Journal of Artificial Intelligence Research 78 (2023) 287-356 Submitted 03/2023; published 10/2023

A Comprehensive Survey on Deep Graph Representation Learning Methods

Ijeoma Amuche Chikwendu IJEOMAAMUCHE@STD.UESTC.EDU.CN
Xiaoling Zhang 1132873501@QQ.COM
Isaac Osei Agyemang IOAGYEMANG@STD.UESTC.EDU.CN
Isaac Adjei-Mensah IADJEIMENSAH@STD.UESTC.EDU.CN
School of Information and Communication Engineering
University of Electronic Science and Technology of China
611731 Chengdu, China

Ukwuoma Chiagoziem Chima UKWUOMA@STD.UESTC.EDU.CN
Chukwuebuka Joseph Ejiyi GREATJEJIYI@STD.UESTC.EDU.CN
School of Information and Software Engineering
University of Electronic Science and Technology of China
610054 Chengdu, China

Abstract

There has been a lot of activity in graph representation learning in recent years. Graph representation learning aims to produce graph representation vectors to represent the structure and characteristics of huge graphs precisely. This is crucial since the effectiveness of the graph representation vectors will influence how well they perform in subsequent tasks like anomaly detection, connection prediction, and node classification. Recently, there has been an increase in the use of other deep-learning breakthroughs for data-based graph problems. Graph-based learning environments have a taxonomy of approaches, and this study reviews all their learning settings. The learning problem is theoretically and empirically explored. This study briefly introduces and summarizes the Graph Neural Architecture Search (G-NAS), outlines several Graph Neural Networks' drawbacks, and suggests some strategies to mitigate these challenges. Lastly, the study discusses several potential future study avenues yet to be explored.

1. Introduction

Envision a hypothetical realm in which interconnectivity flourishes and elaborate patterns emerge from complicated relationships that span extensive networks. In the contemporary era of digital advancements, where knowledge acquisition relies heavily on data, graphs are pivotal as underappreciated protagonists. They encapsulate the fundamental nature of interrelationships that mold our global landscape. Have you ever contemplated how we can unravel these complex networks, decipher their concealed dynamics, and utilize their potential for profound insights? Within the dynamic and ever-changing realm of data science, a particular paradigm emerges as a noteworthy and valuable asset: deep graph representation learning. Notably, graphs provide the potential to elucidate the complexities inherent in interrelated data, including many domains such as social networks and biological networks. Deep graph representation learning could improve our understanding of complex relationships, find hidden patterns, and increase machine learning. This study carefully explores this intriguing area.

The discipline of graph representation learning has become a significant area of study within the broader science of machine learning. Its primary objective is to develop efficient methods for processing and evaluating data that is represented in the form of graphs. Graphs are very effective structures that are utilized to represent and comprehend complex associations between different entities. As a result of their inherent capabilities, graphs are particularly suitable for a wide range of practical applications in the real world, including but not limited to social networks, recommendation systems, bioinformatics, and numerous others.

The initial advancements in graph representation learning were first observed in the field of graph kernels, as documented in historical records. The origins of graph kernel approaches can be traced back to the influential Weisfeiler-Lehman (WL) isomorphic testing (Weisfeiler & Leman, 1968), a fundamental notion that emerged earlier. This methodology, a fundamental pillar in the field, established the foundation for graph kernels - kernel functions carefully crafted to measure the similarity of graphs and their components. Within the domain of current scholarly investigations, the notion of graph kernels reverberates prominently in educational endeavors Nikolentzos et al. (2021), serving as a testimonial to the ongoing significance of this transformative framework.

The essential principle underlying graph kernels is the decomposition of complex graphs into distinct substructures. These substructures are then used to generate vector embeddings, which are carefully designed based on the features of these substructures. The origins of graph representation learning can be traced back to its commencement within the domain of matrix factorization techniques. The initial stage of exploration was heavily influenced by traditional methods of dimensionality reduction, reflecting the influential work of Belkin & Niyogi (2001) and their significant contributions to the area. Several matrix factorization-based models have already been developed to effectively handle large-scale graphs with millions of interconnected nodes (Allab et al., 2016; Gong et al., 2014). Matrix factorization methods play a crucial role in this endeavor due to their intrinsic capability to reduce intricate proximity matrices into products of simpler matrices. The objective is to understand embeddings comprehensively, focusing on their ability to capture and represent the inherent proximity patterns effectively. Between the years 2014 and 2016, there was a notable development in the field, as two significant shallow models, namely DeepWalk (Perozzi et al., 2014) and Node2Vec (Grover & Leskovec, 2016), emerged in the environment. These methodologies utilized shallow neural networks to generate node embeddings. One notable characteristic of these models was their innovative utilization of the skip-gram framework, which was originally grounded in the field of natural language processing. The primary principle that governed these approaches was to enhance the information contained in node embeddings by effectively maximizing the likelihood of neighbouring nodes. The deployment of Stochastic Gradient Descent (SGD) across neural network layers can successfully mitigate computational subtleties, so elegantly harnessing and fine-tuning the strategic basis of this approach. This event was a significant turning point, driving the advancement of several models that were ready for further improvement. A multitude of breakthroughs have arisen, encompassing improved sampling procedures and iterative training processes, jointly influencing the direction of progress in this dynamic subject. Research into graph representation learning has recently gained traction since graphs conveniently utilize and represent most real-world data. Multimedia domain-specific data includes but is not limited to, social systems (Tan et al., 2019), linguistic (word co-occurrence) networks (Agrawal et al., 2021), biological structures (G. Zhou & Xia, 2018), and sundry. Graph models efficiently store and retrieve relational knowledge of interacting entities (Besta et al., 2019). Graph data analysis can help with community discovery, behavior analysis, node classification, link prediction, and clustering

(Daud et al., 2020; Goyal & Ferrara, 2018; Inuwa-Dutse et al., 2021; J. Li et al., 2019; Zitouni et al., 2019). Graph embedding methods often transform unprocessed graph data into a high-dimensional vector while retaining essential graph characteristics. This method is called graph representation learning. Researchers used conventional machine learning methods based on the derived features and the original data format. Pixel and word occurrence statistics are retrieved from photos and text, respectively. In the middle of the current surge of innovation, the objective of our research is to undertake an exploration of the fundamental aspects of graph representation learning. The central inquiry of this study is: How can we effectively navigate the intricate nature of graph data, enable machines to understand connections, and advance the area of deep graph representation learning to unexplored frontiers?

Deep Learning (DL) systems have become popular over the past decades because they can solve learning problems, learn representations from raw data, and make predictions based on the taught representation. In recent times, the field of artificial intelligence has undergone a significant transformation due to the emergence of deep learning techniques. These techniques have demonstrated exceptional achievements in various domains, such as image recognition, natural language processing, and speech recognition. In a similar vein, the fusion of deep learning techniques with graph-based data has resulted in the emergence of deep graph representation learning approaches. The objective of these methodologies is to utilize the computational powers of deep neural networks to acquire meaningful representations from graph data, hence facilitating improved decision-making and prediction skills. The present level of research in deep graph representation learning is characterized by its dynamic and swiftly progressing nature. Scholars and professionals are consistently investigating new designs, methods of optimization, and algorithmic advancements to tackle the distinct obstacles presented by data that is structured as a graph. The utilization of Graph Neural Networks (GNNs), Graph Convolutional Networks (GCNs), and Graph Attention Networks (GATs) has demonstrated encouraging outcomes in several applications, with significant advancements achieved in domains including node categorization, link prediction, and graph construction. Due to graphs' irregular character, which may contain a changeable number of unordered nodes and a changing number of neighbours, several key operations, like convolutions, are simple to perform in the image domain but difficult in the graph sector. In addition, modern machine learning techniques presume, among other factors, that instances stand alone. References, relationships, and engagements relate to graph data; hence this premise no longer applies. Despite advances in computer vision, natural language processing, biological imaging, and bioinformatics, DL still lacks relational and scientific thinking, intellectual abstraction, and other cognitive capacities. Graph Neural Networks (GNNs) structure computations and representations in Deep Neural Networks (DNNs) as graphs to address these issues. GNNs are graph-domain deep learning algorithms. Graphs are difficult to visualize; thus, using deep learning algorithms to evaluate graph data has garnered attention in recent years.

Graphs Having Irregular Structures: Unlike pictures, music, and text, which have a grid structure, graphs have uneven topologies, making some basic mathematical operations harder (Shuman et al., 2013). Graph data makes convolution and pooling difficult.

Heterogeneity and Variety: Graphs with numerous shapes and properties can be complex. Heterogeneous, homogeneous, weighted, and signed graphs are possible. Graph-related activities include node classification, link prediction, graph classification, and graph synthesis. Different model structures are needed to address different types, qualities, and tasks.

Interdisciplinarity: Biology, chemistry, and the social sciences often use graphs. Domain knowledge is essential to solving problems but can also slow model design. Gradient-based training approaches are difficult for molecular graphs due to non-differentiable objective functions and chemical restrictions.

Embedding Dimension and Graph Features: Finding the optimal embedding dimension of representation (Gou et al., 2020) is complex and involves additional challenges (Shen et al., 2020). Higher-dimensional illustrations retain more graph features as much as they need more storage and processing time. Lower-dimensional depictions require more resources, and it could reduce graph noise too. However, the original graph may lose important data. Input graph and application domain affect dimension selection (H. Chen et al., 2018). If a graph has several properties, embedding one may be difficult. Node features, connection designs, meta-data, and more can show graph characteristics. The application determines the most helpful information. Kernel functions (Nikolentzos et al., 2021), summary graph measures (such as degrees or grouping coefficients) (Daud et al., 2020), and carefully chosen features to quantify local neighbourhood structures (J. Li et al., 2019) are frequently used by traditional machine learning algorithms to extract structural information from graphs. However, these systems cannot change during the learning process because of the rigidity of the hand-engineered characteristics. Moreover, it may be costly and time-consuming to implement these functionalities.

Gaining a comprehensive comprehension of the importance of deep graph representation learning is vital within the contemporary context of a data-centric society. As the complexity of our interactions with interconnected data increases, it becomes increasingly important to possess the capability to identify concealed patterns, uncover latent linkages, and execute accurate predictions. Therefore, this extensive examination of deep graph representation learning is a current and essential source, offering a detailed investigation of state-of-the-art techniques, computational enhancements, and probable directions for future scholarly inquiry. By furthering our understanding in this field, we create opportunities to revolutionize our approach to interpreting, analyzing, and deriving insights from intricate data structures, thus paving the way for enhanced, data-centric decision-making processes.

1.2 Scope. This study covers methods for representing nodes, edges, and subgraphs, which provide context, intelligence, and semantics to graphs for applications, and evaluates graph representation learning research. We unify several diverse lines of research that have attracted significant attention in recent years across various domains and venues while also focusing on cutting-edge techniques that are scalable to enormous graphs and inspired by deep learning. GNNs perform well on graph-structured datasets in supervised, semi-supervised, self-supervised, and unsupervised learning contexts. Auto-encoders, contrastive learning, and random walk ideas underpin most graph-based unsupervised learning approaches. The studies primary aim is to conduct a comprehensive investigation of the advanced techniques in the field of deep graph representation learning and to conduct an in-depth assessment of its advanced methodologies, carefully examining their respective advantages and constraints. Graph Neural Architecture Search (G-NAS) is introduced in this study, it incorporates and classifies G-NAS components. This classification, based on graph neural networks (GNNs)' intrinsic problems in architectural design, fills a significant gap in the literature. It helps design GNN architectures with improved efficiency and efficacy by explaining the key components of (G-NAS) and their implications. Furthermore, this study aims to explore the complex restrictions inherent in GNN techniques, focusing on the issues associated with achieving interpretability and scalability in graph-based models, amongst others.

With the increasing number of graph representation learning models in recent years, various approaches have been utilized to find relevant research in this domain. By adopting a strategic approach, a search methodology was developed formulating specific keywords and carefully evaluating reliable sources. The compilation of keywords includes concepts such as graph embedding, graph representation learning, graph neural networks, graph convolution, and graph attention. The search for pertinent research involved prominent and acclaimed conferences and journals, such as AAAI, IJCAI, SIGKDD, ICML, WSDM, Nature Machine Intelligence, and Pattern Recognition, as well as trustworthy internet sources.

Overview of the Survey. The subsequent section of this study is structured as follows: In the second section, a concise summary of relevant literature is presented, encompassing various surveys and overviews within the area. Section 3 introduces the concept of Graph Representational Learning (GRL) and examines several graph tasks based on data with a graph structure. In Section 4, the categorization of GNN-based techniques and learning situations is presented logically to facilitate complete comprehension. Section 5 provides an in-depth analysis of the sequential structure of Graph Neural Networks (GNNs), elucidating their internal mechanisms. In the subsequent section, Section 6, the discourse shifts toward contemporary applications of Graph Neural Networks (GNNs), thereby highlighting their extensive and diverse practical utility. In the following section, an analysis is conducted on the inherent limitations of Graph Neural Networks (GNNs), and various strategies are proposed to overcome these constraints. Section 8 provides a concise overview of Graph Neural Architecture Search (G-NAS), emphasizing its notable importance within the domain. Section 9 elucidates unresolved aspects of graph solutions based on GNNs, opening the path for future research. The research finishes in the concluding part, presenting a comprehensive synthesis of the knowledge acquired throughout the article.

Contributions. The main contributions of this study are summarized as follows;

1. Firstly, a thorough analysis of GNNs is provided. In contrast to the other studies focusing on only one type of learning environment, this study considers all of them.
2. This study introduces and categorizes G-NAS constituents based on the building challenges; this is not provided in previous surveys.
3. This study outlines GNN-based method limits and workarounds. Limitations include over-smoothing, scalability, expressiveness, over-squashing, and destructive loss, to mention but a few.

2. Related Work: Surveys in Graph Representational Learning

The present literature inventory on Graph Neural Networks (GNNs) primarily consists of survey studies that either cover a wide range of topics or go into a specific learning environment (Ahmad et al., 2020; Chami et al., 2022; C. Chen et al., 2022; F. Chen et al., 2020; J. Zhou et al., 2020; Y. Zhou et al., 2022). In their paper, Abadal et al. (2021) conducted a comprehensive examination of Graph Neural Networks (GNNs), focusing on their computational aspects. Furthermore, the research encompassed a thorough examination of the several software and hardware acceleration techniques already employed. The authors of this study were provided with a communication-focused, hardware-software hybrid that represents an appropriate solution for GNN accelerators. In their study, Zhou et al. (2020) provided a thorough design process for Graph Neural Networks

(GNNs) and discussed several GNN variations employed in each module. The authors comprehensively analyzed the theoretical and empirical aspects of Graph Neural Networks (GNNs) in the study. The initial phase of the paper's discourse involved categorizing GNN applications into two distinct groups: structural and non-structural scenarios.

Furthermore, the article elucidated four outstanding matters concerning GNNs and deliberated on probable prospects. Conversely, it lacked a clearly defined taxonomy for individual learning scenarios. The study conducted by Z. Wu et al. (2020) introduced a novel categorization framework for Graph Neural Networks (GNNs). This framework categorizes GNNs into many subtypes: recurrent, convolutional, spatial-temporal, and graph autoencoder architectures. Nevertheless, the study could have thoroughly examined every learning environment. The majority of the current survey studies in the field of Graph Neural Networks (GNNs) primarily concentrate on either the individual learning scenario or on the broader scope of GNNs, as shown in Table 1. To bridge this knowledge gap and expand upon the existing body of literature, the current research undertakes a comparative examination of different graph-based deep learning architectures. Significantly, we explore G-NAS captivating domain. This innovative addition has been carefully designed to address the distinct construction constraints associated with GNNs. Through a methodical approach in addressing these issues, the G-NAS presents a novel viewpoint and framework to enhance the domain of GNNs, thereby paving the way for groundbreaking progress.

Papers	**Difference between this survey and existing ones**
Abadal et al. (2021)	They thoroughly examined Graph Neural Networks (GNNs) from a perspective well-grounded in the field of computing. It carried out the various methodologies used for software and hardware acceleration. This investigation led to the development of a novel vision emphasizing GNN accelerators' significance. These accelerators are distinguished by their graph-awareness, hardware-software integration, and communication-centric features. In contrast, our study deviates from this trajectory by pursuing a unique analysis path. Our attention is directed toward the many learning environments present in the domain of Graph Neural Networks (GNNs). We proficiently establish a delineated classification system for various learning environments by employing a systematic and thorough methodology. This endeavor results in a unified framework that enhances the comprehension of Graph Neural Networks (GNNs) within these heterogeneous settings.
(Z. Zhang et al., 2020)	Predominantly delved into classical and representative Graph Neural Network (GNN) architectures, hence, by passing the exploration of deep graph representation learning from the vantage point of the latest advanced paradigms like graph self-supervised learning, our research carves a distinctive path, as it stands as a beacon of comprehensive scrutiny. Significantly, the study explores the complex domain of deep graph representation learning, revealing concealed insights and innovative approaches. In order to enhance the level of intellectual discussion, this study presents the innovative notion of Graph Neural Architectural Search (GNAS).
(Z. Wu et al., 2020)	They introduced a new categorization approach that divided well-known GNNs into four groups: recurrent, convolutional, spatial-temporal, and graph autoencoders. This theoretical paradigm fails to explain in depth the learning settings. Contrarily, our article seeks to enrich academic discourse, by introducing taxonomies tailored to GNN learning contexts. This strategic method helps our inquiry by disclosing the deep complexity and subtle details of each GNNs learning situation.

(J. Zhou et al., 2020)	This study presents a comprehensive overview of the design pipeline of GNNs and provides a detailed analysis of several module variants employed in GNNs. The article conducted a comprehensive analysis of GNNs from both a theoretical and empirical standpoint. The research delineated the applications of GNNs by categorizing them into two distinct scenarios: structural and non-structural. The report additionally presented four unresolved issues pertaining to GNNs and provided prospects for future research in this area. However, it is worth noting that the paper has not presented a distinct taxonomy for each of the many learning situations and this study takes advantage to research about various learnings of GNNs.
(Khoshraftar & An, 2022)	Classifies works in graph representation learning, accurately differentiating between static and dynamic graphs. However, although these taxonomies effectively highlight the core principles of Graph Neural Networks (GNNs), their coverage of learning paradigms needs to be improved. Our research is a comprehensive study examining various learning settings inherent to Graph Neural Networks (GNNs).
(Y. Zhou et al., 2022)	Presents a comprehensive overview of Graph Neural Network (GNN) designs, briefly discussing their applications. Concurrently, our study follows a comparable path, investigating GNN designs and pushing the limits of exploration and presenting a novel aspect called Graph Neural Architectural Search (G-NAS) during our thorough investigation. The innovative addition discussed in this work serves as a valuable resource for scholars, providing guidance and insight into the complex domain of Graph Neural Networks (GNNs). It sheds light on the inherent obstacles in constructing GNNs, enhancing our understanding of this field. The G-NAS framework provides a comprehensive experience of the fundamental components of GNNs, offering researchers essential insights into this rapidly evolving and impactful domain.

Table 1: Difference between this survey and existing ones.

3. Graph Representation Learning (GRL)

The Graph Representation Learning (GRL) techniques seek to develop vector representations for various graph elements to capture the structure and semantics of a graph-structured or networked rich dataset to achieve a good representation. Learning to represent graphs uses various methodologies derived from graph theory, manifold learning, topological data analysis, neural networks, and generative graph models. These methodologies all have their origins in conventional network research. When applying machine learning to networks, the most challenging aspect is undoubtedly extracting information about interactions between nodes and combining it into a machine-learning model. Traditional machine learning methods utilize either summary statistics (such as degrees or clustering coefficients) or specifically built features to quantify local neighbourhood structures to extract relevant information from networks. Examples of these statistics and features include: (e.g., network motifs). Representation learning systems can automatically learn to encode network structure into low-dimensional representations by replacing g existing methods with deep learning and non-linear dimensionality reduction. The adaptability of learned embeddings enables them to be helpful in various modelling problems. In graph representational learning, there exists a collection of models that may be categorized into separate groups: Graph Kernels, Matrix Factorization, Shallow Models, and Deep Graph Models. These models demonstrate an elegant alignment within their respective categories. Each category represents distinct strategies that contribute to the overall graph representation, providing diverse approaches to analyze and comprehend the complicated relationships embedded inside large data structures.

Graph kernels and matrix factorization-based models are foundational concepts in graph representation learning. Graph kernels are widely recognized for their utilization in graph embeddings. They utilize a definite mapping function, which allows for exploring the complexities associated with graph classification problems (Shervashidze et al., 2009; Togninalli et al., 2019). This domain has two separate classifications: graph kernels, which aim to reveal the subtle features of graph similarity, and node-based kernels on graphs, which are carefully crafted to uncover the everyday relationships between individual nodes in graph structures. Graph kernels examine graphs or their complex substructures, including nodes, subgraphs, and edges, to evaluate their similarity. At the core of this endeavor resides the fundamental task of assessing the resemblance between graphs in an unsupervised fashion. Numerous ways emerge to quantify the degree of similarity between pairs of graphs. The tactics employed comprise various techniques, including graphlet kernels, WL kernels, random walks, and shortest paths (Shervashidze et al., 2009). Graphlet kernels stand out as a straightforward yet powerful approach within the vast array of kernel approaches. Graphlet kernels are a method that operates by quantifying subgraphs of limited size. This approach allows for exploring graph similarities, revealing concealed patterns contributing to a deeper comprehension of the subject(Kondor et al., 2009). In conclusion, graph kernels serve as effective models, offering a range of advantages that highlight their importance:

- Graph kernels are widely recognized as valuable tools for quantifying the similarity between graph items by implementing various methodologies for graph kernel discovery. The statement posits that the concept above can be perceived as an overarching representation of conventional statistical approaches (Kriege et al., 2020).
- Numerous kernel techniques have been suggested in the literature to mitigate the computational burden associated with graph-based kernel methods (Urry & Sollich, 2013). The utilization of kernel tricks has the potential to decrease the spatial dimensions and computing complexity associated with substructures, all while maintaining the effectiveness of kernels.

Despite the several advantages of kernel approaches, their scalability could be improved by certain limitations.

- The majority kernel models exhibit a limitation in their ability to learn node embedding for newly introduced nodes. In practical applications, graphs possess dynamic characteristics, allowing their constituent elements to undergo evolutionary changes. Hence, re-learning charts are necessary for graph kernels whenever a new node is introduced, resulting in a time-consuming and challenging application in practical scenarios.
- Most graph kernel models do not consider the presence of weighted edges, resulting in the potential loss of structural information. This could decrease the likelihood of graph representation within the latent space.
- The computational complexity of graph kernels is classified as NP-hard (Borgwardt & Kriegel, 2005). While various kernel-based models have been developed to decrease computational time by incorporating substructure distribution, this approach may inadvertently introduce greater complexity and hinder the model's capacity to represent the overall structure accurately.

Matrix factorization-based models aim to capture the fundamental characteristics of a graph by representing it as matrices. These models extract embeddings by decomposing the matrices through a complex procedure (Ou et al., 2016; Z. Zhang et al., 2018). The present scene is embellished with a diverse range of strategic options that dictate the course of factorization modeling. The fundamental objective of these models is to effectively estimate nodes' complex interconnectedness by utilizing high-order proximity. Matrix factorization is a technique that aims to reduce the size of high-dimensional matrices, such as the adjacency matrix or the Laplacian matrix, representing the graph structure. By transforming these matrices into a lower-dimensional space, matrix factorization simplifies the representation of complex graph relationships, making it more concise. Numerous decomposition techniques, such as Singular Value Decomposition (SVD) and Principal Component Analysis (PCA), are extensively utilized in graph representation learning and recommendation systems. Various models have been developed to decrease computational complexity in matrix factorization by optimizing sampling procedures (Lian et al., 2022; R. Yang et al., 2019). The primary concept of the NRL-MF model (Lian et al., 2022) revolved around developing a hashing function specifically designed for the computation of dot products. The hashing function efficiently computes a binarized vector representation for each node using Exclusive (XOR) operators. The proposed model can acquire binary and quantized codes by utilizing matrix factorization techniques while maintaining a high level of preservation for higher-order closeness. Matrix factorization-based models offer numerous advantages.

- The models have a low dependency on the quantity of data required for learning embeddings. Compared to alternative methodologies, such as neural network-based models, these models offer distinct advantages in scenarios with a limited amount of training data available.
- Including the Laplacian matrix or transition matrix in the presentation of the graphs allows the models to represent the proximity of the nodes within the charts effectively. The connections between every pair of nodes are seen at least once in the matrix, enabling the models to effectively handle networks with sparse connectivity.

Despite the widespread utilization of matrix factorization in graph embedding problems, it is essential to acknowledge that this approach has drawbacks.

- The computational complexity of matrix factorization poses challenges regarding time and memory when dealing with enormous graphs containing millions of nodes. One primary factor contributing to this phenomenon is the temporal duration required to decompose the matrix into a series of smaller matrices (S. Cao et al., 2015).
- Models that rely on matrix factorization cannot effectively handle incomplete graphs that contain unseen and missing variables (Safavi & Koutra, 2020). When the available graph data is insufficient, matrix factorization-based models may encounter challenges in effectively learning generalized vector embeddings. Hence, there is a requirement for neural network models that can generalize graphs and enhance the accuracy of entity prediction within charts.

