Reproducibility and Improvement Analysis on Graph Recurrent Attention Networks

Dan Liu
School of Computer Science
McGill University
3480 Rue University, Montréal, QC H3A 2A7
dan.liu4@mail.mcgill.ca

Abstract

In this report, three different modifications are applied to the GRAN model for the ablation studies. To examine the changes in model performance, different model components will be removed and an extra layer will be added in this experiment. In terms of removing the model components, the size of the GRU layers will be reduced from seven to two and the number of the hidden units (e.g. 512,256 and 128) will be reduced to 50. The extra angular layer is introduced to Bernoulli mixture model, so as to fulfill the improvement task. There are four groups of experiments running 5,000 iterations to study the impact of the modifications. The baseline model is the original model with the default configurations. The experiments compare the performance between the modified models and the original model. The results show that the reduction methods achieve similar performance on the lobster and point cloud dataset, while the addition of the angular layer improves models' performance.

1 Introduction

The purpose of this project is to reproduce and modify the Graph Recurrent Attention Networks (GRAN) [6] model for ablation study. The source codes and datasets are provided by the author. This project mainly involves following experiments: (i) the experiment that reduces the number of the layers and the hidden units of the GRU components; (ii) the experiment that applies the angular layer to explore any performance changes; (iii) the reproduction experiment that runs the codes provided by the author.

The GRAN uses a linear mapping function to capture the graph adjacency information. Then a large and deep recurrent neural network is used to learn all of the graph in the training dataset. As the maximum number of edges is only 10,886 in the provided configurations, the GRAN may only memorizes the training data rather than learns from the data. To test this assumption, the layers and the hidden units of the GRU components in RNNs are reduced to two and 50, respectively.

Another work in this report is to introduce an angular layer to test the sampling performance. Such a layer uses weight regularization methods to constraint the weights in the angular space. It can reduce the intra-class distance and enlarge the inter-class distance.

The experiment aims to examine the performance of the models. The experiments are based on the provided datasets and source codes. Each experiment only has one modified component. The results indicate that the angular layer can improve the overall performance. The modifications, e.g. two-layer GRU and reduced hidden units, show different performance on different datasets.

The work contributes in the following ways: (i) different modified models are compared against the baseline model; (ii) the proposed weight regularization methods can improve the clustering coefficient metric in most cases.

Section 2 of this report presents related work. Section 3 describes the datasets and the experiment setup. Section 4 illustrates the proposed approaches. Section 5 shows the experimental results. Section 6 provides the conclusion and thoughts on future directions.

2 Related Works

The GRAN, proposed by Liao et al. in 2019, has architecture similar to the GraphRNN [12] that uses the Bernoulli distribution to sample the edges, the GRAN uses the Bernoulli mixture model to generate the edges. However, unlike the GraphRNN that uses the adjacency vector to represent the connection between the new node and the current graph, the GRAN first linearly maps the adjacency matrix to a subspace, then uses each row of the dimension-reduced matrix as the node representation. There are several other works related to the deep graph generating models, for example, variational auto-encoder [3] and RNNs based framework [4, 11, 7, 5, 12]. Some of the models generate new nodes and edges separately. However, such methods are limited to the generation of small graphs. Besides, generating parts of the graph adjacency matrix independently does not reflect the complex structural dependencies in the real world. Another type of methods involves the use of the auto-regressive models to capture the dependencies from history to generate new nodes and edges. The auto-regressive method builds a graph by learning the joint probability of existing nodes and edges in the graph, which is similar to language modeling that predicts new weights based on previous information. One of the challenges in the auto-regressive method is the modeling of the node sequential orderings of the adjacency matrix. The recent works related to the auto-regressive method are the GraphRNN and GRAN.

