CSN Project - Graph Neural Networks and network metrics

Students:
Wangyang Ye,
Pau Rodríguez Esmerats

FIB UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Date:
January 23, 2019

Abstract

A Graph Neural Network (GNN) is a representation learning model that learns to represent graph structure in a low-dimensional embedding space. Graph Neural Networks and its recent variations belong to a group of techniques ranging from summary graph statistics to matrix factorization and random-walk based methods that aim to summarize graph properties by extracting features that are later on used on well-known machine learning tasks, usually to perform node or sub-graph classification or link prediction. In this project, we will use the Graph Convolutional Network (GCN) model to study how an embedding related to the PageRank metric is generated. We will also study the effect of combining different graph metrics and node features when generating the resulting embedding aimed at node classification, and possibly explore the application of another Graph Neural Network model in a trivial sub-graph classification task.

1 Introduction

A Graph Neural Network is a representation learning model that learns to represent graph structure in a low-dimensional embedding space. The first Graph Neural Network model (Scarcelli et al. [1]) reads a graph with its nodes, their attributes and their edges as the input of a neural network that mimics the graph structure, using iterative convergence mechanism in the forward and backward passes to update unit weights. Many different variations have appeared since then. They all approach different aspects like training method, node neighborhood information representation as well as type of resulting embedding (node, edge, sub-graph or graph embedding) in different ways. For example, Graph Convolutional Network does not require an iterative convergence mechanism in forward and backward passes in the training steps.

Moreover, those models belong to a group of techniques that aim to summarize graph properties by extracting features. They go from summary graph statistics, kernel functions and hand-engineered features, to shallow unsupervised embeddings based on matrix factorization and random-walk based techniques, to encoder-decoder architectures and more recently to the Graph Neural Networks and their variants. These series of techniques can be grouped by the embedding targets which are: node embeddings, aggregation node embeddings, structural role embeddings, subgraph embeddings and edge embeddings.

The main goal of those models is to perform node, subgraph or graph classification and link prediction. They usually learn an embedding that is later used in a well-known machine learning algorithm for classification, regression or clustering. The idea is that geometric relations in the embedding space correspond to interactions, structure or features in the original graph.

Many fields can benefit from Graph Neural Network models. Mainly all tasks that need to perform node or sub-graph classification and link prediction will benefit from GNNs, e.g. recommender systems (missing friendship links, affinities between users and content), computational biology (protein interaction graphs that are incomplete or noisy), program understanding (satisfaction of properties), statistical relational learning (predict missing relations between entities in a knowledge graph).

In this project, we will use the Graph Convolutional Network model to study how an embedding related to the PageRank metric is generated. We will also study the effect of combining different graph metrics and node features when generating the resulting embedding aimed at node classification, and possibly explore the application of another Graph Neural Network model in a trivial sub-graph classification task.

Section 2 of this report will summarize the current representation learning techniques on graphs with a very basic comparison between them and then explain the important details of the selected models with which experiments are performed. Section 3 will explain the experiments performed by stating their goal, the data set used and the expected results. Section 4 will present the results of the experiments and section 5 will discuss the differences between the results and our initial expectations and/or assumptions. In section 6 we will conclude the report by reviewing the most important results and state in what direction future work could be oriented.

2 Related Work

This section reviews the different kinds of models that perform representation learning on graphs. We start by giving a characterization of different types of results that a model could aim for and the similar alternative techniques that exist, and then we enumerate the most significant Graph Neural Network derived models.

Graph Neural Network (GNN) models started a saga with a great number of models that learn representations of graph structure and node features. There were already several good approaches with the same goal of representing information of a graph by extracting features from it. Those approaches range from traditional summary graph statistics, kernel functions and hand-engineered features, that are non-adaptable and time consuming, to more advanced low-dimensional embedding techniques based on matrix factorization and random-walk based approaches. A important way to characterize all these techniques is by the specific graph information they aim to summarize:

- **node embeddings** - summarize the node's graph position and structure of their local graph neighborhood. Some approaches are Graph Factorization and Laplacian Eigenmaps (Matrix Factorization approaches), DeepWalk and Node2vec (Random-walk approaches) and Deep Neural graph representation, Structural Deep Network Embeddings (encoder-decoder approaches).
- **aggregation node embeddings** - rely on node attributes and aggregated information from its local neighborhood. Some approaches are Graph Convolutional Networks, Column Networks and GraphSAGE (graph neural network approaches).
- **structural role embeddings** - repesentations that correspond to the structural roles of the nodes, independent of their global positions, e.g. communication or transportation networks. Some approaches are node2vec with some biased random-walk mechanism, struct2vec and GraphWave.
- **subgraph embeddings** - encode a set of nodes and edges into a low-dimensional vector embedding. Possible approaches are averaging node embeddings or introducing a virtual node that represents the whole subgraph. The latter approach is used in the original Graph Neural Networks, in Message Passing Neural Networks and in Graph Attention Networks.
- **edge embeddings** - edge embedding aims to represent an edge as a low-dimensional vector. Some approaches include node2vec with modifications and knowledge graph embedding [2].

