Inference of Sequential Patterns for Neural Message Passing in Temporal Graphs

Jan von Pichowski Vincenzo Perri Lisi Qarkaxhija Ingo Scholtes
Chair of Machine Learning for Complex Networks
Center for Artificial Intelligence and Data Science (CAIDAS)
Julius-Maximilians-Universität Würzburg, DE
{jan.pichowski,vincenzo.perri,lisi.qarkaxhija,ingo.scholtes}@uni-wuerzburg.de

Abstract

The modelling of temporal patterns in dynamic graphs is an important current research issue in the development of time-aware Graph Neural Networks (GNNs). However, whether or not a specific sequence of events in a temporal graph constitutes a *temporal* pattern not only depends on the frequency of its occurrence. We must also consider whether it deviates from what is expected in a temporal graph where timestamps are randomly shuffled. While accounting for such a random baseline is important to model temporal patterns, it has mostly been ignored by current temporal graph neural networks. To address this issue we propose HYPA-DBGNN, a novel two-step approach that combines (i) the inference of anomalous sequential patterns in time series data on graphs based on a statistically principled null model, with (ii) a neural message passing approach that utilizes a higher-order De Bruijn graph whose edges capture overrepresented sequential patterns. Our method leverages hypergeometric graph ensembles to identify anomalous edges within both first- and higher-order De Bruijn graphs, which encode the temporal ordering of events. Consequently, the model introduces an inductive bias that enhances model interpretability and leads to improved performance.

1 Introduction

Graphs are powerful representations of complex data used to describe static and dynamic systems. Not surprisingly, there is a growing collection of successful methods for learning on static [1–3] and dynamic graphs [4, 5]. A common theme between static and temporal GNNs is that the observed graphs are usually directly used for message passing. Recently data augmentation techniques have been proposed to improve the generalizability of GNNs. Such data augmentation techniques have been considered for a variety of reasons such as to reduce oversquashing [6], improve class homophily for node classification [7], foster diffusion [8], or include non-dyadic relation-ships [4]. Another motivation that has recently been highlighted in [9] is the presence of noise in empirically observed graphs. This motivates augmentation techniques for GNNs that ideally prune spuriously observed edges, while adding erroneously unobserved edges.

However, addressing noise in observed graphs arguably requires *graph correction* methods accounting for a "random baseline" that allows to distinguish significant patterns from noise, rather than *augmentation* methods that are based on heuristics or adjust the graph based on ground truth node classes. Moreover, the application of GNNs to temporal graphs introduces unique challenges for data augmentation as we typically want to focus on temporal patterns that are due to the time-ordered sequence of events. To the best of our knowledge, no existing works have considered graph correction methods that combine a statistically principled inference of sequential patterns with temporal GNNs.

The contributions of our work are as follows:

Pichowski et al., Inference of Sequential Patterns for Neural Message Passing in Temporal Graphs (Extended Abstract). Presented at the Third Learning on Graphs Conference (LoG 2024), Virtual Event, November 26–29, 2024.

(i) We propose a novel approach to augment message passing based on a statistical null model. This allows us to infer which temporal sequences in a time-stamped interaction sequence are over- or under-represented compared to a random baseline graph in which edge frequencies are preserved while their temporal ordering is shuffled. This approach leads to the time-aware temporal graph neural network architecture HYPA-DBGNN with statistical principled corrections.

(ii) We demonstrate the correction ability in synthetic temporal graphs. We further show the practical relevance of our method by evaluating node classification in five empirical temporal graphs capturing time-stamped proximity events between humans. A comparison of HYPA-DBGNN with standard De Bruijn Graph Neural Networks without our HYPA-based inference reveals that our approach yields an improved accuracy in all five data sets. Moreover, a comparison to seven baseline techniques shows that our method yields the best performance in all empirical data.

Different from prior works, with our work we propose a *statistically principled* data augmentation for temporal graph neural networks that uses a *statistical ensemble* of temporal graphs with a given weighted topology. Apart from improving temporal GNNs, we further argue that the general approach of utilizing well-known *statistical ensembles of graphs* from network science for *graph correction* could help to improve the performance of GNNs in data affected by noise.

2 Related Work

Data augmentation for graphs has been explored from various directions with the goal of allowing machine learning models to better generalize and attend to signal over noise [10]. Many methods have utilized heuristic graph modification strategies like randomly removing nodes [11, 12], edges [13], or subgraphs [11, 14] to improve performance and generalizability. Other works have considered adding virtual nodes [15, 16] or rewiring the network topology, which also addresses oversquashing [6, 17], with graph transformers operating on a fully connected topology representing an extreme case [18–20]. Another area of research focused on learning the graph augmentations from the data either in a preprocessing step [21, 22] or by embedding the augmentation into an end-to-end differentiable pipeline [23–27]. Network data augmentation has also been explored by going beyond pairwise connections, either through mediating node interactions via subgraphs [28–31] or by utilizing higher-order graphs. Examples of higher-order approaches include simplicial networks [32], cellular complexes [33, 34], hypergraphs [35–37], and time-respecting node sequences [4].

Our work addresses temporal graph data. Temporal GNNs have been developed for both discrete and continuous time settings [38]. Discrete-time approaches segment the temporal data into time windows [39, 40] and thus lose information on time-respecting paths within those time windows. In contrast, continuous-time approaches produce time-evolving node embeddings, focusing on the temporal variability of network activity at different time points rather than on the patterns occurring across temporally-ordered interaction sequences [5, 41]. [42, 43] explore temporal augmentation.

3 Background

We consider time respecting paths on graphs as ordered sequences $(v_1, v_2, \ldots, v_l)$ of nodes. The transitions correspond to edges that are not transitive, i.e. the order needs to be respected. To respect this property, we use higher-order De Bruijn graphs model that encode the probabilities of whole path sequences explicitly. In this graph model, k-th-order nodes encode overlapping subpaths. The edges contain the frequency of paths between those nodes: $(\langle v_0 \ldots v_{k-1}\rangle, \langle v_1 \ldots v_k\rangle)$. We use the transitivity assumption as a null model to define anomalies with respect to this reference base. Anomalies occur in sequences that deviate from this baseline, likely due to correlations and interdependencies not captured by the transitivity assumption.

