測試。首先作者重現所有 state-of-the-art 的實驗用來與 Pro-GNN 在防禦對抗式攻擊的表現做比較，而所有 SOTA 模型的 hyperparameters 都根據驗證集的 loss 和 accuracy 進行調整，其餘模型之參數則採用原作者實作時的默認參數。

作者額外設定一個變形的 Pro-GNN-fs，它是透過不考慮 feature smoothness（設定 λ = 0）而得到的變形。是為了 Polblogs 資料集所使用，因為這個資料集當中並沒有 Node features。

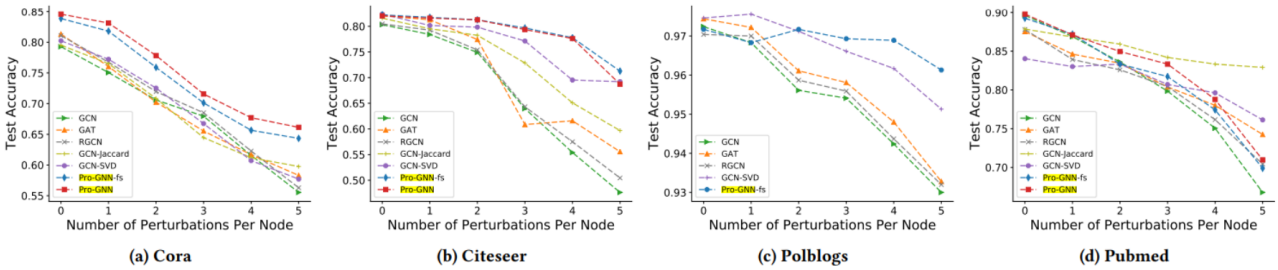
下圖是作者所使用四個資料集的基本資訊：

	|V|	|E|	N_{LCC}	E_{LCC}	Classes	Features
Cora	2,708	5,429	2,485	5,069	7	1,433
Citeseer	3,312	4,732	2,110	3,668	6	3,703
Polblogs	1,490	19,025	1,222	16,714	2	/
Pubmed	19,717	44,338	19,717	44,338	3	500

實驗一：Pro-GNN在防禦對抗式攻擊的表現

The performance of Pro-GNN under targeted attack

下圖顯示各個防禦演算法在受到nettack攻擊之下的表現，例如受到nettack攻擊的Citeseer資料集上(在每個node有5個budget的條件)，Pro-GNN將基本的GCN準確率提高了 23%，並且也優於其他防禦方法，顯示Pro-GNN的方法可以抵抗有針對性的對抗式攻擊。



(a) Cora (b) Citeseer (c) Polblogs (d) Pubmed

Figure 3: Results of different models under *nettack*

The performance of Pro-GNN under non-targeted attack

下圖顯示Pro-GNN在防禦non-targeted attack的結果優於其他防禦演算法。

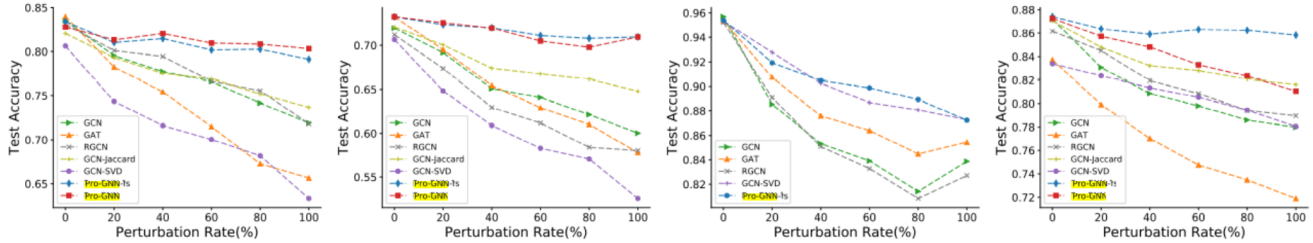
Table 2: Node classification performance (Accuracy±Std) under non-targeted attack (*metattack*).