The Shallow models have demonstrated considerable achievements over the previous decade, as evidenced by studies (Perozzi et al., 2014; Ribeiro et al., 2017). The primary objective of these models is to represent nodes, edges, and subgraphs as vectors with minimal dimensions while maintaining the integrity of the graph structure and the proximity between entities. In general, the models initially employ a sampling strategy to capture the form of the graph and the proximity relation. Subsequently, they acquire embeddings using shallow neural network algorithms. Various sampling procedures can be employed to collect local and global data in graphs (C. Wang et al., 2020). Specific approaches focus on maintaining the integrity of the graph's structure by devising sampling methodologies capable of capturing the inherent structure within samples of specified lengths. Various sample strategies have been developed to capture local and global graph structures. These techniques include random-walk sampling, role-based sampling, and edge reconstruction. The model subsequently utilizes shallow neural network approaches to acquire vector embeddings within the latent space through unsupervised learning. Selecting an appropriate technique for capturing the graph structure is crucial in enabling external models to reach vector embeddings effectively. The graph structure can be sampled by examining the connections between nodes within the graphs or sub-structures. In the past ten years, several models have been suggested to effectively represent the graph structure and acquire embeddings (R. Liu et al., 2023; Perozzi et al., 2017). Random-walk-based tactics are among the most prevalent approaches for sampling graph structures, as evidenced by many models. The primary concept behind the random-walk technique is acquiring knowledge regarding the form of a network to generate pathways that can be interpreted as phrases within texts. Deep Walk (Perozzi et al., 2014) and Node2Vec (Grover & Leskovec, 2016) can be regarded as seminal models that have paved the way for exploring novel approaches in node embedding learning.

Motivated by the limitations of matrix factorization-based models, the DeepWalk model employs random-walk sampling to maintain node neighbourhoods, enabling the collection of global information in graphs. Furthermore, both DeepWalk and Node2Vec aim to maximize the likelihood of witnessing neighbouring nodes through the utilization of stochastic gradient descent on individual single-layer neural networks. Hence, these models effectively mitigate the duration of execution and decrease the level of computational intricacy. DeepWalk is a node embedding model that uses a random-walk sampling strategy to build node sequences, which are then treated as word sentences. One of the drawbacks inherent to this model is its inability to effectively enhance the quality of the sampling graph structure by navigating random-walk sampling. To address the constraints of DeepWalk, Node2Vec was proposed, which incorporates a versatile random-walk sampling approach to facilitate the traversal of random walks at each time step. Several limitations are associated with shallow models.

- In the context of graphs, the limited capacity of shallow models prevents them from acquiring embeddings for newly introduced nodes. To develop embeddings for novel nodes, the models must incorporate new patterns. This can be achieved by random-walk sampling to generate fresh pathways for the new nodes. Subsequently, the models must undergo re-training to learn the embeddings. The implementation of re-sampling and re-training techniques may provide challenges in real applications.
- Shallow models, such as DeepWalk and Node2Vec, are primarily effective in analyzing homogenous graphs, but they tend to overlook the properties or labels associated with individual nodes. However, it is worth noting that in practical applications, numerous charts possess features and brands that can provide valuable information for graph representation learning. Limited research has been conducted on the characteristics and designations of nodes and edges. Nevertheless, the model's inefficiency and heightened computing complexity have been exacerbated by the constraints imposed by domain knowledge in the context of diverse and dynamic graphs.
- One limitation of shallow models is the absence of parameter sharing, which prevents the models from sharing parameters during the training phase. From a statistical standpoint, parameter sharing can decrease the computational time required and the number of weight updates needed throughout the training process. To address these constraints, deep neural network models are recommended to substitute shallow models. Deep neural network models

have shown improved generalization capabilities and the ability to capture more graph entity interactions and structures.

Recently, several graph embedding models' efficacies has been tested by large-scale graphs' presence. Conventional approaches, such as shallow neural networks or statistical techniques, could improve their ability to effectively represent intricate graph topologies due to their simplistic architectural design. In recent times, there has been a surge in research focusing on deep graph neural networks. These networks have gained significant attention due to their remarkable capability to handle intricate and extensive graph structures (Bojchevski & Günnemann, 2017; M. Liu et al., 2020). In contrast to previous models, most deep neural network-based models utilize the graph structure and node attributes/features to get node embeddings. For example, individuals using the social network platform may own textual data, such as personal details featured in their profiles. When nodes lack attribute information, the attributes or features can be encoded using node degree or one-hot vectors (Kipf & Welling, 2016b). A notable category known as Graph Autoencoders has emerged in deep graph networks. These unsupervised learning algorithms specialize in encoding graph items into latent spaces and reconstructing these entities using the encoded information. The intricate process of encoding and reconstructing data is a defining characteristic of Graph Autoencoders, granting them a distinctive and influential position in deep graph representation learning.

Graph Autoencoder models can be categorized into two major groups based on their architectural attributes: Multilayer Perceptron (MLP)-based models and Recurrent Graph Neural Networks (RGNNs). Early Graph Autoencoder models mostly used the Multilayer Perceptron (MLP) architecture during the first stages of their development. The design decision demonstrated its ability to effectively incorporate complex embeddings, as evidenced by the groundbreaking studies (S. Cao et al., 2016; Tu, Cui, Wang, Wang et al., 2018), which established the foundation of this lineage. Another model that can be considered is the RGNNs, which stands out as one of the pioneering approaches in utilizing deep neural networks for graph representation learning. RGNNs are built upon the foundation of GNNs. The primary concept underlying GNNs is the incorporation of messages passing between target nodes and their neighbouring nodes until a state of equilibrium is reached. Recurrent graph neural networks offer numerous advantages in comparison to shallow learning techniques.

- RGNNs have demonstrated enhanced learning capabilities in processing scattered information, particularly in multi-relational graphs with nodes with numerous connections. This capability is attained by modifying the states of every node within each concealed layer.
- Parameter sharing is a technique employed by RGNNs to share parameters across several locations. This allows RGNNs to capture the inputs of sequence nodes effectively. This benefit could decrease computational complexity during training by utilizing fewer parameters, enhancing the models' performance.

Nevertheless, a drawback of RGNNs lies in their utilization of recurrent layers that possess identical weights throughout the weight update procedure. This phenomenon results in inefficiencies when specifying various relationship constraints between neighbouring and target nodes. In recent years, convolutional graph neural networks (CGNNs) have demonstrated significant efficacy in addressing the limitations of RGNNs by leveraging distinct weights in each hidden layer. Convolution operators can be defined and applied to graph mining, as image data can be seen as a specific instance of graph data. Two distinct methodologies exist for implementing convolution operators in the graph domain. The initial approach relies on the principles of graph spectrum theory, wherein graph entities are converted from the spatial domain to the spectral domain. Subsequently, convolution filters are employed on the spectral domain. The alternative approach involves utilizing convolution operators within the spatial domain of the graph. Most spectral models acquire node embeddings by converting graph data into the signal domain and use convolutional filters, resulting in heightened computational complexity. Kipf & Welling (2016a) presented the concept of graph convolutional networks (GCNs), which were seen as connecting spectral and spatial methodologies. Despite the effectiveness of spectral CGNNs for performing convolution filters on the spectrum domain, they exhibit numerous drawbacks, which are outlined below:

- The computational complexities associated with the decomposition of the Laplacian matrix, specifically in obtaining matrices rich in eigenvectors, have been well recognized as time-consuming tasks. The temporal commitment is significantly increased by the repetitive dot product calculations that occur between eigenvectors and Laplacian matrices throughout the training process.
- In the context of computational systems, a significant issue arises when confronted with extensive networks. There exists a clear association between the parameters that govern the kernels and the number of nodes contained inside the graphs. As a result, the domain of spectral models, which heavily depend on these parameters, may face constraints in situations involving large graph dimensions. The aforementioned subtle constraint highlights the pragmatic factor that spectral models may not be optimally suitable for graphs of substantial scale.
- The task of addressing the challenges posed by dynamic graphs entails dealing with a distinct array of intricacies. The application of convolution filters and the training of the model need the conversion of graph data into the spectral domain, often accomplished by utilizing a Laplacian matrix. However, this particular transition presents a significant difficulty. The model's effectiveness is compromised in situations typified by dynamic graphs, where the data within the graph is fundamentally fluid and liable to change. The current framework, which is highly attuned to the spectral domain, faces difficulties in accurately representing the constantly changing intricacies of dynamic graphs. As a result, it presents a notable difficulty in capturing and accommodating these fluctuations.

Spatial models have emerged as a promising option for addressing the limitations inherent in spectral domain-based CGNNs. Spatial models introduce a fresh approach by utilizing convolution operators within the graph domain, enabling the efficient acquisition of node embeddings more powerfully.

The field's current state displays various spatial CGNNs, each with its unique methodology. These networks have garnered significant recognition for their ability to efficiently navigate complex graph structures, often surpassing the performance of spectral equivalents (Chiang et al., 2019). However, a shortcoming of CGNNs becomes apparent at the hidden layer. In this context, the model effectively coordinates updating the state of surrounding nodes. However, this dynamism might unintentionally result in slow training and updating protocols, particularly when inactive nodes are present.

To overcome this obstacle, researchers have strengthened CGNNs by strategically enhancing the sampling approach (J. Chen et al., 2018; Z. Huang et al., 2021). (J. Chen et al., 2018) proposed the FastGCN model, which aims to enhance both training efficiency and overall model performance, surpassing traditional CGNNs. Amidst a multitude of technological breakthroughs, the issue of scalability emerges as a significant worry. Current GNN models face challenges in dealing with the rapid growth of neighbourhoods, which leads to an increase in computational complexity. The urgency of this matter highlights the need for novel approaches that not only explore the extent to which scalability may be achieved but also establish a balanced relationship between performance and computing requirements.

By transcending traditional paradigms, the proposed model seamlessly incorporates neighbourhood sampling into each convolutional layer, employing a strategic approach focusing on crucial surrounding nodes. The intelligent methodology enables the model to adaptively acquire knowledge about the essential neighbouring nodes unique to each batch, effectively focusing on the most critical aspects. One notable example in this field is the Graph Attention Networks (GATs) model, which was proposed by (Velickovic et al., 2017). This innovative model is at the forefront of utilizing the attention mechanism in the complex domain of graph representation learning. The fundamental nature of the attention mechanism is in its capacity to coordinate a deliberate message for every adjacent node throughout the iterative process of message-passing inherent to Graph Neural Networks (GNNs). The significance of each neighbouring node to the target node can be quantified by calculating an attention score carefully and deliberately. After performing the necessary calculations, the score is subjected to a normalization process, efficiently aided by the SoftMax function. The normalization process successfully ensures that the scores can be compared across all neighbouring nodes of the target node. Following the harmonization procedure, a node's embeddings are generated by skillfully combining the states of its nearby nodes. The orchestration presented in this context demonstrates a dynamic interplay, effectively capturing the fundamental nature of graph relationships in a coherent and influential manner.

Furthermore, the GAT model utilized the effectiveness of multi-head attention, a strategic technique that increased the model's capabilities and introduced improved learning stability. However, throughout this pioneering endeavor, a subtle constraint emerges. The GAT model, which relies on attention coefficients to govern its mechanism, unconditionally prioritizes attention. Consequently, this limited framework needs to improve its ability to encompass the intricate details of the global graph structure fully. Recently, there has been a notable increase in the development of novel models, all stemming from the fundamental principle of GAT. The main focus of these models is to enhance the intrinsic capacity of the self-attention mechanism, therefore fostering a more profound connection with the extensive network of global graph structures (Ma et al., 2021).

In this quest, researchers actively explore techniques that effectively explore and comprehend the complex harmonies within the more fantastic graph world. Deep neural network models offer several notable advantages:

- One notable advantage of deep neural network models is the deliberate utilization of parameter sharing, wherein weights are shared strategically throughout the training process. This innovative methodology results in three advantageous outcomes: a decrease in the duration of the training, a limitation in the number of training parameters, and a simultaneous enhancement in the model's performance. Furthermore, the fundamental principle of parameter sharing expands its scope to enable models to incorporate multi-task learning, highlighting this approach's inherent versatility and efficiency.
- Another notable advantage that sets deep models apart from shallow models is their ability to engage in inductive learning. This crucial characteristic endows deep-learning models with the unique capability to surpass the limitations of their training material and extend their knowledge to include unknown instances. The particular skill of these models allows

them to effectively navigate unfamiliar areas, which gives them practical value and significance in real-world situations.

Nevertheless, some limitations are:

- CGNNs rely primarily on an aggregation process for mapping the complex paths of graph structure and entity interactions. The method above diligently collects data from adjacent nodes to enhance the comprehension of target nodes. When many graph convolutional layers are stacked in CGNNs to capture higher-order graph structures, a significant problem arises. The decision to increase the depth of the convolutional layer may unintentionally lead to a problem of excessive smoothing. This occurrence might hamper the model's capacity to accurately detect and analyze small fluctuations and subtleties in the data (T. Chen et al., 2022; L. Zhao & Akoglu, 2019).
- Limitations in Disassortative Graphs: The discourse diverges when considering disassortative graphs, which are characterized by nodes with various labels that tend to connect. The inherent aggregation mechanism included in GNNs emerges as a constraint and obstacle. Despite the varied tags assigned to nodes, the aggregation process uniformly selects attributes from surrounding nodes, concealing the subtle disparities that are the basis of disassortative graphs. This constraint becomes especially significant in classification tasks, as GNNs struggle to effectively integrate their aggregation approach with the complexities of disassortative graph structures.

3.1 Notation and Fundamentals of Graph Representation Learning

Mathematically, a graph is represented as $G = (V; E)$ where $V = \{v_1; ::: ; v_N\}$ is a set of $N = |V|$ nodes and $E \subseteq V \times V$ is a set of $M = |E|$ edges between nodes. We use $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ to denote the adjacency matrix, whose i^{th} row, j^{th}column, and an element are denoted as $A(i,:); A(:,j); A(i,j)$, respectively. A node v and an edge e_{uv} can store characteristics or qualities represented by vectors $\boldsymbol{x}_v$ and $\boldsymbol{x}_{uv}^e$, respectively. The node characteristics of a graph are expressed by a matrix $\boldsymbol{X} \in \mathbb{R}^{\boldsymbol{n} \times \boldsymbol{d}}$, where d represents a node feature size. A matrix represents the edge features of a graph $\boldsymbol{X}^e \in \mathbb{R}^{\boldsymbol{m} \times \boldsymbol{c}}$, where m and c depict the number of edges and an edge feature size, respectively. $\boldsymbol{X}^T$ denotes the transposition of a matrix, and the element-wise multiplication is written as $\boldsymbol{X}1 \odot \boldsymbol{X}2$. In this study, matrices are represented by uppercase bold characters and vectors by lowercase bold characters unless otherwise noted. Table 2 contains all the symbols and abbreviations used throughout the study.

3.2 Classification of Tasks Via Graph-based Data Hierarchies

As seen in Figure 1, the data represented by graphs have the potential to have information integrated at various levels of the structure. At the node level, the different node-based task has their definitions. In addition, edge-level tasks are also definable. The tasks that need to be completed at the graph level can be tailored to the requirements of the various applications.

Representing Node Tasks: The goal of representing node tasks is to learn representations of various network elements, including nodes. Node-level taxonomy is the focus of node classification, node clustering, and node regression, among other related techniques. Node classification, unlike regression, classifies nodes rather than predicting their values. It is important that representations

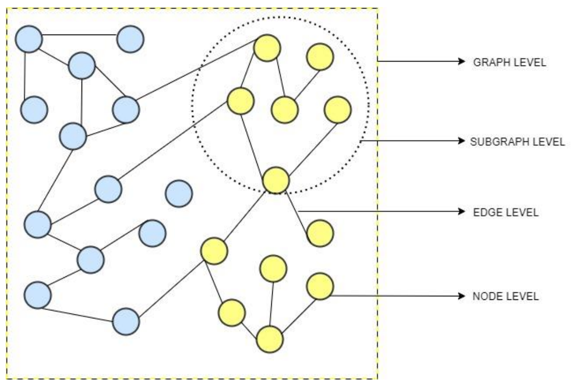

Figure 1: Types of Graph Tasks.

Notations	Descriptions
$G = (V; E)$	A graph
n	The number of nodes, $n = \|V\|$
m	The number of edges, $m = \|E\|$
$V = \{v_1; ::: ; v_N\}$	The set of nodes
$\boldsymbol{X} \in \mathbb{R}^{\boldsymbol{n\times d}}$	Graph feature matrix
$\boldsymbol{x}_v \in \mathbb{R}^{\boldsymbol{n}}$	Node v feature vector
$\boldsymbol{X}^e \in \mathbb{R}^{\boldsymbol{m\times c}}$	Feature matrix for a graph edge
$\boldsymbol{x}^e_{uv} \in \mathbb{R}^{\boldsymbol{c}}$	The feature vector of the edge
A	The adjacency matrix
$\odot$	Element-wise product.
$\|.\|$	The length of a set.
X^T	The transposition of the matrix X
$\boldsymbol{D}(i,i) = \sum_j \boldsymbol{A}(i,j)$	The degree matrix of A
$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$	The Laplacian matrix
$\boldsymbol{Q}\wedge\boldsymbol{Q}^{\boldsymbol{T}} = \boldsymbol{L}$	The eigen decomposition of $\boldsymbol{L}$
$\boldsymbol{H} \in \mathbb{R}^{n\times b}$	Node hidden feature matrix
$\boldsymbol{W}, \omega, \theta$	Learnable model parameters
S	A search space

Table 2: Notations.

optimally embed the input graph so that algebraic operations on the embedded graph accurately reflect the graph's topology. Finally, representations can be used as inputs in models to predict the property of graph elements, such as the role of proteins in an interactome network (i.e., node classification task).

Representing Edge Tasks: Among the edge-level tasks are linked prediction and edge categorization. Link prediction and edge classification entail the model predicting whether two nodes have an edge and classifying the edges. Recommendation systems are an excellent illustration of an edge problem, such as the prediction of whether or not a medicine will bind to a specific target protein (recommend the items users might like). It is indispensable to model the representation of a network to find solutions to problems involving classification, regression, and matching. At the graphs' level, gaining insights can be accomplished by classification. Examples of small molecular graphs include antibiotics and other medications. Drug discovery and toxicity profiling are two of the most prevalent graph-level operations (i.e., graph classification tasks). In this metaphor, the atoms function as the "nodes," and the chemical bonds that connect them act as the "edges." A simulation of physical occurrences, where the particles themselves serve as the vertices, inter-particle interactions serve as the edges, and so on. It is possible to break tasks at the graph level down into those at the subgraph level.

4. Learning Methods for Complex Graphical Representations

Graph learning parameters depend on data accessibility and real-world needs, like classical machine learning. According to the literature, graph-based learning tasks can be supervised, semi-supervised, unsupervised, or self-supervised.

4.1 Supervised Learning on Graphs

The intent is to use labeled data for model training (e.g., labeled nodes). In contrast to supervised learning, which relies on predetermined labels, the currently dominant strategies for generating graphs are unsupervised. Labeled instances are used before to improve the graph for downstream study tasks. Dhillon et al. (2010) examine node pair similarities using labeled points. If the manifold sampling rate is high enough, the optimum solution for a neighbourhood graph can be considered a KNN graph subgraph, as Rohban & Rabiee (2012) shows. Berton & Lopes (2014) propose Graph-based labeled instance informativeness (GBILI), which builds on the work of (Ozaki et al., 2011) by using the label information differently: they utilize a graph to determine which instances are most informative.

Berton et al. (2017) improved the Robust Graph that Considers the Labeled Instances (RGCLI) technique, based on GBILI (Berton & Lopes, 2014), to build more sturdy graphs by resolving an optimization difficulty. L. Zhuang et al. (2017) presented low-rank semi-supervised representation as a unique way to integrate labeled data in the low depiction for enhanced accuracy (LRR). The added supervised information from the created similarity graph greatly improves the subsequent label inference process.

Complex algorithms and models have been presented over the past few decades, with the majority falling into two broad approaches: regularized graph Laplacian-based approaches and graph embedding-based methods (Kipf & Welling, 2016a). Label propagation using Gaussian fields and harmonic functions (X. Zhu et al., 2003), Manifold propagation (Belkin et al., 2006), and deep embeddings (Weston et al., 2008) are all examples of the first category. Examples of works from the latter category are DeepWalk (Perozzi et al., 2014), LINE (Tang, Qu, Wang, et al., 2015), and node2vec (Grover & Leskovec, 2016). These procedures take their cue from the skip-gram model presented by (Mikolov et al., 2013), employing a wide range of different random-walk and search-based techniques. Supervised graph representation learning provides explicit labelled model optimisation. This advantage produces high-performance findings with abundant tagged data. Models understand labelled data. They improve forecasts and categorization. Professionals who can accurately and subtly predict or categorise, especially with machine learning, excel. Human annotators can offer accurate labels for high-quality training data. Performance evaluation objectivity is improved by comparing supervised methods to ground truth labels. Depending on labelled data is supervised learning's main drawback as it requires expenditure of time and funds, and obtaining enough labelled samples tends to be tedious. Supervised approaches rely on labelled data patterns, which may limit their applicability to unknown or non-conforming data distributions. When labelled data is abundant and rich in information, supervised learning approaches can produce excellent results. Prediction and classification tasks support this claim. However, label noise, bias, or an unequal data distribution in the training set may hinder supervised techniques. Supervised learning has these issues. Scalability depends on graph data and learning model complexity. Complex models with wider parameter spaces may require more computational time and resources.

4.2 Semi-Supervised Learning on Graphs

When only a small amount of tagged data is available, a semi-supervised study trains models using both input types. Due to the graph's spatial connectedness trait, semi-supervised graph learning techniques can effectively use unlabeled data. According to the manifold assumption, low-dimensional manifold nodes that are physically closer together are more comparable and hence should be given the same label. Semi-supervised learning has evolved to make use of a wide variety of techniques. Since the graph structure is compatible with numerous assumptions in semi-supervised learning, this burgeoning topic is a strong fit. Nodes reflect data instances, and edges represent similarity in graph-based semi-supervised learning. The manifold premise states that high-edge-weight nodes are comparable and have the same label category. Graph structures are simple and expressive, successfully making manifold-based graph semi-supervised learning approaches. Several semi-supervised survey studies focus on traditional methods of dealing with semi-supervised circumstances (Prakash & Nithya, 2014). Some recent efforts, including (Van Engelen & Hoos, 2020), investigate semi-supervised learning by examining graph creation and regularization. Multistep approaches cannot construct an end-to-end optimization and learning framework. Still, the methods mentioned above (supervised learning) do this by first learning the graph's embeddings, then optimizing the object functions.

Fortunately, new developments in deep learning can bridge the gaps between graph embedding and explicit learning problems in both node level and graph level learning. Since graph classification and regression aim to train a classifier or regression model to anticipate the unobserved labels or targets, these activities can be branded under (semi)-supervised learning. This section discusses recent developments in graph embeddings, an important part of Graph semi-supervised learning. Current methods for Graph semi-supervised learning are summarized in Table 3.