We rely on the hypergeometric ensemble to test for anomalous edge frequencies based on node activity, i.e., their in- and out-degrees. This methodology is extended to test if the frequencies of paths of length k are anomalous given those of paths of length $k-1$. More specific, the configuration models [44] provide randomization methods for graphs that shuffle edges while preserving vertex degrees. Casiraghi and Nanumyan [45] contributed a closed-form expression for the *soft* configuration model, which fixes the *expected* vertex degrees rather than the exact degree sequence. They provide a probability mass function $\Pr(X_{uv} = A_{ij})$ that describes the probability of observing A_{ij} edges between nodes i and j in any random realization X. The statistical *HYPA* framework, introduced

by LaRock et al. [46], quantifies the anomalousness of the edge frequencies in higher-order De Bruijn graphs by evaluating the HYPA score $HYPA^{(k)}(u,v) = \Pr(X_{uv} \leq A_{uv})$.

4 HYPA De Bruijn Graph Neural Network Architecture

We now introduce the HYPA-DBGNN architecture that relies on statistical principled graph augmentation. As outlined before, the used k-th-order De Bruijn graphs capture the observed temporal dynamics as frequencies of the k-th-order sequences through the edges between k-th-order nodes. This potentially biased representation yields the foundation for hypergeometric ensembles whose edge frequencies are induced by the k-1-th-order sub-sequences. The used HYPA score encodes a highly represented edge whereas a HYPA score approaching zero describes edges that are observed less than expected with respect to any random realization. Leveraging the HYPA scores as adjacency matrix $A^{(k)}_{uv} = HYPA^{(k)}(u,v)$ leads to corrected graphs with reduced under-represented edges and balanced overrepresented edges encoding the temporal pattern. The inferred graph corrections directly include anomaly statistics in the graph topology. This leads to a multi-order augmented message passing scheme that relies on the inferred graphs with induced bias. For layer l, we define the update rule of the message passing as

$$\vec{h}_v^{k,l} = \sigma\left(\mathbf{W}^{k,l} \sum_{\{u \in V^{(k)}:(u,v)\in E^{(k)}\}\cup\{v\}} \frac{HYPA^{(k)}(u,v)\cdot \vec{h}_u^{k,l-1}}{\sqrt{H(v)\cdot H(u)}}\right), \tag{1}$$

with the previous hidden node representation $\vec{h}_u^{k,l-1}$, the inferred HYPA score $HYPA^{(k)}(u,v)$ (capturing the induced bias), the trainable weight matrices $\mathbf{W}^{k,l} \in \mathbb{R}^{H^l \times H^{l-1}}$, the normalization factor based on the HYPA score sum of incoming edges $H(v)$, and the non-linear activation function σ. An overview of the inference process and the proposed neural network architecture for the first- and second-order graph is shown in Figure 1. We discuss the computational complexity in appendix D.

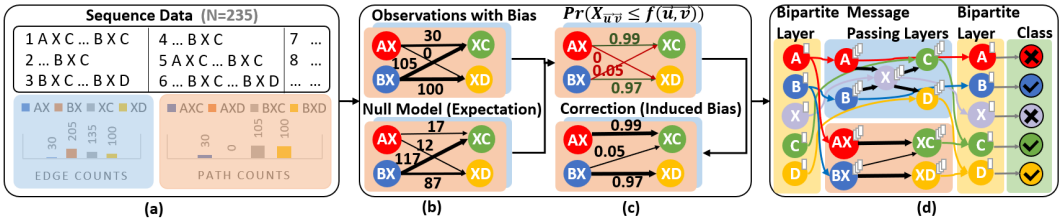

Figure 1: Inference procedure leading to the dynamic graph used for neural message passing. (a) Example of sequence data adapted from LaRock et al. [46]. (b) Multi-order De Bruijn Graphs are obtained form the sequences. First-order is blue (background) and higher-order is orange. They are compared to a random graph ensemble null model with shuffled time-stamped k-1-order edges. (c) The graphs are corrected by introducing a statistical principled bias that revalues all edges ($w_{\langle AXC\rangle} \approx w_{\langle BXD\rangle} > w_{\langle BXC\rangle}$) and removes under-represented edges, i.e. edges that appear with a high probability less than expected ($\langle AXD\rangle$). (d) The multi-order graph neural network is trained respecting the inferred graphs. The node features are transferred between the higher- and first-order nodes with bipartite layers.

5 Experimental Evaluation

We compare our architecture with graph representation learning methods (**EVO** [47], **HONEM** [48], **DeepWalk** [49] and **Node2Vec** [50]) and deep graph learning methods (**GCN** [1], **LGNN** [51], **TGN** [5] and **DBGNN** [4]). The models are evaluated with cross validation with parameter search.We describe the evaluation in detail in Appendix A and the model architecture in detail in Appendix B.

We use synthetic data with two classes of nodes to demonstrate the type of patterns that only our model can learn. Importantly, it contains a heterogeneous sequence (e.g. $\langle v_0, v_1, v_2\rangle_f$) distribution of time-stamped edges or events (here: $(v_0,v_1)_{t_0}, (v_1,v_2)_{t_1}, t_0 < t_1$) between nodes. The learnable sequential pattern is an increased class-assortativity, i.e. edges are temporally ordered such that same class events are preferred followed by each other (e.g. leading to $\langle A, A, A, B\rangle$). Hence, these higher-order sequences with nodes from the same group are over-expressed compared to what we

would expect by shuffling the temporal-order of the timestamped-edges between the nodes (e.g. $\langle A, B, A, A \rangle$). Our experiments on empirical data leverage the five empirical time series datasets on dynamic graphs from [4]. The data sets are *Highschool2011* and *Highschool2012* [52], *Hospital* [53], *StudentSMS* [54], and *Workplace2016* [55]. Table 1 and Table 2 show that HYPA-DBGNN has not only superior performance but is also the only model that is able to learn the synthetic pattern.

Table 1: Comparison of HYPA-DBGNN baselines for the synthetic data sets. The table presents the balanced accuracy and its standard deviation for the static node classification task on dynamic graphs as obtained through the outlined experiments. The *Unweighted Sampling* data set contains a heterogeneous sequence distribution of time-stamped edges with shuffled temporal order. The adapted distribution of sequences in *Weighted Sampling* encodes a sequential pattern such that time-stamped edges between nodes of the same class are overrepresented but not necessarily very frequent.