Dataset	Ptb Rate (%)	GCN	GAT	RGCN	GCN-Jaccard[2]	GCN-SVD	Pro-GNN-fs	Pro-GNN[3]
Cora	0	83.50±0.44	**83.97±0.65**	83.09±0.44	82.05±0.51	80.63±0.45	83.42±0.52	82.98±0.23
	5	76.55±0.79	80.44±0.74	77.42±0.39	79.13±0.59	78.39±0.54	**82.78±0.39**	82.27±0.45
	10	70.39±1.28	75.61±0.59	72.22±0.38	75.16±0.76	71.47±0.83	77.91±0.86	**79.03±0.59**
	15	65.10±0.71	69.78±1.28	66.82±0.39	71.03±0.64	66.69±1.18	76.01±1.12	**76.40±1.27**
	20	59.56±2.72	59.94±0.92	59.27±0.37	65.71±0.89	58.94±1.13	68.78±5.84	**73.32±1.56**
	25	47.53±1.96	54.78±0.74	50.51±0.78	60.82±1.08	52.06±1.19	56.54±2.58	**69.72±1.69**
Citeseer	0	71.96±0.55	73.26±0.83	71.20±0.83	72.10±0.63	70.65±0.32	73.26±0.38	**73.28±0.69**
	5	70.88±0.62	72.89±0.83	70.50±0.43	70.51±0.97	68.84±0.72	**73.09±0.34**	72.93±0.57
	10	67.55±0.89	70.63±0.48	67.71±0.30	69.54±0.56	68.87±0.62	72.43±0.52	**72.51±0.75**
	15	64.52±1.11	69.02±1.09	65.69±0.37	65.95±0.94	63.26±0.96	70.82±0.87	**72.03±1.11**
	20	62.03±3.49	61.04±1.52	62.49±1.22	59.30±1.40	58.55±1.09	66.19±2.38	**70.02±2.28**
	25	56.94±2.09	61.85±1.12	55.35±0.66	59.89±1.47	57.18±1.87	66.40±2.57	**68.95±2.78**
Polblogs	0	**95.69±0.38**	95.35±0.20	95.22±0.14	-	95.31±0.18	93.20±0.64	-
	5	73.07±0.80	83.69±1.45	74.34±0.19	-	89.09±0.22	**93.29±0.18**	-
	10	70.72±1.13	76.32±0.85	71.04±0.34	-	81.24±0.49	**89.42±1.09**	-
	15	64.96±1.91	68.80±1.14	67.28±0.38	-	68.10±3.73	**86.04±2.21**	-
	20	51.27±1.23	51.50±1.63	59.89±0.34	-	57.33±3.15	**79.56±5.68**	-
	25	49.23±1.36	51.19±1.49	56.02±0.56	-	48.66±9.93	**63.18±4.40**	-
Pubmed	0	87.19±0.09	83.73±0.40	86.16±0.18	87.06±0.06	83.44±0.21	**87.33±0.18**	87.26±0.23
	5	83.09±0.13	78.00±0.44	81.08±0.20	86.39±0.06	83.41±0.15	**87.25±0.09**	87.23±0.13
	10	81.21±0.09	74.93±0.38	77.51±0.27	85.70±0.07	83.27±0.21	**87.25±0.09**	87.21±0.13
	15	78.66±0.12	71.13±0.51	73.91±0.25	84.76±0.08	83.10±0.18	**87.20±0.09**	87.20±0.15
	20	77.35±0.19	68.21±0.96	71.18±0.31	83.88±0.05	83.01±0.22	87.09±0.10	**87.15±0.15**
	25	75.50±0.17	65.41±0.77	67.95±0.15	83.66±0.06	82.72±0.18	86.71±0.09	**86.76±0.19**

[1] [2] JaccardGCN and Pro-GNN cannot be directly applied to datasets where node features are not available.

