VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY

NGUYEN QUANG DUC

OPTIMIZING SUBGRAPH ISOMORPHISM PREDICTION MODELS WITH APPLICATION ORIENTATION TO THE DRUG DESIGN PROCESS

Major: COMPUTER SCIENCE
Major code: 8480101

MASTER'S THESIS

HO CHI MINH CITY, January 2025

THIS THESIS IS COMPLETED AT
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY – VNU-HCM

Supervisor: Assoc. Prof. Quan Thanh Tho

Supervisor signature: _____

Examiner 1: Dr. Le Thanh Van

Examiner signature: _____

Examiner 2: Dr. Ha Thi Thanh Huong

Examiner signature: _____

This master's thesis is defended at HCM City University of Technology, VNU-HCM City on January 15^{th}, 2025.

Master's Thesis Committee:

1. Chairman: Assoc. Prof. Le Hong Trang
2. Secretary: Dr. Le Trong Nhan
3. Member: Assoc. Prof. Nguyen Thi Thuy Loan
4. Reviewer 1: Dr. Le Thanh Van
5. Reviewer 2: Dr. Ha Thi Thanh Huong

Approval of the Chairman of Master's Thesis Committee and Dean of Faculty of Computer Science and Engineering after the thesis being corrected (If any).

CHAIRMAN OF THESIS COMMITTEE	**HEAD OF FACULTY OF COMPUTER SCIENCE AND ENGINEERING**

VIETNAM NATIONAL UNIVERSITY - HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY

SOCIALIST REPUBLIC OF VIETNAM
Independence – Freedom - Happiness

THE TASK SHEET OF MASTER'S THESIS

Full name: Nguyễn Quang Đức
Date of birth: 24 / 12 / 2000
Major: Computer Science

Student ID: 2270664
Place of birth: An Giang
Major ID: 8480101

I. THESIS TITLE (In Vietnames): Tối ưu hóa các mô hình dự đoán đẳng cấu đồ thị con với định hướng áp dụng cho quy trình thiết kế thuốc

II. THESIS TITLE (In English): Optimizing subgraph isomorphism prediction models with application orientation to the drug design process

III. TASKS AND CONTENTS:

1. Investigating the problem of subgraph isomorphism testing and its applications in drug design
2. Finding and experimenting with available approaches to solve the subgraph isomorphism problem
3. Proposing a neural-based approach to solve the problem of subgraph isomorphism and providing theoretical justification (if possible)
4. Comprehensively evaluating the proposed approach in terms of accuracy, efficiency, scalability, and generalization

IV. THESIS START DAY: 06/02/2023

V. THESIS COMPLETION DAY: 23/12/2024

VI. SUPERVISOR: Associate Professor Quan Thanh Tho

Ho Chi Minh City, March 03rd 2025

SUPERVISOR
(Full name and signature)

CHAIR OF PROGRAM COMMITTEE
(Full name and signature)

DEAN OF FACULTY OF
COMPUTER SCIENCE AND ENGINEERING
(Full name and signature)

Acknowledgements

I wish to extend my heartfelt appreciation to Prof. Quan Thanh Tho, my supervisor, whose unwavering patience, guidance, and support have been invaluable throughout my academic journey. His wealth of knowledge and meticulous editing have significantly contributed to my growth as a student. I am profoundly grateful for his belief in me, starting from my early days as a student and continuing steadfastly over the years.

I am deeply thankful to Dr. Nguyen Thanh Toan, a predecessor whose advice and encouragement have been invaluable to me. I am also indebted to the Vingroup Innovation Foundation (VINIF) for their support through the Master, PhD Scholarship Programme, under code VINIF.2022.ThS.023.

My family merits boundless gratitude for their enduring love, sacrifices, and unwavering support, which have been the cornerstone of my motivation and confidence. Their belief in me has been instrumental in my accomplishments and success. Words cannot capture the depth of my love and appreciation for my family.

Abstract

In the domain of drug design, numerous issues necessitate the identification of specific structural patterns within molecules. This quandary aligns with the computational problems of subgraph isomorphism or subgraph matching in computer science. Subgraph matching poses a multifaceted challenge with extensive applications, spanning database systems, biochemistry, and cognitive science. This task involves confirming the existence of a specified query graph within a larger target graph. Traditional algorithms for this task yield precise outcomes but struggle with scalability when handling large graph instances due to the NP-complete nature of the problem, limiting their practical use. Recent neural network-based approaches offer more scalable solutions but often lack clear interpretations of node correspondences.

This thesis introduces a novel approach named xNeuSM: E**x**plainable **Neu**ral **S**ubgraph **M**atching, addressing the above limitations. It presents Graph Learnable Multi-hop Attention Networks (GLeMA), which dynamically learn parameters governing attention decay across nodes over multiple hops instead of relying on fixed hyperparameters. Theoretical analysis establishes error margins for GLeMA's multi-hop attention approximation relative to the number of hops. Furthermore, it is proven that learning distinct attention decay factors for each node accurately approximates multi-hop attention.

Empirical evaluation on real-world datasets including chemistry and bioinformatic datasets demonstrates that xNeuSM significantly enhances prediction accuracy, showing improvements of up to 34% compared to approximate baselines. Impressively, it also delivers query times at least seven times faster than exact algorithms. These results suggest that the proposed approach is feasible to solve more complex problems in the drug design process. The work presented in this thesis has been published in the IEEE Access journal [1].

Tóm Tắt

Trong lĩnh vực thiết kế thuốc, nhiều vấn đề đòi hỏi việc xác định các mẫu cấu trúc cụ thể trong phân tử. Bài toán này tương ứng với các vấn đề đẳng cấu đồ thị con hoặc khớp mẫu đồ thị con trong khoa học máy tính. Bài toán khớp mẫu đồ thị con là một thách thức đa chiều với nhiều ứng dụng quan trọng, bao gồm hệ thống cơ sở dữ liệu, sinh hóa và khoa học nhận thức. Nhiệm vụ này liên quan đến việc xác minh sự tồn tại của một đồ thị truy vấn cụ thể trong một đồ thị mục tiêu lớn hơn. Các thuật toán truyền thống cho bài toán này thường cho kết quả chính xác nhưng gặp khó khăn về khả năng mở rộng khi xử lý các đồ thị lớn do tính chất NP-đầy đủ của bài toán, làm hạn chế tính ứng dụng thực tế. Các phương pháp dựa trên mạng nơ-ron gần đây cung cấp giải pháp có khả năng mở rộng tốt hơn nhưng thường thiếu khả năng giải thích rõ ràng về sự tương ứng giữa các nút giữa đồ thị truy vấn và đồ thị mục tiêu.

Luận án này giới thiệu một phương pháp mới có tên xNeuSM: E**x**plainable **Neu**ral **S**ubgraph **M**atching, nhằm khắc phục các hạn chế trên. Phương pháp này đề xuất Mạng chú ý đa bước học được trên đồ thị (Graph Learnable Multi-hop Attention Networks - GLeMA), cho phép mô hình học các tham số điều chỉnh sự suy giảm chú ý (attention decay) giữa các nút qua nhiều bước thay vì dựa vào các siêu tham số cố định. Phân tích lý thuyết thiết lập các biên độ lỗi cho phép xấp xỉ chú ý đa bước (multi-hop attention approximation) của GLeMA theo số bước. Hơn nữa, nghiên cứu chứng minh rằng việc học các hệ số suy giảm chú ý riêng biệt cho từng nút giúp xấp xỉ chính xác hơn sự chú ý đa bước.

Đánh giá thực nghiệm trên các tập dữ liệu thực tế trong hóa học và tin sinh học cho thấy xNeuSM cải thiện đáng kể độ chính xác dự đoán, với mức tăng lên đến 34% so với các phương pháp xấp xỉ khác. Đặc biệt, thời gian truy vấn cũng nhanh hơn ít nhất bảy lần so với các thuật toán chính xác. Những kết quả này cho thấy phương pháp đề xuất có tiềm năng giải quyết các bài toán phức tạp hơn trong quy trình thiết kế thuốc. Công trình này đã được công bố trên tạp chí IEEE Access [1].

Declaration of Authenticity

I declare that this research is my own work, conducted under the supervision and guidance of Assoc. Prof. Quan Thanh Tho. The result of our research is legitimate and has not been published in any form prior to this. All materials used in this research were collected by myself from various sources and are appropriately listed in the references section.

In addition, within this research, I also used the results of several other authors and organizations. They have all been aptly referenced.

In any case of plagiarism, I stand by my actions and will be responsible for it. Ho Chi Minh City University of Technology - Vietnam National University Ho Chi Minh City therefore is not responsible for any copyright infringements conducted within our research.

