Link Prediction with Untrained Message Passing Layers

Lisi Qarkaxhija[*] **Anatol E. Wegner**[*] **Ingo Scholtes**
Chair of Machine Learning for Complex Networks
Center for Artificial Intelligence and Data Science (CAIDAS)
Julius-Maximilians-Universität Würzburg, DE
`name.surname@uni-wuerzburg.de`

Abstract

Message passing neural networks (MPNNs) operate on graphs by exchanging information between neigbouring nodes. MPNNs have been successfully applied to various node-, edge-, and graph-level tasks in areas like molecular science, computer vision, natural language processing, and combinatorial optimization. However, most MPNNs require training on large amounts of labeled data, which can be costly and time-consuming. In this work, we explore the use of various untrained message passing layers in graph neural networks, i.e. variants of popular message passing architecture where we remove all trainable parameters that are used to transform node features in the message passing step. Focusing on link prediction, we find that untrained message passing layers can lead to competitive and even superior performance compared to fully trained MPNNs, especially in the presence of high-dimensional features. We provide a theoretical analysis of untrained message passing by relating the inner products of features implicitly produced by untrained message passing layers to path-based topological node similarity measures. As such, untrained message passing architectures can be viewed as a highly efficient and interpretable approach to link prediction.

1 Introduction

Graph neural networks (GNNs) are a powerful class of machine learning models that can learn from graph-structured data, such as social networks, molecular graphs, and knowledge graphs. GNNs have emerged as an important tool in the machine learning landscape, due to their ability to model complex relationships and dependencies within data with applications in a variety of fields where data exhibits a complex topology that can be captured in a graph. This is shown by a multitude of studies, including [1–5], which highlight the versatility and adaptability of GNNs for machine learning tasks across a range of fields.

One of the key concepts underlying GNNs is message passing (MP), as introduced by Gilmer et al. [6], which operates by propagating and aggregating information between nodes in the graph, using message and update functions possibly with learnable parameters. However, designing and training effective GNNs can be challenging, as they may suffer from issues such as over-smoothing or over-parameterization, and since training can be computationally demanding.

In order to address these shortcomings recent efforts have concentrated on finding simplified architectures that are both more interpretable and easier to optimize. In this work, we aim to complement existing works by analysing simplified and untrained architectures from the perspective of link prediction [7]. Link prediction is an important task for graph learning algorithms with applications such as recommender systems, spam mail detection, drug repurposing, and many more [7]. Moreover, we show that link prediction can also provide a complementary perspective for the theoretical analysis of GNNs [8, 9].

[*]Equal contribution

In formulating Untrained Message Passing (UTMP) layers, we follow an approach similar to that of *Simplified Graph Convolutional Networks* introduced by Wu et al.[10]. This approach simplifies GNN architectures by removing trainable parameters and nonlinearities resulting, in an architecture that can be clearly separated into two components: an untrained message passing/feature propagation steps followed by a linear classifier resulting in models that scale to larger data sets while being naturally interpretable.

Recent studies underscore the potential of untrained MPNNs and GNNs in approximating graph measures and identifying robust untrained subnetworks. For instance, [11] demonstrates that MPNNs with random weights can approximate first-order common neighbor measures, supporting the feasibility of training-free GNN models. Complementary research has leveraged sparsity to discover effective untrained subnetworks in GNNs, mitigating issues like over-smoothing and enabling deeper architectures with increased robustness to input perturbations [12]. Building on these foundations, our study shows that UTMP layers, derived from widely-used MPNN frameworks, compute common neighbor measures exactly to arbitrary order, enhancing the precision of these untrained models. Untrained GNNs have also shown promising results in graph classification [13], further supporting the practical viability of training-free architectures. DotHash [14] aligns with these objectives by focusing on set similarity measures, although it does not explore broader connections to GNNs or UTMPs beyond link prediction. Recently, other works have expanded training-free GNN applications to semi-supervised and transductive node classification in text-attributed graphs [15, 16], reflecting a growing interest in training-free methods across diverse tasks.

We base our analysis on untrained versions of four widely used MP architectures, namely Graph Convolutional Networks (GCN)[17], SAGE [18], GraphConv [19] and GIN [20]. We test these UTMP layers on a variety of datasets and find that for the large majority of datasets, untrained message passing layers actually result in higher link prediction performances than their fully trained counterparts while being highly interpretable and much easier to optimize.

In our theoretical analysis we establish a direct connection between features produced by UTMP layers and path based measures. Path based methods and measures can be seen as a way of capturing the indirect connection strength between node pairs in the absence of a direct link connecting them and consequently are widely used in traditional link prediction methods [21, 22] as well as the state of the art link prediction methods [7, 23]. Our theoretical analysis is based on the assumption that initial node features are orthonormal which covers widely used initialization schemes such as as one-hot encodings and high dimensional random features, and also holds approximately for many empirical data sets with high dimensional features and provide new insights into the effectiveness of the widely used initialization schemes of one-hot encodings and high dimensional random features in graph representation learning. Our results show that untrained versions of message passing layers are highly amenable to theoretical analysis and hence could potentially serve as an general ansatz for the theoretical analysis of GNNs in settings beyond link prediction.

2 Message Passing Architectures

We start by introducing the notation used throughout the text. Let $G(V, E)$ be an undirected graph with vertex set V, edges $E \subseteq V \times V$. We denote the adjacency matrix of the graph as $\mathbf{A}$ and define $\tilde{\mathbf{A}} := \mathbf{A} + \mathbf{I}$, i.e. the adjacency matrix of G that includes all self-loops. We use $\mathcal{N}(v)$ to denote the neighborhood of a node v and use $\tilde{\mathcal{N}}(v) := \mathcal{N}(v) \cup \{v\}$. Similarly, we denote the degree of a node v as $d(v)$ and $\tilde{d}(v) = d(v) + 1$. Although we restrict our discussion to undirected and unweighted graphs the generalization of our definitions and results to weighted graphs is straightforward. Prior to defining UTMP layers, we first review the GNN architectures these layers are based on.

Graph Convolutional Networks [17]. GCNs were introduced as a scalable approach for semi-supervised learning on graph-structured data. GCNs are based on an efficient variant of convolutional neural networks which operate directly on graphs. The MP layer of GCN is given by: $h_v^{(l)} = W^{(l),\top} \sum_{u \in \tilde{\mathcal{N}}(v)} \frac{1}{\sqrt{\tilde{d}_u \tilde{d}_v}} h_u^{(l-1)}$ where $h_v^{(l)}$ is the feature vector of node v at step l and $W^{(l),\top}$ is the transposed weight matrix for this layer.

GraphSAGE [18]. GraphSAGE is another widely used of GNN architecture that can use different types of functions to aggregate information from neighboring nodes. We use the following version of the SAGE layer: $h_v^{(l)} = W_1^{(l)} \cdot \frac{1}{\tilde{d}_v} \sum_{u \in \tilde{\mathcal{N}}(v)} h_u^{(l-1)}$, which we found to produce superior results for link prediction.

