LINK PREDICTION WITH UNTRAINED MESSAGE PASSING LAYERS

Anonymous authors
Paper under double-blind review

ABSTRACT

In this work, we explore the use of untrained message passing layers in graph neural networks for link prediction. The untrained message passing layers we consider are derived from widely used graph neural network architectures by removing trainable parameters and nonlinearities in their respective message passing layers. Experimentally we find that untrained message passing layers can lead to competitive and even superior link prediction performance compared to fully trained message passing layers while being more efficient and naturally interpretable, especially in the presence of high-dimensional features. We also provide a theoretical analysis of untrained message passing layers in the context of link prediction and show that the inner product of features produced by untrained message passing layers relate to common neighbour and path-based topological measures which are widely used for link prediction. As such, untrained message passing layers offer a more efficient and interpretable alternative to trained message passing layers in link prediction tasks.

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 and have found applications in a variety of fields where data exhibits a complex topology that can be captured as a graph. This is shown by a multitude of studies, including Basu et al. (2022); Wang et al. (2022); Fu et al. (2022); Jaini et al. (2021); Puny et al. (2021), 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) (Gilmer et al., 2017), which operates by propagating and aggregating information between nodes in the graph, using message and update functions possibly with learnable parameters. However, designing effective GNNs can be challenging, as they can suffer from issues such as over-smoothing or over-parameterization, and because training GNNs 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.

With our work, we aim to complement existing works on simplified and untrained MP architectures, previously formulated in the context of node classification, from the perspective of link prediction (Zhang & Chen, 2018). Link prediction is an important task in graph ML with many applications such as recommender systems, spam mail detection, drug re-purposing, and many more (Zhang & Chen, 2018). Current state of the art LP methods (Wang et al., 2023) in general rely on a combination of GNNs and structural features hence formulating effective, efficient and interpretable MP architectures has the potential to further improve current LP methods in these regards. Therefore we focus our analysis on MP layers rather than trying to formulate a novel end-to-end LP method.

In this work, we explore the use of untrained message passing layers for link prediction in graph datasets with high dimensional features. By formulating Untrained Message Passing (UTMP) layers, we follow an approach similar to that of *Simplified Graph Convolutional Networks* introduced by Wu et al. (2019). 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. In addition to these we also consider fully untrained architectures based on simple inner products of features obtained after l iterations of UTMP layers as a baseline and find that these features produced by UTMP layers are already highly informative leading to surprisingly high link prediction performances.

We base our analysis on untrained versions of four widely used MP architectures, namely Graph Convolutional Networks (GCN)(Kipf & Welling, 2016a), SAGE (Hamilton et al., 2017), GraphConv (Morris et al., 2019) and GIN (Xu et al., 2018). We test these untrained message passing layers on a variety of datasets that cover a wide range of sizes, node features, and topological characteristics ensuring a comprehensive evaluation of the models. We test UTMP layers on a variety of datasets that cover a wide range of sizes, node features, and topological characteristics ensuring a comprehensive evaluation of the UTMP layers. Mirroring the results reported by Wu et al. (2019) for node classification we find that UTMP layers in many cases outperform their fully trained counterparts in LP tasks while being highly interpretable and easier to optimize.

We also show that link prediction provides a complementary perspective for the theoretical analysis of UTMP layers (Zhou et al., 2022; Chamberlain et al., 2022). In our theoretical analysis we establish a direct connection between features produced by UTMP layers and various path based node similarity measures. Path based measures capture the indirect connection strength between node pairs nodes in the absence of a direct link connecting the nodes. Consequently, path based measures and methods have been widely used in traditional link prediction methods (Martínez et al., 2016; Kumar et al., 2020) and also play a key role in many state of the art methods (Zhu et al., 2021; Zhang & Chen, 2018).

Our theoretical analysis relies 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. Hence, our theoretical findings also 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. More generally 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 including settings beyond link prediction.

The main contributions of the paper are as follows:

- We show that untrained versions of widely used MP layers often outperform their fully trained counterparts in LP tasks.
- We establish a direct connection between MPNNs and path based node similarity measures both of which are widely used in LP methods.
- Our theoretical analysis further provides insights in to the effectiveness of widely used node initialization schemes such as one-hot-encodings and random features.

2 Related Work

Our work is motivated by recent works that investigate simplified and untrained GNNs from different perspectives. We formulate UTMP layers following the approach of Wu et al. (2019) which simplifies GNNs by successively removing trainable parameters from layers and nonlinearities between consecutive layers. Wu et al. (2019) also provide a theoretical analysis of simplified models in the context of node classification reducing the model to a fixed low-pass filter followed by a linear classifier. The paper also empirically evaluates the simplified architectures on various downstream applications and shows that simplified architectures do not negatively impact accuracy while being computationally more efficient than their fully trained counterparts.

Other works have focused on finding untrained subnetworks. For instance Huang et al. (2022) explores the existence of untrained subnetworks in GNNs that can match the performance of fully trained dense networks at initialization, without any optimization of the weights. The paper leverages sparsity as the core tool to find such subnetworks and shows that they can substantially mitigate the over-smoothing problem, hence enabling deeper GNNs. The paper also shows that the sparse untrained subnetworks have appealing performance in out-of-distribution detection and robustness to input perturbations. Similarly, in Böker et al. (2023) the authors demonstrate that GNNs with randomly initialized weights, without training, can achieve competitive performance compared to their trained counterparts focusing on the problem of graph classification. In Dong et al. (2023) the authors show that certain common neighbour measures can be approximated by MPNNs initialised with random weights and node features without training. Other more recent works on untrained GNNs include Dong et al. (2024) where the authors propose a training free linear GNN model for semi supervised node classification in text attributed graphs and Sato (2024) that defines training free GNNs for transductive node classification based on using training labels as features.

