Influencer Detection with Dynamic Graph Neural Networks

Elena Tiukhova
Research Center for Information Systems Engineering
Faculty of Economics and Business
KU Leuven
Naamsestraat 69, 3000 Leuven, Belgium
elena.tiukhova@kuleuven.be

Emiliano Penaloza
Department of Statistical and Actuarial Sciences
The University of Western Ontario
1151 Richmond Street,
London, Ontario N6A 5B7, Canada
epenaloz@uwo.ca

Hernan Garcia
Rappi
Cl.93 #19-58, Bogotá, Colombia
javier.garcia@rappi.com

Alejandro Correa Bahnsen
Rappi
Cl.93 #19-58, Bogotá, Colombia
alejandro.correa@rappi.com

María Óskarsdóttir
Department of Computer Science
Reykjavík University
Menntavegi 1, 102 Reykjavík, Iceland
mariaoskars@ru.is

Bart Baesens
Research Center for Information Systems Engineering
Faculty of Economics and Business
KU Leuven
Naamsestraat 69, 3000 Leuven, Belgium
bart.baesens@kuleuven.be

Monique Snoeck
Research Center for Information Systems Engineering
Faculty of Economics and Business
KU Leuven
Naamsestraat 69, 3000 Leuven, Belgium
monique.snoeck@kuleuven.be

Cristián Bravo
Department of Statistical and Actuarial Sciences
The University of Western Ontario
1151 Richmond Street
London, Ontario N6A 5B7, Canada
cbravoro@uwo.ca

Abstract

Leveraging network information for prediction tasks has become a common practice in many domains. Being an important part of targeted marketing, influencer detection can potentially benefit from incorporating dynamic network representation. In this work, we investigate different dynamic Graph Neural Networks (GNNs) configurations for influencer detection and evaluate their prediction performance using a unique corporate data set. We show that using deep multi-head attention in GNN and encoding temporal attributes significantly improves performance. Furthermore, our empirical evaluation illustrates that capturing neighborhood representation is more beneficial that using network centrality measures.

NeurIPS 2022 Temporal Graph Learning Workshop held in New Orleans, United States

1 Introduction

Advances in data collection and processing have enhanced the use of automated data workflows for decision-making. A primary source of data comes in the form of networks that capture connections between people. When relational information is leveraged, it is assumed that people in the network influence each other's behavior and decisions, which has been shown to be true in many domains such as fraud detection [Baesens et al., 2015] or e-commerce recommendations [Sun et al., 2015].

A common way information flows through a network is by the Word-of-Mouth effect which is seen as a powerful tool for spreading influence among customers in marketing [Puigbo et al., 2014]. Customers who succeed in utilizing this effect in order to change others perspectives are considered influencers [Rogers and Cartano, 1962]. It is possible to model such scenarios as a network, in which its topology plays a crucial role in identifying influencers. This is a large area of study with many standard approaches for encoding the network topology, such as neighborhood and centrality metrics as well as collective inference algorithms [Baesens et al., 2015]. Due to the rising popularity of deep learning, graph neural networks (GNNs) are extensively used for end-to-end tasks of graph learning [Rhee et al., 2017, Guo et al., 2019]. However, the research on influencer detection with GNNs is limited, especially when it comes to networks that evolve in time and when there exist several types of connections in a network. To the best of our knowledge, there exists no research about influencer detection on dynamic attributed edge-colored networks with GNNs.

The main purpose of this paper is to add to the body of research on influencer detection by evaluating different dynamic GNNs configurations and to investigate whether encoding network topology using GNNs together with capturing its dynamic evolution have an added value for performance. By doing so, the following contributions are made. Firstly, we adapt dynamic GNNs for ex-post influencer detection, that is, identifying current users of the product or service who influence neighboring non-users to acquire it in the future. Secondly, we evaluate different GNN configurations in combination with different RNN configurations for our problem[1]. Finally, we compare the results to baseline static graph neural networks and dynamic non-GNN approaches.