3 Dataset and Experiment Setup

Four types of datasets from the GRAN are used in this work. The experiment configurations are provided by the author. In order to compare with other models, the GRAN generates two datasets containing random grid graphs and uses two external datasets of protein [1] and point cloud [10] respectively. There are 100 standard 2D grid graphs with random number of nodes (from 100 to 400) in the Grid dataset. The lobster [2] dataset contains 100 random lobster graphs xxx with the number of nodes from 10 to 100 in each graph. The Protein dataset includes 918 protein graphs 100 to 500 nodes in each graph. For the Point Cloud dataset, there are 41 generated 3D objects with 1,377 nodes per graph on average. The author splits each dataset into 80% and 20% for training and test. Part of training samples (20%) are used for validation. The node ordering in this study is DFS only (default order in the provided code). Both the baseline model and the modified models are running on the provided code with 5,000 iterations. The detailed information of the measurement metrics can be found in the GRAN paper.

4 Proposed Approach

4.1 Angular Linear Layer

Inspired by the work of [8, 9], which used the modified linear function to enlarge the distance between the decision boundary and the sample points during training, a modified linear layer is implemented to constrain the weights W of the Bernoulli mixture model. The angular layer can enlarge the inter-class distance among different mixture components and reduce the intra-class distances, hence reducing the variance for each component.

The original softmax loss can be written as:

$$L = \frac{1}{N}\sum_i L_i = \frac{1}{N}\sum_i -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right) \tag{1}$$

where f_j denotes the j_{th} element of the class score vector f, $j \in [1, K]$, K is the class number, and N is the number of the training samples. The f is the output of a fully connected layer W, so $f_j = \boldsymbol{W}_j^T \boldsymbol{x}_i + b_j$ and $f_{y_i} = \boldsymbol{W}_{y_i}^T \boldsymbol{x}_i + b_{y_i}$ where x_i denotes the i_{th} training sample. The L can be

written as:

$$L_i = -\log\left(\frac{e^{\boldsymbol{W}_{y_i}^T \boldsymbol{x}_i + b_{y_i}}}{\sum_j e^{\boldsymbol{W}_j^T \boldsymbol{x}_i + b_j}}\right) = -\log\left(\frac{e^{\|\boldsymbol{W}_y\|\|\boldsymbol{x}_i\|\cos\left(\theta_{y_i,i}\right) + b_{y_i}}}{\sum_j e^{\|\boldsymbol{W}_j\|\|\boldsymbol{x}_i\|\cos(\theta_{j,i}) + b_j}}\right) \tag{2}$$

where $\theta_{j,i}\,(0 \leq \theta_{j,i} \leq \pi)$ is the angle between the vectors W_j and x_i. The binary classification decision boundary of the original softmax function is:

$$\left(\boldsymbol{W}_1 - \boldsymbol{W}_2\right)\boldsymbol{x} + b_1 - b_2 = 0. \tag{3}$$

Normalizing the $\|\boldsymbol{W}_j\| = 1, \forall j$ and the modified softmax function is defined as:

$$L_{\text{modified}} = \frac{1}{N}\sum_i -\log\left(\frac{e^{\|\boldsymbol{x}_i\|\cos\left(\theta_{\boldsymbol{y}_i,i}\right)}}{\sum_j e^{\|\boldsymbol{x}_i\|\cos(\theta_{j,i})}}\right) \tag{4}$$

Thus the decision boundary of the modified softmax function for binary classification is:

$$\|\boldsymbol{x}\|\left(\cos\theta_1 - \cos\theta_2\right) = 0. \tag{5}$$

The function is multiplied by a factor to enlarge the margin between the decision boundaries:

$$L_{\text{ang}} = \frac{1}{N}\sum_i -\log\left(\frac{e^{\|\boldsymbol{x}_i\|\cos\left(m\theta_{y_i,i}\right)}}{e^{\|\boldsymbol{x}_i\|\cos\left(m\theta_{y_i},i\right)} + \sum_{j\neq y_i} e^{\|\boldsymbol{x}_i\|\cos(\theta_{j,i})}}\right) \tag{6}$$

