Searching for Optimal Heterogeneous Graph Neural Networks: A Comparative and Explainable Approach with VAC-HGNN

Category: Research

Paper Type: application/design study

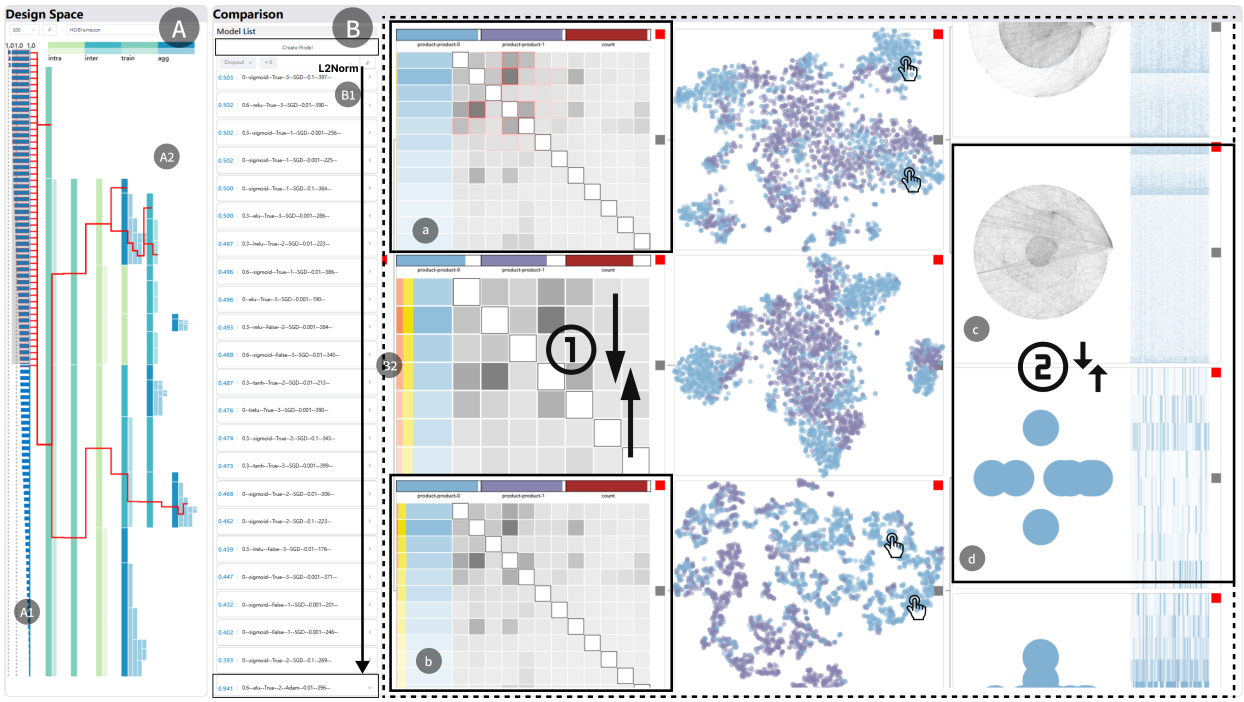


Fig. 1: *VAC-HGNN* is composed of two main components: the *Design Space View* (A) and the *Comparison View* (B). The *Design Space View* displays a subset of models within the design space and their corresponding search directions. The *Comparison View*, on the other hand, allows users to perform a detailed comparison of selected models, which is further divided into three subcomponents: the *Model List* (B1), and *Model Exploration* (B2).

Abstract—Heterogeneous graphs have found widespread application in real-world data sets that contain both semantic and structural information. Graph neural networks (GNNs) have emerged as a popular tool for analyzing such graphs, and their design space has been extensively researched. However, existing methods for searching the design space of heterogeneous graph neural networks (HGNNs) tend to produce suboptimal results and can be computationally expensive, while also lacking explainability. To address these limitations, we propose a methodology that compares searched models to identify optimal search directions, while also incorporating explainability. Additionally, we create a visual analytics system called *VAC-HGNN* (**V**isual **A**nalytics for **C**omparing **HGNN**s), which enables HGNN practitioners to understand and compare HGNNs at three levels: graph level, group level, and individual level. The system includes a design space view and a comparison view with rich interactions. We demonstrate the effectiveness and usability of *VAC-HGNN* through two case studies and expert interviews.

Index Terms—Heterogeneous graphs, graph neural network, neural architecture search, design space, comparative analysis

1 Introduction

Heterogeneous graphs (HGs) are networks that consist of diverse entities (nodes) and relationships (edges) [45], providing a means to model intricate real-world data that contains both semantic and structural information [53, 66]. One of the key downstream tasks of HGs is link prediction in graph analysis [30], which includes tasks such as website hyperlink prediction in web science [73], recommendations in e-commerce [24, 27], and synthetic lethality prediction in bioinformatics [52]. The rising trend of link prediction using machine learning has led to the proposal of graph neural networks (GNNs), which can effectively harness and consolidate information from graph structures, and possess strong representational abilities for graph data. Consequently, GNNs have become a prevalent method for graph analysis [56, 71]. Following this development, to better capitalize on the heterogeneous structural (graph) and content (node embedding) information available in HGs, heterogeneous graph neural networks (HGNNs) have been introduced, resulting in an enhancement in the predictive accuracy of link prediction tasks on HGs [53, 66].

The proliferation of HGNNs has resulted in the emergence of various novel model architectures [13, 70], such as HAN [54] and MAGNN [11]. Nonetheless, a pressing issue that arises from this development is the need to effectively compare and identify a suitable model for a specific dataset and task context. In particular, the selection of a suitable HGNN entails an essential consideration of the search space of HGNN [13, 61, 70]. This search space of HGNNs contains two components: *architecture components (ACs)*, such as aggregation functions, and *hyperparameters (HPs)*, such as learning rate [13, 37, 70]. Given

the typically vast size of the HGNN search space, several studies have proposed methods for automating the search for graph neural structures to facilitate the selection and comparison of HGNNs [12, 13, 61, 70]. However, these studies are deficient in the following two aspects. First, although the network architectures generated through these methods have demonstrated commendable performance, their results are **either suboptimal within a restricted search space or require significant computational resources for large-scale searches**. For instance, certain techniques like *Auto-GNN* [72] and *GraphNAS* [12] concentrate solely on intra-layer designs such as activation functions, without taking into account inter-layer designs such as layer connectivity. Additionally, some search space design techniques [12, 69] explore only ACs with fixed HPs, which thus can result in suboptimal models. Despite other methods [5, 61, 70] that consider a combination of ACs and HPs with diverse possible options [37], the high computational costs still pose a significant challenge in conducting the search. Therefore, it is crucial to identify appropriate search strategies that minimize search costs and ensure optimization by employing fine-grained comparisons. Second, the search process in studies on graph neural network search (GraphNAS) has been observed to prioritize the selection of AC and HP based on quantitative metrics, while **failing to provide explanations for the distinct behaviors of the designs**. This underscores the importance of interpretability in HGNNs, particularly in contexts such as biomedical research [6], crime prediction [50], and fraud detection [19], where relevant stakeholders must comprehend and rely on the predictions of HGNNs [59]. To achieve a better understanding of the behavior of HGNN models, various GNN interpretation algorithms [63] and visual analytics (VA) solutions [7, 22, 25, 34, 40] have been proposed. GNN interpretation algorithms are often employed to identify the nodes and features that significantly contribute to the prediction, with the goal of improving the understanding of the model. However, these methods are generally restricted to detailed analysis of individual models, and fail to provide comparative insights across multiple models. On the other hand, existing VA solutions mainly focus on error pattern diagnosis of individual GNNs from a geometric metric perspective [25], or evaluating the extent to which graph node embedding preserves input graph features through correspondence between the three spaces (i.e. the latent space, the topology space, and the feature space) [34]. While these methods attempt to provide insight into model behavior, they are unable to facilitate comparisons between different GNNs. Consequently, there arises a necessity to conduct comprehensive model comparisons and thoroughly investigate the search directions within the space, with the aim of shedding light on the performance of HGNN models and identifying appropriate ones for downstream tasks.

In this study, we propose a novel visual analytics system, named *VAC-HGNN* (**V**isual **A**nalytics for **C**omparing **HGNN**s), designed to facilitate in-depth comprehension of HGNNs with regard to link prediction, as well as enable comparative analyses of these models. To achieve a comprehensive understanding of various HGNNs, a two-stage visualization approach is proposed, comprising a design space view and a comparison view. In the design space view, we evaluate the performances of thousands of HGNNs for different datasets and categorize them using Decision Tree algorithms. Representative models are then selected for exploration in the comparison view, which incorporates a three-level visualization consisting of a graph level, a group level, and an individual level. At the graph level, the latent space view aids in exploring the overall projection of results and identifying groups of interest. Users can select a group and analyze behavioral patterns. At a more granular level, the individual visualization with k-hop graphs assists users in checking the structural information learned, with GNN interpretation algorithms supporting the analysis of significance. To enable detailed comparative studies between different HGNNs, the two stages are connected, and coordinated interactions are provided for comparisons at the three aforementioned levels. Two case studies are performed on two real-world scenarios to demonstrate the efficacy of the system. The main contributions of this study are summarized as follows: **1) A dataset** consisting of numerous HGNNs searched across various domains; **2) A visual analytics system** that enables comprehensive understanding and comparative analysis of HGNNs at three distinct levels within the design space; **3) Two case studies** and **expert interviews** in different domains, which serve to showcase the effectiveness and usability of the proposed *VAC-HGNN* system.