Another characterization that can be made is to separate Spectral models from Spatial models. Spectral models have a foundation in signal processing, and implement graph signal filters. Those models are limited to undirected graphs, are more costly in terms of efficiency with increasing graph sizes and tend to generalize poorly with new or different graphs. Examples of those models are Spectral Graph CNN, ChebNet. Spatial methods imitate the convolution operation of a conventional convolutional neural network on images, to define a graph convolution based on a node's spatial relations. It takes the aggregation of the central node representation and its neighbors representation to get a new representation for this node, usually stacking multiple graph convolution layer together. Further division exists between recurrent-based and composition-based spatial GCNs. Recurrent-based methods apply same graph convolution layer to update hidden representations, while composition-based methods apply different graph convolution layer to update hidden representations.

The main idea of GNN is to generate node embeddings by the means of aggregating neighborhood information. There are many variants of GNN and the main difference among them are the approaches on aggregating neighbors' messages.

- Graph Neural Network [1] [3]: as a basis, GNN generate node embeddings using local neighborhood by averaging messages from neighbors and applying neural networks.
- Graph Convolutional Network (GCN): in [4], a slight variation on the neighborhood aggregation idea is proposed. Instead of simple average, a normalization is carried out across neighbors in a similar way to how convolution operations are carried out in image data.
- Graph Attention Networks [5]: similar to GCN it uses an aggregation function on neighborhood information but attention mechanisms are used to assign larger weights to the more important nodes. The attention weights are learned together with neural network parameters.

- Gated Graph Neural Network (GGNN) [6]: it is a recurrent-based spatial GCN that uses back-propagation through time to learn the parameters. It does not need to constrain parameters to ensure convergence but has a higher cost in time and memory resources.
- Graph Sampling and Aggregation (GraphSAGE) [7]: it uses an aggregation function to define the graph convolution. This aggregation function essentially assembles a node's neighborhood information, in a way that is invariant to permutations of node orderings. Training is performed in a batch-style training.
- Message Passing Neural Network (MPNN) [8]: generalizes several existing graph convolution networks into a unified framework named Message Passing Neural Networks. They consist of two phases, the message passing phase (where many graph convolutions are run) and the readout phase, actually a pooling operation that represents the graph based on hidden representations of each individual node.

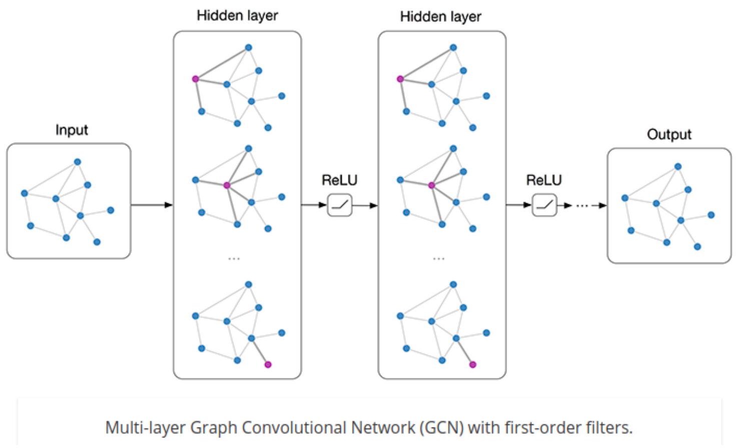
Hidden layer
Hidden layer
Input
ReLU
ReLU
Output
Multi-layer Graph Convolutional Network (GCN) with first-order filters.

Figure 1: Graph Convolutional Network (credit Thomas Kipf)

We choose to study the Graph Convolutional Network because it's the most efficient in terms of computational cost. This model learns a function of signals/features on a graph G = (V,E) which takes as input a feature description x_i for every node i, summarized in a N x D feature matrix X (N: number of nodes, D: number of input features), and an adjacency matrix A. The output of the network is a node-level output Z (N x F feature matrix, where F is the number of output features per node). Every neural network layer can be written as :

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

with $H^{(0)} = X$ and $H^{(L)} = Z$, L being the number of layers and

$$f(H^{(l)}, A = \sigma(AH^{(l)}W^{(l)})$$

where W is a weight matrix for the l-th neural network layer and $\sigma(\Delta)$ is a non-linear activation function.