Ho Chi Minh City, February 2025
Author

Nguyen Quang Duc

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement

The drug design process involves multifaceted tasks, including predicting molecular properties, verifying specific functional groups within designed molecules, forecasting interactions among multiple compounds, etc. In the early stages, verifying functional groups becomes imperative to assess whether a molecule aligns with predefined criteria for consideration in subsequent phases [2] (see Figure 1.1). This verification challenge aligns with the problem of testing subgraph isomorphism in computer science [2].

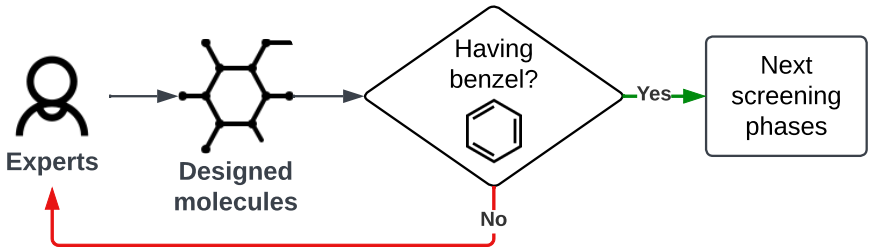


Figure 1.1: Example of early screening process in drug design. Experts initially design molecules, which are then evaluated for the presence of a particular feature (e.g., a benzene ring). If the molecule contains a benzene ring, it proceeds to the next screening phase. Otherwise, it is sent back to experts for further refinement. This iterative approach ensures that only molecules meeting specific structural criteria advance in the screening pipeline.

In recent years, there has been a notable surge in efforts directed toward devising viable approaches for NP-hard graph problems, especially subgraph isomorphism testing. This wave of interest has been motivated by the abundance of diverse graph data in the public domain [3]. 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 testing or subgraph matching involves verifying whether a specified query graph exhibits isomorphism within a substructure of a larger target graph. Despite its inherent NP-completeness, this problem holds pivotal importance not only in the drug design process but also across varied domains, including physics [4], social network analysis [5], bioinformatics [6]–[8], graph retrieval [9], and computer vision [10]. This momentum has spurred researchers to craft scalable and efficient algorithms tailored specifically to analyze vast graphs akin to those prevalent in social and biological networks. Over the past decades, a multitude of scholarly endeavors have culminated in a diverse spectrum of practical solutions, comprising a range of algorithms [11]–[17].

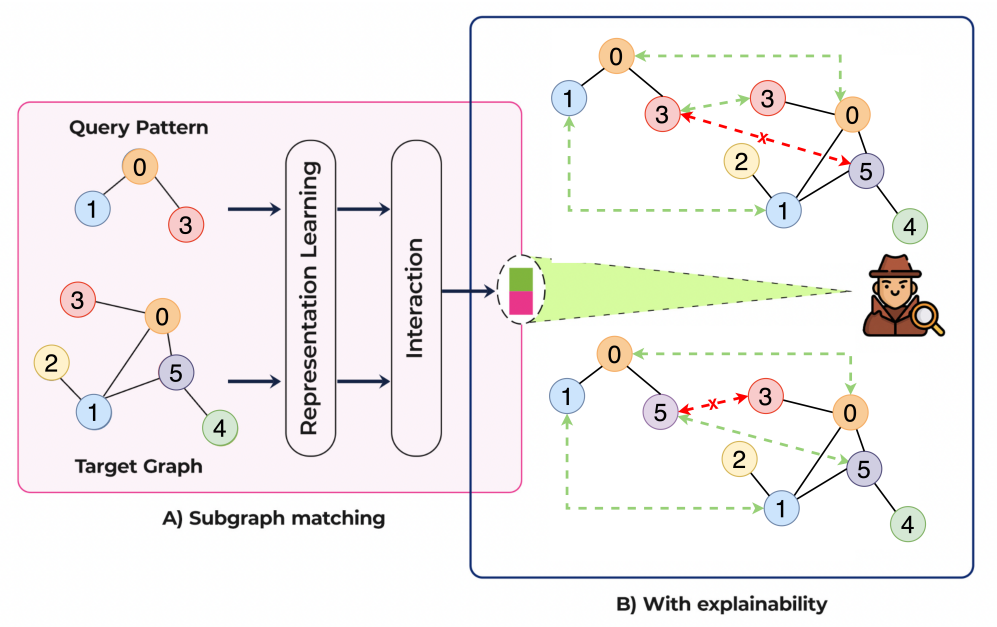


Figure 1.2: A common subgraph matching pipeline. The query pattern and target graph are represented appropriately. Then the matching algorithm will be applied. The pipeline results should contain binary results indicating whether the pattern is subgraph isomorphic to the target graph or not and the node mapping between those two graphs.

The conventional methodologies [12], [18] predominantly rely on precise combinatorial search algorithms. While the exact computation of subgraph matching delivers accurate outcomes for both *subgraph matching* (Figure 1.2-A) and the elucidation of *matching explanations* (facilitating node-to-node correspondences across graphs) as depicted in (Figure 1.2-B), their scalability is constrained when dealing with larger query pattern sizes owing to the NP-complete nature of the problem. Recent endeavors [13]–[15] have aimed to bolster scalability by devising optimized matching orders and formulating robust filtering strategies aimed at reducing the number of subgraph candidates to be considered. Despite extending matching capabilities to larger target graphs, these efforts are still bound by limitations in query size, accommodating only a few tens of nodes. This level of scalability falls short of the practical requirements demanded by real-world applications.

To address the scalability challenges posed by exact approaches, recent neural-based methodologies [16], [17], [19] have emerged, aiming for a trade-off between computational 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 of these methodologies lies in employing graph neural networks (GNNs) to learn the matching function. However, these learning algorithms heavily rely on first-order dependencies within the GNN architecture's layers, restricting the receptive field of a single GNN layer to immediate one-hop network neighbors in the graph, as emphasized in [16], [17]. 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 [20]. This contrasts with the earlier assumption of solely first-order network relationships. Oversimplifying this assumption might overlook the generalizability of these methods in capturing diverse patterns, potentially leading to a substantial performance decline across different domains [16], [19].

Striking a balance between efficiency (**O.1**) and explanatory power (**O.2**) within a subgraph matching algorithm remains a desirable yet challenging endeavor. Inspired by the success of Graph Multi-hop Attention Networks (GMA) [21], this work adopts the GMA concept to concurrently address these dual objectives.

Diverging from existing multi-hop attention mechanisms that use a fixed attention decay factor for all nodes, this study introduces Graph Learnable Multi-hop Attention Networks (GLeMa), allowing the adaptive learning of node-specific attention decay factors. GLeMa parameterizes unique attention decay factors for each node, governing their influence across neighborhoods during multi-hop message propagation. Theoretical analysis establishes error bounds for GLeMa's modeling of multi-hop attention relative to the number of hops. Additionally, it formally verifies that learning node-specific attention decays enables GLeMa to accurately capture relational patterns in graph-structured data.

By integrating the learnable multi-hop attention mechanism, GLeMA achieves a notable level of generalization within the data graph while upholding efficiency (**O.1**). However, simultaneously achieving both **O.1** and **O.2** remains a challenge. Unlike prior neural-based approaches [16] that directly learn from separate adjacency matrices for pattern and target graphs, our novel unified proxy inputs facilitate comprehensive capture of intra- and inter-relations between the pattern and target graphs, enhancing explicit node alignment (**O.2**). Moreover, we optimize subgraph matching and matching explanation tasks concurrently in an end-to-end multi-task manner, fostering a mutually reinforcing synergy between both objectives, thus enhancing the framework's overall effectiveness and efficiency.

1.2 Goals

In this thesis, I develop an approach named xNeuSM (Explainable Neural Subgraph Matching with Graph Learnable Multi-hop Attention Networks) to solve the problem of subgraph matching concurrently with the matching explanation. My contributions are presented as follows.

1. **Graph Learnable Multi-hop Attention Networks (GLeMa):** We introduce GLeMa, a novel mechanism capable of dynamically learning node-specific attention decay factors directly from the data. This adaptability mitigates over-smoothing problems that might arise from fixed attention decay mechanisms applied universally across nodes. By ensuring the model's decisions are data-driven, our approach circumvents potential suboptimal outcomes stemming from suboptimal parameter choices.

2. **Theoretical Analysis and Justification:** Our work is supported by theoretical justifications concerning the approximation error in multi-hop attention computation. Additionally, we provide formal proof validating the concept of learning node-specific attention decay factors. This demonstration showcases the consistency of utilizing distinct decay factors in approximating multi-hop attention, representing an advancement from earlier models that relied on universal attention decay factors in Graph Multi-hop Attention Networks (GMA).

3. **Multi-task Learning Framework:** We develop the multi-task learning framework capable of optimizing both subgraph matching and matching explanation tasks concurrently. This strategic integration fosters a symbiotic relationship between these tasks, enhancing the overall effectiveness and performance of our framework.

These contributions collectively underpin the efficacy and novelty of our approach, establishing a comprehensive foundation for further exploration and application in the domain of drug design.