2 Related Work

2.1 Heterogeneous Graphs

Embedding is a critical task in hypergraph analysis [53]. Embedding methods can be broadly categorized into shallow or deep models [53]. Shallow network embeddings can be obtained through random-walk methods or homogeneous graph decomposition. In contrast, deep methods use neural networks to obtain embeddings and commonly rely on message passing and GNNs [56]. Our work primarily focuses on deep network embeddings, which offer greater potential for deeper exploration despite their lower interpretability.

Homogeneous GNNs can be categorized into spectral and spatial methods. Spectral-based GNNs, such as the Graph Convolutional Network (GCN) [26], utilize the Graph Fourier Transform, but require access to the entire graph's data and cannot generate embeddings for new nodes. To overcome this limitation, GraphSAGE [15] introduced a generalized aggregation function and utilized sampling. Graph Attention Networks (GAT) [49] incorporate attention mechanisms. Heterogeneous GNNs, such as Relational Graph Convolutional Network (RGCN) [43], use multiple convolution matrices to address edge heterogeneity. Heterogeneous attention mechanisms are used by models like Heterogeneous Graph Attention Networks (HGAT) [31], Heterogeneous Graph Transformer (HGT) [20], and Heterogeneous Self-Attention Neural Network (HetSANN) [18]. Meta-path-based models, including Heterogeneous Graph Attention Network (HAN) [54], Graph Transformer Networks (GTN) [65], and Multi-Agent Graph Neural Network (MAGNN) [11], account for the special structure of heterogeneous graphs. Another aspect that researchers consider when designing models for heterogeneous graphs is the downstream application that the model will be used for, such as recommendation [10, 32] and classification [68, 74], among others. In our work, we focus on link prediction as a specific task to enhance practitioners' understanding of the proposed models.

2.2 Design Space of GNN

Given the significant progress that GNNs have recently achieved, designing and applying them has become a crucial research challenge that requires domain knowledge from developers. To this end, GraphGym [61] has established a GNN design space and a GNN task space, aiming to identify the most suitable GNN design for a specific task. The design space comprises intra-layer design, inter-layer design, and learning configuration, resulting in 315,000 potential designs. After extensive experiments, a condensed design space of 96 designs has been proposed. Additionally, a design space for GNNs based on collaborative filtering has been profiled [55]. As this concept can also be applied to heterogeneous graphs, Space4HGNN [70] has defined a design space for HGNNs, inspired by GraphGym. Nevertheless, due to the heterogeneity of HGs, the condensed design space is much larger, containing 70,000 combinations, which requires random search. Although the design space has been significantly condensed, developers still need to devote effort to testing and selecting appropriate designs in practice. Neural Architecture Search (NAS) has been applied to GNNs, with reinforcement learning commonly used as the search algorithm [37]. Examples of such approaches include GraphNAS [12] and Autognn [72]. To facilitate better understanding of the architecture space, ArchExplorer [64] has utilized visualization techniques. Furthermore, Graph NAS methods have been extended to heterogeneous networks as well [13]. In contrast to the aforementioned design space approaches, these methods focus solely on the architecture within the layer, without considering the task space.

Although techniques like unified frameworks provide an efficient way to evaluate models, they often lack interpretability and may not provide sufficient insight into the model's inner workings. Result-oriented evaluations may not be fine-grained enough to provide a detailed understanding of the process. In our work, we leverage visualization techniques to enhance model interpretability in a fine-grained manner.

2.3 GNN Interpretation Algorithms

GNNs are often regarded as "black boxes" that lack human-interpretable explanations, which undermines their credibility and transparency when applied. To address this issue, many explanation methods have been proposed for GNNs [63], which can be classified into model-level (e.g., XGNN [62]) and instance-level methods. Instance-level methods include gradients/feature-based methods [3, 39], perturbation-based methods [35, 59], decomposition methods [39, 44], and surrogate methods [21, 67]. Gradients/feature-based methods assess the input importance by analyzing the gradients or hidden feature map. Perturbation-based methods study the output with different input perturbations. GNNExplainer [59], for instance, uses Mutual Information (MI) to identify subgraphs and explain the GNN. Decomposition methods decompose the original model predictions into several terms to determine the input importance. Surrogate methods aim to approximate the complex deep model with a simple surrogate model to predict the neighboring areas of the input example. For example, GraphLime [21] learns a nonlinear interpretable model locally from the neighborhoods of the selected node's subgraph. Sanchez-Lengeling et al. [42] have underscored the importance of crucial features for predictions as a means to assess the dependability of GNN techniques. Agarwal et al. [1], on the other hand, have carried out a theoretical and empirical investigation using three metrics, namely faithfulness, stability, and fairness preservation, to evaluate the aforementioned methods.

Prior research has explored diverse algorithms to provide interpretability for GNN models, which serves as the basis of our analysis. Our current study centers primarily on perturbation-based approaches.

2.4 Visual Analytics of Explainable AI

Many VA solutions have been proposed to help ML practitioners understand models, including the comparison of embedding spaces and the corresponding visualization design, and some VA studies on GNN.

Visualizations of Embedding Spaces. *EmbComp* [17] and *Embedding Comparator* [4] are interactive systems for comparing global and local embeddings. Meanwhile, *Parallel Embeddings* [2] rely on clustering under comparison conditions. Cutura et al. [8] used dimensionality reduction algorithms and matrix visualization for comparison. Emblaze [46] supported comparisons in multiple embedding spaces. Several works compare embeddings generated by several network embedding techniques at the node and graph level, such as *EmbeddingVis* [29] and *GEMvis* [7]. These systems do not take into account the embedding model or focus only on embeddings of homogeneous graphs, while we focus on comparing embeddings generated by different heterogeneous network embedding methods. In addition, some studies visualize the embedding space. For example, LYi et al. [36] surveyed more than 100 research papers that refer to Gleicher's theory [14] and summarized the comparison layouts. Liu et al. [33] proposed latent space cartography to map and compare vectors in the embedding space. In addition, Stahnke et al. [47] designed a set of interaction techniques for low-dimensional data. These studies can provide inspiration for us to design an interactive visual comparison system.

Visual Analytics of GNN. Since it is difficult for developers to understand details of the GNN process and make evaluations deeply, many VA systems have been designed as useful tools. *GNNLens* [25] can help developers quickly identify error patterns through Parallel Sets View and Projection View. BiaScope [40] shows a visual unfairness diagnosis workflow. They both use diagnosis as the purpose of their study. *GNNVis* [22] is a framework for visualizing high-dimensional data in low-dimensional space. *CorGIE* [34] represents the input graph with its node embeddings to understand GNN. *DrugExplorer* [51] focused on explanations of GNN-based drug repurposing. Unlike them, our work is to compare graph embeddings generated by different models and help developers to make decisions without the restriction of a certain research field.

3 Observational Study

To understand how to choose and compare HG embedding models, we conducted 40-minute independent semi-structured interviews with four machine learning (ML) experts (E1-E3) from various fields to identify their needs and challenges. E1, a male algorithm engineer with two years of experience in the gaming industry, E2, a male university professor with 12 years of experience in the bioinformatics field, and E3, a male Ph.D. student in a medical intelligence laboratory with two years of experience. Through this process, we were able to derive two demands based on the needs expressed by the interviewees.

3.1 Demands Related to HG Embedding Model Selection

In light of the findings from the interviews, we have synthesized two key demands deemed significant by domain experts in the process of selecting the HG embedding model.

D1) Select and compare candidate models. Due to the considerable number of established models and associated research concerning HG, developers are faced with a plethora of possible alternatives prior to conducting comparisons. E1 and E3 indicated that they typically initiate their search by experimenting with various prevalent graph neural network algorithms within their respective fields, and subsequently fine-tune the hyperparameters in an effort to obtain satisfactory outcomes. Moreover, based on our previous literature research, NAS [12] can be employed as a means of exploring the design space and identifying potential models [70]. E1 has attempted to utilize NAS in their research endeavors; however, the vast search space posed a significant hindrance, ultimately preventing successful implementation. On the other hand, both E2 and E3 have yet to employ NAS in their research, but harbor a sense of optimism towards its potential benefits. Following the comparative analysis, developers aim to identify appropriate models that are suitable for deployment in downstream tasks.

D2) Explore the reasons behind the model selection. GNNs are often regarded as "black boxes", which presents a challenge for developers to understand the factors that contribute to the superior performance of a chosen model based solely on the input and output. As E2 pointed out, in the field of biology, explainability is highly valued, "*particularly in the context of drug development where a clear rationale is necessary prior to regulatory approval*"; "*while statistics has emerged as a common language in this domain, machine learning has yet to achieve comparable acceptance*". E2 posited that visualization techniques may serve to enhance explainability and promote the seamless integration of ML into the field of biology. E1 expressed a similar sentiment, stating that "*explainability is highly beneficial if the accuracy of a model can be ensured"*.

3.2 Design Requirements

In light of the demands previously discussed, we have formulated six design requirements that cater to different spaces, i.e., the latent and topology space and levels, which encompass local/individual, group, and global levels.

R1: Provides guidance for selecting candidate models in the design space for comparison. Given the vastness of the design space, it is crucial to determine the search direction of candidate models to facilitate subsequent comparisons **[D1]**. To this end, E1 expresses his thoughts on the NAS approach and design space, "*we have considered NAS before, but it is computationally expensive and cumbersome to search, so we prefer to start with simple mainstream models and subsequently tune them gradually*; "*Alternatively, an approach that highlights the importance of each dimension can also be employed*. Therefore, a method that can help developers meticulously scrutinize the design dimensions and parameters of candidate models to arrive at informed decisions is needed.