2 Related work

Puigbo et al. [2014] highlight the importance of influencer detection since the rise of the Internet. However, most of the traditional approaches lack more advanced indicators of the relationships between network actors. Due to the rising popularity of GNNs and the demand for improved relationship extraction techniques, the graph influence network framework has been proposed by Shi et al. [2022]. The framework is aimed at finding the influential neighbors of a node. However, it is not designed for detecting global influencers and can be applied on static networks only.

Networks can be seen as an unstructured data source, requiring specific methods in order to extract network topology and be able to incorporate it into prediction models. Some approaches to learning on networks are based on matrix factorization including spectral clustering [Von Luxburg, 2007] and learning with modularity matrix [Tang and Liu, 2009]. More advanced methods are based on learning by performing random walks on a networks, e.g., DeepWalk [Perozzi et al., 2014] and node2vec [Grover and Leskovec, 2016]. Network topology can be incorporated into the model by extracting centrality information using PageRank-like algorithms which have been proved to be beneficial for, e.g., credit risk prediction in multilayer networks [Óskarsdóttir and Bravo, 2021]. The enhancements in deep learning have brought GNNs to the forefront of the field where they demonstrate cutting-edge performance [Zhou et al., 2020]. The general design pipeline of GNNs includes the steps of specifying the network type and scale, deciding on the task type and building the model using carefully designed computational modules [Zhou et al., 2020].

3 Influencer detection with Discrete Time Dynamic Graphs

Following the design pipeline of Zhou et al. [2020], we define the task of future influencer detection in this paper as a supervised node-level learning problem on a dynamic heterogeneous undirected network. A typical network learning process consists of an encoder and decoder [Hamilton et al.,

[1]The code is available at `https://github.com/Banking-Analytics-Lab/DynamicGraphLearning`

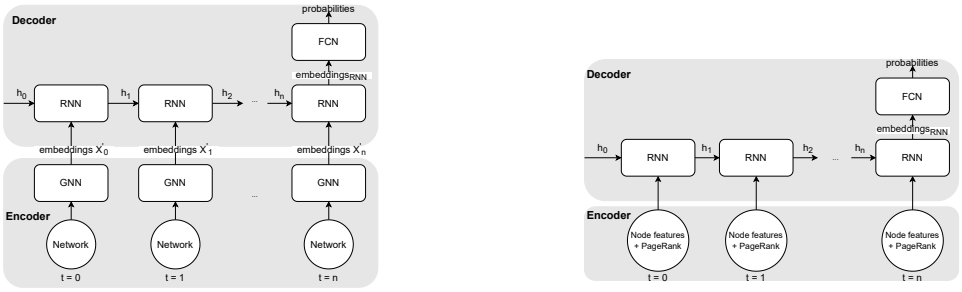
2017b]. The encoder part of the model is aimed at learning node embeddings while the decoder part is used to solve a prediction task, e.g., node classification. Zhu et al. [2022] survey different encoder-decoder architectures that exist for supervised dynamic graph learning and classify them into Discrete Time Dynamic Graph (DTDG) learning that uses network snapshots and Continuous Time Dynamic Graph (CTDG) learning that deals with an updating event stream in a network, e.g., the Temporal Graph Networks framework for deep learning on dynamic network represented as sequences of timed events [Rossi et al., 2020]. Following their taxonomy, we capture the network topology at each timestamp by applying the DTDG encoder for attributed static networks, namely, Graph Convolutional Networks (GCNs) [Kipf and Welling, 2017] and Graph Attention Networks (GATs) [Veličković et al., 2018]. We capture the dynamic nature of the networks by employing the models from the RNN family as a DTDG decoder. We also compare these models with baseline models, namely, the PageRank+RNN, static GNNs and dynamic non-GNN models.

3.1 GNN + RNN

In order to deal with complex data structures and arbitrary size of neighborhood, Kipf and Welling [2017] propose Graph Convolutional Networks that can learn on non-euclidean data such as networks. A GCN model learns node embeddings by aggregating information from a node's neighborhood. However, it assigns the same importance to all the neighboring nodes which is rarely the case in practice. Hence, Veličković et al. [2018] introduce Graph Attention Networks where nodes follow a self-attention mechanism and assign an importance to each connection by attending over their neighbors.