4.2.1 Graph Embedding

Two distinct kinds of embeddings are distinguishable in semi-supervised learning approaches that use graphs. While one describes the entire Graph, the other represents a particular node (W. L. Hamilton et al., 2017). Both embeddings aim to accomplish the same aim: to represent the item in a space with limited dimensions. Graph-based semi-supervised learning challenges rely on embedding nodes. It represents a vertex in a low-dimensional space with a local structure. The node embedding on graph $G = (V, E)$ is a mapping $h_z : p \rightarrow \boldsymbol{z}_p \in \mathbb{R}^d, \forall p \in V$ where d is smaller than $|V|$. h_z maintains graph G's node closeness metric. The function of loss used by graph embedding techniques is described in the plenary Equation 1.

$$\mathbb{C}(f) = \sum_{(x_i, y_i) \epsilon \mathbb{D}_i} \mathbb{C}_s(h(\, h_z(x_i)), y_i) + \mu \sum_{x_i \, \epsilon \, \mathbb{D}_i + \mathbb{D}_u} \mathbb{C}_r(h(\, h_z(x_i))) \qquad (1)$$

Paper	Embed-ding Ar-chitec-ture	Embedding Techniques	Loss Function	Decoder (Dec)	Comparison Formula
(S. Cao et al., 2015)	Shallow	Factorization	$\|\|Dec(z_p, z_q) - S[p,q]\|\|_2^2$	$z_p^T z_q$	A_{pq}
(Ou et al., 2016)	Shallow	Factorization	$\|\|Dec(z_p, z_q) - S[p,q]\|\|_2^2$	$z_p^T z_q$	Similarity matrix **S**
(Roweis & Saul, 2000)	Shallow	Factorization	$\sum_p \|\|z_p - \sum_q A_{pq} z_q\|\|^2$	$z_p - \sum_q A_{pq} z_q$	A_{pq}
(Belkin & Niyogi, 2003)	Shallow	Factorization	$Dec(z_p, z_q).S[p,q]$	$(z_p - z_q)\|\|_2^2$	A_{pq}
(Tang, Qu, Wang, et al., 2015)	Shallow	Random Walk	$\sum_{(p,q)\in\mathcal{D}} -\log(\sigma(z_p^T z_{qn})) - \gamma \mathrm{E}_{pn} \sim P_n(V)[\log(-\sigma(z_p^T z_{qn}))]$	$\frac{1}{1 - e^{-z} p^T z_k}$	$\mathcal{PG}(p\|q)$
(Grover & Leskovec, 2016)	Shallow	Random Walk	$\sum_{(p,q)\in\mathcal{D}} -\log(\sigma(z_p^T z_{qn})) - \gamma \mathrm{E}_{pn} \sim P_n(V)[\log(-\sigma(z_p^T z_{qn}))]$	$\frac{e^z p^T z_p}{\sum_{k\in V} e^z p^T z_k}$	$\mathcal{PG}(p\|q)$
(Tang, Qu, & Mei, 2015)	Shallow	Random Walk	$-\boldsymbol{S}[p,q] log(\mathcal{PG}(p\|q))$	$\frac{1}{1 - e^{-z} p^T z_k}$	$\mathcal{PG}(p\|q)$
(Z. Yang et al., 2016)	Shallow	Random Walk	$E_{pn} \sim P_n(V)[\log(-\sigma(z_p^T z_{qn}))]$	$\frac{e^z p^T z_p}{\sum_{k\in V} e^z p^T z_k}$	$\mathcal{PG}(p\|q)$
(Perozzi et al., 2014)	Shallow	Random Walk	$-\boldsymbol{S}[p,q] log(Dec(z_p, z_q))$	$\frac{e^z p^T z_p}{\sum_{k\in V} e^z p^T z_k}$	$\mathcal{PG}(p\|q)$
(Pan et al., 2019)	Deep	Auto-en-coder	$min_{\mathcal{G}} max_{\mathcal{D}} E_{z\sim u_z}[log \mathcal{D}(Z)] + E_{x\sim u(x)}[\log(1 - \mathcal{D}(\mathcal{G}(X, A)))]$	$z_p^T z_q$	A_{pq}
(Taheri et al., 2019)	Deep	Auto-en-coder	$\sum_{p\epsilon V} \|\|Dec(z_p) - S_p\|\|_2^2$	LSTM	s_p
(Kipf & Welling, 2016b)	Deep	Auto-en-coder	$E_{u(Z\|X,A)}[logp(A\}Z] - KL[u(Z\|X,A)\|\|p(Z)]$	$z_p^T z_q$	A_{pq}
(D. Wang et al., 2016)	Deep	Auto-en-coder	$\sum_{p\epsilon V} \|\|Dec(z_p) - S_p\|\|_2^2$	MLP	s_p
(Tu, Cui, Wang, Yu, et al., 2018)	Deep	Auto-en-coder	$\sum_{p\epsilon V} \|\|(z_p) - \sum_{p\epsilon \mathcal{N}(p)} LSTM(z_p)\,\|\|_2^2$	LSTM	s_p

Table 3: GNN-Based Semi-Supervised Learning Methods.

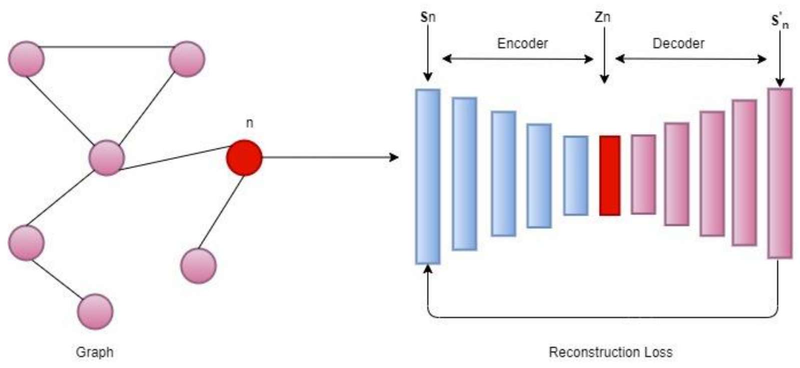

Figure 2: An auto-encoder-based approach to constructing a zi embedding of a few dimensions from a high-dimensional vector Sn.

The embedding function is denoted here by h_z. The regularization of graphs makes use of a function analogous to this one. Graph regularization is similar to traditional regularization, except it relies on node embedding results for model training rather than node attributes. The encoder-decoder framework is a generalization of all graph embedding approaches. The encoder embeds the input nodes into low-dimensional spaces, and the decoder reconstructs the original data for each node, as seen in Figure 2.

Encoder: The encoder is formally seen as a function that maps $p \in V$ nodes to $z_p \in \mathbb{R}^d$ vector embeddings. Embeddings formed in the higher-dimensional latent space have better discriminatory power.
This decoder module is also simpler to reverse-engineer into the actual function vector. From an arithmetical standpoint, we have the encoder: $V \rightarrow \mathbb{R}^d$.

Decoder: The decoder module rebuilds graph statistics from the implanted node. The decoder can try to anticipate the next neighbour $N(v)$ or row node embedding z_v in the adjacency matrix $A[u]$. Pair-form decoders are proven to predict node similarity. Mathematically, Decoder: $\mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^+$.

4.2.2 SHALLOW GRAPH EMBEDDING

It is preferred for node embeddings to reflect graph structure characteristics immediately surrounding them. The framework known as shallow network embeddings aims to optimize a neural network to generate embeddings that keep these features intact. A crucial characteristic of this feature is that the degree of resemblance in the embedding space ought to closely correspond to the degree of similarity in the source Graph (Figure 3). Different approaches are taken due to the numerous synonyms for “method.” For instance, the brief path length between two nodes can be used to describe network similarity, whereas the dot product can be used to define embedding space similarity. More advanced approaches can capture more detailed similarity measurements, allowing a more accurate reflection of the network structure. The semantic meaning of edges (i.e., relation types) can be useful to incorporate into learned embeddings for heterogeneous graphs. Since each relation is split into many embeddings representing the head node, the tail node, and the relation type, knowledge graph embedding methods (Bordes et al., 2013; Dong et al., 2017; Nickel et al., 2011; Z. Sun et al., 2019; Trouillon et al., 2016; B. Yang et al., 2014) generate similarity metrics by considering a wide variety of node relations.

Factorization: For the class of factorization-based approaches, a matrix is factorized to provide the node embedding, with the matrix specifying the relationship between each pair of nodes. When constructing a similarity network, this matrix typically includes essential structure information like normalized Laplacian and adjacency matrices. The factorization of such matrices can take many forms depending on several factors. The eigenvalue decomposition of the normalized Laplacian Matrix makes sense because it is also positive and semi-definite.

Random Walk: A useful method for approximating several properties of a Graph, such as its nodes' centrality (Newman, 2005) and similarity (Fouss et al., 2007). One of the attributes that may be approximated is the degree to which two nodes are similar. Therefore, random node embedding strategies are useful when just a portion of the graph is available or where the graph is too big to be effectively managed. According to one definition (Perozzi et al., 2014), "similarity" is defined as "co-occurrence in a series of random walks of length k." It is possible to reproduce random walks on graphs through various approaches, such as depth-first search algorithms, breadth-first search walks, and hybrids (Grover & Leskovec, 2016).

Shallow embedding approaches have performed well on many semi-supervised tasks, but researchers have struggled to overcome their drawbacks. Shallow embedding generates only one graph embedding. It also does not assess node properties, and few common parameters are used. Since the encoder creates a fresh vector at each node, it requires its unique set of parameters. Shallow embedding approaches exclude node characteristics, another major concern. Encoding may contain extensive feature information. Semi-supervised learning uses this since each node contributes feature information. Shallow embedding techniques have always relied on a transductive approach (W. L. Hamilton et al., 2017). Nodes identified after the training phase cannot generate their embeddings. Because of this limitation, inductive applications cannot use shallow embedding techniques.

4.2.3 DEEP GRAPH EMBEDDING

Recently, many deep embedding approaches have been created to circumvent these limits. Embedding something shallow requires a different set of skills than embedding something deep. In this case, the encoder would consider the properties of the graph in addition to the graph's structure. When doing semi-supervised learning tasks with a transductive setup, the node embeddings are utilized for training a top-level classifier, which then makes predictions regarding the class labels of unlabeled nodes. Auto-encoder-based approaches diverge from shallow embedding approaches in two key respects; they rely on Deep Learning (DL) models and employ a unary decoder rather than a paired one. As shown in Figure 2, the goal of the Auto-encoder based method is to encode each node according to the linked vector s_i and then restructure it using the embedding findings, expecting the restored one to be as close to the initial as possible.

4.3 Graph-based Unsupervised Learning

Data samples and annotated labels from the real world are presented when working in a supervised or semi-supervised scenario. In the unsupervised context, labeled samples are unavailable; therefore, the loss function must infer the properties of the Graph's nodes, edges, and topology. Perhaps the most emblematic task in unsupervised graph learning is the link prediction problem, which

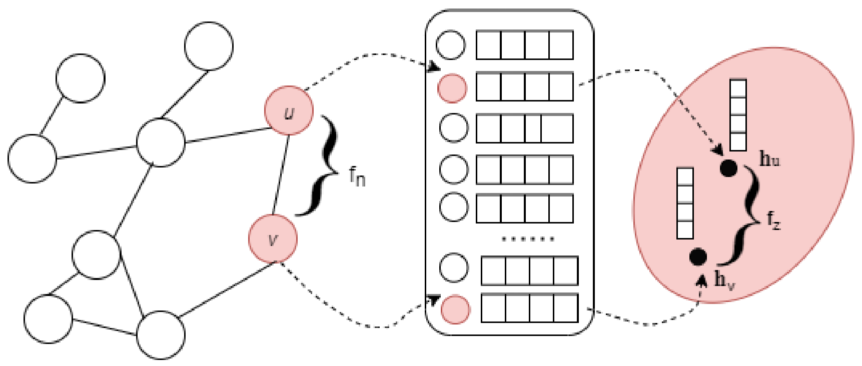

Figure 3: Shallow Network Embeddings.

involves predicting the existence of unseen edges in a graph. Table 4 represents contrastive learning, auto-encoders, and random walks as the basis of several unsupervised graph learning methods.

4.3.1 Contrastive Learning

The unsupervised learning environment uses contrastive learning to acquire expertise in graph representations. Deep Graph Infomax (DGI), which is an extension of the deep InfoMax described by (Hjelm et al., 2018), was proposed by (Velickovic et al., 2019). DGI exploits the shared information amid the representations of the nodes and the graph. Infograph (F.-Y. Sun et al., 2019) simplifies learning graph depictions by optimizing the information gained across graph-level and subgraph-level illustrations of varying sizes, like nodes, links, and triangles. First-order adjacency matrices representation and graph diffusion are compared in (Hassani & Khasahmadi 2020), which achieves State-of-the-art (SOTA) results on various graph learning problems. (Okuda et al., 2021) recently employed an unsupervised graph representation study to identify generic objects and create a localization strategy for accumulating images of individual objects.

4.3.2 Graph-based Auto-Encoders

In their ground-breaking work, Kipf & Welling (2016b) introduced the Graph Auto-encoder (GAE), an enhanced auto-encoder designed to handle graph-structured data effectively. This paradigm's fundamental training principle is centered on a loss computation that effectively contrasts the original adjacency matrices with their carefully recreated counterparts.

The emergence of the Variational Graph Auto-encoder (VGAE) methodology reveals a fundamental role played by variation in the learning process. In innovation, two parallel investigations were conducted by C. Wang et al. (2017) and J. Park et al. (2019). The researchers aimed to deviate from using the adjacency matrix and instead focused on accurately reproducing feature matrices in their investigations. The minimized Graph Autoencoder (MGAE) is a novel approach that utilizes minimized noise removal principles to capture the characteristics of flexible nodes effectively. In a notable advancement, J. Park et al. (2019) endeavored to enhance the breadth of decoding capabilities. The researchers improved the decoding process by strategically incorporating Laplacian sharpening, effectively revealing concealed states with exceptional precision. The culmination of this transformation resulted in the emergence of an asymmetrical Graph Auto-encoder known as

Paper	Feature retrieval	Method	Task	Important Functions
(F.-Y. Sun et al., 2019)	Contrastive	K-layer Graph Convolutional Network (GCN)	Node Classification (NC), Link prediction (LP), Graph Classification (GC)	Graph-level depictions.
(Hjelm et al., 2018)	Contrastive	Neural Network (NN)	NC	New method for representation study without supervision.
(Velickovic et al., 2019)	Contrastive	CNN	NC	Increasing node-graph correlation
(Hassani & Khasahmadi, 2020)	Contrastive	GCN	NC, GC	Acquiring knowledge of node and graph-level illustrations.
(Pan et al., 2018)	Graph Auto-encoder (GAE)	GCN	LP, GC	Low dimensional modelling of graph-structured data for graph analytics.
(J. Park et al., 2019)	GAE	GCN	NC, LP, GC	A graph's irregular regions were used to extract low-dimensional latent representations.
(G. Cui et al., 2020)	GAE	GCN	NC, LP	Vectorize node properties and network structure via graph embedding.
(C. Wang et al., 2017)	GAE	GCN	GC	Marginalized graph auto-encoder clustering technique.
(Kipf & Welling, 2016b)	GAE	GCN	LP	The underlying representations of undirected graphs that are comprehensible are taught.
(I. Li et al., 2021)	GAE	NN	NC	Mastering knowledge acquisition processes in both familiar and distant domains.
(Dong et al., 2017)	Random Walk (RW)	NN	NC	Learning a Heterogeneous Representation of Network Nodes.
(Adhikari et al., 2018)	RW	NN	NC	Create a subgraph embedding issue
(Tang, Qu, Wang, et al., 2015)	RW	NN	NC, LP	Embeddings of low-dimensional nodes in extremely large networks.

Table 4: GNN-based unsupervised learning methods.

the Graph Convolutional Auto-encoder using Laplacian smoothening (GALA). GALA is anticipated to pave the way for new advancements in the ever-evolving landscape of graph-based auto-encoders.

4.3.3 Random Walk

Random walks have also been shown to compensate for structural equivocation, which occurs when two vertices have similar local structures and similar embeddings, and when two vertices have similar embeddings and belong to the same community (Du et al., 2018). Perozzi et al. (2014) efficiently capture large-scale networks' graph structure through the Deepwalk approach. Du et al. (2018) and Perozzi et al. (2014) revealed that random walks and contemporary language modeling representation learning methods might produce high-quality vertex representations for downstream learning tasks like vertex and edge prediction. Adhikari et al. (2018) and Dong et al. (2017) have expanded random walk-based approaches to capture vertex representations in heterogeneous graphs and subgraph embeddings.

4.4 Graph-based Self-Supervised Learning

Self-supervised learning is a cutting-edge field of study in DL. In semi-supervised learning, the reliance on hand labeling for identification, and ground-truth labeling, which requires intensive processing, prediction accuracy, and inadequate protection in contrast to adversarial attacks, are all addressed (X. Liu, Zhang, et al., 2021). By teaching a model to complete carefully crafted "PretextTasks," self-supervised learning overcomes the abovementioned drawback. Self-supervised learning performs "downstream tasks" (Y. You et al., 2020) such as node, edge, and graph level tasks better because it learns more generic representations given unmarked input. Table 5 presents an overview of the current approaches utilized in graph self-supervised learning.

4.4.1 PRETEXT TASKS

Self-supervised learning relies heavily on the model's ability to perform well on downstream tasks; hence their development is crucial. We classify the pretext task as follows: Masked Feature Regression (MFR), Auxiliary Property Prediction (APP), Same-Scale Contrasting (SSC), Cross-Scale Contrasting (CSC), and Hybrid Self-supervised Learning (HSL).

Masked Feature Regression. The computer vision task of picture inpainting inspired a new category of pretext challenges called Masked Feature Regression (MFR) (J. Yu, Lin, et al., 2018). The goal of this technique is to change the attribute of a node or edge to zero or another number. The primary objective is to unearth the original node/edge information prior to when GNNs obfuscate the data. (Y. You et al., 2020) node-based MFR approach allows GNN to obtain features from environmental data. Reconstructing raw features from noisy input data, ideal input data, and noisy feature embeddings are context problems used to acquire robustly generalized models.

Table abbreviations: Node Classification (NC), Link Prediction (LP), Graph Classification (GC), Augmentation Same-Scale Contrasting (ASSC), Context same-scale contrasting (CSSC), Cross-Scale Contrasting (CSC), Classification Auxiliary Property Prediction (CAPP), Hybrid Self-Supervised Learning (HSSL), Masked Feature Regression (MFR), Pre-training and Fine-tuning (PT&FT), Collaborative Learning (CL), Unsupervised Representation Learning (URL).

Paper	Pretext Task Category	Mode of Training	Method	Task	Important Functions
(Y. Zhu et al., 2021)	ASSC	URL	GCN NC	NC	Graph contrast learning with personalized enhancement.
(H. Zhang et al., 2020)	ASSC	URL/CL	GCN/GIN	GC	Iteratively performed self-distillation with graph augmentations.
(Qiu et al., 2020)	ASSC	URL/ PT&FT	GIN	NC, GC	Random walks supplement subgraphs, and artificial positional node embeddings are node characteristics.
(J. Zeng & Xie, 2021)	ASSC	URL/ PT&FT/CL	NN	GC	Marginalized graph auto-encoder clustering technique.
(J. Wu et al., 2021)	ASSC	PT&FT	GCN	LP	GCN recommendation reliability and durability improvement.

(Choudhary et al., 2021)	ASSC	URL	GCN	NC, LP, GC	Automating several pretext tasks.
(Che et al., 2021)	ASSC	PT&FT	GCN	NC, LP, GC	Graph representation study.
(Jin et al., 2020)	CSSC	PT &FT /(CL)	GCN	NC	Building domain-specific pretext tasks using unlabelled data reduces DL's need for expensive annotated data.
(Kipf & Welling, 2016b)	CSSC	URL	GCN	LP	Design explainable undirected graph implicit representations.
(Z. Peng, Dong, et al., 2020)	CSSC	CL	NN	NC, LP	Established a subtask to forecast meta-paths using node embeddings.
(Kim & Oh, 2022)	CSSC	CL	GAT	NC	Graph attention system for noisy graphs.
(J. Zhang et al., 2020)	CSSC	PT&FT	NN	NC	Pre-training a graph-based transformer model involves structure recovery.
(Y. You et al., 2020)	CAPP	PT&FT/CL	GCN	NC	Pre-computed cluster index for node clustering.
(Rong et al., 2020)	CAPP	PT&FT	NN	NC, LP, GC	Acquiring rich molecular structure and semantic knowledge from massive untagged molecular data.
(K. Sun et al., 2020)	CAPP	CL	GCN	NC	Train a pseudo-label encoder architecture.
(Z. Hu et al., 2019)	CAPP	PT&FT	GCN	NC, LP, GC	GNN-pretrained structural feature extraction.
(Subramonian, 2021)	CSC	URL	GCN	GC	Discovering pattern motifs iteratively and improving graph-motif embeddings.
(Hjelm et al., 2018)	CSC	URL	GCN	NC	Learning graph-structured node representations.
(Q. Sun et al., 2021)	CSC	CL	GCN	GC	Differentiating the subgraph between graphs using the local subgraph and global Graph representations.
(Ren et al., 2020)	CSC	URL	GCN/GAT	NC	Enhanced local and global shared knowledge for representation learning in heterogeneous graphs.
(C. Park et al., 2020)	CSC	URL	GCN	NC, LP	Embedding a multiplex network with attributes.
(J. Cao et al., 2021)	CSC	URL	NN	NC, LP	Maximizing mutual information completes a bipartite graph embedding. Maximum knowledge sharing.
(Opolka et al., 2019)	CSC	URL	GCN	NC	Training embeddings to solve node-level regression.

(Hassani & Khasahmadi, 2020)	CSC	URL	GCN	NC, GC	Optimizing node mutual information for graph learning.
(F.-Y. Sun et al., 2019)	CSC	URL	NN	GC	Improving substructure-graph representation knowledge to improve graph-level tasks.
(Jiao et al., 2020)	CSC	URL	GCN	NC	Graph representations learning uses good compatibility among key nodes and their observed subgraphs to capture regional structure information.
(P. Wang et al., 2021)	CSC	PT&FT	Shallow NN	LP	Contextual node forecasting in heterogeneous networks.
(Jin et al., 2021)	HSSL	CL	GCN/HGCN	NC, GC	Effectively leverage several pretext tasks automatically.
(S. Li et al., 2021)	HSSL	URL	GCN	NC	Identifying Ethereum Phishing Scams.
(Wan et al., 2021)	HSSL	CL	GCN/HGCN	NC	Generative and contrastive Convolutional graph network
(J. Lin et al., 2021)	HSSL	URL	GCN	NC, LP, GC	Categorization of fundus images using several labels.
(W. Hu et al., 2019)	MFR	PT&FT	GCN	NC	An innovative method for pretraining GNNs on nodes and graphs.
(Manessi & Rozza, 2021)	MFR	CL	GCN	NC	Develop GNN models using a multi-tasking approach.

Table 5: GNN-based self-supervised learning methods.

Auxiliary Property Prediction In addition to the aforementioned MFR methods, there are further methods that investigate the fundamental attribute data of nodes and edges, as well as the graph topology, to generate new pretext jobs for the self-supervision models. Additional property prediction methods include regression and categorization (Y. Liu et al., 2022). In contrast to MFR, the regression-based method emphasizes context problems, such as the prediction of graph properties based on features and structures, rather than on numerical analysis. One such local structure-aware pretext job that considers both local and global structure information is the NodeProperty pretext task proposed by (Jin et al., 2020). A method called Distance2Cluster was put out by (Jin et al., 2020) to determine how far unlabeled nodes are from preset clusters in the graph. This method causes the node illustration to reflect the training's overall location. PairwiseAttrSim, also proposed by (Jin et al., 2020), attempts to minimize the discrepancy between a pair of nodes' similarity value and their feature matching on the representation dispersion. It is based on improving feature modification for local structures to avoid over-smoothing.

Creating pseudo labels to aid model training is taken on by classification-based methods instead of regression-based techniques. Multi-Stage Self-Supervised (M3S) is a technique proposed by K. Sun et al. (2020) that involves training an encoder design to assign dummy labels to unlabeled nodes. Over the entirety of the training procedure, the DeepCluster (Caron et al., 2018) network is the backbone of this approach. Y. You et al. (2020) created the Node Clustering approach, which is quite similar and uses self-supervised labels in the form of a previously computed cluster index.

Same-Scale Contrasting: Unlike the first two methods, which build upon a single node in the graph, contrastive learning techniques improve by continually practicing consensus across two nodes (such as a single node). This strategy aims to increase the number of positive pairs or samples that share the same semantic data while minimizing the number of negative pairs. Same-scale contrast (SSC) splits two graph parts by comparing them on a similar scale, such as graph-to-graph and node-to-node. Additionally, SSC-based procedures are classified into two groups (context and augmentation) depending on the idea of good and adverse pairings.

Based on the context-based same-scale contrasting (CSSC), the contextual nodes are brought closer together in the embedding space. In a Graph, the contextual nodes are typically located close to one another. The Homophily hypothesis (McPherson et al., 2001) forms the basis for the perception that entities with similar semantic data should be related. Creating collections of nodes with comparable semantic data via a random walk defines a context. Positive pairs refer to nodes closer together, whereas opposing pairs refer to node pairs obtained through negative sampling.
In the past few years, contrastive visual feature learning has significantly progressed (K. He et al., 2020). These developments also drive augmentation-based same-scale contrasting (ASSC), which creates fresh augmented examples for real-world data sets. ASSC's definition of the data augmentation process is essential. Positive pairs of enhanced samples from genuine data are treated as such, whereas negative pairings of augmented samples from various actual data are so-called. These methods utilize mutual information (MI) estimate (Hjelm et al., 2018) and InfoNCE for estimation (Oord et al., 2018) and come under this category. For node-level tasks, Qiu et al. (2020) introduced a technique known as Graph contrastive coding (GCC), concentrating on universal unattributed graphs. This method augments subgraphs using random walks and restarts for each node before using positional node embeddings that were purposefully created as node features. Graph Contrastive Representation Learning (GRACE) by Y. Zhu et al. (2020) learns two augmentation procedures by deleting masked node characteristics and edges to improve graph representation. External and internal negative pairings are contrasted.

Cross-Scale Contrasting: Cross-Scale Contrasting (CSC) is an alternative contrasting learning method to SSC used to learn representations of graphs at different scales (node-subgraph, node-graph contrasting). The graph or subgraph summary is often obtained via a readout function. The goal of these techniques is the same as that of ASSC, which is to maximize mutual information. Hjelm et al. (2018) suggested utilizing the Jensen-Shannon divergence to estimate mutual information. (Velickovic et al., 2019) created Deep Graph Infomax (DGI) to maximize the mutual information between the top-level graph summary and the related patch representations to learn node representations. DGI obtains negative samples of each Graph by deliberately contaminating its node attributes. The reliability of the Graph's architecture is preserved through DGI. Also along these lines is Multi-view representation learning on graphs (MVGRL), a technique developed by (Hassani & Khasahmadi, 2020) to pay attention to multi-view contrasts. This method separates the perspectives of the original graph structure and the diffusion. The goal is to maximize the two-way information flow between various perspectives represented in different ways. Jiao et al. (2020) have presented the Subgraph Contrast (Subg-Con) to contrast the subgraph context and the node embeddings to comprehend the regional spatial relationships of the graph.

