Final Project Report
Evaluating graph layouts using Graph Neural Networks
ECS 289H

Sam Lee

March 12, 2022

1 Motivation and Objectives

With the recent advances of deep learning, several works[1, 6] have been proposed to evaluate graph layouts using deep learning approaches. However, they both use CNN-based models and thus have weak generality in terms of layout styles. The model has to be retrained with new datasets if the targeted layouts is changed. Even minors changes such as node color or edge width will require the retrain.

To overcome the weak generality of CNN models, in this project I use Graph Neural Networks(GNNs) to evaluate the graph layouts. Node coordinates are used to represent layouts instead of images. Using node coordinates have several benefits. The first is that the generality of the model is significantly improved, because the model does not require retraining if layout styles are changed. Second is that using node coordinates enables the usage of GNN models, which usually have shorter training time comparing to CNNs, as the number of parameters is significantly less. Moreover, previous CNN-based works can not explicitly take graph connectivity into account. Since GNNs are trained on the graph adjacent matrix, GNN-based approaches can evaluate the layouts based on the graph connectivity features.

2 Related Work

2.1 Graph Layout Evaluation

Evaluation of graph layouts is a significant problem in graph drawing. Traditionally, researchers either uses HCI methodology to conduct *qualitative* evaluation, or uses aesthetic criteria(or readability metrics) to conduct *quantitative* evaluation. Several works[1, 6] have been proposed to evaluate graph layouts using deep learning approaches. [1] tries to predict human preferences on graph layouts to conduct qualitative evaluation. [6] tries to predict readability metrics on graph

layouts to conduct quantitative evaluation. Both works focus on aesthetic criteria and use Convolutional Nerual Networks(CNN) to capture the aesthetic features in the layout image. However, CNN models tend to have relatively weak generality. Changes in the layout styles, e.g. layout algorithms or node color, requires the model to be re-trained. To make things worse, CNN models usually have long training time when evaluating graph layouts, because of the relatively large image size. In [6] the model is trained on images of size 325×260, and the training process took four days to complete 20 epoches. Since there are hundreds of graph layout algorithms, the generality on layout styles is a serious issue.

2.2 Graph Neural Networks

Graph Neural Networks (GNN) are deep learning models aiming at addressing graph-related tasks[13], such as graph classification, node classification, and link prediction. Most GNN models handles graph-structure data by performing the *graph convolution* operation. Similar to the convolution defined in CNNs, graph convolution aggregates neighbor information to generate the target node's representation.

GNN models have been used in the field of graph visualization by previous works[12, 8]. Both works use GNNs to generate graph layouts. [12] uses a typical GNN defined by several graph convolution layers and a loss function that explicitly considers optimizing several aesthetic metrics. Whereas in [8] a Graph Autoencoder (GAE) architecture is designed to generate a new layout from a set of given layouts. GAEs are unsupervised learning frameworks that encode nodes/graphs into a latent vector space and reconstruct graph data from the encoded information. This project is more similar to [8], where a GAE is used to learn the complex relationship between the structure and the layouts of a graph.