Link prediction is widely studied problem with a multitude of available methods. UTMP layers are related to a both GNN based methods such as Variational Graph Autoencoders (V-GAE) (Kipf & Welling, 2016b) and more traditional methods that are rely on path and random walk based measures for link prediction (Kumar et al., 2020). On the other hand state of the art methods such as SEAL (Zhang & Chen, 2018), NBFnet (Zhu et al., 2021), BUDDY (Chamberlain et al., 2022), Neo-GNN (Yun et al., 2021) and NCNC (Wang et al., 2023) in general rely on combining GNNs and structural features. Some methods such as SEAL (Zhang & Chen, 2018) and WalkPool (Pan et al., 2021) are based on extracting and performing MP on local subgraphs around target links effectively framing link prediction as a graph classification problem. Although subgraph extraction based methods can out perform purely GNN based methods in link prediction tasks the subgraph extraction process can be resource intensive for large networks negatively affecting the scalability of these methods. NBFnet (Zhu et al., 2021) is another state-of-the-art method that is motivated by the Bellman-Ford algorithm. NBFnet is based on learning representations of paths between target nodes and aggregation functions for these representations. While NBFnet scales more favorably compared to subgraph extraction based methods it still needs to compute representations for large numbers of paths to predict links and hence has worse scaling behavior compared to purely GNN based link prediction methods (Zhu et al., 2021). Neo-GNN (Yun et al., 2021) and BUDDY (Chamberlain et al., 2022) circumvent these difficulties by using pairwise similarity measures between higher order neighbourhoods of nodes, which are then used together with GNN based node features for LP. Notably BUDDY includes a feature propagation step that can be seen as a special case of UTMP (see UTSAGE in Section 3.1). In Wang et al. (2023) the authors propose a GNN based approach that in addition to using node level features also aggregates features of common neighbours for link prediction and further propose Neural Common Neighbour Completion (NCNC) to counteract the negative effects of graph incompleteness on LP performance.

As detailed above GNNs are widely employed as sub-components in LP methods. Hence rather than proposing a new method for link prediction we consider the advantages of using UTMP layers over their trained counterparts in existing LP methods. Moreover, our theoretical results establish a link between MP based approaches and structural features by showing that features resulting from UTMP implicitly encode neighbourhood information that underlies many widely used common neighbour and path based structural features.

From a theoretical perspective, our results also relate to recent works on the effectiveness of random node initializations and one-hot encodings. For instance Sato et al. (2021) and Abboud et al. (2020) focus on the effect of random node initializations of the expressivity of GNNs in the context of graph classification while Cui et al. (2022) explores various encodings for the task including node and graph classification. Our analysis complements these works from the point of view of LP and establishes a link between features derived from random and one hot initializations and, path based topological features.

Finally, in our theoretical analysis we rely on the fact that collections of high dimensional vectors tend to be mutually orthogonal. This is a widely know fact that has wide raging applications in ML more broadly given the pervasive use of high dimensional vector representations in modern ML methods (Kanerva, 2009; 2018).

3 MESSAGE PASSING ARCHITECTURES

Prior to introducing the message passing architectures investigated in our work, we first clarify the notation used throughout the paper. Let $G(V, E)$ be an undirected graph with vertex set $V = \{v_1, v_2 \dots v_N\}$, edges $E \subseteq V \times V$ and no self-loops, i.e. $(v, v) \notin E \quad \forall v \in V$. We denote the adjacency matrix of the graph as A and define $\tilde{\mathbf{A}} := \mathbf{A} + \mathbf{I}$, where $\mathbf{I}$ is the identity matrix, i.e. $\tilde{\mathbf{A}}$ denotes the adjacency matrix of the graph G that explicitly includes all possible self-loops. We use $\mathcal{N}(v)$ to denote the neighborhood of a node v, i.e. the set $\{w \in V : (v, w) \in E\}$, and use $\tilde{\mathcal{N}}(v) := \mathcal{N}(v) \cup \{v\}$ to denote the neighborhood of v in the graph with self-loops. Similarly, we denote the degree of a node v as $d(v)$ and $\tilde{d}(v) = d(v) + 1$. The initial feature vector of node v is denoted as $h_v^{(0)}$ and we use $h_v^{(l)}$ to denote the updated feature vector of node v after l rounds of message passing. 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 untrained versions, we first introduce the message passing rules of the four GNN architectures considered in our work, using a unified notation above.

Graph Convolutional Networks (Kipf & Welling, 2016a) 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:

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

where $W^{(l)}$ is the weight matrix for layer l.

GraphSAGE (Hamilton et al., 2017) GraphSAGE is a type of Graph Neural Network that uses different types of aggregators such as mean, gcn, pool, and lstm to aggregate information from neighboring nodes. The MP layer in GraphSAGE uses the following formula:

$$h_v^{(l)} = W_1^{(l)} \cdot h_v^{(l-1)} + W_2^{(l)} \cdot \mathrm{AGG}_{u \in \mathcal{N}(v)} h_u^{(l-1)},$$

where $W_{\{1,2\}}^{(l)}$ are learned weight matrices at layer l, AGG is an aggregation function (such as mean, sum, max).