R2: Present a summary of the selected models' performances at a global level. To facilitate users in forming intuitive impressions, it is essential to provide an overall picture of the selected models' performances. All domain experts confirmed that quantitative metrics are still commonly used to evaluate model performance. For instance, *AUC* is frequently utilized in link prediction tasks, and *Micro-F1* and *Macro-F1* are often employed in classification, while *top-k* related metrics are applied in recommendation tasks. Hence, the system should retain these performance metrics of the chosen models **[D1]**. Additionally, the graph information in the topology space could also help in swiftly comprehending the dataset's situation.

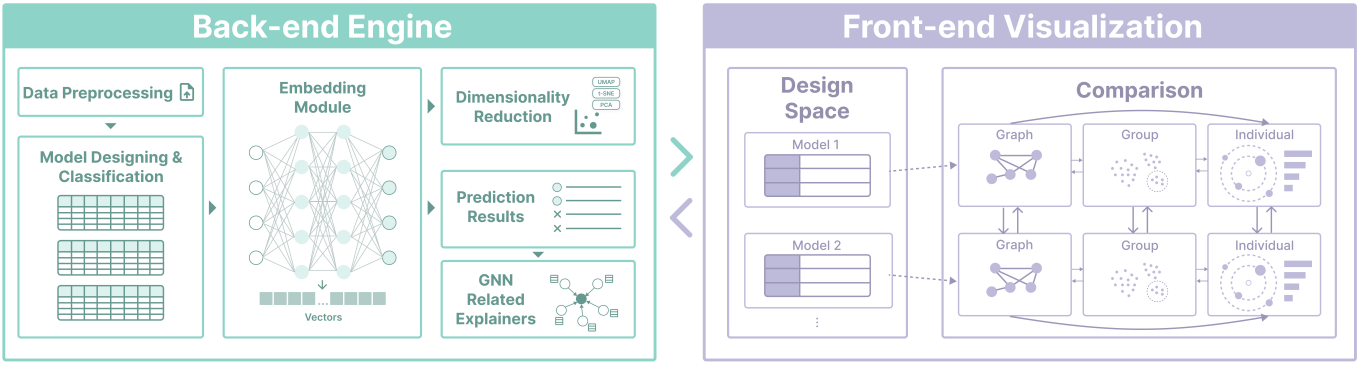


Fig. 2: The overview of *VAC-HGNN* comprises a two-fold structure, encompassing a back-end engine and a front-end visualization interface. After preprocessing the dataset, a set of HGNN models is identified and categorized into groups to facilitate a summary of search directions. The visualization interface provides an overview of the design space, while three-level comparisons allow for coordinated interactions between graph-level, group-level, and individual-level visualizations.

R3: Display latent information at the global and group level. In the latent space, clustering and relative positions can be exhibited by using dimensionality reduction methods. E3 expressed interest in the rules in the projection view and all domain experts were keen on discovering valuable information learned by the model beyond the "black box". Since it is challenging to determine the embedding characteristics solely based on quantitative metrics, details in the latent space should be presented, and then clustering groups can be tracked in series **[D2]**.

R4: Specify important nodes and links for model understanding. At the local/individual level, researchers should identify and examine significant nodes or links that may be responsible for the model performance, such as isolated nodes or those with incorrect results **[D2]**. As highlighted by domain experts E1 and E3, it is crucial to identify the key nodes in the model to understand the underlying mechanisms. Therefore, visual designs should incorporate such information, including nodes themselves and their related entities, to provide explanations for models' performances.

R5: Investigate the topological configuration at the individual level. At the individual level, nodes or links may have connections or neighbors that are considered topological features of the topology space. As HGs contain distinct types of edges and nodes, examining the topological configuration of a specific entity can provide experts with a comprehensive comprehension of the model process and enhance the interpretability **[D2]**.

R6: Facilitate interactive data exploration. The findings of our literature review [36] indicate that the incorporation of interactive design elements can significantly enhance the process of aligning models for comparative analysis. Additionally, our investigation revealed that domain experts often rely solely on materials provided by authoritative sources such as Readme, which may lack sufficient self-exploration and intuitive features. Consequently, to cater to the diverse needs of users, the system should support flexible and highly interactive data analysis capabilities **[D1, D2]**.

4 Approach Overview

The visual analytics system, *VAC-HGNN*, aims to assist developers in comparing various graph neural network (GNN) models in a heterogeneous design space. The system is composed of a back-end engine and a front-end visualization interface, as illustrated in Fig. 2. Initially, the dataset undergoes preprocessing for training and testing purposes. Then, a predetermined number of GNN models are identified within the design space and categorized into groups to facilitate a summary of search directions. The embeddings generated by these previously designed models are computed for downstream tasks. GNN-related explainers, such as *GNNExplainer*, may be employed to generate explanations for the prediction results. Furthermore, dimensionality reduction techniques are applied to embeddings for further exploration. The outputs generated by the back-end engine are subsequently fed into the visualization module for GNN model analysis. This interface supports a two-stage iterative workflow aimed at enhancing user comprehension of the design space and the models it encompasses. The design space visualization view provides an overview of the models' performances and their associated architecture parameters in the design space **[D1]**. The three-level comparisons encompass graph-level visualization, group-level visualization, and individual-level visualization **[D2]**, which can be coordinated using rich interactions.

5 Back-end Engine

5.1 The NAS-HGNN Dataset

Based on a literature review of NAS for GNNs [12, 70, 72] and feedback from experts in the observational study (Sec. 3), it has been identified that there are two ways for HGNN practitioners to select HGNN models. The first approach involves starting with one of the State-of-the-Art (SOTA) models and tuning the architecture configurations (ACs) or hyperparameters (HPs) to improve performance. The second approach involves using automatic graph NAS methods. To cater to both of these scenarios, we have constructed an architecture dataset for HGNN NAS on the link prediction task, which we refer to as *NAS-HGNNl*. To build this dataset, we follow the procedure of other NASBench approaches [9, 58]. Specifically, we introduce the design space for HGNNs, discuss implementation details, and provide the statistics of the dataset.

Design Space for HGNN. In order to obtain a comprehensive representation of HGNNs, various design spaces were compared in the literature [12, 60, 61, 70, 72]. After a thorough evaluation, *Space4HGNN* was ultimately selected as the platform for dataset construction. This choice was made because it not only considers general designs but also proposes a unified framework for specific designs of HGNN. *Space4HGNN* categorizes the design aspects of HGNNs into four categories: *intra-layer*, *inter-layer*, *training settings* and *unique design for HGNN*. In total, 15 design dimensions are considered, as shown in **Appendix B**.

Implementation Details. The models were trained on three datasets, namely *HGBl-amazon* (with 147k training links and 15k testing links), *HGBl-LastFM* (with 142k training links and 19k testing links), and *HGBl-PubMed* (with 190k training links and 9k testing links) using the *OpenHGNN* framework [16]. The training process was conducted on a single NVIDIA Tesla K80, with three repetitions of the training and evaluation of all designs to mitigate the effect of randomness. To improve training efficiency, early stopping was employed. Consequently, a mapping was established from the (AC, HP, id) space to (validation accuracy, training time, and best epoch) for each dataset. Accuracy was measured using *ROC-AUC*. Two training strategies were employed, namely *random search* and *evolutionary learning*. The pseudocode for the evolutionary strategy is presented in **Appendix A** where *Gen* denotes the number of generations for evolution and *N* refers to the population size. Empirically, the fitness measurement was set as *ROC_AUC*, while the crossover and mutation rates were set as 0.5 and 0.1, respectively.

Dataset Statistics. The fundamental characteristics of the dataset are presented in Tab. 1. Figure 3 examines the association between training time and validation accuracy of HGNN models included in the dataset.

The figure suggests that the most computationally intensive design is not necessarily the best-performing one. These results underscore the importance of thoughtful design considerations, rather than relying solely on the addition of computational resources.

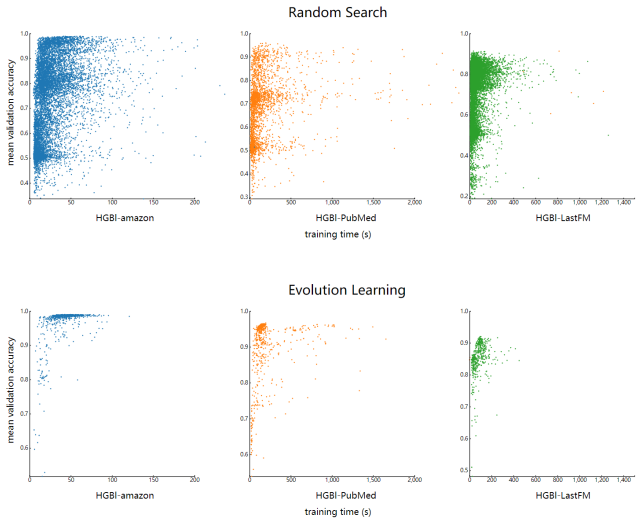


Fig. 3: The relationship between accuracy on validation data and the duration of the training process.

Dataset	Algorithm	#Designs	Wall-Clock Time(seconds)
HGBl-LastFM	Random Search	12,783	735,847.0
	Evolution Learning	474	41,860.0
HGBl-PubMed	Random Search	2,651	460,077.8
	Evolution Learning	650	143,741.6
HGBl-amazon	Random Search	10,010	231,659.7
	Evolution Learning	971	41,038.1

Table 1: Design Count and Computation Cost for Each Dataset on Two Search Algorithms. The evolution learning algorithms on all datasets have converged.