The performance of Pro-GNN under random attack

下圖顯示Pro-GNN在防禦random attack的表現，可以看到在每個資料集中也都優於其他防禦演算法。



(a) Cora (b) Citeseer (c) Polblogs (d) Pubmed

Figure 4: Results of different models under random attack

實驗二：說明Pro-GNN學習圖結構資訊的重要性

Table 3: Node classification accuracy given the graph under 25% perturbation by *metattack*.

	GCN	GCN-NoGraph	Pro-GNN
Cora	47.53±1.96	62.12±1.55	**69.72±1.69**
Citeseer	56.94±2.09	63.75±3.23	**68.95±2.78**
Polblogs	49.23±1.36	51.79±0.62	**63.18±4.40**
Pubmed	75.50±0.17	84.14±0.11	**86.86±0.19**

可以觀察到在受到25% non-targeted attack的嚴重干擾之下，基本的GCN準確率下降很多，因此如果將Graph structure移出GCN model(鄰接矩陣變成0的情況)，僅利用Node features做預測，會有比較好的準確率。

我們可以注意到 Pro-GNN 比 GCN-NoGraph 的結果還要更好。根據這一個結果顯示，即使圖受到嚴重干擾，Pro-GNN 也可以學習到有用的圖結構資訊。

在此作者進一步利用一個實驗說明學習圖結構的重要性， One-stage 是 Pro-GNN 演算法提到的邊訓練圖結構邊訓練 GNN 模型的作法，Two-stage是先將圖還原成clean後再基於 clean graph 訓練 GNN 模型，作者觀察到雖然使用 Two-stages 的 Pro-GNN-two 在擾動率大的擾動圖效果較原本演算法使用 One-Stage 的 Pro-GNN 好，但擾動率小的擾動圖則還是Pro-GNN效果較好

Table 4: Classification performance of Pro-GNN-two and Pro-GNN on Cora dataset

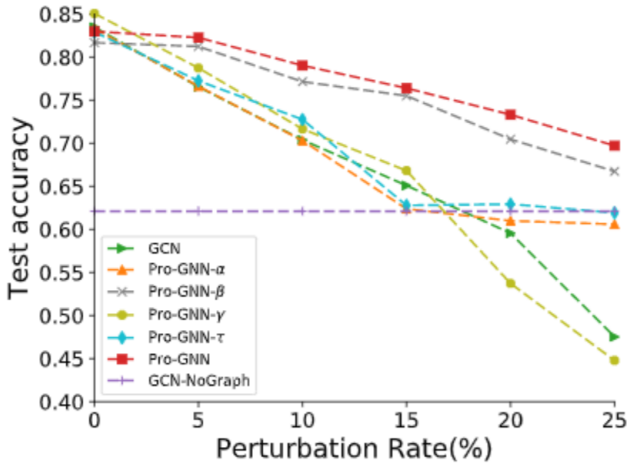
Ptb Rate (%)	0	5	10	15	20	25
Pro-GNN-two	73.31±0.71	73.70±1.02	73.69±0.81	75.38±1.10	73.22±1.08	**70.57±0.61**
Pro-GNN	**82.98±0.23**	**82.27±0.45**	**79.03±0.59**	**76.40±1.27**	**73.32±1.56**	69.72±1.69

