IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 22 July 2024, accepted 4 September 2024, date of publication 11 September 2024, date of current version 23 September 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3458050

RESEARCH ARTICLE

Explainable Neural Subgraph Matching With Learnable Multi-Hop Attention

DUC Q. NGUYEN[1], THANH TOAN NGUYEN[2], JUN JO[3], FLORENT POUX[4], SHIKHA ANIRBAN[5], AND THO T. QUAN[1]

[1]Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology—VNU-HCM, Ho Chi Minh City 700000, Vietnam
[2]Faculty of Information Technology, HUTECH University, Ho Chi Minh City 700000, Vietnam
[3]School of Information and Communication Technology, Griffith University, Southport, QLD 4222, Australia
[4]Faculty of Sciences, Department of Geography, University of Liège, 4000 Liège, Belgium
[5]School of Information Technology, College of Science, Technology, Engineering and Mathematics, Murdoch University, Murdoch, WA 6150, Australia

Corresponding author: Thanh Toan Nguyen (nt.toan@hutech.edu.vn)

This work was supported by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under Grant IZVSZ2.203310. The work of Duc Q. Nguyen was supported by the master's and Ph.D. Scholarship Programme of Vingroup Innovation Foundation (VINIF) under Grant VINIF.2022.ThS.023.

ABSTRACT Subgraph matching is a challenging problem with a wide range of applications in drug discovery, social network analysis, biochemistry, and cognitive science. It involves determining whether a given query graph is present within a larger target graph. Traditional graph-matching algorithms provide precise results but face challenges in large graph instances due to the NP-complete nature of the problem, limiting their practical applicability. In contrast, recent neural network-based approximations offer more scalable solutions but often lack interpretable node correspondences. To address these limitations, this article presents a multi-task learning framework called xNeuSM: E**x**plainable **Neu**ral **S**ubgraph **M**atching, which introduces Graph Learnable Multi-hop Attention Networks (GLeMA) that adaptively learn the parameters governing the attention factor decay for each node across hops rather than relying on fixed hyperparameters. Our framework jointly optimizes both subgraph matching and finding subgraph-isomorphism mappings. We provide a theoretical analysis establishing error bounds for GLeMA's approximation of multi-hop attention as a function of the number of hops. Additionally, we prove that learning distinct attention decay factors for each node leads to a correct approximation of multi-hop attention. Empirical evaluation on real-world datasets shows that xNeuSM achieves substantial improvements in prediction F1 score of up to 34% compared to approximate baselines and, notably, at least a seven-fold faster query time than exact algorithms. With these results, xNeuSM can be applied to solve matching problems in various domains spanning from biochemistry to social science.

INDEX TERMS Explainability, graph neural networks, learnable multi-hop attention, subgraph matching.

I. INTRODUCTION

In recent decades, a significant focus has been developing practical solutions for NP-hard graph problems. This wave of interest has been motivated by the abundance of diverse graph data in the public domain [1]. One prominent challenge in this domain is tackling large graphs, and a vital aspect of this is addressing the issue of *subgraph matching*. Essentially, subgraph isomorphism or subgraph matching involves determining whether a given query graph is isomorphic to a subgraph within a target graph. Despite the inherent NP-completeness, this problem holds paramount significance, as it applies to various domains, including physics [2], social network analysis [3], [4], bioinformatics [5], [6], [7], graph retrieval [8], and computer vision [9]. This acceleration has driven researchers to devise scalable and efficient algorithms tailored to analyze extensive graphs like those found in social, biological networks and chemical molecules. Throughout the past decades, numerous works have been developed, yielding a spectrum of practical

The associate editor coordinating the review of this manuscript and approving it for publication was Giacomo Fiumara.

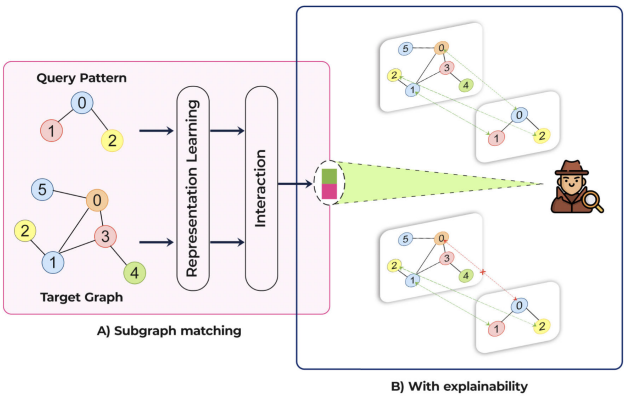


FIGURE 1. A typical subgraph matching result.

solutions encompassing various algorithms [10], [11], [12], [13], [14], [15], [16].

The conventional approaches [11], [17] rely on exact combinatorial algorithms. While exact computation of subgraph matching yields precise results in both *subgraph matching* (Figure 1-A) and *matching explanation* (finding node-to-node correspondences) among nodes across the two graphs (Figure 1-B), they confront with limited *scalability* when applied to larger query pattern sizes, which is attributed to the NP-complete nature of the problem. Recent efforts [12], [13], [14] aimed to enhance scalability by devising efficient matching orders and formulating powerful filtering strategies to shrink the number of candidates within the target graph. Although these efforts enable matching to be extended to larger target graphs, the query size remains restricted to just a few tens of nodes; this scalability level falls below what is needed for practical applications.

Subgraph matching is paramount due to its wide-ranging applications, but it poses a substantial challenge. In response to this challenge, neural-based approaches [15], [16], [18] have been proposed. These approaches aim to strike a balance between speed and accuracy. They demonstrate that by training a matching function to approximate the matching metric, it becomes feasible to identify candidate matches for a query pattern more rapidly than traditional combinatorial methods. The key innovation in these approaches lies in the use of graph neural networks (GNNs) to learn the matching function. However, these learning algorithms heavily rely on first-order dependencies within each layer of the GNN architecture, as highlighted in [15] and [16]. This implies that the receptive field of a single GNN layer is limited to one-hop network neighbors, which are immediate neighbors in the graph. Nevertheless, recent research has revealed that data obtained from various complex systems may exhibit dependencies that extend beyond the first order, going as far as fifth-order dependencies [19]. This contrasts with the earlier assumption of solely first-order network relationships. Oversimplifying the assumption of first-order network dependencies may lead to neglecting the generalizability of these methods in fully capturing patterns of varying sizes. Consequently, this oversimplification can result in a significant drop in performance across different domains [15], [16], [18].

Balancing efficiency (**O.1**) and explanation (**O.2**) in a subgraph matching algorithm are desirable goals. However, developing a learning-based approach that achieves both objectives poses a significant challenge (see Section IV-A for a detailed discussion). Inspired by the recent success of Graph Multi-hop Attention Networks (GMA) [20], we have adopted the concept of GMA to address these two objectives simultaneously. Unlike existing multi-hop attention mechanisms [21], [22], [23] in GMA that rely on a fixed attention decay factor for all nodes, this work introduces an innovative variant capable of adaptively learning this factor in a node-specific manner. This variant, termed Graph Learnable Multi-hop Attention Networks (GLeMa), parameterizes distinct attention decay factors for each node to govern its contributions across neighborhoods during multi-hop message passing. Furthermore, we provide a theoretical analysis establishing approximation error bounds for GLeMa's modelling of multi-hop attention as a function of the number of hops. Additionally, we formally prove that learning node-specific attention decays enables GLeMa to capture relational patterns in graph-structured data accurately.

By incorporating the learnable multi-hop attention mechanism, GLeMA achieves a commendable level of generalization within the data graph while maintaining efficiency (achieving **O.1**). However, simultaneously achieving **O.1** and **O.2** is still challenging. Unlike previous neural-based approaches [15] that directly learn from separate adjacency matrices for the pattern and target graphs, our newly devised unified proxy inputs facilitate the comprehensive capture of both intra- and inter-relations between the pattern and the target graph. This, in turn, enhances the explanation regarding explicit node alignment (achieving **O.2**). Additionally, we optimize the tasks of subgraph matching and matching explanation concurrently in an end-to-end multi-task manner. This approach leads to a mutually reinforcing synergy between both tasks, contributing to the overall effectiveness and efficiency of the framework. *To the best of our knowledge, none of the existing learning-based methods have been able to accomplish both of these objectives simultaneously.*

We named our approach xNeuSM, **E**xplainable **Neu**ral **S**ubgraph **M**atching with Graph Learnable Multi-hop Attention Networks. Our contributions are stated as follows:

- *Graph Learnable Multi-hop Attention Networks:* We introduce GLeMa, which possesses the ability to directly learn node-specific attention decay factors from the data. This adaptable mechanism mitigates biases that could emerge from fixed attention decay mechanisms applicable universally across nodes, thus averting potential suboptimal outcomes. Our approach guarantees that the model's decisions are data-driven, steering clear of potential influences stemming from suboptimal parameter selections.

- *Theoretical justifications for node-specific multi-hop attention mechanism:* In addition to the extensive empirical results, we conduct a theoretical analysis to approximate the error in multi-hop attention computation. Furthermore, we offer theoretical proof regarding the learning of attention decay factors specific to nodes. Our demonstration illustrates that the utilization of distinct decay factors remains consistent with the approximation of multi-hop attention and represents a generalized version of the previous universal attention decay factor utilized in earlier GMA models.
- *Multi-task learning:* We optimize both the subgraph matching and matching explanation tasks simultaneously within an end-to-end multi-task learning framework. This strategy fosters a mutually reinforcing synergy between these tasks, ultimately enhancing the overall effectiveness and performance of our framework.

The following sections outline the structure of this paper. Section II provides the study's foundational background. Section III discusses related works, situating our contributions within existing literature. Section IV introduces our framework and its components. In Section V, we propose a method to construct a joint representation for pattern and target graphs, facilitating intra- and inter-graph relationships. Section VI details GLeMA, designed for learning graph representation and inter-graph interactions. Section VII explains the aggregation of node embeddings to address subgraph matching and matching explanation tasks using a novel objective function. Section VIII provides theoretical justifications for approximating multi-hop attention with node-specific teleport probability. Section IX evaluates our approach using public datasets across various domains, showing a seven-fold increase in subgraph matching speed, a 27% improvement in accuracy, and a 34% increase in F1 score compared to NeuroMatch on the COX2 dataset, while maintaining comparability with exact methods. Finally, Section X summarizes the paper.

II. BACKGROUND

In this section, we establish precise notations necessary preliminaries and formally define our targeted problem involving subgraph matching and matching explanation.

A. PRELIMINARIES

In this study, we centre on solving subgraph matching and matching explanation problems on labelled, undirected, and connected graphs. Nevertheless, our proposed framework readily accommodates extensions to encompass directed graphs. The notations employed throughout this study are briefly outlined and summarized in Table 1.

We subsequently provide formal definitions for labelled, undirected, and connected graphs in Definition 1, and extend this to include directed graphs in Definition 2. The concepts of labelled subgraphs, including both non-induced and induced subgraphs, are defined in Definitions 3, 4, and 5, respectively. Definition 6 presents the concept of labelled graph isomorphism. Building on these definitions, we introduce the problems of non-induced and induced labelled subgraph isomorphism in Definitions 7 and 8.

TABLE 1. Summary of notation used.

Symbol	Definition
$\mathcal{D}$	Set of data graphs
$\mathcal{T}$	The target graph
$\mathcal{P}$	The query pattern
V	Set of nodes
E	Set of edges
Σ	Set of node labels
l	Node labelling function
$\boldsymbol{X}$	Initial node feature matrix
$\boldsymbol{X}^{\ell}$	Node embedding matrix at ℓ-layer
x_i^{ℓ}	Embedding of node v_i at ℓ-layer
$\boldsymbol{A}^{in}$	Intra-graph adjacency matrix
$\boldsymbol{A}^{cr}$	Cross-graph adjacency matrix
$\boldsymbol{I}$	Identity matrix
$\mathcal{A}^{(1)}$	1-hop attention matrix
$\mathcal{A}$	Attention diffusion matrix
$\mathcal{A}^{(K)}$	Approximate K-hop attention diffusion matrix
H	Number of attention heads
L_G	Number of GLeMa layers
L_{FC}	Number of fully-connected layers
$\boldsymbol{W}, b$	Learnable weights, biases
θ	Attention decay factor
δ	Non-linear activation function
σ	Sigmoid function
$y/\hat{y}$	Label/Prediction
$\mathcal{L}$	Loss function

This article focuses on induced labelled subgraph isomorphism, which is inherently more challenging to solve than the non-induced variant due to the limited number of polynomial-time solvable exceptional cases [24].

Definition 1 (Labelled Undirected Connected Graph): A labelled undirected connected graph is a graph represented with a 3-tuple $\mathcal{G} = (V, E, l)$ where

1) *V is a set of nodes,*
2) *$E \subseteq [V]^2$ is a set of edges (u, v), where $u, v \in V$*
3) *$\forall v \in V, deg(v) \geq 1$*
4) *$l : V \rightarrow \Sigma$ is a labelling function and Σ is the set of node labels.*

Definition 2 (Labelled Directed Connected Graph): A labelled directed connected graph is a graph represented with a 3-tuple $\mathcal{G} = (V, E, l)$ where

1) *V is a set of nodes,*
2) *$E \subseteq [V]^2$ is a set of edges (u, v), where u is tail node, v is head node and $u, v \in V$*
3) *$\forall v \in V, (deg_{in}(v) \geq 1) \vee (deg_{out}(v) \geq 1)$*
4) *$l : V \rightarrow \Sigma$ is a labelling function and Σ is a set of node labels*

Definition 3 (Labelled Subgraph): Let $\mathcal{G} = (V_{\mathcal{G}}, E_{\mathcal{G}}, l_{\mathcal{G}})$ and $\mathcal{S} = (V_{\mathcal{S}}, E_{\mathcal{S}}, l_{\mathcal{S}})$ be two labelled graphs. $\mathcal{S}$ is a subgraph of $\mathcal{G}$ (denoted as $\mathcal{S} \subseteq \mathcal{G}$) if and only if:

1) *$V_{\mathcal{S}} \subseteq V_{\mathcal{G}}$ and*
2) *$E_{\mathcal{S}} \subseteq E_{\mathcal{G}}$ and*
3) *$\forall v \in V_{\mathcal{S}}, l_{\mathcal{S}}(v) = l_{\mathcal{G}}(v)$.*

Definition 4 (Non-Induced Labelled Subgraph): Let $\mathcal{G} = (V_\mathcal{G}, E_\mathcal{G}, l_\mathcal{G})$ *and* $\mathcal{S} = (V_\mathcal{S}, E_\mathcal{S}, l_\mathcal{S})$ *be two labelled graphs.* $\mathcal{S}$ *is a non-induced subgraph of* $\mathcal{G}$ *(denoted as* $\mathcal{S} \subseteq_{ni} \mathcal{G}$*) if and only if*:

1) $\mathcal{S} \subseteq \mathcal{G}$ *and*
2) $\exists u, v \in V_\mathcal{S}, (u, v) \notin E_\mathcal{S} \wedge (u, v) \in E_\mathcal{G}$.