GIN [20]. The Graph Isomorphism Network Convolution (GIN) is a simple architecture that is provably the most expressive among the class of GNNs and is as powerful as the Weisfeiler-Lehman graph isomorphism test. The mathematical formula for GIN is as follows: $h_v^{(l)} = \Theta\big((1+\epsilon) \cdot h_v^{(l-1)} + \sum_{u \in \mathcal{N}(v)} h_u^{(l-1)}\big)$, where Θ denotes a Multilayer Perceptron after each message passing layer.

GraphConv [19]. GraphConv is a generalization of GNNs, designed to take higher-order graph structures at multiple scales into account. The MP layer of GraphConv is defined as follows: $h_v^{(l)} = W_1^{(l)} h_v^{(l-1)} + W_2^{(l)} \sum_{u \in \mathcal{N}(v)} h_u^{(l-1)}$, where $W_{\{1,2\}}^{(l)}$ are learned weight matrices.

2.1 Untrained MP layers

We now define the untrained counterparts of the four MP architectures introduced in the previous section by eliminating non-linearities and replacing all learnable components with identity matrices. Resulting in the following MP functions:

UTGCN: $h_v^{(l)} = \sum_{u \in \tilde{\mathcal{N}}(v)} \frac{1}{\sqrt{\tilde{d}_u \tilde{d}_v}} h_u^{(l-1)}$

UTSAGE: $h_v^{(l)} = \frac{1}{\tilde{d}_v} \sum_{u \in \tilde{\mathcal{N}}(v)} h_u^{(l-1)}$

UTGIN/UTGraphConv: $h_v^{(l)} = \sum_{u \in \tilde{\mathcal{N}}(v)} h_u^{(l-1)}$

Where we set $\epsilon = 0$ for GINs which results in the same formula for UTGIN and UTGraphConv.

The untrained message passing layers can also be expressed in matrix form: $\mathbf{H}^{(l)} = \mathbf{S}\mathbf{H}^{(l-1)} = \mathbf{S}^l \mathbf{H}^{(0)}$, where $\mathbf{H}^{(0)} \in \mathbb{R}^{n \times d}$ is the initial feature matrix, and $\mathbf{H}^{(l)}$ the feature matrix after l iterations of message passing. Following, the definitions of UTMP layers above we have $\mathbf{S} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$ for UTGCN, $\mathbf{S} = \tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}$ for UTSAGE and $\mathbf{S} = \tilde{\mathbf{A}}$ for UTGIN, where $\tilde{\mathbf{D}}$ is the degree matrix. The generalization of UTMP layers to undirected weighted graphs can be obtained by simply replacing the relevant matrices with their weighted counterparts.

Simplified architectures. Following the construction of Wu et al. for the case of node classification we add a final trained linear layer before the final dot product resulting in an architecture where the final node features are given by: $\mathbf{H}^{(l)} = \Theta \mathbf{S}^l \mathbf{H}^{(0)}$, where Θ is the learned weight matrix of the linear layer. We refer to such architectures as 'simplified' in accordance with [10] and include an 'S' in the abbreviations of these models, e.g. SGCN.

In the case of link prediction features produced by UTMP layers can also be used to construct fully untrained architectures that only consist of feature propagation steps followed by an inner product. In practice we found that such architectures based solely on UTMP layers do work well showing that UTMP layers produce highly informative features for link prediction.

2.2 UTMP layers and path based measures

In the following we show that the inner products of features resulting from untrained message passing layers can be related to path based measures. Such path based measures quantify the indirect connection strength between node pairs in the absence of a direct link connecting the nodes. For this we will assume that initial feature vectors are pairwise orthonormal i.e. $< h_v^{(0)}, h_u^{(0)} >= \delta_{u,v}$. Although the condition of orthonormality might seem quite restrictive at first glance it applies in many practical settings, though in some cases only approximately. High dimensional features of empirical data sets also show similar characteristics to their random counterparts. For instance, empirical feature vectors of randomly selected node pairs tend to be approximately orthogonal, although features of connected node pairs can be highly correlated [24], as can be verified experimentally (see Sec.E).

A path of length l is defined as a sequence of $l+1$ vertices $(v_0, v_1 \dots v_l)$ such that $(v_i, v_{i+1}) \in E$ for all $0 \leq i < l$. We denote the space of a set of all paths of length l between u and v as P_{uv}^l. The number of paths of length l between any u and v is given by the l^{th} power of the adjacency matrix i.e.

$|P^l_{uv}| = \tilde{\mathbf{A}}^l_{uv}$. Similarly, paths of length l between vertices u and v also determine the probability of a random walk starting at u reaching v which in matrix form is given by $P(u \xrightarrow{l} v) = (\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}})^l_{uv}$.

Now we consider inner products of features after l iterations of message passing: $< h^{(l)}_u, h^{(l)}_v >= (\mathbf{S}^l\mathbf{H}^{(0)}\mathbf{H}^{(0)\top}(\mathbf{S}^l)^\top)_{uv}$. For orthonormal features we have $\mathbf{H}^{(0)}\mathbf{H}^{(0)\top} = \mathbf{I}$ and the inner product reduces to: $< h^{(l)}_u, h^{(l)}_v >= (\mathbf{S}^l(\mathbf{S}^l)^\top)_{uv}$. For UTGCN we have $\mathbf{S} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$ and the inner product reduces to: $< h^{(l)}_u, h^{(l)}_v >= \frac{1}{\sqrt{\tilde{d}(u)\tilde{d}(v)}} \sum_{p \in P^{2l}_{uv}} \prod_{i \in [p]} \frac{1}{\tilde{d}_i}$, where $[p]$ denotes the path p with the first and last vertices removed. This in turn is equivalent to $\sqrt{P(u \xrightarrow{2l} v)P(v \xrightarrow{2l} u)}$ i.e. the geometric mean of the probabilities that a random walk starting at either u or v to reaches the other in $2l$ steps. For UTSAGE we have $\mathbf{S} = \tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}$ and the inner product is given by: $< h^{(l)}_u, h^{(l)}_v >= \sum_{p \in P^{2l}_{uv}} \prod_{i \in p - m(p)} \frac{1}{\tilde{d}_i}$, where $m(p)$ is the midpoint of the path p. This is equivalent to $< h^{(l)}_u, h^{(l)}_v >= \sum_i P(u \xrightarrow{l} i)P(v \xrightarrow{l} i)$ and hence corresponds to the probability that two simultaneous random walks starting at u and v, respectively, meet after l steps at some midpoint. Finally, for UTGIN we have $\mathbf{S} = \tilde{\mathbf{A}}$ and $< h^{(l)}_u, h^{(l)}_v >= |P^{2l}_{uv}|$. The relation between UTMP layers an other path based measures is discussed in appendix (Sec A).

3 Experiments and Results

We evaluate GNN architectures on a variety data sets summarized in Table 2. The data sets cover both attributed graphs where nodes have high-dimensional features as well as graphs without node features. Data sources and summary statistics of the data sets can be found in Table 2.

Table 1: Link Prediction accuracy for attributed networks as measured by ROC-AUC. Red values correspond to the overall best model for each dataset, and blue values indicate the best-performing model within the same category of message passing layers.