實驗三：利用控制實驗變量的方式來比較參數對結果的影響

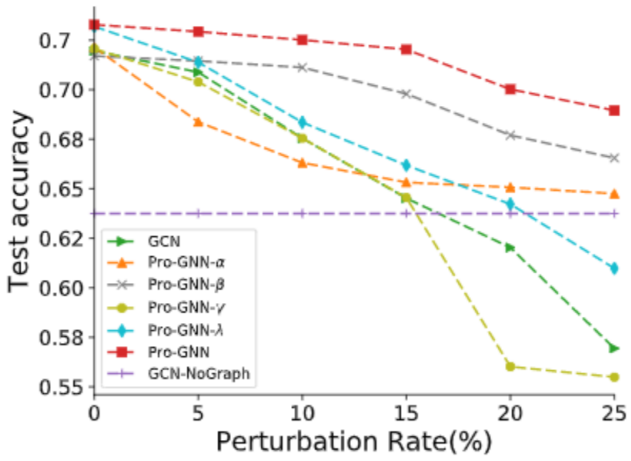
在Pro-GNN架構中我們也提到根據調整四個超參數，可以調整各特性對整體Loss的影響。而 α β γ λ 分別代表以下這四個特性，在目標函數中對這四個特性都各有一個超參數在控制，作者將其中一個設為1、其他設為0來觀察各特性對模型效果的影響，得出以下結論：

- Sparsity(|S|1)：擾動率小時對模型效果沒什麼提升，與一般的GNN差不多，但是擾動率大時效果提升逐漸加大
- low rank(|S|*)：與feature smoothness是模型效果的主要貢獻，在低擾動率與高擾動率都有不錯的效果

- Graph Loss：沒有太大的差別，因為本身GNN就只有這項
- feature smoothness：與low rank影響的效果差不多



(a) Cora



(b) Citeseer

Paper Summary

- GNN容易受到對抗式攻擊所影響準確率，本篇論文提出的Pro-GNN能在不知道圖受到何種攻擊的情況下有效的提升節點分類的準確率，實驗證實在低擾動率與高擾動率下都有不錯的效果。
- 提出了正常圖會有的幾個特性並對限制圖符合該特性的公式做了討論。
- 透過實驗說明了同時學習圖結構資訊與GNN參數可以有效防禦對抗式攻擊。

Paper Weaknesses

1. 在比較one-stage與two-stages的實驗中，可觀察到two-stages在高擾動圖的防禦上有較好的效果，作者還是得出了one-stage比較好的結論，這部分需要更多的實驗來支持。
2. 由於Non-targeted attack的目標是要降低整體GNN的節點分類準確率，其擾動的邊會散布在整個Graph上，我們可以預期在現實世界的大圖上，執行Pro-GNN的方法來重建Graph以防禦metattack需要花費龐大時間。統計顯示由於metattack會傾向於連接兩個特徵不相干的節點，因此我們針對這個攻擊特性引入Overlapping Community Detection Approach，針對整個Graph先做前處理，將Graph做一次分群，利用其演算法時間複雜度低的優勢來輔助大圖結構的去噪還原。

Ours Achievements

Task1 : Pro-GNN vs Pro-GNN-two

Pro-GNN 會在調整結構與學習參數之間交互進行，作者在原論文中提到過 one-stage 和 two-stages 的比較，從實驗結果來看 one-stage 在低擾動率的 Graph 中有較好的效果，而我們重做這個實驗後發現，即使在高擾動率的 Graph 下，one-stage 與 two-stages 的效果也不會差太多，以攻擊方的角度來看，有讓攻擊不被發現的 issue 在，所以擾動率不會到非常大，即使提高攻擊預算做擾動，one-stage也能發揮效果，所以我們認同作者的結論， Pro-GNN 應該採用 one-stage。