1.3 Scope

Within this thesis, we introduce a pioneering methodology that tackles the challenge of subgraph matching while simultaneously providing matching explanations. Our focus centers on evaluating this approach extensively using chemistry and bioinformatics datasets, aligning with our orientation toward drug design processes. This comprehensive assessment spans multiple facets, encompassing efficiency, accuracy, scalability, and generalization to gauge the model's performance rigorously. The final result of this thesis is a framework featuring a theoretically substantiated GNN-based model.

1.4 Thesis Structure

The subsequent sections outline the structure of this thesis. Chapter 3 discusses related works, positioning our contributions within the existing literature. Chapter 2 provides the foundational background, establishing the study's contextual framework. Chapter 4 comprises five sections: Section 4.1 introduces our framework and outlines its constituent components. Section 4.2 presents our method for constructing a joint representation for pattern and target graphs, emphasizing the facilitation of intra-graph and inter-graph relationships within these representations. Section 4.3 elaborates on GLeMA, our proposed mechanism specifically designed for learning graph representation and inter-graph interactions. Section 4.4 explicates the aggregation of node embeddings to address subgraph matching and matching explanation tasks simultaneously via a novel objective function. The theoretical justification of GLeMA is presented in Section 4.5. Chapter 5 details our approach's evaluation using diverse public datasets focusing on chemistry and bioinformatics. Notably, our framework exhibits substantial advancements, achieving over a tenfold increase in subgraph matching speed compared to the fastest baseline. Additionally, it showcases significant improvements in accuracy (27%) and F1-score (34%) compared to NeuroMatch, the state-of-the-art approximation approach, on the COX2 dataset. Furthermore, our results demonstrate comparability with exact methods. Conversely, Chapter 6 provides a concise summary and future development of this thesis.

Chapter 2

Theoretical Background

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

2.1 Subgraph Isomorphism Problem

This thesis primarily focuses on addressing subgraph matching and matching explanation dilemmas concerning labeled, undirected, and connected graphs. However, the framework we propose can easily expand to include directed graphs. The symbols used in this study are concisely explained and summarized in Table 2.1.

Afterward, we subsequently provide formal definitions for labeled, undirected, and connected graphs in Definition 1, and extend this to include directed graphs in Definition 2. The concepts of labeled subgraphs, including both non-induced and induced subgraphs, are defined in Definitions 3, 4, and 5, respectively. Definition 6 presents the concept of labeled graph isomorphism. Building on these definitions, we introduce the problems of non-induced and induced labeled subgraph isomorphism in Definitions 7 and 8. We also plot an example illustrating the difference between induced and non-induced subgraph matching in Figure 2.1.

This thesis specifically focuses on induced labeled 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 [22].

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

1. V *is a set of nodes,*

Table 2.1: Summary of used notations

Symbol	Definition
$\mathcal{D}$	Dataset
$\mathcal{T}$	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

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. $l : V \to \Sigma$ *is a labelling function and* Σ *is the set of node labels.*

Definition 2 (Labelled Directed Connected Graph) *A labeled 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. $l : V \to \Sigma$ *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*

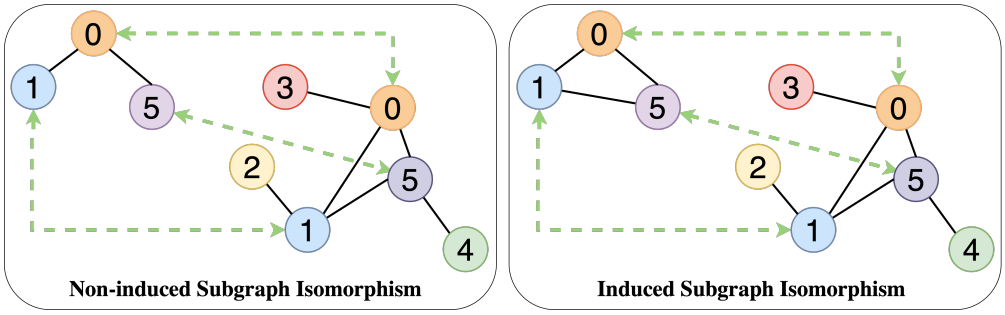


Figure 2.1: Examples of non-induced and induced subgraph isomorphism. The left subfigure illustrates a non-induced subgraph match, where the query graph (on the top-left) is embedded within the target graph while allowing additional edges in the matched subgraph that are not present in the query. The right subfigure demonstrates an induced subgraph match, where the subgraph must strictly maintain the same edge structure as the query graph, ensuring that no extra edges are present.

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. $\mathcal{P} \cong \mathcal{S}$.

2.2 Formal Problem Statement

This thesis 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 an induced 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 $\mathcal{T}$.*

Chapter 3

Related Works

In this chapter, we first explore the role of subgraph matching algorithms in drug discovery and highlight the importance of explaining the matching results (Section 3.1). The chapter then examines two different approaches to subgraph matching in research, covering both non-induced subgraph matching (Section 3.2) and induced subgraph matching (Section 3.3). Additionally, given our use of a neural-based method for subgraph matching, we investigate related approaches in neural subgraph matching and explanation in Section 3.4.

3.1 Subgraph Matching in Drug Discovery

Subgraph matching has emerged as a crucial computational technique in drug discovery, enabling efficient molecular search, drug-target interaction (DTI) prediction, and pharmacophore screening [23]–[25]. Given the graph-structured nature of molecular data, subgraph matching methods provide a means to identify key structural patterns, facilitating various AI-driven drug discovery applications. Recent advancements leverage graph neural networks (GNNs) and contrastive learning to improve efficiency and accuracy in large-scale molecular databases [2], [26].

One line of research focuses on correlated subgraph searches (CSSs), which identify molecules that are structurally similar to a given query molecule. Corgi [23], a recent framework, addresses the scalability issue of CSS by dynamically excluding unnecessary subgraphs while maintaining top-k search accuracy. This significantly reduces computational costs, making CSS methods more viable for AI-powered drug discovery applications. Similarly, exact subgraph matching techniques, such as the GNN-based Path Dominance Embedding (GNN-PE) method [26], ensure

robust and efficient search mechanisms by leveraging path embeddings and dominance relationships to enhance pruning strategies without false dismissals.

Another critical application of subgraph matching in drug discovery is drug-target interaction (DTI) prediction. ZeroBind introduces a meta-learning approach that employs subgraph matching to recognize binding pockets within proteins, thereby improving the generalizability of DTI predictions to novel proteins and drugs [24]. By incorporating a weakly supervised subgraph information bottleneck (SIB) module, ZeroBind effectively identifies informative subgraphs, which enhances interpretability and predictive accuracy. This approach demonstrates superior performance compared to conventional DTI models, particularly in zero-shot learning settings.

Pharmacophore screening, a fundamental step in virtual screening, has also benefited from subgraph matching techniques. PharmacoMatch [25], a neural subgraph matching framework, reformulates pharmacophore screening as an approximate subgraph matching problem, significantly reducing runtime while preserving accuracy. By encoding query-target relationships in an embedding space, PharmacoMatch enables fast and efficient screening of large molecular databases, making it a promising solution for high-throughput drug discovery applications.

Moreover, subgraph selection and encoding play a pivotal role in drug-drug interaction (DDI) prediction. Recent efforts explore neural architecture search (NAS) to optimize these components in a data-driven manner [27]. By designing a relaxation mechanism to navigate the large search space efficiently, customized subgraph selection methods improve interpretability and performance in predicting DDIs. The integration of these advanced subgraph-based techniques highlights the growing importance of graph representation learning in drug discovery, paving the way for more scalable, accurate, and interpretable AI-driven solutions.

In short, subgraph matching plays a fundamental role in drug discovery, facilitating tasks such as molecular search, drug-target interaction prediction, and pharmacophore screening. In addition to identifying subgraph isomorphism, understanding node mapping provides valuable insights into molecular interactions, binding sites, and functional group alignments. These explainability-driven approaches enhance trust and usability in AI-powered drug discovery pipelines [28], [29]. As research progresses, integrating robust matching explanations alongside efficient subgraph matching will be critical for advancing data-driven solutions in pharmaceutical development.

3.2 Non-induced Subraph Matching

Non-induced subgraph matching is a fundamental problem in graph analysis where a given pattern graph, denoted as $\mathcal{P}$, is searched as a partial embedding within a subgraph $\mathcal{S}$ of a larger data graph G. Unlike induced subgraph matching, which requires an exact correspondence of both nodes and edges, non-induced matching allows additional edges in $\mathcal{S}$ that are not present in $\mathcal{P}$. In other words, while every edge $e \in E_{\mathcal{P}}$ must have a corresponding edge in $E_{\mathcal{S}}$, the converse is not necessarily true, meaning that $\mathcal{S}$ may contain extra edges that do not appear in $\mathcal{P}$.