Models	Cora (small)	CiteSeer (small)	Cora	Cora ML	PubMed	CiteSeer	DBLP
GCN	92.82 ± 0.83	91.67 ± 1.14	97.87 ± 0.18	94.67 ± 0.51	97.54 ± 0.09	94.22 ± 0.77	96.63 ± 0.12
SGCN	95.3 ± 0.68	96.22 ± 0.4	98.6 ± 0.07	96.82 ± 0.43	97.94 ± 0.16	95.9 ± 0.77	97.1 ± 0.14
UTGCN	93.82 ± 0.68	96.0 ± 0.24	95.72 ± 0.12	93.93 ± 0.45	94.29 ± 0.24	92.74 ± 0.81	94.91 ± 0.32
SAGE	91.41 ± 0.38	90.78 ± 1.79	97.7 ± 0.08	94.38 ± 0.58	95.6 ± 0.18	93.58 ± 0.87	96.16 ± 0.25
SSAGE	94.47 ± 0.64	95.75 ± 0.29	98.23 ± 0.1	95.74 ± 0.6	95.88 ± 0.22	95.14 ± 0.9	96.29 ± 0.12
UTSAGE	92.77 ± 0.5	96.07 ± 0.43	96.85 ± 0.13	93.45 ± 0.81	88.12 ± 0.29	91.78 ± 0.85	93.36 ± 0.31
GIN	91.65 ± 0.73	90.62 ± 1.17	97.69 ± 0.13	94.54 ± 0.32	96.27 ± 0.13	92.88 ± 0.87	96.11 ± 0.25
GraphConv	92.06 ± 0.67	91.24 ± 0.32	97.94 ± 0.11	95.34 ± 0.25	96.39 ± 0.15	92.7 ± 0.73	96.09 ± 0.23
SGIN	92.87 ± 0.37	93.6 ± 0.48	97.82 ± 0.11	95.26 ± 0.41	96.37 ± 0.23	94.13 ± 0.4	95.85 ± 0.11
UTGIN	85.45 ± 1.28	85.7 ± 0.82	88.93 ± 0.35	86.86 ± 0.98	88.76 ± 0.25	91.11 ± 0.93	92.25 ± 0.29

Results for attributed graphs are given in Table 1 where we find that the simplified model SGCN performs best on all attributed datasets, with performance comparable to more sophisticated state-of-the-art models [7, 23, 25] (See Table 9 in the Appendix) . Moreover, we find that in general simplified models perform better than or on par with their fully trained counterparts on almost all datasets, with the single exception of GIN on DBLP. We also find that the fully untrained (UT) architectures already provide a very good baseline and some cases even outperform fully trained versions. This demonstrates that the raw features produced by UTMP layers, which the simplified models are trained on, are already highly informative for link prediction in accordance with our theoretical results. The fully untrained (UT) models can be computed very efficiently via sparse matrix multiplication.

4 Conclusion

In this work, we explored the application of untrained message passing layers to link prediction. Our experimental evaluation shows that simplifying GNNs architectures by eliminating trainable parameters and non-linearities not only enhances the link prediction performance of GNNs, but also improves their interpretability and offer a computationally efficient alternative to trained Mp layers counterparts that naturally scales to large graphs while producing results comparable to the state-of-the-art. Link prediction also offers a complementary theoretical perspective, analytically establishing a link UTMP layers and path-based topological measures and widely used initialization schemes such as random features and one-hot encodings.

References

[1] Sourya Basu, Jose Gallego-Posada, Francesco Viganò, James Rowbottom, and Taco Cohen. Equivariant mesh attention networks. *arXiv preprint arXiv:2205.10662*, 2022. 1

[2] Rui Wang, Robin Walters, and Rose Yu. Approximately equivariant networks for imperfectly symmetric dynamics. In *International Conference on Machine Learning*, pages 23078–23091. PMLR, 2022.

[3] Xiang Fu, Tian Xie, Nathan J Rebello, Bradley D Olsen, and Tommi Jaakkola. Simulate time-integrated coarse-grained molecular dynamics with geometric machine learning. *arXiv preprint arXiv:2204.10348*, 2022.

[4] Priyank Jaini, Lars Holdijk, and Max Welling. Learning equivariant energy based models with equivariant stein variational gradient descent. *Advances in Neural Information Processing Systems*, 34:16727–16737, 2021.

[5] Omri Puny, Matan Atzmon, Heli Ben-Hamu, Ishan Misra, Aditya Grover, Edward J Smith, and Yaron Lipman. Frame averaging for invariant and equivariant network design. *arXiv preprint arXiv:2110.03336*, 2021. 1

[6] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017. 1

[7] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018. 1, 2, 4, 12, 13

[8] Yangze Zhou, Gitta Kutyniok, and Bruno Ribeiro. Ood link prediction generalization capabilities of message-passing gnns in larger test graphs. *Advances in Neural Information Processing Systems*, 35:20257–20272, 2022. 1

[9] Benjamin Paul Chamberlain, Sergey Shirobokov, Emanuele Rossi, Fabrizio Frasca, Thomas Markovich, Nils Hammerla, Michael M Bronstein, and Max Hansmire. Graph neural networks for link prediction with subgraph sketching. *arXiv preprint arXiv:2209.15486*, 2022. 1

[10] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019. 2, 3, 11

[11] Kaiwen Dong, Zhichun Guo, and Nitesh V Chawla. Pure message passing can estimate common neighbor for link prediction. *arXiv preprint arXiv:2309.00976*, 2023. 2

[12] Tianjin Huang, Tianlong Chen, Meng Fang, Vlado Menkovski, Jiaxu Zhao, Lu Yin, Yulong Pei, Decebal Constantin Mocanu, Zhangyang Wang, Mykola Pechenizkiy, et al. You can have better graph neural networks by not training weights at all: Finding untrained gnns tickets. In *Learning on Graphs Conference*, pages 8–1. PMLR, 2022. 2

[13] Jan Böker, Ron Levie, Ningyuan Huang, Soledad Villar, and Christopher Morris. Fine-grained expressivity of graph neural networks. *arXiv preprint arXiv:2306.03698*, 2023. 2

[14] Igor Nunes, Mike Heddes, Pere Vergés, Danny Abraham, Alex Veidenbaum, Alex Nicolau, and Tony Givargis. Dothash: Estimating set similarity metrics for link prediction and document deduplication. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1758–1769, 2023. 2

[15] Kaiwen Dong, Zhichun Guo, and Nitesh V Chawla. You do not have to train graph neural networks at all on text-attributed graphs. *arXiv preprint arXiv:2404.11019*, 2024. 2

[16] Ryoma Sato. Training-free graph neural networks and the power of labels as features. *arXiv preprint arXiv:2404.19288*, 2024. 2

[17] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 2

[18] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017. 2

[19] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4602–4609, 2019. 2, 3

[20] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018. 2, 3

[21] Víctor Martínez, Fernando Berzal, and Juan-Carlos Cubero. A survey of link prediction in complex networks. *ACM computing surveys (CSUR)*, 49(4):1–33, 2016. 2