The disadvantage of both GCN and GAT architectures is that they are static and do not take into account the dynamic changes in the network that happen over time. To take into account the time dimension, we use the GNN models mentioned above together with Recurrent Neural Networks, namely the Long Short-Term Memory (LSTM) model [Hochreiter and Schmidhuber, 1997] and Gated Recurrent Units (GRUs) model [Cho et al., 2014]. The LSTM model is capable of learning long-term dependencies by storing long-term memory in a cell state while capturing the most recent information in its hidden states. The GRU model is a less complex version of LSTMs as it does not have a cell state and stores long-term memory directly in its hidden states. In both the LSTM and GRU models, the output embeddings of a GNN model are used as an input.

Considering all the above, we investigate four different model configurations, i.e., GCN+LSTM, GCN+GRU, GAT+LSTM, and GAT+GRU. Their architecture is displayed in Figure 1.



(a) GNN+RNN

(b) Features(+PageRank)+RNN

Figure 1: Models' architecture

3.2 PageRank + RNN & Features + RNN

Following the strategy used in the research by Óskarsdóttir et al. [2017, 2022], we can enrich non-relational classifiers with network features using feature engineering [Verdonck et al., 2021]. One of such network features that can summarize node importance is PageRank [Brin and Page, 1998]. As the PageRank value represents the relative importance of the node within one component, we calculate the value for PageRank separately within each of the connected components. PageRank values are used as an additional node feature and subsequently utilized for dynamic node classification with LSTMs and GRUs (Figure 1b).

4 Experiments

We utilize data from one city of a Super-App company operating in Latin America that has both a delivery app and issues credit cards to its customers. We construct monthly snapshots based on different types of connections between the users (Figure 2), resulting in a dynamic attributed undirected network with implicit time and colored edges. The nodes in the network represent the customers and are attributed with the features that characterize the customer's credit card usage in each monthly snapshot. The edges in the network are colored by different types of connections between network actors as displayed in Appendix A, Table A.1. Coloring the edges is performed by creating edge features (transformed to edge weights in the GCN+RNN models): three binary edge features for credit card, geohash and contacts edge types with the last one being enriched with the references edges created in the network snapshot the month following the month of referral. We note that existing connections never disappear from the network while it is possible that new connections are established (Appendix A, Figure A.2). The network is labelled: customers who referred (i.e., extended an invitation to the Super-App's services to someone they are connected to) other customers in the past at least once are labelled as influencers while the remaining customers are labelled as non-influencers (see Figure 2).

The network is imbalanced, as being an influencer is less common than being a non-influencer (imbalance ratio is $\sim 13\%$). We check if oversampling helps to handle the data imbalance during the model validation step and apply the Synthetic Minority Oversampling TEchnique (SMOTE) [Chawla et al., 2002] to the embeddings generated by the RNN model. We follow the oversampling strategy of Zhao et al. [2021] and oversample the nodes in the embeddings generated by the RNN model or by the GNN encoder in static GNN models. We generate synthetic nodes of the minority class of different quantities that we set as a hyperparameter (Appendix A, Table A.2).

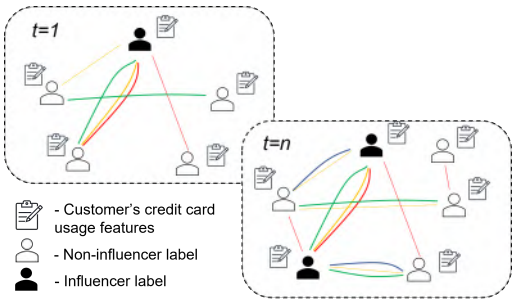


Figure 2: Network

The data splits, a general pipeline of training, validating and testing the models as well as the best hyperparameter specifications found by grid search are displayed in Appendix A, Figure A.1 and Table A.2, respectively. The type of resources used is displayed in Appendix A, Table A.3.