Task 2 : Inner-steps vs Outer-steps

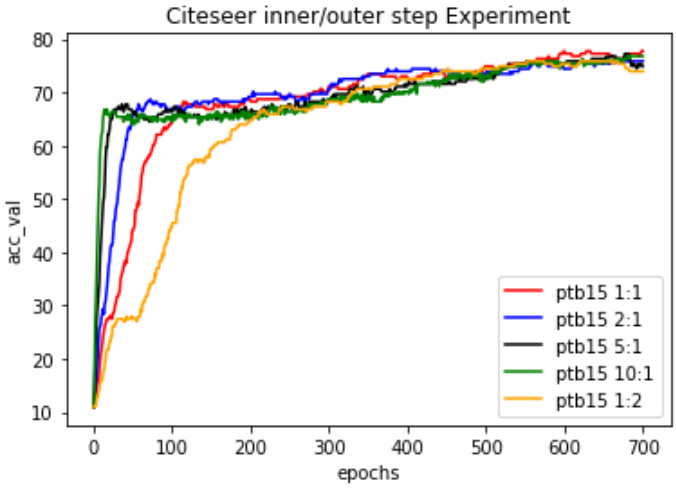
原論文的實驗中，作者把演算法還原架構的部分做到收斂再訓練 GNN 參數的方法稱為 two-stages，透過實驗說明 one-stage 會比 two-stages 好，而我們透過實驗也認同這個結果。two-stage 直接對還原後的圖做訓練，會比還原一點就訓練再去還原更有效率，基於這個特性延伸出了一個議題，我們想試著找到一個可以兼顧 one-stage 的效果及 two-stages 的效率 (先把圖還原再一次訓練) 的方法來改善 Pro-GNN 的演算法，定義Inner-steps指得是每次epoch中對結構還原的次數，Outer-steps指得是每次epoch中對目前結構train gcn的次數，假設**Inner-steps = x1, Outer-steps = x2, epoch = 100**，代表的是每次epoch都對被擾動圖做 x1 次還原，在對還原後的graph訓練 x2 次，從 **One-Stage vs Two-Stages** 的實驗中知道如果先把圖結構都還原再訓練 (two stages)，效果是沒有 one stage 好的，但我們很好奇一次還原大量的結構是否也會得到相同的結論，於是設計了以下實驗：

Citeseer

這裡對 Citeseer 資料集使用 meta attack 、擾動率 15 % 的圖做實驗，比較inner-steps 和 outer-steps 的比例對正確率的影響， inner-steps 與 outer-steps 分別用 (1,1), (2,1), (5,1), (10,1), (1,2) 的參數組合來實驗，從結果可以看出當 inner-steps 相對於 outer-steps 的比例越高，得到的 test accuracy 會些微提升。

Dataset	ptb rate	Inner-steps	Outer-steps	acc_val (%)	acc_test (%)
Citeseer	15 %	1	2	75.83	70.56
Citeseer	15 %	1	1	77.73	71.27
Citeseer	15 %	2	1	75.83	71.68
Citeseer	15 %	5	1	77.25	72.63
Citeseer	15 %	10	1	77.25	72.75

下圖為以上五個實驗設定的 val accuracy 變化，可以看到當 inner-steps 相對 outer-steps 比例增加時，收斂的速度會變快。

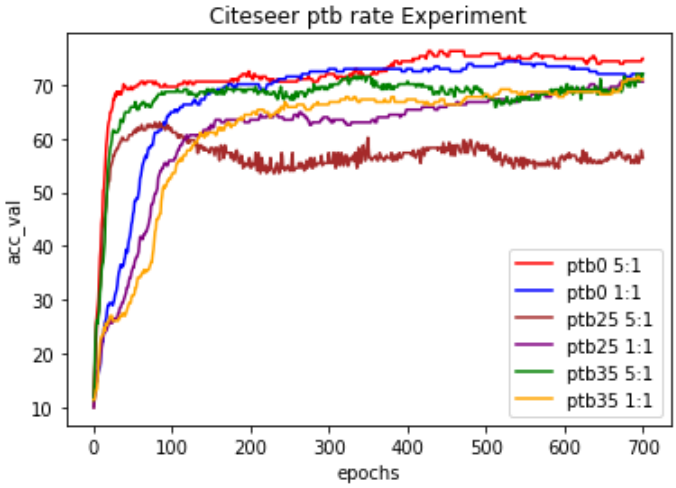


我們也比較了不同擾動率下，inner-steps 比例的提高對結果的影響，以下是分別在 0%, 5%, 15%, 25%, 35% 的實驗數據，我們發現 inner-steps 多做幾次除了幫助收斂外，也能得到較好的效果，在高擾動率的情況下， inner-steps 較低的設定使結構能慢慢收斂，可以得到更好的效果，但由於高擾

