Design Space Exploration of Graph Neural Networks for Inductive Link Prediction

A Thesis

Presented to

The Division of Computer Science

McGill University

In Partial Fulfillment

of the Requirements for the Degree

Masters of Science

Jacob Danovitch

August 2023

Approved for the Division
(Computer Science)

Reihaneh Rabbany	Fernando Diaz

Acknowledgements

Thank you to my supervisors, Professor Reihaneh Rabbany and Professor Fernando Diaz, for their time and efforts in guiding this work, reviewing this thesis, and supporting my research throughout my graduate studies. This research was also supported by Mila - Quebec AI Institute and the Digital Research Alliance of Canada, each providing very generous compute resources to conduct the experiments necessary to complete this thesis.

Table of Contents

List of Tables

List of Figures

Abstract

Link prediction is a task with various important applications, including recommendation systems and data mining. Recent approaches have leveraged powerful Graph Neural Networks (GNNs) to achieve state-of-the-art performance by optimizing unique representations for each node in a graph (transductive learning) or refining existing numeric attributes of each node (inductive learning). While these GNN-based approaches are undoubtedly effective, their assumptions of the underlying data can make them difficult to apply in realistic scenarios. In domains where the distribution of data may shift, such as a social media platform constantly registering new users, transductive approaches will not have optimized representations for unseen nodes and will likely struggle. This problem is mitigated by inductive approaches which utilize and refine existing node attributes; however, many graphs are completely unattributed, preventing the application of inductive GNNs. An ideal solution for link prediction in realistic settings would remedy these challenges by automatically and inductively learning node representations end-to-end using the structure of the graph, such that the same model maintains effectiveness in the presence of new nodes, edges, and entirely different graphs, without depending on fixed node representations or the presence of node attributes. In this thesis, we construct and comprehensively evaluate a family of GNN-based models for link prediction satisfying these properties in order to identify the key factors in building efficient, generalizable link prediction models. We propose a standardized framework consisting of several "design dimensions" based on state-of-the-art graph representation learning methods, which we evaluate on the challenging cross-graph inductive link prediction task. We complete an in-depth investigation of the generalization abilities of models within our framework, both at the population level as well as broken down by design dimension, and examine global trends such as the presence of a reduced subspace of optimal models and the effects of the underlying data on generalization ability. By providing insight into what factors lead to well-performing and generalizable models, the results of our studies serve as a starting point to accelerate future work in the still-young area of inductive link prediction.

Abrégé

La prédiction de liens est une tâche avec diverses applications importantes, notamment les systèmes de recommandation et l'exploration de données. Les approches récentes ont exploité des puissants réseaux neuronaux graphiques (GNN) pour atteindre des performances de pointe en optimisant des représentations uniques pour chaque nœud d'un graphe (apprentissage transductif) ou en affinant les attributs numériques existants de chaque nœud (apprentissage inductif). Bien que ces approches basées sur les GNN soient indéniablement efficaces, leurs hypothèses sur les données sous-jacentes peuvent les rendre difficiles à appliquer dans des scénarios réalistes. Dans les domaines où la distribution des données peut varier, comme une plateforme de médias sociaux enregistrant constamment de nouveaux utilisateurs, les approches transductives n'auront pas optimisé les représentations pour les nœuds invisibles et auront probablement du mal. Ce problème est atténué par les approches inductives qui utilisent et affinent les attributs de nœuds existants ; cependant, de nombreux graphes sont entièrement non attribués, ce qui empêche l'application de GNN inductifs. Une solution idéale pour la prédiction de liens dans des contextes réalistes remédierait à ces défis en apprenant automatiquement et inductivement des représentations de nœuds de bout en bout en utilisant la structure du graphe, de sorte que le même modèle maintienne son efficacité en présence de nouveaux nœuds, de nouvelles arêtes et de graphes entièrement différents, sans dépendre de représentations de nœuds fixes ou de la présence d'attributs de nœuds. Dans cette thèse, nous construisons et évaluons de manière exhaustive une famille de modèles basés sur les GNN pour la prédiction de liens satisfaisant à ces propriétés afin d'identifier les facteurs clés dans la construction de modèles de prédiction de liens efficaces et généralisables. Nous proposons un cadre normalisé composé de plusieurs « dimensions de conception » basées sur des méthodes d'apprentissage de représentations de graphe de pointe, que nous évaluons sur la tâche difficile de prédiction de liens inductifs entre graphes. Nous menons une enquête approfondie sur les capacités de généralisation des modèles dans notre cadre, à la fois au niveau de la population et en fonction de la dimension de conception, et examinons les tendances globales telles que la présence d'un sous-espace réduit de modèles optimaux et les effets des données sous-jacentes sur la capacité de généralisation. En fournissant des informations sur les facteurs qui conduisent à des modèles performants et généralisables, les résultats de nos études servent de point de départ pour accélérer les travaux futurs dans le domaine encore jeune de la prédiction de liens inductifs.

Dedication

This thesis is dedicated to my parents Andrea and Mark and my grandmother Shiela, who have always gone above and beyond in supporting me through whatever life has thrown my way, and to my partner Emma, who is by my side no matter what through every difficult day and late night. Without you all, this would not have been possible.

Chapter 1

Introduction

1.1 Motivation

Complex networks are ubiquitous to daily life. From the technology powering our everyday communications and interactions to the structures that form our DNA, networks are a vital component of our everyday life. Their wide-ranging relevance and importance has spurred a large body of research focused on understanding their sophisticated dynamics. For example, modeling travel patterns as networks can help forecast traffic levels (Cui, Henrickson, Ke, & Wang, 2018; Jiang & Luo, 2021) to more accurately estimate arrival times (Derrow-Pinion et al., 2021; Fang et al., 2020; D. Wang, Zhang, Cao, Li, & Zheng, 2018) and optimize the availability of suitable alternatives (Jiang, 2022; Yao et al., 2018), or even help prevent or mitigate the spread of infectious diseases in epidemics and pandemics (Colizza, Barrat, Barthelemy, & Vespignani, 2005; Kamp, Moslonka-Lefebvre, & Alizon, 2013; So, Chu, Tiwari, & Chan, 2020). Similarly, understanding the interactions occurring in biological structures can help advance modern medicine by facilitating the discovery of novel drugs (Abbas et al., 2021; MacLean, 2021). Networks can also be used to represent our interests and preferences. Many recommendation scenarios can be framed as bipartite graphs between users and an item or product, with network-theoretic algorithms providing data-driven recommenations to enhance end-user experience by suggesting new and useful items (Daud, Ab Hamid, Saadoon, Sahran, & Anuar, 2020; Huang, Li, & Chen, 2005; Talasu, Jonnalagadda, Pillai, & Rahul, 2017).

Many of these applications are founded on the fundamental task of *link prediction* (Adamic & Adar, 2003; Liben-Nowell & Kleinberg, 2007; Popescul & Ungar, 2003; Taskar, Wong, Abbeel, & Koller, 2003), which concerns the identification of missing or future edges in complex networks. The importance of link prediction has led to a significant body of research towards developing effective and generally applicable techniques, from heuristic methods Zhou, Lü, & Zhang (2009) to learning-based approaches (Backstrom & Leskovec, 2010; Hasan, Chaoji, Salem, & Zaki, 2006; Leskovec, Huttenlocher, & Kleinberg, 2010), especially those based on deep learning (Islam, Aridhi, & Smaïl-Tabbone, 2020; Mutlu, Oghaz, Rajabi, & Garibay, 2020). Deep learning methods for link prediction have shown to be quite powerful, automatically learning representations for nodes, edges, and entire graphs without the need for complex feature engineering. An important caveat to this approach is that they are often *transductive*, meaning that they learn representations specific to each node in a graph (Grover & Leskovec, 2016; Perozzi, Al-Rfou, & Skiena, 2014; Tang, Qu, Wang, et al., 2015). This is a powerful approach as it allows the representations to be optimized end-to-end, but comes with drawbacks that limit their applicability in realistic settings where the full scope of the graph may not be visible during training. This is often the case in domains such as social networks and recommender systems, in which new users register for a service, or in temporal networks where new nodes appear over time. In these scenarios, a transductive model will be unable to generalize to the unseen nodes as their representations were not optimized during training. While there are solutions (Ma, Guo, Ren, Tang, & Yin, 2020; Perozzi et al., 2014), they are often complex or computationally infeasible to apply in practice, as they require additional training before performing inference. In contrast, *inductive* approaches (Hamilton, Ying, & Leskovec, 2017; Velickovic et al., 2018) use existing numeric properties (or attributes) associated with each node instead of associating a learnable, unique representation; for example, representing a node (an academic paper) in a citation network as a binary vector $\mathbf{x} \in \mathbb{Z}^{|V|}$ where $\mathbf{x}_j$ equals 1 if the paper contains the word V_j from vocabulary V, otherwise 0. This approach is much simpler, but such properties are not always available in practice, limiting their applicability as well.

An ideal solution for link prediction in realistic settings would remedy these challenges by automatically and inductively learning node representations end-to-end using the structure of the graph, such that **the same model maintains effectiveness in the presence of new nodes, edges, and even across entirely different graphs**. This ideal model would open the door to new advancements in many real-world applications of link prediction by providing a general, flexible approach capable of handling unseen data without depending on the presence of node attributes. As such, this thesis will focus on the questions and considerations surrounding the construction of a fully inductive, learning-based approach to link prediction, which we detail in the following section.

1.2 Research Objectives

The overarching goal of this thesis is to quantify the ability of inductive link prediction (ILP) models to generalize to unseen data. The objectives of this work are formalized below. In particular, we will evaluate the ability of ILP models to generalize to an *entirely new graph*, the most challenging inductive learning scenario in which all test-time data is completely novel. In this section, we lay out our hypotheses concerning different aspects of the proposed task, as well as the proposed methodology and measure(s) to sufficiently answer the question.

1.2.1 Population-level Objectives

Our first two objectives examine trends across the full population of ILP models to establish their ability to generalize to unseen data, and if their ability is better or worse in different scenarios.

Hypothesis 1.1 (How well do ILP models generalize to unseen data?)**.** The primary objective of this thesis is to assess the ability of inductive link prediction models to generalize to unseen data. We will quantify this by computing performance metrics (such as AUC, accuracy, or F1-score) for a given model on both seen and unseen data, and measuring the *relative* generalization ability as shown by a percentage increase or decrease in performance from seen to unseen data. We will consider these measures at an aggregate level to determine the overall generalization ability of the model architectures within our proposed framework. We hypothesize that in the aggregate, **ILP models will exhibit a decrease in performance when applied to unseen data**.

Hypothesis 1.2 (Is it more difficult to generalize across domains?)**.** While our first objective will indicate whether or not ILP models are able to maintain their performance in the face of unseen data, it is important to understand if this trend is consistent in different scenarios. In particular, it is important to consider the *underlying data* when analyzing generalization; it could be the case that some types of data contain useful characteristics present in many different datasets, making them ideal for learning from, whereas others may be outliers from which it is difficult to extrapolate knowledge. We will measure this by evaluating relative generalization in two scenarios: intra- and inter-domain transfer. Intra-domain transfer will compare generalization between two different graphs in the same domain (e.g., between citation networks), while inter-domain transfer will compare generalization between graphs from different domains (e.g., from a citation network to a social network). We expect that on average, **models will generalize better between two graphs from the same domain than between two graphs from different domains**.

1.2.2 Model-level Objectives

After developing our understanding of how ILP models behave at the population level, our final two objectives shift focus to their behavior at the model level; namely, exploring the design of an optimal ILP model which can maintain strong performance on unseen data.

Hypothesis 1.3 (What components of an ILP model affect its ability to generalize to unseen data?)**.** We begin by making the assertion that certain components within our proposed framework will have a significant effect on the ability of a given model to generalize. This will be measured by evaluating each candidate method for each component of our proposed framework in terms of their generalization ability. Our prior belief is that **there exists a significant effect on generalization between different components of our framework**. This would indicate that our choice of method for some model components is more important to performance than others, which we will further explore by comparing multiple options for each component to identify a set of optimal selections.

Hypothesis 1.4 (Is there a subspace of ILP model architectures that generalize better to unseen data?)**.** If a significant effect on performance does exist between different model components, then optimally selecting these components should elicit well-performing models. In particular, these selections should yield a reduced *subspace* of model architectures within our framework which demonstrate better performance on average than models sampled from the full set of architectures. Such a space would be highly useful for future work in the area by providing a compact set of model architectures significantly smaller than the full set, greatly reducing the time and resources required to obtain a strong model. We will establish the existence of this reduced subspace by using the results of RQ3 to obtain a set of optimal choices for each component of our model, forming a candidate subspace. We will then sample a number of models from our proposed subspace as well as the full model space to determine if the performance of models from the proposed subspace is superior to those from the full space. Our hypothesis is that **our reduced space of optimal architectures will demonstrate superior generalizability versus the full space**.

Table 1.1: An overview of the hypotheses explored in this work.

Objective	Hypothesis	Measures
RQ1	ILP models will exhibit a decrease in performance when applied to unseen data.	Generalization (Δ_m); Mean % change in performance metric m between source & target graphs
RQ2	ILP models will show better generalization between graphs from the same domain than different domains.	Comparison of Δ_m when source/target graphs are in same or different domains
RQ3	Certain components of ILP models will have a significant effect on model generalization.	Comparison of Δ_m by model component
RQ4	There exists a reduced subspace of optimal ILP models which are more generalizable on average.	Comparison of Δ_m between models sampled from reduced and full spaces

The contents of the thesis will focus on comprehensively answering the above questions. We will first lay the foundations of our work by defining important concepts and notations in Section 2.1, followed by a brief background of the applications of machine learning to graphs in Section 2.2. Sections 2.3 and 2.4 will review the existing body of work concerning deep learning methods for graph representation learning, particularly for link prediction. With the groundwork solidified, Section 3 will detail the experimental methodology used to answer the listed research questions. We begin by proposing a standardized, modular framework for inductive link prediction based on modern graph representation learning (GRL) techniques which is both effective and efficient, detailing its various components and state-of-the-art candidate choices for each, before describing the cross-graph ILP task which we use to evaluate each model in the most difficult setting, and outlining the experiments we will conduct to validate each of our hypotheses. The results of our experiments are discussed in Section 4, wherein we determine the validity of our hypotheses and discuss their implications. We first examine the behavior of ILP models within our proposed framework at a population level, establishing their generalization ability and considering the effects of the underlying data, and then utilize the Randomized Controlled Search method (You, Ying, & Leskovec, 2020) to efficiently analyze the various model instances within our framework and identify the optimal choices for each design dimension, before finally examining whether our proposed set of optimal architectures contains a higher concentration of well-performing, generalizeable models in comparison to the full "design space" (You et al., 2020) of model architectures. We conclude with a review of our contributions which can be summarized as follows:

1. We conduct a novel study on the generalization abilities of numerous model architectures for the inductive link prediction task. To the best of our knowledge, no previous work has performed a comprehensive evaluation of ILP models, particularly in the challenging cross-graph setting.
2. We propose a standardized framework for inductive link prediction based on positional and structural encodings, message passing networks, and subgraph pooling. The proposed framework is *efficient*, *effective*, and *generalizable*, making it usable for real-world applications. This framework encapsulates the many model architectures evaluated as part of our study, allowing us to zero-in on the most important facets of the framework and the best choices to maximize performance in different scenarios.
3. We examine the *population-level* behavior of ILP models. We demonstrate the difficulty of the cross-graph ILP task by establishing a decrease in performance on unseen data, and then we analyze the effect of the *underlying data* on generalization ability to determine if generalizing between graphs from the same domain is lesser or greater than doing so between graphs of different domains.
4. We examine the *model-level* behavior of ILP models. Not only do we evaluate the effectiveness of many architectures, we go further by analyzing specific model components in order to identify key design choices to construct effective ILP models. We also facet our analysis by metric, demonstrating how the best model depends on the task and specific objective at hand. We also explore the existence of an optimal subspace of model architectures that perform well on the cross-graph ILP task.

Chapter 2

Background

This section of the thesis will provide a brief overview on graphs and their applications to machine learning.

2.1 Preliminaries

This section will define several foundational concepts in network science and graph machine learning. A simple graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ is a data structure which defines the connections - or *edges*, $(u, v) \in \mathcal{E}$ - between a set of nodes $\mathcal{V}$. A graph is often represented by its adjacency matrix $A \in \mathbb{R}^{n \times n}$, where $A_{i,j}$ is the weight of the connection between nodes i and j. In *weighted* graphs, we have $A_{i,j} \in \mathbb{R}$, whereas in unweighted graphs $A_{i,j} \in \{0, 1\}$ to indicate the binary presence or absence of a connection. There are many variations and extensions to the simple graph structure. For instance, a *heterogeneous* graph - also frequently referred to as *multi-relational* or *typed* - is a graph where each node and edge can be assigned a particular type label. Closely related are k-partite networks, which are heterogeneous networks in which nodes of the same type are never connected, as well as the *knowledge graph*, wherein labelled edges are expressed as $(\text{head}, \text{relation}, \text{tail})$ triples. A more complex graph structure is the *multi-layer network* $\mathcal{G} = \{V, \{\mathcal{E}_0, \mathcal{E}_1, \ldots, \mathcal{E}_n\}\}$, in which one set of nodes is connected by multiple independent sets of edges. Heterogeneous and knowledge graphs can both be expressed as multi-layer networks by storing the edges for each relation as independent layers. A multi-layer network can be represented by an adjacency *tensor* $\mathbf{A} = [A_0; A_1; \ldots; A_n] \in \mathbb{R}^{n \times n \times n}$. Finally, there are graphs in which an arbitrary number of nodes can be connected in a given edge. This is known as a hypergraph, and is said to be k-uniform if all edges have cardinality k. A simple graph is then equivalent to a 2-uniform hypergraph.

Table 2.1: Definitions of key terms.

Term	Notation	Definition	Example(s)	Additional Names
Node	$v \in \mathcal{V}$	A discrete object or entity.		Vertex, entity
Edge	$(u, v) \in \mathcal{E}$	A connection between two nodes.		
Graph	$\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$	A set or tuple of nodes and edges.	Social network, academic citations, internet hyperlinks	Network
Multi-layer Network	$\mathcal{G} = \{\mathcal{V}, \{\mathcal{E}_0, \ldots, \mathcal{E}_n\}\}$	A set of n edge sets connecting a node set.	Temporal networks (e.g. monthly snapshots), protein interaction networks	Multiplex network
Adjacency Matrix	$A \in \mathbb{R}^{\lvert V \rvert \times \lvert V \rvert}$	A matrix representation of a graph, where $A_{i,j} > 0$ if $(i, j) \in \mathcal{E}$.		

Term	Notation	Definition	Example(s)	Additional Names
Adjacency Tensor	$\mathbf{A} \in \mathbb{R}^{\lvert V\rvert \times \lvert V\rvert \times \lvert V\rvert}$	A tensor representation of a multi-layer network, where each axis is the adjacency matrix for a given layer.		
Neighbors	$u_i \in \mathcal{N}(v)$	All nodes which are connected to node v.		
Random Walk	$[w_0, w_1, \ldots, w_\ell]$	The process of randomly sampling nodes starting from $w_0 \in \mathcal{V}$, such that $w_i \in \mathcal{N}(w_{i-1})$.		
Heterogeneous Graph	*Same as multi-layer network.*	A multi-layer network where each layer represents a different edge label. May optionally assign each node a label as well.	Academic network	Multi-relational network, labelled graph

Term	Notation	Definition	Example(s)	Additional Names
Knowledge Graph	$\{(h, r, t)\} \in (\mathcal{E} = \mathcal{V} \times \mathcal{R} \times \mathcal{V})$	A heterogeneous graph, generally having a large $n >> 1000$ number of edge labels.	Wikipedia	

2.2 Applications of Machine Learning on Graphs

There are a number of tasks pertaining to graphs which machine learning models can be applied to. At their cores, each task on a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ fundamentally concerns a subset of its nodes - a *node set* - $s = \{v_i\} \in \mathcal{V}^k$. In general, we want to find a function (i.e., machine learning model) $f : \mathcal{V}^k \rightarrow \mathbb{R}^d$ such that $\hat{y}_i = f(s_i)$ is corresponds to a *ground-truth label* y_i, which is done by minimizing some cost function $\mathcal{L}$ aligned to the goal of the task.