5 Results and Discussion

Table 1: Models' performance (models are implemented using the pytorch geometric library [Fey and Lenssen, 2019]). Confidence intervals of AUC values are obtained from bootstrapping AUC values.

Model	Test AUC seen nodes	Test AUC unseen nodes	Total time (seconds)
GCN+LSTM	0.756±0.006	0.786±0.014	12747.2
GCN+GRU	0.843±0.005	0.730±0.015	12752.5
GAT+LSTM	0.842±0.005	**0.831±0.012**	58919.6
GAT+GRU	**0.864±0.004**	**0.823±0.013**	58801.3
PageRank+LSTM	0.672±0.007	0.685±0.009	2844.3
PageRank+GRU	0.801±0.004	0.665±0.015	4637.8
Features+LSTM	0.673±0.006	0.686±0.009	2275.4
Features+GRU	0.799±0.006	0.673±0.009	4635.3
Static GAT	0.639±0.005	0.700±0.016	40142.5
Static GCN	0.635±0.006	0.663±0.024	1742.3

As can be seen from Table 1, GNN+RNN models in general outperform baseline models with the GAT+GRU configuration being the best one on both seen and unseen nodes and the GAT+LSTM being statistically identical to GAT+GRU over unseen nodes. A notable improvement over baseline

models is obtained on unseen nodes with an AUC increase of 0.13 obtained on the best GAT+LSTM model. We also note that the models with GAT as an encoder outperform the models with a network topology encoded by GCN.

The best GCN+RNN configurations consist of 200 embeddings generated by GCN with one hidden layer and 200 hidden dimensions in both GCN and RNN (LSTM or GRU). In contrast, the best GAT+RNN configurations are deep GATs with 4 layers and 4 heads generating 200 embeddings and 100 hidden layers in GAT and RNN (Appendix A, Table A.2). Also, we note that upsampling does not increase the performance meaning that most of the models can deal with the data imbalance. Therefore, using deep multi-head attention mechanism helps to better capture network topology than just aggregating information from a node's neighborhood over both balanced and unbalanced sets.

Among the baseline models, non-GNN dynamic models consistently outperform non-dynamic GNNs on seen nodes while the static GAT model being the best over unseen nodes. Hence, capturing time-evolving patterns plays an important role in predicting future influencers among seen nodes while the network typology encoding is crucial for generalizing to unseen nodes. Moreover, adding PageRank as an additional feature does not result in a significant performance improvement. Thus, neighbor feature representations captured by GNNs are more important for influencer detection than using centrality measures such as PageRank.

6 Conclusion

Early detection and targeting of influencers allows for efficient spread of information through the network. Hence, different model architectures for influencer detection with networks should be evaluated. For these reasons, we researched different dynamic GNNs configurations and investigated whether encoding network topology with GNNs and capturing the dynamic evolution of the network have an added value to the prediction performance.

First, neighbor feature representations captured by GNNs are more important than centrality measures such as PageRank especially when it comes to generalizing to unseen nodes where using multi-head attention in the encoder boosts the performance. Second, we conclude that dynamics of the network plays an important role; thus, the decoder of the model should capture time. As the use of the influencer detection model is intended for marketing, we foresee the best models will allow optimizing the frequency and tenor of targeted marketing actions some users can be subjected to.

Our work has a few limitations. First, there could exist other connections in the network that are not captured by a current network setup. Moreover, the way edges are created based on the Geohash 7 (Appendix A, Table A.1) proximity can affect the connectivity strength of the network. Future improvements include unsupervised learning algorithms for influencer detection including anomaly detection methods for dynamic networks. Next, we can evaluate models from a profit-driven perspective that will enable us to bring more business context into the results. Furthermore, we can explore behavioral node features from the delivery app which can potentially increase predictive power. Moreover, we can expand the network to more than one city to study how generalizable the results are in a larger geographical area. The future research avenues also include exploring more encoder configurations such as GraphSAGE [Hamilton et al., 2017a] or Graph Isomorphism Networks [Xu et al., 2018].