Representation Learning	EVO	HONEM	DeepWalk	Node2Vec	
Unweighted Sampling	40.00 ± 31.62	80.00 ± 25.82	60.00 ± 21.08	60.00 ± 21.08	
Weighted Sampling	40.00 ± 31.62	80.00 ± 25.82	60.00 ± 21.08	60.00 ± 21.08	
Deep Graph Learning	GCN	LGNN	DBGNN	TGN	HYPA-DBGNN
Unweighted Sampling	50.00 ± 33.33	50.00 ± 0.00	45.00 ± 28.38	50.00 ± 0.00	45.00 ± 15.81
Weighted Sampling	45.00 ± 28.38	50.00 ± 0.00	45.00 ± 15.81	50.00 ± 0.00	100.00 ± 0.00

Table 2: Comparison of HYPA-DBGNN with node representation learning and deep graph learning baselines for dynamic graphs. The table presents the balanced accuracy and its standard deviation for the models on empirical static node classification tasks for dynamic graphs. We mark the best results.

Model	Highschool2011	Highschool2012	Hospital	StudentSMS	Workplace2016
EVO	43.68 ± 10.91	50.05 ± 7.30	25.83 ± 8.29	55.05 ± 6.39	26.50 ± 12.08
HONEM	59.00 ± 10.61	50.49 ± 9.31	39.44 ± 17.57	53.81 ± 7.28	83.17 ± 11.14
DeepWalk	54.64 ± 17.70	49.65 ± 12.97	24.58 ± 10.92	52.78 ± 7.83	20.54 ± 9.51
Node2Vec	54.64 ± 17.70	49.65 ± 12.97	24.58 ± 10.92	52.31 ± 7.70	20.54 ± 9.51
GCN	55.00 ± 13.37	59.35 ± 11.13	43.47 ± 9.03	54.50 ± 6.40	73.33 ± 12.60
LGNN	57.72 ± 9.85	51.43 ± 17.94	44.03 ± 9.03	52.71 ± 6.63	84.83 ± 14.77
DBGNN	61.54 ± 11.13	64.93 ± 15.26	52.50 ± 19.27	57.72 ± 5.29	84.42 ± 15.59
TGN	61.52 ± 11.25	41.52 ± 6.19	50.27 ± 14.83	50.67 ± 4.10	80.16 ± 18.71
HYPA-DBGNN	**63.25 ± 16.18**	**66.41 ± 10.24**	**76.39 ± 17.12**	**60.66 ± 6.11**	**88.29 ± 10.51**

6 Conclusion

In this work, we propose HYPA-DBGNN, a novel deep graph learning architecture that accounts for time-respecting paths in temporal graph data with high temporal resolution. Different from existing graph learning methods that employ neural message passing along time-respecting paths, we introduce a two-step approach which first infers anomalous sequential patterns based on an analytically tractable null model for time-respecting paths that preserves both the topology and the frequency, but not the temporal ordering, of time-stamped edges. In a second step, we apply neural message passing on an augmented higher-order De Bruijn graph, whose edges capture time-respecting paths that are overrepresented compared to the expectation from that random baseline. An experimental evaluation of our approach in a synthetic model and five empirical data sets on temporal graphs reveals that our proposed method considerably improves node classification compared to seven baseline methods in all studied data sets, with performance gains ranging from 2.27 % to 45.5 %.

Despite these contributions, our work raises a number of open questions that we did not address within the scope of this work. Future studies building on our work could thus additionally consider richer node and edge information, which is likely to further improve the performance of our model. Moreover, the framework of hypergeometric statistical ensembles allows to include non-homogeneous "edge propensities" based, e.g., on a homophily of nodes with similar attributes. This could possibly be used to generate domain-specific null models leading to a graph learning architecture that includes a non-trivial inductive bias, which we did not explore in this work. Bridging the gap between the application of statistical graph ensembles in network science and deep graph learning, we finally argue that our work opens broader perspectives for the integration of statistical graph inference, graph

augmentation, and neural message passing. In particular, applying our method to the inference of (first-order) edges in static graphs could be a promising approach to address the issue that empirical graphs are rarely unspoiled reflections of reality, but are often subject to measurement errors and noise. The need to combine graph inference techniques with neural message passing [56–58] has recently been identified as a major challenge for deep graph learning, and our work can be seen as a step in this direction.

References

[1] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017. 1, 3, 9, 13

[2] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

[3] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2018. 1

[4] Lisi Qarkaxhija, Vincenzo Perri, and Ingo Scholtes. De bruijn goes neural: Causality-aware graph neural networks for time series data on dynamic graphs, 2022. 1, 2, 3, 4, 9, 10, 11

[5] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, 2020. 1, 2, 3, 9, 14

[6] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv preprint arXiv:2111.14522*, 2021. 1, 2

[7] Yongxu Liu, Zhi Zhang, Yan Liu, and Yao Zhu. Gatsmote: Improving imbalanced node classification on graphs via attention and homophily. *Mathematics*, 10(11), 2022. ISSN 2227-7390. doi: 10.3390/math10111799. URL https://www.mdpi.com/2227-7390/10/11/1799. 1

[8] Jialin Zhao, Yuxiao Dong, Ming Ding, Evgeny Kharlamov, and Jie Tang. Adaptive diffusion in graph neural networks. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=0Kb33DHJ1g. 1

[9] Tong Zhao, Yozen Liu, Leonardo Neves, Oliver Woodford, Meng Jiang, and Neil Shah. Data augmentation for graph neural networks. In *Proceedings of the aaai conference on artificial intelligence*, volume 35, pages 11015–11023, 2021. 1

[10] Tong Zhao, Wei Jin, Yozen Liu, Yingheng Wang, Gang Liu, Stephan Günnemann, Neil Shah, and Meng Jiang. Graph data augmentation for graph machine learning: A survey. *arXiv preprint arXiv:2202.08871*, 2022. 2

[11] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in neural information processing systems*, 33:5812–5823, 2020. 2

[12] Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. Graph random neural networks for semi-supervised learning on graphs. *Advances in neural information processing systems*, 33:22092–22103, 2020. 2

[13] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903*, 2019. 2

[14] Yiwei Wang, Wei Wang, Yuxuan Liang, Yujun Cai, and Bryan Hooi. Graphcrop: Subgraph cropping for graph classification. *arXiv preprint arXiv:2009.10564*, 2020. 2

[15] Trang Pham, Truyen Tran, Hoa Dam, and Svetha Venkatesh. Graph classification via deep learning with virtual nodes. *arXiv preprint arXiv:1708.04357*, 2017. 2

[16] EunJeong Hwang, Veronika Thost, Shib Sankar Dasgupta, and Tengfei Ma. Revisiting virtual nodes in graph neural networks for link prediction. 2021. 2

[17] Federico Barbero, Ameya Velingker, Amin Saberi, Michael Bronstein, and Francesco Di Giovanni. Locality-aware graph-rewiring in gnns. *arXiv preprint arXiv:2310.01668*, 2023. 2

[18] Grégoire Mialon, Dexiong Chen, Margot Selosse, and Julien Mairal. Graphit: Encoding graph structure in transformers. *arXiv preprint arXiv:2106.05667*, 2021. 2

[19] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in neural information processing systems*, 34:28877–28888, 2021.