This relaxed constraint makes non-induced subgraph matching particularly useful in various applications within data management and graph analytics. It is widely employed in tasks such as graph indexing, where efficient searching of substructures within large graph databases is required; graph similarity search, which focuses on identifying structurally similar subgraphs despite minor variations; and graph retrieval, where relevant subgraphs are extracted based on query patterns [9]. Due to its flexibility, non-induced subgraph matching is particularly useful in domains where data is inherently noisy or incomplete, such as social network analysis, biological network discovery, and knowledge graph exploration.

Recent advancements in explainable subgraph matching have further increased interest in this area, driven by the development of efficient algorithms that enhance both interpretability and computational feasibility [30]. Explainability in subgraph matching is particularly crucial for domains like bioinformatics and fraud detection, where understanding the rationale behind detected patterns is essential for decision-making. These methods leverage heuristics, machine learning techniques, and combinatorial optimization strategies to improve the speed and accuracy of non-induced subgraph matching while maintaining transparency in their results.

Despite the significant progress in non-induced subgraph matching, the focus of our current study lies in induced subgraph matching, which imposes stricter conditions by requiring that all edges in the pattern graph be preserved exactly in the data graph. The subsequent section will delve into the theoretical and practical implications of induced subgraph matching, highlighting its distinct challenges and applications.

3.3 Induced Subraph Matching

The problem of induced subgraph matching has been formally established as NP-complete [31]. Given a pattern graph and a target graph, the goal of induced subgraph matching is to determine whether there exists an isomorphic subgraph within the target graph that exactly preserves both the vertex set and the edge structure of the pattern. Due to its combinatorial complexity, this problem is computationally expensive, particularly for large-scale graphs common in domains such as bioinformatics, social networks, and knowledge graphs. To tackle this computational challenge, numerous algorithms have been developed, leveraging efficient matching strategies, filtering mechanisms, and indexing techniques to reduce the candidate search space. These methods primarily focus on two key areas: (1) optimizing the matching order to minimize the search complexity and (2) employing pruning strategies to filter out non-promising candidates early in the process. Below, we summarize notable algorithms and their contributions.

- *VF3* [32] extends the original VF2 algorithm[33] proposed by the same authors. VF3 is tailored to handle larger graphs and builds upon a modified version of the VF2 algorithm. It employs an enhanced bit vector representation for graph states and introduces a novel matching order for query nodes, effectively reducing the search space. Additionally, the VF3 algorithm leverages various heuristics, including degree-based filtering, target graph decomposition, and dynamic reordering of query nodes to enhance the efficiency of the matching process.

- *TurboISO* [12] is a subgraph isomorphism algorithm designed to solve the problem more efficiently by employing a combination of filtering techniques and dynamic programming. It begins by pre-processing the input graphs and constructing an index for rapid filtering of candidate subgraphs. TurboISO then utilizes a divide-and-conquer strategy to break down the search space into smaller subproblems, which are independently solved using dynamic programming. The algorithm also integrates several heuristics, such as degree-based filtering and forward checking to narrow down the search space further.

- *CFL* [13] introduces a novel framework that minimizes redundant Cartesian products in the search space by strategically postponing them based on

the query structure. Additionally, this approach proposes a new path-based auxiliary data structure for generating a matching order and conducting subgraph matching more efficiently. This new data structure is notably more compact than its predecessors.

- *CECI* [14] partitions the data graph into multiple embedding clusters for parallel processing and employs BFS-based filtering, reverse-BFS-based refinement, and set intersection techniques to optimize the search process.

- *QuickSI* [11] employs QI-Sequence to constrain the search space in subgraph isomorphism testing. The QI-Sequence order is determined based on the frequencies of features to further reduce the search space. This algorithm enhances existing verification techniques, providing a speed-up of up to four orders of magnitude. Additionally, the method introduces a novel index called Swift-Index, where the mined tree features are represented as QI-Sequences. All QI sequences in the index are organized as a prefix tree, significantly reducing the cost in the filtering phase.

- *DAF* [15] is proposed to address the limitations of existing algorithms, such as lower pruning power, sub-optimal matching order, and redundant computation. The DAF algorithm introduces three novel concepts: dynamic programming between a directed acyclic graph (DAG) and a graph, adaptive matching order with DAG ordering, and pruning by failing sets. These concepts result in a much faster algorithm for subgraph matching.

- *GraphQL* [18] employs a specialized query language for graph databases that allows arbitrary attributes on nodes, edges, and graphs. Graphs serve as the fundamental unit of information, and each query manipulates one or more collections of graphs. The method extends the notion of formal languages from strings to the graph domain and introduces a graph algebra that extends the relational algebra. Additionally, the method addresses the challenge of a vast search space in subgraph isomorphism by utilizing neighborhood subgraphs and profiles, jointly reducing the search space, and optimizing the search order.

3.4 Neural Subgraph Matching and Explanation

The foundational study by Scarselli et al.[34] was among the first to explore the application of Graph Neural Networks (GNNs) in subgraph matching. Their work demonstrated the potential of GNNs in handling small-scale subgraphs and suggested that GNNs could outperform traditional feedforward neural networks in learning structural graph patterns. Since then, significant advancements in GNN architectures[35]–[37] have led to state-of-the-art subgraph matching techniques [16], [38], [39], which leverage neural networks to enhance efficiency and scalability. However, despite their empirical success, a fundamental challenge persists: establishing a direct and interpretable correlation between the learned graph representations and the underlying structure of the data graph. This limitation hinders the direct applicability of GNN-based approaches in critical downstream tasks such as subgraph isomorphism testing.

To address this interpretability gap, recent research has focused on explaining the decision-making processes of GNNs [40]–[42]. These studies aim to elucidate which features of the input graph contribute most significantly to the model's predictions. In the context of subgraph matching, Wu et al. [42] have specifically investigated how GNNs capture structural patterns, shedding light on their ability to perform subgraph isomorphism tasks. However, while these interpretability efforts are relevant to understanding GNN behavior, they are distinct from our focus in this thesis. Our work primarily concerns methodological advancements in subgraph matching, with interpretability remaining an area of future exploration.

In this thesis, we mainly consider two state-of-the-art GNN-based approaches for subgraph matching which are described in the following.

- NeuralMatch [16] represents a cutting-edge approach that employs a specialized GNN architecture to efficiently locate the neighborhood within a large target graph that contains a given query graph as a subgraph. NeuralMatch leverages a GNN to learn expressive graph embeddings in an ordered space, preserving key structural properties such as transitivity, antisymmetry, and non-trivial intersections. This design enables real-time approximate subgraph matching at an unprecedented scale, significantly improving computational efficiency over traditional combinatorial methods. Figure 3.1 is the overview of the NeuralMatch pipeline, which was taken from the original work [16].

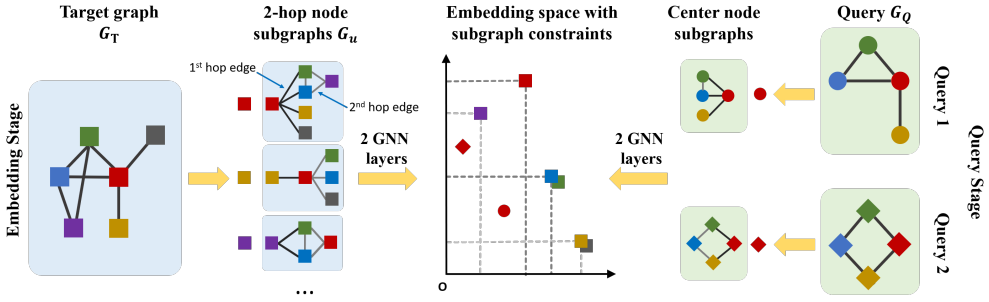


Figure 3.1: Overview of the NeuralMatch pipeline [16]. Conceptually, this method partitions the target graph into multiple 2-hop node subgraphs and embeds them in an ordered space. When processing a query graph, it similarly decomposes the query into multiple center-node subgraphs. It then verifies whether the embeddings of these query subgraphs in the ordered space are contained within the embeddings of the target's 2-hop node subgraphs to determine subgraph isomorphism.

- DualMP [17] is a state-of-the-art method designed for subgraph counting and matching. It introduces Dual Message Passing Neural Networks (DMPNNs), which simultaneously learn node and edge representations through an efficient asynchronous update mechanism. This approach enhances the model's ability to capture fine-grained structural information, making it well-suited for complex subgraph isomorphism tasks. DualMP naturally extends to heterogeneous multi-graphs and has demonstrated strong performance across multiple tasks, including subgraph counting, matching, and unsupervised node classification. Furthermore, it incorporates a multi-task learning framework that enables mutual supervision between tasks, thereby improving overall performance and robustness.

A closely related problem to exact subgraph matching is inexact matching, where approximate solutions are considered instead of strict isomorphisms. Although our approach utilizes a neural-based framework, our validation strictly relies on exact pattern matches. Several alternative methods have been proposed for handling inexact subgraph matching, including structural equivalence-based techniques [43], approximate matching methods [44], and knowledge graph-based approaches [45]. While these methods offer promising directions for extending neural subgraph matching beyond exact isomorphisms, they fall outside the scope of this thesis. We intend to explore these approaches in future work when expanding our framework to address approximate subgraph matching challenges.