動率本身就會收斂得較慢，在效果差不多的情況下，我們認為提高 inner-steps 減少訓練時間是個值得採用的方法。

Dataset	ptb rate	Inner-steps	Outer-steps	acc_val (%)	acc_test (%)
Citeseer	0 %	1	1	74.41	72.33
Citeseer	0 %	5	1	76.30	74.76
Citeseer	5 %	1	1	76.30	71.92
Citeseer	5 %	5	1	74.41	74.17
Citeseer	15 %	1	1	77.73	71.27
Citeseer	15 %	5	1	77.25	72.63
Citeseer	25 %	1	1	75.83	70.73
Citeseer	25 %	5	1	68.72	68.42
Citeseer	35 %	1	1	73.46	69.73
Citeseer	35 %	5	1	73.46	68.60

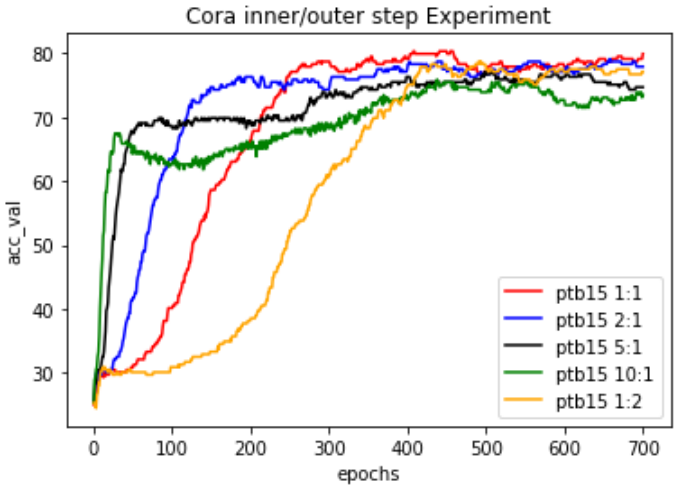
以下為用擾動率 = 0%, 25%, 35% 的實驗結果圖，可以看到在不同的擾動率下，以 (inner-steps, outer-steps) = (5,1) 的收斂速度比 (1,1) 快很多，即使是在高擾動率下也有明顯差別，下圖中最慢收斂的三個分別是擾動率 0%, 25%, 35% 的 (1,1)設定。



Cora

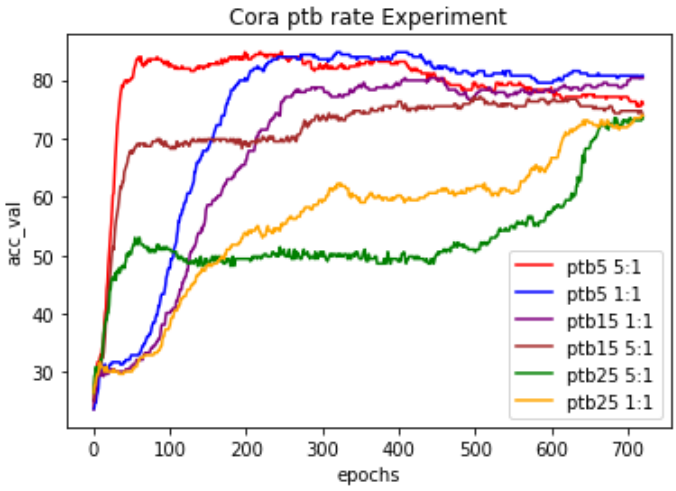
我們也對 Cora 資料集做相同實驗，得到的數據如下，跟 citeseer 不同的是，隨著 inner-steps 增加， test_val 會下降。