Graph learning tasks are often **classification** based, with the goal of mapping s to one of several class labels $c_i \in \mathcal{C}$. Models are provided binary vectors $y \in \{0, 1\}^{|\mathcal{C}|}$ as labels and produce probability scores $\hat{y} \in \mathbb{R}^{|\mathcal{C}|}$ as output. The objective of single-label classification tasks is generally to find a model whose outputs $\hat{y}$ minimize the cross entropy cost function $\mathcal{L} = \sum_i^{|\mathcal{C}|} y_i \log(\hat{y}_i)$, or in the case $|\mathcal{C}| = 1$, the binary cross entropy:

$$\mathcal{L} = -(y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})) \tag{2.1}$$

If a node set can have more than one label, known as multilabel classification, the sum of binary cross entropy losses across all labels can be used. Alternatively, a graph learning task can be **regression** based, where we wish to map a node set to a single scalar value $s \rightarrow \mathbb{R}$. Mean squared error is commonly used as a cost function for regression tasks, with $\mathcal{L} = (y - \hat{y})^2$.

In all tasks, we optimize a machine learning model on a training dataset, and evaluate its performance on an inference dataset to ensure that the model is capable of generalizing to unseen data. This necessitates the partitioning of a given graph dataset into training and inference sets, which requires special consideration for graph learning tasks. In the scenario where a node may appear in both the training and inference sets, the task is said to be **transductive** in nature. In contrast, an **inductive** task is one in which the nodes that appear in the training set are *disjoint* from those which appear in the inference set. A more extreme inductive scenario could involve completely separate graphs, where the model is trained on one or more graphs and then performs inference on fully novel graphs. Broadly speaking, transductive tasks measure a model's ability to perform a task across an observed set of nodes, while inductive tasks measure a model's ability to perform a task across an unseen set of nodes.

This section will provide a brief, non-comprehensive overview of several node-level ($k = 1$), set-level ($1 < k < |\mathcal{V}|$), and graph-level ($k = |\mathcal{V}|$, i.e. $s = \mathcal{V}$) tasks alongside their respective categorizations (classification vs. regression, transductive vs. inductive, etc.) and evaluation metrics.

2.2.1 Node-level Tasks

A node-level task considers a single node in a graph at a time; for example, when we may wish to categorize a user in a social network, or to estimate the volume of traffic a web page will receive. The former case refers to **node classification**, wherein we try to find a function $f : \mathcal{V} \to \mathcal{C}$ that maps a single given node to one of several class labels. We measure the quality of f using metrics such as accuracy and F_1 score. The latter task is an example of **node regression**, where f must map a given node to a real-valued scalar and is evaluated by mean average error. In both tasks, the dataset is partitioned node-wise into training and inference nodes $V_{\text{train}}, V_{\text{test}} \subset \mathcal{V}$. In a transductive node-level task, the graph's full set of edges are available during both training and inference along with the nodes' features $X \in \mathbb{R}^{|\mathcal{V}| \times D}$ (should they exist). The only information withheld is the label of each inference node in V_{test}. In the inductive setting, however, these edges and features are also withheld, meaning that the model will be evaluated on nodes it has never seen before.

2.2.2 Set-level Tasks

Several tasks involve making predictions on sets of nodes with cardinality in $[2, |V|-1]$, most notably including **link prediction**. In link prediction, a model must identify edges that are missing from the graph. In the classical setting, edges consist of two nodes, and so a model must accept an edge, a set $s \in \mathcal{V}^2$, and perform binary classification to determine whether the edge belongs in the graph. Hyperedge link prediction generalizes this to higher values of k in hypergraphs. Binary link prediction can be evaluated using any classification-oriented metrics such as accuracy, F_1, precision, and recall. While most frequently treated as a binary classification task, link prediction can also be a regression task when a graph is weighted. In this case, a model attempts to map an edge to the correct real-valued weight. Link prediction can also be trained and evaluated in the *ranking* setting, where instead of classifying edges as valid or invalid, models must return a relative ordering over a set of edges such that the edges which do exist are ranked higher than those which don't. In this settings, models are trained with a ranking cost function such as *margin loss*, $\mathcal{L} = \max(0, m + \hat{y}_{+} - \hat{y}_{\neg})$, where $\hat{y}_{+}$ is a model's prediction for a positive (valid) edge and $\hat{y}_{\neg}$ is a model's prediction for a negative (invalid) edge. Intuitively, this cost function enforces that predictions for positive edges should be higher than for negative edges. Models trained for ranking-based link predictions can be evaluated using ranking metrics such as mean reciprocal rank (MRR) and mean average precision (MAP), and in the case of a weighted graph, graded relevance measures such as normalized discounted cumulative gain (NDCG).

For transductive link prediction, a random sample of the edges are withheld to form e_{test}, and the remaining edges as well as any node features are available during training and inference. The inductive case is similar to node classification in that a subset of nodes, along with their edges and features, are once again held out during training. At inference, however, not all of the edges associated with V_{test} are made available as they are during classification; only a subset of these nodes' edges are provided, and the task of the model is to use these along with any given node features to identify the remaining held-out edges.

A closely related task specific to knowledge graphs is **question answering**, where a model accepts a query graph $Q = \{s, \tilde{E}\} : s \in \mathcal{V}^k, \tilde{E} \subset \mathcal{E}$ representing a logical question, which is the induced subgraph of $\mathcal{G}$ over s. The goal is to find the entity which satisfies the query graph, and the task can be evaluated using link prediction metrics. There are additional set-level tasks such as **community detection**. These tasks are out of the scope of this thesis, however, and are therefore left for further reading.

2.2.3 Graph-level Tasks

Finally, a graph-level task introduces the scenario when one instance is constituted by an entire graph. These tasks are similar to node-level tasks, primarily taking the simple form of either classification or regression, with the main difference being that graph-level tasks are all inherently inductive to some extent, as each instance presents a novel graph (though nodes can of course appear in multiple graphs). **Graph classification** is the task of mapping a graph to a class label $\mathcal{G} \to \mathcal{C}$, with real-world applications such as identifying a category for a group of connected documents on Wikipedia or detecting if a discussion thread on a social media platform contains any instances of hate speech. It follows that **graph regression** is the task of mapping a graph to a real-valued scalar $\mathcal{G} \to \mathbb{R}$, such as predicting the toxicity level of a chemical compound.

2.3 Graph Representation Learning

Representation learning (Bengio, Courville, & Vincent, 2013) is the task of computing d-dimensional real-valued feature vectors (or *embeddings*) $x \in \mathbb{R}^d$ for each point in a given dataset. These feature vectors are then used as inputs to a downstream machine learning model to solve a classification or regression task, or for visualization and analysis of a dataset. In the context of graphs, this often means learning feature vectors for each node. The rise of neural-network based approaches to representation learning (Krizhevsky, Sutskever, & Hinton, 2012; Mikolov, Chen, Corrado, & Dean, 2013; Mikolov, Sutskever, Chen, Corrado, & Dean, 2013) made it possible to obtain high-quality feature vectors for large language and image datasets. This section will discuss how those approaches were extended to learn representations for graph-based data.

2.3.1 Random Walk and Factorization-based Methods

A common approach to learning word representations is by sampling sentences from a large corpus and optimizing the representation of each word to be closest to that of the words it most commonly co-occurs with. Perozzi et al. (2014) connected this process to performing random walks on graphs, resulting in the DeepWalk node embedding model. For a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ DeepWalk learns a node embedding function $\Phi : \mathcal{V} \to \mathbb{R}^d$ by drawing a corpus $\mathcal{C} = \{w_i\}$ of random walks, where $w_i = [v_0, v_1, \dots, v_n : v_i \in \mathcal{V}]$. The parameters of Φ are optimized to maximize the co-occurrence probability of nodes which appear in close proximity in the same random walk(s) using the SkimGram method (Mikolov, Chen, et al., 2013). Each walk is split into $2w + 1$-length windows, and we minimize the negative log likelihood of the first and last w nodes given the embedding of the middle node:

$$\min_{\Phi} : \; -\log \Pr\left(\{v_{i-w}, \ldots, v_{i-1}, v_{i+1}, v_{i+w}\} | \Phi(v_i)\right)$$

A number of related approaches quickly followed, most notably Node2Vec (Grover & Leskovec, 2016), which generalized the random walk extraction approach proposed in DeepWalk to allow configuration of depth versus breadth and restart probability in each walk. A concurrent line of work examined factorization of the adjacency matrix and its powers as an avenue to obtain node embeddings. LINE (Tang, Qu, Wang, et al., 2015) is a highly efficient factorization-based approach which preserves both first and second-order proximity in its resulting embedding space by decomposing. This was extended to the heterogeneous setting by PTE (Tang, Qu, & Mei, 2015), which jointly optimizes the embeddings of word-word, word-document, and word-label bipartite networks to produce content-aware node embeddings. In fact, factorization-based approaches are particularly well suited to highly multi-relational networks and schema-rich knowledge graphs, whose adjacency tensors can be decomposed to obtain knowledge graph embeddings (KGE) (Balazevic, Allen, & Hospedales, 2019; Bordes, Usunier, García-Durán, Weston, & Yakhnenko, 2013; Kazemi & Poole, 2018; Nickel, Tresp, & Kriegel, 2011; Trouillon et al., 2017).

Interestingly, connections have been identified between both lines of work. NetMF (Qiu et al., 2018; Xie et al., 2021) established equivalencies between DeepWalk, LINE, PTE, and node2vec under the unified umbrella of matrix factorization. In addition, the recently proposed ReFactor-GNN (Chen et al., 2022) demonstrated how factorization-based methods could be recast as message-passing architectures, the subject of the following section.

2.3.2 Graph Neural Networks

Neural Message Passing

The above random walk and matrix factorization-based approaches integrate multi-layer neural networks with graph-theoretic objective functions in order to produce node embeddings. While these methods are able to preserve certain properties of the underlying graph, they lack the strong inductive bias found in neural network architectures specialized for other domains, such as Convolutional Neural Networks for image processing. In contrast, Graph Neural Networks (GNNs) are generally based on the *message passing* framework (Gilmer, Schoenholz, Riley, Vinyals, & Dahl, 2017), in which nodes' representations are computed as a function of their neighbors' representations. As explained by (Bronstein, Bruna, Cohen, & Veličković, 2021; Veličković, 2022), for a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with node representations $\mathbf{X} \in \mathbb{R}^{\mathcal{V} \times d}$, the message passing procedure in GNNs can be summarized as a sequence of three steps:

- **Message**: A function $\psi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^k$ which computes a message between two nodes as a function of their representations;
- **Aggregate**: A permutation invariant function $\bigoplus : \mathbb{R}^{s \times k} \to \mathbb{R}^l$ which aggregates a set of vector-valued messages;
- **Update**: A function $\phi : \mathbb{R}^k \times \mathbb{R}^l \to \mathbb{R}^m$ which updates a node's representation as a function of its current representation and the aggregated messages of its neighbors.

Several rounds of message passing can be performed consecutively to propagate node representations across multiple *hops*, increasing a GNN's receptive field. In this case, we compute the embedding $\mathbf{h}_v$ of a node v at the i^{th} round of message passing as:

$$\mathbf{h}_v^{(i)} = \phi\left(\mathbf{h}_v^{(i-1)}, \bigoplus_{u \in \mathcal{N}(v)} \psi(\mathbf{h}_v^{(i-1)}, \mathbf{h}_u^{(i-1)})\right)$$

The resulting node representations are then used to optimize an objective function for a downstream task as described above. In node classification, for example, a linear map $\rho : \mathbb{R}^m \to \mathbb{R}^c$ (where c is the number of class labels) is applied to each node's embedding $\mathbf{h}_v$ to predict an output label.

Convolutional architectures

There are a number of instantiations of the message passing framework described above. An instance of a message passing model is considered *convolutional* if its message passing function satisfies:

$$\psi(\mathbf{h}_v, \mathbf{h}_u) = c_{v,u}\Psi(\mathbf{h}_v)$$

Where $c_{v,u}$ is a fixed constant specific to the edge between nodes v and u (e.g., an edge weight). Kipf & Welling (2017) introduced the most notable instance of a convolutional message passing model, the Graph Convolutional Network (GCN). The GCN used $\psi(\mathbf{h}_v, \mathbf{h}_u) = \frac{1}{|\mathcal{N}(v)|}\mathbf{h}_u$ and $\bigoplus = \Sigma$; intuitively, this corresponds to taking the mean of neighboring nodes' representations. A learnable weight matrix $\mathbf{W}$ was used to parameterize ϕ. This approach showed strong performance on semi-supervised node classification as well as link prediction (Kipf & Welling, 2016). Subsequent efforts improved the efficiency of the GCN model by simplifying the model architecture to consist almost exclusively of message passing, eliminating non-linearities and collapsing model parameters (X. He et al., 2020; Wu et al., 2019). Additional follow-up work (Schlichtkrull et al., 2018) showed that the message-passing paradigm extends naturally to the multi-relational setting by applying a GCN to each relation type. Of course, this becomes quite inefficient with a large number of relations, making it necessary in most cases to apply basis decomposition in order to effectively share parameters between different relations. To leverage existing work on knowledge graph embedding and obtain embeddings for relations in addition to nodes, CompGCN (Vashishth, Sanyal, Nitin, & Talukdar, 2020) applied KGE scoring functions (Bordes et al., 2013; Nickel, Rosasco, & Poggio, 2016; B. Yang, Yih, He, Gao, & Deng, 2015) to improve the quality of the output embeddings. This may, however, be unnecessary; Degraeve, Vandewiele, Ongenae, & Hoecke (2022) show that the inductive bias provided by message passing may be enough on its own, even when using completely random embeddings and untrained model weights.

Attentional architectures

Another class of message passing models are *attentional* models, which occur when:

$$\psi(\mathbf{h}_v, \mathbf{h}_u) = \alpha(\mathbf{h}_v, \mathbf{h}_u)\Psi(\mathbf{h}_v)$$

Where $\alpha : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a learnable function which maps a pair of node representations to a scalar weight, also known as an attention mechanism (Bahdanau, Cho, & Bengio, 2015; Vaswani et al., 2017). The output of the attention mechanism is a real-valued scalar signifying the importance of node u to node v, and subsequently the weight of node u's representation in computing node v's representation. When using the sum operator as the aggregation function, this corresponds to taking a weighted average over neighboring nodes' representations. This is the approach taken by the Graph Attention Network (GAT) (Velickovic et al., 2018), which uses learnable parameters $\mathbf{W} \in \mathbb{R}^{k \times d}, \mathbf{a} \in \mathbb{R}^{2k}$ to compute importance weights as:

$$\alpha(\mathbf{h}_v, \mathbf{h}_u) = \text{softmax}(\mathbf{a}^T[\mathbf{W}\mathbf{h}_v; \mathbf{W}\mathbf{h}_u])$$

2.3.3 Expressiveness of Graph Neural Networks

Despite their empirically strong results, message passing architectures are inherently limited by their connections to the Weisfeiler-Lehman (WL) algorithm (Leman & Weisfeiler, 1968). The WL algorithm assigns a color to each node in a graph, which can be used to test for an isomorphism between two graphs by deriving a mapping between the nodes in both graphs based on matching colors (known as the WL test). For a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, each node $v \in \mathcal{V}$ is assigned an initial color $c_v^{(0)} \in \mathcal{C}$. This color is iteratively updated over k steps (or until convergence) as:

$$c_v^{(i)} = \text{hash}\left((L_v^{(i-1)}, c_v^{(i-1)})\right)$$

Where $L_v^{(i-1)} = \{c_u^{(i-1)} : u \in \mathcal{N}(v)\}$ and hash : $\mathcal{C}^k \times \mathcal{C} \to \mathbb{Z}$ is an injective hashing function. Intuitively, a node's color captures its k-hop neighborhood, and two nodes with the same k-hop neighborhood structure will receive the same colors. We can test for isomorphism between two graphs by applying the WL algorithm to each graph and comparing their color histograms; if they differ (i.e., one graph has a different number of nodes of a given color than the other), then we can reject the possibility of an isomorphism. Otherwise, it is possible that the graphs are isomorphic, but cannot be confirmed as the graphs may only be isomorphic up to k iterations.

Note that the WL algorithm fits our definition of message passing with $\phi = \text{hash}$, $\bigoplus = \text{multiset}$ (a function which returns its arguments as a multiset), and $\psi(\mathbf{h}_v, \mathbf{h}_u) = \mathbf{h}_u$. This indicates that message passing GNNs (MP-GNNs) are only as powerful as the WL test; that is, if the WL test is unable to distinguish a pair of non-isomorphic graphs, neither will an MP-GNN. This means that if two non-isomorphic nodes share similar local k-hop neighborhoods (thus indistinguishable by k iterations of the WL test), then they will receive identical representations from MP-GNNs, making certain tasks like node classification impossible when the two nodes belong to different classes. Unfortunately, the limitations of the WL test are well-documented (Arvind, Köbler, Rattan, & Verbitsky, 2015; Kiefer, Schweitzer, & Selman, 2015), which has led to stream of research examining to what extent MP-GNNs share the same limitations (Feng, Chen, Li, Sarkar, & Zhang, 2022; Morris, Lipman, et al., 2021; Morris, Fey, & Kriege, 2021; Xu, Hu, Leskovec, & Jegelka, 2019) and ways in which they can be overcome (Balcilar et al., 2021; Maron, Ben-Hamu, Serviansky, & Lipman, 2019; L. Zhao, Jin, Akoglu, & Shah, 2022), which can be grouped into several classes of approaches as follows.

Feature augmentations In general, surpassing the limits of the WL test is done by augmenting nodes with additional information so as to distinguish their representations when message passing alone cannot. A simple approach is to append one-hot encodings of each node's unique ID to their embeddings, which will guarantee the distinguishability of all nodes; however, this technique is inherently unable to generalize to unseen nodes or new graphs with more nodes than seen during training. We can instead heuristics based on a node's degree (C. Ying et al., 2021) or PageRank score (Postavaru et al., 2020) to derive uniquely identifying features for each node. In fact, even randomly generating additional features for each node has shown to be a surprisingly effective method for boosting the expressiveness of GNNs (Abboud, Ceylan, Grohe, & Lukasiewicz, 2020; Degraeve et al., 2022; Egressy & Wattenhofer, 2022; Sato, Yamada, & Kashima, 2020).