The Graph Convolutional Network consists of several hidden convolutional layers, which have several filters that consists of parameters that are shared over all locations in the graph or a subset of it and that represent some features of the data. Those hidden layers are usually calling a ReLU activation function, except for the last layer that uses an identity activation function and a softmax function to compute the probabilities of each class.
The greatest advantage of this model is that it doesn't need an iterative convergence procedure. It also swaps spectral convolution for spatial convolution and it can be understood as a generalized version of the Weisfeiler-Lehman algorithm on graphs.

3 Methodology

Four different experiments are considered in this study:

- Node classification: create a graph or find a dataset and choose a clustering method to group members into clusters. Train the GCN model to classify nodes to one of the clusters. We could generate small graphs that are easy to plot, and then also plot the resulting embeddings (obviously slices of 2D planes of the embedding only) and see whether the position of the embeddings make sense.
- Exploration of embeddings: Pick different families of networks, and explore their embeddings under some metrics.
- Transferability: Are embeddings transferable accross different "problems", e.g. betweenness and other types of centrality measures?
- Embedding for prediction on new graph: are these models useful to approximate metrics, in the following sense: once we learn an embedding, can we use it to approximate the value of some metric for new graphs that were not part of the training set?

3.1 Analysis

Our experiments will consist in performing the comparison of the resulting embedding both graphically and numerically, when using a Graph Convolutional Network trained for approximating a graph local metric. We will look for similarities and differences of resulting embeddings when applied to different networks and metrics.

We will also modify the Graph Convolutional Network to include an embedding of only 2 dimensions. To do that, we will add 2 layers, one with only 2 units which is the one responsible for extracting an embedding in 2 dimensions. Another one after as the output with the same number of layers as target classes,playing the role of output layer which will allow to train the network with the corresponding loss when comparing to supervised classes.

3.2 Data

We will generate different kinds of networks (with different models, for instance: ER, BA, Watts-Strogatz, etc.) with different sizes and we will compute different network metrics (PageRank, centrality measures, clustering coefficients, etc) on each of them.

4 Results

All the computations needed for obtaining the results are available from the Jupyter Notebook that follows this report, which is also available online at https://colab.research.google.com/drive/1ia1qc2x8TTxnq3KjPxg2IGcBFd3BiK5A
.

4.1 Node classification

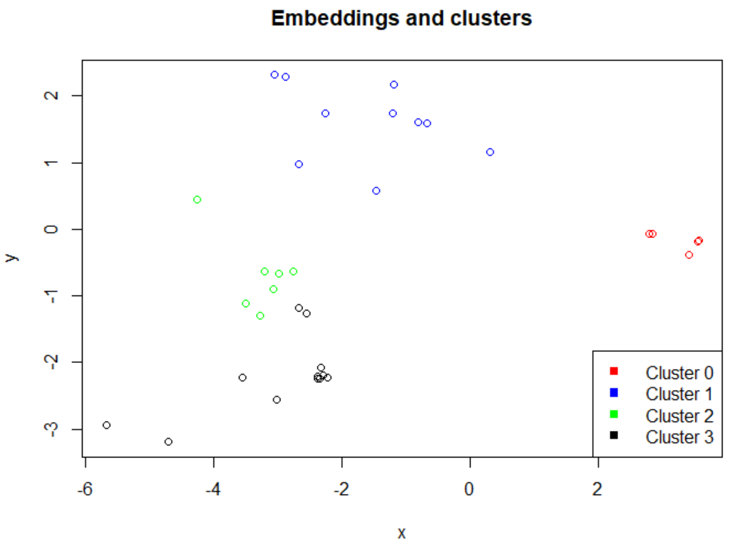

Figure 2: Plot of the first 2 dimensions of the embeddings and the corresponding clusters

4.2 Exploration of embeddings

The exploration of embeddings has been performed by printing and comparing the values of the resulting embedding after training a Graph Convolutional Network (GCN) for four different metrics. The standard architecture of the GCN is aimed at doing multiclass classification, so in order to use it with real metrics we transform the metric value into one of 10 ranges covering from the maximum to the minimum of the observed metric values within the data.