Hybrid Self-supervised Learning: Few proposed systems effectively merge pretext tasks from several domains into multitasking learning algorithms. The Generative Pre-Trained GNN (GPT-GNN) (Z. Hu et al., 2020), based on the multi-feature regression (MFR) algorithm, is a pretraining approach for the GNN that replaces the edge prediction problem with the production of new graphs. Z. Peng et al. (2020) devised a method called graphical mutual information (GMI) to include and optimize shared knowledge between the basic feature of a nearby node and the node embedding. At the same time, it optimizes the edge similarity metric for graph representation learning (node embedding of two adjacent nodes). Bert (J. Zhang et al., 2020) offers a node feature reconstruction methodology. The MFR and graph structure retrieval are used to pre-train a transformer-based graphs model. To acquire knowledge about context-based SSCs and augmentation as self-supervised study signals, Wan et al. (2021) used downstream node classification tasks.

4.4.2 SELF-SUPERVISED TRAINING STRATEGIES

Based on the relationship between pretext tasks, downstream tasks, and graph encoders, self-supervised learning techniques can employ one of three training strategies: Collaborative Learning (CL), Pretraining and Fine-tuning (PT & FT), or Unsupervised Representation Learning (URL). For Collaborative study, the encoder and the pretext task are trained together. Self-supervised learning is combined with task loss function error to form the combined loss function. A trade-off hyperparameter can adjust the relative importance of individual errors in determining the total error. It's being passed off as practice for multitasking or as an attempt to standardize the work that comes later. In Pretraining and Fine-tuning, the encoder and the pretext jobs are taught beforehand; this can be thought of as setting the encoder's default values. Also, the prediction head and pre-trained encoder are simultaneously tuned per the instructions of precise downstream jobs. Like PT and FT, Unsupervised Representation Learning completes pretext tasks and pre-trains the encoder at the outset. However, the second phase is distinct due to the fixed encoder parameters. URL is more challenging than other training approaches because each encoder is trained separately.

5. Graph Neural Networks

Graph Neural Networks (GNNs) is a Deep Learning Neural Networks (NN) appropriate for analyzing graph-structured data. It is analogous to a graph, with nodes representing the data to be studied and edges representing the connections between them. Graph theory and deep learning underpin GNNs. The graph neural network is a model class that learns model class that learns data structures and graph tasks using graph representations. Feature propagation and aggregation in GNNs improve graph representations. Graphs are frequently used in representation learning tasks, where the Graph includes some domain knowledge that, while not explicitly stated in the graph structure, can be learned through instances. In a nutshell, graph neural networks extract more information from data with less organized labeling, iteratively spreading neighbourhood information until convergence (recognizes the structure of a given graph or node and learns to represent it).

These computationally intensive investigations belong to recurrent Graph neural networks (RecGNNs) (Z. Wu et al., 2020). Despite efforts to overcome these weaknesses, many scholars have used the achievement of CNNs in computer vision to advance new convolution algorithms related to graphs, such as convolutional GNNs (ConvGNNs). Spectral-based strategy models (Y. Zhang et al., 2019) and spatial-based strategy models (Micheli, 2009; Niepert et al., 2016; H. Peng et al., 2020) can be used to categorize ConvGNNs. In the spectral-based approach, models are

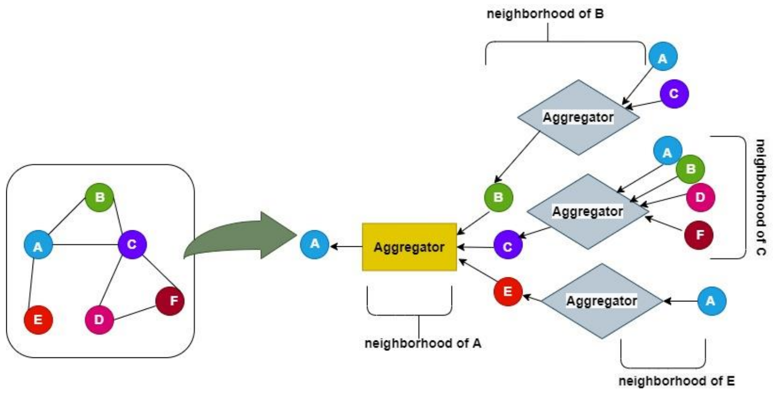

Figure 4: Neighbourhood Aggregation.

constructed using spectral graph theory, and eigen decomposition of Laplacian graphs filters signals on a graph (Oellermann & Schwenk, 1991). In the spatial-based method, the graphs themselves undergo the convolution processes, and the convolution itself is embodied as a blend of feature exchange from the neighbours of each node. In contemporary GNN models, the illustration vector of a node is calculated by iteratively aggregating and altering the illustration vectors of its neighbours; this approach is also known as neighbourhood aggregation (or message forwarding) (Gilmer et al., 2017).

The goal is to teach a feature function $f(G)$ to recognize patterns in G. The function receives the following input: Matrix representation of the Graph G structure, normally in the shape of an Adjacency matrix **A** (or other functions) and yielding node-level or graph-level output, and a feature depiction x_v for each node $v \in V$ summed in a feature matrix **X**. The application must specify the vectors used to represent the node features in this calculation. In a database context, they may take the form of word vectors, pixel values, or a hybrid of image characteristics and word vectors to describe scenes. The aim is to acquire a good vector illustration over time; hence aggregation is performed recursively over neighbouring nodes. After aggregation by k rounds $(k = 1,2,\ldots,K)$, the representation of a node gets the depth information inside the k-hop network neighbourhood that node (Figure 4).

The formal description of the feature vector of a node v in a GNN model at the k-th iteration, where k is the number of layers in the model, is as follows, given a graph G:

$$\boldsymbol{h}_v^{(k)} = ACT^{(k)}(\boldsymbol{W}_k AGG\big(\{\boldsymbol{h}_u^{k-1}, \forall\, u \in N(v)\}\big), \gamma \tag{2}$$

$$\gamma = \boldsymbol{B}_k \boldsymbol{h}_v^{k-1}, \boldsymbol{h}_v^0 = x_v \tag{3}$$

where $k \in [1, K]$ shows number of iterations, $\boldsymbol{W}_k$ is a learnable weight matrix in the k-th layer, $\boldsymbol{B}_k$ is a bias, $\boldsymbol{B}_k \boldsymbol{h}_v^{k-1}$ v is a self-loop stimulation for node v, $ACT^{(k)}(.)$ is a non-linear stimulation function in the $k-th$ layer, e.g., Rectified Linear Unit (ReLU), sigmoid, and $AGG^{(k)}(.)$ is a combination function in the k-th layer, e.g., max-pooling, sum. The layer's output typically denotes each node's ultimate representation. Applying pooling to layer representations is useful for activities at the graph level. Optimal graph representation learning by a GNN model relies on the careful selection of $ACT^{(k)}(.)$ and $AGG^{(k)}(.)$. The main challenge is to establish the aggregate schema and aggregation order for each node. Several GNN alternates with distinct neighbourhood combinations and graph-level pooling schemes have evolved in recent years. This includes graph convolutional network (GCN) (Kipf & Welling, 2016a), graph isomorphism network (GIN) (K. Xu, Hu et al., 2018), graph attention network (GAT) (Velickovic et al., 2017), and local extrema graph neural network (LEConv) (Ranjan et al., 2020). These GNNs have demonstrated prevailing performance in a variety of tasks, including semantic segmentation (L.-Z. Chen et al., 2021), node classification (Z. Liu et al., 2019), and recommendation systems (C. Huang et al., 2021; Pang et al., 2022; J. Zhang et al., 2019), among others.

5.1 General Layout and Structures of GNN

Here, we look at GNN models from the perspective of a creative director. First, the study covers the big picture of creating a GNN model, as depicted in Figure 5. Subsequently, we discuss each step-in great depth in the sub-sections below. As a rule, four stages are involved in designing a GNN model for a particular task on a specific graph type. a) determining the application's graph structure, b) characterizing the graph's kind and scale, c) developing a suitable loss function, and d) developing a model incorporating computational modules. n the following paragraphs, we'll discuss the requirements for different design stages.

5.1.1 Graph Structure

We must first ascertain the application's graph structure. Scenarios typically fall into one of two categories: structural or non-structural. Many systems, including physical systems, chemicals, knowledge networks, etc., exhibit graph structures. In addition to being present in structural contexts, implicit graphs also occur in non-structural domains. Thus, we must first construct the Graph from the specified task, such as creating a fully linked "word" graph for text or the scene graphs for an image. Once we have the Graph, we can determine which GNN model will work best.

5.1.2 Types of Graphs

The type and size of the graph should be determined when we understand the Graph's structure within the context of the relevant application. More information about the graph's nodes and edges can be stored in a complex graph. There are a few common ways to classify graphs.

a) Directed or Undirected Graphs: Whether a graph is directed or undirected depends on the direction of its edges, but directed graphs carry more content than undirected graphs. Undirected edges in a graph are considered the same way as two edges in directed graphs.

b) Homogeneous or Heterogeneous Graphs: In contrast to heterogeneous graphs, which feature a wide variety of node and edge types, homogeneous networks have a uniform structure. Researching the different sorts of nodes and edges is essential for understanding heterogeneous graphs.

c) Static or Dynamic Graphs: A dynamic graph is one in which the input attributes or graph structure vary over time. Temporal information in dynamic graphs should be carefully analyzed. Since these classes are orthogonal to one another, they can be combined to form a dynamically directed heterogeneous graph. Signed graphs and hypergraphs are two further types of graphs that serve specialized purposes. Both "small" and "large" graphs look the same to the naked eye. Due to the ever-evolving nature of computing, the criteria are constantly evolving.

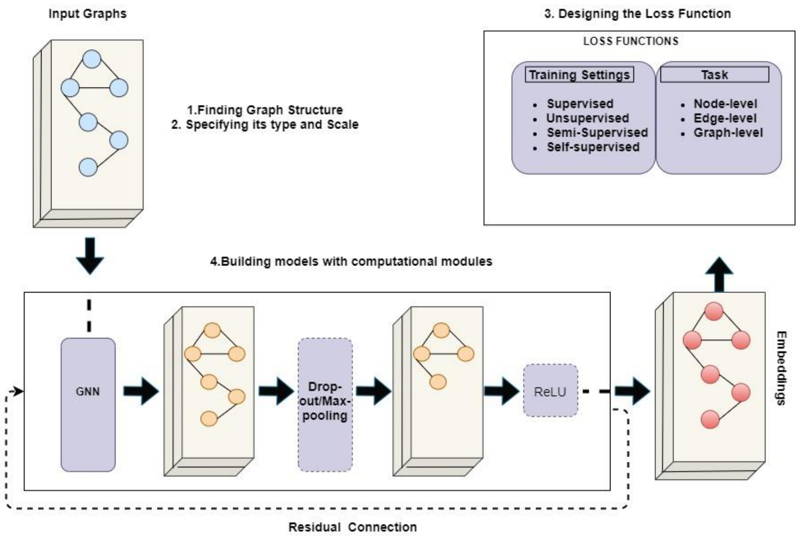

Figure 5: A GNN model's overall design workflow.

5.1.3 DESIGN LOSS FUNCTION

It is the task at hand and the parameters of the training set that inform the optimization of the loss function. There are essentially three categories of graph-learning initiatives.

In node-level tasks, such as node classification, node clustering, node regression, and so on, the qualities of individual nodes are emphasized. The objective of node classification is to divide nodes into manageable chunks. In node clustering, similar nodes are grouped for easier analysis. A valid value is predicted for each node, so this process is called "node regression."

Edge-level tasks include edge classification and link prediction, which entails the model making predictions about the existence of edges between pairs of input nodes and identifying the types of edges in a network. Graph-level tasks like matching, classification, and regression depend on the model learning appropriate graph representations**.** From a supervisor's vantage point, graph learning activities are broken up into diverse learning settings. Training in a supervised setting allows access to labeled data. At the same time, a semi-supervised design provides many untagged nodes and a small number of tagged nodes for the training procedure. In an unsupervised setting, the model can only learn to recognize patterns in data that have not been labeled.

5.1.4 BUILD MODEL USING COMPUTATIONAL MODULES

Using the regular computational modules, we start building the model as follows:

a) Propagation Module: This module spreads data between nodes so that feature and structural data can be added to the aggregated data. Convolution and recurrent operators are widely employed in this module to obtain neighbour information. Furthermore, the skip connection is often used to get information from previous node illustrations and deal with the issue of over-smoothing.

b) Sampling Module: When graphs are large, sampling modules are typically necessary for graph propagation. Combining the sampling and propagation modules is common.

c) Pooling Module: Pooling modules collect data for representation when highly ranked sub-graphs or graphs are required.

Graph-based learning systems can demonstrate different degrees of computational complexity depending on the specific algorithms and approaches utilized. Random walk-based methods, graph convolutional networks, graph attention networks, and graph neural networks are widely acknowledged methodologies for gaining insights into graph structures. The complexity of these approaches can be adjusted by several methods, depending on criteria such as the model's scale, the number of parameters, and the graph's size. The computational cost of graph representation learning methods can exhibit significant variability, contingent upon the particular approach employed and the characteristics of the graph dataset, including its size and complexity. Specific techniques may show high computational complexity and require substantial memory resources, whereas others may be more streamlined and adaptable. GCNs (Kipf & Welling, 2016a) are a type of neural network architecture specifically designed to operate on graph-structured data. The forward pass in GCNs entails the aggregation of information from adjacent nodes and the subsequent application of a neural network layer. The computational complexity of a singular layer is commonly expressed as $O|V| + |E|$, where $|V|$ denotes the cardinality of the set of nodes and $|E|$, represents the cardinality of the set of edges within the graph. The utilization of multiple layers in GCNs enables the learning of hierarchical representations. The computational complexity of stacking L layers is denoted as $O(L * (|V| + |E|)))$. In order to operate effectively, GCNs necessitate the retention of three key components: the adjacency matrix or an edge list, node features, and model parameters, which include weights and biases. The memory demand is contingent upon the dimensions of the graph and the quantity of model parameters. The attention mechanism is employed by GATs (Velickovic et al., 2017) to facilitate information aggregation by assessing the relative significance of neighbouring nodes. The formula $O(d * |V|^2)$ is commonly used to express the computational complexity of the attention mechanism. In this notation, d denotes the number of attention heads, while $|V|$ indicates the number of nodes. In order to enhance their overall effectiveness, GATs commonly employ several attention heads. The overall complexity is augmented to $O(H * d * |V|^2)$, where H represents the total number of attention heads. Similar to GCNs, GATs require a designated storage space for the adjacency matrix or edge list, as well as the node attributes and model parameters. The required memory capacity is contingent upon both the dimensions of the graph and the quantity of model parameters.

Sampling is a crucial aspect of Graph Sample and Aggregated (GraphSAGE), as it enables the efficient processing of large graphs. GraphSAGE achieves this by performing its operations on subgraphs that have been carefully selected through the sampling process. The computational complexity is directly proportional to both the size of the sampled subgraphs and the number of layers integrated into the model. Its algorithm employs a diverse set of aggregation functions, including mean, max, and Long Short-Term Memory (LSTM), to collect and integrate information from neighbouring nodes at each layer. The aforementioned data is subsequently utilized in the subsequent stratum. The computational complexity of aggregation is contingent upon the specific function employed and the magnitude of the neighbourhood (W. Hamilton et al., 2017). The management of large-scale graph datasets presents several challenges, encompassing constraints related to memory capacity, processing time limitations, and issues pertaining to scalability. As the size of the graph increases, the computational requirements in terms of time and memory can experience a significant and rapid escalation. Consequently, conventional methods become inefficient or infeasible to execute.

In order to tackle the computational difficulties associated with graph datasets of significant scale, scholars have put forth a range of optimization methodologies.

Sampling Techniques: Researchers employ sampling approaches to extract subsets of the graph that are representative of the complete chart, circumventing the need to process the entire network. Utilizing these methodologies can significantly reduce the computational load while maintaining the integrity of the graph's overall structure. Sampling techniques are used to select a subset of nodes and edges from a large chart to reduce computational load while preserving the overall design and properties of the network. In research, many sampling methods are frequently employed, such as node, edge, and neighbourhood(X. Liu, Yan, et al., 2021; H. Zeng et al., 2019).
Parallelization: is a technique that can yield advantages for graph algorithms, especially when applied to modern computer architectures like GPUs and TPUs. Parallelizing graph operations can offer advantages such as enhanced computational speed and increased efficiency. Parallelization involves subdividing graph computations into smaller tasks that can be executed simultaneously on multiple processing units, including graphics processing units (GPUs) and central processing units (CPUs). This method facilitates accelerated computations and enhances efficiency (Chiang et al., 2019; D. Zhang et al., 2020).

Approximation methods can be utilized in certain situations to obtain approximate solutions and minimize computational requirements. These estimations may be deemed sufficient for specific applications, particularly when dealing with large-scale graphs. Approximation methods possess the capability to provide approximate solutions while concurrently minimizing the computational resources required, rendering them well-suited for application in scenarios involving large-scale graphs. In pursuit of enhanced computational efficiency, these strategies sacrifice accuracy.
The concept of "sparsity exploitation" pertains to the utilization of the observation that the majority of graphs exhibit a relatively low density of edges relative to the total number of potential connections. The utilization of sparsity has the potential to yield significant reductions in computational requirements. The utilization of distributed frameworks in computing is another method. These frameworks are employed to handle exceedingly large graphs that surpass the storage capacity of a single machine's memory. This is achieved by distributing the graph processing tasks across multiple nodes within a cluster (Weber et al., 2018) (Abu-El-Haija et al., 2020).

In summary, the computational complexity of graph-based learning methods exhibits significant variability, and the analysis of extensive graph datasets requires meticulous deliberation of optimization strategies, parallelization techniques, approximation methodologies, and the integration of dedicated hardware accelerators. Academic researchers consistently seek novel methodologies to surmount these challenges and render graph-based learning feasible and adaptable to practical applications (C. Li et al., 2023) (Lee et al., 2023).

6. Applications

Unlike traditional neural networks, which function with arrays, GNNs can operate on graphs. Graphs' rising popularity can be attributed to their ability to depict complex situations in the actual world. Information within the applications is structured. Social networks, chemical structures, web connection data, and other unstructured data are studied by modeling them as graphs. Different sorts of unstructured data and testing rely on this information.-Although node classification, graph classification, graph generation, network embedding, and spatial-temporal graph prediction are all examples of GNNs, they aim to do slightly different things. This study highlights several research-based applications in the following fields:

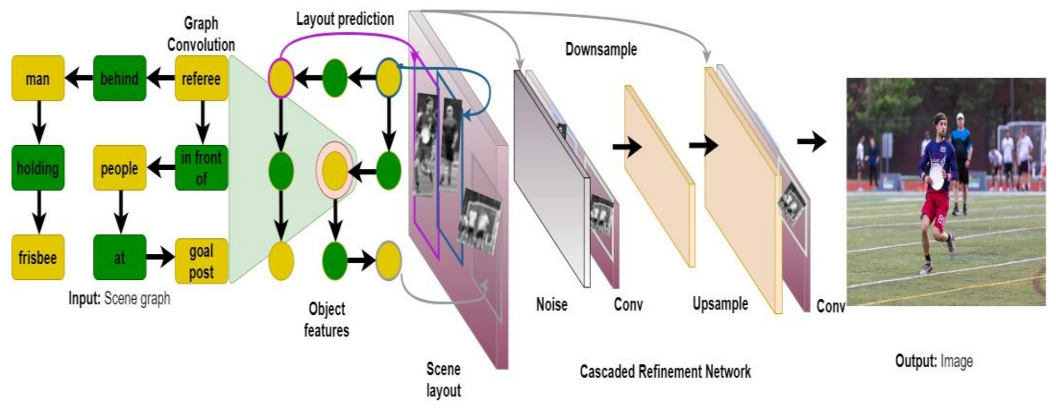

Figure 6: Scene Graph Image Representation.

Computer Vision (CV): In computer vision, GNN activities include the generation of scene graphs, the classification of point clouds, and the detection of actions. Understanding the context in which things are placed helps in visual scene interpretation. Models that generate scene graphs aim to analyze a scene by graphing components and their semantic relationships (Dhingra et al., 2021). For semi-supervised image classification, where both labeled and unlabeled image occurrences are used, the graph neural network simulates the fine-grained region correlations and increases classification performance (Hong et al., 2020; Luo et al., 2016; Satorras & Estrach, 2018). Captioning images was demonstrated by (X. Yang et al., 2019) using a novel Scene Graph Auto-Encoder (SGAE). There are two stages to this particular captioning workflow: First, by employing a Convolutional Neural Network (GCN) to encode the image’s scene graph, then decoding the phrase using the recording illustration; second, by including the image’s scene graph into the captioning model.

Similarly, (L. Li et al., 2019) present the Relation-aware Graph Attention Network (ReGAT), an innovative framework for Visual Question Answering (VQA) to describe multi-type object relations with a query adaptive attention mechanism. To use graph data in images and text, the authors (J. Yu, Lu, et al., 2018) introduce a novel cross-modal retrieval model they call a dual-path neural network with a graph convolutional network. It considers regular vector-structured visual illustrations and irregular graph-structured textual representations to learn linked features concurrently and shared latent semantic space. Furthermore, (S. Wang et al., 2020) generate the text scene and visual scene graphs by mining the image and text for objects and associations (Figure 6). Ultimately, they developed a model called Scene Graph Matching (SGM). This model uses two specialized graph encoders to transform the raw data from the visual and text scene graphs into a feature graph. After gaining knowledge of the properties present at both the object and connection levels in each graph, it will be possible to match the two feature graphs that correspond to the two modalities at two levels with greater success.

Brain Networks: Centralized activities that reveal a region’s importance in a network (Page et al., 1999), together with anatomical and functional connectivity (Fornito et al., 2013), is now among the most studied aspects of the brain. Differences in sex and age (Zuo et al., 2012), mania and depression (Deng et al., 2019), blindness and vision loss (Q. Lin et al., 2021), diabetes and neuropathy (Q.-H. Xu et al., 2020), and genetics and epigenetics (Wink et al., 2018) have all been shown by analyzing functional connectivity centrality.

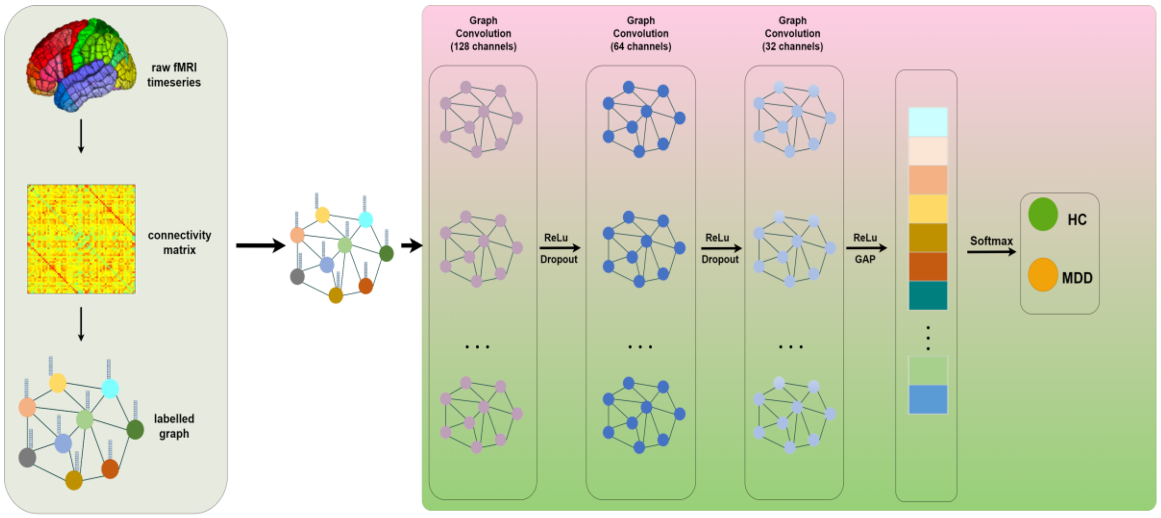

Figure 7: Biomedical Application.

Recommendation System: In social networks and advertising platforms, recommendations play a crucial role (W. L. Hamilton et al., 2017; Mao et al., 2021; Z. Wang et al., 2020; Z. Zhao et al., 2020). Some networks include spatial and temporal data (Kermarrec et al., 2011) and structure, content, and label data. In mobile applications, spatial-temporal embedding (C. Zhang et al., 2017) is a developing field. GNNs are a valuable tool to employ with user and item relational characteristics. KGNN-LS (H. Wang et al., 2019) improves the item representation in a knowledge graph by carrying out aggregations among its associated neighbourhood. Another assumption made by KGNN-LS (H. Wang et al., 2019) is that related objects in the knowledge graph would likely have similar user preferences. It adds the regularization term to acquire such a customized weighted knowledge network. KGCN and KGAT (X. Wang et al., 2019) generally have similar concepts. An auxiliary loss for knowledge graph reconstruction is the only significant difference.

