LER: Spectral Data Augmentation for Graph Neural Network

Wenyi Xue
518030910296
wenyixue@sjtu.edu.cn

Sijia Li
518030910294
qdhlsj@sjtu.edu.cn

Luyuan Jin
518030910293
luyuanjin@sjtu.edu.cn

Abstract

Graph Neural Networks(GNNs) are deep learning based methods that operate on graph domain, and they have shown to be powerful tools for graph analysis. However, the existing GNNs are usually susceptible to the quality of the input graph, which is an important factor for the model accuracy. How to modify or augment input graph to increase the model accuracy and time efficiency has become an advanced question. In this paper, we propose a new algorithm LER for graph data augmentation to improve downstream task node classification accuracy. We first adopt a learnable LPA (Label Propagation) algorithm which approximates the edge weights, then use the weights to do Spectral Sparsification based on using effective resistance. We compute an effective resistance value for each edge, which measures the importance of an edge to the graph structure, and set a sampling amount to know when to finish the sparsification. The experiment show that the sparsified, weighted graphs produced by our algorithm as input have a better accuracy performance compared with other existing algorithms.[1]

Keywords: GNN; LPA; spectral sparsification; effective resistance

1. Introduction

GNN (Graph Neural Network) is the most efficient model to extract valid information from graphs. For GNNs, the input of the graph is an important factor to the result of the entire model and running time. Transforming and augmenting the input may effectively improve the accuracy of the graph neural network. In contrast to various techniques developed in Convolutional Neural Networks, little work studies data augmentation for graphs partly due to the complex, non-Euclidean structure of graphs, which limits possible manipulation operations.

One method of data augmentation is contrastive learning. Yuning You & Yichen Xu et al., 2020 [1] design four types of graph augmentations to incorporate various priors. Yanqiao Zhu et al., 2020 [2] propose a novel graph contrastive representation learning method with adaptive augmentation that incorporates various priors for topological and semantic aspects of the graph. Kezhi Kong et al., 2020 [3] leverage "free" adversarial training method, iteratively augments node features with gradient-based adversarial perturbations during training, and boost performance at test time.

Another present method is input graph sparsification aimed at approximating a given graph by a graph with fewer edges for efficient computation. There are cut-sparsifiers ((Benczúr & Karger, 1996) [4]), pair-wise distance preserving sparsifiers ((Althöfer et al., 1993) [5]) and spectral sparsifiers ((Spielman & Teng, 2004 [6]; Spielman & Srivastava, 2011) [7]).In our work, we take spectral sparsification. There are also work applying sparsification to GNNs like Tong Zhao et al., 2020 [8], which removes the $j|epsilon|$ existing edges with least edge probabilities or like Cheng Zheng et al.(2020) [9] , which propose Neural Sparsification to address the noise brought by the task-irrelevant information.

However, most sparsification applied on Graph Neural Networks are just focused on the time efficiency (Srinivasa,2020[10]) or graph structures. Compared with present work, we apply the spectral sparsification combined with LPA on graph neural networks, which achieves a good performance on the accuracy of downstream tasks. We think we are the first one to combine the spetral sparsification with the downstream tasks' accuracy. Our comtributions are as below.

Contributions. We propose LER, a data augmentation method using **L**abel Propagation Algorithm and Spectral

[1] Our code is availabel at https://github.com/wenyixue/LER

Sparsification based on **E**ffective **R**esistence. Our algorithm works in the structral space of the graph by transforming the original graph into a slightly sparsified graph with reasonable weight. Section 2 introduced out motivations and considerations of using this method. While randomly dropping edges may lead to information loss, our algorithm enhance the critical information and weaken the noise, therefore achieves good results.

In Section 3, we introduce the way we give an unweighted graph reasonable weight, learnable LPA. We adopt this method based on the fact that certain weights that achieve good result in LPA can maximize intra-class label influence.

In Section 4, we use spectral sparsification algorithm whose main idea is sampling based on the effectiveness resistance of all edges. Through this algorithm, we can get the sparsified graph with the assigned spectrum similarity.