[22] Ajay Kumar, Shashank Sheshar Singh, Kuldeep Singh, and Bhaskar Biswas. Link prediction techniques, applications, and performance: A survey. *Physica A: Statistical Mechanics and its Applications*, 553:124289, 2020. 2

[23] Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. Neural bellman-ford networks: A general graph neural network framework for link prediction. *Advances in Neural Information Processing Systems*, 34:29476–29490, 2021. 2, 4, 12, 13

[24] Hoang Nt and Takanori Maehara. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550*, 2019. 3

[25] Liming Pan, Cheng Shi, and Ivan Dokmanić. Neural link prediction with walk pooling. *arXiv preprint arXiv:2110.04375*, 2021. 4, 12, 13

[26] Anatol Rapoport. Spread of information through a population with socio-structural bias: I. assumption of transitivity. *The bulletin of mathematical biophysics*, 15:523–533, 1953. 7

[27] Paul W Holland and Samuel Leinhardt. Transitivity in structural models of small groups. *Comparative group studies*, 2(2):107–124, 1971. 7

[28] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150–1170, 2011. ISSN 0378-4371. doi: https://doi.org/10.1016/j.physa.2010.11.027. URL `https://www.sciencedirect.com/science/article/pii/S037843711000991X`. 7

[29] Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social networks*, 25(3): 211–230, 2003. 7

[30] Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. Predicting missing links via local information. *The European Physical Journal B*, 71:623–630, 2009. 7

[31] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953. 7

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

[33] Glen Jeh and Jennifer Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543, 2002. 7

[34] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016. 8

[35] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. *arXiv preprint arXiv:1707.03815*, 2017. 8

[36] Mark EJ Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical review E*, 74(3):036104, 2006. 8

[37] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *Nature*, 393(6684):440–442, 1998. 8

[38] Robert Ackland et al. Mapping the us political blogosphere: Are conservative bloggers more prominent? In *BlogTalk Downunder 2005 Conference, Sydney*. BlogTalk Downunder 2005 Conference, Sydney, 2005. 8

[39] Neil Spring, Ratul Mahajan, David Wetherall, and Thomas Anderson. Measuring isp topologies with rocketfuel. *IEEE/ACM Transactions on networking*, 12(1):2–16, 2004. 8

[40] Vladimir Batagelj and Andrej Mrvar. Usair data. `http://vlado.fmf.uni-lj.si/pub/networks/data/`, 2006. Accessed: 27-01-2024. 8

[41] Christian Von Mering, Roland Krause, Berend Snel, Michael Cornell, Stephen G Oliver, Stanley Fields, and Peer Bork. Comparative assessment of large-scale data sets of protein–protein interactions. *Nature*, 417(6887):399–403, 2002. 8

[42] Muhan Zhang, Zhicheng Cui, Shali Jiang, and Yixin Chen. Beyond link prediction: Predicting hyperlinks in adjacency space. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 8

[43] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 8

[44] Link prediction on pyg. `https://github.com/pyg-team/pytorch_geometric/blob/master/examples/link_pred.py`, 2021. Accessed: 2023-11-21. 8

A Triadic closure and other path based measures

Triadic closure, also known as transitivity, refers to the tendency for nodes in real-world networks to form connections if they share (many) common neighbors. As such triadic closure has been widely studied as a mechanism that drives link formation in complex real-world networks [26, 27]. Moreover, node similarity measures that build on triadic closure in social networks have been used for similarity-based link prediction algorithms [28].

Given a pair of nodes (u, v), the tendency of them to be connected due to triadic closure can be quantified by simply counting the number of common neighbours between the two vertices i.e. $T(u, v) = |N(u) \cap N(v)|$ which corresponds to $l = 1$ for UTGIN, assuming that u and v are not connected in the graph as is customary in a link prediction scenario. In practice, one might further want to account for the fact that in general nodes with higher degrees also have a larger probability of having common neighbours, for instance by normalizing by the degrees, i.e.: $T_d(u, v) = |N(u) \cap N(v)|/\tilde{d}(u)\tilde{d}(v)$, which corresponds to $l = 1$ for UTSAGE. One can go one step further and also take into account the degrees of the common neighbours themselves since high degree nodes are by definition common neighbours of more node pairs, for instance by weighing common neighbours according to their degree $T_n(u, v) = \frac{1}{\sqrt{\tilde{d}(u)\tilde{d}(v)}} \sum_{i \in N(u) \cap N(v)} \frac{1}{\tilde{d}_i}$ which in our case corresponds to UTGCN with $l = 1$.

Our results also link UTMP layers to other topological similarity measures that are widely used in link prediction heuristics such as the Adamic-Adar (AA) index [29], Resource Allocation (RA) [30], the Katz index [31], rooted PageRank [32] and SimRank [33]. For instance the AA index, given by $AA(u, v) = \sum_{i \in N(u) \cap N(v)} \frac{1}{\log \tilde{d}_i}$, and $RA(u, v) = \sum_{i \in N(u) \cap N(v)} \frac{1}{\tilde{d}_i}$ differ only slightly from the triadic closure measures we obtained for UTMP layers. Similar results also hold for other path based measures such as rooted PageRank, the Katz index and SimRank which can be defined in terms of power series over paths of different lengths. For instance, SimRank similarity between nodes u and v is defined as $s(x, y) = \sum_l P_{uv}(l)\gamma^l$ where $P_{uv}(l)$ is the probability that two random walks starting at u and v meet after l steps and $0 < \gamma < 1$ is a free parameter. Similarly the Katz index is defined as $Katz(u, v) = \sum_l \mathbf{A}^l_{uv}\gamma^l$ and rooted PageRank is defined as $PR(u, v) = (1 - \gamma) \sum_l \frac{P(u \xrightarrow{l} v) + P(v \xrightarrow{l} u)}{2} \gamma^l$ again with $0 < \gamma < 1$ being a free parameter. Hence, the Katz index is closely realted to UTGIN and rooted PageRank is closely related to UTGCN, the main difference being that these measures also include paths of odd length which UTGIN and UTGCN include only indirectly through the inclusion of self loops in their formulation.

B Dataset details

Summary statistics of the datasers are given in Tab 2.

C Experimental Setup

To ensure a fair comparison among models we maintain the same overall architectures across all experiments where each trainable message passing layer is followed by an Exponential Linear Unit (ELU) and the optimal number of layers for models is determined via hyperparameter search. Upon completion of the message passing layers, we introduce a final linear layer for both trained and simplified models. We also consider untrained (UT) models that do not include this final linear layer and directly take the inner product between the propagated features of the source and target nodes resulting in a parameter-free and hence fully untrained model.

Table 2: Overview of the datasets, sources, and node features for attributed graphs (top group) used in our experimental evaluation.