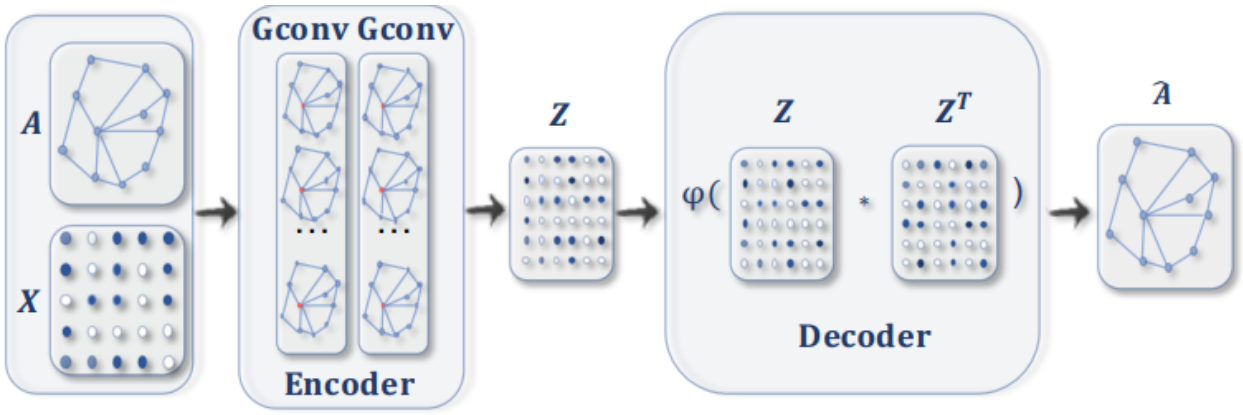
3 Design and Methodology

3.1 Problem Definition

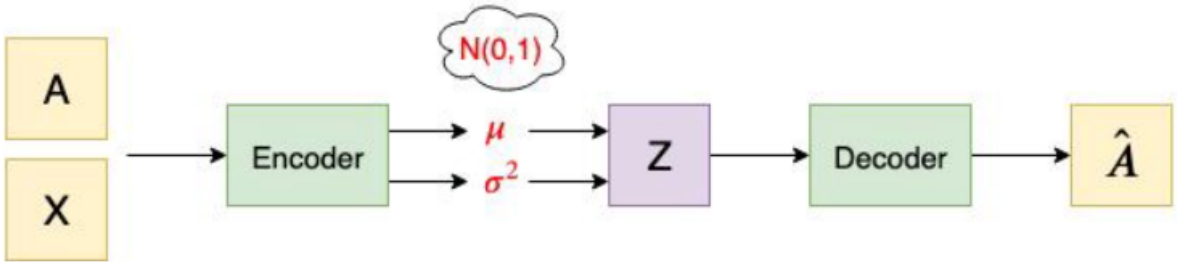
Given an undirected, unweighted graph $G = (V, E)$, we first introduce an adjacent matrix A for G. Then a set of layout algorithms L is applied on G. For each $l_i \in L$, the output $l_i(G)$ can be represented as a node position matrix X_i, where each row is (v_x, v_y), the position of v in 2d space given by l_i.

Then, the goal is to generate a score s_i for each layout. To do that, we first generate the adjacent matrix A' of a subgraph $G' = (V, E')$ with $E' \subset E$, i.e. all the nodes in G are preserved while some edges are missings. Also, we generate a node feature matrix X_i for each $l_i \in L$. Then, a GAE model is used to reconstruct G from G' using X_i. The output of the reconstructed graph is denoted as $\hat{A}_i = GAE(A, X_i)$. Finally, $s_i = recon\_loss(A, \hat{A}_i)$, where $recon\_loss()$ denotes the reconstruction loss defined in the GAE model.

Basically, we first remove some edges in G to generate G', and let the GAE model to reconstruct



(a) Architecture of GAE



(b) Architecture of VGAE

G from G' using the layout features. The score S is thus the reconstruction loss. Note that to give a score for a layout, previous works often take into account the aesthetics metrics. Of course, the score would be more accurately conforming to human preferences if the aesthetics metrics are considered. Here we deliberately not consider that, because we want to articulate the effectiveness of using the reconstruction loss for the layout scroes. In Section 5.2, the correlation between the reconstruction loss and aesthetics metrics is evaluated.

3.2 Model Architecture

The model architecture follows the framework proposed in [7]. The two-layer graph convolution operation is defined as $GCN(\mathbf{X}, \mathbf{A}) = \tilde{\mathbf{A}} ReLU(\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}_0)\mathbf{W}_1$, where $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is the symmetrically normalized adjacency matrix and $\mathbf{W}_i$ are wight metrices.