3.5 Graph Neural Networks

Graph Neural Networks (GNNs) have emerged as a powerful framework for learning on graph-structured data, revolutionizing applications in various domains such as molecular chemistry, drug discovery, pattern recognition, and social network analysis. The development of GNN architectures has evolved from basic neural network adaptations for graphs to more sophisticated models leveraging attention mechanisms, recurrent structures, and message-passing paradigms [34], [46]–[48].

The foundational work on Graph Neural Networks introduced a model that extends traditional neural network architectures to process data in graph domains [34]. This model, designed to handle various graph types, including directed, undirected, cyclic, and acyclic graphs, formalized a mapping function that transforms a graph and its nodes into an m-dimensional Euclidean space. A supervised learning algorithm was derived to optimize this transformation, setting the stage for further advancements in GNN design [34].

A significant milestone in GNN development was the introduction of Graph Convolutional Networks (GCNs) [35], which efficiently perform semi-supervised learning on graphs. GCNs employ a localized first-order approximation of spectral graph convolutions, allowing them to capture both node features and local graph structure. By scaling linearly with the number of graph edges, GCNs demonstrated state-of-the-art performance on node classification tasks in citation networks and knowledge graphs.

Despite the success of GCNs, a deeper theoretical analysis revealed limitations in the expressive power of existing GNNs. The study on the discriminative power of GNNs highlighted that certain architectures fail to distinguish simple graph structures effectively. This analysis led to the development of more expressive GNN variants, including architectures inspired by the Weisfeiler-Lehman graph isomorphism test, Graph Isomorphism Networks (GIN), which achieve enhanced graph classification performance [37].

To address the limitations of traditional GNNs, Graph Attention Networks (GATs) [46] introduced self-attention mechanisms to dynamically weigh node neighborhood contributions. GATs enable adaptive importance assignment without requiring knowledge of the graph structure beforehand, improving performance across various transductive and inductive learning benchmarks. However, standard attention mechanisms in GNNs typically operate only on directly connected nodes,

limiting the ability to capture broader network context. To address this, Multi-hop Attention Graph Neural Network (MAGNA) incorporates multi-hop context information into attention computation by diffusing attention scores across the network [21]. This increases the receptive field at every layer, allowing MAGNA to capture large-scale structural information while filtering out high-frequency noise. MAGNA achieves state-of-the-art results on node classification and knowledge graph completion benchmarks, demonstrating up to 5.7% relative error reduction over prior methods. Additionally, Gated Graph Convolutional Recurrent Neural Networks (GCRNNs) [47] incorporated recurrent mechanisms to model dynamic graph processes, effectively capturing temporal dependencies in data such as earthquake prediction and weather forecasting.

Another breakthrough in GNN research was the development of Message Passing Neural Networks (MPNNs) [48], tailored specifically for molecular property prediction. MPNNs unify various existing models into a common framework that learns message-passing algorithms for effective feature aggregation. This approach has demonstrated state-of-the-art performance in quantum chemistry applications, providing a robust foundation for AI-driven drug discovery.

The continuous evolution of GNN architectures has significantly expanded their applicability and effectiveness across diverse domains. From early graph neural network models to advanced architectures integrating attention mechanisms, recurrent structures, and message passing, GNNs have become a cornerstone of modern graph-based machine learning. In this thesis, we focus on advancing MAGNA by learning distinct attention coefficients for each node, enabling a more comprehensive and intuitive graph representation, thus improving the performance of downstream tasks, i.e., subgraph matching and matching explanation.

Chapter 4

Method

4.1 Overview

4.1.1 Design principles

We proposed xNeuSM which leverages the inherent efficiency of neural network computation, a substantial advantage. 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) Explanability.*** 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 [20]. 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 [49]—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 [50].

4.1.2 The challenges

To address these objectives, we encounter specific challenges:

- ***Explainable Adaptivity.*** The neural-based approach adopted for the subgraph matching problem utilizes coarse-grained embeddings of entire graphs to approximate graph-level similarities (**R1**). However, achieving explicit, explainable alignment between nodes demands finer embeddings between the graphs, introducing an additional layer of supervision during training. Yet, creating such annotated training data is labor-intensive, leading to computationally inefficient procedures.

- ***High Computational Complexity with Graph Multi-hop Attention.*** Incorporating high-order dependencies, as required by (**R2**), is indeed a complex task. Increasing the dependency orders can significantly increase the computational load on the model. Consequently, many existing approaches often limit the dependency to the second order. Finding a balance between maintaining efficiency and capturing a broad receptive field is a challenging endeavor.

- ***Multi-objective Optimization.*** Previous studies [16], [34], [38], [39] have employed neural networks to characterize similarity functions, operating on graph-level or node-level embeddings. While these networks demonstrate competitive performance in approximating similarity and aiding retrieval tasks compared to traditional methods, integrating an additional optimization objective for analyzing node-to-node mappings between query-target

graph pairs (**R3**) presents challenges. Designing a neural model seamlessly accommodating new objectives like this remains a significant architectural challenge.

4.1.3 The approach

An exhaustive depiction of our xNeuSM framework is outlined in Figure 4.1 and its formal algorithmic flow in Algorithm 1, delineating three primary stages as detailed below. Our architecture design was inspired by a study of predicting interaction between drug and target protein [51].

Input Representation. Diverging from prior neural-based methodologies [16] that directly glean information from distinct adjacency matrices of pattern and target graphs, our newly devised unified proxy inputs enable the comprehensive capture of inter-graph relations (**R1**). These inputs also fortify the concurrent learning process. Further explication regarding this facet is available in Section 4.2.

Graph Learnable Multi-hop Attention Networks. Our innovative approach employs a specialized graph neural network, GLeMA, to extract higher-order dependencies. GLeMA facilitates the adept representation of inter- and intra-interactions between pattern and target graphs through a learnable multi-hop attention mechanism, facilitating simultaneous learning of these interactions. This integration of high-order dependencies ensures scalability (**R2**), particularly concerning larger patterns. Detailed insights into this process are delineated in Section 4.3.

Multi-task Optimization. In this third stage, we aggregate node embeddings while concurrently addressing two tasks: *subgraph prediction* and *matching explanation*. Both tasks leverage the acquired features from the preceding stage. Additionally, we introduce a novel objective function aimed at optimizing both tasks concurrently (**R3**). Section 4.4 offers an in-depth exploration of this phase.

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**

$\widehat{\boldsymbol{X}}_{in}^{l} \leftarrow \text{GLeMa}_l(\boldsymbol{X}^{l-1}, \boldsymbol{A}^{in})$

$\widehat{\boldsymbol{X}}_{cr}^{l} \leftarrow \text{GLeMa}_l(\boldsymbol{X}^{l-1}, \boldsymbol{A}^{cr})$

$\boldsymbol{X}^{l} \leftarrow \widehat{\boldsymbol{X}}_{cr}^{l} - \widehat{\boldsymbol{X}}_{in}^{l}$

end

$(\mathcal{A}^{(1)})^{L_G} \leftarrow \text{ExtractAttnMat}(\text{GLeMa}_{L_G}, \boldsymbol{X}^{l-1}, \boldsymbol{A}^{cr})$

$x_{repr}^{0} \leftarrow \frac{1}{|V_{\mathcal{P}}|} \sum_{i \in V_{\mathcal{P}}} x_i^{L_G}$, where $x_i^{L_G} \subset \boldsymbol{X}^{L_G}$

for l ***in*** *Range(*$1 \dots L_{FC} - 1$*)* **do**

$x_{repr}^{l} \leftarrow \delta(\boldsymbol{W}_l x_{repr}^{l-1} + b_l)$

end

$\widehat{y} \leftarrow \sigma(\boldsymbol{W}_y x_{repr}^{L_{FC}-1} + b_y)$

$\boldsymbol{P} \leftarrow \left\{ p_{ij} = \frac{1}{2} \left((a_{ij}^{(1)})^{L_G} + (a_{ji}^{(1)})^{L_G} \right) \right\}$, where $i \in V_{\mathcal{P}}, j \in V_{\mathcal{T}}$, and $(a_{ij}^{(1)})^{L_G} \in (\mathcal{A}^{(1)})^{L_G}$