Dataset	|**V**|	|**E**|	**Features**	**Metric**
Cora small [34]	2,708	10,556	1,433	ROC-AUC
CiteSeer small [34]	3,327	9,104	3,703	ROC-AUC
Cora [35]	19,793	126,842	8,710	ROC-AUC
Cora ML [35]	2,995	16,316	2,879	ROC-AUC
PubMed [35]	19,717	88,648	500	ROC-AUC
CiteSeer [35]	4,230	10,674	602	ROC-AUC
DBLP [35]	17,716	105,734	1,639	ROC-AUC
NS [36]	1,461	2,742	-	ROC-AUC
Celegans [37]	297	2,148	-	ROC-AUC
PB [38]	1,222	16,714	-	ROC-AUC
Power [37]	4,941	6,594	-	ROC-AUC
Router [39]	5,022	6,258	-	ROC-AUC
USAir [40]	332	2,126	-	ROC-AUC
Yeast [41]	2,375	11,693	-	ROC-AUC
E-coli [42]	1,805	15,660	-	ROC-AUC

For each model, the optimal values of the learning rate, the number of layers, and hidden dimensions are determined through an exhaustive search over the values given in Table 3). The optimal hyperparameters values for attributed and non-attributed datasets are given in Table 4 and Table 5, respectively. We implement a three-fold cross-validation procedure to select the optimal hyperparameter values.

We use Adam [43] as an optimization function and employ binary cross entropy with logits as our loss function. All datasets are transformed by normalizing the node features and randomly splitting the dataset, with 10% allocated to the test set, 5% to the validation set, and the remainder to the training set. Each model configuration is run 10 times, with the results averaged over these runs. Our training and testing procedures are based on the methodology outlined in [44], where we perform a new round of negative edge sampling for each training epoch. We limit the maximum number of epochs to 10,000 and also incorporate an early stopping mechanism in our training process by terminating training whenever there is no improvement in the validation set results over a span of 250 epochs.

For the simplified models we pre-compute node features corresponding to the untrained message passing layers as these do not change during training. We use one-hot encoding as initial node features for the non-attributed datasets. To ensure replicability of our results, we make our code available online [2].

D Hyperparameter choices

All hyperparameter searches and experiments were conducted on a workstation with AMD Ryzen Threadripper PRO 5965WX 24-Cores with 256 GB of memory and two Nvidia GeForce RTX 3090 Super GPU, and also AMD Ryzen 9 7900X 12-Cores with 64 GB of memory and an Nvidia GeForce RTX 4080 GPU.

Table 3: The hyperparameter space for our experiments. It is worth noting that only the number of MPNN layers applies to the untrained models.

Hyperparameter	**Values**
Number of MPNN layers	1,2,3
Learning Rate	0.2, 0.1,0.01, 0.001, 0.0001
Hidden Dimensions	16, 64, 128

[2] https://zenodo.org/records/11237762

Table 4: Optimal hyperparameter values for attributed datasets (MaxEpochs=10,000).

	Cora small			CiteSeer small			Cora			Cora ML			PubMed			CiteSeer			DBLP		
	lr.	hd.	nl.	lr.	hd.	nl.	lr.	hd.	nl.	lr.	hd.	nl.	lr.	hd.	nl.	lr.	hd.	nl.	lr.	hd.	nl.
GCN	0.001	128	1	0.001	128	1	0.001	64	1	0.001	64	1	0.01	64	1	0.01	64	1	0.001	128	1
SGCN	0.001	64	1	0.001	128	1	0.001	128	1	0.001	128	2	0.001	128	1	0.001	128	2	0.2	128	2
UTGCN			2			2			2			2			2			3			2
SAGE	0.01	128	1	0.01	16	1	0.01	128	1	0.001	128	1	0.01	128	1	0.001	128	1	0.001	64	1
SSAGE	0.0001	128	1	0.0001	128	2	0.1	64	1	0.001	64	1	0.001	128	2	0.01	128	3	0.01	64	2
UTSAGE			2			2			2			2			2			2			2
GIN	0.001	128	1	0.001	128	1	0.001	128	1	0.001	128	1	0.001	128	1	0.001	128	1	0.001	128	1
GraphConv	0.0001	64	1	0.0001	128	1	0.001	64	1	0.0001	128	1	0.0001	128	1	0.001	128	1	0.001	128	1
SGIN	0.001	64	1	0.0001	128	2	0.0001	128	1	0.001	64	1	0.01	128	1	0.0001	128	1	0.001	128	1
UTGIN			1			1			1			1			1			1			1

Table 5: Hyperparameter choices for each model in each of the non-attributed dataset.

	NS			Celegans			PB			Power			Router			USAir			Yeast			E-coli		
	lr.	hd.	nl.	lr.	hd.	nl.	lr.	hd.	nl.	lr.	hd.	nl.	lr.	hd.	nl.	lr.	hd.	nl.	lr.	hd.	nl.	lr.	hd.	nl.
GCN	0.01	64	3	0.001	128	1	0.01	128	2	0.001	64	3	0.2	128	3	0.001	128	2	0.01	64	3	0.1	128	1
SGCN	0.1	128	3	0.01	128	2	0.1	128	2	0.001	128	3	0.2	64	3	0.1	64	2	0.01	128	2	0.01	16	1
UTGCN			3			2			2			3			2			2			2			2
SAGE	0.01	64	2	0.01	128	2	0.01	128	2	0.01	64	3	0.2	16	1	0.01	64	1	0.01	64	2	0.01	128	1
SSAGE	0.01	128	1	0.01	16	2	0.01	128	1	0.001	128	3	0.001	64	3	0.1	64	2	0.001	128	1	0.01	128	1
UTSAGE			2			2			2			3			2			2			2			2
GIN	0.001	128	3	0.001	64	1	0.001	128	1	0.01	128	3	0.1	128	2	0.0001	128	2	0.001	128	1	0.01	128	1
GraphConv	0.001	128	1	0.0001	128	1	0.0001	64	1	0.0001	128	3	0.001	16	1	0.01	64	2	0.0001	64	1	0.01	64	1
SGIN	0.0001	128	2	0.01	128	1	0.2	64	1	0.0001	128	2	0.0001	128	1	0.1	64	1	0.001	128	1	0.001	64	1
UTGIN			2			1			1			3			2			2			2			1

E Orthogonality of node features in empirical data sets

The distribution of inner products of initial node features for attributed datasets is given in Figure 1. We find that the inner products of feature vectors of randomly selected node pairs are in general close to zero. Note that, the feature vectors of all datasets are non-negative as they represent word occurrences. As expected, for connected nodes the inner products of feature vectors tend to be higher reflecting the increased feature similarity.

In the case of non-attributed graphs (Table 10) we observe that models based on UTMP layers achieve the highest score on 6 out of 8 datasets, with the exceptions being NS and Router datasets. Moreover, we find that the fully untrained UTGCN model outperforms all other models on the 'Celegans', 'PB', 'USAir', 'E-coli' which can be attributed to the reduced dimension of the learned features that come with the linear layers present in the simplified and fully trained models. In contrast to the trained variants UTGCN maintains the higher initial dimensionality of the features and hence can more efficiently discriminate between local neighborhoods by maintaining orthogonality. Furthermore, as we used one hot encodings as initial node features for the unattributed datasets orthonormality is satisfied exactly and therefore there is a one-to-one correspondence between node similarities produced by UTGCN and path based topological measures.