Dataset	ptb rate	Inner-steps	Outer-steps	acc_val (%)	test_val (%)
Cora	15 %	1	2	78.71	73.54
Cora	15 %	1	1	80.32	78.22
Cora	15 %	2	1	79.12	76.06
Cora	15 %	5	1	77.11	76.06
Cora	15 %	10	1	75.90	73.14



我們也用 cora 做了不同擾動率下，inner-steps 比例的提高對結果的影響，以下是分別在 5%, 15%, 25% 的實驗數據，跟 citeseer 不同的是， inner-step 提高的效果會比不上原來的效果，隨著當擾動率提升，他們之間的差距會越來越大。

Dataset	ptb rate	Inner-steps	Outer-steps	acc_val (%)	test_val (%)
Cora	5 %	1	1	84.74	82.19
Cora	5 %	5	1	84.74	82.95
Cora	15 %	1	1	80.32	78.22
Cora	15 %	5	1	77.11	76.06
Cora	25 %	1	1	77.11	70.17
Cora	25 %	5	1	73.90	67.91



從以上兩個 dataset 的實驗發現，提高 inner-steps 比例是個可行的方法，我們可以把多次結構更新視為一次大的結構更新，但這不等於提高 learning rate，它會是多個結構變動的加總而非單個結構變動的增幅，從數據來看，提高 inner-steps 除了訓練速度可以得到顯著的縮短，也能不失去訓練效果甚至在某些資料集反而得到更好的效果，同樣印證了原作者提出的訓練圖架構應該與訓練 GNN 參數同時進行，是值得嘗試的策略。至於兩個資料集會得到不一樣的結果，可能跟 dataset 本身的特性有關，可能要再多對 dataset 本身做研究進一步釐清資料集的差異。

Task 3 : Overlapping Community Detection Approach

首先我們對一張圖施加具有代表性的Non-targeted attack, metattack，由於Non-targeted attack的目標是要降低整體GNN的節點分類準確率，擾動的邊會散落在整個Graph上，我們可以預期在現實世界的大圖上，執行ProGNN的方法來重建Graph以防禦metattack需要花費龐大時間。統計顯示由於metattack會傾向於連接兩個特徵不相干的節點，因此我們針對這個攻擊特性引入Community Detection Algorithm, BigCLAM。
針對整個Graph先做前處理，將Graph做一次分群，利用其演算法時間複雜度較低的優勢，我們運用其還原受擾動的大圖結構。首先對整個圖做還原得到labels，將還原出來的clean labels放到Pro-GNN中做節點預測，驗證這個方法可以有效的防禦Non-targeted attack。

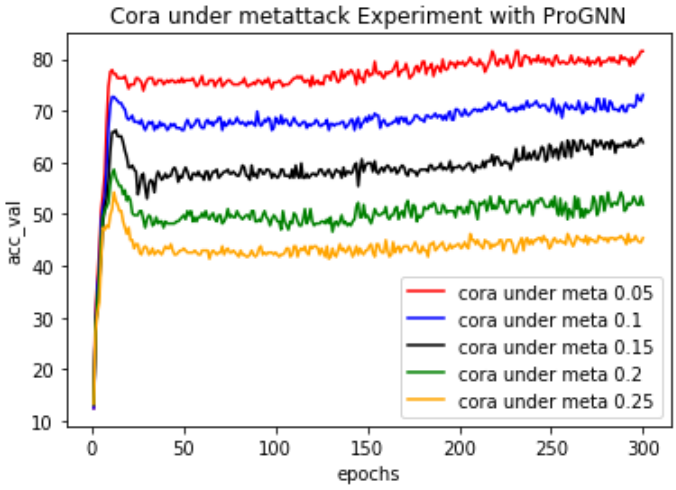
Attack type : metattack

這裡使用Cora資料集當成目標攻擊對象，分別加以施加5%、10%、15%、20%、25%的Non-targeted attack, metattack。

實驗一

首先我們重現原作者的Pro-GNN在受到不同擾動率Metattack攻擊的效果，從結果可以觀察到當擾動率上升時，節點預測準確率會大幅下降。其準確率之表現受到擾動率大小的影響較顯著