5.2 Design Space Partition

Given the expansive design space, determining the search direction of candidate models is crucial to streamline subsequent model comparisons. Clustering is a typical method for classifying or partitioning the NAS data in tabular format. However, the resulting clusters may lack clear interpretation and semantics, particularly when dealing with uniform data generated by random search. An alternative approach involves the use of decision trees, which allow for a more intuitive understanding of certain partitions. Unfortunately, our dataset lacks the required labels for driving the decision process. To address this limitation, we propose a nested unsupervised decision tree algorithm for more effective classification of NAS data.

To represent each design of HGNN, we create a 15-dimensional tensor where each dimension corresponds to a design dimension in *space4hgnn* [70]. This tensor is partitioned into four parts, namely *intra-layer*, *inter-layer*, *training settings*, and *unique design for HGNN*. Mathematically,

$$design_i = \left(a_1^i : a_4^i, b_1^i : b_4^i, c_1^i : c_4^i, d_1^i : d_3^i\right). \quad (1)$$

In order to partition the data, we first use clustering. Given the mixed data type (some dimensions are categorical such as aggregation function and others are numeric such as dropout), we use *K-Prototype* [23] to label each $design_i$ with a class ε_i. Similarly, we cluster each aspect solely to give each design four other classes, namely $\alpha_i, \beta_i, \gamma_i, \delta_i$. As a result, we transform the representation of $design_i$ into a five-dimensional tensor $(\alpha_i, \beta_i, \gamma_i, \delta_i | \varepsilon_i)$.

Based on this representation, we use *CART* [38], a decision tree algorithm, to form a hierarchical partition of the data. Here, ε_i is regarded as the target variable, and the other four dimensions are regarded as the attributes. Since the *CART* algorithm only supports numerical target variables, one-hot encoding is applied before classification. For each split node in the hierarchical partition, the partition is divided by a class which is given by the former clustering of a certain aspect. Based on the design dimensions in that aspect, we first use *K-Prototype* to label the designs in the partition, and then a small hierarchy is formed by the *CART* algorithm to provide a detailed partition focused on the aspect. In this way, we obtain a nested hierarchical partition of the data.

Hyper-parameters such as the number of leaf nodes in *CART* are tuned manually, and the number of clusters in *K-Prototype* is determined by silhouette score [41].

6 Front-end Visualization

Figure 1 depicts the *VAC-HGNN* system, which is composed of two primary views - the *Design Space View* (Figure 1(A)) and the *Comparison View* (Figure 1(B). The latter includes a *Model List* (Figure 1(B1)), and *Model Exploration* (Figure 1(B2)) for facilitating a comprehensive comparison of multiple models. The *Ranking List* (Figure 1(A1)) presents models with favorable performance, and the corresponding design parameters are highlighted in the *Design Dimension Overview* (Figure 1(A2)) for convenient references. Subsequently, users can add multiple models to the *Model List* for comparison purposes (Figure 1(B1)). With the aid of *Model Exploration* (Figure 1(B2)), users can perform detailed comparisons at different levels and evaluate various design dimensions. By iteratively adding models with design dimension changes and comparing them, the optimal model can be identified.

6.1 Design Space View

This view provides a comprehensive display of the performance metrics of various models and their corresponding design dimension parameters, comprising two distinct components, namely the *Ranking List* and the *Design Dimension Overview*. The *Ranking List* displays a subset of models that exhibit optimal performances within the design space, which can be referred to and selected by users **[R1, R2, R6]**. On the other hand, the *Design Dimension Overview*, which displays the combination of design dimensions for the selected model and its associated search direction **[R1]**.

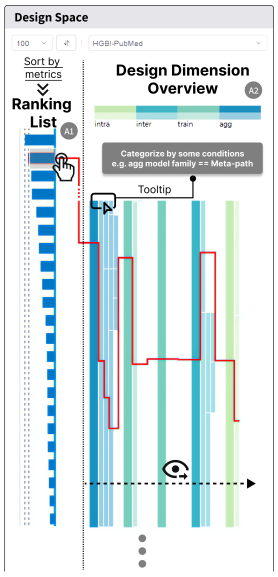


Fig. 4: *Design Space View* is composed of the *Ranking List* and the *Design Dimension Overview*, with a red line connecting to show a selected model's design dimensions.

Ranking List. As shown in Fig. 4(A1), this list is a representation of a subset of design space models. Each row in the list is associated with a selected metric value **[R2]**, which can be sorted using a button to aid users in identifying and selecting models of interest or those with favorable performance metrics **[R1]**. Further analysis and exploration of the selected models' design dimension combinations is possible. The bar chart presented in the list serves as a visual aid in identifying potential candidates for examination and exploring their design space characteristics.

Design Dimension Overview (Figure 4A2). The objective of this component is to aid in the identification of appropriate design dimen-

sion alternatives and search directions within a heterogeneous design space **[R1]**. To achieve this objective, a nested icicle plot has been developed, which draws inspiration from the icicle plot [28]. The plot serves to illustrate the partitioning of the entire design space. We have adopted the design dimensions employed in *Space4HGNN* [70], which encompasses **inter-layer design**, **intra-layer design**, **training settings design**, and **unique design dimensions in HGNN**, as shown in the **Appendix B**. These dimensions are represented by blocks with dark colors in the nested icicle plot. Each category encompasses specific design dimensions, which are organized as a decision tree once a category has been selected. For instance, the intra-layer design category includes *layer connectivity*, *pre-process layers*, *message passing layers*, and *post-process layers*. These dimensions are distinguished by a lighter color and are attached to their corresponding category block. Users can refer to the legend for specific color schemes. The size of each block is proportional to the number of models within that specific design space partition. By tracing the line from left to right, users can easily perceive the choices made for each dimension, which in turn helps identify appropriate search directions.

Design Alternatives. To achieve hierarchical classification objectives, two design alternatives are available. The commonly used approach is a tree structure that presents a detailed and clear hierarchy. However, when applied to a large design space, the tree structure may become too cumbersome to display effectively. Alternatively, the treemap can be utilized to conserve space. However, it has limitations in tracing the path of each design dimension, making it difficult to identify relationships between categories and compare search directions. To address these challenges, we have modified the icicle plot for our system, as previously described.

Interactions. The interactions available within the design space view can be delineated as follows: (1) **Selecting a metric needed.** Users may need to select a specific metric, depending on the downstream task at hand. For instance, in the case of link prediction, users have the ability to adjust the value of k for *top-k* accuracy and establish an order accordingly, thus enabling them to focus on the pertinent metric for their particular task. (2) **Selecting models.** Users can select models by brushing from the *Ranking List* to perform further analysis and comparison. The selected models are then added to the *Model List* in the *Comparison View*, allowing for side-by-side comparison of the models. This functionality facilitates the identification and exploration of models that exhibit favorable performance metrics or are relevant to the task at hand **[R6]**. (3) **Hovering.** Upon hovering the mouse over a particular design dimension, the decision conditions related to that dimension are displayed as a tooltip. This feature was implemented based on the feedback obtained from expert interviews.

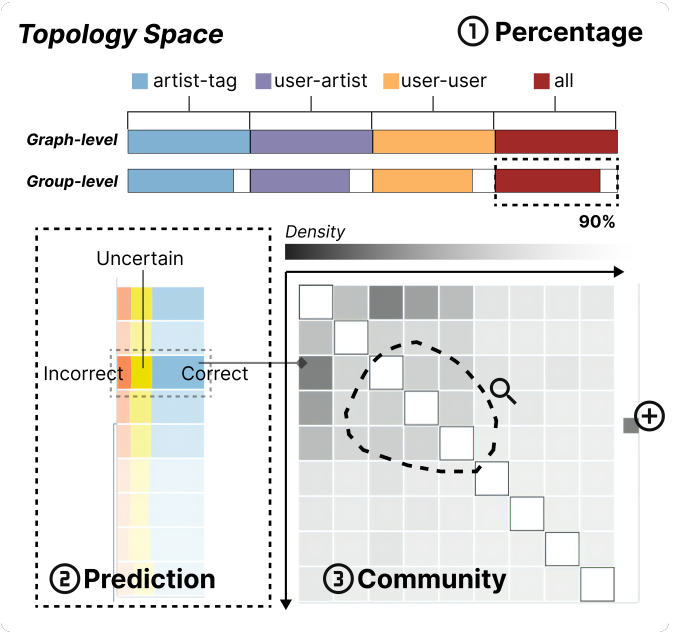
6.2 Comparison View

The primary component of the *Comparison View* is the *Model Exploration*, which allows for the comparison of selected models from the *Model List* at three distinct levels (i.e., the graph level, the group level, and the individual level) **[R2, R3, R4, R5, R6]**.

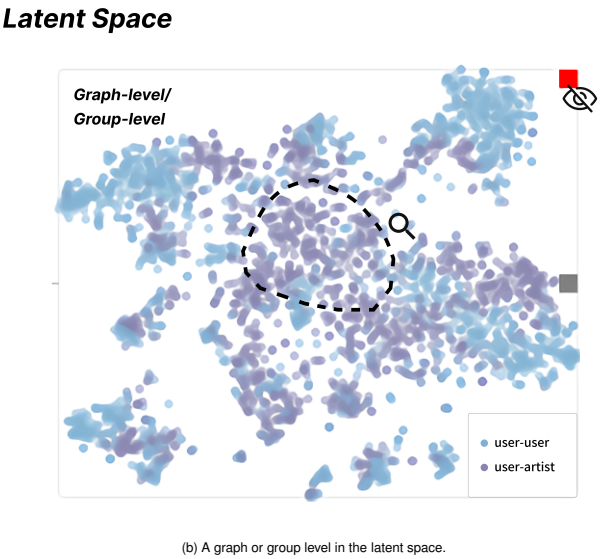
Model List (Figure 1B1). The *Model List* presents models that were selected in the *Design Space View*, along with their performances and specific combinations of design dimensions **[R2]**, thus serving as a bridge between the *Design Space View* and the *Comparison View*. To optimize screen usage and help users concentrate on the ongoing comparison among multiple models, a collapsible tree structure has been integrated into the *Model List*, with the visualizations of the *Model Exploration* (Figure 1B2) serving as its leaves.