Positional encodings Another approach is to imbue each node's representation with an indication of its global position within its graph, also known as a *positional encoding* (PE) (You, Ying, & Leskovec, 2019). PEs can help differentiate nodes which share similar local structures while residing in distant locations in a graph, and conversely, help capture proximities between nearby nodes with differing local structures. Dwivedi, Luu, Laurent, Bengio, & Bresson (2022) propose two positional encoding schemes, Random Walk PE (RWPE) and Laplacian PE (LapPE). For node i, we obtain $\text{RWPE}(i) = [\mathbf{RW}_{i,i}, \mathbf{RW}^2_{i,i}, \ldots, \mathbf{RW}^k_{i,i}]$ where $\mathbf{RW}^l = (AD^{-1})^l$ is the l-step transition matrix, meaning each dimension $\text{RWPE}(i)_j$ represents the probability that node i will end on itself along a j-step random walk. Similarly, $\text{LapPE}(i) = [\mathbf{U}_{i,0,}, \mathbf{U}^2_{i,1}, \ldots, \mathbf{U}^k_{i,k}]$ where $U \in \mathbb{R}^{|\mathcal{V}| \times k}$ are the k smallest non-trivial eigenvectors of the graph Laplacian. However, this approach is limited by the sign ambiguity of eigenvectors which leads to 2^k possible signs for k eigenvectors, requiring the signs to be randomly flipped during training (Dwivedi et al., 2020; Kreuzer, Beaini, Hamilton, Létourneau, & Tossou, 2021). Additionally, many of the random walk and factorization-based methods discussed in Section 2.3.1 such as DeepWalk and LINE can be considered positional encodings as they too capture node proximities. One issue is that the absolute values of these methods may not be stable across graphs, as an independent model must be fit on each graph. H. Wang, Yin, Zhang, & Li (2022) propose the Position Equivariant Graph Neural Network (PEG) as a solution; by updating node features and positional encodings separately, they maintain permutation equivariance and improve the stability of node embeddings.

The vast array of approaches described above were comprehensively evaluated by Rampášek et al. (2022), who devised a highly general framework - GraphGPS - for graph learning including positional and structural (discussed in Section 2.4.2) encodings, message passing layers, and global attention mechanisms. In GraphGPS, a combination of positional and structural encodings are used to obtain node features $\mathbf{X}$ and edge features $\mathbf{E}$. Each layer of GraphGPS uses a message passing layer

$\mathbf{X}_M = \phi(\mathbf{X}, \mathbf{E}, \mathbf{A})$ to propagate the node and edge features within local neighborhoods, a global attention layer $\mathbf{X}_T = \alpha(\mathbf{X})$ to incorporate information from across the whole graph, and finally, the local and global embeddings are aggregated by a multi-layer neural network $\psi(\mathbf{X}_M + \mathbf{X}_T)$. This framework is not only provably expressive and scalable, it is also highly modular, allowing the various components of the model (such as choice of positional and structural encoding or message passing layer) to be substituted depending on the needs of the task at hand. The authors carefully examined each component of their framework, observing that while global attention can be helpful, the choices of positional and structural encodings and message passing layer were far more important to improving model performance. This work is highly relevant to the present thesis from the perspective of designing and comprehensively evaluating a *population* of models within a standardized framework for graph learning tasks. However, GraphGPS presented a more general framework aiming for applicability to all graph learning tasks, whereas this work focuses solely on building generalizable, inductive models for the link prediction task. In the following section, we will briefly review other works which examine populations of models, also known as *design spaces*.

2.3.4 Design Space of Graph Neural Networks

With such a wide range of methods for learning graph representations, it is important to evaluate which methods are best suited for various applications. Examining populations of models in this way is also known as profiling a *design space* (Radosavovic, Johnson, Xie, Lo, & Dollár, 2019). The first large-scale study on design spaces for Graph Neural Networks was contributed by GraphGym (You et al., 2020), which proposed a number of fundamental "design dimensions" for GNNs, including:

- Batch normalization
- Dropout
- Non-linear activation
- Agggregation function
- Number of message passing layers
- Layer connectivity
- Pre-processing layers
- Post-processing layers
- Batch size
- Learning rate

- Optimizer
- Training epochs

With multiple choices for each dimension, there are over 10 million possible model configurations in this design space, making it unrealistic to evaluate all possible models. To make this tractable, the authors devised the Controlled Random Search (CRS) experimental procedure which uses a randomized block design to evaluate each dimension. For example, consider a study wishing to find the optimal batch size $\text{BatchSize} \in \{32, 64, 128\}$ within the design space $\{\text{BatchNorm} : [\text{True}, \text{False}], \text{LR} : [0.01, 0.001, 0.0001]\}$. We would sample $|S|$ configurations from the cartesian product of the free dimensions $S = \text{BatchNorm} \times \text{LR}$, and then take the cartesian product with the fixed dimension $S \times \text{BatchSize}$. This greatly reduces the overall quantity of experiments that must be run to identify the optimal batch size while simultaneously providing a consistent evaluation across all choices for the dimension of interest. This led to a number of interesting findings such as batch normalization being more helpful than dropout, PReLU being highly effective as a non-linear activation function, and a batch size of 32 being preferable to either 16 or 64. It also confirmed several empirically held beliefs, such as sum being the best aggregation function, Adam being generally superior to SGD, and more training epochs lead to better performance.

The methodology proposed by GraphGym was extended to the dynamic graph setting by ROLAND (You, Du, & Leskovec, 2022), who used CRS to conclude that batch normalization, skip-connectivity, and max aggregation were optimal design choices for their proposed architecture, and to the heterogenous setting by T. Zhao et al. (2022), who found that their results on batch normalization, number of message passing layers, optimizer, and training epochs were similar to those of GraphGym. They also found that heterogenous GNNs had several important differences, such as Dropout being a prerequisite of good performance, which the authors suggest is due to over-parameterization created by the node and edge-type specific model weights. Finally, a particularly relevant study was conducted by Z. Wang, Zhao, & Shi (2022), who profiled the design space of models for the collaborative filtering (CF) task. Through their evaluation, the authors identified a “pruned” designed space which was shown to contain a higher concentration of well-performing models, an important contribution for both future research and applications as it provides a significantly improved “starting point” for researchers and practitioners designing CF models. This work is of great interest to the present thesis as it studies the design space of collaborative filtering, which can be formulated as a link prediction problem. However, the study also examines non-GRL models specific to CF whereas we focus solely on GRL-based models. Their study also focuses solely on performance in the transductive setting, as their generalization analysis examines how well their reduced design space transfers to new datasets, not how actual trained models can be applied to new datasets. This thesis goes beyond the transductive setting to investigate the applicability of existing trained models to completely novel datasets, evaluating their effectiveness in the inductive link prediction task where all nodes are previously unseen.

2.4 Link Prediction

In this section, we will review the existing body of literature concerning the application of deep learning methods to the link prediction task. We will first cover the earliest deep learning-based approaches to link prediction, which were followed by the emergence of Graph Neural Networks (GNN). We will then discuss the challenge of inductive link prediction, including the approaches and datasets contributed to its line of research.

2.4.1 Learning Representations for Link Prediction

Embedding-based approaches Link prediction is a task particularly well-suited for supervised learning methods - especially deep learning based-approaches - given that ground-truth labels can automatically be obtained from any graph by sampling from the edges. One of the first works to demonstrate this was Node2Vec (Grover & Leskovec, 2016), which compared established heuristic approaches to deep learning approaches. Node2Vec, DeepWalk (Perozzi et al., 2014), and LINE (Tang, Qu, Wang, et al., 2015) were used to learn embeddings $f : \mathcal{V} \rightarrow \mathbb{R}^D$ for a given graph. Each link $(u, v) \in \mathcal{V}$ to be classified was scored as $f \circ g = g(f(u), f(v))$, where g was a binary operator mapping the two embeddings to a fixed-size vector, such as taking the average of the Hadamard product. In parallel to these works came a number of deep learning approaches to the task of knowledge graph completion (KGC). Knowledge graph completion is an instance of link prediction specific to knowledge graphs, where each edge is a directed triple (h, r, t) consisting of a head node (node), relation (typed edge), and tail node. Optimizing the embeddings of nodes and relations such that linked nodes have similar embeddings proved to be a highly effective approach. The TransE (Bordes et al., 2013) model optimizes embeddings such that $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$, while RotatE (Sun, Deng, Nie, & Tang, 2018) uses the objective $\mathbf{h} \circ \mathbf{r} \approx \mathbf{t}$.

Message passing approaches The learning-based approaches demonstrated their efficacy by outperforming the heuristics by sizeable margins, a trend further demonstrated alongside the development of Message Passing Neural Networks (MPNNs). The Variational Graph Auto-Encoder (VGAE) (Kipf & Welling, 2016) used a Graph Convolutional Network (GCN) as the encoder and a dot-product encoder to extend the Variational Auto-Encoder to the graph domain, resulting in significant performance improvements relative to DeepWalk. These approaches were extended to the knowledge graph completion task as well, with methods such as CompGCN (Vashishth et al., 2020) combining message passing networks and KGC scoring models. The strong performance of MPNNs for link prediction has also spurred improvements in related tasks. For instance, many recommender system problems can be formulated as link prediction, allowing various MPNNs such as the VGAE (Berg, Kipf, & Welling, 2017), GraphSAGE (R. Ying et al., 2018), and Graph Attention Networks (X. Wang, He, Cao, Liu, & Chua, 2019) to be applied to tasks such as collaborative filtering.

2.4.2 Inductive Link Prediction

Methods

A common challenge for link prediction methods is the ability to handle unseen data. Many of the aforementioned methods are transductive, storing a unique set of parameters for each node. When unseen nodes are present in new data, these methods will perform poorly as their representations of these unseen nodes have not been optimized. In constrast, some MPNNs are inductive, meaning that they operate directly upon the attributes of a given node. However, it is not always reasonable to assume the presence of node attributes in realistic scenarios; for example, nodes in knowledge graphs rarely have a standardized, reliable set of properties, and user-derived graphs (such as co-purchase networks, social media networks, and communication networks) may mask node attributes for privacy protection. This has led to *fully inductive* link prediction methods which learn solely from the structure of the graph without depending on the presence of node attributes. There have been a number of approaches toward handling unseen nodes in link prediction tasks, which broadly fall under two main categories.

Neighborhood aggregation The first class of approaches assumes that while a node may be unseen at inference, its neighbors could have been seen during training, making it possible to exploit Message-Passing Neural Networks to fill in unseen node' embeddings. Hamaguchi, Oiwa, Shimbo, & Matsumoto (2017) propose a method for knowledge graph completion that assigns a unique embedding to each node which is then updated using message-passing and aggregation. In contrast to previous KGC methods, this ensures that even when an node is unseen (meaning its initial embedding is unoptimized and effectively random), its final embedding will be supported by the optimized embeddings of its neighbors. The model is trained end-to-end using TransE (Bordes et al., 2013) to score triples, demonstrating consistently strong performance on different partitions of the WordNet knowledge base even when increasing the number of unseen nodes at inference. This idea was extended by the Logic Attention Network (LAN) (P. Wang, Han, Li, & Pan, 2019), which satisfies a set of key requirements for the aggregation step, including permutation invariance, redundancy awareness, and relational awareness. LAN applies relation-specific transformations and adapatively aggregates each node's neighborhood by utilizing the attention mechanism (Bahdanau et al., 2015) to weigh to each neighboring node, and a logic rule mechanism to weigh the importance of the relation between each node and its neighbors. When compared to the mean-based approach proposed by Hamaguchi et al. (2017) using the same datasets, LAN showed significantly improved performance on inductive link prediction. This is similar to the approach taken by the Virtual Neighbor (VN) Network (Y. He, Wang, Zhang, Tu, & Ren, 2020), which offers an improvement to the logic rule mechanism by using a knowledge graph rule-mining tool to assign a soft score to reasoning paths extracted from random walks originating at each unseen node. After applying a threshold-based filter, the remaining paths are used to add new "virtual" triples to the knowledge graph between the original unseen node and the final node along the path. This approach yields superior performance to the mean-based approach as well as LAN when evaluated on datasets extracted from Freebase and YAGO.

Structural encodings The second class of methods for inductive link prediction are known as structural encodings. In contrast to the first approach, SE are built under the assumption that the inference graph will be novel, meaning that each method must generalize to completely new - and possibly unattributed - graphs. In order to do this, a model must be able to learn solely from the structure of the graph without using any node-specific information. These approaches resemble the positional encodings discussed in Section 2.3.3, but instead of deriving *node*-specific distinguishing features, structural encodings compute *edge*-specific features, making them particularly well suited for the link prediction task. The first work to tackle this problem using deep learning was the Weisfeiler-Lehman Neural Machine for Link Prediction (WLNM) (Zhang & Chen, 2017), which proposed a novel approach to link prediction by extracting subgraphs around each target link. Each node in a subgraph was assigned a color using a variation of the Weisfeiler-Lehman algorithm (Leman & Weisfeiler, 1968) that initialized each node's color as a function of its *mean geometric distance to the target link*. These colors were used as a sorting key to impose an order on the vertices for the construction of an adjacency matrix, which was used as input to a neural network scoring function that predicted the validity of the target link. This results in each training (and testing) instance being its own novel graph, allowing the method to learn solely from the extracted structures and easily generalize to unseen graphs.

This work was fortified by SEAL (Zhang & Chen, 2018), which uses a Graph Neural Network in place of a fully-connected Neural Network to best incorporate the full breadth of information provided by a given graph, including the adjacency structure and potential (but still optional) node attributes. SEAL simplifies the WLNM's labeling procedure by introducing Double-Radius Node Labeling (DRNL), which issues progressively larger labels to nodes more distant from the target link. A node's embedding is then computed as the one-hot encoding of its label, which is optionally concatenated to its attributes if available. The embeddings and structure of each subgraph is then used as input to a Subgraph Neural Network, which propagates the embeddings across the subgraph, pools the embeddings into a final fixed-size subgraph encoding, and classifies the subgraph as positive or negative. SEAL outperformed a diverse set of methods including heuristics, factorization, and message passing networks, setting a new state-of-the-art for link prediction. SEAL was quickly adapted to other domains in which inductive learning is necessary, such as collaborative filtering (Zhang & Chen, 2020) and knowledge graph completion (Teru, Denis, & Hamilton, 2020). This class of node labeling approaches was examined in-depth by Zhang, Li, Xia, Wang, & Jin (2020), who coined the term "labeling trick" to describe these approaches. Their work derives important theoretical results proving that the labeling trick enables any sufficiently expressive GNN to learn maximally expressive multi-node representations for set-level tasks such as link prediction. They verify their results empirically, showing that labeling trick methods outperform heuristics and graph auto-encoders on a number of benchmark datasets.

Applications

While the methods described above bear the ability to perform inductive link prediction, little attention has been paid to their application in realistic scenarios. This has led to several works examining the generalization ability of these inductive models across unseen nodes, and even unseen graphs. The most prevalent of these works have focused on knowledge graph completion. For instance, Sadeghi, Malik, Collarana, & Lehmann (2021) designed a benchmark of 32 datasets spanning multiple relational patterns in both transductive and inductive settings. Using the Freebase and Wordnet knowledge bases, the authors extract triples matching specific relational patterns such as symmetry (a relation which is valid even when its head and tail are reversed) and anti-symmetry, inversion (a relation r which can be uniquely mapped to another relation r' such that $(h, r, t) \implies (t, r', h)$, such as Parent-Child and Teacher-Student), composition (a relation which depends on the existence of another relation, such as how Aunt depends on Sister and Child), and inference (a relation such which can be mapped to other relations r'_i such that $\forall h, t : (h, r, t) \implies (h, r_i, t)$. The authors split the extracted triples into training and inference sets following one of several partitioning strategies, including fully transductive and inductive partitions, head-tail ratio inductivity (in which either the head or tail of each triple in the training set is held out of the inference set), and percentage-wise building (in which half of the inference triples are inductive and half are transductive). Training a number of well-established knowledge graph completion models on these datasets revealed clear challenges in the inductive settings, as models tailed for inductive learning outperformed others by wide margins across all relational patterns. This dataset-construction strategy was scaled up by Galkin, Berrendorf, & Hoyt (2022), who shared the largest existing dataset for inductive link prediction by sampling and inductively partitioning WikiData. By sampling hold-out nodes for the inference graph, the authors obtain two datasets (ILPC22-Small and ILPC22-Large) with disjoint node sets, facilitating the evaluation of models in a setting where completely unseen nodes are present at inference.

These works demonstrate that certain inductive models can generalize to previously unseen nodes, but what about entirely new graphs? Zhang & Chen (2020) showed that Inductive Graph Matrix Completion (IGMC), an extension of SEAL, could obtain competitive performance on a given user-product graph even when trained on a different graph from a potentially completely different domain (e.g., training on a user-movie graph and transferring to a user-song graph), without any fine-tuning or transfer learning on the target graph. A similar study was performed by H. Wang et al. (2022), which proposed the *"domain-shift link prediction"* task. The authors examine the in-domain generalization ability of a number of methods by training each on one graph and testing it on a related graph (e.g., training and testing on two different citation networks, such as Cora $\rightarrow$ CiteSeer (Sen et al., 2008)). They find that SEAL and PEG, their proposed method discussed in Section 2.3.3, are easily able to generalize to new graphs. While these studies find that certain methods are able to generalize to unseen data, little consideration is given to what makes this possible. In contrast, this thesis provides a comprehensive examination across the various components of inductive link prediction methods to share insight into what design decisions facilitate or hamper generalization.

Chapter 3

Design Space for Inductive Link Prediction

In this section, we will outline our methodology in profiling the design space of models for Inductive Link Prediction (ILP). We will share our proposed standardized architecture consisting of several components, covering candidate methods for each before detailing the experiments necessary to verify our hypotheses.

3.1 Standardized Architecture

To identify an optimal model (or set of models) for inductive link prediction, we require a standardized, modular architecture which allows us to swap in and out its various components to study the effects of each design choice. Furthermore, in order for the insights yielded by our experiments to be usable in real-world scenarios, our architecture must satisfy several criteria:

1. **Efficient.** Realistic graph datasets often contains tens or hundreds of millions of nodes, and hundreds of millions - if not billions - of edges. Existing GNN architectures which operate on the full adjacency matrix of a graph are simply unable to scale to this size. As such, our framework must be able to operate on a reduced subset of a graph's adjacency matrix if necessary.
2. **Effective.** Of course, an efficient, poorly performing model is no more useful than an inefficient, well-performing model. Therefore, our framework must maintain strong performance even in resource-constrained scenarios.
3. **Generalizable.** Finally, our framework must be *fully* inductive, meaning that it must not depend on (1) the specific identities of nodes in a given graph, or

(2) the presence of pre-existing node or edge attributes for a given graph. The former criterion is important not only for cross-graph generalization ability, but also for efficiency as it will allow the model to operate on arbitary subgraphs if necessary. The latter criterion will increase the applicability of the framework to more realistic scenarios involving non-attributed graphs (such as many social networks).