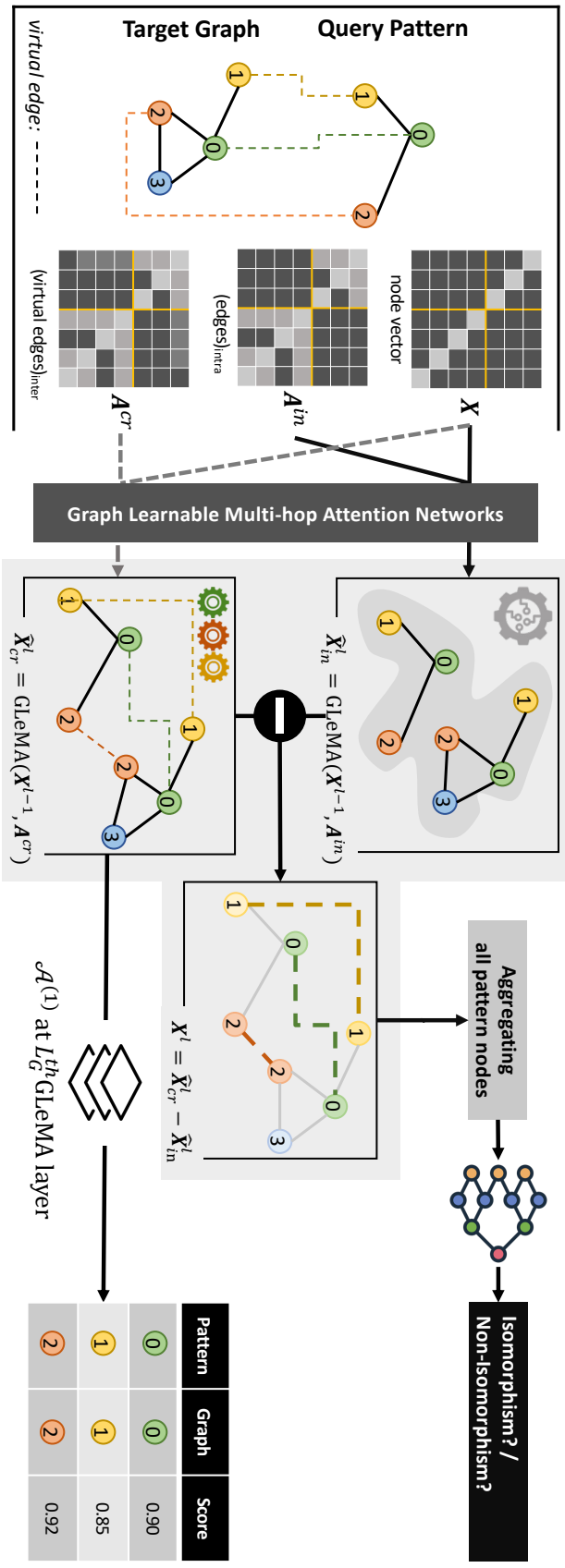


Figure 4.1: Overview of xNeuSM framework

4.2 Input Representation

The initialization process of the model begins by preparing the input data. In the domain of the subgraph isomorphism problem, this input consists of a smaller graph (referred to as a pattern) and a larger graph (referred to as a target). Both the pattern and target are characterized conventionally by sets of nodes and edges. For our specific problem, we denote 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 and E represent sets of nodes and edges, respectively, and $l : V \to T_V$ is a function that assigns labels to nodes.

The construction of the input for our proposed model involves several components. It includes a collection of node feature vectors denoted as x, the primary adjacency matrix $\boldsymbol{A}^{in}$, and the secondary adjacency matrix $\boldsymbol{A}^{cr}$. Each node in the pattern or target is encoded as a one-hot vector of dimensions $2T_V$, where $T_V = T_{V_\mathcal{P}} \cup T_{V_\mathcal{T}}$ represents the maximum count of unique node labels. The first T_V dimensions correspond to the pattern, while the remaining dimensions are for the target graph. This separation of node features for the pattern and target enables distinct embeddings, thereby improving the mapping performance.

Subsequently, all node vectors are combined to create the collective input set denoted as $\boldsymbol{X}$. The primary adjacency matrix $\boldsymbol{A}^{in}$ indicates intra-graph edges, signifying the absence of edges connecting the pattern and target nodes. Conversely, the $\boldsymbol{A}^{cr}$ matrix establishes a “virtual" edge between a pattern node and a target node when they share identical labels. The mathematical representations for $\boldsymbol{X}$, $\boldsymbol{A}^{in}$, and $\boldsymbol{A}^{cr}$ are formally defined in equations (4.1), (4.2), and (4.3), respectively.

$$\boldsymbol{X} = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{|V_\mathcal{P}|}, \vec{x}_{|V_\mathcal{P}|+1}, \ldots, \vec{x}_{|V_\mathcal{P}|+|V_\mathcal{T}|}\} \text{ with } \vec{x}_i \in \mathbb{R}^{2T_V} \tag{4.1}$$

$$\boldsymbol{A}^{in}_{ij} = \begin{cases} 1 & \text{if there is an indirect edge or} \\ & \text{a direct edge that connects } j \text{ to } i \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

$$\boldsymbol{A}^{cr}_{ij} = \begin{cases} \boldsymbol{A}^{in}_{ij} & \text{if } i, j \in \mathcal{P} \text{ or } i, j \in \mathcal{T} \\ 1 & \text{if } l(i) = l(j) \text{ and } i \in \mathcal{P} \text{ and } j \in \mathcal{T}, \\ & \text{or if } l(i) = l(j) \text{ and } i \in \mathcal{T} \text{ and } j \in \mathcal{P} \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

4.3 Graph Learnable Multi-hop Attention Networks

In this section, we commence by delineating the procedure for extracting node features dedicated to the joint pattern-target representation learning, expounded in Section 4.3.1. Sequentially, we explicate the operations of a singular stratum within the Learnable Multi-hop Attention mechanism, expounded in Section 4.3.2. Subsequent to this, we deliberate on the integration of multiple strata of the attention mechanism to enable the acquisition of heightened dependencies within the networks, as expounded in Section 4.3.3.

4.3.1 Extracting Node Features

In this thesis, we introduce a Graph Learnable Multi-hop Attention layer, denoted as GLeMa($\cdot$), which integrates Graph Attention Networks with a Learnable Multi-hop Attention mechanism. This fusion enables the comprehensive assimilation of structural information from both target and pattern graphs. Considering the application of this layer to an abstract graph $\mathcal{G} = (V, E, l)$, where $\boldsymbol{X} \in \mathbb{R}^{|V| \times F}$ signifies the node feature set and E represents the edge set, we represent this graph as $(\boldsymbol{X}, \boldsymbol{A})$. Formally, the output of our proposed layer is denoted as $\widehat{\boldsymbol{X}} = \text{GLeMa}(\boldsymbol{X}, \boldsymbol{A})$. The input to our GLeMa layer consists of a fusion of node features, $\boldsymbol{X}$ (as specified in equation (4.4)), and an adjacency matrix, $\boldsymbol{A}$ (outlined in equation (4.5)).

$$\boldsymbol{X} = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{|V|}\},\ \vec{x_i} \in \mathbb{R}^F, \tag{4.4}$$

where F is the number of features.

$$\boldsymbol{A}_{ij} = \begin{cases} 1 & \text{if there is an edge that connects } j \text{ to } i \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

Subsequently, node feature vectors are projected into an embedding space via a linear transformation represented by $\vec{x'}_i = \boldsymbol{W}_h \vec{x}_i$, where $\vec{x'}_i \in \mathbb{R}^{F'}$ and $\boldsymbol{W}_h \in \mathbb{R}^{F' \times F}$ denotes a trainable weight matrix. The attention coefficients for each node pair are computed utilizing Luong's attention mechanism [52], as outlined in (4.6).

$$e_{ij} = \begin{cases} \delta(\vec{x'}_i^T \boldsymbol{W}_e \vec{x'}_j) & \text{a directed edge } j \text{ to } i, \\ \delta(\vec{x'}_i^T \boldsymbol{W}_e \vec{x'}_j + \vec{x'}_j^T \boldsymbol{W}_e \vec{x'}_i) & \text{an undirected edge } j \text{ to } i, \end{cases} \tag{4.6}$$

where δ is a non-linear activation function and $\boldsymbol{W}_e \in \mathbb{R}^{F' \times F'}$ is a learnable matrix. Afterward, the normalization step involves applying the softmax function to all attention coefficients, leading to the formation of the 1-hop attention matrix denoted as $\mathcal{A}^{(1)}$. Within this normalization procedure, the principle of "masked attention" is integrated, restricting consideration solely to nodes $j \in \mathcal{N}_i$ for the normalization process, where $\mathcal{N}_i$ represents the set of neighboring nodes of node i. Moreover, to effectively nullify the impact of non-neighboring nodes, attention values normalized between nodes i and j are substituted with zeros when no edge connects node i to node j. The mathematical methodology governing the normalization of attention coefficients is detailed in equation (4.7).

$$\begin{cases} \mathcal{A}^{(1)} & = \{a_{ij}^{(1)} | i, j = \overline{1, |V|}\} \\ a_{ij}^{(1)} & = \text{softmax}_j(e_{ij}) \boldsymbol{A}_{ij} = \frac{\exp(e_{ij})}{\sum_{n \in \mathcal{N}_i} \exp(e_{in})} \boldsymbol{A}_{ij} \end{cases} \tag{4.7}$$

After acquiring the 1-hop attention matrix, we generate the attention diffusion matrix $\mathcal{A}$ using the multi-hop mechanism established in Graph Multi-hop Attention Networks (GMA) [21]. Precisely, the matrix $\mathcal{A}$ is formally described in equation (4.8).

$$\begin{cases} (\mathcal{A}^{(1)})^0 & = \boldsymbol{I} \\ \mathcal{A} & = \sum_{k=0}^{\infty} \theta_k (\mathcal{A}^{(1)})^k \text{ where } \sum_{k=0}^{\infty} \theta_k = 1 \text{ and } \theta_k > 0 \end{cases} \tag{4.8}$$

In equation (4.11), the parameter θ_k represents the attention decay factor, meeting the condition $\theta_k > \theta_{k+1}$ to ensure a gradual decrease in importance for more distant nodes. Following this, for each node i, weighted summations are performed between itself and other nodes to construct a new feature vector for the i^{th} node, utilizing the multi-hop attention matrix.