In Section 5, we demonstrate the good results that our algorithm achieved on five datasets. By introducing LER to implement data augmentation, our method surpasses the state-of-the-art GCN-LPA baselines [11].

Figure 1. The procedure of data augmentation

2. Preliminaries

Let $G = (V; E)$ be the input graph with node set V and edge set E. Let $N = |V|$ be the number of nodes. We denote the adjacency matrix as $A \in \{0, 1\}^{N\times N}$, where A_{ij} = 0 indicates node i and j are not connected. We denote the node feature matrix as $X \in R^{N\times F}$, where F is the dimension of the node features and X_i: indicates the feature vector of node i (the ith row of X). We define D as the diagonal degree matrix such that $D_{ii} = \sum_j A_{ij}$.

Graph Neural Networks. We use the well-known graph convolutional network (GCN) [12] as an example when explaining GNNs in the following sections; however, our arguments hold straightforwardly for other GNN architectures. Each GCN layer (GCL) is defined as:

$$H^{(l+1)} = f_{GCL}(A, H^{(l)}; W^{(l)}) = \sigma(D^{-\frac{1}{2}}AD^{-\frac{1}{2}}H^{(l)}W^{(l)})$$

where A is the weighted adjacency matrix, D is the diagonal matrix $D_{ii} = \sum_j A_{ij}$, and σ denotes a nonlinear activation such as ReLU.

Node Classification. The node classification task is to learn a model to predict the label of the unlabeled nodes by inferring from a small portion of labeled node. For a graph described above, The goal is to learn a mapping $M : V \to L$ to predict labels. Nodes of the same labels are denoted as intra-class nodes and nodes of different labels are denoted as inter-class nodes.

Data Augmentation. Augmentation strategies for improving generalization have been broadly studied in contexts outside of graph learning. In recent years, variants of such techniques are widely used in natural language processing (NLP) and computer vision (CV). There are limited studies on doing data augmentation on graphs which consist of structral augmentation and feature augmentation.

Graph Sparsification. Graph sparsification is to approximate an arbitrary graph by a sparse graph and is useful in many applications, such as simplification of social networks, least squares problems, numerical solution of symmetric positive definite linear systems and etc. The simplest way to sparsify a graph is to sample some of its edges.

3. Using LPA to Produce Edge Weights

3.1. Label Propagation Algorithm

Label Propagation Algorithm assumes that two connected nodes are likely to have the same label, and thus it propagates labels iteratively along the edges. Let $Y^{(k)} = [{y_1}^{(k)}, \cdots, {y_n}^{(k)}]^T \in R^{n\times c}$ be the soft label matrix iteration $k > 0$, in which the i-th row ${y_i^{(k)}}^T$ denotes the predicted label distribution for node v_i in iteration k.Let D be the diagonal degree matrix for weighted adjacency matrix A with entries $d_{ii} = \sum_j a_{ij}$. The LPA in iteration k is formulated as the following steps:

$$Y^{(k-1)} = D^{-1}AY^{(k)}$$
$${y_i}^{(k+1)} = {y_i}^{(0)}, \forall i < m$$

All nodes propagate labels to their neighbors according to normalized edge weights. Then, labels of all labeled nodes are reset to their initial values, because LPA wants to persist labels of nodes which are labeled so that unlabeled nodes do not overpower the labeled ones as the initial labels would otherwise fade away.

3.2. Label influence

The label influence of labeled node vb on unlabeled node v_a after k iterations of LPA is the gradient of $y_a^{(k)}$ with respect to y_b:

$$I_l(v_a, v_b; k) = \partial {y_a}^{(k)} / \partial y_b$$

The total label influence of nodes with label y_a on node v_a is proportional to the probability that node v_a is classified as y_a by LPA:

$$I_l(v_a, v_b; k) \propto Pr({y_a}^{lpa}) = y_a$$

3.3. Making LPA learnable

If edge weights a_{ij} maximize the probability that v_a is correctly classified by LPA, then they also maximize the intra-class label influence on node v_a. We can therefore first learn the optimal edge weights A^* by minimizing the loss of predicted labels by LPA:

$$\begin{aligned} A^* &= \arg\min_A L_{lpa}(A) \\ &= \arg\min_A \frac{1}{m} \sum\nolimits_{v_a:a\leq m} J({y_a}^{lpa}, y_a) \end{aligned}$$

where J is the cross-entropy loss, ${y_a}^{lpa}$ and y_a are the predicted label distribution of v_a using LPA and the true one-hot label of v_a, respectively. $a \leq m$ means v_a is labeled.