Model Exploration (Figure 1B2). The process of model exploration entails analyzing performance metrics at the graph level in a comprehensive manner, while also examining details at both the group and individual levels with greater granularity.

Graph-level Visualization. Within the model tree, the graph-level visualization comprises two distinct views: a **graph topology view** (Figure 5a) and a **global latent space view**(Figure 5b), both of which are located in the first row. The graph topology view conveys comprehensive semantic and structural information pertaining to the entire graph **[R4]**,whereas the global latent space view is employed for the identification of patterns or clustered groups **[R3]**.



(a) A graph or group level in the topology space.



(b) A graph or group level in the latent space.

Fig. 5: A graph-level or group-level visualization in different spaces.

In order to enhance the clarity of the view and minimize overlaps, the Leiden algorithm [48] is applied to the graph topology view. This community detection algorithm is designed to identify densely connected communities within a network by employing modularity optimization and a multilevel approach to efficiently handle large and complex graphs. An adjacency matrix of communities is used in the graph topology view to display connections and detailed information about these communities. The rows and columns of the matrix correspond to different communities in the graph, with darker grids indicating a higher density of edges between the corresponding pair of communities. The region above the matrix is uniformly divided into blocks, with the final block representing the proportion of the selected area to the entire graph. The other blocks represent the proportion of different types of edges within the selected area compared to the complete set to which it belongs. For example, if half of the last block is colored, it indicates that the number of edges selected in the current view constitutes half of the total number of edges included in the graph. The area on the left side of the matrix presents a breakdown of the predicted results within the corresponding community, with the proportion depicted by a color-coded bar. The colors orange, yellow, and blue represent incor-

rect, uncertain and correct predictions, respectively, with the level of transparency indicating the number of edges contained.

Besides the topology space, an alternative perspective on the matter is offered by the latent space. After sampling, edges are mapped onto a two-dimensional plane utilizing a chosen layout technique, analogous to a scatterplot. The diversity of the links or predictions is denoted by a spectrum of colors, with the corresponding color codes displayed in the legend. It is noteworthy that the axes are not intended to convey any explicit meanings, as the fundamental purpose is to offer a simplified perspective of the cluster, facilitating the detection of underlying patterns and relationships.

Group-level Visualization. To conduct more in-depth analysis, users possess the capability to investigate distinct sections of the graph, be it a community or internal information within that community. This functionality enables the exploration of the group level at any level of granularity. For instance, users can opt to choose multiple communities at the graph level in the first row or gradually narrow down the scope from the top to the bottom, with a focus on a specific subset of the graph. The group level also comprises two perspectives similar to the graph level, namely the **topology space**, which is manifested through the adjacency matrix and the **latent space**, which is represented by a two-dimensional projection plot **[R3, R4]**.

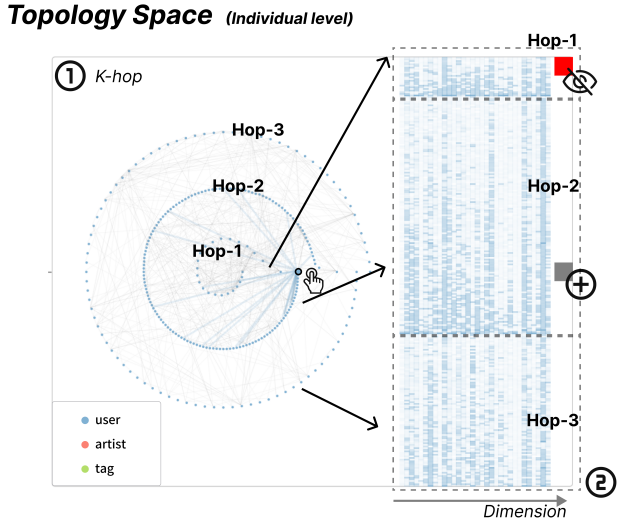


Fig. 6: An individual level in the topology space.

Individual-level Visualization (Figure 6). Individual-level visualization enables users to acquire a deeper understanding of the relative significance of distinct nodes and dimensions in the dataset. Figure 6 provides the **topology space** of the chosen individual, showcasing the influence or importance of its neighbors, which facilitates the interpretability of the results **[R5]**. To alleviate the burden of large-scale graphs, adding filtering conditions is a good choice. However, the nodes' importance levels are distributed in a bimodal manner, with two peaks representing high and relatively low levels of contribution. For instance, in the 2-hop distribution, the lower peak encompasses approximately 1,000 nodes, while the higher peak comprises nearly 100 nodes, as demonstrated in Fig. 7. Therefore, the utilization of a distribution that does not conform to a simple threshold range renders the filtering of points based on such a range ineffective and unsuitable. Thus, it becomes imperative to adopt specialized designs to address this issue. Drawing inspiration from the concept of atomic orbitals expressing energy levels, the current study presents a novel design for visualizing data. Specifically, the design employs an orbital map, where each node is represented as a circle, and the edges between the nodes are shown as lines connecting the circles. To display the k-hop graph, the neighboring nodes of a selected node pair (i.e. link) are illustrated as circles encircling the central link. For instance, the first orbital includes the immediate neighbors of the selected link, while the second orbital incorporates the neighbors of those neighbors. The relative distance of a circle from the center node is indicative of its level of importance. Moreover, the heatmap accompanying the orbital map provides the importance scores of each node in a multi-dimensional vector, with darker hues indicating higher importance. These two plots correspond to each other, where each row of the heatmap represents a node on the orbital, i.e. a node embedding vector.

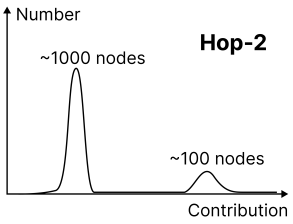


Fig. 7: The distribution example of the 2-hop contribution diagram.

Design Alternatives. For the orbital map at the individual level, a commonly used approach involves employing a node-link diagram with the force-directed collision-avoidance layout as displayed in Fig. 8 [25]. Nevertheless, this method may prove inadequate for displaying graphs that are particularly large or complex. This is primarily due to the fact that the density of the diagram can become overwhelming, thereby impeding the ability of users to form an accurate and holistic perception of the entire graph. In comparison, as previously described, a single orbital can accommodate a large number of entities and can also visually display the topology. This design provides an intuitive representation of the relative importance of other entities, the connections between these entities, or the differences between various types of entities.

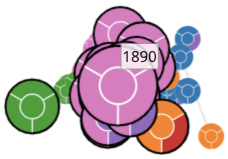


Fig. 8: The k-hop diagram of *GNNLens* [25].

Interactions. The *Comparison View* interactions can be delineated as follows: **1) Selecting certain parts to compare in *Model Exploration*.** The most critical function among all interactions is the ability to select specific parts of a graph for comparison in *Model Exploration*. The visualization operates in a top-down manner, progressing from the graph-to-group level. Users can select communities by brushing the adjacency matrix's grids at the global or group levels to zoom in and view details, with a corresponding branch of the tree growing below. This operation can be performed on any model. Furthermore, users can hide or minimize irrelevant views and focus on specific ones by clicking on the red button in the upper right corner, minimizing the distance between them. Conversely, clicking on the gray button on the right can expand the tree horizontally. Additionally, selecting an individual in the view extends the individual-level visualization horizontally, which can also be closed when no longer needed. **2) Highlighting the selected entities.** In *Model Exploration*, the selected entities are highlighted, allowing users to conveniently compare and contrast models' similarities and differences using the model juxtaposition. Additionally, when hovering over a point representing an edge at the individual level, the associated line is highlighted, with the color representing its type. **3) Selecting a layout.** Users can select different projection methods and encoding displayed to view the latent space based on their needs on the top of views. **4) Hiding or showing detailed parameters of models.** To address the challenge of displaying all information in the *Model List* where models often have multiple design dimensions, the collapse/expand button allows users to hide/show detailed information, with design dimensions displayed in the preview selectable by users. **5) Sliding the view's window.** Since comparing several models necessitates significant screen space, the *Comparison View*'s window can be slid vertically, allowing users to make comparisons freely. **6) Creating a model.** Once users have examined multiple models, they can modify the relevant design dimensions to develop novel models and incorporate them into the *Model List* for subsequent comparisons **[R6]**.

7 Evaluation

7.1 Case Study

In order to evaluate the efficacy of the proposed *VAC-HGNN* system, we carried out two case studies in collaboration with domain experts. These studies were designed to provide a comprehensive assessment of

the system's performance from two distinct viewpoints, namely, system consistency and design principle discovery.

7.1.1 Case One: System Result Consistency

Upon demonstrating the functionality of the system, E_1 exhibited keen interest in utilizing it to identify patterns and raised two inquiries. First, he inquired about the correlation between the model's performance and dimension reduction outcomes, as well as individual explanations. Second, he sought to ascertain the consistency of outcomes at the graph level and group level regarding dimensionality reduction and individual explanation.