Then the decision boundary becomes:

$$\begin{array}{l} \|\boldsymbol{x}\|\left(\cos m\theta_1 - \cos\theta_2\right) = 0 \text{ for class 1} \\ \|\boldsymbol{x}\|\left(\cos\theta_1 - \cos m\theta_2\right) = 0 \text{ for class 2} \end{array} \tag{7}$$

Such decision boundaries will produce an angular margin of $\frac{m-1}{m+1}\theta_2^1$ where θ_2^1 is the angle between W_1 and W_2. The modified function is generalized to a monotonic decreasing angle function $\psi\left(\theta_{y_i,i}\right)$ that equals to $\cos\left(\theta_{y_i,i}\right)$ in $\left[0, \frac{\pi}{m}\right]$. The angular layer with Softmax is:

$$L_{\text{ang}} = \frac{1}{N}\sum_i -\log\left(\frac{e^{\|\boldsymbol{x}_i\|\psi(\theta_y,i)}}{e^{\|\boldsymbol{x}_i\|\psi\left(\theta_{y_i,i}\right)} + \sum_{j\neq y_i} e^{\|\boldsymbol{x}_i\|\cos(\theta_{j,i})}}\right) \tag{8}$$

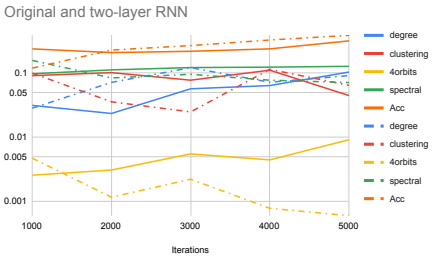
where $\psi\left(\theta_{y_i,i}\right) = (-1)^k\cos\left(m\theta_{y_i,i}\right) - 2k$, $\theta_{y_i,i} \in \left[\frac{k\pi}{m}, \frac{(k+1)\pi}{m}\right]$ and $k \in [0, m-1]$. The distance between the decision boundaries is dependent on the parameter m. In this study, the value of m is 4. The Softmax function in the GRAN is replaced by the modified Softmax.

4.2 Reduced Layers and Hidden Units Model

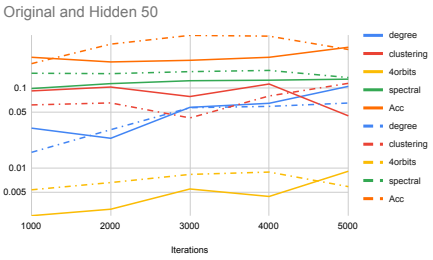
From the provided configurations, there is a seven-layer RNNs in the model. The average number of nodes and edges in the datasets are approximately 500 and 1,000. However, seven layers with 128 to 512 hidden units in each layer are used in the provided source code, which is quite large given the number of training samples. One of the ablation studies can be reducing the number of RNNs' layers from seven to two. The other one is reducing the number of the GRU hidden units to 50. Each ablation study only modifies one variable to observe the impacts on model performance. Therefore, there are two types of models, namely a two-layer RNNs model to study the impact of changing the number of layers and a reduced-hidden-unit model to study the impact of changing the number of the GRU hidden units.

5 Results

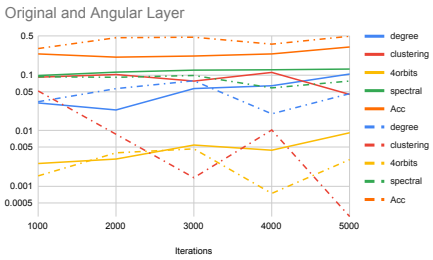
This report covers 16 experiments(Table 1-4). As for the baseline model, the test results after training with 5,000 iterations are listed in Table 1-4. The results indicate that the overall performance on the dataset with a larger average number of nodes and edges are worse than the smaller graph dataset. Specifically, the larger dataset is the point cloud dataset, and the smaller datasets refer to the rest of the datasets. Since all of the experiments are running on the public cluster server and the running time depends on the current system load, there is no specific running time measurement in this study. It is obvious that shallow layers and less hidden units will accelerate the speed. The test trending charts of the experiments are switched to log scale.