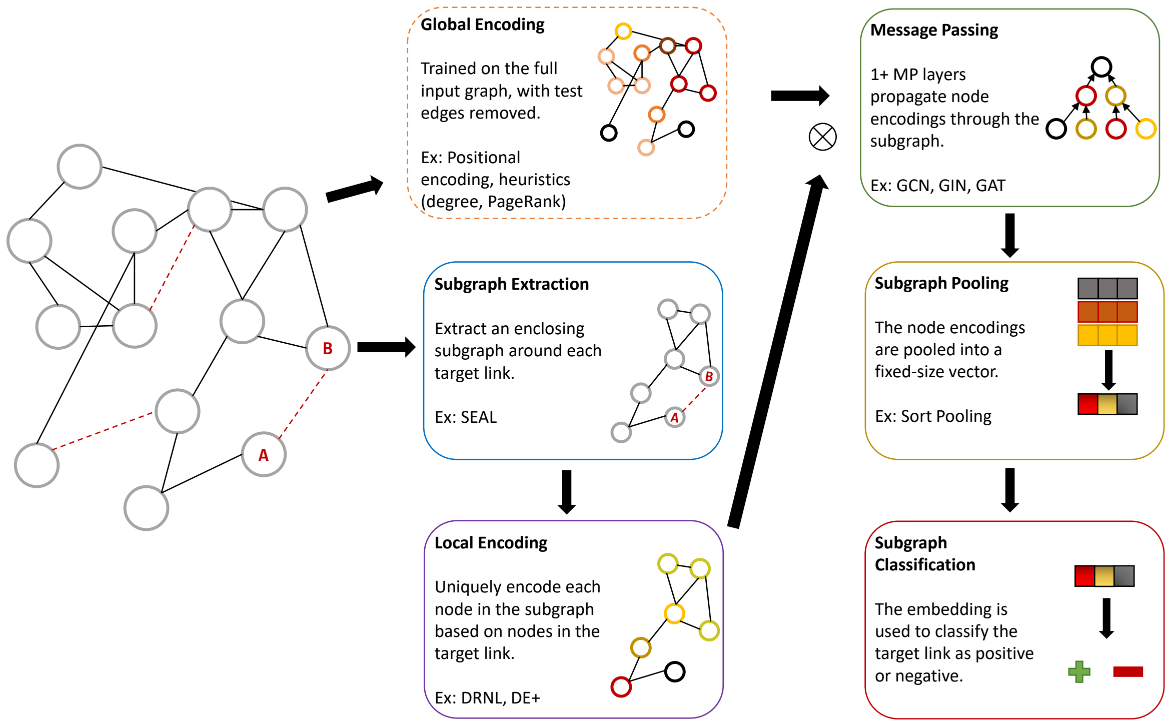

Global Encoding
Trained on the full input graph, with test edges removed.
Ex: Positional encoding, heuristics (degree, PageRank)
Subgraph Extraction
Extract an enclosing subgraph around each target link.
Ex: SEAL
Local Encoding
Uniquely encode each node in the subgraph based on nodes in the target link.
Ex: DRNL, DE+
Message Passing
1+ MP layers propagate node encodings through the subgraph.
Ex: GCN, GIN, GAT
Subgraph Pooling
The node encodings are pooled into a fixed-size vector.
Ex: Sort Pooling
Subgraph Classification
The embedding is used to classify the target link as positive or negative.
A
B

Figure 3.1: Standardized architecture for inductive link prediction (ILP) models. Given an input graph, a model instance computes global encodings on the full graph before sampling subgraphs around each target link. A local encoding is derived for nodes in each subgraph, which is concatenated with their global encoding to form positionally and structurally-aware node embeddings. The embeddings are propagated using message passing layers before being pooled into a subgraph embedding which is used to classify the existence of each target link.

Given the above criteria, we propose a simple, standardized model architecture for inductive link prediction (ILP) comprised of 6 components, shown in Figure 3.1. The modularity of our proposed architecture allows us to swap out different choices for each component to profile the overall design space. It does not depend on the presence of node or edge attributes but is able to utilize them without any additional modification (though we choose not to in this work to focus on learning and generalizing solely from the structure of the graph). Our framework scales to large graphs by operating on k-hop subgraphs around each target link while (optionally) preserving global information. Concretely, an instance of our framework contains the following components:

1. **Global encoding** (optional). Given an input graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, a global encoding module is used to extract global structure-aware node embeddings $\mathbf{X}_g \in \mathbb{R}^{|\mathcal{V}| \times m}$. The importance of this module is to generate representations that uniquely distinguish the nodes in a given graph to help differentiate nodes with isomorphic k-hop neighborhoods. This module must be efficient enough to operate on the full adjacency structure of the graph, and may also be omitted entirely if a graph's local neighborhoods are sufficiently distinguishable. Candidates for this component include simple, lightweight heuristics such as node degree, centrality metrics, or even randomly generated features, as well as more sophisticated options such as positional encodings.
2. **Subgraph extraction**. Following the SEAL framework (Zhang & Chen, 2018), for each target link $\bar{e}_i = ((u, v), y_i) : u, v \in \mathcal{V}, \bar{e} \notin \mathcal{E}, y_i \in \mathcal{Y}$ which we wish to predict, we extract a k-hop subgraph $H_{u,v}^{(k)} = \{\mathcal{V}', \mathcal{E}'\} \subset \mathcal{G}$ enclosing u and v. Each node and edge carries their respective attributes, including the global encoding from step 1 if applicable, into the subgraph $H_{u,v}^{(k)}$ as $\mathbf{X}_a \in \mathbb{R}^{|\mathcal{V}'| \times d}$ and $\mathbf{X}_g \in \mathbb{R}^{|\mathcal{V}'| \times m}$ respectively. This step makes it tractable to operate on large graphs by limiting the receptive field of the model, while still preserving global information from step 1 and node/edge attributes if provided (which we ignore for the purposes of this study).
3. **Local encoding** (optional). Given an enclosing subgraph $H_{u,v}^{(k)}$, the local encoding component derives local structure-aware node embeddings $\mathbf{X}_s \in \mathbb{R}^{|\mathcal{V}'| \times n}$. This component helps distinguish otherwise isomorphic k-hop subgraphs by computing context-sensitive representations with respect to the nodes u, v in the target link. These representations can either be real-valued embeddings, or integer labels $\ell \in \mathbb{Z}_+^l$ which are used as inputs to an embedding layer. If computing integer labels, the labels must generalize between graphs. A simple example of a generalizable label would be Distance Encoding (Li, Wang, Wang, & Leskovec, 2020), which returns a label tuple $\mathbf{x}_q = (\mathrm{ssp}_{q,u}, \mathrm{ssp}_{q,v})$ for all $q \in H_{u,v}^{(k)}$, representing the shortest path distance from every node to each node in the target link. The labels are then embedded and concatenated to form the local encoding. This step may also be omitted if global information is sufficient

to distinguish nodes in the graph.

4. **Message passing** (optional). Given an enclosing subgraph $H_{u,v}^{(k)}$ with node attributes and global and local encodings $\mathbf{X}_a \in \mathbb{R}^{|\mathcal{V}'|\times d}, \mathbf{X}_g \in \mathbb{R}^{|\mathcal{V}'|\times m}, \mathbf{X}_s \in \mathbb{R}^{|\mathcal{V}'|\times n}, d, m, n \geq 0$, this component uses between one or more message passing layers to propagate node embeddings throughout the local subgraph. Note that omissions of node attributes and global and local encodings are accomodated here by allowing $d = 0$ (no attributes), $m = 0$ (no global encoding), and $n = 0$ (no local encoding), though of course we must have at least one of $d > 0 \vee m > 0 \vee n > 0$, otherwise we have no features to learn from. Any given message passing layer can be used here by propagating the concatenated attributes and encodings $[\mathbf{X}_a; \mathbf{X}_g; \mathbf{X}_s]$ to incorporate information from all available aspects. Additional flexibility is also provided for specific MPNNs; for example, edge features can also be passed to the MPNN layer if it supports it, and PEG (H. Wang et al., 2022) or GraphGPS (Rampášek et al., 2022) can be used as the message passing layer by using a positional encoding in step 1 and separately passing $[\mathbf{X}_a; \mathbf{X}_s]$ as node attributes and $\mathbf{X}_g$ as positional encodings.
5. **Subgraph pooling**. With node embeddings $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}'|\times p}$ that capture all aspects available in the enclosing subgraph, $H_{u,v}^{(k)}$, the subgraph pooling component maps $\mathbb{R}^{|\mathcal{V}'|\times p} \to \mathbb{R}^c$. The pooling function must be length agnostic and position invariant, capable of aggregating the unordered set of node embeddings to a fixed-size subgraph embedding. The most simple choice would be a simple arithmetic operation such as summation, while a more sophisticated approach might use a learnable aggregation function such as the attention mechanism.
6. **Subgraph classification**. Finally, the fixed-size subgraph embedding is put through a fully-connected classification layer mapping to an output $\hat{y}_i \in \mathcal{Y}$, where the label space $\mathcal{Y}$ is usually binary in $0, 1$ but can extend to multi-class and ordinal spaces as well as a continuous space for regression problems with a properly-chosen objective function. For this work, we focus on the traditional task of binary link prediction, and optimize the model end-to-end using the binary cross-entropy loss.

3.2 Proposed Design Space

Table 3.1: Proposed design space for inductive link prediction.

Design Dimension	Choices
Local Encoding	None, Degree, RW, RNI, HOPE, RandNE, LE
Global Encoding	None, DE, DE+, DRNL
Message Passing Network	GCN, GIN, GAT, GraphSAGE, PEG
Message Passing Layers	1,2,3
Hidden Channels	32,64,128
Optimizer	Adam, SGD
Learning Rate	0.01, 0.001, 0.0001

With our framework in place, we propose a design space for inductive link prediction shown in Table 3.1. Our design space consists of 7 dimensions, 27 options, and $7,560$ total unique model architectures. Throughout the course of our experiments, we exclude a portion of these from sampling when certain components are incompatible with each other; for example, the PEG message passing layer expects both node attributes and positional encodings, so we exclude all models with *None* as the global encoding and PEG as the message passing layer. Similarly, we exclude all models with *None* as both the local and global encodings, as no node attributes would be computed for the downstream message passing layer. Below, we motivate the selections made for each dimension of our design space.

- **Global encodings**. We use global encodings to capture features which represent each node's position in a given graph, prior to global subgraph extraction. We consider a number of different global encodings, including: node degrees (a single scalar representing the combined in-and-out degrees of each node); Random Node Initialization (RNI) (Abboud et al., 2020), which generates completely random features for each node and has shown to improve the expressivity of MPNNs; Random Walk Positional Encoding (RW) and Laplacian Positional Encoding (LE)(Dwivedi et al., 2022), detailed in Section 2.3.3; and finally High-order Proximity preserved Embedding (HOPE) (Ou, Cui, Pei, Zhang, & Zhu, 2016), a method which produces embeddings preserving the distance between two nodes at multiple powers of the adjacency matrix. We also examine performance when omitting a global encoding, signified by *None*, in which case we only use local encoding features.
- **Local encodings**. We use local encodings, also known as labeling tricks, to inductively extract structural features which generalize across (sub-)graphs. We

consider three local encodings examined by (Zhang et al., 2020), including Distance Encoding (DE) (Li et al., 2020), Distance Encoding Plus (DE+), and Double Radius Node Labeling (DRNL) (Zhang & Chen, 2018). Each of these methods compute the shortest path distance to each of the nodes in the target link upon which the subgraph is rooted, using this as a provably expressive set of features to distinguish isomorphic edges which MPNNs would otherwise be unable to. We also examine performance when omitting a local encoding, signified by *None*, in which case we only use global encoding features.

- **Message passing networks**. We explore several of the most widely used and state-of-the-art message passing networks, including Graph Convolutional Networks (GCN) (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), and Graph Isomorphism Networks (GIN) (Xu et al., 2019), detailed in Section 2.3.2. In addition, we also consider the Position Equivariant Graph Neural Network (PEG) (H. Wang et al., 2022) due to its relevance to our task. PEG is designed to improve the stability of message passing with global encodings. The authors note that using global encodings as features to an MPNN can lead to instability between graphs, as even small differences between graphs can lead to large differences in global encodings and poor transferability as a result. PEG solves this issue by updating node attributes and global encodings separately, using the latter as coordinates to compute the distance between two nodes which is used to modify the edge weights. It demonstrates strong performance on a number of tasks, including *domain-shift link prediction*, a similar setting to ours where models are trained on one graph and applied to another in the same domain (a subet of the present work, which considers cross-domain applications as well). However, its efficacy when applied to subgraphs - a more realistic application facilitating its use on large-scale graphs - has yet to be established, making it an important inclusion in our design space.
- **Message passing layers**. We select $\{1, 2, 3\}$ as the possible numbers of layers within our message passing networks, choosing these values due to the subgraph extraction process in our framework. As the MPNNs are applied to k-hop subgraphs around the target nodes instead of the full graphs, we tie the number of message passing layers to the size of the extracted subgraph to avoid oversmoothing. We examine all values less than 4 due to the neighborhood explosion problem of subgraph sampling as subgraphs grow in size exponentially with k, decreasing extraction efficiency and increasing GPU memory usage during training.

- **Hidden channels**. We select $\{32, 64, 128\}$ as our space of hidden channels within our message passing networks. We selected these options using the following process. In general, dimensions of deep learning models are often chosen as powers of two. We limit the number of choices to three to help maintain a reasonable total number of trials required. As such, we identified $2^7 = 128$ to be the largest value minimizing the frequency of out-of-memory errors, and selected the next two smallest powers of two $2^6 = 64, 2^5 = 32$.
- **Optimizer**. Following (You et al., 2020), we select Adam (Kingma & Ba, 2015) and Stochastic Gradient Descent (SGD) as the optimizers within our design space, as they are the two most widely used optimizers across most deep learning tasks.
- **Learning rate**. We select $\{0.01, 0.001, 0.0001\}$ as our space of learning rates to provide reasonable values of multiple scales. While exploring a wider range of learning rates and different sampling strategies would be an interesting direction for future work, we opt for simplicity as learning rate is not a primary focus of the current work.

3.3 Experimental Design

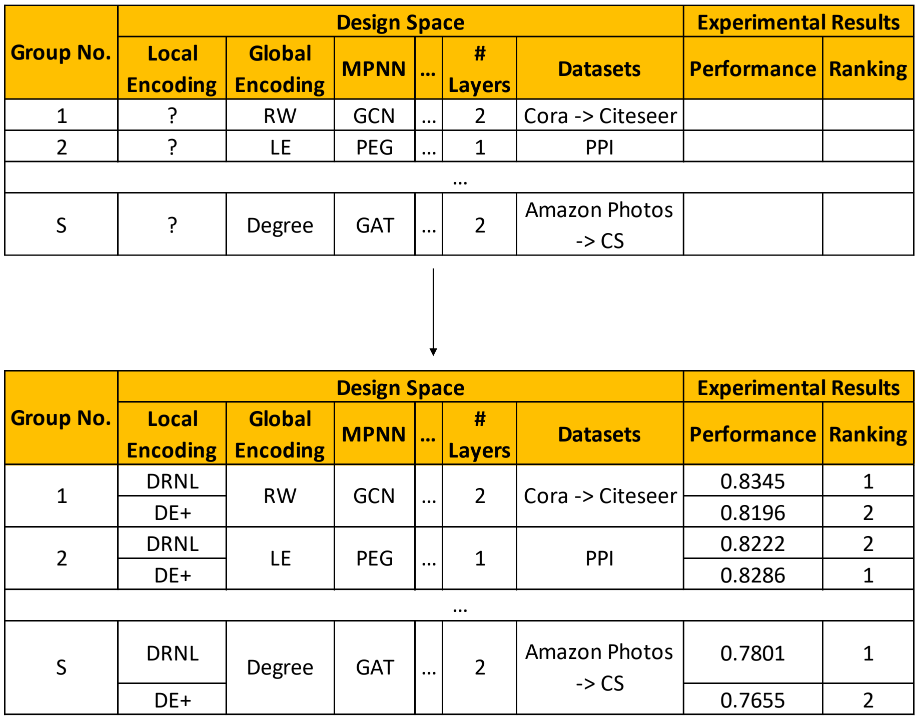
The following section will detail the specific experiments we will conduct to verify the hypotheses detailed in Section 1.2. Each of the experiments we run will consider the design space $\mathcal{D}$ shown in table 3.1 across the task space $\mathcal{T}$ shown in table 4.1. Evaluating the full design and task spaces would require considering $\prod_{d \in \mathcal{D}}(|d|) \cdot \sum_{t \in \mathcal{T}}({}^{|t|}P_2) = 7560 \cdot 418 = 3,160,080$ different trials, with each potentially repeated across multiple random seeds. To mitigate the prohibitive costs of such a large number of trials, we employ Controlled Random Search (CRS) (You et al., 2020) as an effective way to reduce the number of necessary trials by sampling. The process is demonstrated visually in Figure 3.2. As a simple example, suppose we hypothesize that DRNL is a superior local encoding to DE+. This makes our design dimension of interest $\hat{d} = LE$. We first draw $|S|$ configurations $S = [s_i = \{\ s_{i,d} \approx d : d \in \mathcal{D}, d \neq \hat{d}\}]$; simply put, we randomly sample $|S|$ configurations across all design dimensions (including tasks/datasets) other than the one of interest. The final set of experiments is obtained by taking the cartesian product

$$E = S \times \hat{d} = [(i, s_i, \hat{d}_j) : i \in [1, |S|], \hat{d}_j \in \hat{d}]$$

Where S is the set of sampled configurations and $\hat{d}$ is the set of possible values of our dimension of interest, maintaining a reference i to the ID of the sample configuration (shown as "Group No." in Figure 3.2). Conducting each experiment yields a set of performance metrics m, which we compute for each the rankings *within* the sample group number i. Intuitively, this represents the performance rank $r_{\hat{d}_j,i}$ of each value $\hat{d}_j$ within a group i where all other parameters are held constant. Finally, for each $\hat{d}_j$, we take the mean rank:

$$r_{\hat{d}_j} = \frac{1}{|S|}\sum_{i=1}^{|S|} r_{\hat{d}_j,i} \tag{3.1}$$

This represents the expected rank for each value of $\hat{d}$ when all other parameters are held constant. In this case, with $\hat{d} = d_{LE} = \{DRNL, DE+\}$, we decrease the total number of required experiments from $1,580,040$ to just $|S| \cdot |\hat{d}| = 2|S|$, where $|S|$ is specified by the user. Taking $|S| = 75$ for instance, only 150 trials are required to be executed, a reduction of over $10,000$ times.



Group No.	Design Space						Experimental Results	
	Local Encoding	Global Encoding	MPNN	...	# Layers	Datasets	Performance	Ranking
1	?	RW	GCN	...	2	Cora -> Citeseer		
2	?	LE	PEG	...	1	PPI		
...								
S	?	Degree	GAT	...	2	Amazon Photos -> CS		

Group No.	Design Space						Experimental Results	
	Local Encoding	Global Encoding	MPNN	...	# Layers	Datasets	Performance	Ranking
1	DRNL	RW	GCN	...	2	Cora -> Citeseer	0.8345	1
	DE+						0.8196	2
2	DRNL	LE	PEG	...	1	PPI	0.8222	2
	DE+						0.8286	1
...								
S	DRNL	Degree	GAT	...	2	Amazon Photos -> CS	0.7801	1
	DE+						0.7655	2

Figure 3.2: Example of Controlled Random Search to evaluate whether DRNL or DE+ is a better local encoding. We take the cartesian product $S \times \hat{d}$ of a set of S random configurations with the possible values for our dimension of interest $\hat{d}$ to obtain our experiment set, and compute the average in-group rank for each value to determine the optimal choice. Based on the rows visible in the figure, DRNL would have an average rank of 1.33 versus DE+'s average rank of 1.67, making DRNL the preferable choice.

For all experiments, we use $|S| = 25$ for Controlled Random Search (CRS) and repeat each trial with 3 different random seeds, yielding $75|\hat{d}|$ trials in total per experiment. We fix batch size to 64 to ensure each model is allocated an equal amount of computational resources across trials. As the graphs are also of varying size, we limit each trial to 2500 training steps. The result of each trial will yield measurements of $\mathcal{M} =$ accuracy, AUC, F1, precision, and recall on the test set of edges for both the source and target graphs.