[20] Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, 34:21618–21629, 2021. 2

[21] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 66–74, 2020. 2

[22] Tong Zhao, Yozen Liu, Leonardo Neves, Oliver Woodford, Meng Jiang, and Neil Shah. Data augmentation for graph neural networks. In *Proceedings of the aaai conference on artificial intelligence*, volume 35, pages 11015–11023, 2021. 2

[23] Bo Jiang, Ziyan Zhang, Doudou Lin, Jin Tang, and Bin Luo. Semi-supervised learning with graph learning-convolutional networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11313–11320, 2019. 2

[24] Jianglin Lu, Yi Xu, Huan Wang, Yue Bai, and Yun Fu. Latent graph inference with limited supervision. *Advances in Neural Information Processing Systems*, 36, 2024.

[25] Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. Learning discrete structures for graph neural networks. In *International conference on machine learning*, pages 1972–1982. PMLR, 2019.

[26] Bahare Fatemi, Layla El Asri, and Seyed Mehran Kazemi. Slaps: Self-supervision improves structure learning for graph neural networks. *Advances in Neural Information Processing Systems*, 34:22667–22681, 2021.

[27] Anees Kazi, Luca Cosmo, Seyed-Ahmad Ahmadi, Nassir Navab, and Michael M Bronstein. Differentiable graph module (dgm) for graph convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):1606–1617, 2022. 2

[28] Federico Monti, Karl Otness, and Michael M Bronstein. Motifnet: a motif-based graph convolutional network for directed graphs. In *2018 IEEE data science workshop (DSW)*, pages 225–228. IEEE, 2018. 2

[29] Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael M Bronstein, and Haggai Maron. Equivariant subgraph aggregation networks. *arXiv preprint arXiv:2110.02910*, 2021.

[30] Lingxiao Zhao, Wei Jin, Leman Akoglu, and Neil Shah. From stars to subgraphs: Uplifting any gnn with local structure awareness. *arXiv preprint arXiv:2110.03753*, 2021.

[31] Leonardo Cotta, Christopher Morris, and Bruno Ribeiro. Reconstruction for powerful graph representations. *Advances in Neural Information Processing Systems*, 34:1713–1726, 2021. 2

[32] Cristian Bodnar, Fabrizio Frasca, Yuguang Wang, Nina Otter, Guido F Montufar, Pietro Lio, and Michael Bronstein. Weisfeiler and lehman go topological: Message passing simplicial networks. In *International Conference on Machine Learning*, pages 1026–1037. PMLR, 2021. 2

[33] Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yuguang Wang, Pietro Lio, Guido F Montufar, and Michael Bronstein. Weisfeiler and lehman go cellular: Cw networks. *Advances in neural information processing systems*, 34:2625–2640, 2021. 2

[34] Mustafa Hajij, Ghada Zamzmi, Theodore Papamarkou, Nina Miolane, Aldo Guzmán-Sáenz, Karthikeyan Natesan Ramamurthy, Tolga Birdal, Tamal K Dey, Soham Mukherjee, Shreyas N Samaga, et al. Topological deep learning: Going beyond graph data. *arXiv preprint arXiv:2206.00606*, 2022. 2

[35] Jing Huang and Jie Yang. Unignn: a unified framework for graph and hypergraph neural networks. *arXiv preprint arXiv:2105.00956*, 2021. 2

[36] Eli Chien, Chao Pan, Jianhao Peng, and Olgica Milenkovic. You are allset: A multiset function framework for hypergraph neural networks. *arXiv preprint arXiv:2106.13264*, 2021.

[37] Dobrik Georgiev, Marc Brockschmidt, and Miltiadis Allamanis. Heat: Hyperedge attention networks. *arXiv preprint arXiv:2201.12113*, 2022. 2

[38] Antonio Longa, Veronica Lachi, Gabriele Santin, Monica Bianchini, Bruno Lepri, Pietro Lio, Franco Scarselli, and Andrea Passerini. Graph neural networks for temporal graphs: State of the art, open challenges, and opportunities. *arXiv preprint arXiv:2302.01018*, 2023. 2

[39] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In *Proceedings of the 13th international conference on web search and data mining*, pages 519–527, 2020. 2

[40] Ehsan Hajiramezanali, Arman Hasanzadeh, Krishna Narayanan, Nick Duffield, Mingyuan Zhou, and Xiaoning Qian. Variational graph recurrent neural networks. *Advances in neural information processing systems*, 32, 2019. 2

[41] Da Xu, Chuanwei Ruan, Evren Körpeoglu, Sushant Kumar, and Kannan Achan. Inductive representation learning on temporal graphs. *CoRR*, abs/2002.07962, 2020. URL https://arxiv.org/abs/2002.07962. 2

[42] Haoran Liu, Jianling Wang, Kaize Ding, and James Caverlee. Topological and temporal data augmentation for temporal graph networks. In *Temporal Graph Learning Workshop @ NeurIPS 2023*, 2023. URL https://openreview.net/forum?id=HSgx1aJeR8. 2

[43] Yiwei Wang, Yujun Cai, Yuxuan Liang, Henghui Ding, Changhu Wang, Siddharth Bhatia, and Bryan Hooi. Adaptive data augmentation on temporal graphs. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 1440–1452. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/0b0b0994d12ad343511adfbfc364256e-Paper.pdf. 2

[44] Michael Molloy and Bruce Reed. A critical point for random graphs with a given degree sequence. *Random Struct. Alg.*, 6(2-3):161–180, March 1995. ISSN 1098-2418. doi: 10.1002/rsa.3240060204. 2

[45] Giona Casiraghi and Vahan Nanumyan. Configuration models as an urn problem. *Scientific Reports*, 11(1), June 2021. ISSN 2045-2322. doi: 10.1038/s41598-021-92519-y. URL http://dx.doi.org/10.1038/s41598-021-92519-y. 2

[46] Timothy LaRock, Vahan Nanumyan, Ingo Scholtes, Giona Casiraghi, Tina Eliassi-Rad, and Frank Schweitzer. *HYPA: Efficient Detection of Path Anomalies in Time Series Data on Networks*, pages 460–468. Society for Industrial and Applied Mathematics, January 2020. doi: 10.1137/1.9781611976236.52. URL http://dx.doi.org/10.1137/1.9781611976236.52. 3, 10, 11