Dataset	ptb rate	acc_val (%)	test_acc (%)
Cora	5 %	81.53	80.63
Cora	10 %	73.09	68.91
Cora	15 %	66.27	65.64
Cora	20 %	58.63	55.58
Cora	25 %	54.22	51.91

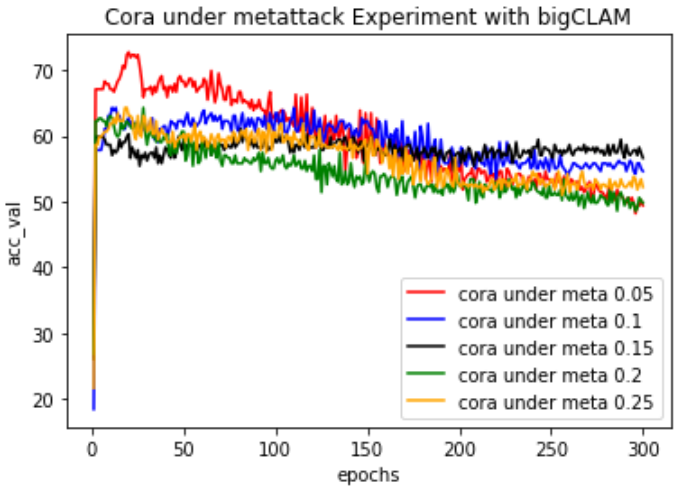


實驗二

針對實驗一延伸的新想法是對於擾動率大的圖結構，我們是否有一個較好的前處理步驟可以施加，讓Pro-GNN的準確率可以提升。

因此以下我們拿受擾動的Cora資料先利用BigCLAM演算法還原其labels，並將受擾動的鄰接矩陣以及標籤一同放到Pro-GNN中做訓練、驗證以及測試，結果發現我們施加的前處理可以讓Pro-GNN訓練出來的model準確率維持在一個區間當中，較不容易受到擾動率大小的影響。

Dataset	ptb rate	acc_val (%)	test_acc (%)
Cora	5 %	72.69	63.78
Cora	10 %	64.26	60.26
Cora	15 %	61.04	56.79
Cora	20 %	64.26	59.51
Cora	25 %	64.26	58.55



施加這個前處理方法的結果雖然讓Pro-GNN在擾動率小的情況之下表現比較差，但根據結果顯示可以發現在圖結構受到較高擾動的情形下，利用BigCLAM可以小幅度的提升Pro-GNN的準確率。由於隨著擾動率提高，鄰接矩陣變得與原圖差異相當巨大，將其配合原標籤放入Pro-GNN中訓練已經無法為模型帶來足夠正確的訓練資訊了，因此前處理是可以用來提高Pro-GNN準確率而採用的策略之一。

Conclusion

- 我們重現了原論文的實驗並多跑了幾個實驗設定後認同原論文的看法， two-stage只有在高擾動率的圖會比one-stage好一點，大部分情況下還是one-stage好，且攻擊模型通常有攻擊預算限制，不會擾動太大。所以我們認為應該採用 one-stage 的模型設定。
- 增加 inner step 在一些 dataset 是可行的方法，可以把此動作視為一次大的結構更新，這不等於提高 learning rate ，它會是多個結構還原變動的加總而非單個結構變動的增幅，雖然在一些 dataset 會犧牲一點正確率，但可以換得更快的收斂速度，對於一些過於複雜的 Graph 是可以考慮的方法，同時也印證了原論文提出的應該邊還原架構邊train GNN model
- 重現論文在受到 metattack 攻擊之下的模型表現，我們發現加入對擾動圖適當的前處理可以提升 Pro-GNN 在受到高擾動率時的準確率。