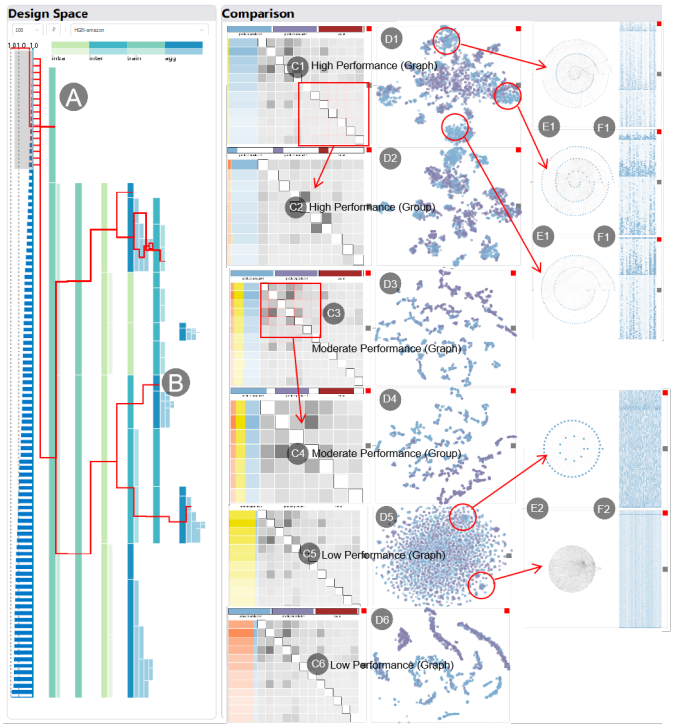


Fig. 9: The system results in consistency examination: (A) Top 100 model of HGBl-amazon (B) Another partition (C,D) Correlation between the model's performance and dimension reduction outcomes (D,E,F) Correlation dimension reduction outcome and individual explanation.

Dimension Reduction Pattern. To address the first question, E_1 selected the top 100 best models and sorted them in descending order by performance. After brushing several models with the highest performance, he found that all of the models fell into the first partition of the nested icicle plot (Figure 9A). He then expanded the area he brushed twice and found that some models fell into other partitions (Figure 9B). This piqued his interest, and he clicked on the first partition to add the models to the list. He sorted the models by accuracy and clicked the collapse to train them. He selected three models for different performance levels (Figure 9C1,C3,C5), and after they finished training, he inspected the graph level embedding (Figure 9D1,D3,D5) by clicking the right button. By comparing the embeddings of models with different performances, he concluded that high-performing models have dimensionality reduction results in clustered communities with significant edge overlap (Figure 9D1), while the dimensionality reduction results of average-performing models exhibit stripe patterns with less edge overlap (Figure 9D3). For poorly performing models, the points are evenly distributed in space, and the two types of edges overlap evenly (Figure 9D5). Based on these observations, he proposed an evolution process hypothesis of node embeddings during model training: "*from a uniform cluster to stripe patterns, and then to communities.*"

To validate his hypothesis, E_1 trained two models at each performance level and discovered that the dimensionality reduction results (Figure 9D6) of poorly performing models (Figure 9C6) had similar distributions and forms to those of models with average performance (Figure 9D3). This led him to believe that the relationship between the form of the model's dimensionality reduction results and the determinacy of the model's prediction results, rather than accuracy, is a more appropriate statement. Finally, he wondered whether the dimensionality reduction results at the group level are consistent with those at the graph level. He confirmed the consistency by selecting different groups and comparing the dimensionality reduction results of the same models (Figure 9D1,D2,D3,D4).

Pattern Explanation. E_1 conducted further analysis to understand the contribution of nodes to the prediction. Specifically, he clicked on links to identify nodes that contributed the most to the prediction. Upon comparing models with different performance levels, he observed that nodes at two hops from the link tended to account for a substantial proportion (Figure 9E1). This phenomenon can be attributed to the information diffusion process in GNNs. Nodes that are one hop away have already diffused their information to the graph, and their contribution to the prediction is relatively lower. In contrast, nodes that are two hops away may have an indirect but significant impact on the target nodes as their influence on the graph topology is more widespread.

Furthermore, E_1 identified patterns in the explanatory subgraph for individual explanations. He found that high-performing models capture more structure information, resulting in denser nodes and edges in the explanatory subgraph, with individuals in different communities sharing similar patterns (Figure 9E1). However, poorly performing models lacked consistency in the explanatory subgraph for different individuals in a single link (Figure 9E2). Moreover, high-performing models were observed to capture more local structural information and exhibited stable behavior. Additionally, the adjacent heatmap of high-performing models exhibited texture features (Figure 9F1), indicating the importance of certain dimensions, while the heatmap of poorly performing models was uniform (Figure 9F2).

7.1.2 Case Two: Summarizing the Design Principles of HGNNs

During the operation of the system by E_1, E_2 was present and able to gain an understanding of the system's capability. Subsequently, E_2 selected the top 100 models to guide the analysis, with a preference for the partition guided by the best model. In contrast to E_1, E_2 was primarily focused on the design of the model, which was based on a trust in the model's efficacy. To this end, E_2 habitually sorted the model by accuracy, cleared all default design dimensions, and selected those dimensions of personal interest, such as *dropout*, *activation function*, and L2 normalization (Figure 1B1). Upon examining the commonalities in the design dimensions of high-performing models, E_2 observed the absence of *L2 normalization* as a prominent feature (Figure 1B1), despite its reputation as a useful tool for optimization in deep learning, as it can enhance the stability and convergence of the optimization algorithm. In an attempt to comprehend this phenomenon, E_2 employed a method of controlling variables, by creating a model (Figure 1B) that shared all designs with the best model except for the utilization of *L2 normalization*.

Upon completing the model training, it was discovered that the accuracy of the model decreased by 4.8%, resulting in a decrease to 94.1% (Figure 1b). Additionally, the color yellow was found to occupy a larger proportion in the prediction distribution (Figure 1b), indicating an increase in model uncertainty. Based on this observation, a hypothesis was formed that *L2 normalization can increase model uncertainty*. In order to investigate this hypothesis, dimensionality reduction was performed on the embeddings of the two models, which revealed minimal differences. However, upon analyzing the embeddings of the edges located at the center of the communities, significant discrepancies were found (Figure 12). The model that did not use *L2 normalization* had a higher density of nodes and edges in the explanatory subgraph, while the model that did use *L2 normalization* had significantly fewer contributing nodes (Figure 12). Further examination of the distribution of the contributing nodes – the bimodal distribution (Figure 7), led to the realization that *L2 normalization* rescales the representation of the node, and subsequently, the range of link score is narrowed down. When the sigmoid acted as the score function, the score naturally centralized

to 0.5, resulting in an increase in prediction uncertainty. The use of L2 normalization can make node features more comparable in terms of their magnitudes, but when PyG[1] is used to identify the nodes that contribute the most to the prediction, nodes with small initial features may not be recognized as influential. Therefore, these nodes may act as noise in the explanation. It can be concluded that *L2 normalization* is a method that can improve interpretability, but it comes at the cost of decreased accuracy.

In accordance with the aforementioned methodology, which involved adhering to the design space view's guidance, analyzing design dimensions patterns, comparing models at varying levels and spaces, and providing explanations and summaries of the phenomenon, we were able to derive additional design principles: 1) *Max* is characterized by a low level of expressiveness; 2) *Ginconv max* is ill-suited for use in HGNNs due to its initial design catering towards GNNs, as outlined in [57]. The subsequent design principles can be readily discerned from the model list via the utilization of the ranking function: 1) Adam optimization typically outperforms Stochastic Gradient Descent (SGD); 2) The most frequently utilized learning rate among the top-performing models is 0.01; 3) Increased training epochs often result in improved performance.

7.2 Expert Interview

After conducting two case studies, we conducted interviews with four individuals, including three experts (E1, E2, and E3) and a female Ph.D. student (E4) with five years of experience in machine learning. Each interview was approximately 30 minutes long and preceded by a 20-minute introduction to the system.

System Usability. In general, the experts expressed a positive opinion regarding the usability of the system and its ability to aid in the comprehension and comparison of various HGNN models. The overall workflow was deemed to be reasonable, and the operations were deemed to be smooth. E3, in particular, noted that "*the connection between the two stages is very tight,*" while E1 appreciated the user-friendly nature of the system, which began with common quantitative metrics for tasks. The experts unanimously agreed that the various interactions provided by the system were highly satisfactory, and the ability to conceal irrelevant views allowed for focused attention on specific components. Moreover, the system was found to enhance the efficiency of model comparison. As E2 stated, "*previously, I might need to compare models one by one, but now I can select multiple models in the design space and add them to the list for subsequent comparison.*" Additionally, the experts all concurred that the presentation of views across three different levels for various spaces facilitated a comprehensive comparison of models.

Visual Encodings. Experts found the visual designs in the system easy to understand due to their intuitive nature and appropriate use of commonly used graphics. The projection view was useful for cluster exploration, while the nested icicle plot facilitated the identification of suitable design dimension alternatives and corresponding search directions. The experts agreed that the use of orbital maps for visualizing GNN interpretability algorithms was visually appealing and user-friendly. After the introduction and example, experts demonstrated a comprehensive understanding of the visual encodings in the system.

Suggestions. Experts provided feedback on improving the optimization of *VAC-HGNN* in our visualization system. They suggested adding more auxiliary information for model comparison and search direction identification. To address this, we added tooltips to display search directions for design dimensions. However, further research is needed to improve the *Comparison View*. The experts also suggested extending the system's capabilities to support more downstream tasks for greater extensibility.