Based on this operation, a simple non-probabilistic GAE is first proposed. The encoder is simply a two-layer GCN: $\mathbf{Z} = GCN(\mathbf{X}, \mathbf{A})$, where Z is the matrix of latent variables $\mathbf{z}_i$. The decoder reconstructs $\hat{\mathbf{A}}$ by: $\hat{\mathbf{A}} = \sigma(\mathbf{Z}\mathbf{Z}^T)$, where $\sigma(\cdot)$ is the logistic sigmoid function. The binary cross-entropy loss function $L = E_{q(Z|X,A)}[logp(A|Z)]$ is used to represent the reconstruction loss between $\mathbf{A}$ and $\hat{\mathbf{A}}$. Figure 1a[1] demonstrates the architecture of GAE. This architecture will be refered to as GAE in later sections.

[1]This figure originates from [13]

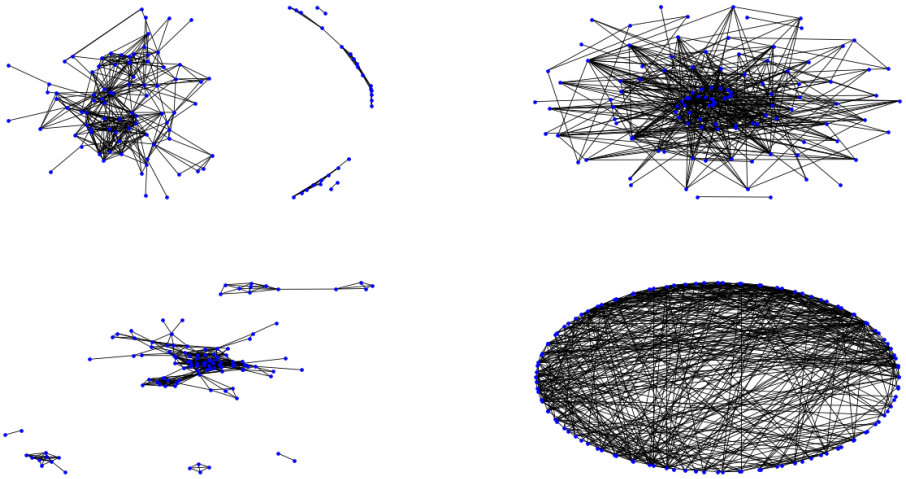


Figure 1: Four selected layouts to evaluate. (1) Top left: MDS, (2) Top right: spiral, (3) Bottom left: spring, (4) Bottom right: circular

[7] also proposes a Variational Graph Autoencoder (VGAE) architecture. The VGAE encoder is also a two-layer GCN. The first layer is exactly the same as GAE. But the second layer now performs two seperate GCN operation, and generates two variables $\mu = GCN_{\mu}(\mathbf{X}, \mathbf{A})$ and $log\sigma^2 = GCN_{\sigma}(\mathbf{X}, \mathbf{A})$. Then $\mathbf{Z} = \mu + \sigma * \varepsilon$, where $\varepsilon \sim N(0, 1)$ is the standard normal distribution. The decoder is the same as GAE. The loss function, in addition to the reconstruction loss defined in GAE, adds a new part $KL[q(Z|X, A)||p(Z)]$, which is the KL-divergence between $q(Z|X, A)$ and $p(Z) \sim N(0, 1)$. This part measures how closely $q(Z|X, A)$ matches $p(Z)$. Figure 1b[2] demonstrates the architecture of VGAE. The two architectures will both be tested upon and evaluated by the aesthetic metrics correlation described in Section 5.2.

4 Implementation

4.1 Model and Layout

Pytorch Geometric (PyG)[4] is used for the implementation of the model architecture described in Section 3.2. PyG provides implementation of many state-of-the-art GNN models, including the VGAE model described in Section 3.2. For the layout algorithms, NetworkX[5] is used, as it provides a variety of efficient layout algorithms and graph data manipulation and analysis utils. Three layout algorithms provided by NetworkX are chosen to be the targeted layout algorithms,

[2]This figure originates from `https://towardsdatascience.com/tutorial-on-variational-graph-auto-encoders-da9333281129`

Name	$\|V\|$	$\|E\|$	$\|E\|/\|V\|$
0	333	5038	15.1
414	150	3386	22.6
3980	52	292	5.6
698	61	540	8.8

Table 1: Statistics for test dataset.

namely the spiral layout, the spring layout and the circular layout. Additionally, the MDS layout implementation provided by scikit-learn[10] is chosen as well. In total, four layout algorithms are used in the project. Figure 1 shows an example of the four layouts.