Definition 5 (Induced Labelled Subgraph): Let $\mathcal{G} = (V_\mathcal{G}, E_\mathcal{G}, l_\mathcal{G})$ *and* $\mathcal{S} = (V_\mathcal{S}, E_\mathcal{S}, l_\mathcal{S})$ *be two labelled graphs.* $\mathcal{S}$ *is an induced subgraph of* $\mathcal{G}$ *(denoted as* $\mathcal{S} \subseteq_{id} \mathcal{G}$*) if and only if*:

1) $\mathcal{S} \subseteq \mathcal{G}$ *and*
2) $\forall u, v \in V_\mathcal{S}, (u, v) \in E_\mathcal{S} \iff (u, v) \in E_\mathcal{G}$.

Definition 6 (Labelled Graph Isomorphism): Given two graphs $\mathcal{P} = (V_\mathcal{P}, E_\mathcal{P}, l_\mathcal{P})$ *and* $\mathcal{S} = (V_\mathcal{S}, E_\mathcal{S}, l_\mathcal{S})$*,* $\mathcal{P}$ *and* $\mathcal{S}$ *are considered isomorphism (denoted as* $\mathcal{P} \cong \mathcal{S}$*) if and only if there exists a bijection* $f : V_\mathcal{P} \to V_\mathcal{S}$ *such that*:

1) $\forall v \in V_\mathcal{P}, l_\mathcal{P}(v) = l_\mathcal{S}(f(v))$ *and*
2) $\forall u, v \in V_\mathcal{P}, (u, v) \in E_\mathcal{P} \iff (f(u), f(v)) \in E_\mathcal{S}$.

Definition 7 (Non-Induced Subgraph Isomorphism): Given two graphs $\mathcal{P} = (V_\mathcal{P}, E_\mathcal{P}, l_\mathcal{P})$ *and* $\mathcal{T} = (V_\mathcal{T}, E_\mathcal{T}, l_\mathcal{T})$*,* $\mathcal{P}$ *is considered as non-induced subgraph isomorphic to* $\mathcal{T}$ *if there exists* $\mathcal{S} = (V_\mathcal{S}, E_\mathcal{S}, l_\mathcal{S})$ *such that*:

1) $\mathcal{S} \subseteq_{ni} \mathcal{T}$ *and*
2) $\mathcal{P} \cong \mathcal{S}$.

Definition 8 (Induced Subgraph Isomorphism): Given two graphs $\mathcal{P} = (V_\mathcal{P}, E_\mathcal{P}, l_\mathcal{P})$ *and* $\mathcal{T} = (V_\mathcal{T}, E_\mathcal{T}, l_\mathcal{T})$*,* $\mathcal{P}$ *is considered as induced subgraph isomorphic to* $\mathcal{T}$ *if there exists* $\mathcal{S} = (V_\mathcal{S}, E_\mathcal{S}, l_\mathcal{S})$ *such that*:

1) $\mathcal{S} \subseteq_{id} \mathcal{T}$ *and*
2) $\mathcal{P} \cong \mathcal{S}$.

B. PROBLEM STATEMENT

Our study concentrates on resolving two pivotal problems: subgraph matching and matching explanation, formally delineated in Problem 1 and Problem 2, respectively.

Problem 1 (Subgraph Matching): The problem of Subgraph Matching involves determining whether a subgraph of a given target graph $\mathcal{T} = (V_\mathcal{T}, E_\mathcal{T}, l_\mathcal{T})$ *is isomorphic to a query pattern* $\mathcal{P} = (V_\mathcal{P}, E_\mathcal{P}, l_\mathcal{P})$*. The input consists of the target graph* $\mathcal{T}$ *and the query pattern* $\mathcal{P}$*, both of which are labeled and connected graphs. The output returns true if there exists a subgraph* $\mathcal{S} \subseteq_{id} \mathcal{T}$ *such that* $\mathcal{S}$ *is isomorphic to* $\mathcal{P}$*, and false if no such subgraph exists. The objective is to identify the presence of a subgraph within the target graph that is structurally identical to the query pattern.*

Problem 2 (Matching Explanation): The problem of Matching Explanation involves finding a precise correspondence between the nodes of a target graph $\mathcal{T} = (V_\mathcal{T}, E_\mathcal{T}, l_\mathcal{T})$ *and a query pattern* $\mathcal{P} = (V_\mathcal{P}, E_\mathcal{P}, l_\mathcal{P})$*. The input consists of the target graph* $\mathcal{T}$ *and the query pattern* $\mathcal{P}$*, both of which are labeled and connected graphs. The output is a one-to-one mapping* $f : V_\mathcal{P} \to V_\mathcal{T}$ *that defines the node correspondences. The required constraints ensure that both graphs are labeled connected graphs, and that there exists a subgraph* $\mathcal{S} \subseteq_{id} \mathcal{T}$ *isomorphic to* $\mathcal{P}$*. The objective is to determine the mapping* f *that accurately reflects the correspondences between the nodes of* $\mathcal{P}$ *and* $\mathcal{T}$.

III. RELATED WORKS

Subgraph matching is addressed through two distinct settings within research communities. This section covers approaches for *non-induced subgraph matching* (Section III-A) and *induced subgraph matching* (Section III-B). Furthermore, since we employ a neural-based approach capable of explaining subgraph matching, we also examine related techniques for *neural subgraph matching and explanation* (Section III-C).

A. NON-INDUCED SUBRAPH MATCHING

The concept of non-induced subgraph matching permits the pattern graph $\mathcal{P}$ to act as a partial embedding within a subgraph $\mathcal{S}$ of the larger target graph $\mathcal{T}$ (see Definition 7). Specifically, this means that while an edge $e \in E_\mathcal{T}$ may exist in the target graph $\mathcal{T}$, it is not required to have a mapping in $E_\mathcal{P}$. This particular setting holds significant utility in various aspects of data management, including tasks like graph indexing, graph similarity search, and graph retrieval [8]. Recently, there has been a surge of interest in the community surrounding the support for explainable subgraph matching, primarily driven by the introduction of efficient approaches [25]. These developments have garnered attention for their potential to enhance the interpretability and applicability of subgraph matching techniques. However, our current work primarily focuses on exploring induced subgraph matching, a topic that will be delved into further in the subsequent section. This approach involves identifying patterns where all the edges in the pattern must also exist in the data graph, providing a more rigorous condition for matching.

B. INDUCED SUBRAPH MATCHING

The induced subgraph matching problem has been proven NP-complete [26]. Various algorithms [10], [11], [12], [13], [14], [17] have been proposed to address this challenge, focusing on generating effective matching orders and designing robust filtering strategies to reduce the number of candidates in the data graph. For instance, QuickSI [10] introduces the *infrequent-edge-first ordering* technique. This approach arranges the edges of the query graph in ascending order based on their frequency of appearance in the data graph. In contrast, GraphQL [17] employs the *left-deep-join ordering* strategy, conceptualizing the enumeration procedure as a joint problem. TurboIso [11] and CFL [12] advocate for the path-based ordering method, which entails decomposing the query graph into several paths and ordering them according to the estimated number of embeddings for each path. In addition to these ordering strategies, state-of-the-art algorithms like TurboIso [11], CECI [13], and CFL [12] adopt a *tree-based framework*. This framework constructs a lightweight, tree-structured index to minimize the number of candidates. Subsequently, it enumerates all matches based on

this index rather than the original data graph. While these techniques have undeniably propelled significant progress, they face performance issues with large-scale graphs.

C. NEURAL SUBGRAPH MATCHING AND EXPLANATION

The initial work [27] attempted to assess the feasibility of Graph Neural Networks (GNNs) in subgraph matching, which was validated with small-scaled subgraphs. This underscored the potential superiority of GNNs over feedforward neural networks. With recent advancements in GNNs [2], [28], [29], [30], contemporary subgraph matching techniques [15], [16], [31], [32] have achieved state-of-the-art results in terms of efficiency. However, a need remains to explain the one-to-one correspondence between the graph pattern and the data graph, which hinders its direct application in downstream tasks like subgraph isomorphism testing. Recent studies have delved into the interpretability of GNNs [33], [34], [35], [36], [37] through a model-intrinsic perspective. They aim to explain which features of the data graph contribute to the GNN's performance on specific tasks, including subgraph matching [34]. However, it is worth noting that this line of inquiry is separate from our work, and we need to address this aspect of GNN interpretability in this article.

One closely related problem to our work is inexact matching. Despite employing an approximate neural-based approach, our validated outcomes are computed only when an exact pattern is matched. These allowable inexact approaches encompass a range of techniques, such as structural equivalence [38], inexact matching [39], and knowledge graph [40]. This article will not delve into this problem, as we aim to reserve its consideration for a future study when we extend our framework to address these issues.

IV. OVERVIEW APPROACH

A. DESIGN PRINCIPLES

xNeuSM benefits from a crucial advantage in terms of efficiency, thanks to the inherent characteristics of neural network computation. However, when it comes to effectiveness, it is essential to carefully design the graph neural network architecture to meet the following three properties, in addition to the common ones like approximate accuracy and efficiency:

- ***(R1) Explainability.*** Ideally, a subgraph matching framework should be capable of identifying the pattern's presence and providing approximate *alignment witnesses*. Given that no existing neural-based approaches offer these characteristics, we prioritize this feature as the utmost property due to its critical importance in numerous real-world applications.
- ***(R2) High-order dependency.*** Conventional network representations, which implicitly assume the Markov property (first-order dependency), can swiftly become constraining. The oversimplification inherent in first-order networks may disregard scalability, particularly pattern size. Recent studies have demonstrated that data from numerous complex systems may exhibit dependencies as high as fifth-order [19]. As we strive for a scalable solution in subgraph matching with explicability, including high-order dependency representation emerges as an essential design principle.
- ***(R3) Multi-task with configurability.*** The model should demonstrate adaptability to various matching metrics by fine-tuning its parameters through training. This is essential because, in certain scenarios, closely matched patterns [41]—those with a matching score surpassing a predetermined threshold—hold even greater significance than exact matches. Take, for instance, vaccine development, where a candidate closely matching the disease-to-be is far more critical than an exact match to the disease pattern. Such a closely matched candidate aids in early disease response. Hence, there may arise situations necessitating an emphasis on configuring the model to prioritize emergency scenarios that align better with human intuition [42].

B. THE CHALLENGES

To accomplish these objectives, we must address the ensuing the following challenges:

- ***Explainable adaptivity.*** The neural-based approach for the subgraph matching problem employs coarse-grained embeddings of entire graphs to approximate graph-level similarities (**R1**). Achieving an explicitly explainable alignment between nodes necessitates fine-grained annotations between the two graphs, adding a layer of supervision to the training process. However, preparing such training data is highly labor-intensive, resulting in computationally inefficient procedures.
- ***High computational complexity with Graph Multi-hop Attention.*** Addressing (**R2**) necessitates the effective integration of high-order dependencies, a task that is far from trivial. Elevating the orders of dependency can impose a computational burden on the model. As a result, many existing works frequently maintain the dependency fixed at the second order. Striking a balance between efficiency and capturing a large receptive field proves to be a challenging attempt.
- ***Multi-objective optimization.*** Previous studies [15], [27], [32], [43] have employed neural networks to characterize the similarity function. These networks operate on graph-level embeddings or sets of node embeddings. Despite their competitive performance in approximating similarity and facilitating retrieval tasks compared to traditional computations, integrating an extra optimization objective for analyzing node-to-node mappings between query-target graph pairs (**R3**) presents challenges. The development of a neural model capable of incorporating new objectives like the aforementioned one poses a significant challenge in architectural design.

C. THE APPROACH

We present a comprehensive overview of our xNeuSM framework in Figure 2 and its formal algorithmic flow in Algorithm 1, delineating three primary stages as follows. Our architecture design was inspired by a study of predicting interaction between drug and target protein [44].

Input representation: Departing from previous neural-based methods [15] that directly learn from separate adjacency matrices of pattern and target graphs, our newly devised unified proxy inputs enable the capture of cross-graph relations (**R1**). These inputs also bolster the learning process concurrently. Further elucidation on this aspect can be found in Section V.

Graph Learnable Multi-hop Attention Networks: Our novel approach employs a specialized graph neural network, GLeMA, to extract higher-order dependencies. GLeMA facilitates the effective representation of inter- and intra-interactions between pattern and target graphs by a learnable multi-hop attention mechanism, enabling simultaneous learning of those interactions. This integration of high-order dependencies ensures scalability (**R2**), particularly with larger patterns. For further details, refer to Section VI.

Multi-task optimization: In this third stage, we aggregate node embeddings while concurrently addressing two tasks: *subgraph prediction* and *matching explanation*. Both tasks utilize the features learned in the preceding stage. Additionally, we introduce a novel objective function aimed at optimizing both tasks simultaneously (**R3**). Section VII presents a detailed exploration of this stage.