[47] Caleb Belth, Fahad Kamran, Donna Tjandra, and Danai Koutra. When to remember where you came from: node representation learning in higher-order networks. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '19, page 222–225, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450368681. doi: 10.1145/3341161.3342911. URL https://doi.org/10.1145/3341161.3342911. 3, 9

[48] Mandana Saebi, Giovanni Luca Ciampaglia, Lance M. Kaplan, and Nitesh V. Chawla. Honem: Learning embedding for higher order networks. *Big Data*, 8(4):255–269, August 2020. ISSN 2167-647X. doi: 10.1089/big.2019.0169. URL http://dx.doi.org/10.1089/big.2019.0169. 3, 9

[49] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '14. ACM, August 2014. doi: 10.1145/2623330.2623732. URL http://dx.doi.org/10.1145/2623330.2623732. 3, 9

[50] Aditya Grover and Jure 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. 3, 9

[51] Zhengdao Chen, Xiang Li, and Joan Bruna. Supervised community detection with line graph neural networks, 2020. 3, 9

[52] Julie Fournet and Alain Barrat. Contact patterns among high school students. *PLoS ONE*, 9 (9):e107878, September 2014. ISSN 1932-6203. doi: 10.1371/journal.pone.0107878. URL http://dx.doi.org/10.1371/journal.pone.0107878. 4, 11

[53] Philippe Vanhems, Alain Barrat, Ciro Cattuto, Jean-François Pinton, Nagham Khanafer, Corinne Régis, Byeul-a Kim, Brigitte Comte, and Nicolas Voirin. Estimating potential infection transmission routes in hospital wards using wearable proximity sensors. *PLoS ONE*, 8(9): e73970, September 2013. ISSN 1932-6203. doi: 10.1371/journal.pone.0073970. URL http://dx.doi.org/10.1371/journal.pone.0073970. 4, 11

[54] Piotr Sapiezynski, Arkadiusz Stopczynski, David Dreyer Lassen, and Sune Lehmann. Interaction data from the copenhagen networks study. *Scientific data*, 6, December 2019. ISSN 2052-4463. doi: 10.1038/s41597-019-0325-x. 4, 11

[55] Mathieu Génois, Christian L. Vestergraad, Julie Fournet, André Panisson, Isabelle Bonmarin, and Alain Barrat. Data on face-to-face contacts in an office building suggest a low-cost vaccination strategy based on community linkers. *Network Science*, 3(3):326–347, March 2015. ISSN 2050-1250. doi: 10.1017/nws.2015.10. URL http://dx.doi.org/10.1017/nws.2015.10. 4, 11

[56] Jiaqi Ma, Weijing Tang, Ji Zhu, and Qiaozhu Mei. A flexible generative framework for graph-based semi-supervised learning. *Advances in Neural Information Processing Systems*, 32, 2019. 5

[57] Soumyasundar Pal, Saber Malekmohammadi, Florence Regol, Yingxue Zhang, Yishi Xu, and Mark Coates. Non parametric graph learning for bayesian graph neural networks. In *Conference on uncertainty in artificial intelligence*, pages 1318–1327. PMLR, 2020.

[58] Yingxue Zhang, Soumyasundar Pal, Mark Coates, and Deniz Ustebay. Bayesian graph convolutional neural networks for semi-supervised classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 5829–5836, 2019. 5

[59] Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A fair comparison of graph neural networks for graph classification. *CoRR*, abs/1912.09893, 2019. URL http://arxiv.org/abs/1912.09893. 9

[60] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *CoRR*, abs/2007.08663, 2020. URL https://arxiv.org/abs/2007.08663.

[61] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: with Applications in R*. Springer, 2013. URL https://faculty.marshall.usc.edu/gareth-james/ISL/. 9

A Details about Experimental Evaluation

We compare our architecture with graph representation learning methods (**EVO** [47], **HONEM** [48], **DeepWalk** [49] and **Node2Vec** [50]) and deep graph learning methods (**GCN** [1], **LGNN** [51], **TGN** [5] and **DBGNN** [4]). For the representation learning models Node2Vec and EVO we adhere to the original configurations, i.e. we use an embedding size of $d = 128$ and a random walk length of $l = 80$, repeated $r = 10$ times. As context size we use $k = 10$. For Node2Vec we select the return parameter (p) and the in-out parameter (q) from the set $0.25, 0.5, 1, 2, 4$. The deep learning models (GCN, LGNN, DBGNN, and our proposed model) consist of three layers. Following the approach of [4], we set the size of the last layer to $h_2 = 16$, while the sizes of the preceding layers are determined during model selection. The study range for h_0 and h_1 encompasses $4, 8, 16, 32$ over a maximum of 5000 epochs as per [4]. The higher-order path length is fixed to $k = 2$ for HYPA-DBGNN and DBGNN because it is shown as optimal by Qarkaxhija et al. [4] for the given data sets. Stochastic Gradient Descent (SGD) serves as our optimization function, with the learning rate set to 0.001, which showed the best performances. We use dropout regularization with a dropout rate of 0.4 to mitigate overfitting and we incorporate class weights in the loss function to address imbalanced training datasets.

The data sets used do not have an independent test set, nor are they large enough to define a robust dedicated test set. This makes it challenging to directly evaluate model performance on unknown data. To compare various Graph Neural Network (GNN) architectures, we adopt a conventional approach as documented in literature [59–61]. For the assessment of model generalizability, we employ a nested cross-validation strategy with $N = 10$ repetitions. The data undergoes stratified partitioning into nine training and one testing fold, further divided into stratified training and validation subsets (80/20%) within each repetition. Subsequently, we select the best-performing model and epoch based on its validation set performance. Finally, we evaluate the chosen model's performance on the test set, reporting the mean and standard deviation of the respective metric across all N repetitions. For comparability, we use the same folds and splits for all experiments. Besides the random splits, the random initialization of the model also contributes to the variability captured by the standard deviation. For reproducibility, we fix the random splits and reuse a common seed in every repetition for the random initialization of model weights and dropout candidates.