Biomedical Application: Biomedical data analysis can make use of graph representation learning. For instance, brain network data can be represented as a graph, with the signals from the brain acting as the nodes (D. Zhang et al., 2018)(B. Li & Pi, 2020). The brain's structures and functions in response to various inputs can be studied using embedding techniques (Figure 7). For the study of Alzheimer's disease (C. Hu et al., 2015) (Si et al., 2019) and the brain's response to magnetoencephalography signals (R. Liu et al., 2018), various frameworks have been put forth. Drug repurposing algorithms have been utilized with Electronic Health Record (EHR) data (Hurle et al., 2013) (Pushpakom et al., 2019). For instance, (Gurwitz, 2020) used EHR using data to modify drugs to treat COVID-19. (Y. Wu et al., 2019) identified several non-cancer drugs as modifiable options for cancer treatment. Based on the omics (proteomics and transcriptomics) signature data of bladder cancer patients, (Mokou et al., 2020) developed a drug repurposing pipeline.

Natural Language Processing (NLP): The text classification problem is somewhat old in the field of NLP. Documents can serve as nodes in the citation network, with references between them serving as edges. The bag-of-words approach is commonly employed when describing the attributes of nodes in a citation network. When dividing articles into distinct sets, node classification is the quickest and easiest option. Numerous Graph convolutional networks have been explored, to name a few: (W. Hamilton et al., 2017; Kipf & Welling, 2016a; Levie et al., 2018; Monti et al., 2017; C. Zhuang & Ma, 2018).

On the other hand, you may use graph classification to sort the texts and analyze the documents graphically (i.e., each document is modeled as a graph) (Defferrard et al., 2016). TextGCN (H. Peng et al., 2018) also models the entire corpus to a heterogeneous graph and simultaneously learns word and document embeddings before classifying the text using a SoftMax classifier. Using a graph pooling layer and hybrid convolutions comprising graph convolution and classic convolution, Gao et al. (H. Gao et al., 2019) use node ordering information to perform better than traditional CNN-based ones GCN-based techniques. As the number of labels increases, especially if they are all at various degrees of topical granularity, the effectiveness of these methods may decrease. Using a graph-of-words to represent long-distance semantics, (H. Peng et al., 2018) apply a recursive regularized graph convolution model to use the pyramid of labels. The foundation of many NLP-related applications is information extraction, and graph convolutional networks have been extensively used in this and its related challenges. For instance, GraphIE (Qian et al., 2018) discovers non-local relationships between textual units, generates local context-aware latent reconstructions of words or sentences, and utilizes a decoder to label words at the word level. GraphIE is compatible with information extraction methods like named entity extraction. Word-relationship extraction (Y. Zhang et al., 2018) and event-extraction (S. Cui et al., 2020; X. Liu et al., 2018; Nguyen & Grishman, 2018) are two applications of convolutional graph networks. Furthermore, Marcheggiani et al. create a syntactic graph convolutional network model that can be applied to syntactic dependent trees and is appropriate for numerous NLP applications, including neural machine translation (Bastings et al., 2017) and semantic role labeling (Marcheggiani & Titov, 2017). Graph convolutional networks can give phrase encoders a semantic bias and enhance performance in semantic machine translation (Marcheggiani et al., 2018).

Traffic: For an intelligent transportation system to function, it must accurately predict traffic density, road capacity, or speed in traffic systems. Several researchers use spatial-temporal GNNs (STGNNs) to develop models that can deal with a wide range of traffic network problems; these include (Y. Li et al., 2017; B. Yu et al., 2017; J. Zhang et al., 2018). The sensors installed along the roads are viewed as nodes in a spatial-temporal graph, with the distances between each pair of sensors representing edges. The average traffic speed at each node throughout a given frame is provided as a dynamic input element, whereas (Agafonov & Myasnikov, 2021; Bing et al., 2020; Bogaerts et al., 2020; L. Cai et al., 2020; L. Zhao et al., 2022) focused on solving the problem of Road traffic speed. (X. Fang et al., 2020; H. He et al., 2021; James, 2021; F. Li et al., 2021; Y. Zhang, Li, et al., 2021) looked at the angle of predicting road travel time, whiles (Bai et al., 2021; J. Chen, Liao, Hou, et al., 2020; X. Chen, Zhang, Du, et al., 2020; S. Fang et al., 2020; Ge et al., 2020; Jiang & Luo, 2022; Yin et al., 2021) also focused on solving and predicting road traffic flow.

7. The Limitations of Using GNNs and Some of the Possible Solutions

The over-smoothing, scalability, and expressive ability of GNNs are only a few of their well-documented shortcomings. This paper provides an overview of the most important articles that deal with these topics.

7.1 Over-Smoothing

It has been established that the depth limitation of GNNs is a significant issue (Q. Li et al., 2018)(Oono & Suzuki, 2019). The first layer considers a node's immediate neighbours when using

Authors	Proposed Solution
(Dong et al., 2021)	AdaGNN
(Klicpera et al., 2018)	APPNP
(K. Zhou et al., 2020)	DGN-GNN
(M. Liu et al., 2020)	DAGNN
(Rong et al., 2019)	DropEdge
(Kipf & Welling, 2016a)	GCN
(M. Chen, Wei, Huang, et al., 2020)	GCNII
(Hasanzadeh et al., 2020)	GDC
(D. He et al., 2022)	GCN+inflation
(K. Xu, Li, et al., 2018)	JKNet-Concat
(Chamberlain et al., 2021)	GRAND-PDE
(Eliasof et al., 2021)	PDE-GCN
(H. Zeng et al., 2021)	SHADOW-SAGE
(F. Wu et al., 2019)	SGC

Table 6: Some proposed over-smoothing techniques.

Authors	Algorithm	Solution Summary
(Azizian & Lelarge, 2020)	FGNN	Adding matrix multiplication to the model
(Balcilar et al., 2021)	GNNML	Using a broad receptive field and spectral domain design to create the convolution.
(Z. Chen et al., 2019)	Ring-GNN	Adding and multiplying by means of a ring of matrices
(Dasoulas et al., 2019)	CLIP	Utilizing colors to separate related node properties
(Z. Huang et al., 2022)	PG-GNN	Using a permutation-aware aggregate to capture the correlation between nearby nodes.
(P. Li et al., 2020)	DEGNN	Incorporating a distance-based additional node functionality
(Maron et al., 2019)	PPGN	Taking higher-order message forwarding into account
(Morris et al., 2019)	k-GNN	Using subgraph structures for message passing rather than nodes
(Murphy et al., 2019)	RP-GNN	Including a special node label
(Papp et al., 2021)	DropGNN	Randomly removing some of the nodes
(Sato et al., 2021)	rGIN	Adding random features to GIN
(H. Wang et al., 2022)	PEG	Updating the node and positional features over distinct channels.
(Wijesinghe & Wang, 2021)	GraphSNN	Facilitating information transfer by incorporating additional framework.
(M. Zhang & Li, 2021)	NGNN	Rather than encoding each node as a rooted subtree, a rooted subgraph is used.

Table 7: A summarization of significant solutions suggested to enhance GNN expressive power.

GNN methods. A node's second layer travels to its two-hop neighbours as a sub-layer, and successive layers are layered similarly in the node's neighbours. The resulting vectors will be over-smoothed since local information for each vertex is lost after several layers. To address the problem of over-smoothing, Graph Random Neural Network (GRAND) proposes a novel architecture.
Drop Node is used in this method to expand the feature matrix of the input graph similarly to the Drop Edge process (Rong et al., 2019). This improvement helps lessen a node's reliance on other

neighbouring nodes. Then, to reduce the likelihood of over-smoothing, it combines neighbours of a node up to K hops away. Consistency regularization is used in the model's training process (Berthelot et al., 2019) to lessen the problem of overfitting in the semi-supervised environment when dealing with scarce labels. Theoretically, it has been demonstrated by (M. Chen, Wei, Huang, et al., 2020) that over-smoothing may be defeated by adding two straightforward approaches to GCN at each layer. An initial linking to the input features must be established to guarantee that some node features are included in the final node representation. To guarantee at least the same efficiency level as a shallow GCN, the identity matrix is recommended to be included in the weight matrix. As shown in Table 6, many approaches have been taken to overcome the over-smoothing problem in GNN.

7.2 Scalability

In addition, GNNs have difficulty scaling since the embeddings of a node are constructed by combining the representations of that node's neighbours. In particular, the temporal complexity of neighbour aggregation for a GNN with many layers is significant. In huge graphs, there may be a lot of neighbours, which slows down GNN training and uses more memory. This issue is resolved by (C. Zhuang & Ma, 2018) and (Chiang et al., 2019; Gomez et al., 2017)by picking a portion of node neighbours. A node's neighbours are sampled separately at each tier (J. Chen et al., 2018). By using clustering algorithms to sample a subgraph one per batch and a graph convolution filter on the subgraph's nodes, Cluster-GCN (Chiang et al., 2019) lessens the memory issue associated with GCN. Specific techniques like SGC (F. Wu et al., 2019) eliminate the non-linear activation function to shorten training time. By using reversible connections (Gomez et al., 2017), RevGNN (G. Li et al., 2021) lessens the memory footprint of GNNs as a function of layer count. The feature matrix is separated into many categories as inputs to this model. The most recent input's propagation method output is stored in memory, an advantage of this method. The scalability of GNNs has also been examined in several other works, including (Bojchevski et al., 2020; M. Chen, Wei, Ding, et al., 2020; M. Ding et al., 2021; Fey et al., 2021; Z. Huang et al., 2021; Yoon et al., 2021).

7.3 Expressive Ability

The ability of a model to distinguish between various graphs is referred to as expressive power. To rephrase, the model may assign equivalent graphs to the same embedding and dissimilar graphs to different embeddings. The Weisfeiler & Lehman (WL) test limits the expressive power of popular GNNs like GCN (Kipf & Welling, 2016a) and GraphSAGE (W. Hamilton et al., 2017), and they are unable to recognize some non-isomorphic substructures. Graph Isomorphism Network (GIN), a more potent GNN-based embedding technique, is suggested in (K. Xu, Hu, et al., 2018). GIN aggregates a node's neighbours using the sum operator rather than the mean or max operators. The expressive capability of GIN is also theoretically demonstrated to be equivalent to that of the WL test. A coloring mechanism is suggested by Identity-aware Graph Neural Networks (ID-GNN) (J.

You et al., 2021) as a means of enhancing the expressive capability of GNNs. P. Li et al. (2020) add distance-based features to each node, increasing the GNNs' expressive power by creating distance vectors for each node relative to each center node. Table 7 compiles the results of some experiments aimed at making GNNs more expressive.

7.4 Destructive Loss

Destructive loss occurs whenever a neural network model is retrained for a different job. To solve this problem in GNNs, H. Liu et al. (2021) suggest a topology-aware weight-preserving module (TWP). This portion evaluates how crucial the GNN's parameters are once each task has been learned. By associating negative reinforcement with adjustments to critical parameters, the model is forced to remember the past settings when studying a new task. When learning new tasks, an experience replay-based model (ER-GNN) (F. Zhou & Cao, 2021) chooses and replays a few nodes from earlier tasks. In model training, the repeated nodes have the most impact and coverage because their qualities are closest to the class average of all features.

7.5 Over-Squashing

One of the problems with GNNs is that they tend to over-squash when trying to capture long-range interactions. When a graph learning task calls for only brief interactions and more layers are added to a GNN, the node embeddings become ambiguous, and a phenomenon known as over-smoothing happens. In GNNs, the input from far-off neighbours is crushed into a fixed-length vector as extra layers are added to account for long-range relationships. This results in over-squashing and poor throughput from GNNs (Alon & Yahav, 2020). Alon & Yahav (2020) offers a straightforward solution to this issue by adding an adjacent layer as the final layer to a GNN model. In this layer, all possible pairs of nodes are linked, allowing for representations of nodes to consider relationships with neighbours beyond those immediately adjacent to the nodes in question. Over-squashing is proven to be caused by negatively curved edges (Topping et al., 2021). Negative curvature occurs in a graph whenever there is a connection between two sides of the graph that would be severed if the edge in question were removed.

7.6 Interpretability

Implementing deep graph representation learning in new domains presents challenges in interpretability. Scholars within the field of computational social science emphasize the importance of integrating prediction and reasoning (Hofman et al., 2021). Neural networks possess a significant level of opacity as perceived by human observers. Deep learning is extensively utilized in diverse fields, although it needs to be more understood by the general public. The understanding of task fulfillment and the assimilation of acquired expertise by a specialist in deep learning continue to present challenging initiatives. The lack of sufficient human feedback degrades the credibility of neural network models, impedes human learning from their results, and limits their potential for further progress. The increasing popularity of Graph Neural Networks (GNNs) has led to the development of several GNN-based approaches that aim to facilitate reasoning on graphs. Recently, scholars have put forth various methodologies to elucidate the rationale behind the predictions generated by these opaque models. The attribute of interpretability holds excellent importance in machine learning models, particularly in practical scenarios where the outcomes of these models might have substantial ramifications. GNN-based methodologies frequently encounter criticism due to their limited interpretability, as the acquired representations tend to be intricate and challenging to comprehend. Nevertheless, recent endeavors have tackled this concern by developing techniques to elucidate the learned representations (Y. Zhang, Tivno, et al., 2021). Previous scholarly literature has given two distinct methods in offering reasons for model outcomes. The initial methodology entails providing impromptu justifications after the production of products. The second methodology entails adjusting the model's framework to improve the caliber of the explanations. The methods to explain these concepts might vary, encompassing techniques such as offering analogous instances, elucidating distinct attributes of the input features, revealing latent layers and extracting semantic knowledge from them, or deducing logical principles. In addition, explanations can be categorized as either local, which concentrates on specific samples, or global, which offers reasons for the entire dataset. One possible strategy for improving the interpretability of GNNs entails the identification of the most salient attributes within the graph, specifically nodes, and edges. The work can be achieved by employing GraphLIME (Q. Huang et al., 2022)(Schnake et al., 2021). The relevance of individual characteristics is assessed by methods that evaluate the influence on the model's output when a specific feature is removed. (Q. Huang et al., 2022) the approach utilizes the Local Interpretable Model-Agnostic Explanations (LIME) technique to clarify the predictions generated by GNNs.

The methodology used in this study comprises creating a set of modified graphs that share similarities with the original chart but with the removal of specific nodes and edges. Afterward, the GNN is trained on each altered graph. The relevance of particular nodes and edges is then determined by assessing the changes in the model's output. The authors illustrate the effectiveness of their methodology in identifying the most crucial nodes and edges in the graph, providing valuable insights into the model's decision-making process. GraphSAINT (H. Zeng et al., 2019) is a technique that leverages subgraph sampling in order to enhance the interpretability of GNNs. The approach involves the process of extracting a collection of subgraphs from the primary graph and subsequently training the GNN on each of these subgraphs. The significance of every node and edge is subsequently calculated by evaluating the impact on the model's output. The study demonstrates the applicability of their approach in identifying the most significant subgraphs within a given graph and offering valuable insights into the decision-making mechanism of the model.
Another potential approach to enhance the interpretability of GNNs involves employing visualization techniques to depict the decision-making process employed by the model. One possible approach to achieve this objective is through the utilization of techniques such as (J. Yu et al., 2021)(Selvaraju et al., 2019). This method facilitates the creation of heatmaps that effectively emphasize the nodes and edges within the graph that hold the greatest significance. The approach operates by calculating the gradient of the output concerning the activations of the final convolutional layer of the GNN. The gradient is subsequently employed to calculate a weighted summation of the activations, with the weights being determined based on the significance of each node and edge within the graph. The authors demonstrate the efficacy of their approach in offering enhanced precision and nuanced elucidation of the decision-making mechanisms employed by GNNs. (Leow et al., 2019; Z. Liu et al., 2022; X. Shi et al., 2021), employ visualization techniques to depict the acquired node representations within the graph. The authors demonstrate the applicability of their methodology in identifying clusters of nodes that exhibit similar characteristics, as well as in visualizing the model's decision-making process. Rule extraction is a technique that entails extracting logical rules from a model's decision-making process. One approach to achieve this is through the utilization of Relational Inductive Biases, which involve the acquisition of a collection of logical rules that can be employed to elucidate the decision-making mechanism of the model. The utilization of rule extraction methods has been employed for the purpose of extracting logical rules from

GNNs. The process of model simplification which is another approach, entails modifying the model's structure in order to enhance its interpretability. One possible approach to achieve this is by employing techniques such as Graph Transformer Networks (Yun et al., 2019). These networks leverage a transformer-based architecture to streamline the model and enhance its interpretability. The aforementioned instances serve as mere illustrations of potential methods for enhancing the interpretability of Graph Neural Networks (GNNs). There remains a lack of consensus regarding the optimal approaches for enhancing the interpretability of a model. Scholars are currently engaged in ongoing investigations to explore various possibilities, resulting in a multitude of challenges and intriguing subjects within this domain.

There exist multiple potential avenues for addressing the challenges associated with scalability and over-smoothing, among other issues, in graph representation learning. One potential avenue for investigation involves exploring novel architectural designs capable of effectively managing dynamic graphs, which represent the evolving structures and attributes of entities. Most graph embedding models primarily focus on static graphs. However, temporal fluctuations impact the structure and features of graphs (C. Chen et al., 2019; Mitrovic & De Weerdt, 2019). The study of graph evolution involves the examination of several dynamic characteristics, including the evolution of topology, features, degree distribution, and node roles. Several models have been developed to capture and reflect evolutionary patterns, although they may not fully encompass dynamic behaviors (Rahman et al., 2018). For example, individuals can modify their place of origin, professional occupation, and position within a restricted group through social networks. Models can depict the dynamic structure and attributes of entities, thereby guiding the direction of research.

An alternative approach involves exploring novel graph embedding models capable of accommodating disassortative graphs, characterized by a tendency for nodes with distinct labels to be interconnected. Upon reviewing the sampling method employed by GNNs and graph transformer models, it is observed that the vector embeddings of the target nodes are updated by considering the properties of their k-hop neighbours (P. Wang et al., 2019; K. Xu, Hu, et al., 2018). This issue pertains to classification tasks in which the aggregation processes operate under the assumption that interconnected nodes should possess identical labels. However, this assumption starkly contrasts with the disassortative graph structure. In recent years, various approaches have been suggested to address the challenges associated with classification tasks on disassortative graphs (Brody et al., 2021)(Xie et al., 2020). Nevertheless, the utilization of message-passing-based techniques poses a persistent obstacle and presents a significant barrier in the context of disassortative graphs.

Furthermore, the scalability of graph-based models can be enhanced through the utilization of optimization techniques, such as sampling and parallelization. In order to mitigate the issue of over-smoothing, future research endeavors might effectively address this concern by incorporating an initial residual connection, employing dropout techniques, and leveraging the utilization of PageRank algorithms (M. Chen, Wei, Huang, et al., 2020; Chien et al., 2020; Xie et al., 2020). Moreover, the development of a deep-learning model to effectively tackle the issue of over-smoothing remains an unresolved matter and a highly encouraging avenue for further investigation. Several researches have suggested the utilization of graph transformer models as a potential solution to address the constraint imposed by the message-passing mechanism through the incorporation of self-attention. The majority of contemporary models are developed in isolation, tailored to address specific tasks that have not yet been generalized, even when dealing with graphs within the same domain (Z. Wu et al., 2020). Despite various pre-trained graph transformer models for related tasks, their transferability to different tasks remains constrained, particularly in the context of particular graph data (Hussain et al., 2022; J. Zhang et al., 2020). This issue arises from the necessity of training models from the beginning whenever fresh graph data and other tasks are introduced, resulting in a time-consuming process that restricts the practical feasibility. The utilization of pre-trained models is advantageous in addressing the limited availability of node labels. Hence, in the event that the graph embedding models have undergone pre-training, they possess the potential to be transferred and effectively employed for the purpose of addressing novel challenges.

8. Graph Neural Architecture Search (G-NAS)

Generating efficient GNN designs for various applications and datasets needs manual labor and subject-matter expertise. Some strategies were offered to automatically construct a suitable architecture for the provided datasets and downstream activities. Reinforcement learning automates architecture design using the Graph Neural Architecture Search method (G-NAS) (Y. Gao et al., 2022). Both the hyperparameters (HPs) of the GNN model, such as the dropout and learning rates, and the GNN architecture constituents (ACs), which comprise the attention, aggregation, and activation functions, are essential. Choosing these components in conventional GNNs is a challenge solved using professional judgment and unwritten conventions.

Consequently, it is challenging to combine these components in an optimum manner. Testing many GNN architectures before settling on the optimal one is necessary to create the finest GNN models for specific applications. Applying such a procedure to massive graph data would take too long or impossible. Different tasks require varied HP tweaking to achieve the best results. Following the recent breakthroughs in neural architecture search (NAS) for CNN and RNN architectures, various scholars are attempting to bring AutoML techniques into the graph space. There has been a recent uptick in research into automatically determining the best GNN models for certain jobs. Common names for it include Graph Neural Architecture Search (G-NAS). G-NAS is receiving more attention due to the increased interest in GNN models. According to the search space, a model comprises ACs and HPs. The mission of G-NAS is to discover the model that gives high the expected efficiency $\epsilon_p(s)$ on a validation set $D_{val} \subset D$, with a searching space, S made up of all feasible designs and dataset D. In other words, it seeks the best model possible.

$$s^* = argmax_{(s\in S)}\epsilon_p(s) \tag{4}$$

(Oloulade et al., 2021) The ability to display diverse strategies within a consistent framework is made possible by categorization. Existing G-NAS frameworks are organized along three dimensions, each representing a different aspect of the complexity with which they were built: (1) searching space, which specifies a collection of GNN designs; (2) searching algorithm, which specifies how to identify the optimal GNN models within the searching space; and (3) efficiency assessment, which evaluates how well a GNN model performs. GNN models are typically constructed from a searching space using a searching method, and their efficacy is measured using a performance assessment method (Figure 8).

8.1 Searching Space

The shape of the chosen architectures is determined by a searching space that includes all possible possibilities for each part of a GNN model. The longer it takes to calculate, however, the more possibilities there are for selecting the best model inside the area of search. It is, therefore, difficult to construct a complete and compact search space that accounts for all possible optimal structures. There have been several efforts to address this issue, each of which has proposed a unique searching

Constituents	**Functions**	**Values**
	Activation function	Sigmoid, tanh, ReLU, linear, softmax
	Attention function	Linear, sym-GAT, cos, linear, gene-linear, constant, and GAT
	Aggregation function	mean, max, sum, mlp
Architecture constituents	Attention head	1,2,4,6
	Combine function	Concat Avg,
	Skip connection	Stack, skip-sum, skip-cat
	Hidden size	2 to 16, 32, 64,128, 256, 512
	Learning rate	0.0001 to 0.1000
	Dropout	0.1 to 0.9
	Weight decay	0.00001 to 0.00100
Hyperparameter	Batch normalization	True, false
	Batch sizes	16, 32, 64
	Optimizers	SGD, Adam Optimizers
	Training epochs	2 to 300

Table 8: Typical values used for hyperparameters and functions.

Model	Architecture constituents								Hyperparameter						
	F1	F2	F3	F4	F5	F6	F7	F8	Hp1	Hp2	Hp3	Hp4	Hp 5	Hp 6	Hp 7
AGNN (K. Zhou et al., 2019)	Y	Y	Y	Y	Y	N	N	Y	N	N	N	N	N	N	N
AutoGraph (Y. Li & King, 2020)	Y	Y	Y	N	Y	Y	Y	Y	N	N	N	N	N	N	N
AutoGM (Yoon et al., 2020, 2022)	Y	Y	Y	N	Y	N	Y	Y	N	N	N	Y	N	N	N
AutoNE (Tu et al., 2019)	N	N	N	N	N	N	N	N	Y	Y	Y	N	Y	Y	Y
DSS (Y. Li et al., 2021)	Y	Y	Y	N	N	Y	N	Y	Y	Y	Y	N	N	N	N
DiffMG (Y. Ding et al., 2021)	N	N	N	N	Y	N	N	Y	Y	Y	Y	N	N	N	N
GNAS (S. Cai et al., 2021)	N	N	Y	N	N	Y	N	N	Y	N	N	N	N	N	N
GeneticGNN (M. Shi et al., 2020)	Y	Y	Y	N	Y	N	N	Y	Y	Y	Y	N	N	N	N

PDNAS(Y. Zhao et al., 2020)	Y	Y	Y	Y	Y	Y	N	Y	N	N	N	N	N	N	N
SANE (H. Zhao et al., 2021)	Y	Y	Y	N	Y	Y	N	Y	Y	N	Y	N	N	N	N
SNAG (H. Zhao et al., 2020)	Y	Y	Y	N	Y	Y	N	Y	N	N	N	N	N	N	N

Table 9: Details of a Searching Space.