F Additional experimental results

Table 6: Link Prediction accuracy for non-attributed networks as measured by ROC-AUC. Red values correspond to the overall best model for each dataset, and blue values indicate the best-performing model within the same category of message passing layers.

Models	NS	Celegans	PB	Power	Router	USAir	Yeast	E-coli
GCN	95.22 ± 1.8	87.98 ± 1.45	92.91 ± 0.3	74.68 ± 2.67	91.42 ± 0.44	93.56 ± 1.53	94.49 ± 0.61	98.48 ± 0.22
SGCN	95.17 ± 0.96	89.38 ± 1.42	93.86 ± 0.42	81.08 ± 1.2	77.51 ± 1.85	94.08 ± 1.43	95.74 ± 0.33	98.32 ± 0.2
UTGCN	94.76 ± 1.03	91.47 ± 1.4	94.49 ± 0.38	72.97 ± 1.27	61.68 ± 1.01	94.81 ± 1.1	94.0 ± 0.43	99.37 ± 0.1
SAGE	95.9 ± 0.86	87.32 ± 1.61	92.94 ± 0.57	74.17 ± 2.03	62.6 ± 3.3	93.37 ± 1.2	94.43 ± 0.67	98.22 ± 0.13
SSAGE	95.21 ± 1.09	88.05 ± 1.8	91.66 ± 0.43	81.84 ± 1.49	70.1 ± 1.3	92.25 ± 1.45	95.72 ± 0.31	93.59 ± 0.14
UTSAGE	94.72 ± 1.07	84.48 ± 1.87	86.46 ± 0.64	72.96 ± 1.26	61.47 ± 0.99	87.94 ± 1.58	93.45 ± 0.45	85.56 ± 0.37
GIN	95.24 ± 1.22	86.74 ± 2.3	93.04 ± 0.99	71.97 ± 2.3	87.84 ± 3.05	92.14 ± 0.98	94.7 ± 0.45	98.43 ± 0.24
GraphConv	95.73 ± 1.4	86.64 ± 2.31	92.99 ± 0.87	74.31 ± 1.93	80.84 ± 1.28	91.16 ± 1.76	94.94 ± 0.38	98.32 ± 0.22
SGIN	95.48 ± 0.88	88.31 ± 1.3	93.72 ± 0.48	73.73 ± 1.69	72.83 ± 1.28	93.02 ± 1.37	95.63 ± 0.49	97.68 ± 0.2
UTGIN	94.62 ± 1.05	86.48 ± 1.29	92.77 ± 0.51	72.93 ± 1.27	61.67 ± 1.02	93.44 ± 0.84	92.94 ± 0.41	95.81 ± 0.22

G Increased numbers of layers

Finally, we also examine the effect of increasing the number of UTMP layers using fully untrained (UT) models. Our results in Fig.2 indicate that, in general, UTGCN and UTSAGE maintain their

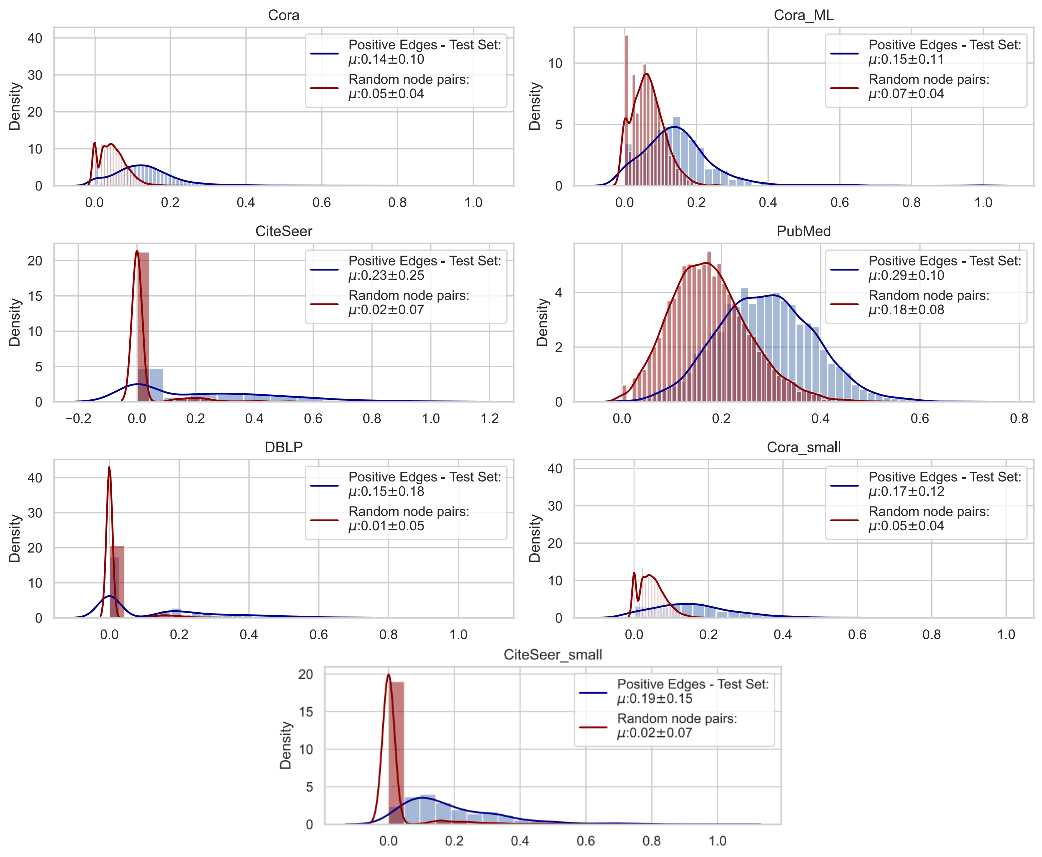

Figure 1: The distribution of feature dot products for pairs of connected and random node pairs for the attributed datasets.

performance as the number of layers is increased whereas the performance of UTGIN decreases sharply with more layers. This behavior can be attributed to the lack of degree based normalization in the formulation of GIN (see Sec.2.2) which leads UTGIN to be dominated by longer paths, and hence longer distance correlations, as the number of layers increases.

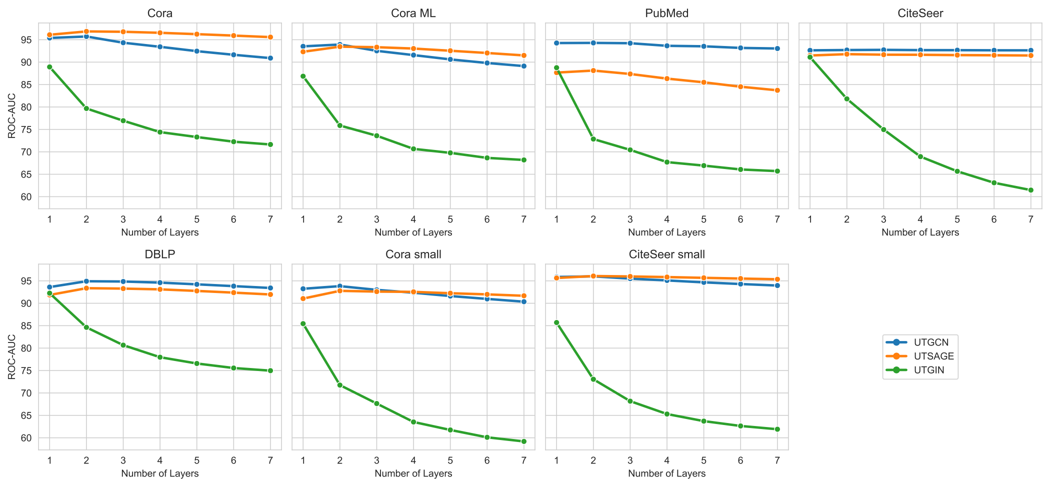

Figure 2: The effect of increased layer size for fully untrained models.