Algorithm 1 xNeuSM Framework

Input : Node feature matrix $\boldsymbol{X}$
Intra-graph adjacency matrix $\boldsymbol{A}^{in}$
Cross-graph adjacency matrix $\boldsymbol{A}^{cr}$
Output: Prediction $\hat{y}$
Weighted mapping matrix $\boldsymbol{P}$

$\boldsymbol{X}^0 \leftarrow \boldsymbol{X}$
for l ***in*** *Range*$(1 \dots L_G)$ **do**
  $\widehat{\boldsymbol{X}}^l_{in} \leftarrow \text{GLeMa}_l(\boldsymbol{X}^{l-1}, \boldsymbol{A}^{in})$
  $\widehat{\boldsymbol{X}}^l_{cr} \leftarrow \text{GLeMa}_l(\boldsymbol{X}^{l-1}, \boldsymbol{A}^{cr})$
  $\boldsymbol{X}^l \leftarrow \widehat{\boldsymbol{X}}^l_{cr} - \widehat{\boldsymbol{X}}^l_{in}$
end
$(\mathcal{A}^{(1)})^{L_G} \leftarrow \text{ExtractAttnMat}(\text{GLeMa}_{L_G}, \boldsymbol{X}^{l-1}, \boldsymbol{A}^{cr})$
$x^0_{repr} \leftarrow \frac{1}{|V_\mathcal{P}|} \sum_{i \in V_\mathcal{P}} x_i^{L_G}$, where $x_i^{L_G} \subset \boldsymbol{X}^{L_G}$
for l ***in*** *Range*$(1 \dots L_{FC} - 1)$ **do**
  $x^l_{repr} \leftarrow \delta(\boldsymbol{W}_l x^{l-1}_{repr} + b_l)$
end
$\hat{y} \leftarrow \sigma(\boldsymbol{W}_y x^{L_{FC}-1}_{repr} + b_y)$
$\boldsymbol{P} \leftarrow \left\{ p_{ij} = \frac{1}{2}\left((a^{(1)}_{ij})^{L_G} + (a^{(1)}_{ji})^{L_G}\right) \right\}$, where
$i \in V_\mathcal{P}, j \in V_\mathcal{T}$, and $(a^{(1)}_{ij})^{L_G} \in (\mathcal{A}^{(1)})^{L_G}$

V. INPUT REPRESENTATION

Initially, the initialization of the model necessitates the preparation of input data. In the context of the subgraph isomorphism problem, the input comprises a subgraph (a pattern) and a larger graph (a target). Sets of nodes and edges conventionally characterize both patterns and targets. Toward our problem, we consider the pattern as $\mathcal{P} = \{V_\mathcal{P}, E_\mathcal{P}, l_\mathcal{P}\}$ and the target as $\mathcal{T} = \{V_\mathcal{T}, E_\mathcal{T}, l_\mathcal{T}\}$ where V, E are the sets of nodes and edges respectively; $l : V \rightarrow \Sigma$ is the labelling function. Subsequently, the formulation of input for our proposed model is undertaken, encompassing a collection of node feature vectors denoted as x, the primary adjacency matrix represented by $\boldsymbol{A}^{in}$, and the secondary adjacency matrix represented by $\boldsymbol{A}^{cr}$. Each node within the pattern or target is encoded as a one-hot vector of $2|\Sigma|$-dimensions, where $|\Sigma| = |\Sigma_\mathcal{P} \cup \Sigma_\mathcal{T}|$ stands for the maximum count of distinct node labels. The former $|\Sigma|$ dimensions are allocated for the pattern, while the remaining dimensions are designated for the target graph. This partitioning of pattern and target node features facilitates distinct embeddings for pattern and target nodes, thereby enhancing the quality of the mapping performance. Following this, the amalgamation of all node vectors culminates in forming the collective input set denoted as $\boldsymbol{X}$. The adjacency matrix $\boldsymbol{A}^{in}$ is created by flagging intra-graph edges, signifying the absence of edges connecting the pattern and target nodes. Conversely, the $\boldsymbol{A}^{cr}$ matrix considers a "virtual" edge connecting a pattern node and a target node in instances where they share identical labels. The mathematical definitions for $\boldsymbol{X}$, $\boldsymbol{A}^{in}$ and $\boldsymbol{A}^{cr}$ are formally expressed in equations (1), (2), and (3), respectively.

$$\boldsymbol{X} = \{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_{|V_\mathcal{P}|}, \vec{x}_{|V_\mathcal{P}|+1}, \dots, \vec{x}_{|V_\mathcal{P}|+|V_\mathcal{T}|}\} \quad \text{with } \vec{x}_i \in \mathbb{R}^{2|\Sigma|} \tag{1}$$

$$\boldsymbol{A}^{in}_{ij} = \begin{cases} 1 & \text{if there is an indirected edge or} \\ & \text{a directed edge that connects } j \text{ to } i \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$$\boldsymbol{A}^{cr}_{ij} = \begin{cases} \boldsymbol{A}^{in}_{ij} & \text{if } i, j \in \mathcal{P} \text{ or } i, j \in \mathcal{T} \\ 1 & \text{if } l(i) = l(j) \text{ and } i \in \mathcal{P} \text{ and } j \in \mathcal{T}, \\ & \text{or if } l(i) = l(j) \text{ and } i \in \mathcal{T} \text{ and } j \in \mathcal{P} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

VI. GRAPH LEARNABLE MULTI-HOP ATTENTION NETWORKS

In this section, we initially outline the process of extracting node features in Section VI-A. Following this, we introduce the workings of Learnable Multi-hop Attention mechanism at each layer in Section VI-B. Subsequently, we discuss incorporating multiple Graph Learnable Multi-hop Attention layers to facilitate learning higher dependencies within the networks, as presented in Section VI-C.

A. EXTRACTING NODE FEATURES

In this study, we proposed a Graph Learnable Multi-hop Attention layer denoted as GLeMa(·), which utilizes Graph

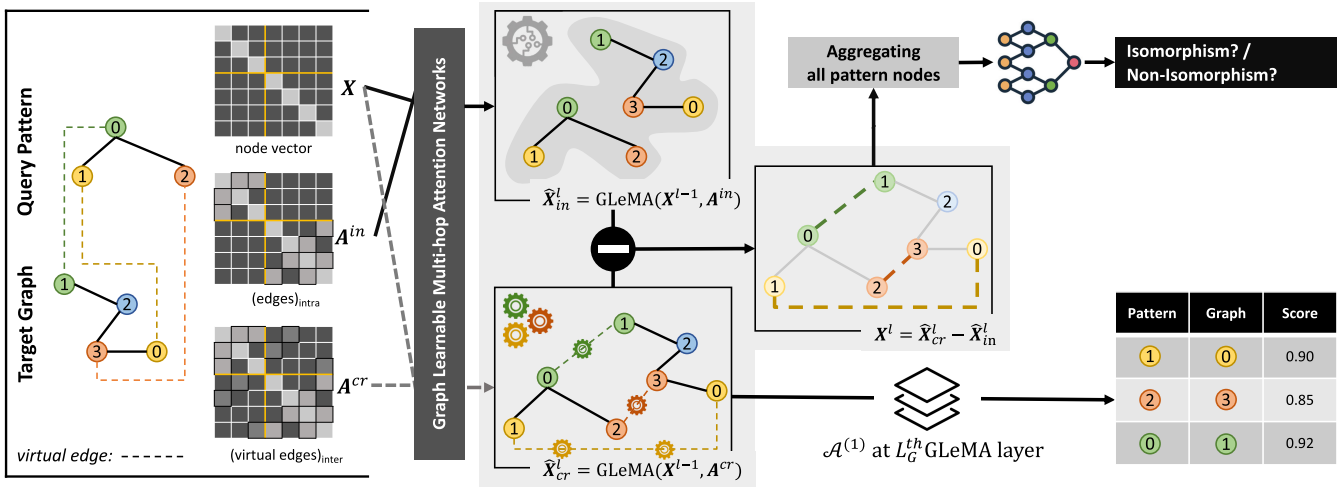


FIGURE 2. Overview of xNeuSM framework.

Attention Networks and Learnable Multi-hop Attention mechanism. This approach facilitates the acquisition of holistic structural information of both targets and patterns. Assuming that we are applying this layer on an abstract graph $\mathcal{G} = \{V, E, l\}$, we present this graph with $(\boldsymbol{X}, \boldsymbol{A})$ where $\boldsymbol{X} \in \mathbb{R}^{|V| \times F}$ is the set of node features; F is the number of features and $\boldsymbol{A}$ is the adjacency matrix. We formally denote the output of our proposed layer as $\widehat{\boldsymbol{X}} = \text{GLeMa}(\boldsymbol{X}, \boldsymbol{A})$. The input to our GLeMa layer encompasses a composite of node features, denoted as $\boldsymbol{X}$, and an adjacency matrix, denoted as $\boldsymbol{A}$, as described in equations (4) and (5), respectively.

$$\boldsymbol{X} = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{|V|}\}, \ \vec{x}_i \in \mathbb{R}^F \tag{4}$$

$$\boldsymbol{A}_{ij} = \begin{cases} 1 & \text{if there is an edge that connects } j \text{ to } i \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Subsequently, node feature vectors are projected into embedding space by a linear transformation $\vec{x}'_i = \boldsymbol{W}_h \vec{x}_i$, $\vec{x}'_i \in \mathbb{R}^{F'}$, where $\boldsymbol{W}_h \in \mathbb{R}^{F' \times F}$ is a learnable weight matrix. Then, we calculate an attention coefficient for each pair of nodes by using Luong's attention [45] as in (6).

$$e_{ij} = \begin{cases} \delta(\vec{x}'^{\top}_i \boldsymbol{W}_e \vec{x}'_j) & \text{a directed edge } j \text{ to } i, \\ \delta(\vec{x}'^{\top}_i \boldsymbol{W}_e \vec{x}'_j + \vec{x}'^{\top}_j \boldsymbol{W}_e \vec{x}'_i) & \text{an indirected edge } j \text{ to } i, \end{cases} \tag{6}$$

where δ is a non-linear activation function and $\boldsymbol{W}_e \in \mathbb{R}^{F' \times F'}$ is a learnable matrix. Subsequently, the normalization process involves subjecting all attention coefficients to the softmax function, resulting in the creation of the 1-hop attention matrix denoted as $\mathcal{A}^{(1)}$. Within this normalization procedure, the concept of *masked attention* is incorporated, wherein solely the nodes $j \in \mathcal{N}_i$ are considered for the normalization operation. Here, $\mathcal{N}_i$ represents the set of neighboring nodes of node i. Furthermore, to rigorously eliminate the influence of non-neighbor nodes, attention values normalized between two nodes i and j are replaced with zeros in instances where no edge is connecting node i to node j. The mathematical framework for normalizing attention coefficients is articulated in (7).

$$\begin{cases} \mathcal{A}^{(1)} = \{a^{(1)}_{ij} | i, j \in (1, |V|)\} \\ a^{(1)}_{ij} = \text{softmax}_j(e_{ij}) \boldsymbol{A}_{ij} = \dfrac{\exp(e_{ij})}{\sum_{n \in \mathcal{N}_i} \exp(e_{in})} \boldsymbol{A}_{ij} \end{cases} \tag{7}$$

Upon obtaining the 1-hop attention matrix, we derive the attention diffusion matrix $\mathcal{A}$ following the concept of multi-hop mechanism in GNNs [20]. Formally, the matrix $\mathcal{A}$ is defined in (8).

$$\begin{cases} (\mathcal{A}^{(1)})^0 = \boldsymbol{I} \\ \mathcal{A} = \sum_{k=0}^{\infty} \theta_k (\mathcal{A}^{(1)})^k \text{ where } \sum_{k=0}^{\infty} \theta_k = 1 \end{cases} \tag{8}$$

In the Equation (11), the parameter θ_k represents the attention decay factor, satisfying the condition $\theta_k > \theta_{k+1} > 0$ to ensure a progressive reduction in importance for more distant nodes. Subsequently, for each node i, weighted summations are conducted between itself and other nodes to create a new feature vector for i^{th} node, employing the multi-hop attention matrix.

Furthermore, the application of multi-head attention [20], [46] is executed to derive diverse feature representations from various distinct perspectives. The resultant vectors generated through multi-head attention are subsequently concatenated to yield the ultimate refined feature vectors for nodes. The formulation delineated in (9) elucidates the process of generating the ultimate output within this layer.

$$\widehat{\boldsymbol{X}} = \left(\Big\Vert_{h=1}^{H} \delta\left(\mathcal{A}_h \boldsymbol{X'}_h\right) \right) \boldsymbol{W}_o \text{ with } \boldsymbol{W}_o \in \mathbb{R}^{HF' \times F'}. \tag{9}$$

In (9), the symbol $\widehat{\boldsymbol{X}}$ represents the collection of resultant node feature vectors, where H signifies the count of attention heads employed in the multi-head attention mechanism. The term $\mathcal{A}_h$ denotes the multi-hop attention matrix associated

with the h^{th} attention head. Correspondingly, $\boldsymbol{X'}_h$ designates the matrix representing projected node features for the h^{th} attention head, while $\boldsymbol{W}_o$ stands for a matrix of learnable weights.

B. LEARNABLE MULTI-HOP ATTENTION MECHANISM

The subsequent challenges confronting our research pertaining to (i) the elevated computational intricacies involved in computing $\mathcal{A}$ due to matrix multiplication [47], as well as (ii) the judicious selection of the suitable values for θ_k, which significantly influences the augmentation or attenuation of the model performance [47].

1) REDUCING MULTI-HOP ATTENTION MATRIX COMPUTATION COMPLEXITY

In this study, following the methodology outlined in the previous GMA work [20], we adopt the geometric distribution to determine θ_k, wherein we choose $\theta_k = \alpha(1-\alpha)^k, \alpha \in (0,1)$ represents the teleport probability. Consequently, an approximation for $\mathcal{A}\boldsymbol{X'}$ is achieved via the utilization of equation (10).

$$\begin{cases} \boldsymbol{Z}^{(0)} = \boldsymbol{X'} \\ \boldsymbol{Z}^{(k)} = (1-\alpha)\mathcal{A}^{(1)}\boldsymbol{Z}^{(k-1)} + \alpha\boldsymbol{Z}^{(0)}, \ k \in (1,K) \end{cases} \tag{10}$$

Proposition 1: $lim_{K\to\infty}\boldsymbol{Z}^{(K)} = \mathcal{A}\boldsymbol{X'}$

This proposition was proved in [20], demonstrating that we can reduce the complexity of calculating $\mathcal{A}$ to $O(K|E|)$ message passing operators. Nonetheless, K can be eliminated by considering K as a constant and $K \ll +\infty$. Doing so raises a concern about how well the $\mathcal{A}X'$ is approximated by $Z^{(K)}$. In other words, the concern is selecting the K that balances the trade-off between approximation error and computing complexity. The former works [20] suggested choosing $3 \leq K \leq 10$ by empirical experiments. In this study, we provide the theoretical justification for this selection strategy in Section VIII-A. Consequently, assuming that $\mathcal{A}$ is approximated by $\mathcal{A}^{(K)}$ (equation 11), the complexity of computing $\mathcal{A}$ is reduced to just $O(|E|)$.

$$\begin{cases} (\mathcal{A}^{(1)})^0 = \boldsymbol{I} \\ \mathcal{A} \approx \mathcal{A}^{(K)} = \boldsymbol{Z}^{(K)}\boldsymbol{X'}^{-1} \end{cases} \tag{11}$$

2) LEARNING TELEPORT PROBABILITIES

In graph attention diffusion theory, a critical concept is the definition of Personalized PageRank (PPR) [48], which reveals the importance of each node. The original work of GMA has proved that the attention matrix in GMA can be viewed as the transition matrix in PPR [20]. However, the original work used the same teleport probability for all nodes, which theoretically limits the power of PPR. Therefore, leveraging the capabilities of PPR, we propose a novel approach that involves the customization of teleport probabilities for individual nodes, denoted as $\beta = \{\beta_v\}_{v=1}^{|V|}$. The primary challenge at hand revolves around selecting appropriate β_v values for each node. To address this challenge, inspired by Gated Recurrent Units [49], we devise a method wherein the network autonomously learns the teleport probabilities via a straightforward linear transformation with $\boldsymbol{W}_\beta \in \mathbb{R}^{2F'\times 1}$. The equation presented in (12) delineates how β is derived.

$$\beta = \sigma((\boldsymbol{X'}||\mathcal{A}^{(1)}\boldsymbol{X'})\boldsymbol{W}_\beta + b), \tag{12}$$

where $||$ denotes the concatenation operator, and b represents the bias term. It is worth noting that employing distinct teleport probabilities for each node results in modifications to equation 8. These alterations encompass $\theta_{kj} = \beta_j(1-\beta_j)^k$, $\sum_{k=0}^{\infty}\theta_{kj} = 1$ for $j \in (1,N)$, and $\theta_{kj} > 0$. Furthermore, they involve a row-wise multiplication between θ_k and $(\mathcal{A}^{(1)})^k$. It is important to emphasize that these changes do not contravene **Proposition 1**, as established in Section VIII-B. Additionally, we provide a comprehensive procedure for computing learnable multi-hop attention in Algorithm 2.