Note: F1 = Attention, F2 = Activation, F3 = Aggregation, F4 = Combine, F5 = Attention head, F6 = Skip connection, F7 = Number of layers, F8 = Hidden size, H1 = Learning rate, H2 = Drop out, H3 = Weight decay, H4 = Batch normalization, H5 = Batch sizes, H6 = Optimizers, and H7 = Training epochs.

space, which we here categorize into three broad classes: There are three distinct types of searching spaces: (1) one that only considers GNN model ACs while leaving HPs at their default values; (2) one that only considers HP tuning choices for a predetermined GNN; and (3) one that considers both GNN model ACs and HPs.

8.1.1 Architecture Constituents Searching Space

Here, fixed HPs are applied to the ACs of GNN models and generate a search space for them (Y. Gao et al., 2019) and (H. Zhao et al., 2020). It's possible that nodes' representations in one layer of a GNN model's ACs don't depend on those in the layer below it, opening up the possibility of a micro search space. While HPs are an integral part of the entire model, the existence of such search spaces is troublesome because it reduces the searching efficiency. Improving only the GNN structure, however, can result in a subpar model because even a small shift in the learning settings can have a major effect on the optimized GNN architecture.

8.1.2 Hyperparameter Searching Space

Using fixed and specified GNN model ACs, this searching space (Tu et al., 2019) optimizes the HPs exclusively for a given GNN model. A less-than-ideal solution is achieved using Hyperparameter Optimization (HPO), the major purpose of which is to optimize the HPs for a given neural network.

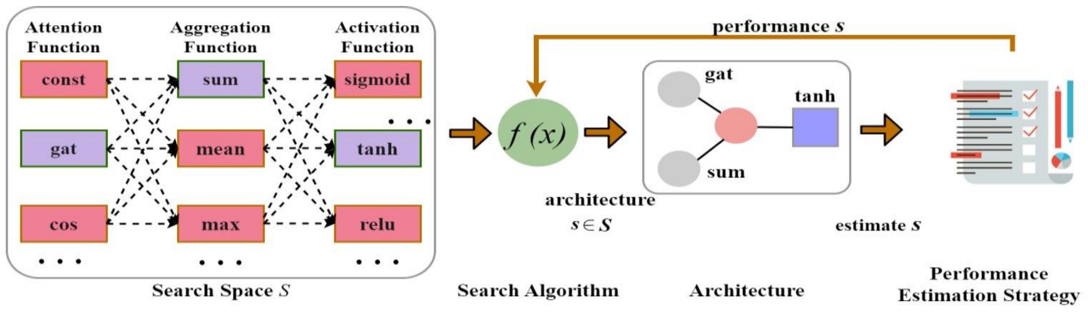

Figure 8: Graph Neural Architecture Search.

Method	Advantage	Disadvantage
Architecture constituents searching space with pre-set hyperparameters.	Reduced computational load and limited searching space.	Ignoring the importance of optimizing hyperparameters can result in a less-than-ideal or unreliable model.
Hyperparameters searching space with pre-set architecture constituents.	Reduced computational load and limited searching space	Only optimizing hyperparameters can lead to a suboptimal model
Architecture constituents searching space, and hyperparameters searching space.	It gives a stabilized solution.	Expenditure of extra computing time due to the size of the searching space

Table 10: Searching Space Design Analysis.

8.1.3 Architecture Constituents and Hyperparameter Searching Space

Here, the spaces where ACs and HPs of GNN models can be found through a search are specified (M. Shi et al., 2020)(Y. Li & King, 2020). The impact of a slight shift in HPs on a model is also considered. However, the scale of the search space may necessitate more calculation costs even if this arrangement ensures good results. While these search spaces might produce excellent outcomes, they are typically not transferable from one GNN architecture to another. The lack of a solution to this problem hinders the identification of effective GNN models. Table 8 provides a breakdown of the ACs and HPs most typically employed by G-NAS frameworks, whereas Table 9 summarizes the functions or HPs existing works have adopted. In Table 10, this study summarizes the results of comparing the current approaches to building search spaces. In order to develop a better model, the calculation time will increase in proportion to the size of the search space. There is a scalability problem with this particular functionality. There have been numerous search strategies developed in an attempt to address this issue. In the following paragraphs, some methods for achieving these ends are elaborated.

8.2 Searching Algorithm

When there are several possible GNN models, the search method specifies how to zero in on the best one. When defining the search algorithm, it is vital to consider the computational time and space requirements. More preferably, the algorithm should require fewer computational resources and reach the ideal answer faster. Therefore, the most difficult part is developing a search algorithm that finds the optimal balance between high efficiency, low cost, and enhanced efficiency. The range of used searching algorithms is extensive, including reinforcement learning, Bayesian optimization, evolution learning, differentiable search, and random search.

8.2.1 Bayesian Optimization (BO)

BO probabilistic approach takes a preceding probability and a likelihood function and uses the result to calculate a posterior probability. As new information becomes available, forecasts and error estimates are revised accordingly. The two primary parts of BO are a Bayesian statistical model, also known as a substitute function, and an acquisition function, which is used to identify where to obtain the next sample. In BO, a probability dispersal is built over samples of interest that are put through the surrogate function's tests. The response from the substitute function is interpreted with the help of the acquisition function, and appropriate samples are found for the ultimate objective function assessment. With each passing step, the acquisition function is fine-tuned to pick the most representative sample possible for subsequent evaluation. The process is repeated after the model has been updated and convergence has been achieved. Only (Yoon et al., 2020) have employed BO to solve the G-NAS problem, and their outcomes are not encouraging despite BO's superior performance in NAS (H. Zhou et al., 2019). BO is computationally demanding for large-scale data and nearly impractical to perform effectively.

8.2.2 Reinforcement Learning (RL)

In RL, an agent learns to maximize reward by making a series of decisions informed by continuous feedback, explained by a machine learning algorithm. There are three key parts: the agent, who is aware and can make decisions; the environment, from which the agent studies and selects which actions to do; and the rewards for those acts. The searching space represents the environment in RL-based approaches for G-NAS, the agent is a neural network defined as an optimizer that seeks to create a great layout (a submodel) from the searching space over time, and the compensation is the evaluation of the generated model's behavior. This method, which can be broken down into two categories, has been implemented in various G-NAS frameworks (K. Zhou et al., 2019). One has had no changes made, whereas the other has made some changes. The first category of frameworks anticipates a total overhaul of the existing good sub-model, regardless of how similar the subsequent sub-model may be to the existing one (Y. Gao et al., 2019). In other words, the performance of a constructed model is evaluated at each level; if the goal has not been met, a new model is generated from scratch. One major drawback of this strategy is that the controller cannot efficiently execute the search since it cannot identify which characteristic of a sub-model is responsible for the observed performance difference compared to the prior sub-model. Second, there is a group of frameworks that, working off the assumption that the amount changed only marginally improves the result generated by the original sub-model, create a new sub-model by modifying a little piece of the old one (Y. Li & King, 2020). It is possible that the controller's number of neural networks would be equal to the total number of model parts in this case. At each iteration, the system's entropy assigns a weight to each part following its relative significance. As a result, with relatively minor modifications to the infrastructure, this approach can guide the exploration of the search space. However, the size of the search spaces used by these models still prevents them from being fully expressive.

8.2.3 Evolutional Learning

Natural selection and genetics inspired the mechanisms behind evolution learning, a generalized population-based metaheuristic optimization approach. Multiple iterations of this method exist, but the genetic algorithm (GA) for NAS frameworks is now the most often used. The people (or population) who inhabit a certain area are the GA's most crucial constituent. An individual is composed of genes or elements of a solution. An individual from an initial population is evaluated using a fitness function, and the best individual or individuals are chosen to populate the next generation. It is possible to produce a new population using a mix of crossover and mutation, but the mutation is the most common method. A new person is thus made by picking the most desirable applicant and adjusting them. In the G-NAS issue, the search space stands in for the sample population, and the genes characterize the framework of a GNN. The size of the search area makes it difficult to ascertain every person's fitness, and in the case of a large graph dataset, training GNN can be very time-consuming. (M. Shi et al., 2020) and (Y. Li & King, 2020) randomly choose a fixed-size population and start it. This method's sluggish convergence is its biggest drawback.

8.2.4 DIFFERENTIABLE SEARCH

The fundamental goal of the differentiated search approach is to lessen the burden on the network's resources by combining training and architecture samples into a supernet. This idea proposes generating a continuous search space as an alternative to doing a series of independent candidate searches. In contrast to traditional search algorithms, constant search spaces facilitate the differentiation of training objectives, lead to more efficient optimization, and lead to faster convergence. The differentiated search, first for the NAS problem (H. X. Liu et al., 2019)(Pham et al., 2018), focuses on discovering repeating patterns in a network of units. The final form of contemporary neural networks is built by stacking one or more computing blocks atop one another (Noy et al., 2020). In G-NAS systems based on a differentiable search, a block is often embodied as a direct acyclic graph (Y. Zhao et al., 2020)(S. Cai et al., 2021) consisting of a sequential set of nodes. We wanted to select a network action for each layer, five in total. A differentiable search for G-NAS is predicated on stacking mixed processes to create a network (supernet). Each layer's candidates are used in each mix operation, and the resulting sets are linearly blended. Our ultimate goal is to have the mixed coefficients serve as thresholds in deciding. Finally, we can do architectural searching by attempting to minimize a loss function, like the cross-entropy loss. When all is said and done, the operation that yields the highest coefficient across all layers is selected for use in the final architecture.

8.2.5 RANDOM SEARCH

In RS, submodels are generated at random from the search space. Some researchers (Lai et al., 2020) have examined RS for G-NAS. Regrettably, even though this technique can produce effective results, it is not frequently used due to its risky outcomes, which are often less desirable than those by other approaches. To mitigate the danger inherent in this method, You et al. (J. You et al., 2020) employed a controlled RS to compare the weight given to one option with that granted to others involving the same component. The method randomly picks the other components and evaluates a predetermined number of GNN models. The program evaluates the performance of the selected GNN models for each option before rating them and examining the ranking distribution. The final GNNs are chosen based on the most crucial option. This method produces good results despite being expensive in terms of computing. Therefore, additional investigation into it provides an intriguing line of inquiry. Even if the majority of the research techniques mentioned above have produced positive results, each has flaws that are important to be aware of. Here, we compare the aforementioned search strategies while pointing out their benefits and drawbacks. Table 11 displays the comparison.

8.3 EFFICIENCY ASSESSMENT

Evaluating the effectiveness of a GNN design produced by a search algorithm requires a clear understanding of how to assess its performance. It can be viewed as part of the evaluation process while comparing the various structures produced by the search algorithm. It helps focus the investigation on the most relevant results. The difficulty of performance evaluation arises from the need to learn the submodel efficiency distribution to direct the search efficiently. It is standard practice to assess the submodel as a whole or the significance of its parts. When developing a new submodel, however, only minor adjustments to the parent model are necessary (M. Shi et al., 2020; K. Zhou et al., 2019). Element relevance in the variance of submodel performances will be evaluated to those elements. In cases where the existing submodel must be erased to make room for the new

Search Strategy	Benefit	Limitation
Reinforcement learning	Optimize results to their fullest extent.	Time-consuming
Bayesian optimization	Employ a stochastic model	Optimization based on a series of models; a heavy computational burden
Evolution learning	Innately parallel, effortlessly distributed	It's contingent on the starting population, and it's simple to get trapped in false convergence.
Differentiable search	Training and sampling in the context of architecture.	Expensive computation cost
Random search	Cheap to compute without sacrificing accuracy.	dangerous effect

Table 11: Analyzing the Search Algorithms.

one, the evaluation will reflect the submodel's overall performance. Metric performance is considered, but only if it makes sense for the given task. Accuracy is used to evaluate the entire model for node classification, while entropy is used to appraise individual features.

The primary contest is improving the efficiency and timeliness of the submodel review process. A few methods available can be used alone or in tandem with some different frameworks. Parameter weight sharing (also known as weight sharing or parameter sharing) was the first strategy to gain instant popularity (Y. Li & King, 2020). In order to avoid retraining a newly constructed submodel to convergence from scratch, weight sharing allows leveraging the weights of already-trained models. However, this approach could backfire if the weights are distributed unevenly among different-sized models. To counteract this shortcoming, numerous researchers have chosen a constraint that limits weight allocation between two models that lack a predetermined level of similarity. Parameter sharing is not useful for experiments, as K. Zhou et al. (2019) demonstrated. Another alternative is the single path one-shot concept (Guo et al., 2020) which uses only a single action between each input and outcome pair in each iteration. The sub-model assessment for G-NAS could be sped up using performance prediction (Klein et al., 2016; Wen et al., 2020; Zheng et al., 2020) in preference of training all generated models to gain the performance metrics. The time needed to conduct the sub-model evaluation could be drastically reduced. However, using this approach, you'll need hundreds of GNN performance distributions along with the graph data properties to build a neural predictor. Graph representation learning is another application area for the buffer method (Lai et al., 2020).

9. Open Cases and Ongoing Research

The subject of graph representation learning is highly active. Despite the development of several techniques, there are still fundamental problems and difficulties that must be resolved. GNNs have been constrained in many ways, including expressive capability, over-smoothing, scalability, and over-squashing. For these problems, various solutions have been put forth. Some of them, like over-squashing, were only recently discovered, and the treatments for them are still in the early stages.

Additionally, the proposed GNN limitations solutions address static GNNs, and there are no efforts to address the issues with dynamic and spatial-temporal GNNs. Furthermore, dynamic settings may introduce new problems that are not yet identified but merit investigation. Finally, novel GNN flaws that are interesting to investigate and address may be found by incorporating GNNs into various real-world issues and datasets.

Higher-Order Structure: Motifs and graphlets play an important role in complex networks, especially in biology, when characterizing protein-protein interactions. However, GNNs can typically only process data at the node and edge levels. The expressive power of graph-based models could be improved by incorporating these components into the message and transmission process.

Investigating the GNN Theory: GNNs are seen to be useful in so many contexts; therefore, researchers have tried to dissect their performance at the theoretical level. Optimization features and generalization over graph sizes are two aspects of GNN models for which there is less theoretical evaluation. The initial stages of comprehending the global meeting of gradient descent in GNNs are examined by (K. Xu et al., 2021). As shown by (Yehudai et al., 2021), GNNs can't generalize to larger unseen graphs when trained on specific graph distributions. Therefore, additional study is needed to fully comprehend these topics.

Including domain knowledge into GNNs: Often, a machine learning model lacks access to crucial information—domain expertise. This information can be incorporated into the models by adjusting the input, loss function, and model structure (Dash et al., 2022). As an example, we can improve the input data bearing in mind additional limitations on the entities' pre-existing associations.

Limited Training Data Labels: Indeed, it has always been difficult to train neural networks with only little data, and the same is true for training graph neural nets. The label scarcity problem is addressed by employing graph neural networks, which use solutions like self-supervised learning, data augmentation, and contrastive learning. However, more research into various forms of learning in GNNs in both static and dynamic contexts is still possible.

Robustness and Assured Efficiency: The reliability of GNNs under adverse conditions, such as when the data are noisy, or the network is under attack, is another crucial and mostly unexplored area of study. Just like the traditional neural network models, GNNs are just as vulnerable to adversarial attacks. Graph attacks account for the structure of a network, as opposed to harmful attacks on images or text that only focus on attributes.

Transferring Advances in Deep Learning Models to GNNs: Deep learning methods are implemented in graph neural networks. Graph neural networks can be modified to use any deep learning technique or model. Particularly bustling is the discipline of computer vision and natural language processing; the new models generated for photos, videos, and texts may all be evaluated in graphs. Since the perception of objects' motion in video frames is analogous to nodes shifting over time, it is possible that principles from video representation learning could be beneficial in dynamic graph learning.

Additional Applications: GNNs have succeeded considerably in various fields, including protein structures, financial networks, and social networks. Some of the more recent uses of GNNs include article publishing prediction (Guan et al., 2021), medicine over-prescription prediction (J. Zhang et al., 2021), and electrical power grid monitoring (Ringsquandl et al., 2021). Therefore, looking into how well GNNs perform in different applications is intriguing.

Conclusion

This study covers graph representation learning extensively. The paper examines many models, including traditional methods like graph kernels and matrix factorization, as well as recent deep-learning models adapted to specific graphs. Graph neural network (GNN)-based models operate well in many real applications, notably those with large graph datasets.
In contrast to previous summaries, the analysis of this study on graph representational learning is comprehensive. This study offered up-to-date literature at the time of writing for every type of learning environment (supervised, unsupervised, semi-supervised, and self-supervised). The various learning strategies used in each setting have partitioned the learning space into discrete sections. Although deep graph embedding models have undeniably exhibited impressive achievements in recent years, they are nevertheless constrained by some restrictions. The challenge of attaining an ideal balance between a network's underlying structure and its nodes' particular properties remains a significant obstacle for these models in various applications in downstream stages. This study reviewed the overarching framework of GNNs, the various types that fall under that umbrella, and the fields in which they find practical use. Some of the issues with GNNs and proposed solutions are included as well. These constraints hinder the capacity to express oneself creatively and capture long-range dependencies. Additionally, this study supplies the G-NAS framework, an environment for designing automated architecture generation for specific datasets and subsequent tasks. Current issues and potential future developments for GNNs have been discussed.

Acknowlegments

This work was supported by the National Natural Science Foundation of China under Grants 61571099 and 61501098.

References

Abu-El-Haija, S., Kapoor, A., Perozzi, B., & Lee, J. (2020). N-gcn: Multi-scale graph convolution for semi-supervised node classification. *Uncertainty in Artificial Intelligence*, 841–851.

Adhikari, B., Zhang, Y., Ramakrishnan, N., & Prakash, B. A. (2018). Sub2vec: Feature learning for subgraphs. *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 170–182.

Agafonov, A., & Myasnikov, V. (2021). Short-term Traffic Flow Prediction in a Partially Connected Vehicle Environment. *2021 3rd International Conference on Control Systems, Mathematical Modeling, Automation and Energy Efficiency (SUMMA)*, 968–972.

Agrawal, A., Ali, A., Boyd, S., & others. (2021). Minimum-distortion embedding. *Foundations and Trends®in Machine Learning*, *14*(3), 211–378.

Ahmad, A., Ullah, A., Feng, C., Khan, M., Ashraf, S., Adnan, M., Nazir, S., & Khan, H. U. (2020). Towards an improved energy efficient and end-to-end secure protocol for iot healthcare applications. *Security and Communication Networks*, *2020*, 1–10.

Allab, K., Labiod, L., & Nadif, M. (2016). A semi-NMF-PCA unified framework for data clustering. *IEEE Transactions on Knowledge and Data Engineering*, *29*(1), 2–16.

Alon, U., & Yahav, E. (2020). On the bottleneck of graph neural networks and its practical implications. *ArXiv Preprint ArXiv:2006.05205*.

Ashraf, S., Ahmed, T., & Saleem, S. (2021). NRSM: Node redeployment shrewd mechanism for wireless sensor network. *Iran Journal of Computer Science*, *4*(3), 171–183.

Ashraf, S., Saleem, S., Chohan, A. H., Aslam, Z., & Raza, A. (2020). Challenging strategic trends in green supply chain management. *Int. J. Res. Eng. Appl. Sci. JREAS*, *5*(2), 71–74.

Azizian, W., & Lelarge, M. (2020). Expressive power of invariant and equivariant graph neural networks. *ArXiv Preprint ArXiv:2006.15646*.

Bai, J., Zhu, J., Song, Y., Zhao, L., Hou, Z., Du, R., & Li, H. (2021). A3t-gcn: Attention temporal graph convolutional network for traffic forecasting. *ISPRS International Journal of Geo-Information*, *10*(7), 485.

Balcilar, M., Héroux, P., Gauzere, B., Vasseur, P., Adam, S., & Honeine, P. (2021). Breaking the limits of message passing graph neural networks. *International Conference on Machine Learning*, 599–608.

Bastings, J., Titov, I., Aziz, W., Marcheggiani, D., & Sima'an, K. (2017). Graph convolutional encoders for syntax-aware neural machine translation. *ArXiv Preprint ArXiv:1704.04675*.

Belkin, M., & Niyogi, P. (2001). Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in Neural Information Processing Systems*, *14*.

Belkin, M., & Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, *15*(6), 1373–1396.

Belkin, M., Niyogi, P., & Sindhwani, V. (2006). Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, *7*(11).

Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., & Raffel, C. A. (2019). Mixmatch: A holistic approach to semi-supervised learning. *Advances in Neural Information Processing Systems*, *32*.

Berton, L., de Paulo Faleiros, T., Valejo, A., Valverde-Rebaza, J., & de Andrade Lopes, A. (2017). Rgcli: Robust graph that considers labeled instances for semi-supervised learning. *Neurocomputing*, *226*, 238–248.

Berton, L., & Lopes, A. D. A. (2014). Graph construction based on labeled instances for semi-supervised learning. *2014 22nd International Conference on Pattern Recognition*, 2477–2482.

Besta, M., Peter, E., Gerstenberger, R., Fischer, M., Podstawski Michałand Barthels, C., Alonso, G., & Hoefler, T. (2019). Demystifying graph databases: Analysis and taxonomy of data organization, system designs, and graph queries. *ArXiv Preprint ArXiv:1910.09017*.

Bing, H., Zhifeng, X., Yangjie, X., Jinxing, H., & Zhanwu, M. (2020). Integrating semantic zoning information with the prediction of road link speed based on taxi GPS data. *Complexity*, *2020*.

Bogaerts, T., Masegosa, A. D., Angarita-Zapata, J. S., Onieva, E., & Hellinckx, P. (2020). A graph CNN-LSTM neural network for short and long-term traffic forecasting based on trajectory data. *Transportation Research Part C: Emerging Technologies*, *112*, 62–77.

Bojchevski, A., & Günnemann, S. (2017). Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. *ArXiv Preprint ArXiv:1707.03815*.

Bojchevski, A., Klicpera, J., Perozzi, B., Kapoor, A., Blais, M., Rózemberczki, B., Lukasik, M., & Günnemann, S. (2020). Scaling graph neural networks with approximate pagerank. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery \& Data Mining*, 2464–2473.

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., & Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. *Advances in Neural Information Processing Systems*, *26*.

Borgwardt, K. M., & Kriegel, H.-P. (2005). Shortest-path kernels on graphs. *Fifth IEEE International Conference on Data Mining (ICDM'05)*, 8--pp.

Brody, S., Alon, U., & Yahav, E. (2021). How attentive are graph attention networks? *ArXiv Preprint ArXiv:2105.14491*.

Cai, L., Janowicz, K., Mai, G., Yan, B., & Zhu, R. (2020). Traffic transformer: Capturing the continuity and periodicity of time series for traffic forecasting. *Transactions in GIS*, *24*(3),

736–755.

Cai, S., Li, L., Deng, J., Zhang, B., Zha, Z.-J., Su, L., & Huang, Q. (2021). Rethinking graph neural architecture search from message-passing. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6657–6666.

Cao, J., Lin, X., Guo, S., Liu, L., Liu, T., & Wang, B. (2021). Bipartite graph embedding via mutual information maximization. *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 635–643.

Cao, S., Lu, W., & Xu, Q. (2015). Grarep: Learning graph representations with global structural information. *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 891–900.

Cao, S., Lu, W., & Xu, Q. (2016). Deep neural networks for learning graph representations. *Proceedings of the AAAI Conference on Artificial Intelligence*, *30*(1).

Caron, M., Bojanowski, P., Joulin, A., & Douze, M. (2018). Deep clustering for unsupervised learning of visual features. *Proceedings of the European Conference on Computer Vision (ECCV)*, 132–149.

Chamberlain, B., Rowbottom, J., Gorinova, M. I., Bronstein, M., Webb, S., & Rossi, E. (2021). Grand: Graph neural diffusion. *International Conference on Machine Learning*, 1407–1418.

Chami, I., Abu-El-Haija, S., Perozzi, B., Ré, C., & Murphy, K. (2022). Machine learning on graphs: A model and comprehensive taxonomy. *The Journal of Machine Learning Research*, *23*(1), 3840–3903.

Che, F., Yang, G., Zhang, D., Tao, J., & Liu, T. (2021). Self-supervised graph representation learning via bootstrapping. *Neurocomputing*, *456*, 88–96.

Chen, C., Tao, Y., & Lin, H. (2019). Dynamic network embeddings for network evolution analysis.

ArXiv Preprint ArXiv:1906.09860.

Chen, C., Wu, Y., Dai, Q., Zhou, H.-Y., Xu, M., Yang, S., Han, X., & Yu, Y. (2022). A survey on graph neural networks and graph transformers in computer vision: a task-oriented perspective. *ArXiv Preprint ArXiv:2209.13232*.

Chen, F., Wang, Y.-C., Wang, B., & Kuo, C.-C. J. (2020). Graph representation learning: a survey.

APSIPA Transactions on Signal and Information Processing, *9*, e15.

Chen, H., Perozzi, B., Hu, Y., & Skiena, S. (2018). Harp: Hierarchical representation learning for networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, *32*(1).

Chen, J., Liao, S., Hou, J., Wang, K., & Wen, J. (2020). GST-GCN: A Geographic-Semantic-Temporal Graph Convolutional Network for Context-aware Traffic Flow Prediction on Graph Sequences. *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 1604–1609.

Chen, J., Ma, T., & Xiao, C. (2018). Fastgcn: fast learning with graph convolutional networks via importance sampling. *ArXiv Preprint ArXiv:1801.10247*.