3.3.1 Population-level Experiments

Evaluating Generalization Ability

The first experiment we will conduct serves to evaluate the generalization ability of ILP models at a population level. We perform Controlled Random Search (CRS) with $|S| = 25$ on the *target* graph, meaning that we jointly and randomly sample 25 model instances and source graphs which we evaluate on each target graph. For each metric $m \in \mathcal{M}$, we measure *relative generalization* as the mean signed difference $\Delta_m = \frac{1}{N}\sum_{i=1}^{N}(m_{i,s} - m_{i,t})$ where $\Delta_m \in [-1, 1]$ (as all $m \in [0, 1]$), $m_{i,s}$ is the value of a given metric on the source graph for trial i, and $m_{i,t}$ is defined similarly for the target graph. A lower Δ_m signifies a smaller decrease in performance, and therefore better generalization ability. The value of Δ_m will determine the validity of Hypothesis 1.1 by measuring the degree to which the performance of an ILP model changes when applied to unseen data. The null hypothesis $H_\emptyset : \forall m : \Delta_m \geq 0$ asserts that ILP models will *not* show a decrease in performance from the source graph to the target graph on average, while the experimental hypothesis $H_1 : \forall m : \Delta_m < 0$ asserts the contrary.

Effects of Domain on Generalization

We extend the results of our first experiment by analyzing relative generalization through the lens of the underlying data. Each target graph was paired with 25 source graphs; as a result, some of these source-target pairs will come from the same datasets (*domains*) while others will not. We refer to the former scenario as *in-domain* ILP (ID-ILP), and the latter *out-of-domain* ILP (OOD-ILP). This allows us to dive more deeply into the behavior of ILP models by analyzing the differences exhibited between these two scenarios; for example, ILP models may successfully generalize to graphs in the same domain in which they were trained, but they may fail to generalize as well to out-of-domain graphs, a useful insight for practitioners who wish to apply ILP models to a new dataset. We measure these effects to validate Hypothesis 1.2 by comparing the average generalization ability of ILP models when trained in the ID setting versus OOD. The null hypothesis for this experiment $H_\emptyset : \forall m : \mu_{\Delta_m,ID} \geq \mu_{\Delta_m,OOD}$ asserts that for each $m \in \mathcal{M}$, average generalization ability in the ID setting will be equally or more difficult than the OOD setting, while our experimental hypothesis $H_1 : \exists m : \mu_{\Delta_m,ID} \leq \mu_{\Delta_m,OOD}$ expects that there exists some metric for which models trained in-domain are able to generalize more effectively than those trained out-of-domain, demonstrating the increased difficulty of the OOD setting.

3.3.2 Model-level Experiments

Optimal Selections by Design Dimension

Our next experiment will provide the evidence necessary to verify Hypothesis 1.3, which states that there is a significant difference in generalizability between the design dimensions of our model. Furthermore, we also wish to identify the best choices for each design dimension. We perform CRS on each design dimension $d \in \mathcal{D}$ shown in Table 3.1, producing results for $N = C \cdot \sum_{d \in \mathcal{D}} |d|$ trials in total. We compute the mean of the ranks for each metric m of each design choice within each experiment group as shown by Figure 3.2. This will yield a sample mean μ_{d,Δ_m} (calculated similarly to Section 3.3.1) for each design dimension d and metric m, as well as an average rank r_{d_i,Δ_m} for each candidate choice $d_i \in d, d \in \mathcal{D}$ and each metric m. The sample mean μ_{d,Δ_m} signifies the average transferability across all candidates for a design dimension d and metric m, holding all other design dimensions constant. We can verify Hypothesis 1.3 using a one-way ANOVA with Bonferroni correction, where our null hypothesis $H_\emptyset : \forall \mu_{d,\Delta_m}, \mu_{d',\Delta_m} : |\mu_{d,\Delta_m} - \mu_{d',\Delta_m}| = 0$ expects no difference among group means and our alternative hypothesis $H_1 : \exists \mu_{d,\Delta_m}, \mu_{d',\Delta_m} : |\mu_{d,\Delta_m} - \mu_{d',\Delta_m}| \neq 0$ expects at least one group mean to significantly differ from the overall mean. In addition, we can use the set of mean ranks $r_{d_i,m}$ to rank the performance of each candidate within a given design dimension, yielding insight into the optimal set of choices for each component of our framework.

Identification of Condensed Design Space

Based on the results of the above experiment, we will construct a condensed design subspace $\hat{\mathcal{D}} = \{\hat{d}_i\}$ such that $\forall i : \hat{d}_i \subset d_i$. Such a subspace would be of great practical utility to both practitioners and researchers interested in the ILP task, as it would greatly reduce the cardinality of the design space to be explored when searching for optimal models. For our proposed subspace to be useful, however, we must verify that it is actually superior to the full design space. As such, we will apply the same technique from the first experiment described in Section 3.3.1 on samples of models from both the proposed subspace and full design space to obtain $\Delta_{m,\hat{\mathcal{D}}}$ and $\Delta_{m,\mathcal{D}}$ respectively, denoting the average generalization ability of models sampled from each respective subspace for a given metric m. For this experiment, we have the null hypothesis $H_\emptyset : \forall m : T\Delta_{m,\hat{\mathcal{D}}} \leq \Delta_{m,\mathcal{D}}$, signifying no improvement in transferability from the proposed subspace upon the full design space, and the experimental hypothesis $H_1 : \exists m : \Delta_{m,\hat{\mathcal{D}}} \leq \Delta_{m,\mathcal{D}}$ expecting the opposite.

Chapter 4

Evaluation

4.1 Methodology

Table 4.1: Statistics for the graph datasets used in our experiments. The number of graphs in each dataset is shown in the third column; for PPI and Twitch, we sample from the total number of graphs in the dataset which is enclosed in parentheses.

Dataset	Domain	# Graphs	# Nodes	# Edges	Degree	Density
Planetoid	Citation	3	8584.00	36102.00	3.71	0.0017
PPI	Biological	2 (20)	2495.50	66483.00	24.89	0.0200
Amazon	Product	2	10701.00	364942.00	33.44	0.0067
Twitch	Social Network	4 (6)	5059.25	124394.75	26.65	0.1515

Task Definition In this work, we focus on the task of *cross-graph inductive link prediction*. While existing work has studied inductive link prediction within one graph, we focus on a more difficult setting in which the target graph is *entirely novel*, forcing models to dynamically adapt to unseen nodes and potentially differing distributions. Formally, a given link prediction model ϕ following our framework detailed in Section 3.1 is trained on a source graph $\mathcal{G}_s$. We then compute a set of performance metrics for the test edges of $\mathcal{G}_s$, and then we repeat the process for an additional *target* graph $\mathcal{G}_t$. We then define the model generalization, Δ_m, for each metric $m \in \mathcal{M}$ as $m_s - m_t$ where m_s and m_t are values of the metric on the source and target graphs respectively for the model ϕ. The lower the value, the less of a decrease is shown in performance when applied to new data. While this task has been briefly examined (H. Wang et al., 2022), we go further by introducing the *in-domain* (ID) and *out-of-domain* (OOD) cross-graph ILP tasks. Existing work has examined the former task, where $\mathcal{G}_s$ and $\mathcal{G}_t$ are graphs from the same domain (such as two social networks), but we introduce the OOD setting where the two graphs may come from completely different domains (e.g., a citation network and a prod co-purchase network).

Datasets Throughout all of our experiments, we consider graphs from four datasets spanning different domains - Planetoid (paper citations) (Z. Yang, Cohen, & Salakhudinov, 2016), PPI (protein-protein interaction) (Zitnik & Leskovec, 2017), Amazon (product co-purchase) (Shchur, Mumme, Bojchevski, & Günnemann, 2018), and Twitch (social relationships) (Rozemberczki, Allen, & Sarkar, 2021). Using graphs from multiple domains helps ensure the generalizability of our results between graphs of different sizes, densities, and interaction dynamics. We provide a brief overview of the datasets in Table 4.1. The number of graphs in each dataset is shown in the third column; for PPI and Twitch, we sample from the total number of graphs in the dataset which is enclosed in parentheses. The following columns show the average number of nodes, number of edges, degree, and density found in the graphs used in our experiments for each domain.

Experimental Procedure We prepare the data for each trial by partitioning each graph's edges into message passing, training, validation, and testing sets. We reserve 10% of edges for validation and 20% for testing. We compute global encodings using *only the training edges* to prevent data leakage. We follow our proposed standardized model architecture comprised of global and local encodings, message passing layers, and subgraph pooling as described in Section 3.1. Each model is trained by minimizing the Binary Cross-Entropy (BCE) loss for a maximum of 2500 steps and evaluated on the complete set test. Our initial threshold for statistical significance is $\alpha = 0.05$ for all hypothesis tests.

4.2 Population-level Analysis

4.2.1 Aggregate Generalization Ability

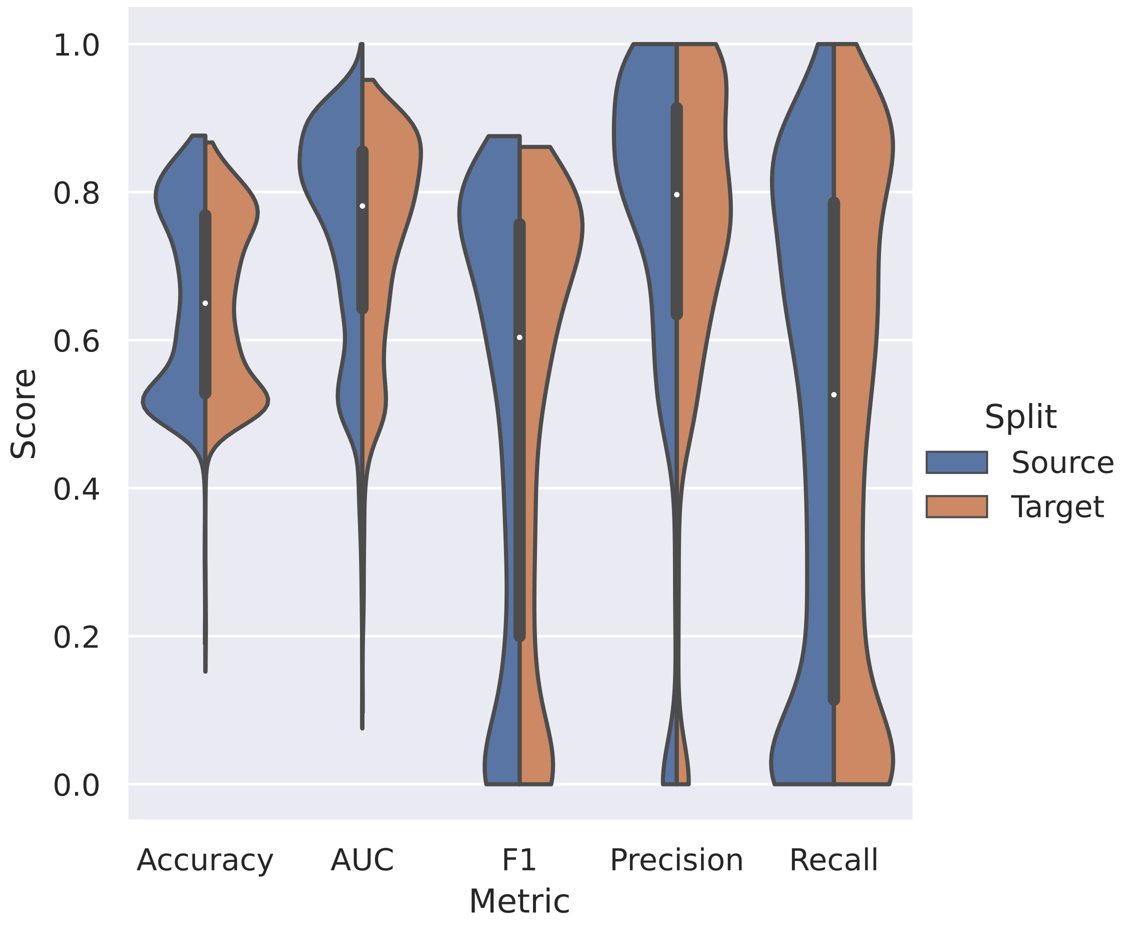


Figure 4.1: Distribution of performance metrics on the cross-graph link prediction task. We see that for all metrics except Recall, distributions skew slightly higher for the source dataset than the target dataset, indicating a drop-off in performance.

Our first objective Hypothesis 1.1 asserts that on the aggregate, ILP models will demonstrate a moderate decrease in performance when applied to new data. This hypothesis is tested by conducting a large number of trials in which various performance measures are obtained on the test edges for both the *source* and *target* graphs. In particular, we Controlled Random Search (CRS) for each of our proposed design dimensions, and aggregate the results to assess the overall behavior of ILP models. With $|S| = 25$ and each trial repeated across 3 different random seeds, we ultimately conduct $2,025$ trials, obtaining the AUC, Accuracy, F1, Precision, and Recall measures for the test edges on both the source and target graphs.

Table 4.2: Mean and standard deviation by metric on the source and target datasets. We find a small but statistically significance decrease in all AUC, Accuracy, and Precision when applying ILP models to unseen data.

Metric	Source	Target	Z	p
AUC	**0.7476** ($\pm$0.1469)	0.7321 ($\pm$0.1444)	13.5506	**2.45E-40**
Accuracy	**0.6567** ($\pm$0.1275)	0.6484 ($\pm$0.1182)	9.7255	**3.67E-22**
F1	**0.4973** ($\pm$0.3047)	0.4955 ($\pm$0.2937)	0.9092	1.82E-01
Precision	**0.7385** ($\pm$0.2653)	0.7245 ($\pm$0.2481)	5.4389	**3.02E-08**
Recall	0.4678 ($\pm$0.3306)	**0.476** ($\pm$0.3335)	-3.3605	1.00E+00

We use a paired-sample one sided t-test with the null hypothesis $H_\emptyset : \forall m : \bar{m}_s - \bar{m}_t \leq 0$, that the means of the source metrics are equal to or worse than their respective target metrics. The alternative hypothesis $H_1 : \exists m : \bar{m}_s - \bar{m}_t > 0$ states that there exists some metric for which performance *decreases* from the source graph to the target. Of course, evaluating these hypotheses repeatedly for multiple metrics increases our chance of Type-I error. We control for this using Bonferroni correction, which sets our threshold for statistical significance at $\alpha = \frac{0.05}{5} = 0.01$. Under this regime, **we find a statistically significant decrease in AUC, Accuracy, and Precision** when applying ILP models to the cross-graph link prediction task. However, **we do not find a significant difference in F1 or Recall**. In spite of this, we nonetheless **reject the null hypothesis** $H_\emptyset$ by showing the existence of a performance decrease in one or more metrics, with the relevant significance measures and test statistics shown in Table 4.2.

Figure 4.1 visualizes the distribution of each metric in both the source and target settings, showing that on average, mass was placed higher for the source metrics than their corresponding target metrics. This result indicates that practitioners may well experience difficulties when applying link prediction models to unseen data; however, it is still possible to achive comparable (albeit slightly diminished) performance. This result also signifies the need for models which are better able to generalize to unseen data to mitigate these difficulties. In addition, we can also observe that in both settings, models are able to achieve strong Precision but struggle with Recall, which may not be suitable for all applications. The existing literature has paid little attention to recall-oriented measures, which is seemingly reflected in these results and a potential direction for future work exploring the development of recall-optimized architectures.

4.2.2 Out-of-Distribution Analysis

Cross-Domain Generalization

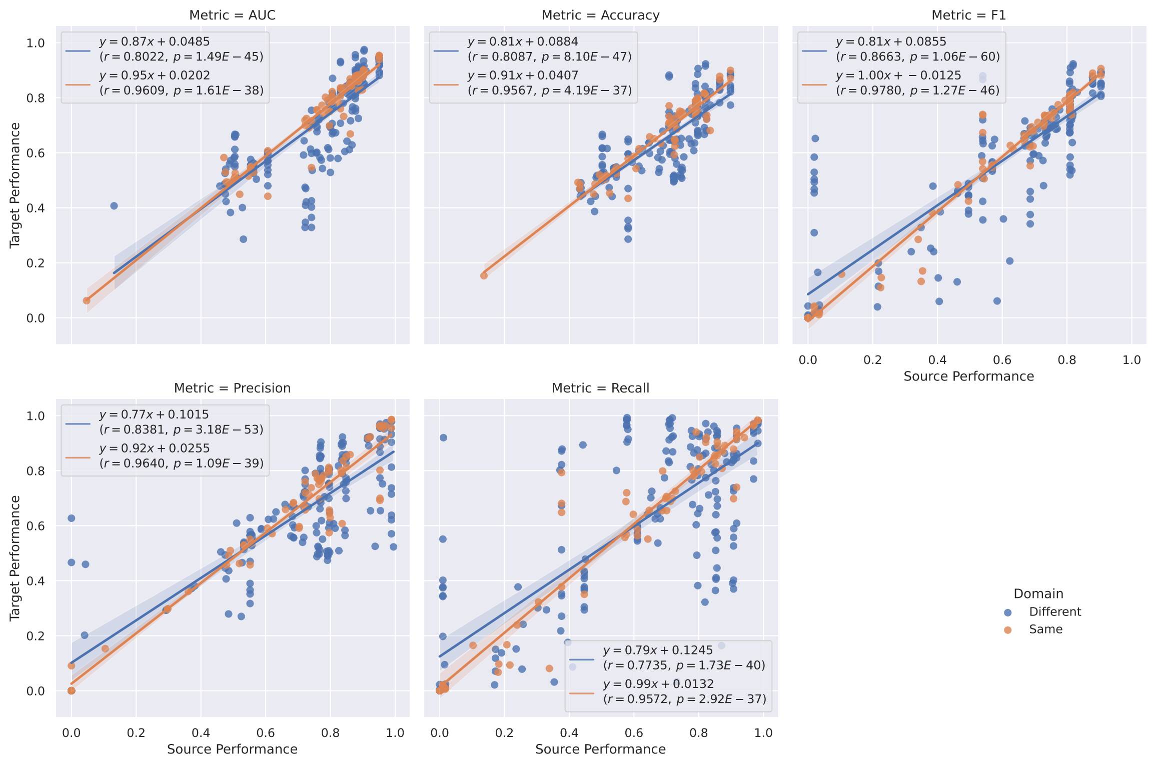


Figure 4.2: Within- and cross-domain relationships between performance on the source and target graphs. Orange points indicate a trial where the source and target datasets belong to the same domain, whereas blue points indicate different domains. We see a much stronger correlation between source and target performance when generalizing in-domain, demonstrated by the orange lines of best fit, than cross-domain, indicated by the blue lines of best fit.

Next, we wish to characterize the impact of the *underlying data* on the ability of a given model to generalize to unseen data. Our prior belief is that if a model is trained on and test on graphs from the same domain, the performance degredation will be smaller than that of a model trained and tested on graphs of different domains. This belief is founded on the notion that models will learn certain dynamics specific to particular domains, which may make it harder to generalize. We begin by examining the correlation between source and target performance for the in- and out-of-domain settings, shown by Figure 4.2. We observe a much stronger correlation between source and target performance when the respective graphs are of the same domain, while we see greater variability in the out-of-domain setting. In particular, we have slope ≤ 1 in almost all cases, signifying that performance is expected to decrease on any new graph; however, slopes are much greater in-domain, meaning that performance tends to degrade more out-of-domain.