Algorithm 2 Learnable Multi-Hop Attention

Input : 1-hop attention matrix $\mathcal{A}^{(1)}$
Node feature matrix $\boldsymbol{X'}$
Number of approximate hops K
Output: Diffused node feature matrix $\widehat{\boldsymbol{X}}$
$\boldsymbol{Z}^{(0)} \leftarrow \boldsymbol{X'}$
$\beta \leftarrow \sigma((\boldsymbol{X'}||\mathcal{A}^{(1)}\boldsymbol{X'})\boldsymbol{W}_\beta + b)$
for k ***in*** *Range*$(1\ldots K)$ **do**
  $\boldsymbol{Z}^{(k)} = (1-\beta)\mathcal{A}^{(1)}\boldsymbol{Z}^{(k-1)} + \beta\boldsymbol{Z}^{(0)}$
end
$\widehat{\boldsymbol{X}} \leftarrow \boldsymbol{Z}^{(K)}$

C. GRAPH LEARNABLE MULTI-HOP ATTENTION NETWORKS

When processing input data that comprises a pattern and a target, denoted as a triple $(\boldsymbol{X}, \boldsymbol{A}^{in}, \boldsymbol{A}^{cr})$, we employ our proposed GLeMa layer (GLeMa(·)) to extract hidden features. Specifically, the input is bifurcated into two tuples: $(\boldsymbol{X}, \boldsymbol{A}^{in})$ and $(\boldsymbol{X}, \boldsymbol{A}^{cr})$, which are subsequently subjected to L_G iterations of GLeMa layers. The resulting representation for $(\boldsymbol{X}, \boldsymbol{A}^{in})$ captures intra-graph features, whereas the representation for $(\boldsymbol{X}, \boldsymbol{A}^{cr})$ captures inter-graph features.

The node features at the l^{th} layer are computed by taking the difference between the inter-graph features and the intra-graph features from the previous $(l-1)^{th}$ layer. This learning of disparities between inter-graph and intra-graph features enhances the signal for verifying subgraph isomorphism. The formal definition of the Graph Learnable Multi-hop Attention network architecture is presented in equation (13).

$$\begin{cases} \boldsymbol{X}^0 = \boldsymbol{X} \\ \widehat{\boldsymbol{X}}^l_{in} = \text{GLeMa}_l(\boldsymbol{X}^{l-1}, \boldsymbol{A}^{in}), \ l \in (1, L_G) \\ \widehat{\boldsymbol{X}}^l_{cr} = \text{GLeMa}_l(\boldsymbol{X}^{l-1}, \boldsymbol{A}^{cr}), \ l \in (1, L_G) \\ \boldsymbol{X}^l = \widehat{\boldsymbol{X}}^l_{cr} - \widehat{\boldsymbol{X}}^l_{in}, \ l \in (1, L_G) \end{cases} \tag{13}$$

VII. MULTI-TASK OPTIMIZATION

In this section, we introduce the optimization mechanisms for both the subgraph matching task and the matching explanation task, discussed in Section VII-A and Section VII-B, respectively. Subsequently, we delve into the optimization objectives for multi-task learning, as outlined in Section VII-C.

A. SUBGRAPH MATCHING TASK

In the context of subgraph matching, after their extraction via L_G GLeMa layers, the node feature vectors derived from patterns are aggregated to generate a representation vector. This representation vector is pivotal in determining the isomorphism between the input pattern and the target graph, achieved through a classifier comprising L_{FC} fully connected layers. The methodology for computing the representation vector is elucidated in equation (14), while equation (15) provides the mathematical formulations underpinning the classifier.

$$x_{repr}^{0} = \frac{1}{|V_{\mathcal{P}}|} \sum_{i \in V_{\mathcal{P}}} x_i^{L_G} \tag{14}$$

$$\begin{cases} x_{repr}^{i} = \delta(\boldsymbol{W}_i x_{repr}^{i-1} + b_i), \ i \in (1, L_{FC} - 1) \\ \widehat{y} = \sigma(\boldsymbol{W}_y x_{repr}^{L_{FC}-1} + b_y) \end{cases} \tag{15}$$

In equation (14), we denote x_{repr}^{0} as the representation vector of the input, and x_i^{LG} as the embedding vector of the i^{th} node after undergoing L_G GLeMa layers. In equation (15), we represent x_{repr}^{i} as the output vector of the i^{th} fully-connected layer, with $\boldsymbol{W}_i$ and b_i signifying the respective learnable weight matrix and bias parameters.

B. MATCHING EXPLANATION TASK

Leveraging the efficacy of the multi-hop attention mechanism, our proposed model can predict the mapping of a pattern within a target graph. The enigmatic aspect of our approach entails filtering pairs of *(pattern node, target node)* based on the 1-hop attention coefficients obtained from the final inter-graph GLeMa layer, subject to a predetermined threshold. Assuming that a threshold-compliant pair signifies a valid mapping between a pattern node and a target node, our model can enumerate all potential mappings, irrespective of whether the input pattern is isomorphic to the target or not. The precise computational details of this matching task are expounded upon in (16). In equation (16), $\mathcal{M}$ is the set of mapping nodes between the pattern and target graph; p_{ij} which is computed by average of 1-hop attention coefficients $((a_{ij}^{(1)})^{L_G}, (a_{ji}^{(1)})^{L_G})$ is the mapping probability between pattern i^{th} node and target j^{th} node.

$$\begin{aligned} &\mathcal{M} = \{(i, j, p_{ij}) | p_{ij} \geq \epsilon\}, \text{where } i \in V_{\mathcal{P}}, j \in V_{\mathcal{T}} \text{ and} \\ &p_{ij} = \frac{1}{2}\left((a_{ij}^{(1)})^{L_G} + (a_{ji}^{(1)})^{L_G}\right) \end{aligned} \tag{16}$$

C. MULTI-TASK LEARNING OPTIMIZATION

In order to optimize our proposed models for the dual tasks of subgraph matching and matching explanation, we introduce a composite loss function consisting of two fundamental components. The first component, denoted as $\mathcal{L}_{sm}$, is a binary cross-entropy loss designed to accurately assess the model's capacity to predict subgraph isomorphism. The second component, $\mathcal{L}_{me}$, is an attention-based loss aimed at reinforcing the attention coefficients between nodes i and j ($i \in V_{\mathcal{P}}, j \in V_{\mathcal{T}}$) that correspond to actual mappings, while simultaneously diminishing the coefficients for node pairs sharing the same label (represented as $m \in V_{\mathcal{P}}, n \in V_{\mathcal{T}}, l(m) = l(n)$) but lacking a mapping relationship.

Our overarching objective function, as expressed in equation (17), as shown at the bottom of the next page, comprises a combination of $\mathcal{L}_{sm}$ and $\lambda\mathcal{L}_{me}$, where N is the total training samples and the weight λ serves as a hyperparameter for regulating the relative importance of the two loss components.

VIII. THEORETICAL JUSTIFICATIONS

A. MULTI-HOP ATTENTION APPROXIMATION ERROR

In this section, we will justify the multi-hop attention approximation error and provide guidance on choosing the appropriate number of approximate hops K. Specifically, let $\mathcal{A}$ denote the exact attention diffusion matrix defined in Equation (8). By Proposition 1, we can derive:

$$\lim_{K \to \infty} \boldsymbol{Z}^{(K)} (\boldsymbol{X'})^{-1} = \mathcal{A}. \tag{18}$$

Let $\mathcal{A}^{(K)} = \boldsymbol{Z}^{(K)}(\boldsymbol{X'})^{-1}$ be the approximated attention diffusion matrix at K-hop. We will show that the error $\text{Err}(\mathcal{A} - \mathcal{A}^{(K)}) \leq (1-\alpha)^{K+1}$, where α is the teleport probability and K is the number of hops.

Firstly, we decompose $\boldsymbol{Z}^{(K)}$ as follows:

$$\begin{aligned} \boldsymbol{Z}^{(K)} = &(1-\alpha)^K (\mathcal{A}^{(1)})^K \boldsymbol{X'} + \alpha(1-\alpha)^{K-1}(\mathcal{A}^{(1)})^{K-1}\boldsymbol{X'} \\ &+ \cdots + \alpha(1-\alpha)(\mathcal{A}^{(1)})\boldsymbol{X'} + \alpha \boldsymbol{X'} \end{aligned} \tag{19}$$

Then, we obtain:

$$\begin{aligned} \boldsymbol{Z}^{(K)}(\boldsymbol{X'})^{-1} &= (1-\alpha)^K (\mathcal{A}^{(1)})^K + \alpha(1-\alpha)^{K-1}(\mathcal{A}^{(1)})^{K-1} \\ &\quad + \cdots + \alpha(1-\alpha)(\mathcal{A}^{(1)}) + \alpha \\ &= (1-\alpha)^K (\mathcal{A}^{(1)})^K + \sum_{k=0}^{K-1} \alpha(1-\alpha)^k (\mathcal{A}^{(1)})^k \end{aligned} \tag{20}$$

Now, let us consider the difference between the attention diffusion matrix $\mathcal{A}$ and its approximation $\mathcal{A}^{(K)}$.

$$\begin{aligned} \mathcal{A} - \mathcal{A}^{(K)} &= \mathcal{A} - \boldsymbol{Z}^{(K)}(\boldsymbol{X'})^{-1} \\ &= \sum_{k=0}^{\infty} \alpha(1-\alpha)^k (\mathcal{A}^{(1)})^k - (1-\alpha)^K (\mathcal{A}^{(1)})^K \\ &\quad - \sum_{k=0}^{K-1} \alpha(1-\alpha)^k (\mathcal{A}^{(1)})^k \\ &= \sum_{k=K}^{\infty} \alpha(1-\alpha)^k (\mathcal{A}^{(1)})^k - (1-\alpha)^K (\mathcal{A}^{(1)})^K \end{aligned}$$

$$\leq \sum_{k=K}^{\infty} \alpha(1-\alpha)^k (\mathcal{A}^{(1)})^k - \alpha(1-\alpha)^K (\mathcal{A}^{(1)})^K$$
$$\text{as } \alpha, a_{ij}^{(1)} \in (0,1)$$
$$\leq \sum_{k=K+1}^{\infty} \alpha(1-\alpha)^k (\mathcal{A}^{(1)})^k \tag{21}$$

We also have $a_{ij}^{(1)} \in (0,1)$ so that:

$$(\mathcal{A}^{(1)})^k \leq (\mathcal{A}^{(1)})^{k-1}. \tag{22}$$

As a consequence, we have:

$$(\mathcal{A}^{(1)})^k \leq \mathcal{A}^{(1)}, \forall k \geq 1 \tag{23}$$

Using (23), equation (21) can be derived as follows:

$$\boldsymbol{E} = \mathcal{A} - \mathcal{A}^{(K)} \leq \left(\sum_{k=K+1}^{\infty} \alpha(1-\alpha)^k \right) \mathcal{A}^{(1)} \tag{24}$$

It is easy to observe that $\boldsymbol{E} \in \mathbb{R}^{|V|\times|V|}$. Then, we can compute the average difference between the exact and approximate attention diffusion matrix as below:

$$\begin{aligned}
\text{Err}(\mathcal{A} - \mathcal{A}^{(K)}) &= \frac{1}{|V|^2} \sum_{i,j} \boldsymbol{E}_{ij} \\
&\leq \frac{1}{|V|^2} \sum_{i,j} \left(\left(\sum_{k=K+1}^{\infty} \alpha(1-\alpha)^k \right) a_{ij}^{(1)} \right) \\
&\leq \sum_{k=K+1}^{\infty} \alpha(1-\alpha)^k \text{ as } a_{ij}^{(1)} \in (0,1) \\
&\leq \alpha \sum_{k=K+1}^{\infty} (1-\alpha)^k \leq \alpha \frac{(1-\alpha)^{K+1}}{1-(1-\alpha)} \\
&\leq (1-\alpha)^{K+1}
\end{aligned} \tag{25}$$

It is readily apparent that when the error is constrained by the condition $\text{Err}(\mathcal{A} - \mathcal{A}^{(K)}) \leq (1-\alpha)^{K+1}$, selecting K from the set $\{3, \ldots, 10\}$ yields errors that are consistently below the threshold of 0.3 for values of α greater than or equal to 0.3. To further illustrate this observation, we have presented a graphical representation of the error as a function of K in Figure 3. Additionally, as $K \to \infty$, $\lim_{K\to\infty} \text{Err}(\mathcal{A} - \mathcal{A}^{(K)}) \leq \lim_{K\to\infty}(1-\alpha)^{K+1} = 0$, which further proves the approximation's accuracy.

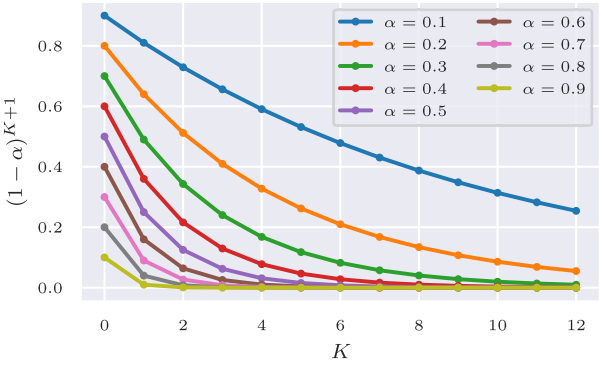


FIGURE 3. Maximal approximation error of each attention coefficient.