7 Acknowledgements

The research was sponsored by the ING Chair on Applying Deep Learning on Metadata as a Competitive Accelerator. The fifth author acknowledges the support of the Icelandic Research Fund (IRF) [grant number 228511-051]. The last author acknowledges the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) [Discovery Grant RGPIN-2020-07114]. This research was undertaken, in part, thanks to funding from the Canada Research Chairs program. This research was enabled in part by support provided by Compute Ontario (`computeontario.ca`) and the Digital Research Alliance of Canada (`alliancecan.ca`) [FT #2070].

References

B. Baesens, V. Van Vlasselaer, and W. Verbeke. *Fraud analytics using descriptive, predictive, and social network techniques: a guide to data science for fraud detection*. John Wiley & Sons, 2015.

S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998.

N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

M. Fey and J. E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 922–929, 2019.

W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017a.

W. L. Hamilton, R. Ying, and J. Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017b.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

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

T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=SJU4ayYgl`.

M. Óskarsdóttir and C. Bravo. Multilayer network analysis for improved credit risk prediction. *Omega*, 105:102520, 2021.

M. Óskarsdóttir, C. Bravo, W. Verbeke, C. Sarraute, B. Baesens, and J. Vanthienen. Social network analytics for churn prediction in telco: Model building, evaluation and network architecture. *Expert Systems with Applications*, 85:204–220, 2017.

M. Óskarsdóttir, W. Ahmed, K. Antonio, B. Baesens, R. Dendievel, T. Donas, and T. Reynkens. Social network analytics for supervised fraud detection in insurance. *Risk Analysis*, 42(8):1872–1890, 2022.

B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

J.-Y. Puigbo, G. Sánchez-Hernández, M. Casabayó, and N. Agell. Influencer detection approaches in social networks: A current state-of-the-art. In *CCIA*, pages 261–264, 2014.

S. Rhee, S. Seo, and S. Kim. Hybrid approach of relation network and localized graph convolutional filtering for breast cancer subtype classification. *arXiv preprint arXiv:1711.05859*, 2017.

E. M. Rogers and D. G. Cartano. Methods of measuring opinion leadership. *Public opinion quarterly*, pages 435–441, 1962.

E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, 2020.

Y. Shi, P. Quan, Y. Xiao, M. Lei, and L. Niu. Graph influence network. *IEEE Transactions on Cybernetics*, 2022.

Z. Sun, L. Han, W. Huang, X. Wang, X. Zeng, M. Wang, and H. Yan. Recommender systems based on social networks. *Journal of Systems and Software*, 99:109–119, 2015. ISSN 0164-1212. doi: https://doi.org/10.1016/j.jss.2014.09.019. URL `https://www.sciencedirect.com/science/article/pii/S0164121214002064`.

L. Tang and H. Liu. Relational learning via latent social dimensions. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 817–826, 2009.

P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rJXMpikCZ`.

T. Verdonck, B. Baesens, M. Óskarsdóttir, et al. Special issue on feature engineering editorial. *Machine Learning*, pages 1–12, 2021.

U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

T. Zhao, X. Zhang, and S. Wang. Graphsmote: Imbalanced node classification on graphs with graph neural networks. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 833–841, 2021.

J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

Y. Zhu, F. Lyu, C. Hu, X. Chen, and X. Liu. Encoder-decoder architecture for supervised dynamic graph learning: A survey, 2022. URL `https://arxiv.org/abs/2203.10480`.