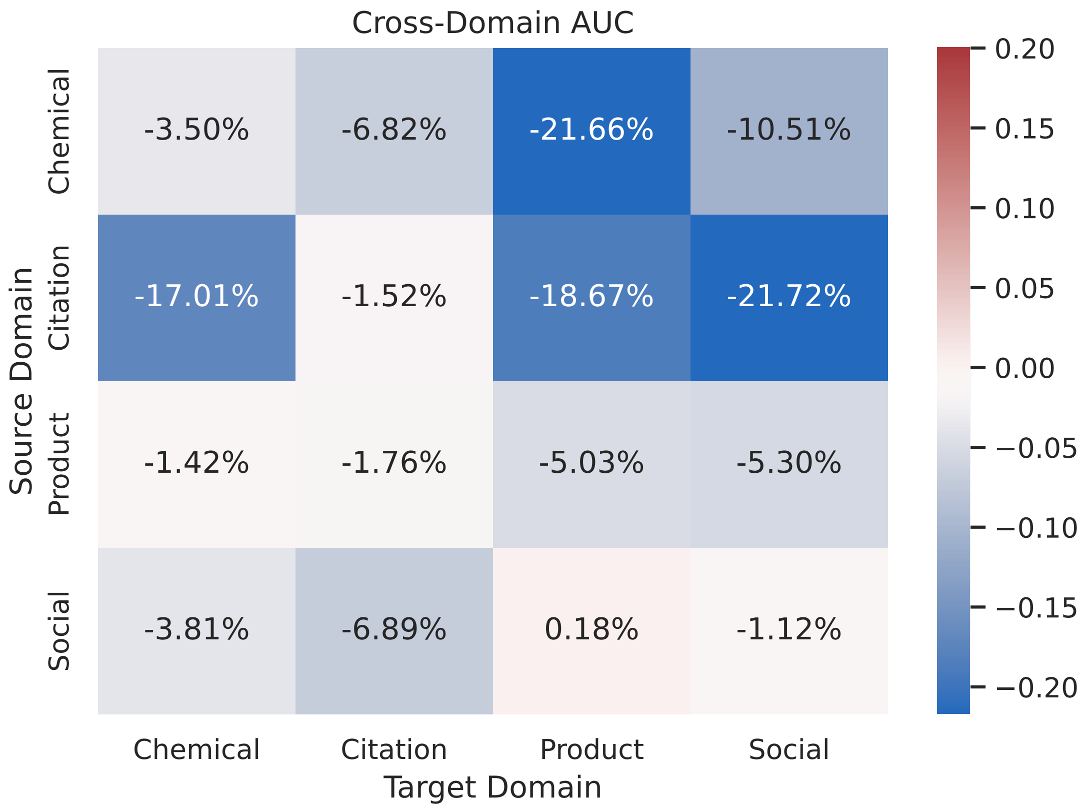


Figure 4.3: Generalization ability between domains for AUC. Each cell represents the average percentage increase in performance when training on a graph in the source domain (Y-axis) and testing on a graph in the target domain (X-axis). Warmer cells signify a greater increase in AUC (thus, better generalization) from graphs in the source domain to the target domain, while colder cells demonstrate a decrease in AUC (worse generalization/performance degradation).

We further explore this behavior by focusing on one particularly widely-used metric, AUC. We compute the percentage change in AUC from the source graph to the target graph and take the mean grouped by the domains of the respective graphs, shown by Figure 4.3. A warmer cell signifies a greater performance increase (or smaller decrease), and in turn, superior generalization between the two domains on average. We observe the strongest generalization occurs when training on social networks and transferring to product co-purchase, which is the only setting in which an *improvement* is seen. In contrast, the weakest generalization is from citation to social networks followed by chemical to product networks, both showing decreases greater than 21%. Note that these trends are not symmetric. We hypothesize that this behavior can be explained by certain characteristics of the respective datasets such as those shown in Table 4.1; for example, the citation networks are less dense and have significantly smaller average degrees on average than social networks which will lead to extracted subgraphs with far fewer edges, making it difficult for message passing layers to adapt to the target graph. In contrast, the product co-purchase networks have the highest average degrees and are almost six times as dense as citation networks (though still much less dense than the social networks), making it easier to generalize.

Table 4.3: The percentage change in performance between graphs from same and different domains, respectively. Though we observe better generalization better within-domain than cross-domain for all metrics, we are unable to reject the null hypothesis.

	Same	Different	T	p
Accuracy	**-0.0277** (±0.0787)	-0.0845 (±0.1772)	-2.4813	0.0138
AUC	**-0.0218** (±0.0809)	-0.1009 (±0.2425)	-2.5574	0.0111
F1	**-0.0905** (±0.432)	-0.1585 (±0.9147)	-0.5726	0.5675
Precision	**-0.0366** (±0.1767)	-0.1104 (±0.2766)	-1.998	0.0468
Recall	**-0.107** (±0.601)	-0.2901 (±1.8979)	-0.7579	0.4492

However, we must prove the significance of our observations. Our null hypothesis $H_\emptyset$ states that performance in the in-domain (ID) setting will equal to or worse than than out-of-domain (OOD) performance for all metrics, whereas the alternative hypothesis asserts that there will be one or more metrics for which the ID performance is better than OOD performance, demonstrating the difficulty of the latter setting. We use a t-test to estimate the significance of the difference between the group means for each metric, the results of which are shown in Table 4.3. Interestingly, while we do show that performance for the in-domain setting is superior to the OOD setting, we prove **unable to reject the null hypothesis** $H_\emptyset$. While a seemingly intuitive notion, we are ultimately unable to conclusively state that any datasets are more or less difficult to generalize between. This result has potentially interesting implications, as suggests the possibility that the generalization ability of ILP models may be agnostic to the underlying data, allowing greater exploration into topics such as pre-training.

Effect of Topological Similarity

While the above heuristic-based approach provides insight into out-of-distribution generalization at a coarse level by separating graphs into discrete domains, we also wish to examine generalization ability at a finer granularity. That is, we believe that a model will perform better on a test graph *similar* to the graph on which it was trained. Validating this proposition requires a continuous measure of similarity between graphs; luckily, this is covered by the well-studied area of graph kernels. A graph kernel function $k : \mathcal{G} \times \mathcal{G} \rightarrow \mathbb{R}$ computes a real-valued similarity score between a pair of graphs $G_s, G_t \in \mathcal{G}$. There are many graph kernels, each of which provides a unique perspective on quantifying topological similarity between two graphs, considering many factors like distributions over node and edge labels, random walk probabilities, graphlet counts, and so on. For our work, we choose the Pyramid Match kernel (Nikolentzos, Meladianos, & Vazirgiannis, 2017) as it matches our criteria of (1) not requiring node or edge labels and (2) operating efficiently enough to scale to larger graphs with many thousands of nodes.

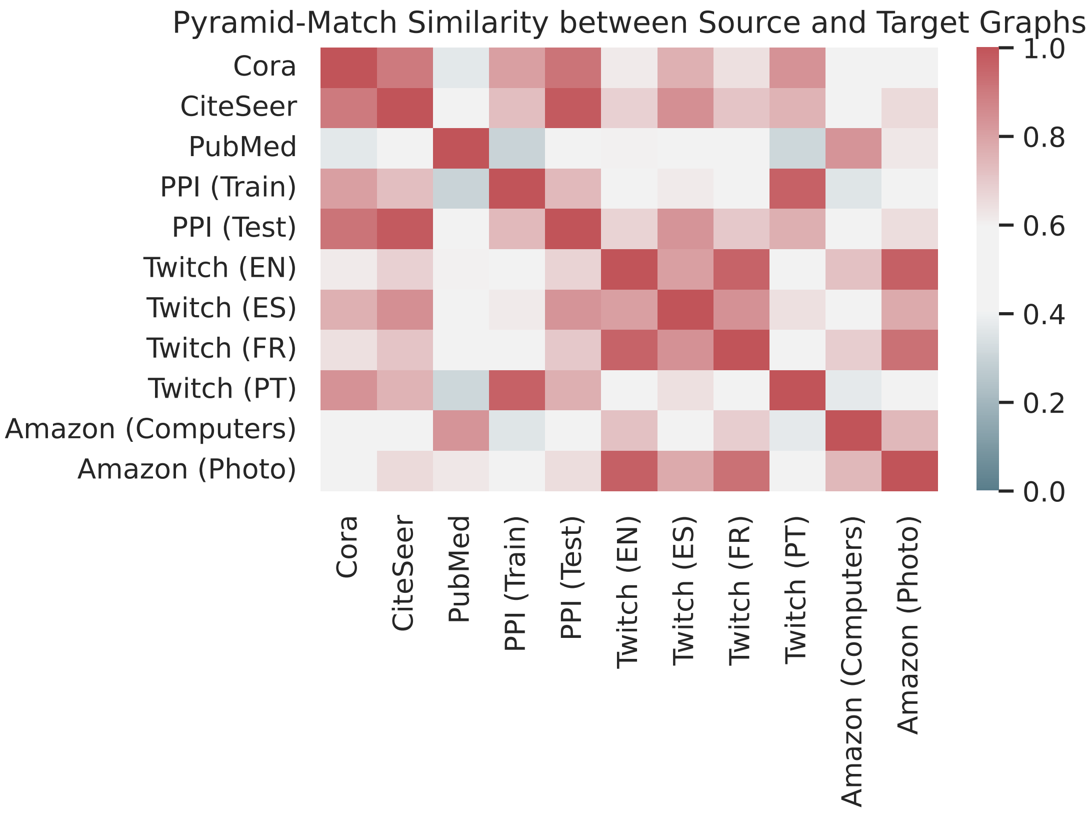


Figure 4.4: Pyramid Match graph similarity between all graphs used in our experiments. We observe that graphs from the same dataset are more similar on average.

We use the Pyramid Match kernel to compute the similarity across all datasets used in our experiments, shown in Figure 4.4. In general, we see greater similarity between graphs belonging to the same dataset, e.g. the two Amazon datasets are quite similar. There are notable exceptions, such as PubMed's dissimilarity to its fellow citation networks Cora and CiteSeer. Overall, the scores align well with our domain-based heuristic. We correlate these similarity scores to the percentage improvement in AUC scores as described in the previous section. Our null hypothesis $H_\emptyset$ states that the correlation between the similarity scores and the performance scores will be less than or equal to 0, while our alternative hypothesis states the contrary. However, we again prove **unable to reject the null hypothesis** $H_\emptyset$, observing a very weak correlation of just -0.06 ($p \leq 0.8623$). This result provides further evidence that the generalization dynamics of ILP models cannot be fully explained by differences in the underlying data.

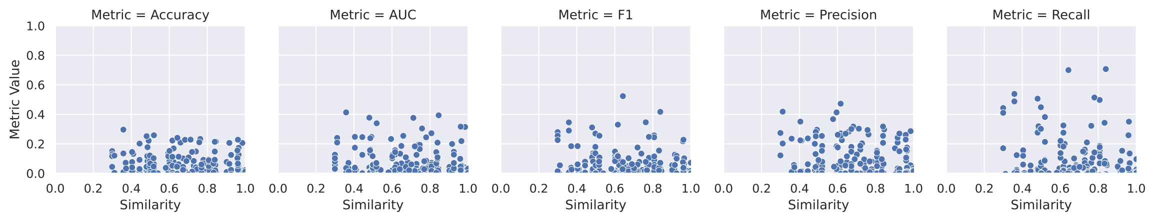


Figure 4.5: Correlation between graph similarity and model generalization. We are unable to reject the null hypothesis ($r = -0.06, p \leq 0.8623$).

4.3 Model-level Analysis

4.3.1 Dimension-level effects

Having considered behavior at the population level, we now shift focus to behavior at the model level. For Hypothesis 1.3, we wish to establish if any design dimensions make a statistically significant impact on generalization ability. Knowledge of such an effect would inform us that certain dimensions may be more important than others when identifying well-generalizing models, allowing us to allocate resources (such as compute, money, or time) more efficiently by focusing less on optimizing dimensions that do not significantly impact performance. For instance, in a compute-limited scenario, we may only have the resources to optimize one or two design dimensions, so the ability to answer a question such as *“is optimizing my choice in learning rate worthwhile given the resources required to do so?”* is crucially important, making it feasible for researchers and practitioners to conduct design space explorations and find optimal models within the constraints of their resources. Note that this does not speak to the importance of each individual option within the dimension itself, but rather the importance of making an optimal selection for a given dimension as a whole.

Table 4.4: The statistical significance of the effect of each design dimension on model generalization across all metrics, which we define as the difference between a given performance metric on the source and target graphs. We identify 3 design dimensions with a significant impact on generalization for at least one metric.

Design Dimension	AUC	Accuracy	F1	Precision	Recall
Global Encoding	**4.70E-06** (F=5.9763)	2.46E-03 (F=3.4352)	1.48E-01 (F=1.5890)	3.36E-02 (F=2.2994)	7.13E-02 (F=1.9490)
Hidden Channels	8.04E-01 (F=0.2179)	8.15E-01 (F=0.2054)	5.59E-01 (F=0.5833)	1.00E-01 (F=2.3268)	7.51E-01 (F=0.2866)
Learning Rate	8.21E-02 (F=2.5283)	6.32E-02 (F=2.7953)	5.61E-02 (F=2.9185)	1.67E-01 (F=1.8014)	2.07E-01 (F=1.5849)
Local Encoding	**5.51E-08** (F=12.9745)	**1.57E-10** (F=17.5836)	**7.55E-07** (F=10.9647)	**1.43E-13** (F=23.3058)	2.63E-02 (F=3.1229)
Message Passing Layers	3.08E-01 (F=1.1833)	1.46E-01 (F=1.9409)	2.89E-02 (F=3.5995)	3.47E-01 (F=1.0632)	2.04E-02 (F=3.9627)
Message Passing Network	**1.87E-19** (F=26.6697)	**1.80E-09** (F=12.3948)	**7.42E-12** (F=15.6697)	1.05E-01 (F=1.9303)	**7.19E-06** (F=7.5693)
Optimizer	4.79E-01 (F=0.5031)	7.04E-01 (F=0.1449)	4.16E-01 (F=0.6657)	5.74E-01 (F=0.3180)	5.42E-01 (F=0.3740)

Answering such a question is complex, but we can use Controlled Random Search (CRS) to effectively design a set of experiments allowing us to do so. For each design dimension $d \in \mathcal{D}$, we use CRS to sample 25 configuration groups and repeat each across 3 random seeds, giving us $75 \cdot |d|$ trials for the design dimension d. This gives us a set of trials $t_{i,j}$, $1 \leq j \leq 75$ for each $d_i \in d$, where we know that the only difference between trials $t_{i,j}$ and $t_{k,j}$ are the selected values i and j for the design dimension of interest d. All other parameters - including model architecture, training hyperparameters, and datasets - are held fixed. For each trial $t_{i,j}$, we compute the difference between each metric $\Delta_{i,j,m} = m_{i,j,s} - m_{i,j,t}$ as the generalization score(s), where $m_{i,j,\cdot}$ is the value of a metric m for a given trial $t_{i,j}$ on the source (s) or target (t) graphs. Taking the mean $\mu_{i,m} = \frac{1}{75}\sum_{j=1}^{75} \Delta_{i,j,m}$ then tells us the mean generalization ability for a given member of the design dimension $i \in d$ and metric m. This allows us to interpret $\mu_{i,m} - \mu_{j,m}$ for some $i, j \in d$ as the *difference in mean generalization ability between the two members of the design dimension, holding all other parameters constant.*

To interpret these pairwise differences in group means, we use a one-way ANOVA with Bonferroni correction ($p = \frac{0.05}{|\mathcal{D}||\mathcal{M}|} = \frac{0.05}{7 \cdot 5} = \frac{0.05}{35}$). Our null hypothesis for each design dimension and metric $d, m \in \mathcal{D} \times \mathcal{M}$ is that $H_\emptyset : \forall i, j \in d : \; \mu_{i,m} - \mu_{j,m} = 0$, which assumes that there are no two members of the design dimension with significantly different generalization abilities. Our alternative hypothesis $H_1 : \exists i, j \in d : |\mu_{i,m} - \mu_{j,m}| > 0$ states that there exists at least one pair of choices within the design dimension with a statistically significant difference in their generalization abilities. We conduct this analysis across all 7 of our design dimensions, executing 2025 total trials. The results are presented in Table 4.7, where we observe a statistically significant effect on generalization ability from 3 design dimensions - Global Encoding, Local Encoding, and Message Passing Network - for at least one metric. This result confirms Hypothesis 1.3, verifying that **particular design dimensions have a statistically significant effect on ILP models' generalization ability**.

4.3.2 Performance by Design Dimension

Having verified the existence of significant effects on generalization *between* the different design dimensions, we now examine the abilities of all options *Within* each design dimension. For each metric m and design dimension d, we compute the mean rank across all sample configurations as described in Figure 3.2 for both the source and target graphs. A lower value indicates a better ranking on average for a given method within a particular design dimension. This section will visualize these rankings across all design dimension and metrics to comprehensively capture the strengths and weakness of each.

Global encoding Figure 4.6 shows the surprising result that none of the global encoding methods are as effective in this setting as a simple degree-based encoding. The degree-based GE outperforms all others in all metrics on both source and target graphs. Overall, the top three best-performing methods - degree, random node initialization, and random walk - can also arguably be considered the simplest methods, suggesting that high-powered positional encoding methods do not demonstrably improve performance relative to more lightweight alternatives, and are sometimes even worse than not using a global encoding at all.

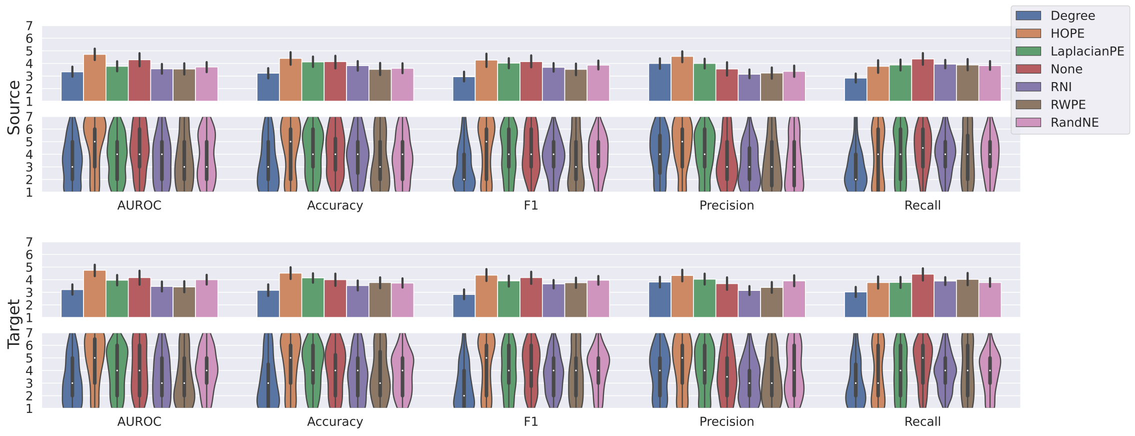


Figure 4.6: Ranking analysis of global encodings across all metrics. Metrics are along the X axis, and bar colors correspond to choices for the design dimension. The top row corresponds to performance on the source graph, while the bottom corresponds to the target graph. The mean and distribution of rankings are shown in the top and bottom halves of each row. A lower average rank indicates superior performance.

Local encoding Among local encodings, it is clear that Double Radius Node Labeling (DRNL) and Distance Encoding Plus (DE+) are markedly superior to Distance Encoding (DE+), as shown by Figure 4.7. Interestingly, there is a pronounced difference between all three options and `None`, indicating that using a local encoding is vital for inductive link prediction.