Throughout this paper use the following slightly modified 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 (Xu et al., 2018) 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 MP step pf 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 an MLP after each message passing layer, which in our implementation includes two Linear layers and a Rectified Linear Unit (ReLU) activation function following each Linear layer (code adapted from Böker et al. (2023); Morris & Ningyuan Huang (2023)).

GraphConv (Morris et al., 2019) GraphConv is a generalization of GNNs, which can take higher-order graph structures at multiple scales into account. The mathematical formulation of this is as follows:

$$h_v^{(l)} = W_1^{(l)} \cdot h_v^{(l-1)} + W_2^{(l)} \cdot \sum_{u \in \mathcal{N}(v)} h_u^{(l-1)}$$

where $W_{\{1,2\}}^{(l)}$ are learned weight matrices.

3.1 Untrained MP Architectures

For the purpose of our theoretical and experimental evaluation, we now define the untrained counterparts of the four Message Passing Neural Network (MPNN) architectures introduced in the previous section. Following Wu et al. (2019) we eliminate all learnable components and replace them with identity matrices. Here our objective is to obtain the simplest form for the update function that retains the general message passing strategy, which includes the predefined update message passing functions and aggregation methods while removing all learnable parameters and nonlinearities. We obtain the following functions that capture the aggregation and update step in the untrained versions of the message passing layers:

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: $h_v^{(l)} = (1+\epsilon) h_v^{(l-1)} + \sum_{u \in \mathcal{N}(v)} h_u^{(l-1)}$

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

In general, we consider the case where all nodes have self-loops, i.e. the features of the node itself are included in the aggregation step. Further setting $\epsilon = 0$ for GINs results in a uniform formula across both models: $h_v^{(l)} = \sum_{u \in \tilde{\mathcal{N}}(v)} h_u^{(l-1)}$. Henceforth we will refer to both models as UTGIN.

The simplified 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 and $\mathbf{S} = \tilde{\mathbf{D}}^{-1} \tilde{\mathbf{A}}$ for UTSAGE, where $\tilde{\mathbf{D}}$ is the degree matrix with diagonal entries $\tilde{\mathbf{D}}_{uu} = \sum_v \tilde{\mathbf{A}}_{uv}$. Similarly, for UTGIN we have $\mathbf{S} = \tilde{\mathbf{A}}$. The generalization of UTMP layers to undirected weighted graphs can be obtained by simply replacing the adjacency matrix and related quantities with their weighted counterparts in the formulation of $\mathbf{S}$.

3.2 Simplified architectures

Following the construction of Wu et al. (2019) for the case of node classification we add a final trained linear layer before the final dot product. We refer to such architectures that include a final trained linear layer after the UTMP layers as 'simplified' in accordance with Wu et al. (2019) and include an 'S' in the abbreviations of these models, e.g. SGCN. This results 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. In the case of simplified GNN architectures, the trained linear layer can also be interpreted as a modified positive semi-definite inner product in the form of $\langle \Theta h_v^l, \Theta h_u^l \rangle$ where Θ is the weight matrix of the linear layer.

In the case of link prediction features produced by UTMP layers can actually 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 can do surprisingly well in terms of LP performance showing that UTMP layers produce highly informative features.

3.3 UTMP layers and path based measures

Building on the formulations of the untrained layers above, in the following we provide a theoretical analysis that relates the inner products of features resulting from untrained message passing layers to pair-wise measures of node similarity that are based on characteristics of *paths* in the underlying graph. Such path based measures offer a way of quantifying the indirect connection strength between node pairs in the absence of a direct link connecting the nodes. In order to relate path based measures and UTMP layers we will assume that initial feature vectors are pairwise orthonormal i.e. $\langle h_v^{(0)}, h_u^{(0)} \rangle = \delta_{u,v}$.

A path of length l is defined as a sequence of $l+1$ vertices $(v_0, v_1 \ldots v_l)$ 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 nodes 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. $|P^l_{uv}| = \tilde{\mathbf{A}}^l_{uv}$. Note that since we assume self-loops on all vertices P^l_{uv} implicitly also includes shorter paths between u and v. Similarly, paths of length l between vertices u and v also determine the probability of a random walk starting at u reaching v which is given by $P(u \xrightarrow{l} v) = \sum_{p \in P^l_{uv}} \prod_{i \in p-[v]} \frac{1}{\tilde{d}_i}$, where $p-[v]$ denotes that the last vertex (v) is not included in the product. The random walk probability can also be expressed in matrix form $P(u \xrightarrow{l} v) = (\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}})^l_{uv}$.