B. CORRECTNESS OF SEPARATED TELEPORT PROBABILITY FOR EACH NODE

This section establishes the validity of employing distinct teleport probability, denoted as $\beta_v \in (0,1)$ for each node $v \in V$, without violating Proposition 1. This variation in teleport probabilities results in varying attention decay factors for each node $v \in V$, denoted as $\eta_v \in \mathbb{R}$. Specifically, with k representing the number of hops, we have:

$$(\eta_v)_k = \beta_v(1-\beta_v)^k > 0. \tag{26}$$

This leads to the important property:

$$\forall v \in V, \sum_{k=0}^{\infty} (\eta_v)_k = \sum_{k=0}^{\infty} \beta_v(1-\beta_v)^k = \frac{\beta_v}{1-(1-\beta_v)} = 1. \tag{27}$$

Let $\eta_k = \{(\eta_v)_k\}_{v=1}^{|V|}$ be the attention decay vector at the k-th hop. With the property in (27), we can generalize equation (8) as follows:

$$\begin{cases} (\mathcal{A}^{(1)})^0 = \boldsymbol{I} \\ \mathcal{A}_\eta = \sum_{k=0}^{\infty} \eta_k (\mathcal{A}^{(1)})^k. \end{cases} \tag{28}$$

In the case where $\beta_i = \beta_j, \forall i,j \in (1,|V|)$, $\mathcal{A}_\eta$ is equivalent to $\mathcal{A}$. Let $\beta = \{\beta_v\}_{v=1}^{|V|}$ be the teleport probability vector. Using the same technique as in Section VI-B, we can approximate $\mathcal{A}_\eta \boldsymbol{X'}$ with $\boldsymbol{Z}_\beta^{(k)}$ as follows:

$$\begin{cases} \boldsymbol{Z}_\beta^{(0)} = \boldsymbol{X'} \\ \boldsymbol{Z}_\beta^{(k)} = (\vec{1}-\beta)\mathcal{A}^{(1)}\boldsymbol{Z}_\beta^{(k-1)} + \beta \boldsymbol{Z}_\beta^{(0)}, \ k = (1,K) \end{cases} \tag{29}$$

To ensure the approximation is correct, we need to prove that as $K \to +\infty$, $\boldsymbol{Z}_\beta^{(k)}$ approximates $\mathcal{A}_\eta \boldsymbol{X'}$.

Proposition 2: $\lim_{K\to\infty} \boldsymbol{Z}_\beta^{(K)} = \mathcal{A}_\eta \boldsymbol{X'}$.

$$\begin{cases} \mathcal{L}_{sm} = -\frac{1}{N} \sum_{k=1}^{N} y_k \cdot \log(\widehat{y}_k) + (1-y_k) \cdot \log(1-\widehat{y}_k) \\ \mathcal{L}_{me} = \frac{1}{N} \sum_{k=1}^{N} \frac{\sum \exp\left(-\left(a_{ij}^{(1)(L_G)}\right)_k\right)}{\sum \exp\left(-\left(a_{mn}^{(1)(L_G)}\right)_k\right) - \sum \exp\left(-\left(a_{ij}^{(1)(L_G)}\right)_k\right) + 1} \\ \mathcal{L} = \mathcal{L}_{sm} + \lambda \mathcal{L}_{me} \end{cases} \tag{17}$$

Proof of Proposition 2: Firstly, we decompose all elements of $\boldsymbol{Z}_\beta^{(K)}$:

$$\begin{aligned}\boldsymbol{Z}_\beta^{(K)} &= \underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{K \text{ times}}\boldsymbol{X'} + \underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{K-1 \text{ times}}\beta\boldsymbol{X'} \\ &+ \underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{K-2 \text{ times}}\beta\boldsymbol{X'} + \dots + (\vec{1}-\beta)\mathcal{A}^{(1)}\beta\boldsymbol{X'} \\ &+ \beta\boldsymbol{X'} \end{aligned} \tag{30}$$

We also have:

$$\begin{aligned}\mathcal{A}_\eta\boldsymbol{X'} &= \left(\sum_{k=0}^{\infty}\eta_k(\mathcal{A}^{(1)})^k\right)\boldsymbol{X'} \\ &= \eta_0\boldsymbol{X'} + \eta_1\mathcal{A}^{(1)}\boldsymbol{X'} + \eta_2(\mathcal{A}^{(1)})^2\boldsymbol{X'} + \dots \\ &= \beta\boldsymbol{X'} + \beta(\vec{1}-\beta)\mathcal{A}^{(1)}\boldsymbol{X'} \\ &\quad + \beta(\vec{1}-\beta)^2(\mathcal{A}^{(1)})^2\boldsymbol{X'} + \dots \end{aligned} \tag{31}$$

To prove Proposition 2, we need to prove the following Lemma 1 and Lemma 2.

Lemma 1: $\underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{k \text{ times}}\beta\boldsymbol{X'} = \beta(\vec{1}-\beta)^k(\mathcal{A}^{(1)})^k\boldsymbol{X'}$

Proof of Lemma 1: By the commutative property of row-wise multiplication between a vector and a matrix, we can write:

$$(\vec{1}-\beta)\mathcal{A}^{(1)} = \mathcal{A}^{(1)}(\vec{1}-\beta). \tag{32}$$

This leads to:

$$\begin{aligned}\underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{k \text{ times}}\beta\boldsymbol{X'} &= (\underbrace{(\vec{1}-\beta)\dots}_{k \text{ times}})(\underbrace{\mathcal{A}^{(1)}\dots}_{k \text{ times}})\beta\boldsymbol{X'} \\ &= (\vec{1}-\beta)^k(\mathcal{A}^{(1)})^k\beta\boldsymbol{X'} \\ &= \beta(\vec{1}-\beta)^k(\mathcal{A}^{(1)})^k\boldsymbol{X'}. \end{aligned} \tag{33}$$

Thus, Lemma 1 has been demonstrated. □

Lemma 2: $\lim_{k\to\infty}\underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{k \text{ times}}\boldsymbol{X'} = O$

Proof of Lemma 2: Because $\beta_v \in (0, 1)$, it follows that $(1-\beta_v) \in (0, 1)$. This implies that:

$$\lim_{k\to\infty}(1-\beta_v)^k = 0. \tag{34}$$

With $\beta = \{\beta_v\}_{v=1}^{|V|}$, we can derive that

$$\lim_{k\to\infty}(\vec{1}-\beta)^k = \{\lim_{k\to\infty}(1-\beta_v)^k\}_{v=1}^{|V|} = \vec{0}. \tag{35}$$

With Lemma 1 and (35), we can conclude:

$$\begin{aligned}\lim_{k\to\infty}\underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{k \text{ times}}\boldsymbol{X'} &= \lim_{k\to\infty}(\vec{1}-\beta)^k(\mathcal{A}^{(1)})^k\boldsymbol{X'} \\ &= (\lim_{k\to\infty}(\vec{1}-\beta)^k)(\lim_{k\to\infty}(\mathcal{A}^{(1)})^k\boldsymbol{X'}) \\ &= \vec{0}(\lim_{k\to\infty}(\mathcal{A}^{(1)})^k\boldsymbol{X'}) \\ &= O. \end{aligned} \tag{36}$$

Thus, Lemma 2 is proven. □

By proving Lemma 1 and Lemma 2, we can prove Proposition 2 as follows:

$$\begin{aligned}&\lim_{K\to\infty}\boldsymbol{Z}_\beta^{(K)} \\ &= \lim_{K\to\infty}\left[\underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{K \text{ times}}\boldsymbol{X'}\right] \\ &\quad + \lim_{K\to\infty}\left[\underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{K-1 \text{ times}}\beta\boldsymbol{X'}\right. \\ &\quad \left. + \dots + (\vec{1}-\beta)\mathcal{A}^{(1)}\beta\boldsymbol{X'} + \beta\boldsymbol{X'}\right] \\ &= O + \lim_{K\to\infty}\left[\beta(\vec{1}-\beta)^{K-1}(\mathcal{A}^{(1)})^{K-1}\boldsymbol{X'} + \dots\right. \\ &\quad \left. + \beta(\vec{1}-\beta)\mathcal{A}^{(1)}\boldsymbol{X'} + \beta\boldsymbol{X'}\right] \\ &= \lim_{K\to\infty}\left[\eta_{K-1}(\mathcal{A}^{(1)})^{K-1}\boldsymbol{X'} + \dots + \eta_1\mathcal{A}^{(1)}\boldsymbol{X'} + \eta_0\boldsymbol{X'}\right] \\ &= \lim_{K\to\infty}\left[\left(\sum_{k=0}^{K-1}\eta_k(\mathcal{A}^{(1)})^k\right)\right]\boldsymbol{X'} \\ &= \mathcal{A}_\eta\boldsymbol{X'} \quad \text{as} \quad \mathcal{A}_\eta = \sum_{k=0}^{\infty}\eta_k(\mathcal{A}^{(1)})^k \quad \text{in (28).} \end{aligned} \tag{37}$$

Thus, Proposition 2 is proven. □

IX. EXPERIMENTS

In this section, we provide comprehensive evaluation protocols designed for xNeuSM. Subsequently, we conduct rigorous experiments using six real-world datasets to assess the performance of xNeuSM. The primary objective is to address the following research questions (RQs) through systematic experiments:

- **RQ1**: Does xNeuSM exhibit superior performance compared to diverse baseline techniques in the subgraph matching task?
- **RQ2**: How confident are the predictions made by xNeuSM?
- **RQ3**: What impact do variations in pattern size and density have on the performance of xNeuSM?
- **RQ4**: How effectively does xNeuSM perform in the matching explanation task?
- **RQ5**: What is the individual contribution and impact of each component of xNeuSM?
- **RQ6**: Can xNeuSM effectively handle inductive settings, adapting to new, previously unseen graphs?
- **RQ7**: Does xNeuSM retain its effectiveness when dealing with directed subgraph matching and explanation scenarios?

Each research question is carefully designed to delve into specific aspects of xNeuSM's performance, scalability, interpretability, and adaptability across varying conditions and settings, providing a nuanced understanding of its capabilities and limitations. The experiments conducted aim to provide robust empirical evidence in addressing the above research inquiries, proving the effectiveness of our framework in reality.

A. EXPERIMENTAL SETUP

In this subsection, a detailed exposition of our experimental framework is presented, encompassing the selection of datasets, baseline methods, data pre-processing methodologies, and the evaluation metrics employed. This comprehensive account aims to elucidate the specifics of our experimental setup, ensuring transparency and replicability while providing clarity on the foundational elements shaping our assessment methodology.

1) DATASETS

We assess the performance of our framework across a diverse range of real datasets encompassing various domains, including bioinformatics, chemistry, computer vision and social networks. To evaluate the effectiveness of xNeuSM, we conduct experiments on six well-established real-world datasets frequently employed in various applications [50] within graph mining research. The statistics for these datasets are summarized in Table 2. In this table, $\overline{|V_{\mathcal{T}}|}$, $\overline{|E_{\mathcal{T}}|}$, $\overline{deg}$, and $|\mathcal{D}|$ represent the average number of nodes, the average number of edges, the average degree of a node, and the number of graphs in the dataset, respectively.

2) BASELINE TECHNIQUES

We provide details of our baseline methods, including state-of-the-art exact and approximate approaches as follows.

Exact approach: We utilize seven distinct approaches as below.

- *VF3* [51] is an extension of the VF2 algorithm [52] designed to handle larger graphs. It employs an enhanced bit vector representation for graph states, introduces a novel matching order for query nodes, and incorporates various heuristics like degree-based filtering and dynamic node reordering to enhance matching efficiency.
- *TurboISO* [11] is an efficient subgraph isomorphism algorithm that utilizes pre-processing, constructing an index for rapid candidate subgraph filtering. It employs a divide-and-conquer strategy along with heuristics such as degree-based filtering and forward checking to narrow down the search space.
- *CFL* [12] minimizes redundant Cartesian products in the search space by strategically postponing them based on query structure. It introduces a compact path-based auxiliary data structure for efficient matching.
- *CECI* [13] partitions the target graph into multiple clusters for parallel processing and uses BFS-based filtering, reverse-BFS-based refinement, and set intersection techniques to optimize the search process.
- *QuickSI* [10] employs QI-Sequence to constrain the search space, determining order based on feature frequencies. It introduces Swift-Index, reducing costs in the filtering phase.
- *DAF* [14] introduces concepts like dynamic programming between a directed acyclic graph (DAG) query and a data graph, adaptive matching order with DAG ordering, and pruning by failing sets to address limitations of existing algorithms.
- *GraphQL* [17] introduces a specialized query language for graph databases, extending formal languages from strings to graphs. It uses neighborhood subgraphs and profiles to optimize the search order and reduce the search space.

TABLE 2. Statistics of real datasets.

Domain	$\mathcal{D}$	$\overline{\lvert V_{\mathcal{T}}\rvert}$	$\overline{\lvert E_{\mathcal{T}}\rvert}$	$\overline{deg}$	$\lvert\mathcal{D}\rvert$	$\lvert\Sigma\rvert$
Bioinformatics	KKI	26.96	48.42	3.19	83	190
Chemistry	COX2	41.22	43.45	2.10	467	8
Chemistry	COX2_MD	26.28	335.12	25.27	303	7
Chemistry	DHFR	42.43	44.54	2.10	756	9
Social networks	DBLP-v1	10.48	19.65	3.43	19456	39
Computer vision	MSRC-21	77.52	198.32	5.10	563	22

Approximate approach: Our comparison includes

- *NeuralMatch* [15], a cutting-edge subgraph matching algorithm employing a specialized graph neural network architecture. This approach efficiently identifies the neighborhood within a large target graph that encompasses a smaller query graph as a subgraph. Through a GNN, it acquires robust graph embeddings in an ordered space, encapsulating structural properties like transitivity, antisymmetry, and non-trivial intersections. As a result, NeuralMatch achieves real-time approximate subgraph matching at an unprecedented scale.
- *DualMP* [16], one of state-of-the-art approaches for performing subgraph counting and matching. This method leverages Dual Message Passing Neural Networks (DMPNNs) to learn both node and edge representations simultaneously in an aligned space through an efficient asynchronous update mechanism. This helps capturing comprehensive graph structure information. The method naturally extends to heterogeneous multi-graphs and shows strong performance across multiple tasks, including subgraph isomorphism counting, matching, and unsupervised node classification. It also incorporate multi-task learning which allows for mutual supervision between tasks, improving overall performance.

3) DATA PREPARATION

In our experimental setup, distinct training and testing datasets are utilized. The testing dataset comprises real-world instances, while the training dataset is artificially synthesized to mirror the size and degree distribution of the respective testing dataset.

For each graph in the testing dataset $\mathcal{D}$, we generate 2000 queries, half of which are isomorphic to the graphs and vice versa. The sizes of the query graphs vary from 2 to the size of the data graph, adhering to a *Uniform* distribution. The average degrees of the queries follow a Normal distribution $\mathcal{N}(\overline{deg}, \sigma^2_{\mathcal{D}}(deg))$. Corresponding to each real dataset, we create a synthetic training dataset that

replicates the graph size and degree distribution. We adopt an identical process to generate 2000 queries for each target graph, akin to the procedure employed in the testing set. The number of target graphs in these synthetic training datasets is four times greater than in the real datasets.