4. Sparsification Algorithm

According to the concepts, GNNs are approximations of spectral graph convolutions, and we aim to sparsify the graph while preserving the spectrum of the graph. . The spectral graph sparsification ensures that the spectral content of $\boldsymbol{H}$ is similar to that of $\boldsymbol{G}$:

$$(1-\epsilon)\lambda_i(\boldsymbol{L_G}) \leq \lambda_i(\boldsymbol{L_H}) \leq (1+\epsilon)\lambda_i(\boldsymbol{L_G}),$$

where ϵ is a desired threshold. L represents the graph Laplacian (defined as $\boldsymbol{L = D - A}$ where D is the degree matrix and A is the adjacency matrix), $\lambda_i(\boldsymbol{L})$ denotes the i-th eigenvalue of $\boldsymbol{L}$ and let $\boldsymbol{A}^\dagger$ denotes the Moore-Penrose inverse of a matrix. $\boldsymbol{L_H}$ denotes Laplacian matrices of the original graph $\boldsymbol{G}$ and the sparsified graph $\boldsymbol{H}$.
The sparsification can be complement using the sampling method based on effective resistance proposed by Spielman and Srivastava in 2011 [6].

4.1. Analysis

Definition (Effective Resistance) The effective resistance between any two nodes of a graph can be defined as the potential difference induced across the two nodes, when a unit current is induced at one node and extracted from the other node.

$$R_e(u,v) = \boldsymbol{b_e^T L^\dagger b_e}$$

where $\boldsymbol{b_e = x_u - x_v}$ ($\boldsymbol{x_k}$ is a standard basis vector with 1 in the k-th position) and $\boldsymbol{L}^\dagger$ is the pseudo-inverse of the graph Laplacian matrix.

The reason why effective resistance is used is that it can measure the importance of an edge to the graph structure, which is important for the accuracy of GNNs.

The main idea of our algorithm is sampling, based on the edge weights and $R_e(u,v)$. If an edge has a higher edge weight and $R_e(u,v)$, it means that this edge is more important for maintaining the graph structure and deleting this edge may lead to the disconnection of sparsified graph.
Choosing ϵ. ϵ is a pruning parameter, which determines the quality of approximation after sparsification, and we set this as 0.2 in our experiment.
Choosing sampling amount. The default argument set of sampling amount is int($0.16N \log N/\epsilon^2$). Or we can set this forcely to be M and the final sampling amount $q = \max(M, (0.16N \log N/\epsilon^2))$. The number of remaining edges is no more than q.
Complexity The complexity of sampling is $O(N \log N)$ and the complexity of computing R_e is $O(M \log N)$ where N is the number of nodes and M is the number of edges of original graph.