Now we consider inner products of features after l iterations of message passing which is given by $\langle h^{(l)}_u, h^{(l)}_v \rangle = (\mathbf{S}^l \mathbf{H}^{(0)} \mathbf{H}^{(0)\top} (\mathbf{S}^l)^\top)_{uv}$. For orthonormal features the inner products of features reduces to $\mathbf{H}^{(0)}\mathbf{H}^{(0)\top} = \mathbf{I}$ and we obtain the following expression for the inner product of the features after l iterations of UTMP layers:

$$\langle h^{(l)}_u, h^{(l)}_v \rangle = (\mathbf{S}^l (\mathbf{S}^l)^\top)_{uv}.$$

For UTGCN we have $\mathbf{S} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$ and the inner product after l layers can be expressed in terms of paths of length $2l$ between u and v as:

$$\langle h^{(l)}_u, h^{(l)}_v \rangle = \frac{1}{\sqrt{\tilde{d}(u)\tilde{d}(v)}} \sum_{p \in P^{2l}_{uv}} \prod_{i \in [p]} \frac{1}{\tilde{d}_i},$$

where $[p]$ denotes the path p with the first and last vertices removed. The above expression 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. Similarly, for UTSAGE we have $\mathbf{S} = \tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}$ and the inner product can be expressed in terms of paths in P^{2l}_{uv} as:

$$\langle h^{(l)}_u, h^{(l)}_v \rangle = \sum_{p \in P^{2l}_{uv}} \prod_{i \in p-m(p)} \frac{1}{\tilde{d}_i},$$

where $m(p)$ is the midpoint of the path p. The above expression is equivalent to $\langle h^{(l)}_u, h^{(l)}_v \rangle = \sum_i P(u \xrightarrow{l} i) P(v \xrightarrow{l} i)$ 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 hence:

$$\langle h^{(l)}_u, h^{(l)}_v \rangle = |P^{2l}_{uv}|.$$

Although the condition of orthonormality might seem quite restrictive at first glance it applies in many practical settings, though in some cases only approximately. Moreover, for the above result to hold orthogonality only needs to be satisfied in the common l-neighbourhood of the nodes. One example of orthonormal features that are widely used in practice are one hot encodings and orthonormality also applies in the case of high dimensional random feature vectors since for sufficiently large dimensions any set of k independent random vectors is quasi orthogonal (Eaton, 2007). Similar results also hold for random high dimensional binary features that are sparsely populated for which the expected value of the inner product of two vectors scales as $O(1/k)$ for dimension k.

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, notwithstanding the fact that features of connected node pairs can be highly correlated (Nt & Maehara, 2019), as can be verified experimentally (see Sec.C).

As mentioned before in the case of simplified GNN architectures, the final trained linear layer can be interpreted as a modified positive semi-definite inner product and the orthogonality results for high dimensional random features also apply to such more general inner products. However, note that normality is no longer guaranteed i.e. $\langle \Theta h_v, \Theta h_u \rangle \sim \delta_{u,v} |h_u|^2_\Theta$.

We would like to note that the assumption of orthogonality is a mathematical assumption we use to establish the connection between UTMP layers and path based measures. However, this does not imply that UTMP require orthogonal features to perform well at LP tasks. On the contrary deviations from orthogonality can enhance the LP performance of UTMP for instance when connected nodes tend to have more similar features which holds for many LP benchmarks (e.g. see Figure 2).

3.4 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 (Rapoport, 1953; Holland & Leinhardt, 1971). Moreover, node similarity measures that build on triadic closure in social networks have been used for similarity-based link prediction algorithms (Lü & Zhou, 2011).

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 typically the case in an LP setting. 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 (Adamic & Adar, 2003), Resource Allocation (RA) (Zhou et al., 2009), the Katz index (Katz, 1953), rooted PageRank (Brin & Page, 1998) and SimRank (Jeh & Widom, 2002). 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 related 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.

4 Experiments and Results

In the following, we provide details on our experimental setup. We evaluate GNN architectures on a variety data sets that cover both attributed graphs where nodes have additional high-dimensional features (Cora small, CiteSeer small, Cora, CoraML, PubMed, CiteSeer, DBLP) and non-attributed graphs that do not contain any node features. Data sources and summary statistics of the data sets can be found in the Appendix Table 3. We use the area under the Receiver Operator Characteristic curve (ROC-AUC) for the non-attributed datasets and Hits@100 for the attributed datasets as our main performance measures.

4.1 Experimental Setup

To ensure a fair comparison among models we maintain the same overall architectures across all experiments and MP layers. For trainable message passing layers, each 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. Since the simplified architectures consist of UTMP layers followed by a trainable linear layer, the consideration of UT models which do not include the linear layer also covers all possible ablation studies.

In principle any LP method that uses GNNs as one of its sub-components can also be formulated using UTMP layers. However, in general state of the art methods consist of many sub-components resulting in more complex and computationally demanding experimental setups where the effect of switching from trained to untrained MP layers is more difficult to isolate. Although we therefore focus on graph auto encoders in our experiments due their simplicity, we also consider two versions of NCNC Wang et al. (2023): one in which uses trained GCNs for MP and another that uses UTGCN instead, which following our naming convention is denoted as SNCNC. For the simplified models we precompute 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. Further details about the experimental setup can be found in the Appendix Sec. B.