4) METRICS

a: SUBGRAPH MATCHING TASK

In this task, we conduct a comparative analysis of our proposed approach by evaluating it against exact methods and approximate methods using various metrics to assess its runtime and performance comprehensively. The metrics utilized in this evaluation are as follows:

- *Execution time*: This metric denotes the average processing time for a query (target graph, query pattern), excluding disk loading time.
- *ROC AUC*: The ROC curve illustrates model performance by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) at various thresholds. The AUC quantifies overall performance based on the curve's area, ranging from 0 to 1. Higher values denote better performance.
- *PR AUC*: This metric uses Precision-Recall curves, emphasizing Precision and Recall rates over TPR and FPR. It quantifies the area under the Precision-Recall curve, providing a more accurate assessment of imbalanced datasets. Higher PR AUC values indicate better performance.
- *F1 score*: The F1 score, a harmonic mean of Precision and Recall, offers a balanced evaluation of the model's performance. Precision measures accuracy in identifying positive instances, while Recall assesses the ability to capture all actual positives.

b: MATCHING EXPLANATION TASK

To demonstrate the efficacy of identifying mappings of isomorphic subgraphs, we extract the attention matrix from the last GLeMA layer in the branch that incorporates cross-graph connections. Subsequently, we rank the mappings of each query node within the transaction according to their respective attention scores. We then employ the following two metrics for evaluation:

- *Average Top-K Accuracy*: Assuming that acc_i^K represents the Top-K accuracy of node i in the query, we compute the average Top-K accuracy across all testing samples using the following equation.

$$TopK = \frac{1}{|\mathcal{D}_{test}|} \sum_{(\mathcal{T},\mathcal{P})\in\mathcal{D}_{test}} \left(\frac{1}{|V_\mathcal{P}|} \sum_{i\in V_\mathcal{P}} Acc_i^K \right)$$

- *Mean Reciprocal Rank*: This metric assesses the model's capability to predict the correct mapping with a high probability. It is computed by taking the multiplicative inverse of the rank of the first correct mapping. Let $rank_i$ represent the ranking of the correct mapping for query node i. The average reciprocal ranking across test samples can be calculated as follows.

$$MRR = \frac{1}{|\mathcal{D}_{test}|} \sum_{(\mathcal{T},\mathcal{P})\in\mathcal{D}_{test}} \left(\frac{1}{|V_\mathcal{P}|} \sum_{i\in V_\mathcal{P}} \frac{1}{rank_i} \right)$$

5) REPRODUCIBILITY

All experiments were conducted on a machine equipped with a 32-core CPU, 128GB of RAM, and an NVIDIA 2080Ti GPU with 12GB of memory. Our implementation can be found at https://github.com/martinakaduc/xNeuSM.

6) HYPERPARAMETER SETTINGS

To support the community to reproduce xNeuSM, we provide our hyperparameter settings for training our xNeuSM model in Table 3.

TABLE 3. Hyperparameter settings for our xNeuSM model.

Hyperparameter(s)	Value(s)
Learning rate	10^{-4}
Optimizer	Adam
Number of epochs	30
Number of GMA layers	4
Number of hops	1 3 5 7
Hidden dimension in GMA layer	140
Number of attention head	1
Number of FC layers	4
Hidden dimension in FC layer	128
λ	1.0

B. SUBGRAPH MATCHING

To address the first research question (**RQ1**), we conduct a comprehensive evaluation of xNeuSM's performance in an end-to-end manner, with a specific focus on several critical aspects: (i) execution time and (ii) performance in subgraph matching tasks, assessed through four distinct metrics which are ROC AUC, PR AUC, F1 score, and accuracy.

Execution Time across real datasets: We present the runtime of all methods in Figure 4. The results unequivocally demonstrate that our proposed solution, xNeuSM, has the shortest execution time compared to all state-of-the-art exact and approximate methods. This underscores xNeuSM's superior scalability, enabling it to process larger graphs effectively.

To perform a deeper analysis, we examine the time complexity of these methods. Let us consider the target graph as $\mathcal{T} = (V_\mathcal{T}, E_\mathcal{T}, l_\mathcal{T})$ and the query pattern as $\mathcal{P} = (V_\mathcal{P}, E_\mathcal{P}, l_\mathcal{P})$. Our approach represents the pair of the query pattern and target graph as a combined graph $\mathcal{G}$ with $V_\mathcal{G} = V_\mathcal{T} \cup V_\mathcal{P}$ and $E_\mathcal{G} = E_\mathcal{T} \cup E_\mathcal{P} \cup E_{virt}$, where E_{virt} denotes the set of cross-graph virtual edges. In each GLeMa layer, the computation of attention coefficients requires $O(H \cdot E_\mathcal{G})$ operations, and aggregating the features from neighboring nodes requires $O(K \cdot H \cdot V_\mathcal{G})$ operations, where H is the number of attention heads and K is the number

of hops. Consequently, our approach has a complexity of $O(2L_G H(E_\mathcal{G} + K \cdot V_\mathcal{G}))$. Given that L_G and H are fixed, the complexity of our method is approximated by $O(|E_\mathcal{T}| + |E_\mathcal{P}| + |E_{virt}| + K \cdot |V_\mathcal{T}| + K \cdot |V_\mathcal{P}|)$.

The complexity of the NeuralMatch method is demonstrated to be $O(L_G(|E_\mathcal{T}| + |E_\mathcal{P}|) + |V_\mathcal{T}| \times |V_\mathcal{P}|)$ [15]. With a fixed number of GNN layers, the complexity can be approximated as $O(|E_\mathcal{T}| + |E_\mathcal{P}| + |V_\mathcal{T}| \times |V_\mathcal{P}|)$. The component of NeuralMatch that incurs the largest cost of $O(|V_\mathcal{T}| \times |V_\mathcal{P}|)$ is the matching process required to determine whether the subgraph is isomorphic. Consequently, this imposes a limitation on its applicability to large target or query graphs.

DualMP has utilized DMPNNs, which are proven to have a complexity of $O(L_G(|E| + |V|))$. This method combines the target and query graph together before passing them into DMPNNs. Thus, the overall complexity of it can be rewritten as $O(L_G(|E_\mathcal{T}| + |E_\mathcal{P}| + |V_\mathcal{T}| + |V_\mathcal{P}|))$, and with a fixed number of DMPNN layers, we can observe the final complexity as $O(|E_\mathcal{T}| + |E_\mathcal{P}| + |V_\mathcal{T}| + |V_\mathcal{P}|)$. This is one of the lowest complexities of a neural-based method for solving the subgraph matching problem.

In comparison, the best exact baseline (GraphQL) has a time complexity of $O(|V_\mathcal{T}| \times |E_\mathcal{P}|) + |V_\mathcal{T}| \times |V_\mathcal{P}|$ for the simplest pattern. Additionally, other exact methods, including QuickSI and TurboISO, exhibit exponential time complexity for completing the matching process [10], [11]. From the above analysis, we can conclude that our approach exhibits one of the lowest time complexities, resulting in faster runtimes compared to almost other methods. Although DualMP has a lower theoretical time complexity, it demonstrates a higher runtime than our approach. This discrepancy arises because DMPNN layers must update edge features, which is more time-consuming than solely updating node features. However, due to the advantage of having the lowest time complexity, the runtime of DualMP increases the least when the query graph size grows (Figure 7).

Performance across real datasets: We showcase the benchmarking results of our xNeuSM and approximate approach in Figure 5. The outcomes unequivocally affirm that our approach achieves performance levels nearly on par with exact methods while surpassing the current SOTA performance of the approximate method across all tested real datasets. This underscores the versatility of our approach, making it applicable in diverse real-world scenarios such as pattern matching in social networks, identifying compounds with specific functions, and beyond.

C. CONFIDENCE ANALYSIS

In this section, we assess the model confidence in its predicted outputs to answer the **RQ2**. We demonstrate that our model maintains high-performance levels by elevating the output probability threshold for the subgraph matching task. Figure 6 depicts the relationship between confidence threshold and model performance. With a confidence threshold of 0.9, our model attains over 90% across all metrics in the testing datasets. This suggests that our xNeuSM exhibits robust confidence in its predictions.

D. SCALABILITY

To assess the scalability of xNeuSM and address research question **RQ3**, we evaluated the performance of all techniques using various real datasets with different levels of graph density, ranging from sparse to dense graphs.

- Vary $D(\mathcal{P})$: We divided queries into two subsets based on their average degree. The first subset, labelled ''dense'', included queries with a degree of three or higher. The second subset, labelled ''sparse'', encompassed queries with a degree less than 3.
- Vary $|V_\mathcal{P}|$: We partitioned the query set into four groups based on query size thresholds: $|V_\mathcal{P}| \leq 20$, $20 < |V_\mathcal{P}| \leq 40$, $40 < |V_\mathcal{P}| \leq 60$, and $60 < |V_\mathcal{P}|$.

We present our results in Figure 7, with runtime represented on a logarithmic scale. These results show that exact methods experience a significant increase in runtime when the number of query nodes is augmented. Specific methods, such as CECI or CFL, exhibit high sensitivity to the number of query nodes. In contrast, despite experiencing increased time requirements, our methods demonstrate relatively small increments due to the parallelizability of all operations using GPU. Consequently, our method proves to be more efficient in large-scale settings.

E. MATCHING EXPLANATION

Quantitative analysis: We conducted experiments on the matching explanation task to address **RQ4**. It is important to note that this task exclusively applies to known isomorphism pairs of (pattern, target). Non-isomorphic cases were deliberately excluded from our testing as they may not represent real-world use cases. Subgraph mapping is necessary when the isomorphism is established. Within this task, we computed the attention scores for all transaction nodes concerning each query node within the inter-attention branch of the final GLeMA layer. Subsequently, we organized the list of corresponding transaction nodes for each query node based on the computed attention scores. The effectiveness of subgraph alignment is evaluated and presented in Table 4.

Table 4 illustrates the superior performance of our method in the subgraph alignment task across diverse datasets, such as KKI, DHFR, DBLP-v1, and MSRC-21. However, the task becomes more challenging with large graphs that

TABLE 4. Performance of in subgraph aligning task.

Dataset	Top-1↑	Top-5↑	Top-10↑	MRR↑
KKI	0.9978	0.9999	0.9999	0.9987
COX2	0.2513	0.6259	0.8395	0.4273
COX2_MD	0.9481	0.9828	0.9881	0.9630
DHFR	0.9999	0.9999	0.9999	0.9999
DBLP-v1	0.9994	0.9999	0.9999	0.9996
MSRC-21	0.9994	0.9999	0.9999	0.9999

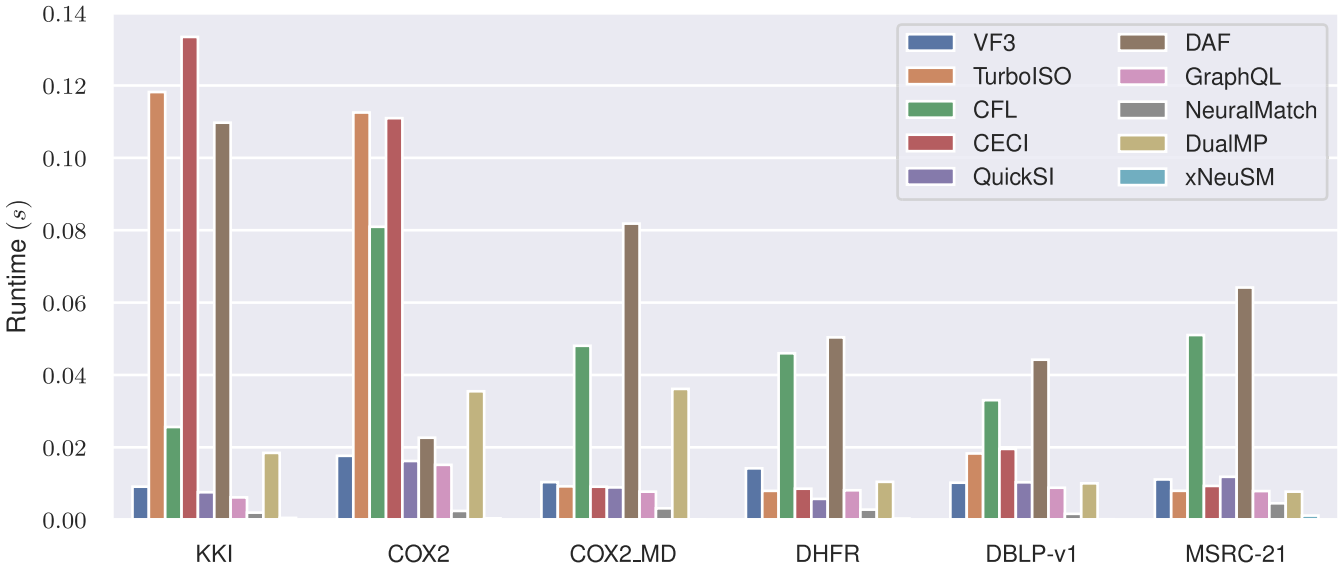


FIGURE 4. Execution time on subgraph matching task.

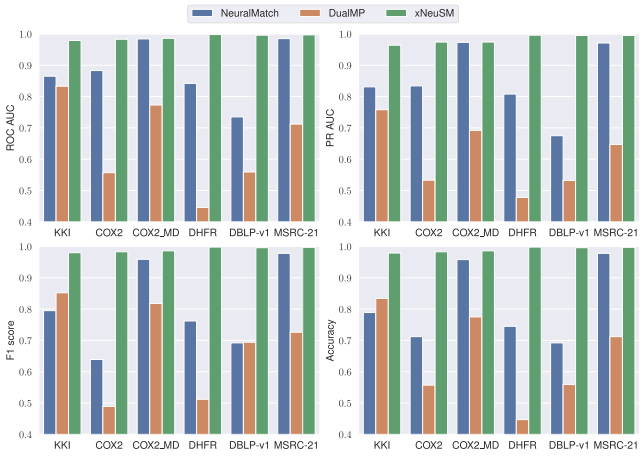


FIGURE 5. Performance comparison between neural match and xNeuSM.

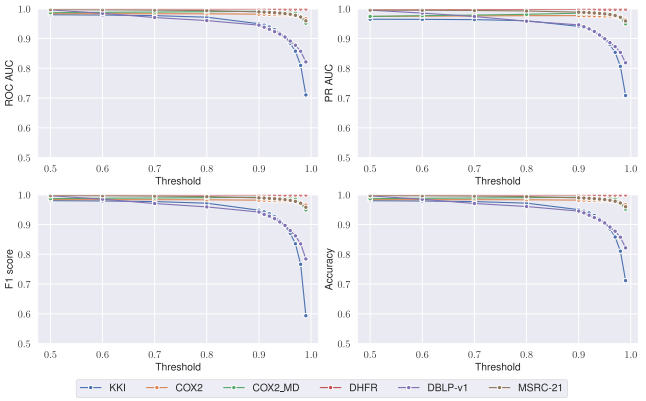


FIGURE 6. Relation between confidence threshold and model performance.

have a limited number of node labels, making it difficult to accurately retrieve subgraph mappings. This complexity arises from the exponential growth in the potential number of mappings.