(a) Original and Two-layer RNN



(b) Original and Hidden50



(c) Original and Angular

Figure 1: The test trending chart of different models on the lobster dataset. The solid line indicates the results of the original model.

5.1 Lobster

The results on the Lobster dataset indicate that the angular layer improves the performance (achieving lower values). The two-layer RNNs achieves better results in the four-orbit and the spectral metrics (Table 1). The results also show that the performance of the original model is similar to the reduced-hidden-unit model (Figure 1).

Type	degree	clustering	4orbits	spectral	Acc
Original	0.1042	0.0447	0.0091	0.1281	0.3200
Angular	**0.0462**	**0.0003**	0.0030	0.0780	**0.5000**
Hidden50	0.0646	0.1136	0.0059	0.1342	0.3000
2-Layer	0.0917	0.0646	**0.0006**	**0.0715**	0.3900

Table 1: The test accuracy of the models on the lobster dataset after 5000 training iterations.

5.2 Grid

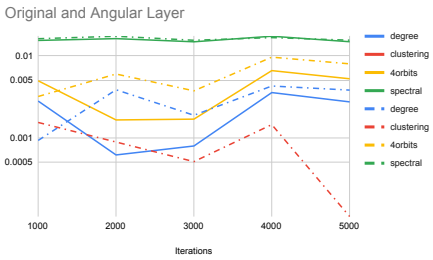
The Grid dataset consists of the synthesized graphs with an average of 210 nodes and 400 edges. The results show that the spectral metrics of the modified models are similar to the original model (Table 2). The reduced-hidden-unit model does not perform as well as the original model. All of the modifications does not result in any improvement on this dataset (Figure 2).

Type	degree	clustering	4orbits	spectral
Original	0.0027	**0.0000**	0.0052	**0.0148**
Angular	0.0038	0.0001	0.0079	0.0154
Hidden50	0.0051	0.0011	0.0111	0.0171
2-Layer	**0.0016**	**0.0000**	**0.0027**	0.0165

Table 2: The test accuracy of the models on the grid dataset after 5000 training iterations.

(a) Original and Two-layer RNN

(b) Original and Hidden50



(c) Original and Angular

Figure 2: The test trending chart of different models on the grid dataset. The solid line indicates the results of the original model.

5.3 Point Cloud (DB)

The Point Cloud dataset contains simulated 3D objects and therefore has more nodes and edges. The results on this dataset show that the angular layer improves the overall performance (Table 3). The performance of the reduced-hidden-unit model is similar to the original model (Figure 3). The two-layer RNNs model does not show any advantage in this experiment. With larger graphs, the deep structure model performs better than the shallow layers model.

Type	degree	clustering	4orbits	spectral
Original	0.2888	**0.3836**	1.0662	0.0452
Angular	**0.0833**	0.4182	0.8853	0.0120
Hidden50	0.1185	0.4353	0.9846	**0.0094**
2-Layer	0.0992	0.4931	**0.8176**	0.0245

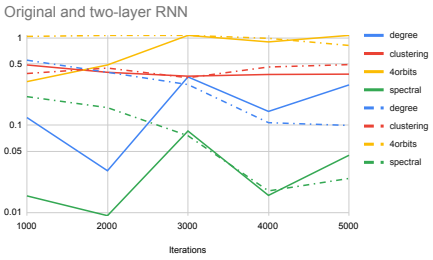
Table 3: The test accuracy of the models on the point cloud dataset after 5000 training iterations.

5.4 Protein (DD)

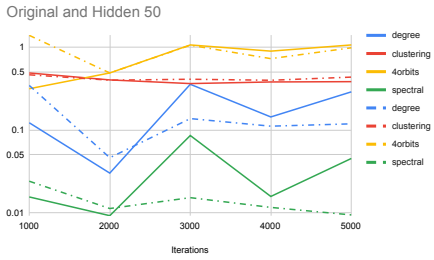
The results on this dataset show that the angular model is better than others (Figure 4). The rest modified models do not show any advantage in this experiment. With larger graphs, the deep structure model performs better than the shallow layers model (Table 4).