Table 1: Link Prediction accuracy for attributed networks as measured by Hits@100. 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
	Hits@100	*Hits@100*	*Hits@100*	*Hits@100*	*Hits@100*	*Hits@100*	*Hits@100*
GCN	80.5 ± 1.59	83.0 ± 1.57	79.75 ± 0.74	82.92 ± 1.44	73.64 ± 2.04	81.06 ± 1.3	69.72 ± 1.71
SGCN	84.73 ± 1.46	88.69 ± 0.57	83.93 ± 0.8	87.31 ± 1.23	69.11 ± 1.25	86.02 ± 1.11	64.81 ± 2.09
UTGCN	64.24 ± 3.4	81.07 ± 1.5	37.5 ± 1.5	58.1 ± 1.74	25.35 ± 2.04	69.39 ± 2.22	31.46 ± 1.04
SAGE	75.67 ± 1.29	80.23 ± 1.09	69.07 ± 1.05	78.33 ± 0.85	56.25 ± 0.7	77.14 ± 2.93	64.81 ± 1.66
SSAGE	80.42 ± 1.71	87.22 ± 1.19	74.16 ± 1.47	79.61 ± 1.75	42.14 ± 1.85	83.78 ± 1.61	56.49 ± 2.64
UTSAGE	57.76 ± 1.51	61.85 ± 3.23	30.51 ± 1.57	51.13 ± 1.27	6.6 ± 0.75	69.88 ± 2.15	19.04 ± 2.2
GIN	74.66 ± 1.63	71.16 ± 1.67	69.83 ± 1.07	78.61 ± 1.07	65.3 ± 1.3	74.64 ± 1.61	64.81 ± 1.66
GraphConv	74.7 ± 1.14	74.89 ± 1.59	62.37 ± 1.87	78.66 ± 1.57	62.84 ± 2.1	77.69 ± 1.32	66.59 ± 1.25
SGIN	74.54 ± 1.69	78.71 ± 2.15	73.11 ± 1.03	77.46 ± 1.77	46.21 ± 0.85	78.56 ± 1.31	66.59 ± 1.15
UTGIN	46.73 ± 2.36	61.85 ± 3.23	22.65 ± 1.13	44.79 ± 1.44	22.01 ± 1.71	58.8 ± 6.18	34.29 ± 1.02
NCNC	83.69±3.13	76.37±2.90	84.55±1.14	87.36±1.83	80.72±0.91	87.22±3.61	73.77±0.75
SNCNC	**88.72±1.20**	**93.42±0.78**	**84.69±1.39**	**89.81±0.86**	**81.26±1.59**	**89.79±1.51**	**74.23±0.117**

4.2 Experimental Results

In the following section, we discuss the results of our experiments for link prediction in graphs with node attributes (i.e. in graphs where nodes have additional features) and non-attributed graphs separately. This diverse selection of data sets allows us to thoroughly evaluate the capabilities of the models for graphs from different application scenarios, with different sizes, and different topological characteristics.

Results for attributed graphs are given in Table 1 where we find that the simplified model SGCN performs best on 5 out of 7 datasets in terms of Hits@100. Moreover, we find that simple GAE type architectures based on UTMP layers can outperform more sophisticated state-of-the-art models such as SEAL, NBFnet and Neo-GNN (See Table 9 in the Appendix). In general, our experiments show that simplified models tend to perform better than or on par with their fully trained counterparts on almost all datasets in terms of Hits@100. 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. Note that, the fully untrained (UT) models can be computed efficiently via sparse matrix multiplication.

We observe that replacing trained GCN layers with their untrained counterpart also improves the LP performance of NCNC. Indeed, some of the results reported in Wang et al. (2023) seem to be obtained using UTGCN layers.

In the case of non-attributed graphs (Table 2) 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 performs best 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. 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 the UT models and path based topological measures. We also find that using UTGCN layers in NCNC instead of trained GCN layers also improves link prediction performance on 6 out of 7 datasets. Additional results for NCNC experiments can be found in AppendixF

Table 2: 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
NCNC	92.66 ± 1.94	86.01 ± 3.13	95.27 ± 0.26	61.63 ± 2.18	73.06 ± 2.96	91.10 ± 2.14	93.41 ± 0.46	99.53 ± 0.09
SNCNC	91.28 ± 2.97	88.18 ± 2.65	95.77 ± 0.22	68.41 ± 1.46	87.29 ± 1.20	93.95 ± 1.36	95.42 ± 0.4	99.62 ± 0.05

Finally, we also examine the effect of increasing the number of UTMP layers using fully untrained (UT) models. Our results in Fig.1 indicate that, in general, UTGCN and UTSAGE maintain their 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.3.3) which leads UTGIN to be dominated by longer paths, and hence longer distance correlations, as the number of layers increases. In general however we find that UTMP layers do not suffer from over-squashing when equipped with proper degree based normalisation which can be attributed to the absence of nonlinearities and mixing between feature dimensions in UTMP layers.

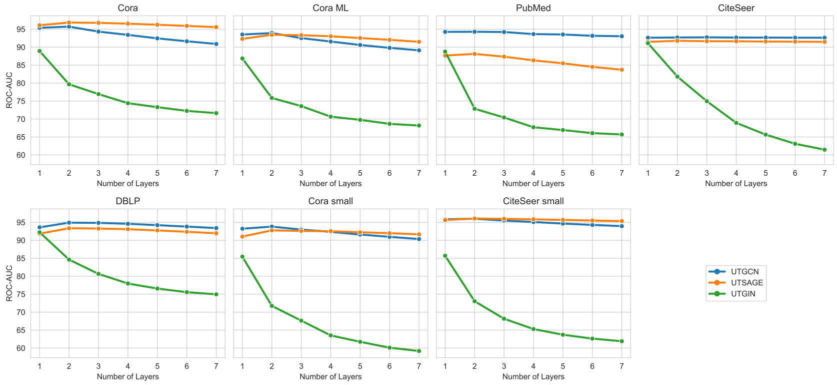

Figure 1: The effect of increased layer size for fully untrained models.