4.2. Algorithm Details

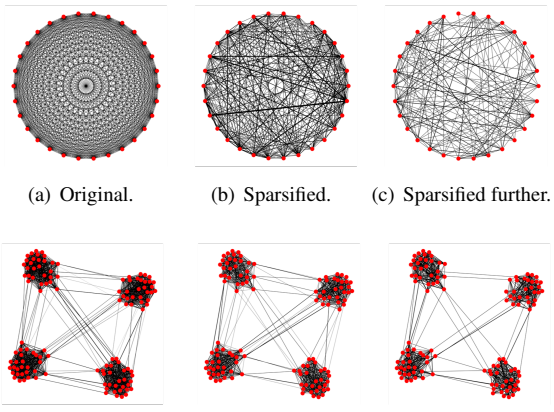
Algorithm 1 Spectral Sparsification

Input: Graph $\boldsymbol{G}(V,E)$, w_e is the edge weight for $e \in E$.
Output: $\boldsymbol{H}$ = Graph(V, E_H)
function SPARSIFY(G, M, ϵ)
  For each edge $e(u,v)$ compute $R_e(u,v)$.
  Set q = max(M, int(0.16NlogN/ϵ^2)), $\boldsymbol{H}(V, E_H = Null)$
  while present samples $\leq q$ **do**
    Sample an edge e from the distribution p_e proportional to $w_e R_e$
    if $e \in E_H$ **then**
      Add w_e/qp_e to its weight
    else
      Add e to E_H with weight w_e/qp_e
    end if
  end while
end function

The main idea of the algorithm is to get a sparsified graph $\boldsymbol{H}$ from the original graph $\boldsymbol{G}$ by the function $H = Sparsify(G, q)$ where q is the number of samples. Choose a random edge e of $\boldsymbol{G}$ with probability p_e proportional to $w_e R_e$, and add e to $\boldsymbol{H}$ with the weight w_e/qp_e. Take q samples independently, summing weights if an edge is chosen more than once. The pseudocode is shown in Algorithm 1.

4.3. Sparsification Results

We can see from Figure 2 that the algorithm not only sparse the graph, but also ensure the basic structure of the graph. At the same time, the invalid miscellaneous information is deleted and some key information is emphasized. The algorithm works well for both circular and rectangular topological structures.



(a) Original. (b) Sparsified. (c) Sparsified further.

(d) Original. (e) Sparsified. (f) Sparsified further.

Figure 2. Sparsification Process Visualization

5. Experiments

We evaluate our method by performing node classification tasks on five datasets including citation networks and coauthor networks. We expect to achieve better test accuracy of classification than the GCN-LPA method with no data augmentation [11]. Also, we are interested in the optimal ratio of sparsified edges and intend to find out when the accuracy will be maximized under different sparsification ratios.

5.1. Datasets

We use the following five datasets in our experiments:

Citation networks: We use three citation networks, Cora, Citeseer [13] and Pubmed [14]. Each dataset contains bag-of-words representation of documents and citation links between the documents. We can construct a graph from these citation network datasets. The nodes are papers with bag-of-words as feature vectors. The edges are citation links between papers, and the label is paper category.

Coauthor networks. We also use two co-authorship graphs Coauthor CS and Coauthor Physics based on the Microsoft Academic Graph from the KDD Cup 2016 challenge 3 [2]. Here, nodes are authors, which are connected by edges if they coauthored a paper. Node features represent paper keywords for each author's papers, and class labels indicate most active fields of study for each author [15].

Statistics of the five datasets are shown in Table 1.

Table 1. Dataset statistics after preprocessing

	Cora	Citeseer	Pubmed	Coauthor-CS	Coauthor-Phy
# nodes	2,708	3,327	19,717	18,333	34,493
# edges	13,624	12,431	108,365	182,121	530,417
# features	1,433	3,703	500	6,805	8,415
# classes	7	6	3	15	5

5.2. Baselines

We compare against the following baselines in our experiments. Both of them utilizes GCN-LPA to train the GNN model [11].

No edge deletion. The first baseline takes the original graph as input. LPA propagates node label information across the edges of the graph [16], while GCN propagates and transforms node feature information [12].

Random edge deletion. The second randomly delete edges from the original graph and get a sparser graph as input. The number of randomly deleted edges is set to be equal to the number of edges removed by our sparsification algorithm, so that its only difference from our algorithm is the sampling strategy.

5.3. Experiment Setup

Our experiment focus on the traditional node classification task where we only know labels of part of nodes but know the whole graph as well as features of all nodes. We follow the semi-supervised setting in most GNN literature [12, 17], and set the proportion of training, validation and testing set to be $6:2:2$. Each experiment is repeated three times and we report the average result. We train our model using Adam [18], and report the test accuracy when the validation accuracy is maximized. We also utilize L2 regularization to the transformation matrices and the dropout technique. The overall hyper-parameter settings are displayed in Appendix A.