4.2 Datasets

The dataset used in this project is collected in [9]. The dataset consists of 'circles' from Facebook. Each 'circle' is an ego-network, i.e. all the nodes are connected to an ego-node. In the project, the ego-node will be ignored and the ego-network is treated simply as an undirected social-network graph with no node/edge attributes. Instead, the node attributes will be replaced with the normalized node coordinates. Since we have four targeted layout algorithms, each network will generate four different graphs, with the same adjacent matrix A but differnt node position matrices X_i as node attributes, as mentioned in Section 3.1.

In the original dataset the graph size in terms of node numbers ranges from $\sim$ 50 to $\sim$ 500. To guarantee the reconstruction accuracy, the large graphs are sub-sampled into several smaller subgraphs. Eventually, all the graphs in the dataset have an average of 135 nodes and 1183 edges. The final dataset consists of 30 graphs (120 layouts), in which 26 graphs are used as training dataset and 4 graphs are used as test dataset. Table 1 shows the statistics of the four test graphs.

5 Evaluation

5.1 Reconstruction Efficacy

The efficacy of reconstructing adjacent matrix from node coordinates is not shown by previous studies. To prove the efficacy, a link prediction is conducted by the model. As described in Section 3.1, the model reconstructs $G = (V, E)$ from a subgraph $G' = (V, E')$ with $E' \subset E$. The model trys to reconstruct the edges from existing edges and node features, which are node coordinates in this case. The missing edge rate is set to 50 percent, i.e. only 50 percent of edges is remaining, and the model will reconstruct the missing 50 percent edges. Table 2 presents the link prediction accuracy on the test dataset. Left in the cell is accuracy on GAE and right is VGAE. The average accuracy is **0.769** for GAE and **0.776** for VGAE, proving that reconstructing adjacency matrix from node coordinates is viable.

graph \ layout	MDS	spiral	spring	circular
0	0.827/0.834	0.832/0.837	0.826/0.832	0.847/0.851
414	0.767/0.782	0.745/0.753	0.740/0.746	0.776/0.782
3980	0.724/0.729	0.723/0.733	0.732/0.746	0.778/0.790
698	0.687/0.671	0.752/0.758	0.785/0.794	0.774/0.782

Table 2: Link prediction accuracy on GAE/VGAE. Each row is a test graph and each column is a layout algorithm. Each cell is accuracy. Left represents GAE and Right represents VGAE.

5.2 Correlation to Readability Metrics

To evaluate the ranking result, and therefore the efficacy of the project, we investigates the correlation between test reconstruction loss and aesthetic metrics. Three aesthetic metrics are used: (1)M_l: *edge length variation* (2) M_a: *Minimum Angle* (3) E_c: *Edge crossings*.

Equation 1 presents the calculation for M_l. l_e refers to the length of e, l_μ refers to the mean of edge length for $e \in E$.

$$M_l = l_a/\sqrt{|E|-1}, \quad l_a = \sqrt{\sum_{e \in E}(l_e - l_\mu)^2/(|E| * l_\mu^2)} \tag{1}$$

Equation 2 presents the calculation for M_a. $\phi_(v)$ is the minimum incident angle of each edge, $\phi_{min}(v)$ is the ideal minimum angle. $deg(v)$ means the degree(number of edges) of v.

$$M_a = 1 - \frac{1}{|V|}\sum_{v \in V}\frac{\phi(v) - \phi_{min}(v)}{\phi_{min}(v)}, \quad \phi_{min}(v) = 360/deg(v) \tag{2}$$