5 Conclusion

In this work, we explored the application of graph neural networks with untrained message passing layers for link prediction. Interestingly, our experimental evaluation of twelve data sets shows that simplifying GNNs architectures by eliminating trainable parameters and nonlinearities not only enhances the link prediction performance of GNNs, but also improves their interpretability and training efficiency. As such untrained message passing layers offer a computationally efficient alternative to their fully trained counterparts that naturally scales to large graphs. To complement our experimental results, we offered a theoretical perspective on untrained message passing, analytically establishing links between features generated by untrained message passing layers and path-based topological measures. We found that the link prediction offers a complementary perspective for analysing MPNNs and provides insights into the topological features captured by widely used initialization schemes such as random features and one-hot encodings.

In future work, we hope to extend our study to other classes of graphs, such as directed, signed, weighted, and temporal networks. The conceptual simplicity of untrained message passing layers might also be a useful guide in designing new graph neural network architectures or adapting existing architectures to directed or temporal networks. We thus believe that our work is of interest both for the community of researchers developing new machine learning methods, as well as for practitioners seeking to deploy efficient and resource-saving models in real-world scenarios.

Reproducibility Statement

All data sets are available from cited sources and the code necessary for replicating the experimental results is available at: `https://zenodo.org/records/11237762`

References

Ralph Abboud, Ismail Ilkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. The surprising power of graph neural networks with random node initialization. *arXiv preprint arXiv:2010.01179*, 2020.

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.

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

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

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

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

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

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.

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.

Hejie Cui, Zijie Lu, Pan Li, and Carl Yang. On positional and structural node features for graph neural networks on non-attributed graphs. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 3898–3902, 2022.

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

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.

Morris L Eaton. Random vectors. In *Multivariate Statistics*, volume 53, pp. 70–103. Institute of Mathematical Statistics, 2007.

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.

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*, pp. 1263–1272. PMLR, 2017.

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

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

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*, pp. 8–1. PMLR, 2022.

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.

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*, pp. 538–543, 2002.

Pentti Kanerva. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive computation*, 1:139–159, 2009.

Pentti Kanerva. Computing with high-dimensional vectors. *IEEE Design & Test*, 36(3):7–14, 2018.

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

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

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

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

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.

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`.

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.

Christopher Morris and Teresa Ningyuan Huang. Gin implementation. `https://github.com/nhuang37/finegrain_expressivity_GNN/blame/main/GNN_untrained/gnn_baselines/gnn_architectures.py`, 2023. Accessed: 2023-11-21.

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, pp. 4602–4609, 2019.

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

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

Liming Pan, Cheng Shi, and Ivan Dokmanić. Neural link prediction with walk pooling. *arXiv preprint arXiv:2110.04375*, 2021.

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.

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

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.

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

Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Random features strengthen graph neural networks. In *Proceedings of the 2021 SIAM international conference on data mining (SDM)*, pp. 333–341. SIAM, 2021.

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

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.

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

Xiyuan Wang, Haotong Yang, and Muhan Zhang. Neural common neighbor with completion for link prediction. *arXiv preprint arXiv:2302.00890*, 2023.

Duncan J Watts and Steven H Strogatz. Collective dynamics of ‘small-world’networks. *Nature*, 393 (6684):440–442, 1998.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pp. 6861–6871. PMLR, 2019.

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

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

Seongjun Yun, Seoyoon Kim, Junhyun Lee, Jaewoo Kang, and Hyunwoo J Kim. Neo-gnns: Neighborhood overlap-aware graph neural networks for link prediction. *Advances in Neural Information Processing Systems*, 34:13683–13694, 2021.

Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

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 32-1, 2018.

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

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.

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.

A Dataset details

Summary statistics and sources of data sets are given in Table 3.

Table 3: Overview of the datasets, sources, and node features for attributed graphs (top group) used in our experimental evaluation.

Dataset	|V|	|E|	Features
Cora small Yang et al. (2016)	2,708	10,556	1,433
CiteSeer small Yang et al. (2016)	3,327	9,104	3,703
Cora Bojchevski & Günnemann (2017)	19,793	126,842	8,710
Cora ML Bojchevski & Günnemann (2017)	2,995	16,316	2,879
PubMed Bojchevski & Günnemann (2017)	19,717	88,648	500
CiteSeer Bojchevski & Günnemann (2017)	4,230	10,674	602
DBLP Bojchevski & Günnemann (2017)	17,716	105,734	1,639
NS Newman (2006)	1,461	2,742	-
Celegans Watts & Strogatz (1998)	297	2,148	-
PB Ackland et al. (2005)	1,222	16,714	-
Power Watts & Strogatz (1998)	4,941	6,594	-
Router Spring et al. (2004)	5,022	6,258	-
USAir Batagelj & Mrvar (2006)	332	2,126	-
Yeast Von Mering et al. (2002)	2,375	11,693	-
E-coli Zhang et al. (2018)	1,805	15,660	-

B Experimental setup and Hyperparameter choices

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 4). The optimal hyperparameters values for attributed and non-attributed datasets are given in Table 5 and Table 6, respectively. We implement a three-fold cross-validation procedure to select the optimal hyperparameter values.