A Appendix

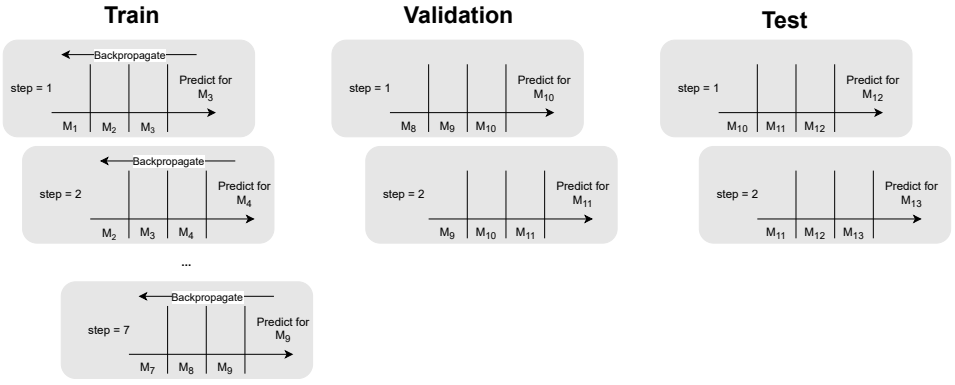


Figure A.1: Train-validation-test split and model training. Windows are obtained by 1-month shift. Backpropagation happens at the end of the time window, and the prediction is made for the last month of the window.

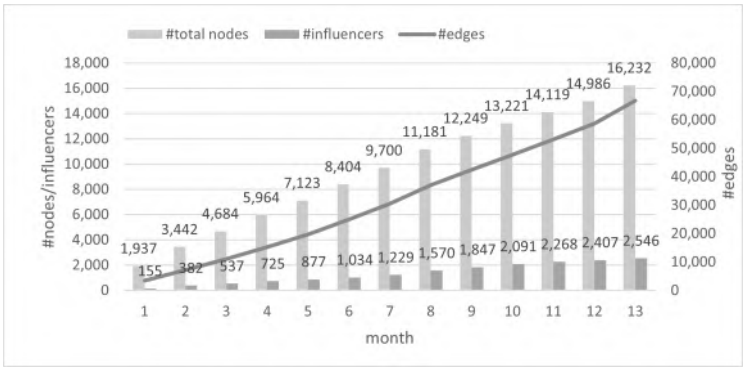


Figure A.2: Network growth characteristics

Table A.1: Edge types

Type	Description	Data source
Credit card	There exist an edge between users who used the same credit card in the app.	Delivery app
References	There exist an edge between users who referenced/got referenced. This type of edge is created in the network snapshot of the month following the month of referral.	Credit card usage
Geohash	There exist an edge between users who ordered more than 4 times in the close geographical proximity (Geohash 7 - 152.9m x 152.4m).	Delivery app
Contacts	There exist an edge between users if at least one of them have the other one in the phone contacts.	Delivery app

Table A.2: Best hyperparameter settings found by grid search: the best model has a maximal average AUC on seen and unseen nodes. Static hyperparameters: epochs = 500 (early stop = 50), learning rate = 0.0001, optimizer = ADAM [Kingma and Ba, 2014]

Model	Val. AUC seen nodes	Val. AUC unseen nodes	#hidden dimensions GNN/RNN	#emb. GNN	#layers GNN	SMOTE sample rate	#heads GAT	dropout rate GNN
GCN+LSTM	0.774±0.004	0.825±0.015	200	200	1	0	\	0.5
GCN+GRU	0.862±0.004	0.769±0.016	200	200	1	0.75	\	0.5
GAT+LSTM	0.860±0.003	0.871±0.013	100	200	4	0	4	0.5
GAT+GRU	0.880±0.003	0.860±0.013	100	200	4	0	4	0.5
PageRank+LSTM	0.676±0.002	0.725±0.017	100	\	\	0	\	\
PageRank+GRU	0.818±0.003	0.726±0.012	200	\	\	0	\	\
Features+LSTM	0.678±0.003	0.725±0.018	100	\	\	0	\	\
Features+GRU	0.815±0.004	0.721±0.018	200	\	\	0	\	\
Static GAT	0.650±0.004	0.720±0.011	\	200	4	0	4	0.5
Static GCN	0.633±0.005	0.665±0.016	\	200	1	0	\	0.5

Table A.3: Resource types

Resource	Specification
Memory per CPU	8G
CPU cores per task	2
Processor	Intel E5-2683 v4 Broadwell @ 2.1GHz