Moreover, the utilization of multi-head attention [21], [46] is employed to generate diverse feature representations from various unique perspectives. The resulting vectors from multi-head attention are then concatenated to produce the final refined feature vectors for nodes. The formulation outlined in equation (4.9) illustrates the procedure for generating the ultimate output within this layer.

$$\widehat{\boldsymbol{X}} = \left(\Big\Vert_{h=1}^{H} \delta\left(\mathcal{A}_h \boldsymbol{X'}_h\right) \right) \boldsymbol{W}_o \text{ with } \boldsymbol{W}_o \in \mathbb{R}^{HF' \times F'}. \tag{4.9}$$

In equation (4.9), the symbol $\widehat{\boldsymbol{X}}$ stands for the set of resulting node feature vectors, where H represents the number of attention heads utilized in the multi-head attention mechanism. The notation $\mathcal{A}_h$ refers to the multi-hop attention matrix related to the h-th attention head. Similarly, $\boldsymbol{X'}_h$ denotes the matrix representing the projected node features for the h-th attention head, and $\boldsymbol{W}_o$ signifies a matrix comprising trainable weights.

4.3.2 Learnable Multi-hop Attention mechanism

The subsequent challenges we face in our thesis involve (i) the increased computational complexity in computing $\mathcal{A}$ due to matrix multiplication, as discussed in prior work [53], and (ii) the critical task of selecting appropriate values for θ_k, significantly influencing the enhancement or degradation of the model performance [53].

Reducing multi-hop attention matrix computation complexity In this thesis, following the approach delineated in prior research on Graph Multi-hop Attention (GMA) [21], we opt for the geometric distribution to determine θ_k. Here, we choose $\theta_k = \alpha(1-\alpha)^k$, where $\alpha \in (0, 1)$ denotes the teleport probability. Consequently, we approximate $\mathcal{A}\boldsymbol{X'}$ using equation (4.10).

$$\begin{cases} \boldsymbol{Z}^{(0)} & = \boldsymbol{X'} \\ \boldsymbol{Z}^{(k)} & = (1-\alpha)\mathcal{A}^{(1)}\boldsymbol{Z}^{(k-1)} + \alpha\boldsymbol{Z}^{(0)},\ k = \overline{1, K} \end{cases} \tag{4.10}$$

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

This proposition was proved in [21], illustrating the reduction of the computational complexity for calculating $\mathcal{A}$ to $O(K|E|)$ message passing operations. However, by considering K as a constant with $K \ll +\infty$, concerns arise regarding the fidelity of the approximation $Z^{(K)}$ to $\mathcal{A}X'$. This concern revolves around selecting an appropriate K that strikes a balance between approximation accuracy and computational complexity. Previous studies [21] proposed empirically choosing $3 \leq K \leq 10$ through experimentation. In this study, we offer theoretical

justification for this selection approach in Section 4.5.1. Consequently, presuming the approximation of $\mathcal{A}$ by $\mathcal{A}^{(K)}$ (equation (4.11)), the computational complexity for computing $\mathcal{A}$ is notably reduced to $O(|E|)$.

$$\begin{cases} (\mathcal{A}^{(1)})^0 & = \boldsymbol{I} \\ \mathcal{A} & \approx \mathcal{A}^{(K)} = \boldsymbol{Z}^{(K)} \boldsymbol{X'}^{-1} \end{cases} \tag{4.11}$$

Learning teleport probabilities Within graph attention diffusion theory, a pivotal concept is the characterization of Personalized PageRank (PPR) [54], which elucidates the significance of individual nodes. The seminal work on GMA has established a connection between the attention matrix in GMA and the transition matrix in PPR [21]. However, this original work employed a uniform teleport probability for all nodes, which theoretically limits the potential of PPR. To harness the capabilities of PPR more effectively, we introduce a novel approach that entails tailoring teleport probabilities for individual nodes, represented as $\beta = \{\beta_v\}_{v=1}^{|V|}$. The primary challenge lies in selecting appropriate β_v values for each node. To tackle this challenge, drawing inspiration from Gated Recurrent Units [55], we devise a method where the network autonomously learns the teleport probabilities via a simple linear transformation involving $\boldsymbol{W}_\beta \in \mathbb{R}^{2F' \times 1}$. Equation (4.12) delineates the derivation of β.

$$\beta = \sigma((\boldsymbol{X'} || \mathcal{A}^{(1)} \boldsymbol{X'}) \boldsymbol{W}_\beta + b), \tag{4.12}$$

where $||$ denotes the concatenation operator, and b represents the bias term. It is important to highlight that the utilization of distinct teleport probabilities for each node leads to adjustments in (4.8). These adjustments include $\theta_{kj} = \beta_j(1-\beta_j)^k$, $\sum_{k=0}^{\infty} \theta_{kj} = 1$ for $j \in \overline{1, N}$, and $\theta_{kj} > 0$. They also entail a row-wise multiplication between θ_k and $(\mathcal{A}^{(1)})^k$. Importantly, these modifications do not contradict Proposition 1, as established in Section 4.5.2. Furthermore, we offer a comprehensive procedure for computing learnable multi-hop attention in Algorithm 2.

4.3.3 Graph Learnable Multi-hop Attention Networks

When handling input data containing a pattern and a target, represented as a triple $(\boldsymbol{X}, \boldsymbol{A}^{in}, \boldsymbol{A}^{cr})$, we utilize our proposed GLeMa layer (GLeMa$(\cdot)$) to extract latent features. Specifically, the input is divided into two sets: $(\boldsymbol{X}, \boldsymbol{A}^{in})$ and $(\boldsymbol{X}, \boldsymbol{A}^{cr})$,

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*$(1 \dots K)$ **do**
  $\boldsymbol{Z}^{(k)} = (1-\beta)\mathcal{A}^{(1)}\boldsymbol{Z}^{(k-1)} + \beta\boldsymbol{Z}^{(0)}$
end
$\widehat{\boldsymbol{X}} \leftarrow \boldsymbol{Z}^{(K)}$

which undergo L_G iterations of GLeMa layers. The resultant representation for $(\boldsymbol{X}, \boldsymbol{A}^{in})$ captures features within the graph, while the representation for $(\boldsymbol{X}, \boldsymbol{A}^{cr})$ captures features across graphs.

At the l^{th} layer, node features are computed by measuring the discrepancy between inter-graph features and intra-graph features from the preceding $(l-1)^{th}$ layer. This learning process emphasizes the differences between inter-graph and intra-graph features, amplifying the signal for subgraph isomorphism verification. The formal definition of the Graph Learnable Multi-hop Attention Networks architecture is articulated in Equation 4.13.

$$\begin{cases} \boldsymbol{X}^0 = \boldsymbol{X} \\ \widehat{\boldsymbol{X}}^l_{in} = \text{GLeMa}_l(\boldsymbol{X}^{l-1}, \boldsymbol{A}^{in}), l = \overline{1, L_G} \\ \widehat{\boldsymbol{X}}^l_{cr} = \text{GLeMa}_l(\boldsymbol{X}^{l-1}, \boldsymbol{A}^{cr}), l = \overline{1, L_G} \\ \boldsymbol{X}^l = \widehat{\boldsymbol{X}}^l_{cr} - \widehat{\boldsymbol{X}}^l_{in}, l = \overline{1, L_G} \end{cases} \tag{4.13}$$

4.4 Multi-task Optimization

In this section, we present the computation strategies for both the subgraph matching task detailed in Section 4.4.1 and the matching explanation task covered in Section 4.4.2. Following this, we explore the optimization method for multi-task learning, elaborated upon in Section 4.4.3.