8 Discussion and Limitation

Contributions Over the Previous Work. This study aims to compare and analyze various HGNN models in the design space using visualization techniques. We provide explanations for "black-box" models during the comparison process, which distinguishes our work from previous interpretability efforts that focus on explaining individual models. The system compares relevant models to provide users with search directions, unlike *ArchExplorer* [64], which requires users to compare models outside the system after exploring patterns. Furthermore, the proposed approach does not require users to switch between the system and external tools. Additionally, the approach differs from *GNNLens* [25], which is primarily designed for diagnostic tasks of a single model. Drawing on relevant comparative layout theory, the authors explore views for comparing HGNN models to advance further research on comparison views in the visualization community.

Generalizability and Scalability. This work identifies two potential avenues for extending the functionality of *VAC-HGNN*. First, while the system is currently designed for link prediction, it could be adapted for various applications such as classification and recommendation. Second, the system could be modified to cater to non-experts, such as biologists, through the inclusion of data storytelling. However, scalability is a significant obstacle due to the large volume of data encompassing models, parameters, and structural information. To address scalability concerns, we adopt a "from global to local" approach by dividing a large-scale graph into distinct communities and encoding them visually through an adjacency matrix. This approach allows users to selectively examine specific sections of the data to gain insights, alleviating scalability concerns to a certain extent. Nevertheless, GPU processing may be required to display the entire large-scale node-link diagram, which could necessitate substantial computing resources and may not always be feasible.

Learning Curve. Domain experts have found the *VAC-HGNN* system to provide valuable information, albeit requiring a period of adaptation to the system's glyphs. To improve user experience, interactive controls could be integrated, allowing users to selectively display information. For example, users could hide edge connections between neighboring nodes at the individual level based on their preferences. A step-by-step user guide could also be created to assist users in mastering the system.

Limitations. The *VAC-HGNN* system has some limitations despite its effectiveness. The deep learning-based community detection algorithm lacks interpretability, and the clustering and decision tree methods used in constructing the design space may not be very stable. Therefore, exploring more appropriate algorithms in the future is necessary. Predefined design spaces may also limit the scope of HGNN models that can be created, so allowing users to define their own design space could be a potential solution. Additionally, domain experts have expressed interest in uploading their own GNN models for further exploration.

9 Conclusion and Future Work

This work presents a new visual analytics system called *VAC-HGNN*, which is designed to assist with the comparison of HGNN models and identifying search directions in the design space. To represent the design space, the authors utilized a nested icicle plot with the Decision Tree algorithm, allowing for efficient navigation and model selection. A collapsible tree structure was also implemented, which enables a detailed comparison of the selected HGNNs at three different levels (graph, group, and individual). The system incorporates GNN interpretation algorithms, which provide clear explanations of the compared models. Two case studies were conducted, and experts were interviewed to validate the effectiveness of the system in aiding users to identify search directions and compare HGNNs. Results demonstrate that the proposed system is effective in achieving these objectives. Future work will focus on addressing the limitations of the algorithms and exploring the potential to support non-specialists in this field.

References

[1] C. Agarwal, M. Zitnik, and H. Lakkaraju. Probing gnn explainers: A rigorous theoretical and empirical analysis of gnn explanation methods. In G. Camps-Valls, F. J. R. Ruiz, and I. Valera, eds., *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, vol. 151 of *Proceedings of Machine Learning Research*, pp. 8969–8996. PMLR, 28–30 Mar 2022. doi: 10.48550/arXiv.2106.09078 3

[1] A library built upon PyTorch to easily write and train Graph Neural Networks for a wide range of applications related to structured data.

[2] D. L. Arendt, N. Nur, Z. Huang, G. Fair, and W. Dou. Parallel embeddings: a visualization technique for contrasting learned representations. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, pp. 259–274, 2020. doi: 10.1145/3377325.3377514 3

[3] F. Baldassarre and H. Azizpour. Explainability techniques for graph convolutional networks. *arXiv preprint arXiv:1905.13686*, 2019. doi: 10.48550/arXiv.1905.13686 3

[4] A. Boggust, B. Carter, and A. Satyanarayan. Embedding comparator: Visualizing differences in global structure and local neighborhoods via small multiples. In *27th international conference on intelligent user interfaces*, pp. 746–766, 2022. doi: 10.1145/3490099.3511122 3

[5] S. Cai, L. Li, J. Deng, B. Zhang, Z.-J. Zha, L. Su, and Q. Huang. Rethinking graph neural architecture search from message-passing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6657–6666, 2021. doi: 10.48550/arXiv.2103.14282 2

[6] Y. Cao, H. Peng, and P. S. Yu. Multi-information source hin for medical concept embedding. In *Advances in Knowledge Discovery and Data Mining: 24th Pacific-Asia Conference, PAKDD 2020, Singapore, May 11–14, 2020, Proceedings, Part II 24*, pp. 396–408. Springer, 2020. doi: 10.48550/arXiv.2103.14282 2

[7] Y. Chen, Q. Zhang, Z. Guan, Y. Zhao, and W. Chen. Gemvis: A visual analysis method for the comparison and refinement of graph embedding models. *The Visual Computer*, 38(9-10):3449–3462, 2022. doi: 10.1007/s00371-022-02548-5 2, 3

[8] R. Cutura, M. Aupetit, J.-D. Fekete, and M. Sedlmair. Comparing and exploring high-dimensional data with dimensionality reduction algorithms and matrix visualizations. In *Proceedings of the International Conference on Advanced Visual Interfaces*, pp. 1–9, 2020. doi: 10.1145/3399715.3399875 3

[9] X. Dong and Y. Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. *arXiv preprint arXiv:2001.00326*, 2020. doi: 10.48550/arXiv.2001.00326 4

[10] S. Fan, J. Zhu, X. Han, C. Shi, L. Hu, B. Ma, and Y. Li. Metapath-guided heterogeneous graph neural network for intent recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2478–2486, 2019. doi: 10.1145/3292500.3330673 2

[11] X. Fu, J. Zhang, Z. Meng, and I. King. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *Proceedings of The Web Conference 2020*, pp. 2331–2341, 2020. doi: 10.1145/3366423.3380297 1, 2

[12] Y. Gao, H. Yang, P. Zhang, C. Zhou, and Y. Hu. Graphnas: Graph neural architecture search with reinforcement learning. *arXiv preprint arXiv:1904.09981*, 2019. doi: 10.48550/arXiv.1904.09981 2, 3, 4

[13] Y. Gao, P. Zhang, Z. Li, C. Zhou, Y. Liu, and Y. Hu. Heterogeneous graph neural architecture search. In *2021 IEEE International Conference on Data Mining (ICDM)*, pp. 1066–1071. IEEE, 2021. doi: 10.1109/ICDM51629.2021.00124 1, 2

[14] M. Gleicher, D. Albers, R. Walker, I. Jusufi, C. D. Hansen, and J. C. Roberts. Visual comparison for information visualization. *Information Visualization*, 10(4):289–309, 2011. doi: 10.1177/1473871611416549 3

[15] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017. doi: 10.48550/arXiv.1706.02216 2

[16] H. Han, T. Zhao, C. Yang, H. Zhang, Y. Liu, X. Wang, and C. Shi. Openhgnn: An open source toolkit for heterogeneous graph neural network. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 3993–3997, 2022. doi: 10.1145/3511808.3557664 4

[17] F. Heimerl, C. Kralj, T. Möller, and M. Gleicher. embcomp: Visual interactive comparison of vector embeddings. *IEEE Transactions on Visualization and Computer Graphics*, 28(8):2953–2969, 2020. doi: 10.1109/TVCG.2020.3045918 3

[18] H. Hong, H. Guo, Y. Lin, X. Yang, Z. Li, and J. Ye. An attention-based graph neural network for heterogeneous structural learning. In *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, pp. 4132–4139, 2020. doi: 10.1609/aaai.v34i04.5833 2

[19] B. Hu, Z. Zhang, C. Shi, J. Zhou, X. Li, and Y. Qi. Cash-out user detection based on attributed heterogeneous information network with a hierarchical attention mechanism. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 946–953, 2019. doi: 10.1609/aaai.v33i01.3301946 2

[20] Z. Hu, Y. Dong, K. Wang, and Y. Sun. Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020*, pp. 2704–2710, 2020. doi: 10.1145/3366423.3380027 2

[21] Q. Huang, M. Yamada, Y. Tian, D. Singh, and Y. Chang. Graphlime: Local interpretable model explanations for graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 2022. doi: 10.1109/TKDE.2022.3187455 3

[22] Y. Huang, J. Zhang, Y. Yang, Z. Gong, and Z. Hao. Gnnvis: Visualize large-scale data by learning a graph neural network representation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 545–554, 2020. doi: 10.1145/3340531.3411987 2, 3

[23] Z. Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data mining and knowledge discovery*, 2(3):283–304, 1998. 5

[24] Z. Huang, X. Li, and H. Chen. Link prediction approach to collaborative filtering. In *Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*, pp. 141–142, 2005. doi: 10.1145/1065385.1065415 1

[25] Z. Jin, Y. Wang, Q. Wang, Y. Ming, T. Ma, and H. Qu. Gnnlens: A visual analytics approach for prediction error diagnosis of graph neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 2022. doi: 10.1109/TVCG.2022.3148107 2, 3, 7, 9

[26] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. doi: 10.48550/arXiv.1609.02907 2

[27] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009. doi: 10.1109/MC.2009.263 1

[28] J. B. Kruskal and J. M. Landwehr. Icicle plots: Better displays for hierarchical clustering. *The American Statistician*, 37(2):162–168, 1983. doi: 10.2307/2685881 6

[29] Q. Li, K. S. Njotoprawiro, H. Haleem, Q. Chen, C. Yi, and X. Ma. Embeddingvis: A visual analytics approach to comparative network embedding inspection. In *2018 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 48–59. IEEE, 2018. doi: 10.1109/VAST.2018.8802454 3