Erdös-Rényi

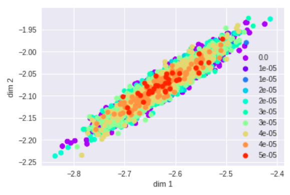

Figure 3: PageRank embedding

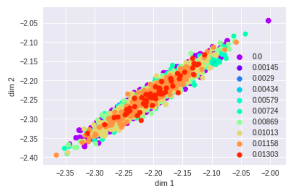

Figure 4: Closeness Centrality embedding

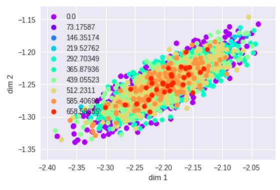

Figure 5: Betweenness centrality embedding

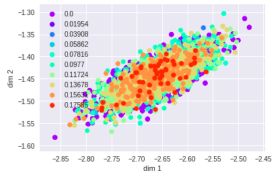

Figure 6: Eigenvector centrality embedding

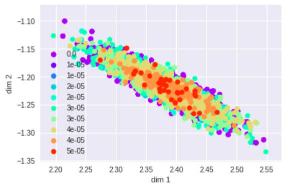

Figure 7: PageRank with 2D embedding

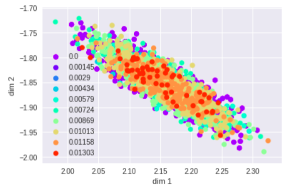

Figure 8: Closeness centrality with 2D embedding

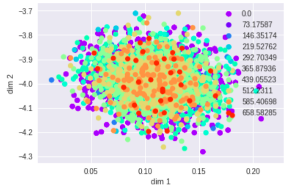

Figure 9: Betweennes centrality with 2D embedding

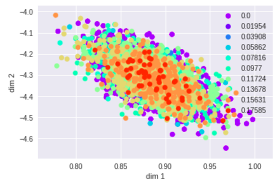

Figure 10: Eigenvector Centrality with 2D embedding

Watts-Strogatz

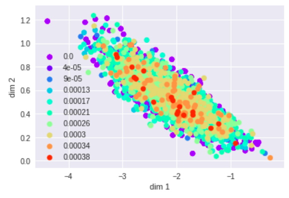
Figure 11: PageRank embedding

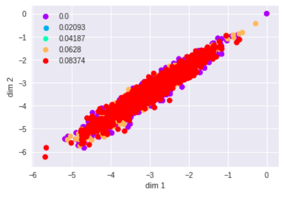
Figure 12: Closeness Centrality embedding

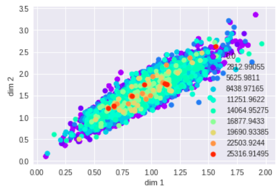
Figure 13: Betweenness centrality embedding

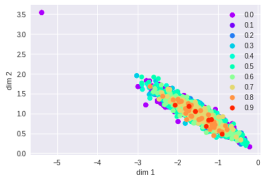
Figure 14: Eigenvector centrality embedding

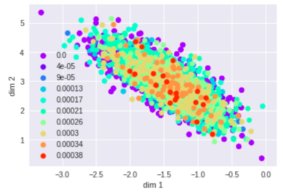
Figure 15: PageRank with 2D embedding

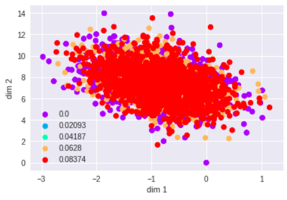
Figure 16: Closeness centrality with 2D embedding

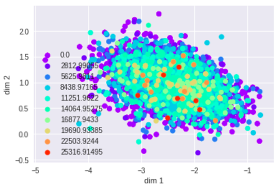
Figure 17: Betweennes centrality with 2D embedding

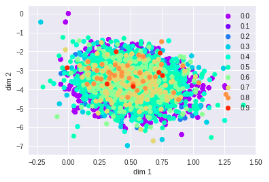
Figure 18: Eigenvector Centrality with 2D embedding

Barabasi-Albert

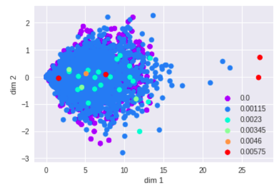
Figure 19: PageRank embedding

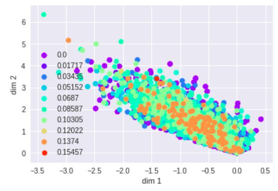
Figure 20: Closeness Centrality embedding