B Model Architecture

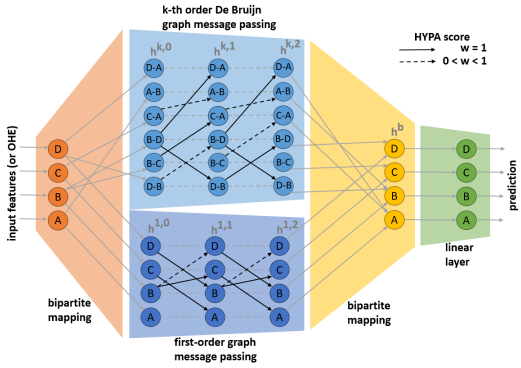

Figure 2: Illustration of the HYPA-DBGNN architecture. The architecture uses node features as inputs. In our case the node id is given as a one-hot encoding even though present features might be used. The bipartite mapping propagates the first-order node features to the first- and higher-order graph. The edge weights are given as HYPA scores such that the HYPA scores define the graph used for neural message passing. Under-represented edges with $HYPA^{(k)}(u, v) = 0$ are removed from the graph. Hence the used graphs are defined through the statistical model. The second bipartite layer merges the lower- and higher-order embedding after three message-passing layers. A final linear layer converts the embedding to the class prediction.

C Similarities in Temporal Sequences Between Empirical and Synthetic Data

The synthetic data set encodes a pattern of increased class-assortativity that is learned by HYPA-DBGNN. Figure 3 shows the deviation from the expected edge frequencies in terms of HYPA scores for the used data sets regarding the incident nodes, i.e. for each node the distribution of the average HYPA score of incident edges is plotted.

The second-order plot shows the increased class-assortativity for nodes of class 0 in the *Weighted Sampling* data set. The incident second-order edges have on average a larger HYPA score and thus are more often overrepresented compared to edges incident to nodes of class 1. Due to the statistic principled inferred graph, HYPA-DBGNN is able to learn this pattern.

Also, *Hospital* and *Workplace2016* emit such under- and overrepresented sequential patterns in both graphs that are related to distinct node classes. In *Hospital* second-order edges incident to nodes of class 0 and 1 are overly often overrepresented. However, the first-order edges incident to nodes of class 0 and 1 differ in its statistics. This observed connection between node classes and the sequential patterns containing the respective nodes supports the superior performance of HYPA-DBGNN for *Hospital* and *Workplace2016*.

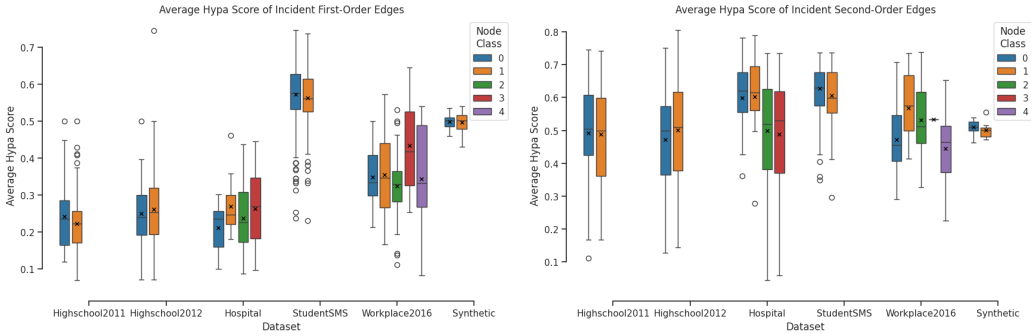

Figure 3: Distribution of average HYPA scores of incident edges. For each node v_j the average HYPA score is determined with $\overline{HYPA}^{(k)}(v_j) = \frac{1}{|S(v_j)|}\sum_{(v_i,v_j)\in S(v_j)} HYPA^{(k)}(v_i, v_j)$ with the incident edges $S(v_j) = \{(v_i, v_j) \in E^{(k)} : v_i \in V^{(k)}\}$. The box plots show the distribution of theses scores with respect to node classes. The synthetic data set is *Weighted Sampling*.

D Comments on Computational Complexity

There are two distinct steps to be considered when arguing about the complexity of our approach. First, there is the preprocessing step that creates the augmented graphs,i.e., the competition of the HYPA scores and the removal of the under-represented paths. Second, the graph neural network is trained on that graphs. For both steps, the complexity is determined by the number of edges in the higher-order De Bruijn graph. In the preprocessing, we calculate the HYPA score for higher-order edges.

The worst-case for the number of higher-order edges is given by the number of different sequences of length k, i.e., $|V|^k$ for a network with $|V|$ nodes. However, two arguments show that we can expect much lower complexity in real-world data. First of all, real-world networks are usually sparse, which implies that most sequences cannot occur as they would otherwise violate the network topology.

LaRock et al. [46] use this argument, and prove that the complexity of their algorithm can be tightened with $\Delta G^{(k)} \leq |V|^2\lambda_1^k$, where $|V|$ denotes the number of nodes in the first-order graph G and λ_1 is the leading eigenvalue of the binary adjacency matrix of G. They conclude, that the HYPA score calculation scales linearly with the number of paths N in the given data set for sparse real-world graphs, a moderate order k, and a sufficiently large N. [4] also uses the argument of sparsity to further limit the complexity of the De Bruijn graph. They note that the number of walks of length k becoming higher-order edges in the higher-order De Bruijn graph is also limited by $\sum_{ij} A^k_{ij} \leq |V|^k$, where A^k is the k-th power of the binary adjacency matrix A of G.

Furthermore, higher-order networks are even sparser than what we would expect based on the first-order topology. This is because the number of different time-respecting paths occurring on a network is generally much lower than the number of possible paths. [4] demonstrate this (see in the appendix) by plotting the number of realized walks at each length and showing that in empirical graphs only a small fraction of walks is realized due to the restriction to time-respecting paths. By studying

the complexity of the used empirical data set, they argue that De Bruijn graphs are applicable to real-world tasks.

We consider a path data set S with N entries. The number of edges in the k-th-order De Bruijn graph is denoted as $\Delta G^{(k)}$. LaRock et al. [46] state that the asymptotic runtime of HYPA is $O(N + \Delta G^{(k)})$. A trivial upper-bound for $\Delta G^{(k)}$ is the fully connected case with $|V|^{k+1}$. This trivial case is also considered by [4] when they argue that the complexity of message passing on the De Bruijn graph is bounded.

E Properties of Empirical Data

Table 3: Overview of time series data and ground truth node classes used in the experiments. δ describes the maximum time difference for edges to be considered part of a casual walk.