Chen, L.-Z., Lin, Z., Wang, Z., Yang, Y.-L., & Cheng, M.-M. (2021). Spatial information guided convolution for real-time rgbd semantic segmentation. *IEEE Transactions on Image Processing*, *30*, 2313–2324.

Chen, M., Wei, Z., Ding, B., Li, Y., Yuan, Y., Du, X., & Wen, J.-R. (2020). Scalable graph neural networks via bidirectional propagation. *Advances in Neural Information Processing Systems*, *33*, 14556–14566.

Chen, M., Wei, Z., Huang, Z., Ding, B., & Li, Y. (2020). Simple and deep graph convolutional networks. *International Conference on Machine Learning*, 1725–1735.

Chen, T., Zhou, K., Duan, K., Zheng, W., Wang, P., Hu, X., & Wang, Z. (2022). Bag of tricks for training deeper graph neural networks: A comprehensive benchmark study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *45*(3), 2769–2781.

Chen, X., Zhang, Y., Du, L., Fang, Z., Ren, Y., Bian, K., & Xie, K. (2020). Tssrgcn: Temporal spectral spatial retrieval graph convolutional network for traffic flow forecasting. *2020 IEEE International Conference on Data Mining (ICDM)*, 954–959.

Chen, Z., Villar, S., Chen, L., & Bruna, J. (2019). On the equivalence between graph isomorphism testing and function approximation with gnns. *Advances in Neural Information Processing Systems*, *32*.

Chiang, W.-L., Liu, X., Si, S., Li, Y., Bengio, S., & Hsieh, C.-J. (2019). Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery \& Data Mining*, 257–266.

Chien, E., Peng, J., Li, P., & Milenkovic, O. (2020). Adaptive universal generalized pagerank graph neural network. *ArXiv Preprint ArXiv:2006.07988*.

Choudhary, N., Rao, N., Katariya, S., Subbian, K., & Reddy, C. K. (2021). Self-supervised hyperboloid representations from logical queries over knowledge graphs. *Proceedings of the*

Web Conference 2021, 1373–1384.

Cui, G., Zhou, J., Yang, C., & Liu, Z. (2020). Adaptive graph encoder for attributed graph embedding. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery \& Data Mining*, 976–985.

Cui, S., Yu, B., Liu, T., Zhang, Z., Wang, X., & Shi, J. (2020). Edge-enhanced graph convolution networks for event detection with syntactic relation. *ArXiv Preprint ArXiv:2002.10757*.

Dash, T., Chitlangia, S., Ahuja, A., & Srinivasan, A. (2022). A review of some techniques for

inclusion of domain-knowledge into deep neural networks. *Scientific Reports*, *12*(1), 1–15.

Dasoulas, G., Santos, L. Dos, Scaman, K., & Virmaux, A. (2019). Coloring graph neural networks for node disambiguation. *ArXiv Preprint ArXiv:1912.06058*.

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.

Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in Neural Information Processing Systems*, *29*.

Deng, W., Zhang, B., Zou, W., Zhang, X., Cheng, X., Guan, L., Lin, Y., Lao, G., Ye, B., Li, X., & others. (2019). Abnormal degree centrality associated with cognitive dysfunctions in early bipolar disorder. *Frontiers in Psychiatry*, *10*, 140.

Dhillon, P. S., Talukdar, P., & Crammer, K. (2010). Learning better data representation using inference-driven metric learning. *Proceedings of the Acl 2010 Conference Short Papers*, 377–381.

Dhingra, N., Ritter, F., & Kunz, A. (2021). BGT-Net: Bidirectional GRU transformer network for scene graph generation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2150–2159.

Ding, M., Kong, K., Li, J., Zhu, C., Dickerson, J., Huang, F., & Goldstein, T. (2021). VQ-GNN: A Universal Framework to Scale up Graph Neural Networks using Vector Quantization. *Advances in Neural Information Processing Systems*, *34*, 6733–6746.

Ding, Y., Yao, Q., Zhao, H., & Zhang, T. (2021). Diffmg: Differentiable meta graph search for heterogeneous graph neural networks. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery \& Data Mining*, 279–288.

Dong, Y., Chawla, N. V, & Swami, A. (2017). metapath2vec: Scalable representation learning for heterogeneous networks. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 135–144.

Dong, Y., Ding, K., Jalaian, B., Ji, S., & Li, J. (2021). AdaGNN: Graph Neural Networks with Adaptive Frequency Response Filter. *Proceedings of the 30th ACM International Conference on Information \& Knowledge Management*, 392–401.

Du, L., Wang, Y., Song, G., Lu, Z., & Wang, J. (2018). Dynamic Network Embedding: An Extended Approach for Skip-gram based Network Embedding. *IJCAI*, *2018*, 2086–2092.

Eliasof, M., Haber, E., & Treister, E. (2021). Pde-gcn: Novel architectures for graph neural networks motivated by partial differential equations. *Advances in Neural Information Processing Systems*, *34*, 3836–3849.

Fang, S., Pan, X., Xiang, S., & Pan, C. (2020). Meta-msnet: Meta-learning based multi-source data fusion for traffic flow prediction. *IEEE Signal Processing Letters*, *28*, 6–10.

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*, 2697–2705.

Fey, M., Lenssen, J. E., Weichert, F., & Leskovec, J. (2021). Gnnautoscale: Scalable and expressive graph neural networks via historical embeddings. *International Conference on Machine Learning*, 3294–3304.

Fornito, A., Zalesky, A., & Breakspear, M. (2013). Graph analysis of the human connectome: promise, progress, and pitfalls. *Neuroimage*, *80*, 426–444.

Fouss, F., Pirotte, A., Renders, J.-M., & Saerens, M. (2007). Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, *19*(3), 355–369.

Gao, H., Chen, Y., & Ji, S. (2019). Learning graph pooling and hybrid convolutional operations for text representations. *The World Wide Web Conference*, 2743–2749.

Gao, Y., Yang, H., Zhang, P., Zhou, C., & Hu, Y. (2019). Graphnas: Graph neural architecture search with reinforcement learning. *ArXiv Preprint ArXiv:1904.09981*.

Gao, Y., Zhang, P., Yang, H., Zhou, C., Tian, Z., Hu, Y., Li, Z., & Zhou, J. (2022). GraphNAS++: Distributed Architecture Search for Graph Neural Networks. *IEEE Transactions on Knowledge and Data Engineering*.

Ge, L., Li, S., Wang, Y., Chang, F., & Wu, K. (2020). Global spatial-temporal graph convolutional network for urban traffic speed prediction. *Applied Sciences*, *10*(4), 1509.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017). Neural message passing for quantum chemistry. *International Conference on Machine Learning*, 1263–1272.

Gomez, A. N., Ren, M., Urtasun, R., & Grosse, R. B. (2017). The reversible residual network: Backpropagation without storing activations. *Advances in Neural Information Processing Systems*, *30*.

Gong, C., Tao, D., Yang, J., & Fu, K. (2014). Signed laplacian embedding for supervised dimension reduction. *Proceedings of the AAAI Conference on Artificial Intelligence*, *28*(1).

Gou, J., Yang, Y., Yi, Z., Lv, J., Mao, Q., & Zhan, Y. (2020). Discriminative globality and locality preserving graph embedding for dimensionality reduction. *Expert Systems with Applications*, *144*, 113079.

Goyal, P., & Ferrara, E. (2018). Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, *151*, 78–94.

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*, 855–864.

Guan, R., Liu, Y., Feng, X., & Li, X. (2021). VPALG: Paper-publication Prediction with Graph Neural Networks. *Proceedings of the 30th ACM International Conference on Information \& Knowledge Management*, 617–626.

Guo, Z., Zhang, X., Mu, H., Heng, W., Liu, Z., Wei, Y., & Sun, J. (2020). Single path one-shot neural architecture search with uniform sampling. *European Conference on Computer Vision*, 544–560.

Gurwitz, D. (2020). Repurposing current therapeutics for treating COVID-19: A vital role of prescription records data mining. *Drug Development Research*, *81*(7), 777–781.

Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Representation learning on graphs: Methods and applications. *ArXiv Preprint ArXiv:1709.05584*.

Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, *30*.

Hasanzadeh, A., Hajiramezanali, E., Boluki, S., Zhou, M., Duffield, N., Narayanan, K., & Qian, X. (2020). Bayesian graph neural networks with adaptive connection sampling. *International Conference on Machine Learning*, 4094–4104.

Hassani, K., & Khasahmadi, A. H. (2020). Contrastive multi-view representation learning on graphs. *International Conference on Machine Learning*, 4116–4126.

He, D., Guo, R., Wang, X., Jin, D., Huang, Y., & Wang, W. (2022). Inflation Improves Graph Neural Networks. *Proceedings of the ACM Web Conference 2022*, 1466–1474.

He, H., Ye, K., & Xu, C.-Z. (2021). Multi-feature Urban Traffic Prediction Based on Unconstrained Graph Attention Network. *2021 IEEE International Conference on Big Data (Big Data)*, 1409–1417.

He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9729–9738.

Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., & Bengio, Y. (2018). Learning deep representations by mutual information estimation and maximization. *ArXiv Preprint ArXiv:1808.06670*.

Hofman, J. M., Watts, D. J., Athey, S., Garip, F., Griffiths, T. L., Kleinberg, J., Margetts, H., Mullainathan, S., Salganik, M. J., Vazire, S., & others. (2021). Integrating explanation and prediction in computational social science. *Nature*, *595*(7866), 181–188.

Hong, D., Gao, L., Yao, J., Zhang, B., Plaza, A., & Chanussot, J. (2020). Graph convolutional networks for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, *59*(7), 5966–5978.

Hu, C., Cheng, L., Sepulcre, J., Johnson, K. A., Fakhri, G. E., Lu, Y. M., & Li, Q. (2015). A spectral graph regression model for learning brain connectivity of Alzheimer's disease. *PloS One*, *10*(5), e0128136.

Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., & Leskovec, J. (2019). Strategies for pre-training graph neural networks. *ArXiv Preprint ArXiv:1905.12265*.

Hu, Z., Dong, Y., Wang, K., Chang, K.-W., & Sun, Y. (2020). Gpt-gnn: Generative pre-training of graph neural networks. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery \& Data Mining*, 1857–1867.

Hu, Z., Fan, C., Chen, T., Chang, K.-W., & Sun, Y. (2019). Pre-training graph neural networks for generic structural feature extraction. *ArXiv Preprint ArXiv:1905.13728*.

Huang, C., Xu, H., Xu, Y., Dai, P., Xia, L., Lu, M., Bo, L., Xing, H., Lai, X., & Ye, Y. (2021). Knowledge-aware coupled graph neural network for social recommendation. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(5), 4115–4122.

Huang, Q., Yamada, M., Tian, Y., Singh, D., & Chang, Y. (2022). Graphlime: Local interpretable model explanations for graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*.

Huang, Z., Wang, Y., Li, C., & He, H. (2022). Going Deeper into Permutation-Sensitive Graph Neural Networks. *ArXiv Preprint ArXiv:2205.14368*.

Huang, Z., Zhang, S., Xi, C., Liu, T., & Zhou, M. (2021). Scaling up graph neural networks via graph coarsening. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery \& Data Mining*, 675–684.

Hurle, M. R., Yang, L., Xie, Q., Rajpal, D. K., Sanseau, P., & Agarwal, P. (2013). Computational drug repositioning: from data to therapeutics. *Clinical Pharmacology \& Therapeutics*, *93*(4), 335–341.

Hussain, M. S., Zaki, M. J., & Subramanian, D. (2022). Global self-attention as a replacement for graph convolution. *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 655–665.

Inuwa-Dutse, I., Liptrott, M., & Korkontzelos, I. (2021). A multilevel clustering technique for community detection. *Neurocomputing*, *441*, 64–78.

James, J. Q. (2021). Citywide Estimation of Travel Time Distributions with Bayesian Deep Graph Learning. *IEEE Transactions on Knowledge and Data Engineering*.

Jiang, W., & Luo, J. (2022). Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications*, 117921.

Jiao, Y., Xiong, Y., Zhang, J., Zhang, Y., Zhang, T., & Zhu, Y. (2020). Sub-graph contrast for scalable self-supervised graph representation learning. *2020 IEEE International Conference on Data Mining (ICDM)*, 222–231.

Jin, W., Derr, T., Liu, H., Wang, Y., Wang, S., Liu, Z., & Tang, J. (2020). Self-supervised learning on graphs: Deep insights and new direction. *ArXiv Preprint ArXiv:2006.10141*.

Jin, W., Liu, X., Zhao, X., Ma, Y., Shah, N., & Tang, J. (2021). Automated self-supervised learning for graphs. *ArXiv Preprint ArXiv:2106.05470*.

Kermarrec, A.-M., Leroy, V., & Trédan, G. (2011). Distributed social graph embedding. *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, 1209–1214.

Khoshraftar, S., & An, A. (2022). A survey on graph representation learning methods. *ArXiv Preprint ArXiv:2204.01855*.

Kim, D., & Oh, A. (2022). How to find your friendly neighborhood: Graph attention design with self-supervision. *ArXiv Preprint ArXiv:2204.04879*.

Kipf, T. N., & Welling, M. (2016a). Semi-supervised classification with graph convolutional networks. *ArXiv Preprint ArXiv:1609.02907*.

Kipf, T. N., & Welling, M. (2016b). Variational graph auto-encoders. *ArXiv Preprint ArXiv:1611.07308*.

Klein, A., Falkner, S., Springenberg, J. T., & Hutter, F. (2016). *Learning curve prediction with Bayesian neural networks*.

Klicpera, J., Bojchevski, A., & Günnemann, S. (2018). Predict then propagate: Combining neural networks with personalized pagerank for classification on graphs. *International Conference on Learning Representations*.

Kondor, R., Shervashidze, N., & Borgwardt, K. M. (2009). The graphlet spectrum. *Proceedings of the 26th Annual International Conference on Machine Learning*, 529–536.

Kriege, N. M., Johansson, F. D., & Morris, C. (2020). *A survey on graph kernels. Appl Netw Sci 5 (1): 6*.

Lai, K.-H., Zha, D., Zhou, K., & Hu, X. (2020). Policy-gnn: Aggregation optimization for graph neural networks. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery \& Data Mining*, 461–471.

Lee, S. Y., Bu, F., Yoo, J., & Shin, K. (2023). Towards Deep Attention in Graph Neural Networks: Problems and Remedies. *ArXiv Preprint ArXiv:2306.02376*.

Leow, Y. Y., Laurent, T., & Bresson, X. (2019). GraphTSNE: a visualization technique for graph-structured data. *ArXiv Preprint ArXiv:1904.06915*.

Levie, R., Monti, F., Bresson, X., & Bronstein, M. M. (2018). Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, *67*(1), 97–109.

Li, B., & Pi, D. (2020). Network representation learning: a systematic literature review. *Neural Computing and Applications*, *32*(21), 16647–16679.

Li, C., Yan, Y., Fu, J., Zhao, Z., & Zeng, Q. (2023). HetReGAT-FC: Heterogeneous Residual Graph Attention Network via Feature Completion. *Information Sciences*, *632*, 424–438.

Li, F., Feng, J., Yan, H., Jin, G., Yang, F., Sun, F., Jin, D., & Li, Y. (2021). Dynamic graph convolutional recurrent network for traffic prediction: Benchmark and solution. *ACM*

Transactions on Knowledge Discovery from Data (TKDD).

Li, G., Müller, M., Ghanem, B., & Koltun, V. (2021). Training graph neural networks with 1000 layers. *International Conference on Machine Learning*, 6437–6449.

Li, I., Yan, V., Li, T., Qu, R., & Radev, D. (2021). Unsupervised cross-domain prerequisite chain learning using variational graph autoencoders. *ArXiv Preprint ArXiv:2105.03505*.

Li, J., Rong, Y., Cheng, H., Meng, H., Huang, W., & Huang, J. (2019). Semi-supervised graph classification: A hierarchical graph perspective. *The World Wide Web Conference*, 972–982.

Li, L., Gan, Z., Cheng, Y., & Liu, J. (2019). Relation-aware graph attention network for visual question answering. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 10313–10322.

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.

Li, Q., Han, Z., & Wu, X.-M. (2018). Deeper insights into graph convolutional networks for semi-supervised learning. *Thirty-Second AAAI Conference on Artificial Intelligence*.

Li, S., Xu, F., Wang, R., & Zhong, S. (2021). Self-supervised incremental deep graph learning for ethereum phishing scam detection. *ArXiv Preprint ArXiv:2106.10176*.

Li, Y., & King, I. (2020). Autograph: Automated graph neural network. *International Conference on Neural Information Processing*, 189–201.

Li, Y., Wen, Z., Wang, Y., & Xu, C. (2021). One-shot graph neural architecture search with dynamic search space. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(10), 8510–8517.

Li, Y., Yu, R., Shahabi, C., & Liu, Y. (2017). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *ArXiv Preprint ArXiv:1707.01926*.

Lian, D., Zhu, Z., Zheng, K., Ge, Y., Xie, X., & Chen, E. (2022). Network Representation Lightening from Hashing to Quantization. *IEEE Transactions on Knowledge and Data Engineering*, *35*(5), 5119–5131.

Lin, J., Cai, Q., & Lin, M. (2021). Multi-label classification of fundus images with graph convolutional network and self-supervised learning. *IEEE Signal Processing Letters*, *28*, 454–458.

Lin, Q., Zhu, F.-Y., Shu, Y.-Q., Zhu, P.-W., Ye, L., Shi, W.-Q., Min, Y.-L., Li, B., Yuan, Q., & Shao, Y. (2021). Altered brain network centrality in middle-aged patients with retinitis pigmentosa: A resting-state functional magnetic resonance imaging study. *Brain and Behavior*, *11*(2), e01983.

Liu, H. X., Simonyan, K., & Yang, Y. M. (2019). DARTS: Differentiable architecture search, presented at the 7th Int. *Conf. Learning Representations, New Orleans, LA, USA*, 9055–9067.

Liu, H., Yang, Y., & Wang, X. (2021). Overcoming catastrophic forgetting in graph neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(10), 8653–8661.

Liu, M., Gao, H., & Ji, S. (2020). Towards deeper graph neural networks. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery \& Data Mining*, 338–348.

Liu, R., Hirn, M., & Krishnan, A. (2023). Accurately modeling biased random walks on weighted networks using node2vec+. *Bioinformatics*, *39*(1), btad047.

Liu, R., Nejati, H., & Cheung, N.-M. (2018). Joint estimation of low-rank components and connectivity graph in high-dimensional graph signals: application to brain imaging. *ArXiv Preprint ArXiv:1801.02303*.

Liu, X., Luo, Z., & Huang, H. (2018). Jointly multiple events extraction via attention-based graph information aggregation. *ArXiv Preprint ArXiv:1809.09078*.

Liu, X., Yan, M., Deng, L., Li, G., Ye, X., & Fan, D. (2021). Sampling methods for efficient training of graph convolutional networks: A survey. *IEEE/CAA Journal of Automatica Sinica*, *9*(2), 205–234.

Liu, X., Zhang, F., Hou, Z., Mian, L., Wang, Z., Zhang, J., & Tang, J. (2021). Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering*.

Liu, Y., Jin, M., Pan, S., Zhou, C., Zheng, Y., Xia, F., & Yu, P. (2022). Graph self-supervised learning: A survey. *IEEE Transactions on Knowledge and Data Engineering*.

Liu, Z., Chen, C., Li, L., Zhou, J., Li, X., Song, L., & Qi, Y. (2019). Geniepath: Graph neural networks with adaptive receptive paths. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(01), 4424–4431.

Liu, Z., Wang, Y., Bernard, J., & Munzner, T. (2022). Visualizing graph neural networks with corgie: Corresponding a graph to its embedding. *IEEE Transactions on Visualization and Computer Graphics*, *28*(6), 2500–2516.

Luo, R., Liao, W., Huang, X., Pi, Y., & Philips, W. (2016). Feature extraction of hyperspectral images with semisupervised graph learning. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *9*(9), 4389–4399.

Ma, L., Rabbany, R., & Romero-Soriano, A. (2021). Graph attention networks with positional embeddings. *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 514–527.

Manessi, F., & Rozza, A. (2021). Graph-based neural network models with multiple self-supervised auxiliary tasks. *Pattern Recognition Letters*, *148*, 15–21.

Mao, K., Zhu, J., Xiao, X., Lu, B., Wang, Z., & He, X. (2021). UltraGCN: ultra simplification of graph convolutional networks for recommendation. *Proceedings of the 30th ACM International Conference on Information \& Knowledge Management*, 1253–1262.

Marcheggiani, D., Bastings, J., & Titov, I. (2018). Exploiting semantics in neural machine translation with graph convolutional networks. *ArXiv Preprint ArXiv:1804.08313*.

Marcheggiani, D., & Titov, I. (2017). Encoding sentences with graph convolutional networks for semantic role labeling. *ArXiv Preprint ArXiv:1703.04826*.

Maron, H., Ben-Hamu, H., Serviansky, H., & Lipman, Y. (2019). Provably powerful graph networks. *Advances in Neural Information Processing Systems*, *32*.

McPherson, M., Smith-Lovin, L., & Cook, J. M. (2001). Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 415–444.

Micheli, A. (2009). Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, *20*(3), 498–511.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, *26*.

Mitrovic, S., & De Weerdt, J. (2019). Dyn2Vec: Exploiting dynamic behaviour using difference networks-based node embeddings for classification. *Proceedings of the International Conference on Data Science*, 194–200.

Mokou, M., Lygirou, V., Angelioudaki, I., Paschalidis, N., Stroggilos, R., Frantzi, M., Latosinska, A., Bamias, A., Hoffmann, M. J., Mischak, H., & others. (2020). A novel pipeline for drug repurposing for bladder cancer based on patients' omics signatures. *Cancers*, *12*(12), 3519.

Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., & Bronstein, M. M. (2017). Geometric deep learning on graphs and manifolds using mixture model cnns. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5115–5124.

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., & Grohe, M. (2019). Weisfeiler and leman go neural: Higher-order graph neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(01), 4602–4609.

Murphy, R., Srinivasan, B., Rao, V., & Ribeiro, B. (2019). Relational pooling for graph representations. *International Conference on Machine Learning*, 4663–4673.

Newman, M. E. J. (2005). A measure of betweenness centrality based on random walks. *Social Networks*, *27*(1), 39–54.

Nguyen, T., & Grishman, R. (2018). Graph convolutional networks with argument-aware pooling for event detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, *32*(1).

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

Niepert, M., Ahmed, M., & Kutzkov, K. (2016). Learning convolutional neural networks for graphs. *International Conference on Machine Learning*, 2014–2023.

Nikolentzos, G., Siglidis, G., & Vazirgiannis, M. (2021). Graph kernels: A survey. *Journal of Artificial Intelligence Research*, *72*, 943–1027.

Noy, A., Nayman, N., Ridnik, T., Zamir, N., Doveh, S., Friedman, I., Giryes, R., & Zelnik, L. (2020). Asap: Architecture search, anneal and prune. *International Conference on Artificial Intelligence and Statistics*, 493–503.

Oellermann, O. R., & Schwenk, A. J. (1991). The Laplacian spectrum of graphs. *Graph Theory, c, Appl*, *2*, 871–898.

Okuda, M., Satoh, S., Sato, Y., & Kidawara, Y. (2021). Unsupervised common particular object discovery and localization by analyzing a match graph. *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1540–1544.

Oloulade, B. M., Gao, J., Chen, J., Lyu, T., & Al-Sabri, R. (2021). Graph neural architecture search: A survey. *Tsinghua Science and Technology*, *27*(4), 692–708.

Oono, K., & Suzuki, T. (2019). Graph neural networks exponentially lose expressive power for node classification. *ArXiv Preprint ArXiv:1905.10947*.

Oord, A. van den, Li, Y., & Vinyals, O. (2018). Representation learning with contrastive predictive coding. *ArXiv Preprint ArXiv:1807.03748*.

Opolka, F. L., Solomon, A., Cangea, C., Veličković, P., Liò, P., & Hjelm, R. D. (2019). Spatio-temporal deep graph infomax. *ArXiv Preprint ArXiv:1904.06316*.

Ou, M., Cui, P., Pei, J., Zhang, Z., & Zhu, W. (2016). Asymmetric transitivity preserving graph embedding. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1105–1114.

Ozaki, K., Shimbo, M., Komachi, M., & Matsumoto, Y. (2011). Using the mutual k-nearest neighbor graphs for semi-supervised classification on natural language data. *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, 154–162.

Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). *The PageRank citation ranking: Bringing order to the web.*

Pan, S., Hu, R., Fung, S., Long, G., Jiang, J., & Zhang, C. (2019). Learning graph embedding with adversarial training methods. *IEEE Transactions on Cybernetics*, *50*(6), 2475–2487.

Pan, S., Hu, R., Long, G., Jiang, J., Yao, L., & Zhang, C. (2018). Adversarially regularized graph autoencoder for graph embedding. *ArXiv Preprint ArXiv:1802.04407*.

Pang, Y., Wu, L., Shen, Q., Zhang, Y., Wei, Z., Xu, F., Chang, E., Long, B., & Pei, J. (2022). Heterogeneous global graph neural networks for personalized session-based recommendation. *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, 775–783.