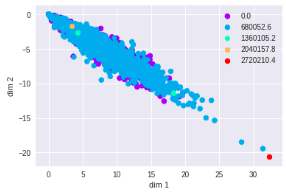
Figure 21: Betweenness centrality embedding

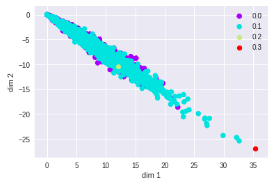
Figure 22: Eigenvector centrality embedding

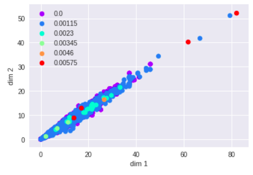
Figure 23: PageRank with 2D embedding

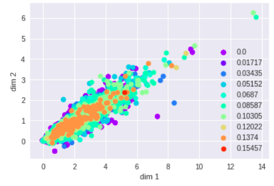
Figure 24: Closeness centrality with 2D embedding

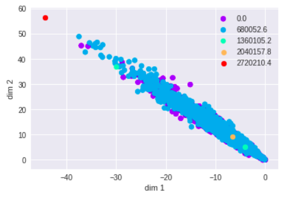
Figure 25: Betweennes centrality with 2D embedding

As we can observe through all the figures, for the big majority of Networks and metrics we could reuse a metric learnt for one specific Graph Convolutional Network. It seems possible to write a linear transformation from one embedding to another one. However there are some exceptions that may require also scaling (Betweenness centrality of Erdös-Renyí of gcn2 network, as well as Closeness , betweenness and eigenvector closeness from Watts-Strogatz model, and PageRank of the Barabasi Albert algorithm).

4.3 Transferability

As we have already seen in the previous experiment, the resulting embeddings of different metrics on a same graph are similar in terms of shape and orientation.

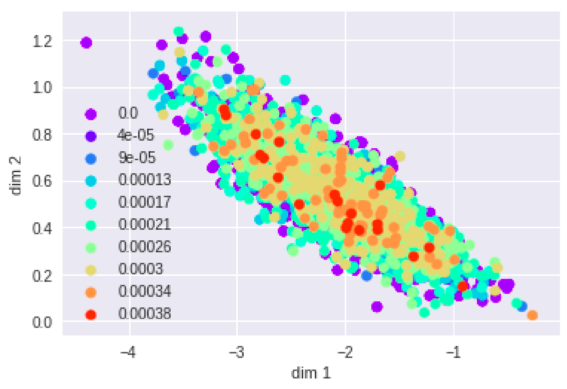

Figure 26: PageRank embedding for the Watts-Strogatz

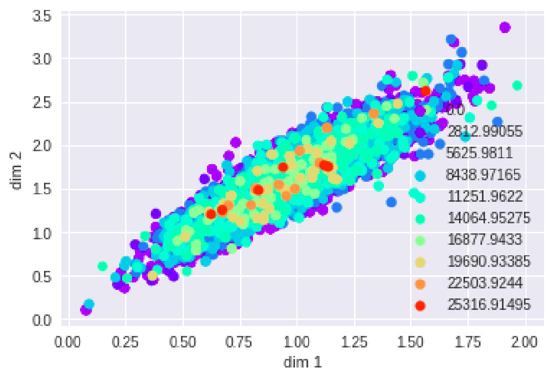

Figure 27: Closeness Centrality embedding for the Watts-Strogatz

PageRank (approximated by GCN)	Betweenness (real)
0.0001747608285346717	8734.044160196192
0.0001747608285346717	8473.320805228846
0.0001310706214010038	5669.8691317393195
0.0001747608285346717	3099.3710366424075
0.0001747608285346717	7997.7968411075435
...	...

Table 1: PageRank and Betweenness centrality

Let's compare the values of the PageRank embedding with the actual Betweenness Centralities of each node. As the plots of the embeddings seem similar, we could take the values of the PageRank, and appply centering and normalizing and then translation and scaling to obtain the corresponding values of the Betweenness centrality measure:

$$B_{transfered} = \frac{(Pr - avg(Pr))}{(std(Pr))} \cdot (std(B)) + avg(B)$$

Betweenness (linear transformation from PageRank)
29361.45659587
29361.45659587
-142665.6590039
...

Table 2: Betweenness from PageRank result by linear transformation

We can clearly observe that the linear transformation is no working, the Betweenness centrality values are very different from real values. The Root Mean Squared Error is very high: $RMSE = 58063.6$.
If we determine the correlation between the real Betweenness centrality and the learnied PageRank, we se correlation is **0.044895** which is low enough to say that there is no correlation between both values. In that case we would say that the transferability is not possible. Meaning that the embedding learn for one task cannot be transfered to another task unless it is transformed.