We use Adam Kingma & Ba (2014) as an optimization function and employ binary cross entropy with logits as our loss function. All datasets are preprocessed by normalizing the node features and randomly splitting them. For non-attributed datasets, 10% of the data is allocated to the test set, 5% to the validation set Zhang & Chen (2018), and the remaining data is used for the training set. In contrast, for attributed datasets, the split is 20% for the test set, 10% for the validation set, and the remainder for the training set Wang et al. (2023).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 pyGteam (2021), 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.

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 4: 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

Table 5: 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 6: 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

C Orthogonality of node features in empirical data sets

The distribution of inner products of initial node features for attributed datasets is given in Figure 2. 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.

D 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 Wu et al. (2019) 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 Table5, 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

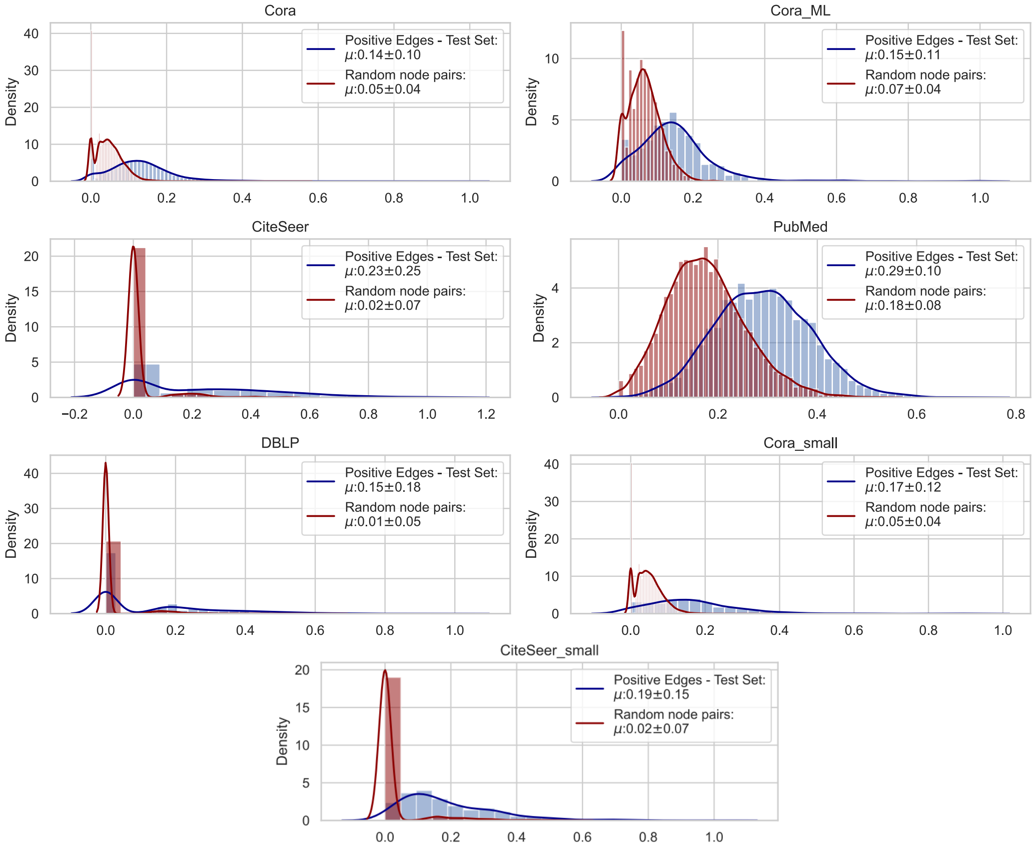

Figure 2: The distribution of feature dot products for pairs of connected and random node pairs for the attributed datasets.

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			**CiteSeer**			**Cora large**			**Cora ML**			**PubMed**			**CiteSeer large**			**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**	**CiteSeer**	**Cora large**	**Cora ML**	**PubMed**	**CiteSeer large**	**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 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.

E Comparison to state-of-the-art

In Table 9, we compare UTMP-based architectures with several state-of-the-art link prediction models, including SEAL Zhang & Chen (2018), NBFNet Zhu et al. (2021), Neo-GNN Yun et al. (2021), BUDDY Chamberlain et al. (2022), and NCNC Wang et al. (2023), on attributed datasets. Our results demonstrate that UTGCN-based NCNC, consistently outperforms other models across these datasets with BUDDY ranking second on the Cora and CiteSeer small datasets, and SEAL ranking second on PubMed.

Moreover, SGCN architecture outperform leading methods like SEAL, NBFNet, and Neo-GNN in terms of Hits@100. Notably, SGCN outperforms SEAL and Neo-GNN on two out of three datasets and performs better than NBFNet across all datasets.

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. It should also be noted that SGCN is considerably more efficient than both methods, which rely on subgraph extraction.