Data Set	Ref.	$\|V\|$	$\|E\|$	$\|V^{(2)}\|$	$\|E^{(2)}\|$	Classes (Sizes)	δ
Highschool2011	[52]	126	3355	3042	17141	2 (85/41)	4
Highschool2012	[52]	180	4399	3965	20614	2 (132/48)	4
Hospital	[53]	75	2052	2028	15500	4 (29/27/11/8)	4
StudentSMS	[54]	429	1160	733	846	2 (314/115)	40
Workplace2016	[55]	92	1491	1431	7121	5 (34/26/15/13/4)	4

F Additional Results

Table 4: Comparison of our architecture (HYPA-DBGNN) with different machine learning models. The balanced accuracy is given in Table 2. The results are obtained as described in Appendix A. The best results are marked.

Data Set	Model	F1-score-macro	Precision-macro	Recall-macro
Highschool2011	EVO	39.51 ± 11.50	39.38 ± 19.64	43.68 ± 10.91
	HONEM	57.54 ± 11.52	58.19 ± 13.09	59.00 ± 10.61
	DeepWalk	53.70 ± 18.55	53.47 ± 19.61	54.64 ± 17.70
	Node2Vec	53.70 ± 18.55	53.47 ± 19.61	54.64 ± 17.70
	GCN	48.55 ± 15.49	49.45 ± 18.52	55.00 ± 13.37
	LGNN	52.66 ± 14.71	53.57 ± 15.97	57.72 ± 9.85
	DBGNN	57.08 ± 11.35	61.78 ± 10.75	61.54 ± 11.13
	TGN	57.32 ± 9.84	59.56 ± 10.59	61.52 ± 11.25
	HYPA-DBGNN	**59.60 ± 15.04**	**62.55 ± 14.38**	**63.25 ± 16.18**
Highschool2012	EVO	46.83 ± 9.44	47.97 ± 18.15	50.05 ± 7.30
	HONEM	50.58 ± 9.49	53.89 ± 15.27	50.49 ± 9.31
	DeepWalk	48.79 ± 13.02	49.75 ± 13.77	49.65 ± 12.97
	Node2Vec	48.79 ± 13.02	49.75 ± 13.77	49.65 ± 12.97
	GCN	54.53 ± 10.82	56.94 ± 12.00	59.35 ± 11.13
	LGNN	45.32 ± 16.88	51.43 ± 14.63	51.43 ± 17.94
	DBGNN	60.22 ± 13.73	63.18 ± 12.57	64.93 ± 15.26
	TGN	38.32 ± 5.37	35.86 ± 5.36	41.52 ± 6.19
	HYPA-DBGNN	**60.58 ± 12.12**	**66.23 ± 13.01**	**66.41 ± 10.24**
Hospital	EVO	20.05 ± 6.64	19.12 ± 9.20	25.00 ± 7.86
	HONEM	34.88 ± 18.22	36.88 ± 23.53	37.50 ± 17.35
	DeepWalk	20.00 ± 9.53	18.76 ± 9.68	23.89 ± 10.91
	Node2Vec	20.00 ± 9.53	18.76 ± 9.68	23.89 ± 10.91
	GCN	37.38 ± 8.67	33.83 ± 8.00	43.47 ± 9.03
	LGNN	35.81 ± 8.96	32.75 ± 10.64	44.03 ± 9.03
	DBGNN	47.87 ± 20.02	48.21 ± 21.79	51.67 ± 20.34
	TGN	46.50 ± 13.60	50.83 ± 8.89	49.16 ± 16.95
	HYPA-DBGNN	**71.80 ± 19.18**	**71.50 ± 20.95**	**74.31 ± 17.45**
StudentSMS	EVO	54.62 ± 7.73	55.63 ± 9.53	55.05 ± 6.39
	HONEM	52.46 ± 9.71	55.65 ± 14.29	53.81 ± 7.28
	DeepWalk	52.08 ± 7.19	53.18 ± 7.61	52.78 ± 7.83
	Node2Vec	51.87 ± 7.39	52.13 ± 6.90	52.31 ± 7.70
	GCN	53.85 ± 6.39	54.39 ± 6.27	54.50 ± 6.40
	LGNN	46.79 ± 5.27	52.70 ± 6.07	52.71 ± 6.63
	DBGNN	56.87 ± 5.05	58.55 ± 5.58	57.72 ± 5.29
	TGN	48.98 ± 4.50	50.71 ± 3.10	50.67 ± 4.10
	HYPA-DBGNN	**60.47 ± 6.68**	**61.40 ± 7.00**	**60.66 ± 6.11**
Workplace2016	EVO	22.74 ± 12.34	21.84 ± 14.18	26.50 ± 12.08
	HONEM	77.75 ± 11.70	79.53 ± 13.50	79.46 ± 10.32
	DeepWalk	17.23 ± 8.77	16.30 ± 9.42	20.54 ± 9.51
	Node2Vec	17.23 ± 8.77	16.30 ± 9.42	20.54 ± 9.51
	GCN	68.56 ± 14.78	66.21 ± 16.88	73.33 ± 12.60
	LGNN	82.96 ± 15.65	84.32 ± 15.04	84.83 ± 14.77
	DBGNN	81.16 ± 19.16	81.33 ± 20.14	84.42 ± 15.59
	TGN	78.71 ± 20.32	79.95 ± 22.21	80.16 ± 18.71
	HYPA-DBGNN	**85.82 ± 12.23**	**85.42 ± 13.75**	**88.29 ± 10.51**

G Variants of HYPA-DBGNN

In this section, we present other variations of our main HYPA-DBGNN architecture.

G.1 Base Architecture without Anomalies (HYPA-DBGNN$^-$)

Replacing the HYPA scores with the absolute edge frequencies in the message passing procedure leads to the original message passing layers proposed by Kipf and Welling [1]. The overall structure including the bipartite layers is kept. The comparison of this model (HYPA-DBGNN$^-$) with HYPA-DBGNN reinforces the understanding of the significance of HYPA scores.

G.2 Edge Embedded HYPA Scores (HYPA-DBGNN^E)

For HYPA-DBGNN the HYPA scores are used in a graph model selection step to enhance the message passing. Whereas for HYPA-DBGNN^E the HYPA scores are understood as additional edge attributes whose significance is learned by an adapted graph convolution operation that embeds the edge attributes into the incident node attributes during message passing in the first graph neural network layers. The augmented propagation rule is given as

$$\vec{h}_{v_i}^{k,1} = \sigma\left(\sum_j \frac{1}{c_{ij}}\left(\vec{h}_{v_j}^{k,0} W^{k,1} + \vec{h}_{e_{ij}^k} W^{k,e}\right)\right), \tag{2}$$

with the first hidden representation $\vec{h}_{v_j}^{k,0}$ of node $u \in V^{(k)}$, the inferred HYPA scores in $\vec{h}_{e_{ij}^k}$ for the k-th-order edge $e_{ij}^k \in E^{(k)}$, the trainable weight matrices $W^{k,1} \in \mathbb{R}^{H^1 \times H^0}$ for the nodes and $W^{k,e} \in \mathbb{R}^{H^1 \times 1}$ for the edges and the normalization factor c_{ij} as defined by Kipf and Welling [1].