[30] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proceedings of the twelfth international conference on Information and knowledge management*, pp. 556–559, 2003. doi: 10.1145/956863.956972 1

[31] H. Linmei, T. Yang, C. Shi, H. Ji, and X. Li. Heterogeneous graph attention networks for semi-supervised short text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4821–4830, 2019. doi: 10.18653/v1/D19-1488 2

[32] J. Liu, C. Shi, C. Yang, Z. Lu, and S. Y. Philip. A survey on heterogeneous information network based recommender systems: Concepts, methods, applications and resources. *AI Open*, 2022. doi: 10.1016/j.aiopen.2022.03.002 2

[33] Y. Liu, E. Jun, Q. Li, and J. Heer. Latent space cartography: Visual analysis of vector space embeddings. In *Computer graphics forum*, vol. 38, pp. 67–78. Wiley Online Library, 2019. doi: 10.1111/cgf.13672 3

[34] Z. Liu, Y. Wang, J. Bernard, and T. Munzner. Visualizing graph neural networks with corgie: Corresponding a graph to its embedding. *IEEE Transactions on Visualization and Computer Graphics*, 28(6):2500–2516, 2022. doi: 10.1109/TVCG.2022.3148197 2, 3

[35] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, and X. Zhang. Parameterized explainer for graph neural network. *Advances in neural information processing systems*, 33:19620–19631, 2020. doi: 10.48550/arXiv.2011.04573 3

[36] S. LYi, J. Jo, and J. Seo. Comparative layouts revisited: Design space, guidelines, and future directions. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1525–1535, 2020. doi: 10.1109/TVCG.2020.3030419 3, 4

[37] B. M. Oloulade, J. Gao, J. Chen, T. Lyu, and R. Al-Sabri. Graph neural architecture search: A survey. *Tsinghua Science and Technology*, 27(4):692–708, 2021. doi: 10.26599/TST.2021.9010057 1, 2

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 5

[39] P. E. Pope, S. Kolouri, M. Rostami, C. E. Martin, and H. Hoffmann.

Explainability methods for graph convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10772–10781, 2019. doi: 10.1109/CVPR.2019.01103 3

[40] A. Rissaki, B. Scarone, D. Liu, A. Pandey, B. Klein, T. Eliassi-Rad, and M. A. Borkin. Biascope: Visual unfairness diagnosis for graph embeddings. In *2022 IEEE Visualization in Data Science (VDS)*, pp. 27–36. IEEE, 2022. doi: 10.1109/VDS57266.2022.00008 2, 3

[41] P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987. 5

[42] B. Sanchez-Lengeling, J. Wei, B. Lee, E. Reif, P. Wang, W. Qian, K. McCloskey, L. Colwell, and A. Wiltschko. Evaluating attribution for graph neural networks. *Advances in neural information processing systems*, 33:5898–5910, 2020. doi: 10.5555/3495724.3496219 3

[43] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*, pp. 593–607. Springer, 2018. doi: 10.48550/arXiv.1703.06103 2

[44] T. Schnake, O. Eberle, J. Lederer, S. Nakajima, K. T. Schütt, K.-R. Müller, and G. Montavon. Higher-order explanations of graph neural networks via relevant walks. *IEEE transactions on pattern analysis and machine intelligence*, 44(11):7581–7596, 2021. doi: 10.1109/TPAMI.2021.3115452 3

[45] C. Shi, Y. Li, J. Zhang, Y. Sun, and S. Y. Philip. A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering*, 29(1):17–37, 2016. doi: 10.1109/TKDE.2016.2598561 1

[46] V. Sivaraman, Y. Wu, and A. Perer. Emblaze: Illuminating machine learning representations through interactive comparison of embedding spaces. In *27th International Conference on Intelligent User Interfaces*, pp. 418–432, 2022. doi: 10.1145/3490099.3511137 3

[47] J. Stahnke, M. Dörk, B. Müller, and A. Thom. Probing projections: Interaction techniques for interpreting arrangements and errors of dimensionality reductions. *IEEE transactions on visualization and computer graphics*, 22(1):629–638, 2015. doi: 10.1109/TVCG.2015.2467717 3

[48] V. A. Traag, L. Waltman, and N. J. Van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific reports*, 9(1):5233, 2019. doi: 10.1038/s41598-019-41695-z 6

[49] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017. doi: 10.48550/arXiv.1710.10903 2

[50] C. Wang, Z. Lin, X. Yang, J. Sun, M. Yue, and C. Shahabi. Hagen: Homophily-aware graph convolutional recurrent network for crime forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, pp. 4193–4200, 2022. doi: 10.1609/aaai.v36i4.20338 2

[51] Q. Wang, K. Huang, P. Chandak, M. Zitnik, and N. Gehlenborg. Extending the nested model for user-centric xai: A design study on gnn-based drug repurposing. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):1266–1276, 2022. doi: 10.1109/TVCG.2022.3209435 3

[52] S. Wang, F. Xu, Y. Li, J. Wang, K. Zhang, Y. Liu, M. Wu, and J. Zheng. Kg4sl: knowledge graph neural network for synthetic lethality prediction in human cancers. *Bioinformatics*, 37(Supplement_1):i418–i425, 2021. doi: 10.1093/bioinformatics/btab271 1

[53] X. Wang, D. Bo, C. Shi, S. Fan, Y. Ye, and S. Y. Philip. A survey on heterogeneous graph embedding: methods, techniques, applications and sources. *IEEE Transactions on Big Data*, 2022. doi: 10.1109/TBDATA.2022.3177455 1, 2

[54] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu. Heterogeneous graph attention network. In *The world wide web conference*, pp. 2022–2032, 2019. doi: 10.1145/3308558.3313562 1, 2

[55] Z. Wang, H. Zhao, and C. Shi. Profiling the design space for graph neural networks based collaborative filtering. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pp. 1109–1119, 2022. doi: 10.1145/3488560.3498520 2

[56] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020. doi: 10.1109/TNNLS.2020.2978386 1, 2

[57] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018. 9

[58] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *International Conference on Machine Learning*, pp. 7105–7114. PMLR, 2019. doi: 10.48550/arXiv.1902.09635 4

[59] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32, 2019. doi: 10.48550/arXiv.1903.03894 2, 3

[60] M. Yoon, T. Gervet, B. Hooi, and C. Faloutsos. Autonomous graph mining algorithm search with best speed/accuracy trade-off. In *2020 IEEE International Conference on Data Mining (ICDM)*, pp. 751–760. IEEE, 2020. doi: 10.1109/ICDM50108.2020.00084 4

[61] J. You, Z. Ying, and J. Leskovec. Design space for graph neural networks. *Advances in Neural Information Processing Systems*, 33:17009–17021, 2020. doi: 10.48550/arXiv.2011.08843 1, 2, 4

[62] H. Yuan, J. Tang, X. Hu, and S. Ji. Xgnn: Towards model-level explanations of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 430–438, 2020. doi: 10.1145/3394486.3403085 3

[63] H. Yuan, H. Yu, S. Gui, and S. Ji. Explainability in graph neural networks: A taxonomic survey. *arXiv preprint arXiv:2012.15445*, 2020. doi: 10.1109/TPAMI.2022.3204236 2, 3

[64] J. Yuan, M. Liu, F. Tian, and S. Liu. Visual analysis of neural architecture spaces for summarizing design principles. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):288–298, 2023. doi: 10.1109/TVCG.2022.3209404 2, 9

[65] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim. Graph transformer networks. *Advances in neural information processing systems*, 32, 2019. doi: 10.48550/arXiv.1911.06455 2

[66] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 793–803, 2019. doi: 10.1145/3292500.3330961 1

[67] Y. Zhang, D. Defazio, and A. Ramesh. Relex: A model-agnostic relational model explainer. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pp. 1042–1049, 2021. doi: 10.1145/3461702.3462562 3

[68] Y. Zhang, Y. Xiong, X. Kong, S. Li, J. Mi, and Y. Zhu. Deep collective classification in heterogeneous information networks. In *Proceedings of the 2018 World Wide Web Conference*, pp. 399–408, 2018. doi: 10.1145/3178876.3186106 2

[69] H. Zhao, L. Wei, and Q. Yao. Simplifying architecture search for graph neural network. *arXiv preprint arXiv:2008.11652*, 2020. doi: 10.48550/arXiv.2008.11652 2

[70] T. Zhao, C. Yang, Y. Li, Q. Gan, Z. Wang, F. Liang, H. Zhao, Y. Shao, X. Wang, and C. Shi. Space4hgnn: a novel, modularized and reproducible platform to evaluate heterogeneous graph neural network. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2776–2789, 2022. doi: 10.1145/3477495.3531720 1, 2, 3, 4, 5, 6

[71] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020. doi: 10.1016/j.aiopen.2021.01.001 1

[72] K. Zhou, Q. Song, X. Huang, and X. Hu. Auto-gnn: Neural architecture search of graph neural networks. *arXiv preprint arXiv:1909.03184*, 2019. doi: 10.48550/arXiv.1909.03184 2, 4

[73] J. Zhu, J. Hong, and J. G. Hughes. Using markov models for web site link prediction. In *Proceedings of the thirteenth ACM conference on Hypertext and hypermedia*, pp. 169–170, 2002. doi: 10.1145/513338.513381 1

[74] Z. Zhu, X. Fan, X. Chu, and J. Bi. Hgcn: A heterogeneous graph convolutional network-based deep learning model toward collective classification. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1161–1171, 2020. doi: 10.1145/3394486.3403169 2