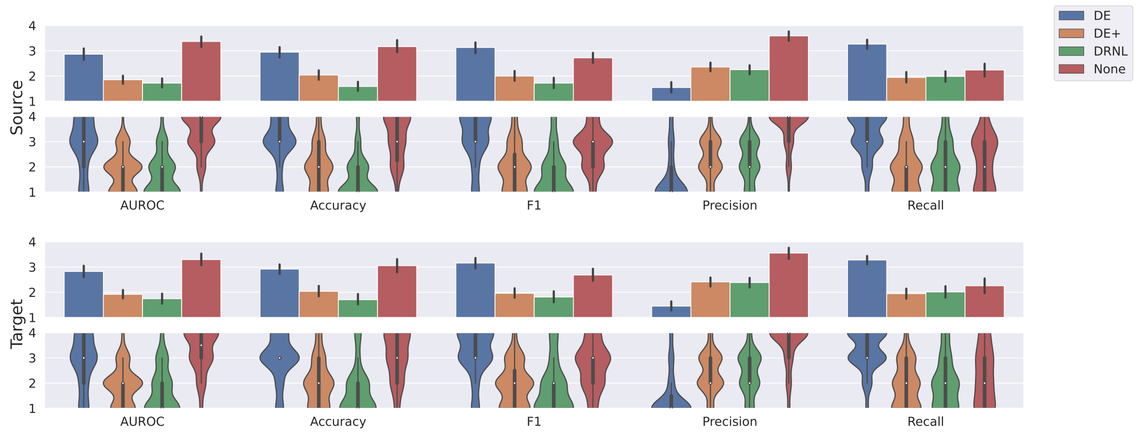


Figure 4.7: Ranking analysis of local encodings across all metrics. Metrics are along the X axis, and bar colors correspond to choices for the design dimension. The top row corresponds to performance on the source graph, while the bottom corresponds to the target graph. The mean and distribution of rankings are shown in the top and bottom halves of each row. A lower average rank indicates superior performance.

Message passing network It is shown by 4.8 that the Graph Convolutional Network (GCN) consistently outperforms the alternatives in terms of AUC, accuracy, F1, and recall. However, GraphSAGE produces the highest precision, providing a good example of the importance of considering performance across multiple metrics. These findings suggest that most practitioners in general should opt for the GCN, but that those focused on tasks requiring high precision should actually choose GraphSAGE instead.

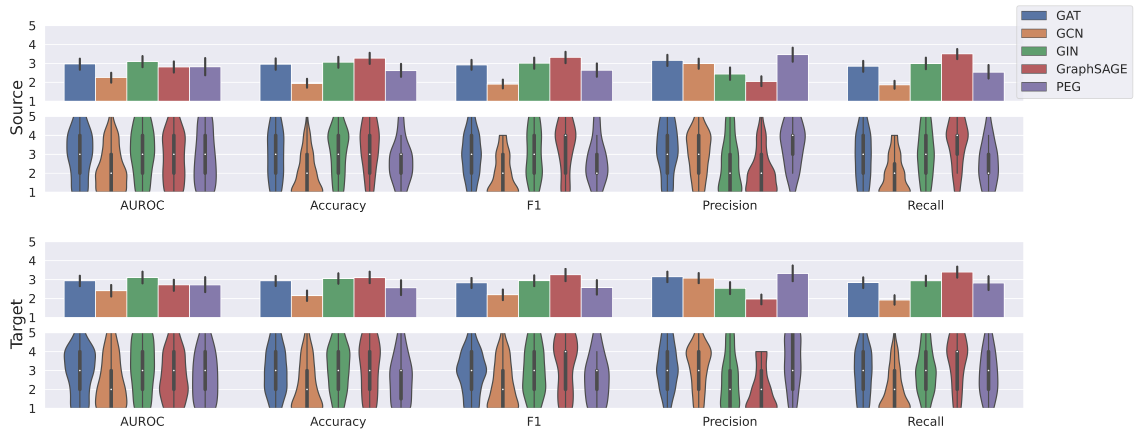


Figure 4.8: Ranking analysis of message passing networks across all metrics. Metrics are along the X axis, and bar colors correspond to choices for the design dimension. The top row corresponds to performance on the source graph, while the bottom corresponds to the target graph. The mean and distribution of rankings are shown in the top and bottom halves of each row. A lower average rank indicates superior performance.

Message passing layers When considering the number of message passing layers used, a *less is more* effect is found. Figure 4.9 shows that in most cases, it seems that one message passing layer is sufficient, if not outright superior, and in almost every case using three message passing layers hampers performance. We believe that this is likely the effect of the well-studied oversmoothing phenomenon in message passing networks, in which node embeddings tend to converge to a similar point as the number of message passing rounds increases, ultimately making it difficult to distinguish positive and negative subgraphs.

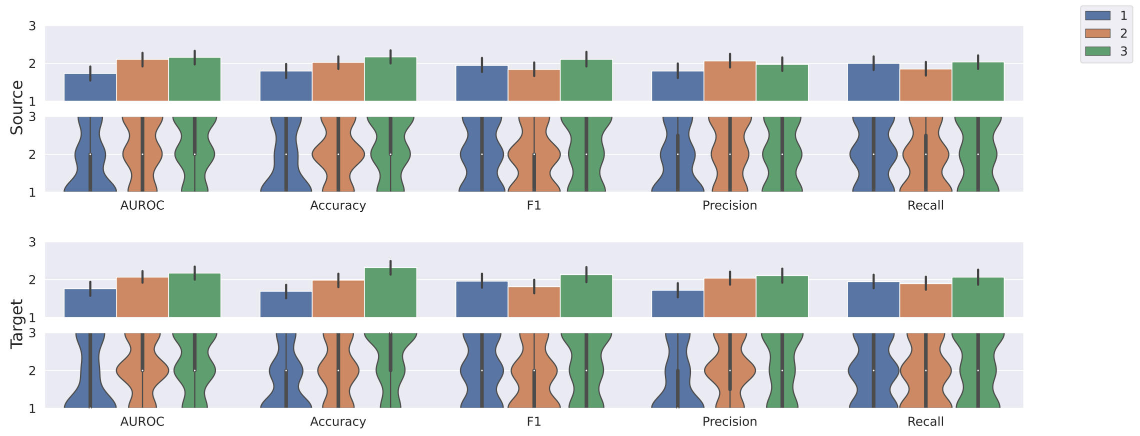


Figure 4.9: Ranking analysis of message passing layers across all metrics. Metrics are along the X axis, and bar colors correspond to choices for the design dimension. The top row corresponds to performance on the source graph, while the bottom corresponds to the target graph. The mean and distribution of rankings are shown in the top and bottom halves of each row. A lower average rank indicates superior performance.

Hidden channels Another interesting finding comes in the number of hidden channels used to embed nodes within the model. Similarly to the number of message passing layers, Figure 4.10 shows the same *less is more* effect is present in the number of hidden channels. In most cases, either 32 or 64 channels is optimal for best performance, with 128 channels consistently leading to poor performance. This finding can help practitioners reduce training time and increase their model efficiency by avoiding needlessly large models.

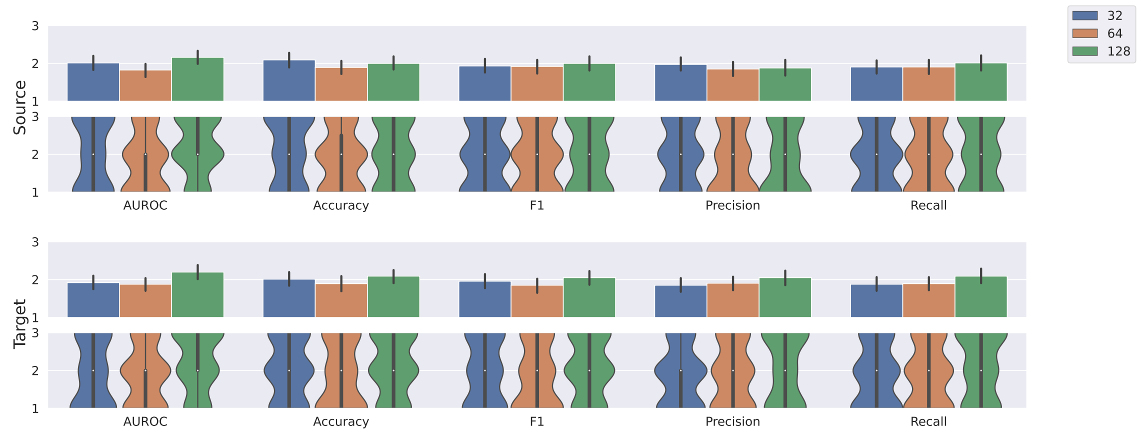


Figure 4.10: Ranking analysis of hidden channels across all metrics. Metrics are along the X axis, and bar colors correspond to choices for the design dimension. The top row corresponds to performance on the source graph, while the bottom corresponds to the target graph. The mean and distribution of rankings are shown in the top and bottom halves of each row. A lower average rank indicates superior performance.

Optimizer Figure 4.11 paints a somewhat unclear picture of which optimizer is best for the inductive link prediction task. Adam is superior in AUC and precision, while SGD is stronger in F1 and recall, and both are comparable in terms of accuracy. This aligns with findings from previous work that Adam is generally comparable or superior to SGD in a naive setting, though SGD can be better when tuned. Though the outright best choice is unclear, our findings help guide practitioners by demonstrating the nuanced differences between the two on a per-metric basis.

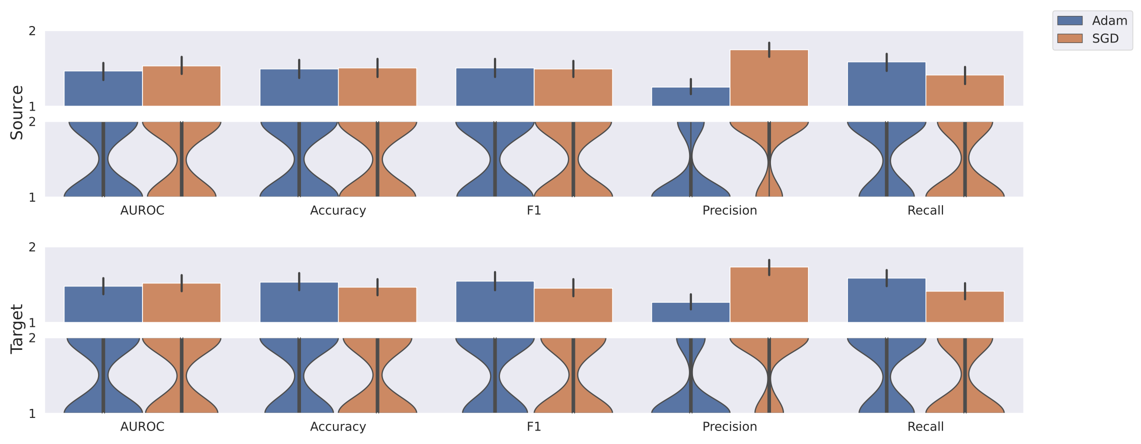


Figure 4.11: Ranking analysis of optimizers across all metrics. Metrics are along the X axis, and bar colors correspond to choices for the design dimension. The top row corresponds to performance on the source graph, while the bottom corresponds to the target graph. The mean and distribution of rankings are shown in the top and bottom halves of each row. A lower average rank indicates superior performance.

Learning rate Finally, it is seen in Figure 4.12 that a lower learning rate of 0.0001 is generally better for performance. In addition, we again observe a difference precision, which is increased with a slightly higher learning rate of 0.001.

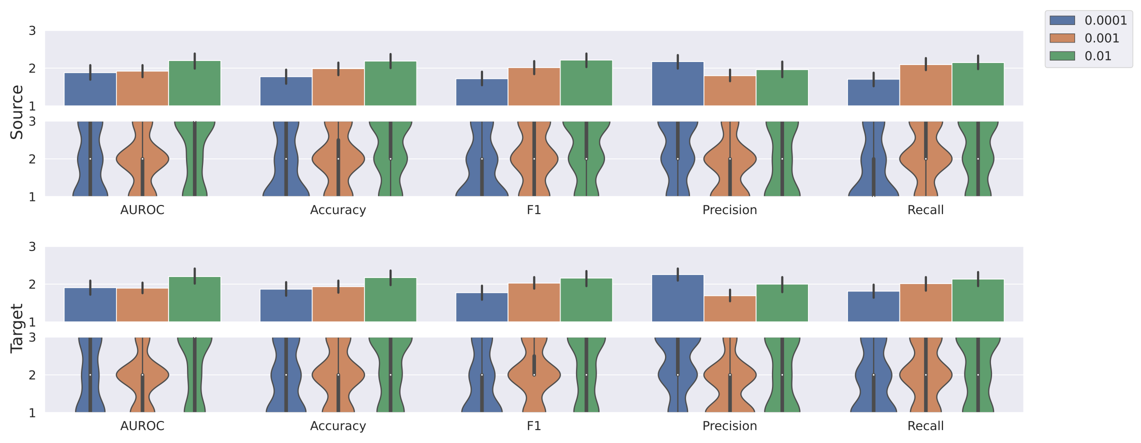


Figure 4.12: Ranking analysis of learning rate across all metrics. Metrics are along the X axis, and bar colors correspond to choices for the design dimension. The top row corresponds to performance on the source graph, while the bottom corresponds to the target graph. The mean and distribution of rankings are shown in the top and bottom halves of each row. A lower average rank indicates superior performance.

4.3.3 Reduced Design Space

Table 4.5: Optimal design choices across all metrics for source and target datasets.

	Global encoding	Local encoding	Message passing network	Message passing layers	Hidden channels	Optimizer	Learning rate
Source AUC	Degree	DRNL	GCN	1	64	Adam	0.0001
Source Accuracy	Degree	DRNL	GCN	1	64	Adam	0.0001
Source F1	Degree	DRNL	GCN	2	64	SGD	0.0001
Source Precision	RNI	DE	GraphSAGE	1	64	Adam	0.001
Source Recall	Degree	DE+	GCN	2	32	SGD	0.0001
Target AUC	Degree	DRNL	GCN	1	64	Adam	0.001
Target Accuracy	Degree	DRNL	GCN	1	64	SGD	0.0001
Target F1	Degree	DRNL	GCN	2	64	SGD	0.0001
Target Precision	RNI	DE	GraphSAGE	1	32	Adam	0.001
Target Recall	Degree	DE+	GCN	2	32	SGD	0.0001

A key application of the above results is to *reduce* the design space. The number of experiments required to evaluate the full design space is prohibitive (if not effectively intractable with finite resources), but even with Controlled Random Search, each design dimension must still be evaluated. This can make it quite costly to run a sufficient number of trials to obtain a strong-performing model, and this cost will increase combinatorially for future work which may introduce new design dimensions beyond the scope of the current study. As such, it would be extremely valuable to eliminate the choices from each dimension which we know to not be worth exploring, greatly reducing the number of trials required to find well-performing models. Concretely, we seek a subspace $\mathcal{D}\prime \subset \mathcal{D}$ such that the average performance of models in $\mathcal{D}\prime$ is comparable or superior to those in $\mathcal{D}$. To find such a subspace, we examine the best-performing choices for each design dimension $d \in \mathcal{D}$ across all metrics on both the source and target graphs, shown in Table 4.5. Each row shows the best-performing choices across all design dimensions for a given metric. We greedily select the *column space* of this table as our reduced design space; that is, the set of choices for each dimension consist of the best-performing choices for each metric.

Table 4.6: A reduction of our proposed full design space obtained by greedily selecting the best-performing choices for each dimension.

Design Dimension	Choices
Positional encoding	Degree, RNI
Structural encoding	DE, DE+, DRNL
Message passing network	GCN, GraphSAGE
Message passing layers	1, 2
Hidden channels	32, 64
Optimizer	Adam, SGD
Learning rate	0.0001, 0.001

The resulting design space $\mathcal{D}\prime$ is shown in Table 4.6. However, for this space to provide genuine utility as a high-quality starting point which can accelerate future research, it must be shown to produce *comparable or superior performance* to the full space of models. This is the goal of our third research objective, Hypothesis 1.3. Our null hypothesis $H_\emptyset : \forall m : \bar{m}_{\mathcal{D}} = \bar{m}_{\mathcal{D}\prime}$ states that the mean of each metric will be identical between the full and pruned design spaces, while our experimental hypothesis $H_1 : \exists m : \bar{m}_{\mathcal{D}} - \bar{m}_{\mathcal{D}\prime} > 0$ states that there will be one or more metrics whose means are higher for the reduced space than the full space, indicating the superiority of the full space. We investigate our hypotheses by using classical random sampling to conduct 100 trials sampled from both the full and reduced spaces, and computing performance metrics for the target graph. We determine the significance of our results using a one-way ANOVA with Bonferroni correction. Unfortunately, while we do observe that the pruned design space outperforms the full space in several metrics, we are **unable to reject the null hypothesis** $H_{\emptyset}$, as the test statistics are insufficient to demonstrate statistical significance as shown by Table 4.7. We hypothesize that this is a result of our selection procedure in determining our reduced space, as it may have been too lenient in the options it preserved; as such, identifying an optimal procedure for selecting a reduced design space could be a valuable direction for future work.

Table 4.7: Performance comparison between models sampled from the full and reduced design spaces on the target graph.

	Full	Reduced	F	p
Accuracy	**0.0325** (±0.0667)	0.0358 (±0.067)	0.1462	0.7026
AUC	**0.0504** (±0.0967)	0.0584 (±0.1168)	0.3157	0.5749
F1	0.0246 (±0.1523)	**0.0177** (±0.124)	0.1153	0.7345
Precision	**0.0375** (±0.1313)	0.0699 (±0.1503)	2.71	0.1013
Recall	0.0233 (±0.226)	**-0.0189** (±0.1955)	1.9779	0.1612

Conclusion

Summary of Contributions In conclusion, we perform the first (to the best of our knowledge) exhaustive evaluation of GNN-based methods for inductive link prediction, with a particular focus on generalization ability. We first provided a comprehensive overview of existing work on graph representation learning and inductive link prediction in Section 2. In Section 3 we propose a standardized framework for the inductive link prediction task, and detailed the experiments necessary to investigate our hypotheses.

We presented the results of our work in Section 4, beginning by establishing a statistically significant decrease in performance when applying ILP models to unseen graphs,demonstrating the difficulty of the cross-graph ILP task and verifying our first hypothesis. We then analyzed the effects of the underlying data on this performance decrease, and found the surprising result that shifting domains from the source graph to the target could not be confirmed as responsible for the crease in performance. This counterintuitive result has interesting and potentially significant applications, as it suggests that if the true cause for performance degradation is identified, it may be possible to transfer ILP models between arbitrary, potentially very different domains of data. Next, we focused on trends in individual components of our model framework. We demonstrated a statistically significant difference between said components, indicating that certain components of ILP models are more important to generalization than others. We also provided an in-depth ranking analysis of the candidate methods for each model component, yielding illuminating insights such as the importance of local encodings and a *less-is-more* effect where larger models were not always necessarily better, making it possible to both improve performance *and* conserve computational resources. Finally, we attempted to demonstrate the existence of an optimal reduced design space in order to accelerate future work in the area. However, we were unfortunately unable to do so; we hypothesize this is due to our simplistic and limited pruning method, which can be improved in future work.

In summary, our work sheds light on multiple important behavioral patterns of models for inductive link prediction. We establish the difficulty of the task and identify important considerations for maximizing generalization ability. We hope that the present work provides a helpful starting point for future work in the area, and that our framework and insights will help reduce the barrier to entry for those new to the space.

Future Work We believe that the present work sets the foundation for a multitude of future research. Firstly, our difficulties in identifying a reduced design space point to the need for an optimal method in doing so. Our approach was a simple heuristic that selected all options from all design dimensions which ranked best in at least one metric. This approach has clear shortcomings, as options which perform very well in one metric but poorly in others may not be ideal relative to options which perform moderately well across all metrics. Future work could consider methods similar to various hyperparameter tuning algorithms for automatically identifying reduced design spaces to alleviate the need for such simplistic heuristics.