For each dataset, we run multiple experiments with various values sampling amount q so as to result in different sparsification ratios. We take our best result out of different sparsification ratios to represent our algorithm.

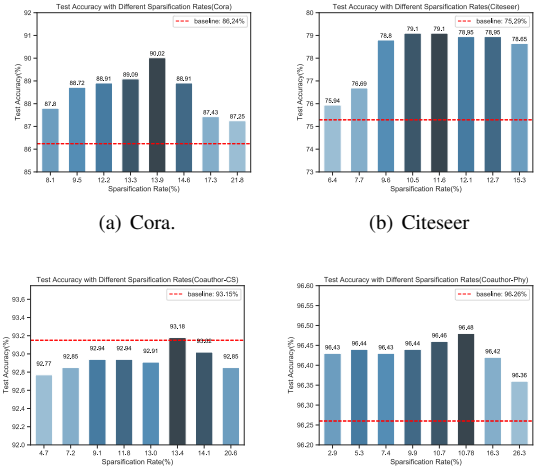
[2] https://kddcup2016.azurewebsites.net/

Table 2. The final test accuracy of node classification on the five datasets

	Cora	Citeseer	Pubmed	Coauthor-CS	Coauthor-Phy
No edge deletion	0.8624	0.7529	0.8707	0.9315	0.9626
Random edge deletion	0.7715	0.6257	0.8667	0.8145	0.8711
LER (ours)	**0.9002**	**0.7910**	**0.8811**	**0.9318**	**0.9648**

5.4. Experiment Results

Overview. The results of final test accuracy of node classification are listed in Table 2. We adopt different sparsification ratio on each dataset, and take the best result to represent our method. The results indicates that our method LER outperforms the state-of-the-art GCN-LPA model [11]. Compared with the baseline which does not delete edges, our spectral sparsification algorithm has made improvements on the model accuracy. However, the random deleting strategy performs worse than the baseline, indicating that dropping edges randomly is unhelpful to enhance the task-relevant information. Therefore, our method provides a reasonable way to denoise the input graph and thus improve the performance and robustness of GNN.



(a) Cora. (b) Citeseer

(c) Coauthor-CS. (d) Coauthor-Phy.

Figure 3. Test accuracy with Different sparsification rate

Sparsification ratio. We also evaluate the different test accuracy on each dataset corresponding to different sparsification ratios, that is to say, the proportion of the removed edges. The relationship between sparsification ratio and final test accuracy is shown in Figure 3. For each dataset, the optimal sparsification ratio is approximately 10%. When the sparsification ratio is too small, not enough task-irrelevant edges are dropped, and the sparsified graphs still suffer from noise, and thus the test accuracy is suboptimal. The test accuracy increases as we improve the sparsification ratio until the ratio reaches its optimal value. Nevertheless, if we continue to improve the sparsification ratio, we may hurt the useful information and result in decreasing test accuracy.

Variance on datasets. For dataset Cora and Citeseer, the improvement of our algorithm over the baseline is impressive, while for the remaining three larger datasets Pubmed, Coauthor-CS, and Coauthor-Phy, the improvement is relatively small, considering the baseline is already fairly high.

6. Conclusion and Future Work

In this paper, we propose a model LER which combines the LPA and spectral sparsification to conduct spectral data augmentation for graph neural network. Present work are most focused on the time efficiency but our experiment shows that on the prediction accuracy for the downstream task node classification, our method has a better performance compared with other existing methods.

In the future, we plan to increase our algorithm's performance on time efficiency and CPU space. Also, we plan to apply more sparsification algorithms on GNNs systematicaly and compared their differences on time and accuracy according to the algorithm theories and experiment results.

Acknowledgments

Sincere gratitude to our teacher for valuable advice to our problem formulation and teaching us necessary knowledge of algorithm.

Sincere gratitude to the mentor Mr. Dongyue Li for all the guidance for our ideas and methods.

References

[1] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020.

[2] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Graph contrastive learning with adaptive augmentation," *CoRR*, vol. abs/2010.14945, 2020.