The correlation score $coef$ is calculated by the Pearson correlation coefficients. A score of 1 means strong positive correlation, -1 means strong negative correlation, and 0 means no correlation. Since the three aesthetic metrics are expected to be minimized as well as the reconstruction loss, a $coef$ score of 1 should indicate a better performance. Table 3 shows the correlation score between reconstuction loss and the three aesthetic metrics on four test graphs for GAE/VGAE. Overall VGAE has better $coef$ scores. On test graph #0,#414 and #698, there is a strong correlation between the reconstruction loss and one of the aesthetic metrics, as indicated by the highlighted score. On graph #698 there is a strong correlation between reconstruction loss and both M_l and M_a. For GAE, however, there is no observable strong correlation on all three metrics. Only one strong correlation on M_l on graph #3980 can be found. Interestingly VGAE is not performing well on graph #3980. Looking back to the statistics of graph #3980 shown in Table 1, the $|E|/|V|$ is significantly lower than the other graphs. As mentioned in Section 4.2, the average $|E|/|V|$ is ~ 10 in the train dataset. Therefore #3980 is probably an outlier and no similar graph is seen by the model in the train dataset.

Another interesting observation is that the model seems to be able to 'choose' a probable metric for different graphs. There is correlation on M_l for graph #0, M_a for graph #414, and both M_l

graph \ metric	M_l	M_a	E_c
0	-0.648/**0.904**	-0.530/0.612	0.604/0.040
414	-0.425/0.479	0.147/**0.861**	-0.425/-0.289
3980	**0.816**/0.437	-0.943/-0.695	-0.684/-0.985
698	0.873/**0.954**	0.937/**0.984**	-0.428/-0.001

Table 3: Pearson correlation coefficients on GAE/VGAE. Each row is a test graph and each column is an aesthetic metric. Each cell is $coef$. Left represents GAE and Right represents VGAE.

and M_a for graph #698. It is understandable that the importance of different aesthetic metrics would be differnt for different graphs, and it is hard to choose which metric to use beforehand. This shows that the model is able to find the aesthetically best layout based on the graph connectivity features, even though in the architecture no aesthetic metric is taken into account.

6 Discussion

6.1 Comparison to previous works

The evaluation presented in Section 5 shows that the model is able to rank the layouts that have the best aesthetic metrics. The results have proven that using GNNs to evaluate graph layouts is probable. Compared to previous works[1, 6] that used deep learning apporoaches to evaluate graph layouts, this project has many significant advantages. First is the training time. It is acknowledged that CNNs require an extremely long training time, especially when the image size is large. However, for graph layouts, the image size is inevitably large. In [6], even though the edge width is as small as 0.5 pixel, the node size still needs to be 8 pixel in radius. As a result, there is only a limited amount of nodes that can be contained in a fixed-size image, or from another perspective, a graph layout intrinsically has a limitation on the minimum image size. In the end, CNN-based approaches have to make a trade-off between image size(training time) and graph size. GNN-based apporoaches takes a significantly lower training time. Even though the training time scales up with the graph size, it is still significantly lower, and most importanly, there is no limitation on the graph size.

Another advantage is the generality of the model. CNN-based approaches need to be retrained on different layouts, even when the difference is only minor changes, e.g. node color. Considering the amount of time to train a CNN, it becomes a serious issue. Whereas the model in this project only needs to be retrained when the targeted layout algorithm is changed, which significantly affects the node position features. Compared to [6], which is forced to focus on force-directed layouts because of the generality issue of CNN, this project can simultaneouly evaluate four layouts once trained, and theoratically there is no limitation on the amount of layouts to evaluate.