Type	degree	clustering	4orbits	spectral
Original	0.0120	0.0838	**0.0447**	0.0087
Angular	**0.0049**	**0.0419**	0.0478	**0.0062**
Hidden50	0.0246	0.0762	0.5619	0.0128
2-Layer	**0.0141**	0.1025	0.0990	0.0112

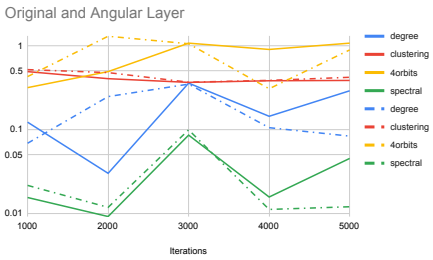
Table 4: The test accuracy of the models on the protein dataset after 5000 training iterations.



(a) Original and Two-layer RNN



(b) Original and Hidden50



(c) Original and Angular

Figure 3: The test trending chart of different models on the point cloud dataset. The solid line indicates the results of the original model.

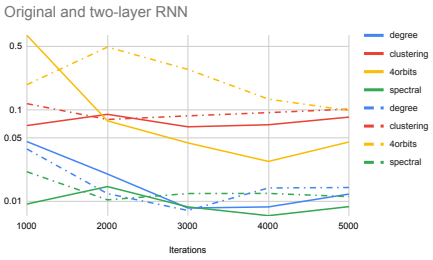
6 Discussion and Conclusion

It is an interesting result that the orbit and degree metrics always show the same trend among all of the experiments. The experiment results show that the deep RNNs (seven layers) may not be necessary for the current size of the datasets. Unlike the GraphRNN that uses the adjacency vectors to represent the relation between new nodes and the current graph, mapping the whole graph into a subspace through a linear function may not be sufficient to represent the nodes and edges efficiently. It may not be possible to learn graph-invariant node features through a single linear transformation. Besides, from the view of NLP, a group of semantically equivalent sentences in different languages can be seen as the same graphs represented by different adjacency matrix. To capture the latent contextual meaning of the sentences, either a large size of training samples or a complex deep neural network is needed. If the model is designed to memorize all of the training graphs, the size of the dataset and the size of the hidden units will have a real impact on the performance of the model. The experiment results support the assumption that the models trained on the larger datasets need more hidden units and layers to achieve better performance.

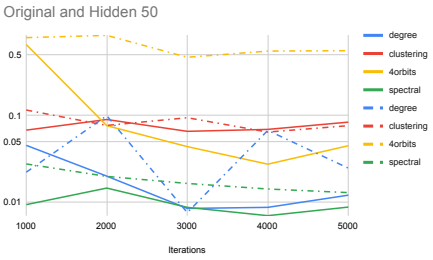
Due to the time and resource limit, the models' performance after each iteration is not tested. This study cannot fully prove that shallow RNNs is able to achieve a performance comparable to the provided complex model. However, based on the observed results, part of the models trained on small datasets (e.g. the lobster dataset) perform better than the baseline model.

The paper of GRAN contains ambiguous descriptions. In the provided source code, the attention component takes both node and edge features. However, the details of this part are not mentioned in the paper.

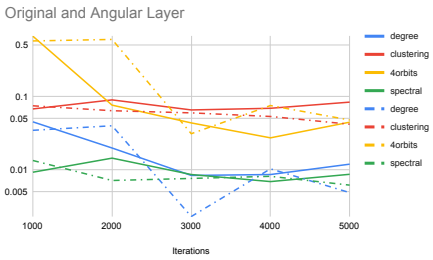
In future work, more iterations can be further tested. Research into the graph-invariant features can help the model to generate more real graphs.



(a) Original and Two-layer RNN



(b) Original and Hidden50



(c) Original and Angular

Figure 4: The test trending chart of different models on the protein dataset. The solid line indicates the results of the original model.