5 Discussion

5.1 Node classification

For simplicity and understandability, the graph dataset that we have used is the well-known Zachary's karate club. We have clustered the club members into different clusters using *optimal.community*() in *igraph* library which is a modularity based clustering method, in total we have obtained 4 clusters. We trained the neural network with few nodes and we performed the testing on the whole dataset. The accuracy we obtained was 94.12% which is quite good in the sense that almost all the members are correctly classified, and it's worth to mention that the only input feature that we have used was an identity matrix. Although the provided features ware very simple, surprisingly, the resulting embeddings were actually very good.

From figure 2, we can observe that the embeddings of the members of the same cluster are more closely located. Therefore, we could conclude that for karate dataset, with limited information (mainly the adjacency list), the GCN performed well on the node classification task.

5.2 Exploration of embeddings

For this experiment we expected to find GCN embeddings to differ between them, because they are expressed in the last layer of the GCN which is optimized for the metric in question. Furthermore we expected the special 2D embeddings to be more similar as they should be capturing features related to the nodes relationships with their neighbors. Also, this embedding should be more similar if compared networks have a similar topology.

The result is that almost all embedding results for specific metric are similar and they could be easily converted from one to the other. We observe most of them can be equivalent by just using a scaling transformation, that means that we could use the embedding obtained for computing the PageRank on Erdös-Renyí graph to obtain the Closeness centrality on a Barabasi-Albert graph by just applying a linear transformation (scaling and translation).

5.3 Transferability of embeddings across different metrics

After inspecting the plots of Embeddings of different metrics we observed that in terms of shape the embeddings of different metrics seemed very similar, but a linear mapping was not working well. We expected this to be correct but it turned out that the correlation coefficient is very low. That is why the linear mapping between an embedding for the PageRank metric to the Betweenness metric is not working well. We conclude the embeddings are not transferable.

5.4 Embeddings for prediction on new graph

In this experiment we want to figure out that once we learned an embedding for a graph, whether we can use it to approximate the value of PageRank for new graphs (under same model or not) that were not part of the training set. In order to do so, we have prepared two models which are Barabási Albert model and Watts–Strogatz model. We set to use part of the nodes of BA model to train the neural network and to obtain the corresponding embeddings. For the testing set, we have included nodes from both models. As we can from figure **??**, most of the nodes have PageRank classified to the class that has the smallest value which coincide with the original label of the nodes. We observed there are many purple points, fewer blue and light blue points and other colors are negligible, we could say that it's consistent with the distribution of the labels.

After repeating this experiment, although further analysis are needed in order to have a more systematical conclusion, we observed that the trained embeddings could be used to approximate the value of the metric for an unseen graph, despite the fact that the accuracy may not be stable due to the training instances and the provided features.

6 Conclusion

This project has allowed us to introduce ourselves to the world of Graph Neural Networks to some extent. We've read enough publications to get a solid idea of the current variants of Graph Neural Networks, their specific purposes and their state-of-the art performance.
We've chosen one specific variant, the Graph Convolutional Network (GCN), to study it's behaviour for learning/approximating Network metrics. The different experiments showed that the learned embedding by the GCN performs well for new unseen graphs but that this embedding is not transferable to different metrics. We also tried to modify the GCN model to produce an intermediate embedding in 2 dimensions, but the results didn't vary significantly. Further analysis that can be carried are the tests related to global graph metrics, but for this task a new algorithm is needed for instance Message Passing Neural Network or the original Graph Neural Network. Other study could consist in testing the applications to other tasks more complex thatn network metrics, for example knowledge graph reasoning, protein to protein interactions o recommender systems.

References

[1] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[2] Antoine Bordes, Jason Weston, Ronan Collobert, Yoshua Bengio, et al. Learning structured embeddings of knowledge bases. In *AAAI*, volume 6, page 6, 2011.

[3] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

[4] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[5] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 1(2), 2017.

[6] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[7] Z. Ying W. Hamilton and J. Leskovec. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems, pp. 1024–1034*, 2017.

[8] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017.

[9] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[10] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[11] Thomas Kipf. Graph convolutional networks. https://github.com/tkipf/gcn. Online; accessed January 2019.

[12] Hongyun Cai, Vincent W Zheng, and Kevin Chang. A comprehensive survey of graph embedding: problems, techniques and applications. *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[13] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019.