H Runtime Analysis and Training Efficency

Efficiency of SMPNNs. While we allocated a very generous limit of 10,000 epochs for training models in the main paper to ensure models can reach their best possible performance in order to compare the computational efficiency of the simplified models to their fully trained counterparts we also consider an experimental setting where we restrict the maximum number of training epochs to 100. We find that simplified models achieved convergence even for larger learning rates and considerably faster than their fully trained models. Even when constrained to 100 training epochs simplified models maintain scores that are almost identical to those presented in Table 8, while fully trained architectures suffer from the increased learning rates and require in general more epochs to converge. This leads to training efficiency gains similar to those reported by [10] in the case of node classification.

In Table 8, it is evident that the simplified models consistently outperform the fully trained models across all datasets by a considerable margin. Furthermore, as demonstrated in Table 1, the fully trained models nearly achieve their peak accuracy within just 100 epochs, indicating that extended training offers minimal additional benefit. This also implies that the Simplified models are more efficient in terms of both time and resources required for training.

The hyperpameter space used for the computational efficiency experiments is the same as in Table4, except that we only use 100 epochs.

Efficiency of UTMP. In Figure 3, we presented the training times for both simplified and fully trained models. The prediction times for UT models are excluded, as they require only a single "epoch" for making the predictions, unlike other methods that necessitate prolonged training periods. This characteristic of UT models leads to a substantial reduction in both time consumption and electricity costs.

Despite a minor trade-off in accuracy on attributed graphs, UT models frequently outperform in terms of accuracy on unattributed graphs across numerous datasets. In practical applications, the efficiency of UTMP models could translate to significant savings in energy consumption and hence environmental footprint which can outweigh marginal improvements in accuracy in settings where either computational resources are limited or reducing energy consumption/cost and environmental impact of models take priority. This makes UT models particularly appealing for large-scale applications where operational efficiency and cost reduction are critical. Additionally, the societal impact of using UT models includes a lower environmental footprint due to reduced energy consumption, aligning with sustainable and environmentally friendly practices.

Table 7: Optimal hyperparameter values for attributed datasets for MaxEpochs=100.

	Cora small			**CiteSeer small**			**Cora**			**Cora ML**			**PubMed**			**CiteSeer**			**DBLP**		
	lr.	hd.	nl.	lr.	hd.	nl.	lr.	hd.	nl.	lr.	hd.	nl.	lr.	hd.	nl.	lr.	hd.	nl.	lr.	hd.	nl.
GCN	0.01	128	1	0.01	128	1	0.01	64	1	0.01	64	1	0.01	64	1	0.01	64	1	0.01	128	1
SGCN	0.1	128	1	0.1	128	2	0.01	128	1	0.01	128	1	0.01	128	1	0.01	64	2	0.1	128	2
SAGE	0.01	128	1	0.01	128	1	0.01	64	1	0.01	64	1	0.01	128	1	0.01	128	1	0.01	128	1
SSAGE	0.2	128	2	0.1	128	2	0.01	128	1	0.01	128	1	0.01	128	1	0.01	64	1	0.1	128	2
GIN	0.001	128	1	0.001	128	1	0.001	128	1	0.001	128	1	0.001	128	1	0.01	64	1	0.01	64	1
GraphConv	0.001	128	1	0.001	128	1	0.001	128	1	0.001	128	1	0.001	128	1	0.01	128	1	0.001	128	1
SGIN	0.001	128	1	0.001	128	1	0.001	128	1	0.001	128	1	0.001	128	1	0.001	128	1	0.001	128	1

Table 8: Link Prediction accuracy for attributed networks as measured by ROC-AUC. Red values correspond to the overall best model for each dataset, and blue values indicate the best-performing model within the same category of message passing layers. The models are trained only for MaxEpochs = 100.

Models	**Cora small**	**CiteSeer small**	**Cora**	**Cora ML**	**PubMed**	**CiteSeer**	**DBLP**
GCN	91.44 ± 1.31	91.48 ± 0.67	96.45 ± 0.29	93.95 ± 0.54	96.56 ± 0.22	93.48 ± 0.81	95.57 ± 0.18
SGCN	94.58 ± 1.27	96.4 ± 0.97	97.99 ± 0.06	96.75 ± 0.3	97.1 ± 0.17	95.41 ± 0.76	96.95 ± 0.1
SAGE	90.2 ± 1.67	90.34 ± 1.87	95.42 ± 0.22	92.53 ± 0.69	92.68 ± 0.5	91.29 ± 1.32	94.36 ± 0.32
SSAGE	93.98 ± 1.08	95.77 ± 1.02	97.72 ± 0.08	95.61 ± 0.38	94.52 ± 0.18	94.48 ± 0.96	96.34 ± 0.12
GIN	90.39 ± 0.6	88.27 ± 0.61	95.38 ± 0.29	93.75 ± 0.24	94.84 ± 0.28	90.94 ± 0.72	94.71 ± 0.26
GraphConv	91.57 ± 1.33	90.79 ± 0.91	96.68 ± 0.16	94.56 ± 0.48	95.17 ± 0.3	92.04 ± 0.96	94.94 ± 0.11
SGIN	92.72 ± 1.23	93.11 ± 0.25	97.29 ± 0.08	95.43 ± 0.27	95.95 ± 0.21	93.18 ± 0.56	95.84 ± 0.15