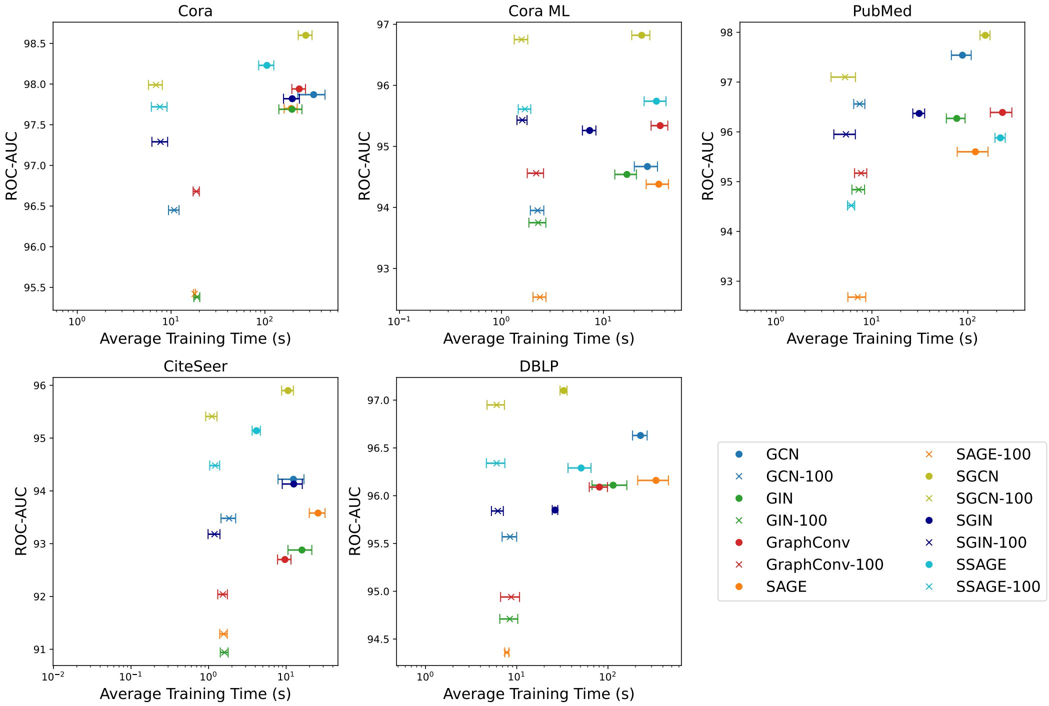

Figure 3: Average runtimes (in seconds) for training and inference for attributed data sets.

Table 9: Link Prediction accuracy for attributed networks as measured by Hits@100 compared to heuristics and state-of-the-art models SEAL, NBFnet,BUDDY, Neo-GNN, and NCNC. Hits@100 values for the heuristics and the state-of-the-art models were taken from Wang et al. (2023).

Models	**Cora small**	**CiteSeer small**	**PubMed**
	Hits@100	*Hits@100*	*Hits@100*
CN	33.92±0.46	29.79±0.90	23.13±0.15
AA	39.85±1.34	35.19±1.33	27.38±0.11
RA	41.07±0.48	33.56±0.17	27.03±0.35
BUDDY	88.00±0.44	92.93±0.27	74.10±0.78
Neo-GNN	80.42±1.31	84.67±2.16	73.93±1.19
NCNC (Wang et al.)	89.65±1.36	93.47±0.95	81.29±0.95
NCNC(GCN)	83.69±3.13	76.37±2.90	80.72±0.91
SNCNC (UTGCN)	88.72±1.20	93.42±0.78	81.26±1.59
SEAL	81.71±1.30	83.89±2.15	75.54±1.32
NBFnet	71.65±2.27	74.07±1.75	58.73±1.99
GCN	80.5 ± 1.59	83.0 ± 1.57	73.64 ± 2.04
SGCN	84.73 ± 1.46	88.69 ± 0.57	69.11 ± 1.25
SAGE	75.67 ± 1.29	80.23 ± 1.09	56.25 ± 0.7
SSAGE	80.42 ± 1.71	87.22 ± 1.19	42.14 ± 1.85
GIN	74.66 ± 1.63	71.16 ± 1.67	65.3 ± 1.3
GraphConv	74.7 ± 1.14	74.89 ± 1.59	62.84 ± 2.1
SGIN	74.54 ± 1.69	78.71 ± 2.15	46.21 ± 0.85

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 Zhang & Chen (2018) and for WalkPool from Pan et al. (2021).

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

F ADDITIONAL NCNC RESULTS

In this section, we present additional results showcasing the performance of NCNC on unattributed datasets. The experiments reveal that NCNC with untrained MP layers outperforms NCNC with trained layers on 5 out of 8 datasets.

Table 11: Comparison of NCNC with trained and untrained MP layers on unattributed datasets.The table reports Hits@100 scores for each dataset. Bold values indicate the best performance for a given dataset.

Models	**NS**	**Celegans**	**PB**	**Power**	**Router**	**USAir**	**Yeast**	**E-coli**
	Hits@100	*Hits@100*	*Hits@100*	*Hits@100*	*Hits@100*	*Hits@100*	*Hits@100*	*Hits@100*
NCNC	**91.53 ± 2.76**	92.83 ± 3.90	79.13 ± 2.41	42.14 ± 1.94	60.32 ± 4.20	95.80 ± 1.58	89.91 ± 0.85	**99.12 ± 0.20**
SNCNC	90.15 ± 3.14	**94.32 ± 3.38**	**79.89 ± 1.40**	**47.89 ± 2.53**	**75.60 ± 1.89**	**96.65 ± 1.87**	**90.62 ± 0.76**	98.98 ± 0.15