Qualitative analysis: For the qualitative analysis of the matching explanation task, we have generated visualizations for two exemplary results derived from the KKI dataset. Specifically, Figure 8a illustrates an isomorphic case, while Figure 8b portrays a non-isomorphic case. A numerical label denotes each node within these figures, and nodes predicted to be aligned are color-coded uniformly. Our model demonstrates exceptional performance in the isomorphic case, accurately predicting all node mappings. Conversely, the model generates candidate mappings for all potential subgraphs within the pattern in the non-isomorphic case. For instance, in Figure 8b, the subgraph $154-54-129-152$ can be transformed into an isomorphic subgraph of the target within the pattern by removing the edge $154-152$.

F. ABLATION TESTING

In this section, we conduct an ablation study to better understand the interactions between the components in xNeuSM, which is the answer for **RQ5**. We evaluate the performance of our proposed framework using the KKI, COX2, and DBLP-v1 datasets against several variants:

- **Model architecture:**
  - *cross-only:* This configuration exclusively employs inter-connections between the graph and subgraph.
  - *intra-only:* This configuration solely relies on intra-connections within the graph and subgraph.
  - *both:* This configuration combines intra- and inter-connections of the graph and subgraph.

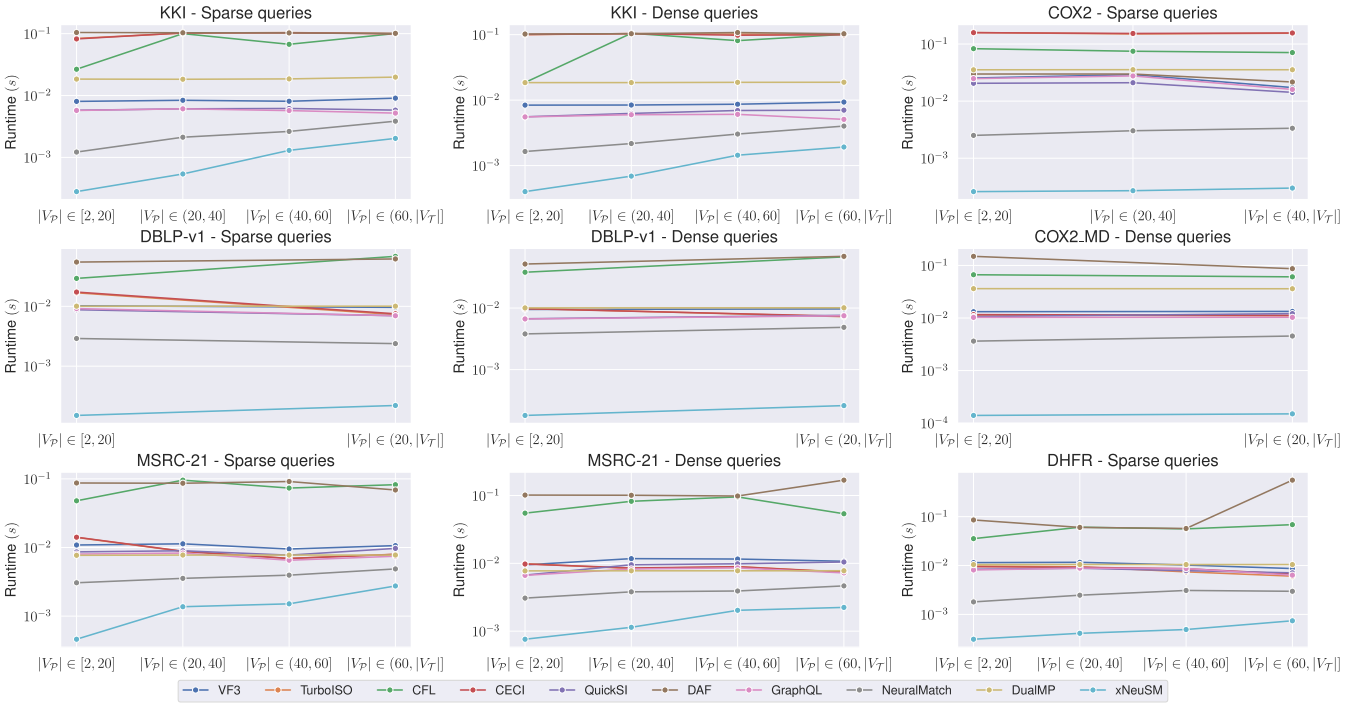
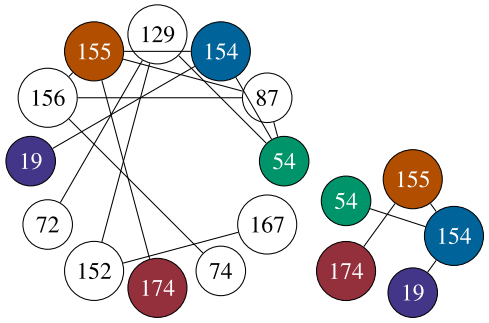
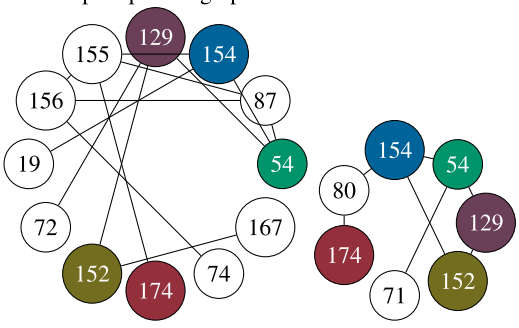


FIGURE 7. Runtime scalability analysis on six datasets including: KKI, COX2, COX2_MD, DHFR, DBLP-v1 and MSRC-21.



(a) Subgraph isomorphism case. From left to right: The target graph and the isomorphic pattern graph



(b) Subgraph non-isomorphism case. From left to right: The target graph and the non-isomorphic pattern graph

FIGURE 8. Examples of isomorphism and non-isomorphism cases resulted from our model in the KKI dataset.

- **Attention head:** We modify the base xNeuSM with 2 and 4 attention heads. These settings are used to understand the relationship between increasing model complexity (by increasing attention heads) and model performance.
- **Multi-hop mechanism:**
  - *1-hop:* Here, we replace the GLeMA layer with a standard 1-hop GAT layer.
  - *increasing-hop:* This configuration employs the GLeMA with a continuously increasing number of hops in the deeper layers ($K^{(L_G)} = L_G$).
  - *interleaved-hop:* This configuration uses the GLeMA with an interleaving-increasing number of hops. In this study, we use $K^{(L_G)} = 2L_G - 1$.

The results in Table 5 provide evidence that our proposed model architecture strikes a balanced and optimal trade-off between performance and computational efficiency. These findings demonstrate that modelling intra-connections is crucial for achieving outstanding performance. Combining inter- and intra-connections enhances the model's ability to distinguish unaligned nodes, resulting in higher performance than using intra- or inter-connections in isolation. Additionally, increasing the number of hops enables the model to effectively capture the global structure of graphs, further boosting performance. However, a continuous increase in hops leads to a slowdown in the model. The interleaved-hop strategy is the most suitable option for maintaining high performance while reducing computational time.

G. GENERALISATION ANALYSIS

In this section, we conduct experiments to demonstrate the generalization capabilities of xNeuSM in out-of-distribution settings and answer to **RQ6**. In these settings, we utilize the

TABLE 5. Impact of each model component in xNeuSM.

Model	KKI						COX2						DBLP-v1					
	Time↓	ROC↑	PR↑	F1↑	Acc↑	MRR↑	Time↓	ROC↑	PR↑	F1↑	Acc↑	MRR↑	Time↓	ROC↑	PR↑	F1↑	Acc↑	MRR↑
Cross-only 1-hop	0.56	0.979	0.959	0.979	0.979	**0.999**	0.12	0.967	0.946	0.968	0.967	0.306	0.07	**0.996**	**0.995**	0.996	**0.996**	0.996
Cross-only increasing-hop	0.60	0.977	0.956	0.977	0.977	0.996	0.12	0.962	0.931	0.964	0.962	0.191	0.09	0.980	0.964	0.981	0.980	0.983
Cross-only interleaved-hop	0.42	0.978	0.958	0.978	0.978	0.997	0.11	0.974	0.953	0.974	0.974	0.203	0.08	0.980	0.963	0.981	0.980	0.985
Intra-only 1-hop	0.49	0.611	0.578	0.485	0.612	–	0.11	0.472	0.498	0.007	0.472	–	0.09	0.640	0.598	0.579	0.640	–
Intra-only increasing-hop	0.40	0.628	0.593	0.515	0.628	–	0.11	0.457	0.499	0.003	0.458	–	0.13	0.643	0.593	0.634	0.643	–
Intra-only interleaved-hop	0.42	0.669	0.626	0.602	0.670	–	0.12	0.491	0.495	0.180	0.491	–	0.09	0.618	0.576	0.573	0.618	–
Both 1-hop	0.62	0.968	0.939	0.968	0.968	**0.999**	0.12	0.962	0.950	0.961	0.962	0.177	0.09	**0.996**	0.992	0.996	**0.996**	0.995
Both increasing-hop	0.70	**0.980**	0.963	**0.980**	**0.980**	**0.999**	0.39	0.972	0.961	0.972	0.972	0.298	0.10	0.918	0.910	0.912	0.918	0.989
Both interleaved-hop	0.51	0.979	**0.964**	**0.980**	0.979	0.998	0.38	**0.983**	**0.974**	**0.984**	**0.983**	**0.427**	0.13	**0.996**	**0.995**	**0.997**	**0.996**	**0.999**
With 2 attention heads	0.86	0.956	0.935	0.954	0.957	0.998	0.49	0.954	0.925	0.955	0.954	0.298	0.17	0.976	0.964	0.975	0.976	**0.999**
With 4 attention heads	1.19	0.938	0.923	0.936	0.937	**0.999**	0.63	0.907	0.892	0.900	0.907	0.215	0.28	0.986	0.984	0.985	0.986	**0.999**

TABLE 6. ROC AUC of out-distribution settings. For each dataset, the model trained on a different dataset that achieved the highest ROC AUC is in [*italic*].

Train \ Test	KKI	COX2	COX2_MD	DHFR	DBLP-v1	MSRC-21
KKI	**0.979**	0.634	0.499	*0.970*	*0.923*	*0.928*
COX2	0.500	**0.983**	0.500	0.500	0.501	0.500
COX2_MD	0.534	0.412	**0.986**	0.499	0.565	0.497
DHFR	0.547	0.797	0.499	**0.998**	0.758	0.668
DBLP-v1	0.502	*0.883*	0.491	0.689	**0.996**	0.505
MSRC-21	*0.863*	0.539	*0.604*	0.961	0.712	**0.997**

TABLE 7. Performance of xNeuSM directed subgraph matching and matching explanation.

Dataset	ROC↑	PR↑	F1↑	Acc↑	Top-1↑	Top-2↑	Top-10↑	MRR↑
KKI	0.975	0.953	0.975	0.975	0.996	0.999	0.999	0.998
COX2	0.947	0.908	0.949	0.947	0.103	0.396	0.640	0.261
COX2_MD	0.989	0.979	0.989	0.989	0.999	0.999	0.999	0.999
DHFR	0.969	0.944	0.970	0.969	0.999	0.999	0.999	0.999
DBLP-v1	0.960	0.940	0.960	0.960	0.745	0.996	0.999	0.866
MSRC-21	0.988	0.977	0.988	0.988	0.999	0.999	0.999	0.999

model trained on one dataset to test the others with the same datasets as in the previous experiments. The testing results are presented in Table 6. Upon examination of Table 6, the following observations can be made:

- A model trained on a dataset with a large $|\Sigma|$ exhibits generalization to datasets with smaller $|\Sigma|$ (trained on KKI and tested on DHFR, DBLP-v1, MSRC-21).
- A model trained on a dataset with a lower incidence of duplicated node graphs exhibits poor generalization to datasets characterized by a higher frequency of duplicated node graphs (no model trained on other datasets well generalizes to COX2).
- Furthermore, a model trained on dense graphs can generalize to datasets with sparser graphs (trained on MSRC-21 and tested on DHFR, DBLP-v1).

These results collectively demonstrate the strong generalization abilities of our model, especially when the model is trained on sufficiently large datasets.

H. DIRECTED SUBGRAPH MATCHING AND EXPLANATION

To answer the final **RQ7**, we assessed our xNeuSM using directed graphs. We utilized the same datasets as those in previous experiments to accomplish this. We converted all edges within these datasets into directed edges, designating the tail node as the one with a smaller label and the head node as the one with a larger label. Subsequently, we present the results of our evaluation within the context of both subgraph matching and matching explanation tasks in Table 7. The results in Table 7 demonstrate the effectiveness of our proposed method, even with directed graphs.

X. CONCLUSION

In this study, we proposed a novel framework called xNeuSM for explainable neural subgraph matching. xNeuSM aims to address the limitations of previous neural-based approaches, which lack interpretability while achieving superior performance compared to existing approximate and exact algorithms. We introduced key contributions, including the Graph Learnable Multi-hop Attention Networks and a multi-task learning framework to optimize matching and matching explanation tasks jointly. Theoretical justifications were provided to analyze the approximation error of multi-hop attention and prove the correctness of learning node-specific attention decays.

Extensive experimental evaluations on real-world datasets demonstrated that xNeuSM achieves substantial improvements over state-of-the-art techniques in both runtime and accuracy for subgraph matching. Its capability is further manifested in the precise identification of node mappings, as evidenced by the matching explanation results. Further ablation studies validated the effectiveness of individual components in xNeuSM's architecture. This work also explored the generalizability, scalability and applicability of xNeuSM to directed graphs and inductive settings. Overall, xNeuSM achieves the dual objectives of superior performance and interpretability, making it a practical solution for a wide range of real-world tasks involving subgraph matching and pattern analysis in large graphs.

There are several promising directions for future work. Firstly, xNeuSM can be extended to handle inexact subgraph matching. Secondly, integrating advanced GNN modules may further boost xNeuSM's representation power. Lastly, applications of xNeuSM to real-world domains like drug discovery and network alignment could be explored. In summary, xNeuSM presents a significant step towards building interpretable and scalable neural solutions for graph-related problems.

REFERENCES

[1] S. Sahu, A. Mhedhbi, S. Salihoglu, J. Lin, and M. T. Özsu, "The ubiquity of large graphs and surprising challenges of graph processing: Extended survey," *VLDB J.*, vol. 29, nos. 2–3, pp. 595–618, May 2020.