Papp, P. A., Martinkus, K., Faber, L., & Wattenhofer, R. (2021). Dropgnn: random dropouts increase the expressiveness of graph neural networks. *Advances in Neural Information Processing Systems*, *34*, 21997–22009.

Park, C., Kim, D., Han, J., & Yu, H. (2020). Unsupervised attributed multiplex network embedding. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(04), 5371–5378.

Park, J., Lee, M., Chang, H. J., Lee, K., & Choi, J. Y. (2019). Symmetric graph convolutional autoencoder for unsupervised graph representation learning. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 6519–6528.

Peng, H., Li, J., He, Y., Liu, Y., Bao, M., Wang, L., Song, Y., & Yang, Q. (2018). Large-scale hierarchical text classification with recursively regularized deep graph-cnn. *Proceedings of the 2018 World Wide Web Conference*, 1063–1072.

Peng, H., Wang, H., Du, B., Bhuiyan, M. Z. A., Ma, H., Liu, J., Wang, L., Yang, Z., Du, L., Wang, S., & others. (2020). Spatial temporal incidence dynamic graph neural networks for traffic flow forecasting. *Information Sciences*, *521*, 277–290.

Peng, Z., Dong, Y., Luo, M., Wu, X.-M., & Zheng, Q. (2020). Self-supervised graph representation learning via global context prediction. *ArXiv Preprint ArXiv:2003.01604*.

Peng, Z., Huang, W., Luo, M., Zheng, Q., Rong, Y., Xu, T., & Huang, J. (2020). Graph representation learning via graphical mutual information maximization. *Proceedings of The Web Conference 2020*, 259–270.

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*, 701–710.

Perozzi, B., Kulkarni, V., Chen, H., & Skiena, S. (2017). Don't walk, skip! online learning of multi-scale network embeddings. *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, 258–265.

Pham, H., Guan, M., Zoph, B., Le, Q., & Dean, J. (2018). Efficient neural architecture search via parameters sharing. *International Conference on Machine Learning*, 4095–4104.

Prakash, V. J., & Nithya, D. L. M. (2014). A survey on semi-supervised learning techniques. *ArXiv Preprint ArXiv:1402.4645*.

Pushpakom, S., Iorio, F., Eyers, P. A., Escott, K. J., Hopper, S., Wells, A., Doig, A., Guilliams, T., Latimer, J., McNamee, C., & others. (2019). Drug repurposing: progress, challenges and recommendations. *Nature Reviews Drug Discovery*, *18*(1), 41–58.

Qian, Y., Santus, E., Jin, Z., Guo, J., & Barzilay, R. (2018). GraphIE: A graph-based framework for information extraction. *ArXiv Preprint ArXiv:1810.13083*.

Qiu, J., Chen, Q., Dong, Y., Zhang, J., Yang, H., Ding, M., Wang, K., & Tang, J. (2020). Gcc: Graph contrastive coding for graph neural network pre-training. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery \& Data Mining*, 1150–1160.

Rahman, M., Saha, T. K., Hasan, M. Al, Xu, K. S., & Reddy, C. K. (2018). Dylink2vec: Effective feature representation for link prediction in dynamic networks. *ArXiv Preprint ArXiv:1804.05755*.

Ranjan, E., Sanyal, S., & Talukdar, P. (2020). Asap: Adaptive structure aware pooling for learning hierarchical graph representations. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(04), 5470–5477.

Ren, Y., Liu, B., Huang, C., Dai, P., Bo, L., & Zhang, J. (2020). HDGI: An Unsupervised Graph Neural Network for Representation Learning in Heterogeneous Graph. *AAAI Workshop*.

Ribeiro, L. F. R., Saverese, P. H. P., & Figueiredo, D. R. (2017). struc2vec: Learning node representations from structural identity. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 385–394.

Ringsquandl, M., Sellami, H., Hildebrandt, M., Beyer, D., Henselmeyer, S., Weber, S., & Joblin,

M. (2021). Power to the Relational Inductive Bias: Graph Neural Networks in Electrical Power Grids. *Proceedings of the 30th ACM International Conference on Information \& Knowledge Management*, 1538–1547.

Rong, Y., Bian, Y., Xu, T., Xie, W., Wei, Y., Huang, W., & Huang, J. (2020). Self-supervised graph transformer on large-scale molecular data. *Advances in Neural Information Processing Systems*, *33*, 12559–12571.

Rong, Y., Huang, W., Xu, T., & Huang, J. (2019). Dropedge: Towards deep graph convolutional networks on node classification. *ArXiv Preprint ArXiv:1907.10903*.

Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, *290*(5500), 2323–2326.

Safavi, T., & Koutra, D. (2020). Codex: A comprehensive knowledge graph completion benchmark. *ArXiv Preprint ArXiv:2009.07810*.

Sato, R., Yamada, M., & Kashima, H. (2021). Random features strengthen graph neural networks. *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, 333–341.

Satorras, V. G., & Estrach, J. B. (2018). Few-Shot Learning with Graph Neural Networks. *International Conference on Learning Representations*.

Schnake, T., Eberle, O., Lederer, J., Nakajima, S., Schütt, K. T., Müller, K.-R., & Montavon, G. (2021). Higher-order explanations of graph neural networks via relevant walks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *44*(11), 7581–7596.

Selvaraju, R. R., Lee, S., Shen, Y., Jin, H., Ghosh, S., Heck, L., Batra, D., & Parikh, D. (2019). Taking a hint: Leveraging explanations to make vision and language models more grounded. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2591–2600.

Shen, X.-J., Liu, S.-X., Bao, B.-K., Pan, C.-H., Zha, Z.-J., & Fan, J. (2020). A generalized least-squares approach regularized with graph embedding for dimensionality reduction. *Pattern Recognition*, *98*, 107023.

Shervashidze, N., Vishwanathan, S. V. N., Petri, T., Mehlhorn, K., & Borgwardt, K. (2009). Efficient graphlet kernels for large graph comparison. *Artificial Intelligence and Statistics*, 488–495.

Shi, M., Wilson, D. A., Zhu, X., Huang, Y., Zhuang, Y., Liu, J., & Tang, Y. (2020). Evolutionary architecture search for graph neural networks. *ArXiv Preprint ArXiv:2009.10199*.

Shi, X., Lv, F., Seng, D., Zhang, J., Chen, J., & Xing, B. (2021). Visualizing and understanding graph convolutional network. *Multimedia Tools and Applications*, *80*, 8355–8375.

Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., & Vandergheynst, P. (2013). The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, *30*(3), 83–98.

Si, S., Wang, B., Liu, X., Yu, C., Ding, C., & Zhao, H. (2019). Brain network modeling based on mutual information and graph theory for predicting the connection mechanism in the progression of Alzheimer's disease. *Entropy*, *21*(3), 300.

Subramonian, A. (2021). Motif-driven contrastive learning of graph representations. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(18), 15980–15981.

Sun, F.-Y., Hoffmann, J., Verma, V., & Tang, J. (2019). Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *ArXiv Preprint ArXiv:1908.01000*.

Sun, K., Lin, Z., & Zhu, Z. (2020). Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(04), 5892–5899.

Sun, Q., Li, J., Peng, H., Wu, J., Ning, Y., Yu, P. S., & He, L. (2021). Sugar: Subgraph neural network with reinforcement pooling and self-supervised mutual information mechanism. *Proceedings of the Web Conference 2021*, 2081–2091.

Sun, Z., Deng, Z.-H., Nie, J.-Y., & Tang, J. (2019). Rotate: Knowledge graph embedding by relational rotation in complex space. *ArXiv Preprint ArXiv:1902.10197*.

Taheri, A., Gimpel, K., & Berger-Wolf, T. (2019). Learning to represent the evolution of dynamic graphs with recurrent models. *Companion Proceedings of the 2019 World Wide Web Conference*, 301–307.

Tan, Q., Liu, N., & Hu, X. (2019). Deep representation learning for social network analysis. *Frontiers in Big Data*, *2*, 2.

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*, 1165–1174.

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*, 1067–1077.

Togninalli, M., Ghisu, E., Llinares-López, F., Rieck, B., & Borgwardt, K. (2019). Wasserstein weisfeiler-lehman graph kernels. *Advances in Neural Information Processing Systems*, *32*.

Topping, J., Di Giovanni, F., Chamberlain, B. P., Dong, X., & Bronstein, M. M. (2021). Understanding over-squashing and bottlenecks on graphs via curvature. *ArXiv Preprint ArXiv:2111.14522*.

Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., & Bouchard, G. (2016). Complex embeddings for simple link prediction. *International Conference on Machine Learning*, 2071–2080.

Tu, K., Cui, P., Wang, X., Wang, F., & Zhu, W. (2018). Structural deep embedding for hyper-

networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, *32*(1).

Tu, K., Cui, P., Wang, X., Yu, P. S., & Zhu, W. (2018). Deep recursive network embedding with regular equivalence. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery \& Data Mining*, 2357–2366.

Tu, K., Ma, J., Cui, P., Pei, J., & Zhu, W. (2019). Autone: Hyperparameter optimization for massive network embedding. *Proceedings of the 25th ACM SIGKDD International Conference on*

Knowledge Discovery \& Data Mining, 216–225.

Urry, M. J., & Sollich, P. (2013). Random walk kernels and learning curves for gaussian process regression on random graphs. *The Journal of Machine Learning Research*, *14*(1), 1801–1835.

Van Engelen, J. E., & Hoos, H. H. (2020). A survey on semi-supervised learning. *Machine Learning*, *109*(2), 373–440.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. *Stat*, *1050*, 20.

Velickovic, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., & Hjelm, R. D. (2019). Deep Graph Infomax. *ICLR (Poster)*, *2*(3), 4.

Wan, S., Pan, S., Yang, J., & Gong, C. (2021). Contrastive and generative graph convolutional networks for graph-based semi-supervised learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(11), 10049–10057.

Wang, C., Pan, S., Long, G., Zhu, X., & Jiang, J. (2017). Mgae: Marginalized graph autoencoder for graph clustering. *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 889–898.

Wang, C., Wang, C., Wang, Z., Ye, X., & Yu, P. S. (2020). Edge2vec: Edge-based social network embedding. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *14*(4), 1–24.

Wang, D., Cui, P., & Zhu, W. (2016). Structural deep network embedding. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1225–1234.

Wang, H., Yin, H., Zhang, M., & Li, P. (2022). Equivariant and stable positional encoding for more powerful graph neural networks. *ArXiv Preprint ArXiv:2203.00199*.

Wang, H., Zhang, F., Zhang, M., Leskovec, J., Zhao, M., Li, W., & Wang, Z. (2019). Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery \& Data Mining*, 968–977.

Wang, P., Agarwal, K., Ham, C., Choudhury, S., & Reddy, C. K. (2021). Self-supervised learning of contextual embeddings for link prediction in heterogeneous networks. *Proceedings of the Web Conference 2021*, 2946–2957.

Wang, P., Wu, Q., Cao, J., Shen, C., Gao, L., & Hengel, A. van den. (2019). Neighbourhood watch: Referring expression comprehension via language-guided graph attention networks. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1960–1968.

Wang, S., Wang, R., Yao, Z., Shan, S., & Chen, X. (2020). Cross-modal scene graph matching for relationship-aware image-text retrieval. *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 1508–1517.

Wang, Z., Lin, G., Tan, H., Chen, Q., & Liu, X. (2020). CKAN: collaborative knowledge-aware attentive network for recommender systems. *Proceedings of the 43rd International ACM*

SIGIR Conference on Research and Development in Information Retrieval, 219–228.

Weber, M., Chen, J., Suzumura, T., Pareja, A., Ma, T., Kanezashi, H., Kaler, T., Leiserson, C. E., & Schardl, T. B. (2018). Scalable graph learning for anti-money laundering: A first look. *ArXiv Preprint ArXiv:1812.00076*.

Weisfeiler, B., & Leman, A. (1968). The reduction of a graph to canonical form and the algebra which appears therein. *Nti, Series*, *2*(9), 12–16.

Wen, W., Liu, H., Chen, Y., Li, H., Bender, G., & Kindermans, P.-J. (2020). Neural predictor for neural architecture search. *European Conference on Computer Vision*, 660–676.

Weston, J., Ratle, F., & Collobert, R. (2008). Deep learning via semi-supervised embedding. *Proceedings of the 25th International Conference on Machine Learning*, 1168–1175.

Wijesinghe, A., & Wang, Q. (2021). A New Perspective on" How Graph Neural Networks Go Beyond Weisfeiler-Lehman?". *International Conference on Learning Representations*.

Wink, A. M., Tijms, B. M., Ten Kate, M., Raspor, E., de Munck, J. C., Altena, E., Ecay-Torres, M., Clerigue, M., Estanga, A., Garcia-Sebastian, M., & others. (2018). Functional brain network centrality is related to APOE genotype in cognitively normal elderly. *Brain and Behavior*, *8*(9), e01080.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., & Weinberger, K. (2019). Simplifying graph convolutional networks. *International Conference on Machine Learning*, 6861–6871.

Wu, J., Wang, X., Feng, F., He, X., Chen, L., Lian, J., & Xie, X. (2021). Self-supervised graph learning for recommendation. *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 726–735.

Wu, Y., Warner, J. L., Wang, L., Jiang, M., Xu, J., Chen, Q., Nian, H., Dai, Q., Du, X., Yang, P., & others. (2019). Discovery of noncancer drug effects on survival in electronic health records of patients with cancer: a new paradigm for drug repurposing. *JCO Clinical Cancer Informatics*, *3*, 1–9.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Philip, S. Y. (2020). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(1), 4–24.

Xie, Y., Li, S., Yang, C., Wong, R. C.-W., & Han, J. (2020). When do gnns work: Understanding and improving neighborhood aggregation. *IJCAI'20: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence,$\{$IJCAI$\}$ 2020*, *2020*(1).

Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2018). How powerful are graph neural networks? *ArXiv Preprint ArXiv:1810.00826*.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K., & Jegelka, S. (2018). Representation learning on graphs with jumping knowledge networks. *International Conference on Machine Learning*, 5453–5462.

Xu, K., Zhang, M., Jegelka, S., & Kawaguchi, K. (2021). Optimization of graph neural networks: Implicit acceleration by skip connections and more depth. *International Conference on*

Machine Learning, 11592–11602.

Xu, Q.-H., Li, Q.-Y., Yu, K., Ge, Q.-M., Shi, W.-Q., Li, B., Liang, R.-B., Lin, Q., Zhang, Y.-Q., & Shao, Y. (2020). Altered brain network centrality in patients with diabetic optic neuropathy: a resting-state FMRI study. *Endocrine Practice, 26*(12), 1399–1405.

Yang, B., Yih, W., He, X., Gao, J., & Deng, L. (2014). Embedding entities and relations for learning and inference in knowledge bases. *ArXiv Preprint ArXiv:1412.6575*.

Yang, R., Shi, J., Xiao, X., Yang, Y., & Bhowmick, S. S. (2019). Homogeneous network embedding for massive graphs via reweighted personalized pagerank. *ArXiv Preprint ArXiv:1906.06826*.

Yang, X., Tang, K., Zhang, H., & Cai, J. (2019). Auto-encoding scene graphs for image captioning. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10685–10694.

Yang, Z., Cohen, W., & Salakhudinov, R. (2016). Revisiting semi-supervised learning with graph embeddings. *International Conference on Machine Learning*, 40–48.

Yehudai, G., Fetaya, E., Meirom, E., Chechik, G., & Maron, H. (2021). From local structures to size generalization in graph neural networks. *International Conference on Machine Learning*, 11975–11986.

Yin, X., Wu, G., Wei, J., Shen, Y., Qi, H., & Yin, B. (2021). Deep learning on traffic prediction: Methods, analysis and future directions. *IEEE Transactions on Intelligent Transportation Systems*.

Yoon, M., Gervet, T., Hooi, B., & Faloutsos, C. (2020). Autonomous graph mining algorithm search with best speed/accuracy trade-off. *2020 IEEE International Conference on Data Mining (ICDM)*, 751–760.

Yoon, M., Gervet, T., Hooi, B., & Faloutsos, C. (2022). Autonomous graph mining algorithm search with best performance trade-off. *Knowledge and Information Systems*, 1–32.

Yoon, M., Gervet, T., Shi, B., Niu, S., He, Q., & Yang, J. (2021). Performance-Adaptive Sampling Strategy Towards Fast and Accurate Graph Neural Networks. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery \& Data Mining*, 2046–2056.

You, J., Gomes-Selman, J. M., Ying, R., & Leskovec, J. (2021). Identity-aware graph neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(12), 10737–10745.

You, J., Ying, Z., & Leskovec, J. (2020). Design space for graph neural networks. *Advances in Neural Information Processing Systems*, *33*, 17009–17021.

You, Y., Chen, T., Wang, Z., & Shen, Y. (2020). When does self-supervision help graph convolutional networks? *International Conference on Machine Learning*, 10871–10880.

Yu, B., Yin, H., & Zhu, Z. (2017). Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *ArXiv Preprint ArXiv:1709.04875*.

Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., & Huang, T. S. (2018). Generative image inpainting with contextual attention. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5505–5514.

Yu, J., Lu, Y., Qin, Z., Zhang, W., Liu, Y., Tan, J., & Guo, L. (2018). Modeling text with graph convolutional network for cross-modal information retrieval. *Pacific Rim Conference on Multimedia*, 223–234.

Yu, J., Xu, T., & He, R. (2021). Towards the explanation of graph neural networks in digital pathology with information flows. *ArXiv Preprint ArXiv:2112.09895*.

Yun, S., Jeong, M., Kim, R., Kang, J., & Kim, H. J. (2019). Graph transformer networks. *Advances in Neural Information Processing Systems*, *32*.

Zeng, H., Zhang, M., Xia, Y., Srivastava, A., Malevich, A., Kannan, R., Prasanna, V., Jin, L., & Chen, R. (2021). Decoupling the depth and scope of graph neural networks. *Advances in Neural Information Processing Systems*, *34*, 19665–19679.

Zeng, H., Zhou, H., Srivastava, A., Kannan, R., & Prasanna, V. (2019). Graphsaint: Graph sampling based inductive learning method. *ArXiv Preprint ArXiv:1907.04931*.

Zeng, J., & Xie, P. (2021). Contrastive self-supervised learning for graph classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(12), 10824–10832.

Zhang, C., Zhang, K., Yuan, Q., Peng, H., Zheng, Y., Hanratty, T., Wang, S., & Han, J. (2017). Regions, periods, activities: Uncovering urban dynamics via cross-modal representation learning. *Proceedings of the 26th International Conference on World Wide Web*, 361–370.

Zhang, D., Huang, X., Liu, Z., Hu, Z., Song, X., Ge, Z., Zhang, Z., Wang, L., Zhou, J., Shuang, Y., & others. (2020). Agl: a scalable system for industrial-purpose graph machine learning. *ArXiv Preprint ArXiv:2003.02454*.

Zhang, D., Yin, J., Zhu, X., & Zhang, C. (2018). Network representation learning: A survey. *IEEE Transactions on Big Data*, *6*(1), 3–28.

Zhang, H., Lin, S., Liu, W., Zhou, P., Tang, J., Liang, X., & Xing, E. P. (2020). Iterative graph self-distillation. *ArXiv Preprint ArXiv:2010.12609*.

Zhang, J., Kuo, A.-T., Zhao, J., Wen, Q., Winstanley, E., Zhang, C., & Ye, Y. (2021). RxNet: Rx-refill Graph Neural Network for Overprescribing Detection. *Proceedings of the 30th ACM International Conference on Information \& Knowledge Management*, 2537–2546.

Zhang, J., Shi, X., Xie, J., Ma, H., King, I., & Yeung, D.-Y. (2018). Gaan: Gated attention networks for learning on large and spatiotemporal graphs. *ArXiv Preprint ArXiv:1803.07294*.

Zhang, J., Wang, Y., Yuan, Z., & Jin, Q. (2019). Personalized real-time movie recommendation system: Practical prototype and evaluation. *Tsinghua Science and Technology*, *25*(2), 180–191.

Zhang, J., Zhang, H., Xia, C., & Sun, L. (2020). Graph-bert: Only attention is needed for learning graph representations. *ArXiv Preprint ArXiv:2001.05140*.

Zhang, M., & Li, P. (2021). Nested graph neural networks. *Advances in Neural Information Processing Systems*, *34*, 15734–15747.

Zhang, Y., Li, Y., Zhou, X., Liu, Z., & Luo, J. (2021). C 3-GAN: Complex-Condition-Controlled Urban Traffic Estimation through Generative Adversarial Networks. *2021 IEEE International Conference on Data Mining (ICDM)*, 1505–1510.

Zhang, Y., Qi, P., & Manning, C. D. (2018). Graph convolution over pruned dependency trees improves relation extraction. *ArXiv Preprint ArXiv:1809.10185*.

Zhang, Y., Ti\vno, P., Leonardis, A., & Tang, K. (2021). A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *5*(5), 726–742.

Zhang, Y., Wu, B., Liu, Y., & Lv, J. (2019). Local community detection based on network motifs. *Tsinghua Science and Technology*, *24*(6), 716–727.

Zhang, Z., Cui, P., Wang, X., Pei, J., Yao, X., & Zhu, W. (2018). Arbitrary-order proximity preserved network embedding. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery \& Data Mining*, 2778–2786.

Zhang, Z., Cui, P., & Zhu, W. (2020). Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, *34*(1), 249–270.

Zhao, H., Wei, L., & Yao, Q. (2020). Simplifying architecture search for graph neural network. *ArXiv Preprint ArXiv:2008.11652*.

Zhao, H., Yao, Q., & Tu, W. (2021). Search to aggregate neighborhood for graph neural network. *ArXiv Preprint ArXiv:2104.06608*.

Zhao, L., & Akoglu, L. (2019). Pairnorm: Tackling oversmoothing in gnns. *ArXiv Preprint ArXiv:1909.12223*.

Zhao, L., Chen, M., Du, Y., Yang, H., & Wang, C. (2022). Spatial-Temporal Graph Convolutional Gated Recurrent Network for Traffic Forecasting. *ArXiv Preprint ArXiv:2210.02737*.

Zhao, Y., Wang, D., Gao, X., Mullins, R., Lio, P., & Jamnik, M. (2020). Probabilistic dual network architecture search on graphs. *ArXiv Preprint ArXiv:2003.09676*.

Zhao, Z., Zhang, X., Zhou, H., Li, C., Gong, M., & Wang, Y. (2020). HetNERec: Heterogeneous network embedding based recommendation. *Knowledge-Based Systems*, *204*, 106218.

Zheng, X., Ji, R., Wang, Q., Ye, Q., Li, Z., Tian, Y., & Tian, Q. (2020). Rethinking performance estimation in neural architecture search. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11356–11365.

Zhou, F., & Cao, C. (2021). Overcoming catastrophic forgetting in graph neural networks with experience replay. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(5), 4714–4722.

Zhou, G., & Xia, J. (2018). OmicsNet: a web-based tool for creation and visual analysis of biological networks in 3D space. *Nucleic Acids Research*, *46*(W1), W514--W522.

Zhou, H., Yang, M., Wang, J., & Pan, W. (2019). Bayesnas: A bayesian approach for neural architecture search. *International Conference on Machine Learning*, 7603–7613.

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

Zhou, K., Huang, X., Li, Y., Zha, D., Chen, R., & Hu, X. (2020). Towards deeper graph neural networks with differentiable group normalization. *Advances in Neural Information Processing Systems*, *33*, 4917–4928.

Zhou, K., Song, Q., Huang, X., & Hu, X. (2019). Auto-gnn: Neural architecture search of graph neural networks. *ArXiv Preprint ArXiv:1909.03184*.

Zhou, Y., Zheng, H., Huang, X., Hao, S., Li, D., & Zhao, J. (2022). Graph neural networks: Taxonomy, advances, and trends. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *13*(1), 1–54.

Zhu, X., Ghahramani, Z., & Lafferty, J. D. (2003). Semi-supervised learning using gaussian fields and harmonic functions. *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 912–919.

Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., & Wang, L. (2020). Deep graph contrastive representation learning. *ArXiv Preprint ArXiv:2006.04131*.

Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., & Wang, L. (2021). Graph contrastive learning with adaptive augmentation. *Proceedings of the Web Conference 2021*, 2069–2080.

Zhuang, C., & Ma, Q. (2018). Dual graph convolutional networks for graph-based semi-supervised classification. *Proceedings of the 2018 World Wide Web Conference*, 499–508.

Zhuang, L., Zhou, Z., Gao, S., Yin, J., Lin, Z., & Ma, Y. (2017). Label information guided graph construction for semi-supervised learning. *IEEE Transactions on Image Processing*, *26*(9), 4182–4192.

Zitouni, M. S., Sluzek, A., & Bhaskar, H. (2019). Visual analysis of socio-cognitive crowd behaviors for surveillance: A survey and categorization of trends and methods. *Engineering Applications of Artificial Intelligence*, *82*, 294–312.

Zuo, X.-N., Ehmke, R., Mennes, M., Imperati, D., Castellanos, F. X., Sporns, O., & Milham, M. P. (2012). Network centrality in the human functional connectome. *Cerebral Cortex*, *22*(8), 1862–1875.