G.3 Z-Score as Replacement for HYPA Scores (HYPA-DBGNN^Z)

The HYPA scores are based on the CDF. A a replacement for the CDF, a transformed Z-score instead of the HYPA score is implemented in HYPA-DBGNN^Z. The underlying soft configuration model provides the needed expected value and variance with

$$\mathbb{E}[X_{ij}] = m\frac{\Xi_{ij}}{M} \tag{3}$$

and

$$Var[X_{ij}] = m\frac{M-m}{M-1}\frac{\Xi_{ij}}{M} \tag{4}$$

needed to define the Z-score as

$$z(A_{ij}) = \frac{A_{ij} - \mathbb{E}[X_{ij}]}{\sqrt{Var[X_{ij}]}}. \tag{5}$$

Opposing to the HYPA score the Z-score is unbounded and possibly negative. Edges with negative Z-score are excluded because they are under-represented. Likewise in HYPA-DBGNN in most cases under-represented edges are removed, too, because their HYPA scores is approximately zero. Additionally, edges with a Z-score smaller than one are removed with the same argument of not having an unexpected large contribution to the graph and only beeing larger than 0 due to noisy fluctuations in the frequencies. The resulting restricted Z-score is logarithmically transformed due to observed large spread in empirical data, leading to the final replacement for the HYPA-score:

$$z'(e_{ij}) = \begin{cases} 0 & \text{if } z(e_{ij}) < 1, \\ \log(z(e_{ij})) & \text{otherwise} \end{cases} \tag{6}$$

H Ablation Study - Impact of Statistical Information

We conduct an ablation study in which we compare our architectures HYPA-DBGNN, HYPA-DBGNN^E and HYPA-DBGNN^Z to the base architecture HYPA-DBGNN$^-$ that is not using statistical

information. We aim to answer the question of what effect the addition of statistical information has on the prediction capability of the architectures in Table 5.

By comparing HYPA-DBGNN to HYPA-DBGNN$^-$ we see that the statistical information play an important role for all data sets but most importantly it becomes visible that the improvements for *Hospital* are indeed related to the additional information.

HYPA-DBGNN^E with edge encoded statistical features performs better than the uninformed baseline but is most of the time significant weaker than HYPA-DBGNN. The structural graph correction applied in HYPA-DBGNN is still missing even when the edge encoder is able to learn the significance of the HYPA scores. HYPA-DBGNN^Z performs weak for data sets where we don't see direct patterns in the analysis but works well for *Hospital*. It needs to be explored why the Z-score is more susceptible for data sets with weak or no patterns.

Table 5: Ablation study for HYPA-DBGNN. The best results are marked.

Model	Highschool2011	Highschool2012	Hospital	StudentSMS	Workplace2016
HYPA-DBGNN	**63.25 ± 16.18**	**66.41 ± 10.24**	**76.39 ± 17.12**	**60.66 ± 6.11**	88.29 ± 10.51
HYPA-DBGNN^E	61.54 ± 13.62	64.94 ± 17.71	59.03 ± 12.72	60.46 ± 9.42	**88.50 ± 13.57**
HYPA-DBGNN^Z	53.97 ± 17.59	59.63 ± 15.74	69.31 ± 11.74	53.45 ± 7.50	88.42 ± 10.88
HYPA-DBGNN$^-$	57.67 ± 17.16	64.49 ± 15.27	55.83 ± 19.27	56.23 ± 10.41	86.46 ± 12.65

I Ablation Study - Impact of Individual Parts

Table 6: Ablation study for HYPA-DBGNN showing the balanced accuracy. Subsequently parts of the model are removed. (a) contains the complete HYPA-DBGNN model. In (b) the HYPA scores are removed such that the statistical information are not passed to the model. In (c) we further remove the first bipartite layer that maps the first-order node features to the second-order nodes and replace it by a second-order one-hot encoding (OHE). In (d) we additionally remove the complete second-order message passing (MP).
In (e) we use the base HYPA-DBGNN but replace the first-order OHE with available features. Only Highschool2011 and Highschool2012 contain node features. Those are the classes the students belong to. We suspect that those features are not informative for the given prediction task.

Model	Highschool2011	Highschool2012	Hospital	StudentSMS	Workplace2016
(a) base HYPA-DBGNN	**63.25 ± 16.18**	**66.41 ± 10.24**	**76.39 ± 17.12**	**60.66 ± 6.11**	**88.29 ± 10.51**
(b) without HYPA scores	57.67 ± 17.16	64.49 ± 15.27	55.83 ± 19.27	56.23 ± 10.41	86.46 ± 12.65
(c) OHE instead bipartite layer	61.54 ± 11.13	64.93 ± 15.26	52.50 ± 19.27	57.72 ± 5.29	84.42 ± 15.59
(d) without second-order MP	55.00 ± 13.37	59.35 ± 11.13	43.47 ± 9.03	54.50 ± 6.40	73.33 ± 12.60
(e) HYPA-DBGNN with features	59.12 ± 20.24	62.43 ± 10.06	-	-	-

J TGN Adaptations

We implement TGN as proposed by Rossi et al. [5] Instead of a link prediction layer, we add a node prediction layer as the last stage. The embedding size is fixed to 32 as for the other models. For TGN the training procedure is adapted due to its dynamic origin. The proposed training procedure for dynamic node predictions splits the events into fixed-size temporal batches and predicts the next node state for the nodes affected by the events. The batches are temporally divided into train and test batches. Opposing, the static prediction task splits the nodes into train and test sets. We try to keep as much from the original training procedure as possible to favor the memory based architecture. Hence, we train the model on all event batches of size 200 but restrict the training nodes to the train set with fixed class. The last prediction for the given test nodes is used to evaluate the performance. This is not necessary in the last batch of events. For the synthetic data the batch size is increased to 200.000 since each of the 2^{23} events has its own timestamps which leads to infeasible training time with lower batch sizes. Compared to the other deep learning methods the model the losses are updated more often because they are updated for every event batch and not only for every node batch. Consequently, we adapt the learning rate to 0.0001 and the originally used optimizer Adam to obtain improved results.

K Reference Implementation

A reference implementation is given at https://github.com/jvpichowski/HYPA-DBGNN