[2] S. P. Nandanoori, S. Guan, S. Kundu, S. Pal, K. Agarwal, Y. Wu, and S. Choudhury, "Graph neural network and Koopman models for learning networked dynamics: A comparative study on power grid transients prediction," *IEEE Access*, vol. 10, pp. 32337–32349, 2022.

[3] T. Ma, S. Yu, J. Cao, Y. Tian, A. Al-Dhelaan, and M. Al-Rodhaan, "A comparative study of subgraph matching isomorphic methods in social networks," *IEEE Access*, vol. 6, pp. 66621–66631, 2018.

[4] Y. Zheng, Y. Lin, L. Zhao, T. Wu, D. Jin, and Y. Li, "Spatial planning of urban communities via deep reinforcement learning," *Nature Comput. Sci.*, vol. 3, no. 9, pp. 748–762, Sep. 2023.

[5] Y. Peng, Y. Lin, X.-Y. Jing, H. Zhang, Y. Huang, and G. S. Luo, "Enhanced graph isomorphism network for molecular ADMET properties prediction," *IEEE Access*, vol. 8, pp. 168344–168360, 2020.

[6] Y. Zhang, Q. Yao, L. Yue, X. Wu, Z. Zhang, Z. Lin, and Y. Zheng, "Emerging drug interaction prediction enabled by a flow-based graph neural network with biomedical network," *Nature Comput. Sci.*, vol. 3, no. 12, pp. 1023–1033, Dec. 2023.

[7] O. Zhang, T. Wang, G. Weng, D. Jiang, N. Wang, X. Wang, H. Zhao, J. Wu, E. Wang, G. Chen, Y. Deng, P. Pan, Y. Kang, C.-Y. Hsieh, and T. Hou, "Learning on topological surface and geometric structure for 3D molecular generation," *Nature Comput. Sci.*, vol. 3, no. 10, pp. 849–859, Oct. 2023.

[8] I. Roy, V. S. B. R. Velugoti, S. Chakrabarti, and A. De, "Interpretable neural subgraph matching for graph retrieval," in *Proc. AAAI Conf. Artif. Intell.*, vol. 36, 2022, pp. 8115–8123.

[9] J. Li, D. Xiao, K. Li, and J. Li, "Graph matching for marker labeling and missing marker reconstruction with bone constraint by LSTM in optical motion capture," *IEEE Access*, vol. 9, pp. 34868–34881, 2021.

[10] H. Shang, Y. Zhang, X. Lin, and J. X. Yu, "Taming verification hardness: An efficient algorithm for testing subgraph isomorphism," *Proc. VLDB Endowment*, vol. 1, no. 1, pp. 364–375, Aug. 2008.

[11] W.-S. Han, J. Lee, and J.-H. Lee, "Turbo$_{iso}$: Towards ultrafast and robust subgraph isomorphism search in large graph databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Jun. 2013, pp. 337–348.

[12] F. Bi, L. Chang, X. Lin, L. Qin, and W. Zhang, "Efficient subgraph matching by postponing Cartesian products," in *Proc. Int. Conf. Manage. Data*, Jun. 2016, pp. 1199–1214.

[13] B. Bhattarai, H. Liu, and H. H. Huang, "CECI: Compact embedding cluster index for scalable subgraph matching," in *Proc. Int. Conf. Manage. Data*. New York, NY, USA: ACM, Jun. 2019, pp. 1447–1462.

[14] M. Han, H. Kim, G. Gu, K. Park, and W.-S. Han, "Efficient subgraph matching: Harmonizing dynamic programming, adaptive matching order, and failing set together," in *Proc. Int. Conf. Manage. Data*. New York, NY, USA: ACM, Jun. 2019, pp. 1429–1446.

[15] R. Ying, Z. Lou, J. You, C. Wen, A. Canedo, and J. Leskovec, "Neural subgraph matching," 2020, *arXiv:2007.03092*.

[16] X. Liu and Y. Song, "Graph convolutional networks with dual message passing for subgraph isomorphism counting and matching," in *Proc. AAAI Conf. Artif. Intell.*, vol. 36, Jun. 2022, pp. 7594–7602.

[17] H. He and A. K. Singh, "Graphs-at-a-time: Query language and access methods for graph databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*. New York, NY, USA: ACM, Jun. 2008, pp. 405–418.

[18] K. Xu, L. Wang, M. Yu, Y. Feng, Y. Song, Z. Wang, and D. Yu, "Cross-lingual knowledge graph alignment via graph matching neural network," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, Florence, Italy, A. Korhonen, D. Traum, and L. Màrquez, Eds., Stroudsburg, PA, USA: Association for Computational Linguistics, Jul. 2019, pp. 3156–3161.

[19] J. Xu, T. L. Wickramarathne, and N. V. Chawla, "Representing higher-order dependencies in networks," *Sci. Adv.*, vol. 2, no. 5, May 2016, Art. no. e1600028.

[20] G. Wang, R. Ying, J. Huang, and J. Leskovec, "Multi-hop attention graph neural networks," in *Proc. 30th Int. Joint Conf. Artif. Intell. (IJCAI)*, Z.-H. Zhou, Ed., Aug. 2021, pp. 3089–3096.

[21] G. Ren, L. Diao, F. Guo, and T. Hong, "A co-attention based multi-modal fusion network for review helpfulness prediction," *Inf. Process. Manage.*, vol. 61, no. 1, Jan. 2024, Art. no. 103573.

[22] Y. Zhao, T. Xia, Y. Jiang, and Y. Tian, "Enhancing inter-sentence attention for semantic textual similarity," *Inf. Process. Manage.*, vol. 61, no. 1, Jan. 2024, Art. no. 103535.

[23] J. Ma, D. Li, H. Zhu, C. Li, Q. Zhang, and Y. Qiao, "GAFM: A knowledge graph completion method based on graph attention faded mechanism," *Inf. Process. Manage.*, vol. 59, no. 5, Sep. 2022, Art. no. 103004.

[24] M. M. SysŁ;o, "The subgraph isomorphism problem for outerplanar graphs," *Theor. Comput. Sci.*, vol. 17, no. 1, pp. 91–97, 1982.

[25] H. Kim, Y. Choi, K. Park, X. Lin, S.-H. Hong, and W.-S. Han, "Versatile equivalences: Speeding up subgraph query processing and subgraph matching," in *Proc. Int. Conf. Manage. Data*. New York, NY, USA: ACM, Jun. 2021, pp. 925–937.

[26] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman, 1990.

[27] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2008.

[28] L. Wang, W. Yuan, L. Zeng, J. Xu, Y. Mo, X. Zhao, and L. Peng, "Dementia analysis from functional connectivity network with graph neural networks," *Inf. Process. Manage.*, vol. 59, no. 3, May 2022, Art. no. 102901.

[29] G. Lu, J. Li, and J. Wei, "Aspect sentiment analysis with heterogeneous graph neural networks," *Inf. Process. Manage.*, vol. 59, no. 4, Jul. 2022, Art. no. 102953.

[30] O. Amine and M. Mestari, "Graph oriented attention networks," *IEEE Access*, vol. 12, pp. 47057–47067, 2024.

[31] R. Jovanovic, M. Palk, S. Bayhan, and S. Voss, "Applying graph neural networks to the decision version of graph combinatorial optimization problems," *IEEE Access*, vol. 11, pp. 38534–38547, 2023.

[32] Z. Zhang and W. S. Lee, "Deep graphical feature learning for the feature matching problem," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 5086–5095.

[33] Q. Huang, M. Yamada, Y. Tian, D. Singh, and Y. Chang, "GraphLIME: Local interpretable model explanations for graph neural networks," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 7, pp. 6968–6972, Jul. 2023.

[34] F. Wu, S. Li, X. Jin, Y. Jiang, D. Radev, Z. Niu, and S. Z. Li, "Rethinking explaining graph neural networks via non-parametric subgraph matching," in *Proc. 40th Int. Conf. Mach. Learn.*, vol. 202, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., 2023, pp. 37511–37523.

[35] Y. Zhang, W. K. Cheung, Q. Liu, G. Wang, L. Yang, and L. Liu, "Towards explaining graph neural networks via preserving prediction ranking and structural dependency," *Inf. Process. Manage.*, vol. 61, no. 2, Mar. 2024, Art. no. 103571.

[36] H. Yuan, J. Tang, X. Hu, and S. Ji, "XGNN: Towards model-level explanations of graph neural networks," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. New York, NY, USA: ACM, Aug. 2020, pp. 430–438.

[37] M. N. Vu and M. T. Thai, "PGM-explainer: Probabilistic graphical model explanations for graph neural networks," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.* Red Hook, NY, USA: Curran Associates, 2020, pp. 12225–12235.

[38] D. Yang, Y. Ge, T. Nguyen, D. Molitor, J. D. Moorman, and A. L. Bertozzi, "Structural equivalence in subgraph matching," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 4, pp. 1846–1862, Jul. 2023.

[39] S. Sun and Q. Luo, "Subgraph matching with effective matching order and indexing," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 1, pp. 491–505, Jan. 2022.

[40] Y. Sun, G. Li, J. Du, B. Ning, and H. Chen, "A subgraph matching algorithm based on subgraph index for knowledge graph," *Frontiers Comput. Sci.*, vol. 16, no. 3, Jun. 2022, Art. no. 163606.

[41] D. L. Sussman, Y. Park, C. E. Priebe, and V. Lyzinski, "Matched filters for noisy induced subgraph detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 11, pp. 2887–2900, Nov. 2020.

[42] J. Jiménez-Luna, F. Grisoni, and G. Schneider, "Drug discovery with explainable artificial intelligence," *Nature Mach. Intell.*, vol. 2, no. 10, pp. 573–584, Oct. 2020.

[43] Y. Bai, H. Ding, Y. Sun, and W. Wang, "Convolutional set matching for graph similarity," in *Proc. Neural Inf. Process. Syst. Relational Represent. Learn. Workshop*, 2018, pp. 1–14.

[44] J. Lim, S. Ryu, K. Park, Y. J. Choe, J. Ham, and W. Y. Kim, "Predicting Drug–Target interaction using a novel graph neural network with 3D structure-embedded graph representation," *J. Chem. Inf. Model.*, vol. 59, no. 9, pp. 3981–3988, Sep. 2019.

[45] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process.* Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 1412–1421.
[46] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–12.
[47] J. Gasteiger, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized PageRank," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–15.
[48] P. Lofgren, "Efficient algorithms for personalized PageRank," Ph.D. thesis, Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, 2015.
[49] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, A. Moschitti, B. Pang, and W. Daelemans, Eds., Doha, Qatar: Association for Computational Linguistics, 2014, pp. 1724–1734.
[50] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann, "TUDataset: A collection of benchmark datasets for learning with graphs," in *Proc. ICML Workshop Graph Represent. Learn. Beyond (GRL)*, 2020, pp. 1–11.
[51] V. Carletti, P. Foggia, A. Saggese, and M. Vento, "Introducing VF3: A new algorithm for subgraph isomorphism," in *Graph-Based Representations in Pattern Recognition*, P. Foggia, C.-L. Liu, and M. Vento, Eds., Cham, Switzerland: Springer, 2017, pp. 128–139.
[52] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, "A (sub)graph isomorphism algorithm for matching large graphs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 10, pp. 1367–1372, Oct. 2004.

DUC Q. NGUYEN received the B.Eng. degree in computer science from Ho Chi Minh City University of Technology—VNU-HCM, in 2022, where he is currently pursuing the master's degree.

Since 2022, he has been a Research Assistant with the URA Laboratory. Since April 2023, he has also been a Teaching Fellow with the Faculty of Computer Science and Engineering, VNU-HCM. His research interests include artificial intelligence, computational biology, and graph representation learning. His biggest ambition is to make human life better and better using his talent and experiences.

THANH TOAN NGUYEN received the bachelor's and master's degrees from Ho Chi Minh City University of Technology and the Ph.D. degree from Griffith University, Australia. He has publications in prestigious journals and conferences, such as TIST, *Information Sciences*, ESWA, KBS, TOIS, and IJCNN. His main research interests include database systems, machine learning, decision support systems, and software design.

JUN JO received the Ph.D. degree from the University of Sydney. He is currently a pioneering figure in robotics and artificial intelligence. As the Chair of the International Robot Olympiad Committee and the President of Australian Robotics Association, he spearhead initiatives fostering innovation and education in robotics worldwide. At Griffith University, he is also the Director of the Robotics Laboratory and has been holding the title of an Associate Professor with the School of ICT, since 1996. He was a Postdoctoral Research Fellow with the University of Sydney. With a passion for sustainable development, he actively contribute to the advancement of sustainable cities and communities through his expertise in artificial intelligence, information systems, and software engineering. He continues to drive progress at the intersection of technology and societal needs, leaving an indelible mark on the field of robotics and beyond.

FLORENT POUX received the Ph.D. degree in sciences.

He focuses mainly on research about 3D data acquisition (laser scanning and photogrammetry), automation (object recognition and modeling) and integration in immersive processes (virtual and augmented reality). He is recognized as an outstanding researcher through the ISPRS Jack Dangermond 2019 Award. He bridges high-level research & knowledge transmission as an Adjunct Professor in 3D geodata with the University of Liège, a Mentor in data sciences & machine learning (OpenClassrooms), and a 3D Consultant. During his young academic and scientific career, he has given dozens of tech talks, served on the scientific and program committee of several international conferences and assisted as an editor and a reviewer for leading international GIS and automation journals. His activities aim at transmitting knowledge and solving automation problematics, through various form of communication and developments. He relies on his expertise and a close group of skilled collaborators that share a vision of global cooperation toward the advancement of technological science for a better society.

SHIKHA ANIRBAN is currently an Enthusiastic Academician and a Researcher with a profound interest in advancing knowledge in the field of information technology. Her research journey is marked by a relentless pursuit of innovative solutions, particularly in the area of graph databases, machine learning, and related technologies. Her research interests include graph databases, data science, data engineering, query optimization, information retrieval, machine learning, and artificial intelligence. Her teaching expertise lies in database analysis and design, artificial intelligence, machine learning, and computer programming. She believes in bridging the gap between theory and practice by integrating a diverse range of teaching methodologies, including active learning, real-world applications, and collaborative projects. Her aim is to cultivate critical thinking skills and a deep understanding of the subject matter among her students.

THO T. QUAN received the B.Eng. degree in information technology from Ho Chi Minh City University of Technology (HCMUT), Vietnam, in 1998, and the Ph.D. degree from Nanyang Technological University, Singapore, in 2006. He is currently an Associate Professor with the Faculty of Computer Science and Engineering, HCMUT. He is also the Acting Dean of the Faculty of Computer Science and Engineering. His current research interests include artificial intelligence, natural language processing, intelligent systems, and formal methods.