In addition, the task of inductive link prediction is highly relevant and applicable to many real world scenarios. In particular, recommendation systems could benefit from high-quality inductive methods, as many recommendation tasks are *privacy sensitive*. In scenarios where data must be isolated between users, inductive models could be combined with federated learning approaches to obtain high-performance, privacy-preserving recommendation models. Even in scenarios where privacy is not of concern, inductive models still have many benefits. By removing the need to transductively represent each user and item with their own unique embedding, inductive models are many orders of magnitude more efficient than their transductive counterparts, a vital consideration when operating on millions of users and items. Inductive models also bear the benefit of naturally generalizing to unseen data, whereas transductive models need to be retrained frequently to handle new users and items. In all, understanding the behavior of inductive models for link prediction and identifying optimal design choices can help improve the viability, performance, and efficiency in recommendation systems and related tasks.

References

Abbas, K., Abbasi, A., Dong, S., Niu, L., Yu, L., Chen, B., ... Hasan, Q. (2021). Application of network link prediction in drug discovery. *BMC Bioinformatics*, *22*.

Abboud, R., Ceylan.Ismail .Ilkan, Grohe, M., & Lukasiewicz, T. (2020). The surprising power of graph neural networks with random node initialization. *ArXiv*, *abs/2010.01179*.

Adamic, L. A., & Adar, E. (2003). Friends and neighbors on the web. *Soc. Networks*, *25*, 211–230.

Arvind, V., Köbler, J., Rattan, G., & Verbitsky, O. (2015). On the power of color refinement. In *International symposium on fundamentals of computation theory.*

Backstrom, L., & Leskovec, J. (2010). Supervised random walks: Predicting and recommending links in social networks. In *Web search and data mining.*

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. *CoRR*, *abs/1409.0473*.

Balazevic, I., Allen, C., & Hospedales, T. M. (2019). TuckER: Tensor factorization for knowledge graph completion. *ArXiv*, *abs/1901.09590*.

Balcilar, M., Héroux, P., Gaüzère, B., Vasseur, P., Adam, S., & Honeine, P. (2021). Breaking the limits of message passing graph neural networks. In *International conference on machine learning.*

Bengio, Y., Courville, A. C., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*, 1798–1828.

Berg, R. van den, Kipf, T. N., & Welling, M. (2017). Graph convolutional matrix completion. *arXiv Preprint arXiv:1706.02263.*

Bordes, A., Usunier, N., García-Durán, A., Weston, J., & Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In *NIPS.*

Bronstein, M. M., Bruna, J., Cohen, T., & Veličković, P. (2021). Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *ArXiv*, *abs/2104.13478*.

Chen, Y., Mishra, P., Franceschi, L., Minervini, P., Stenetorp, P., & Riedel, S. (2022). ReFactorGNNs: Revisiting factorisation-based models from a message-passing perspective. *ArXiv*, *abs/2207.09980*.

Colizza, V., Barrat, A., Barthelemy, M., & Vespignani, A. (2005). Prediction and predictability of global epidemics: The role of the airline transportation network. *Bulletin of the American Physical Society*, 2015.

Cui, Z., Henrickson, K. C., Ke, R., & Wang, Y. (2018). Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting. *IEEE Transactions on Intelligent Transportation Systems*, *21*, 4883–4894.

Daud, N. N., Ab Hamid, S. H., Saadoon, M., Sahran, F., & Anuar, N. B. (2020). Applications of link prediction in social networks: A review. *Journal of Network and Computer Applications*, *166*, 102716.

Degraeve, V., Vandewiele, G., Ongenae, F., & Hoecke, S. van. (2022). R-GCN: The r could stand for random. *ArXiv*, *abs/2203.02424*.

Derrow-Pinion, A., She, J., Wong, D., Lange, O., Hester, T., Perez, L., ... Velickovic, P. (2021). ETA prediction with graph neural networks in google maps. In *Proceedings of the 30th ACM international conference on information &amp; knowledge management* (pp. 3767–3776). New York, NY, USA: Association for Computing Machinery. http://doi.org/10.1145/3459637.3481916

Dwivedi, V. P., Joshi, C. K., Luu, A. T., Laurent, T., Bengio, Y., & Bresson, X. (2020). Benchmarking graph neural networks. *arXiv Preprint arXiv:2003.00982*.

Dwivedi, V. P., Luu, A. T., Laurent, T., Bengio, Y., & Bresson, X. (2022). Graph neural networks with learnable structural and positional representations. In *International conference on learning representations*. Retrieved from `https://openreview.net/forum?id=wTTjnvGphYj`

Egressy, B., & Wattenhofer, R. (2022). Graph neural networks with precomputed node features. *ArXiv*, *abs/2206.00637*.

Fang, X., Huang, J., Wang, F., Zeng, L., Liang, H., & Wang, H. (2020). ConSTGAT: Contextual spatial-temporal graph attention network for travel time estimation at baidu maps. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Feng, J., Chen, Y., Li, F., Sarkar, A., & Zhang, M. (2022). How powerful are k-hop message passing graph neural networks. In A. H. Oh, A. Agarwal, D. Belgrave, & K. Cho (Eds.), *Advances in neural information processing systems*. Retrieved from `https://openreview.net/forum?id=nN3aVRQsxGd`

Galkin, M., Berrendorf, M., & Hoyt, C. T. (2022). An open challenge for inductive link prediction on knowledge graphs. *ArXiv*, *abs/2203.01520*.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *Proceedings of the 34th international conference on machine learning - volume 70* (pp. 1263–1272). Sydney, NSW, Australia: JMLR.org.

Grover, A., & Leskovec, J. (2016). node2vec: Scalable feature learning for networks. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Hamaguchi, T., Oiwa, H., Shimbo, M., & Matsumoto, Y. (2017). Knowledge transfer for out-of-knowledge-base entities : A graph neural network approach. *ArXiv*, *abs/1706.05674*.

Hamilton, W. L., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. In *NIPS*.

Hasan, M. A., Chaoji, V., Salem, S., & Zaki, M. J. (2006). Link prediction using supervised learning. In.

He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., & Wang, M. (2020). LightGCN: Simplifying and powering graph convolution network for recommendation. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

He, Y., Wang, Z., Zhang, P., Tu, Z., & Ren, Z. (2020). VN network: Embedding newly emerging entities with virtual neighbors. *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*.

Huang, Z., Li, X., & Chen, H. (2005). Link prediction approach to collaborative filtering. In *Proceedings of the 5th ACM/IEEE-CS joint conference on digital libraries* (pp. 141–142).

Islam, K., Aridhi, S., & Smaïl-Tabbone, M. (2020). A comparative study of similarity-based and GNN-based link prediction approaches. *ArXiv*, *abs/2008.08879*.

Jeh, G., & Widom, J. (2002). SimRank: A measure of structural-context similarity. *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Jiang, W. (2022). Bike sharing usage prediction with deep learning: A survey. *Neural Computing & Applications*, *34*, 15369–15385.

Jiang, W., & Luo, J. (2021). Graph neural network for traffic forecasting: A survey. *Expert Syst. Appl.*, *207*, 117921.

Kamp, C., Moslonka-Lefebvre, M., & Alizon, S. (2013). Epidemic spread on weighted networks. *PLoS Computational Biology*, *9*.

Kazemi, S. M., & Poole, D. L. (2018). SimplE embedding for link prediction in knowledge graphs. *ArXiv*, *abs/1802.04868*.

Kiefer, S., Schweitzer, P., & Selman, E. (2015). Graphs identified by logics with counting. In *International symposium on mathematical foundations of computer science.*

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In Y. Bengio & Y. LeCun (Eds.), *3rd international conference on learning representations, ICLR 2015, san diego, CA, USA, may 7-9, 2015, conference track proceedings.* Retrieved from http://arxiv.org/abs/1412.6980

Kipf, T., & Welling, M. (2016). Variational graph auto-encoders. *ArXiv*, *abs/1611.07308*.

Kipf, T., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. *ArXiv*, *abs/1609.02907*.

Kreuzer, D., Beaini, D., Hamilton, W., Létourneau, V., & Tossou, P. (2021). Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, *34*, 21618–21629.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems* (Vol. 25). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

Leman, A., & Weisfeiler, B. (1968). A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsiya*, *2*(9), 12–16.

Leskovec, J., Huttenlocher, D. P., & Kleinberg, J. M. (2010). Predicting positive and negative links in online social networks. *ArXiv*, *abs/1003.2429*.

Li, P., Wang, Y., Wang, H., & Leskovec, J. (2020). Distance encoding: Design provably more powerful neural networks for graph representation learning. *Advances in Neural Information Processing Systems*, *33*, 4465–4478.

Liben-Nowell, D., & Kleinberg, J. M. (2007). The link-prediction problem for social networks. *J. Assoc. Inf. Sci. Technol.*, *58*, 1019–1031.

Lu, L., & Zhou, T. (2010). Link prediction in complex networks: A survey. *ArXiv*, *abs/1010.0725*.

Ma, Y., Guo, Z., Ren, Z., Tang, J., & Yin, D. (2020). Streaming graph neural networks. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval* (pp. 719–728). New York, NY, USA: Association for Computing Machinery. http://doi.org/10.1145/3397271.3401092

MacLean, F. (2021). Knowledge graphs and their applications in drug discovery. *Expert Opinion on Drug Discovery*, *16*(9), 1057–1069.

Maron, H., Ben-Hamu, H., Serviansky, H., & Lipman, Y. (2019). Provably powerful graph networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 32). Curran Associates, Inc. Retrieved from `https://proceedings.neurips.cc/paper_files/paper/2019/file/bb04af0f7ecaee4aae62035497da1387-Paper.pdf`

Mikolov, T., Chen, K., Corrado, G. S., & Dean, J. (2013). Efficient estimation of word representations in vector space. In *ICLR*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *ArXiv*, *abs/1310.4546*.

Morris, C., Fey, M., & Kriege, N. (2021). The power of the weisfeiler-leman algorithm for machine learning with graphs. In Z.-H. Zhou (Ed.), *Proceedings of the thirtieth international joint conference on artificial intelligence, IJCAI-21* (pp. 4543–4550). International Joint Conferences on Artificial Intelligence Organization. http://doi.org/10.24963/ijcai.2021/618

Morris, C., Lipman, Y., Maron, H., Rieck, B., Kriege, N. M., Grohe, M., … Borgwardt, K. (2021). Weisfeiler and leman go machine learning: The story so far. Retrieved from `https://arxiv.org/abs/2112.09992`

Mutlu, E. C., Oghaz, T. A., Rajabi, A., & Garibay, I. (2020). Review on learning and extracting graph features for link prediction. *Mach. Learn. Knowl. Extr.*, *2*, 672–704.

Nickel, M., Rosasco, L., & Poggio, T. A. (2016). Holographic embeddings of knowledge graphs. In *AAAI*.

Nickel, M., Tresp, V., & Kriegel, H.-P. (2011). A three-way model for collective learning on multi-relational data. In *ICML*.

Nikolentzos, G., Meladianos, P., & Vazirgiannis, M. (2017). Matching node embeddings for graph similarity. In *AAAI conference on artificial intelligence*. Retrieved from `https://api.semanticscholar.org/CorpusID:3637603`

Ou, M., Cui, P., Pei, J., Zhang, Z., & Zhu, W. (2016). Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1105–1114). ACM.

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). DeepWalk: Online learning of social representations. *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

Popescul, A., & Ungar, L. H. (2003). Statistical relational learning for link prediction. In *IJCAI workshop on learning statistical models from relational data* (Vol. 2003).

Postavaru, S., Tsitsulin, A., Almeida, F., Tian, Y., Lattanzi, S., & Perozzi, B. (2020). InstantEmbedding: Efficient local node representations. *ArXiv*, *abs/2010.06992*.

Qiu, J., Dong, Y., Ma, H., Li, J., Wang, K., & Tang, J. (2018). Network embedding as matrix factorization: Unifying DeepWalk, LINE, PTE, and node2vec. *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining.*

Radosavovic, I., Johnson, J., Xie, S., Lo, W.-Y., & Dollár, P. (2019). On network design spaces for visual recognition. Retrieved from `https://arxiv.org/abs/1905.13214`

Rampášek, L., Galkin, M., Dwivedi, V. P., Luu, A. T., Wolf, G., & Beaini, D. (2022). Recipe for a General, Powerful, Scalable Graph Transformer. *Advances in Neural Information Processing Systems*, *35*.

Rozemberczki, B., Allen, C., & Sarkar, R. (2021). Multi-scale attributed node embedding. *Journal of Complex Networks*, *9*(2), cnab014.

Sadeghi, A., Malik, H. A., Collarana, D., & Lehmann, J. (2021). Relational pattern benchmarking on the knowledge graph link prediction task. In *NeurIPS datasets and benchmarks*.

Sarkar, P., Chakrabarti, D., & Moore, A. W. (2011). Theoretical justification of popular link prediction heuristics. In *International joint conference on artificial intelligence*.

Sato, R., Yamada, M., & Kashima, H. (2020). Random features strengthen graph neural networks. In *SDM*.

Schlichtkrull, M., Kipf, T., Bloem, P., Berg, R. van den, Titov, I., & Welling, M. (2018). Modeling relational data with graph convolutional networks. In *ESWC*.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., & Eliassi-Rad, T. (2008). Collective classification in network data. *AI Magazine*, *29*(3), 93. http://doi.org/10.1609/aimag.v29i3.2157

Shchur, O., Mumme, M., Bojchevski, A., & Günnemann, S. (2018). Pitfalls of graph neural network evaluation. *arXiv Preprint arXiv:1811.05868*.

So, M. K. P., Chu, A. M. Y., Tiwari, A., & Chan, J. N. L. (2020). On topological properties of COVID-19: Predicting and controling pandemic risk with network statistics. *medRxiv*.

Sun, Z., Deng, Z., Nie, J.-Y., & Tang, J. (2018). RotatE: Knowledge graph embedding by relational rotation in complex space. *ArXiv*, *abs/1902.10197*.

Talasu, N., Jonnalagadda, A., Pillai, S. S. A., & Rahul, J. (2017). A link prediction based approach for recommendation systems. In *2017 international conference on advances in computing, communications and informatics (ICACCI)* (pp. 2059–2062). IEEE.

Tang, J., Qu, M., & Mei, Q. (2015). PTE: Predictive text embedding through large-scale heterogeneous text networks. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015). LINE: Large-scale information network embedding. *Proceedings of the 24th International Conference on World Wide Web*.

Taskar, B., Wong, M. F., Abbeel, P., & Koller, D. (2003). Link prediction in relational data. In *NIPS*.

Teru, K. K., Denis, E., & Hamilton, W. L. (2020). Inductive relation prediction by subgraph reasoning. In *ICML*.

Trouillon, T., Dance, C. R., Gaussier, Éric, Welbl, J., Riedel, S., & Bouchard, G. (2017). Knowledge graph completion via complex tensor factorization. *J. Mach. Learn. Res.*, *18*, 130:1–130:38.

Vashishth, S., Sanyal, S., Nitin, V., & Talukdar, P. (2020). Composition-based multi-relational graph convolutional networks. In *International conference on learning representations*. Retrieved from `https://openreview.net/forum?id=BylA_C4tPr`

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *30*.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio', P., & Bengio, Y. (2018). Graph attention networks. *ArXiv*, *abs/1710.10903*.

Veličković, P. (2022). Message passing all the way up. Retrieved from `https://arxiv.org/abs/2202.11097`

Wang, D., Zhang, J., Cao, W., Li, J., & Zheng, Y. (2018). When will you arrive? Estimating travel time based on deep neural networks. In *AAAI conference on artificial intelligence*.

Wang, H., Yin, H., Zhang, M., & Li, P. (2022). Equivariant and stable positional encoding for more powerful graph neural networks. In *International conference on learning representations.*

Wang, P., Han, J., Li, C., & Pan, R. (2019). Logic attention based neighborhood aggregation for inductive knowledge graph embedding. In *AAAI.*

Wang, X., He, X., Cao, Y., Liu, M., & Chua, T.-S. (2019). KGAT: Knowledge graph attention network for recommendation. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.*

Wang, Z., Zhao, H., & Shi, C. (2022). Profiling the design space for graph neural networks based collaborative filtering. *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining.*

Wu, F., Zhang, T., Souza, A. H. de, Fifty, C., Yu, T., & Weinberger, K. Q. (2019). Simplifying graph convolutional networks. *ArXiv*, *abs/1902.07153.*

Xie, Y., Qiu, J., Yu, W., Feng, X., Chen, Y., & Tang, J. (2021). NetMF+: Network embedding based on fast and effective single-pass randomized matrix factorization. *ArXiv*, *abs/2110.12782.*

Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2019). How powerful are graph neural networks? In *International conference on learning representations.* Retrieved from `https://openreview.net/forum?id=ryGs6iA5Km`

Yang, B., Yih, W., He, X., Gao, J., & Deng, L. (2015). Embedding entities and relations for learning and inference in knowledge bases. *CoRR*, *abs/1412.6575.*

Yang, Z., Cohen, W., & Salakhudinov, R. (2016). Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning* (pp. 40–48). PMLR.

Yao, H., Wu, F., Ke, J., Tang, X., Jia, Y., Lu, S., ... Li, Z. J. (2018). Deep multi-view spatial-temporal network for taxi demand prediction. In *AAAI conference on artificial intelligence.*

Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., ... Liu, T.-Y. (2021). Do transformers really perform bad for graph representation? In *Neural information processing systems.*

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., & Leskovec, J. (2018). Graph convolutional neural networks for web-scale recommender systems. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.*

You, J., Du, T., & Leskovec, J. (2022). ROLAND: Graph learning framework for dynamic graphs. In *Proceedings of the 28th ACM SIGKDD conference on knowledge* *discovery and data mining* (pp. 2358–2366).

You, J., Ying, R., & Leskovec, J. (2019). Position-aware graph neural networks. In *International conference on machine learning* (pp. 7134–7143). PMLR.

You, J., Ying, R., & Leskovec, J. (2020). Design space for graph neural networks. In *NeurIPS*.

Zhang, M., & Chen, Y. (2017). Weisfeiler-lehman neural machine for link prediction. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Zhang, M., & Chen, Y. (2018). Link prediction based on graph neural networks. *ArXiv*, *abs/1802.09691*.

Zhang, M., & Chen, Y. (2020). Inductive matrix completion based on graph neural networks. In *International conference on learning representations*. Retrieved from `https://openreview.net/forum?id=ByxxgCEYDS`

Zhang, M., Li, P., Xia, Y., Wang, K., & Jin, L. (2020). Labeling trick: A theory of using graph neural networks for multi-node representation learning. In *Neural information processing systems*.

Zhao, L., Jin, W., Akoglu, L., & Shah, N. (2022). From stars to subgraphs: Uplifting any GNN with local structure awareness. In *International conference on learning representations*. Retrieved from `https://openreview.net/forum?id=Mspk_WYKoEH`

Zhao, T., Yang, C., Li, Y., Gan, Q., Wang, Z., Liang, F., ... Shi, C. (2022). Space4HGNN: A novel, modularized and reproducible platform to evaluate heterogeneous graph neural network. *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Zhou, T., Lü, L., & Zhang, Y.-C. (2009). Predicting missing links via local information. *The European Physical Journal B*, *71*, 623–630.

Zitnik, M., & Leskovec, J. (2017). Predicting multicellular function through multilayer tissue networks. *Bioinformatics*, *33*(14), i190–i198.