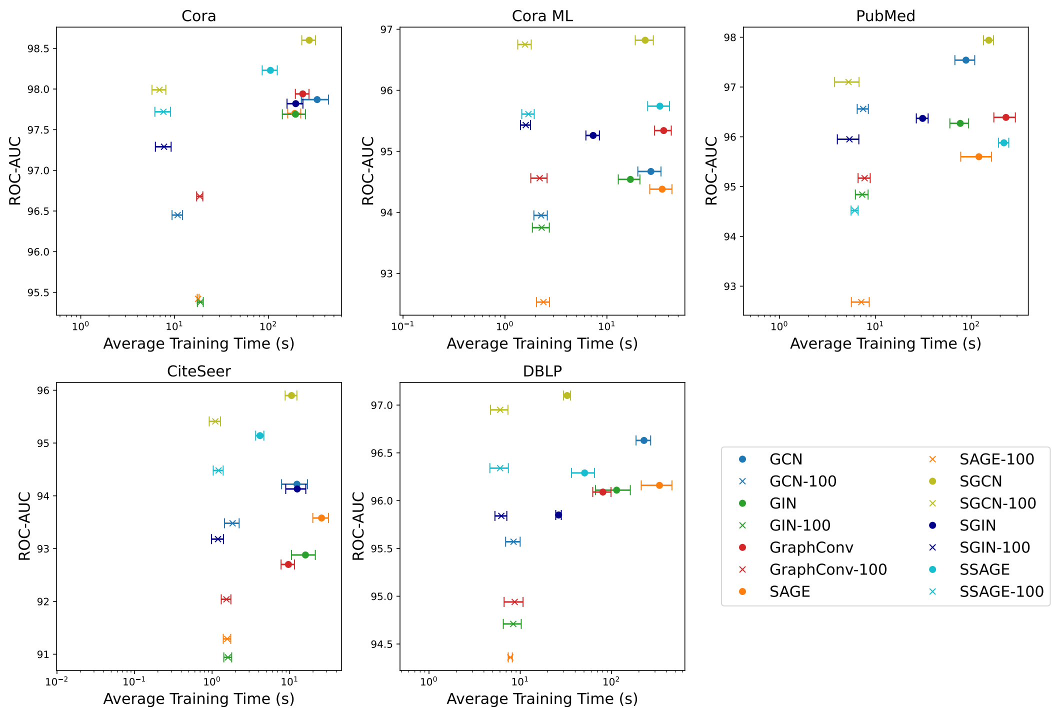

Figure 3: Average runtimes (in seconds) for training and inference for attributed data sets.

Figure 3 illustrates that the simplified models, when trained for extended periods, generally achieve higher accuracy and converge faster to their optimal values compared to fully trained models. Notably, when trained for a shorter duration (100 epochs), the simplified models not only outperform the fully trained counterparts by a larger margin but also require considerably fewer epochs to reach relatively high accuracies. Additionally, the accuracy gap between shorter and longer training durations is smaller for simplified models than for fully trained models.

I Comparison to state-of-the-art

In Table 9 we compare UTMP architectures to three state-of-the-art link prediction methods, namely SEAL [7], WalkPool [25] and NBFNet [23] on attributed datasets. Compared to SEAL SGCN achieves higher scores on all 3 datasets. While NBFNet outperforms SGCN on Cora and PubMed by a small margin SGCN achieves a significantly higher score on CiteSeer. For WalkPoll the results on Cora and Citeseer are within one standard deviation of each other while WalkPool does better on PubMed.

For the unattributed datasets (see Table 10) we find that UTGCN has the highest overall score for E-coli while being within a standard deviation of the other methods on Celegans with WalkPool performing best on the remaining datasets. Note that WalkPool can be set up with different base methods as can be seen in Table 9 and hence using SGCN or UTGCN as a base method for WalkPool could potentially lead to performance improvements, which however is beyond the scope of this paper. It should also be noted that SGCN is considerably more efficient than all three methods, which rely either on subgraph extraction (SEAL and WalkPool) or learning explicit path representations (NBFNet).

Table 9: Link Prediction accuracy for attributed networks as measured by ROC-AUC compared to state-of-the-art models SEAL and NBFNet. ROC-AUC values for NBFNet and SEAL are from [23] (no standard deviations reported) and results for WalkPool from [25].

Models	**Cora (small)**	**CiteSeer (small)**	**PubMed**
SEAL	93.3	90.5	97.8
NBFNet	95.6	92.3	98.3
WalkPool (VGAE)	94.64 ± 0.55	94.32 ± 0.90	98.49 ± 0.13
WalkPool (ARGVA)	94.71 ± 0.85	94.53 ± 1.77	98.52 ± 0.14
WalkPool (GIC)	95.90 ± 0.50	95.94 ± 0.59	98.72 ± 0.10
GCN	92.82 ± 0.83	91.67 ± 1.14	97.54 ± 0.09
SGCN	95.3 ± 0.68	96.22 ± 0.4	97.94 ± 0.16
UTGCN	93.82 ± 0.68	96.0 ± 0.24	94.29 ± 0.24

Table 10: Link Prediction accuracy for non-attributed networks as measured by ROC-AUC compared to state-of-the-art models SEAL and WalkPool. ROC-AUC values SEAL are taken from [7] and for WalkPool from [25].

Models	**NS**	**Celegans**	**PB**	**Power**	**Router**	**USAir**	**Yeast**	**E-coli**
SEAL	97.71±0.93	89.54±2.04	95.01±0.34	84.18±1.82	95.68±1.22	97.09±0.70	97.20±0.64	97.22±0.28
WalkPool	98.95±0.41	92.79±1.09	95.60±0.37	92.56±0.60	97.27±0.28	98.68±0.48	98.37±0.25	98.58±0.19
GCN	95.22 ± 1.8	87.98 ± 1.45	92.91 ± 0.3	74.68 ± 2.67	91.42 ± 0.44	93.56 ± 1.53	94.49 ± 0.61	98.48 ± 0.22
SGCN	95.17 ± 0.96	89.38 ± 1.42	93.86 ± 0.42	81.08 ± 1.2	77.51 ± 1.85	94.08 ± 1.43	95.74 ± 0.33	98.32 ± 0.2
UTGCN	94.76 ± 1.03	91.47 ± 1.4	94.49 ± 0.38	72.97 ± 1.27	61.68 ± 1.01	94.81 ± 1.1	94.0 ± 0.43	99.37 ± 0.1
SAGE	95.9 ± 0.86	87.32 ± 1.61	92.94 ± 0.57	74.17 ± 2.03	62.6 ± 3.3	93.37 ± 1.2	94.43 ± 0.67	98.22 ± 0.13
SSAGE	95.21 ± 1.09	88.05 ± 1.8	91.66 ± 0.43	81.84 ± 1.49	70.1 ± 1.3	92.25 ± 1.45	95.72 ± 0.31	93.59 ± 0.14
UTSAGE	94.72 ± 1.07	84.48 ± 1.87	86.46 ± 0.64	72.96 ± 1.26	61.47 ± 0.99	87.94 ± 1.58	93.45 ± 0.45	85.56 ± 0.37
GIN	95.24 ± 1.22	86.74 ± 2.3	93.04 ± 0.99	71.97 ± 2.3	87.84 ± 3.05	92.14 ± 0.98	94.7 ± 0.45	98.43 ± 0.24
GraphConv	95.73 ± 1.4	86.64 ± 2.31	92.99 ± 0.87	74.31 ± 1.93	80.84 ± 1.28	91.16 ± 1.76	94.94 ± 0.38	98.32 ± 0.22
SGIN	95.48 ± 0.88	88.31 ± 1.3	93.72 ± 0.48	73.73 ± 1.69	72.83 ± 1.28	93.02 ± 1.37	95.63 ± 0.49	97.68 ± 0.2
UTGIN	94.62 ± 1.05	86.48 ± 1.29	92.77 ± 0.51	72.93 ± 1.27	61.67 ± 1.02	93.44 ± 0.84	92.94 ± 0.41	95.81 ± 0.22