References

[1] P. D. Dobson and A. J. Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*, 330(4):771–783, 2003.

[2] S. W. Golomb. *Polyominoes: puzzles, patterns, problems, and packings*, volume 16. Princeton University Press, 1996.

[3] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[4] T. N. Kipf and M. Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[5] Y. Li, O. Vinyals, C. Dyer, R. Pascanu, and P. Battaglia. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.

[6] R. Liao, Y. Li, Y. Song, S. Wang, W. Hamilton, D. K. Duvenaud, R. Urtasun, and R. Zemel. Efficient graph generation with graph recurrent attention networks. In *Advances in Neural Information Processing Systems*, pages 4257–4267, 2019.

[7] Q. Liu, M. Allamanis, M. Brockschmidt, and A. Gaunt. Constrained graph variational autoencoders for molecule design. In *Advances in Neural Information Processing Systems*, pages 7795–7804, 2018.

[8] W. Liu, Y. Wen, Z. Yu, and M. Yang. Large-margin softmax loss for convolutional neural networks. In *ICML*, volume 2, page 7, 2016.

[9] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song. Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 212–220, 2017.

[10] M. Neumann, P. Moreno, L. Antanas, R. Garnett, and K. Kersting. Graph kernels for object category prediction in task-dependent robot grasping. In *Online Proceedings of the Eleventh Workshop on Mining and Learning with Graphs*, pages 0–6, 2013.

[11] M. Simonovsky and N. Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *International Conference on Artificial Neural Networks*, pages 412–422. Springer, 2018.

[12] J. You, R. Ying, X. Ren, W. L. Hamilton, and J. Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. *arXiv preprint arXiv:1802.08773*, 2018.

Appendix

Test Results

Original					
Iterations	degree	clustering	4orbits	spectral	Acc
1000	0.0314	0.0915	0.0026	0.0979	0.2400
2000	0.0235	0.1022	0.0031	0.1127	0.2100
3000	0.0571	0.0778	0.0055	0.1225	0.2200
4000	0.0641	0.1115	0.0044	0.1243	0.2400
5000	0.1042	0.0447	0.0091	0.1281	0.3200
Angular					
Iterations	degree	clustering	4orbits	spectral	Acc
1000	0.0331	0.0517	0.0015	0.0932	0.3000
2000	0.0569	0.0086	0.0040	0.0911	0.4700
3000	0.0789	0.0014	0.0047	0.0981	0.4800
4000	0.0200	0.0102	0.0007	0.0587	0.3600
5000	0.0462	0.0003	0.0030	0.0780	0.5000
Hidden50					
Iterations	degree	clustering	4orbits	spectral	Acc
1000	0.0157	0.0612	0.0054	0.1518	0.2000
2000	0.0301	0.0648	0.0066	0.1500	0.3500
3000	0.0565	0.0421	0.0083	0.1590	0.4500
4000	0.0587	0.0791	0.0089	0.1646	0.4400
5000	0.0646	0.1136	0.0059	0.1342	0.3000
2-Layer RNN					
Iterations	degree	clustering	4orbits	spectral	Acc
1000	0.0284	0.1003	0.0047	0.1583	0.1200
2000	0.0718	0.0358	0.0012	0.0843	0.2300
3000	0.1220	0.0248	0.0022	0.0961	0.2700
4000	0.0723	0.1150	0.0008	0.0778	0.3300
5000	0.0917	0.0646	0.0006	0.0715	0.3900

Table 5: The test results after each 1000 iterations on the lobster dataset.