[3] K. Kong, G. Li, M. Ding, Z. Wu, C. Zhu, B. Ghanem, G. Taylor, and T. Goldstein, "FLAG: adversarial data augmentation for graph neural networks," *CoRR*, vol. abs/2010.09891, 2020.

[4] A. A. Benczúr and D. R. Karger, "Approximating *s-t* minimum cuts in $\tilde{O}(n^2)$ time," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, G. L. Miller, Ed. ACM, 1996, pp. 47–55.

[5] I. Althöfer, G. Das, D. P. Dobkin, D. Joseph, and J. Soares, "On sparse spanners of weighted graphs," *Discret. Comput. Geom.*, vol. 9, pp. 81–100, 1993.

[6] D. A. Spielman and S. Teng, "Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems," in *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, L. Babai, Ed. ACM, 2004, pp. 81–90.

[7] D. A. Spielman and N. Srivastava, "Graph sparsification by effective resistances," *SIAM J. Comput.*, vol. 40, no. 6, pp. 1913–1926, 2011.

[8] T. Zhao, Y. Liu, L. Neves, O. J. Woodford, M. Jiang, and N. Shah, "Data augmentation for graph neural networks," *CoRR*, vol. abs/2006.06830, 2020.

[9] C. Zheng, B. Zong, W. Cheng, D. Song, J. Ni, W. Yu, H. Chen, and W. Wang, "Robust graph representation learning via neural sparsification," in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 2020, pp. 11 458–11 468.

[10] R. S. Srinivasa, C. Xiao, L. Glass, J. Romberg, and J. Sun, "Fast graph attention networks using effective resistance based graph sparsification," *CoRR*, vol. abs/2006.08796, 2020.

[11] H. Wang and J. Leskovec, "Unifying graph convolutional neural networks and label propagation," *arXiv preprint arXiv:2002.06755*, 2020.

[12] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[13] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008.

[14] G. Namata, B. London, L. Getoor, B. Huang, and U. EDU, "Query-driven active surveying for collective classification," in *10th International Workshop on Mining and Learning with Graphs*, vol. 8, 2012.

[15] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," *arXiv preprint arXiv:1811.05868*, 2018.

[16] X. Zhu, J. Lafferty, and R. Rosenfeld, "Semi-supervised learning with graphs," Ph.D. dissertation, Carnegie Mellon University, language technologies institute, school of computer science, 2005.

[17] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[19] Z. Yang, W. Cohen, and R. Salakhudinov, "Revisiting semi-supervised learning with graph embeddings," in *International conference on machine learning*. PMLR, 2016, pp. 40–48.

[20] G. B. Hermsdorff and L. Gunderson, "A unifying framework for spectrum-preserving graph sparsification and coarsening," in *Advances in Neural Information Processing Systems*, 2019, pp. 7736–7747.

[21] D. Luo, W. Cheng, W. Yu, B. Zong, J. Ni, H. Chen, and X. Zhang, "Learning to drop: Robust graph neural network via topological denoising," *arXiv preprint arXiv:2011.07057*, 2020.

[22] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *arXiv preprint arXiv:1812.08434*, 2018.

Appendix A. Hyper-parameters settings for experiments

The detailed hyper-parameter settings for our experiments based on GCN-LPA are listed in Table 3.

Table 3. Hyper-parameter settings for all the five datasets

	Cora	Citeseer	Pubmed	Coauthor-CS	Coauthor-Phy
Dimension of hidden layers	32	16	32	32	32
# GCN layers	5	2	2	2	2
# LPA iterations	5	10	5	5	2
L2 weight	1e-4	5e-4	2e-4	1e-6	1e-6
LPA weight (λ)	10	5	1	10	1
Dropout rate	0.2	0	0	0.2	0.2
Learning rate	0.05	0.1	0.1	0.1	0.05
# epochs	200	200	200	100	100
epsilon (ϵ)	0.2	0.2	0.2	0.2	0.2
eta	5e-5	5e-5	5e-5	5e-5	4e-5