6.2 Limitations and future works

As a class project, there is obviously many limitations in terms of model complexity and dataset size, and many ideas have to be cut off because of the short amount of time. First of all, the model is almost the same as proposed in [7]. The node positions, which is the core difference between this project and previous works, is treated as simply node features. Originally there is another idea of using *Geometric GNNs*[3, 11, 2], which directly incorporates node positions into the model. These kinds of models are originally designed for molecule researches, where the relative position of the nodes(atoms) or the edge(bonds) length significantly affects the molecule(graph) properties. However, it turns out that it would take too much time to integrate the GCN layers proposed in these works into the GAE model, i.e. replace the GCN layers. I believe implementing this method is a probable future direction.

In addition, the dataset size is probably too small for the model to perform well. As we can see in Section 5.2, the model is not performing well on a certain graph because the characteristics of the graph is not represented in the train dataset. Theoratically the model is able to be trained on graph of any size, and the performance can be further improved on a larger dataset with more diverse graphs. Also, the amount of targeted layouts is not limited. One limitation on the layouts, however, is that the train dataset needs to include samples of the targeted layout algorithms. It was observed in preliminary experiments on earlier stages in this project, that the reconstruction loss is very high if no sample of a layout algorithm exists in the train dataset. This means that the model would need to be retrained if there is a new targeted layout algorithm. It remains unknown if this limitation is intrinsic to GNNs or if there is a solution and further investigations are required.

Finally, in this project we deliberately do not consider incorporating aesthetic metrics in the scoring process. This was to prove that the reconstruction loss alone is able to produce reasonable scoring results. Now that the efficacy is proven, a more accurate scoring system can be developed by incorporating aesthetic metrics explicitly.

7 Conclusion

Evaluating graph layouts with deep learning approaches has the potential to overcome limitations in traditional methods, but new challenges are also introduced. As intuitive as it is to use CNN-based approaches for evaluting graph layouts, the weak generality as well as long training time is inevitable. This project is an attempt to use GNNs for graph layout evaluation. The layouts are represented by node coordinates, and are used as node features in GNN models. The evaluation scores are given by the reconstruction loss from a subgraph to the original graph using a Graph autoencoder architecture. As a class project, there are limitations on model complexity and dataset size, but the results show that the model is already able to find the aesthetically best layout even though it is not explicitly trained to do so. Potential future works include designing a more suitable model, increasing dataset size, and explicitly incorporating aesthetic metrics in the scoring process.

References

[1] Shijun Cai, Seok-Hee Hong, Jialiang Shen, and Tongliang Liu. A machine learning approach for predicting human preference for graph layouts. In *2021 IEEE 14th Pacific Visualization Symposium (PacificVis)*, pages 6–10. IEEE, 2021.

[2] Daniel T Chang. Geometric graph representations and geometric graph convolutions for deep learning on three-dimensional (3d) graphs. *arXiv preprint arXiv:2006.01785*, 2020.

[3] Hyeoncheol Cho and Insung S Choi. Three-dimensionally embedded graph convolutional network (3dgcn) for molecule interpretation. *arXiv preprint arXiv:1811.09794*, 2018.

[4] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[5] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[6] Hammad Haleem, Yong Wang, Abishek Puri, Sahil Wadhwa, and Huamin Qu. Evaluating the readability of force directed graph layouts: A deep learning approach. *IEEE computer graphics and applications*, 39(4):40–53, 2019.

[7] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[8] Oh-Hyun Kwon and Kwan-Liu Ma. A deep generative model for graph layout. *IEEE Transactions on visualization and Computer Graphics*, 26(1):665–675, 2019.

[9] Jure Leskovec and Julian Mcauley. Learning to discover social circles in ego networks. *Advances in neural information processing systems*, 25, 2012.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[11] Przemysław Spurek, Tomasz Danel, Jacek Tabor, Marek Smieja, Lukasz Struski, Agnieszka Slowik, and Lukasz Maziarka. Geometric graph convolutional neural networks. *arXiv preprint arXiv:1909.05310*, 2019.

[12] Xiaoqi Wang, Kevin Yen, Yifan Hu, and Han-Wei Shen. Deepgd: A deep learning framework for graph drawing using gnn. *IEEE Computer Graphics and Applications*, 41(5):32–44, 2021.

[13] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.