Original				
Iterations	degree	clustering	4orbits	spectral
1000	0.0028	0.0031	0.0049	0.0153
2000	0.0006	0.0000	0.0016	0.0161
3000	0.0008	0.0000	0.0017	0.0148
4000	0.0035	0.0000	0.0066	0.0173
5000	0.0027	0.0000	0.0052	0.0148
Angular				
Iterations	degree	clustering	4orbits	spectral
1000	0.0009	0.0015	0.0032	0.0161
2000	0.0038	0.0000	0.0060	0.0173
3000	0.0019	0.0005	0.0037	0.0154
4000	0.0043	0.0014	0.0096	0.0166
5000	0.0038	0.0001	0.0079	0.0154
Hidden50				
Iterations	degree	clustering	4orbits	spectral
1000	0.0404	0.2574	0.0792	0.0316
2000	0.0014	0.0026	0.0072	0.0152
3000	0.0035	0.0000	0.0088	0.0146
4000	0.0030	0.0002	0.0053	0.0162
5000	0.0051	0.0011	0.0111	0.0171
2-Layer RNN				
Iterations	degree	clustering	4orbits	spectral
1000	0.0211	0.0345	0.0366	0.0154
2000	0.0040	0.0093	0.0105	0.0136
3000	0.0033	0.0004	0.0082	0.0157
4000	0.0033	0.0011	0.0067	0.0158
5000	0.0016	0.0000	0.0027	0.0165

Table 6: The test results after each 1000 iterations on the grid dataset.

Original				
Iterations	degree	clustering	4orbits	spectral
1000	0.1221	0.4885	0.3149	0.0155
2000	0.0301	0.4026	0.4887	0.0092
3000	0.3564	0.3636	1.0662	0.0856
4000	0.1436	0.3803	0.8973	0.0157
5000	0.2888	0.3836	1.0662	0.0452
Angular				
Iterations	degree	clustering	4orbits	spectral
1000	0.0685	0.5161	0.4263	0.0216
2000	0.2464	0.4755	1.2893	0.0118
3000	0.3467	0.3655	1.0490	0.0990
4000	0.1051	0.3841	0.3054	0.0112
5000	0.0833	0.4182	0.8853	0.0120
Hidden50				
Iterations	degree	clustering	4orbits	spectral
1000	0.3451	0.4641	1.3916	0.0240
2000	0.0461	0.3997	0.4887	0.0112
3000	0.1372	0.4103	1.0581	0.0152
4000	0.1110	0.3987	0.7279	0.0116
5000	0.1185	0.4353	0.9846	0.0094
2-Layer RNN				
Iterations	degree	clustering	4orbits	spectral
1000	0.5543	0.3900	1.0369	0.2119
2000	0.4028	0.4503	1.0652	0.1586
3000	0.2914	0.3490	1.0662	0.0756
4000	0.1067	0.4630	0.9860	0.0177
5000	0.0992	0.4931	0.8176	0.0245

Table 7: The test results after each 1000 iterations on the point cloud dataset.

Original				
Iterations	degree	clustering	4orbits	spectral
1000	0.0451	0.0677	0.6618	0.0093
2000	0.0199	0.0899	0.0762	0.0145
3000	0.0084	0.0655	0.0436	0.0086
4000	0.0086	0.0690	0.0273	0.0069
5000	0.0120	0.0838	0.0447	0.0087
Hidden50				
Iterations	degree	clustering	4orbits	spectral
1000	0.0221	0.1156	0.7958	0.0275
2000	0.0994	0.0767	0.8452	0.0197
3000	0.0075	0.0942	0.4716	0.0163
4000	0.0672	0.0643	0.5561	0.0141
5000	0.0246	0.0762	0.5619	0.0128
2-Layer RNN				
Iterations	degree	clustering	4orbits	spectral
1000	0.0374	0.1177	0.1902	0.0210
2000	0.0121	0.0789	0.4919	0.0103
3000	0.0079	0.0865	0.2796	0.0121
4000	0.0140	0.0938	0.1319	0.0122
5000	0.0141	0.1025	0.0990	0.0112
Angular				
Iterations	degree	clustering	4orbits	spectral
1000	0.0346	0.0746	0.5707	0.0134
2000	0.0399	0.0640	0.5984	0.0072
3000	0.0023	0.0598	0.0311	0.0076
4000	0.0103	0.0534	0.0752	0.0082
5000	0.0049	0.0419	0.0478	0.0062

Table 8: The test results after each 1000 iterations on the protein dataset.