4.4.1 Subgraph Matching Task

Within the domain of subgraph matching, subsequent to their extraction via L_G GLeMa layers, the node feature vectors obtained from patterns are consolidated to produce a representative vector. This representative vector plays a crucial role in discerning the isomorphism between the input pattern and the target graph, accomplished via a classifier consisting of L_{FC} fully connected layers. The procedure for computing the representative vector is delineated in Equation 4.14, while Equation 4.15 furnishes the mathematical expressions supporting the classifier.

$$x_{repr}^{0} = \frac{1}{|V_{\mathcal{P}}|} \sum_{i \in V_{\mathcal{P}}} x_i^{L_G} \tag{4.14}$$

$$\begin{cases} x_{repr}^{i} & = \delta(\boldsymbol{W}_i x_{repr}^{i-1} + b_i), i = \overline{1, L_{FC} - 1} \\ \widehat{y} & = \sigma(\boldsymbol{W}_y x_{repr}^{L_{FC}-1} + b_y) \end{cases} \tag{4.15}$$

In Equation 4.14, we denote x_{repr}^{0} as the resultant representation vector derived from the input, and x_i^{LG} as the embedding vector of the i^{th} node following L_G GLeMa layers. In (4.15), x_{repr}^{i} represents the output vector of the i-th fully-connected layer, where $\boldsymbol{W}_i$ and b_i indicate the corresponding trainable weight matrix and bias parameters.

4.4.2 Matching Explanation Task

Utilizing the effectiveness of the multi-hop attention mechanism, our proposed model can forecast the placement of a pattern within a target graph. The distinctive aspect of our approach involves filtering pairs of *(pattern node, target node)* based on the 1-hop attention coefficients obtained from the final inter-graph GLeMa layer, determined by a preset threshold. Assuming that a pair meeting the threshold signifies a valid mapping between a pattern node and a target node, our

model can enumerate all potential mappings, regardless of whether the input pattern exhibits isomorphism with the target or not. The detailed computational steps of this matching task are elaborated upon in Equation 4.16. In (4.16), $\mathcal{M}$ denotes the set of mapped nodes between the pattern and target graph; p_{ij}, calculated as the average of 1-hop attention coefficients $((a_{ij}^{(1)})^{L_G}, (a_{ji}^{(1)})^{L_G})$, represents the probability of mapping between the i^{th} node in the pattern and the j^{th} node in the target.

$$\begin{aligned} \mathcal{M} =& \{(i, j, p_{ij}) | p_{ij} \geq \epsilon\}, \text{where } i \in V_{\mathcal{P}}, j \in V_{\mathcal{T}} \text{ and} \\ p_{ij} =& \frac{1}{2}\left((a_{ij}^{(1)})^{L_G} + (a_{ji}^{(1)})^{L_G}\right). \end{aligned} \tag{4.16}$$

4.4.3 Multi-task Learning Optimization

To optimize our proposed models for both subgraph matching and matching explanation tasks, we introduce a combined loss function comprising two core elements. The first component, labeled as $\mathcal{L}_{sm}$, employs binary cross-entropy to precisely evaluate the model ability to predict subgraph isomorphism. The second component, $\mathcal{L}_{me}$, constitutes an attention-based loss intending to reinforce attention coefficients between nodes i and j ($i \in V_{\mathcal{P}}, j \in V_{\mathcal{T}}$) that correspond to actual mappings while reducing coefficients for node pairs with identical labels (represented as $m \in V_{\mathcal{P}}, n \in V_{\mathcal{T}}, l(m) = l(n)$) but lack a mapping relationship.

Our comprehensive objective function, detailed in Equation 4.17, combines $\mathcal{L}_{sm}$ and $\lambda\mathcal{L}_{me}$, where the parameter λ functions as a hyperparameter regulating the relative significance of the two loss components.

$$\begin{cases} \mathcal{L}_{sm} & = -\frac{1}{|\mathcal{D}|}\sum_{k=1}^{|\mathcal{D}|} y_k \cdot \log(\widehat{y}_k) + (1 - y_k) \cdot \log(1 - \widehat{y}_k) \\ \mathcal{L}_{me} & = \frac{1}{|\mathcal{D}|}\sum_{k=1}^{|\mathcal{D}|} \frac{\sum \exp\left(-\left(a_{ij}^{(1)(L_G)}\right)_k\right)}{\sum \exp\left(-\left(a_{mn}^{(1)(L_G)}\right)_k\right) - \sum \exp\left(-\left(a_{ij}^{(1)(L_G)}\right)_k\right) + 1} \\ \mathcal{L} & = \mathcal{L}_{sm} + \lambda\mathcal{L}_{me} \end{cases} \tag{4.17}$$

4.5 Theoretical Justifications

4.5.1 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 (4.8). By Proposition 1, we can derive:

$$\lim_{K\to\infty} \boldsymbol{Z}^{(K)}(\boldsymbol{X'})^{-1} = \mathcal{A}. \tag{4.18}$$

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 $\mathrm{Err}(\mathcal{A} - \mathcal{A}^{(K)}) \leq (1-\alpha)^{K+1}$, where α is the teleport probability and K is the number of hops.
Firstly, we decompose $\boldsymbol{Z}^{(K)}$ as follows:

$$\begin{aligned}\boldsymbol{Z}^{(K)} = (1-\alpha)^K(\mathcal{A}^{(1)})^K\boldsymbol{X'} + \alpha(1-\alpha)^{K-1}(\mathcal{A}^{(1)})^{K-1}\boldsymbol{X'} \\ + \cdots + \alpha(1-\alpha)(\mathcal{A}^{(1)})\boldsymbol{X'} + \alpha\boldsymbol{X'}\end{aligned} \tag{4.19}$$

Then, we obtain:

$$\begin{aligned}\boldsymbol{Z}^{(K)}(\boldsymbol{X'})^{-1} &= (1-\alpha)^K(\mathcal{A}^{(1)})^K + \alpha(1-\alpha)^{K-1}(\mathcal{A}^{(1)})^{K-1} \\ &\quad + \cdots + \alpha(1-\alpha)(\mathcal{A}^{(1)}) + \alpha \\ &= (1-\alpha)^K(\mathcal{A}^{(1)})^K + \sum_{k=0}^{K-1}\alpha(1-\alpha)^k(\mathcal{A}^{(1)})^k\end{aligned} \tag{4.20}$$

Now, let us consider the difference between the attention diffusion matrix $\mathcal{A}$ and

its approximation $\mathcal{A}^{(K)}$.

$$\begin{aligned}
\mathcal{A} - \mathcal{A}^{(K)} &= \mathcal{A} - \boldsymbol{Z}^{(K)}(\boldsymbol{X'})^{-1} \\
&= \sum_{k=0}^{\infty} \alpha(1-\alpha)^k(\mathcal{A}^{(1)})^k - (1-\alpha)^K(\mathcal{A}^{(1)})^K - \sum_{k=0}^{K-1} \alpha(1-\alpha)^k(\mathcal{A}^{(1)})^k \\
&= \sum_{k=K}^{\infty} \alpha(1-\alpha)^k(\mathcal{A}^{(1)})^k - (1-\alpha)^K(\mathcal{A}^{(1)})^K \\
&\leq \sum_{k=K}^{\infty} \alpha(1-\alpha)^k(\mathcal{A}^{(1)})^k - \alpha(1-\alpha)^K(\mathcal{A}^{(1)})^K \text{ as } \alpha, a_{ij}^{(1)} \in (0,1) \\
&\leq \sum_{k=K+1}^{\infty} \alpha(1-\alpha)^k(\mathcal{A}^{(1)})^k
\end{aligned} \tag{4.21}$$

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

$$(\mathcal{A}^{(1)})^k \leq (\mathcal{A}^{(1)})^{k-1}. \tag{4.22}$$

As a consequence, we have:

$$(\mathcal{A}^{(1)})^k \leq \mathcal{A}^{(1)}, \forall k \geq 1 \tag{4.23}$$

Using (4.23), equation (4.21) can be derived as follows:

$$\boldsymbol{E} = \mathcal{A} - \mathcal{A}^{(K)} \leq \left(\sum_{k=K+1}^{\infty} \alpha(1-\alpha)^k \right) \mathcal{A}^{(1)} \tag{4.24}$$

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:

$$\begin{aligned}
\text{Err}(\mathcal{A} - \mathcal{A}^{(K)}) &= \frac{1}{|V|^2}\sum_{i,j}\boldsymbol{E}_{ij} \\
&\leq \frac{1}{|V|^2}\sum_{i,j}\left(\left(\sum_{k=K+1}^{\infty}\alpha(1-\alpha)^k\right)a_{ij}^{(1)}\right) \\
&\leq \sum_{k=K+1}^{\infty}\alpha(1-\alpha)^k \text{ as } a_{ij}^{(1)} \in (0,1) \\
&\leq \alpha\sum_{k=K+1}^{\infty}(1-\alpha)^k \leq \alpha\frac{(1-\alpha)^{K+1}}{1-(1-\alpha)} \\
&\leq (1-\alpha)^{K+1}
\end{aligned} \tag{4.25}$$

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 4.2. 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.

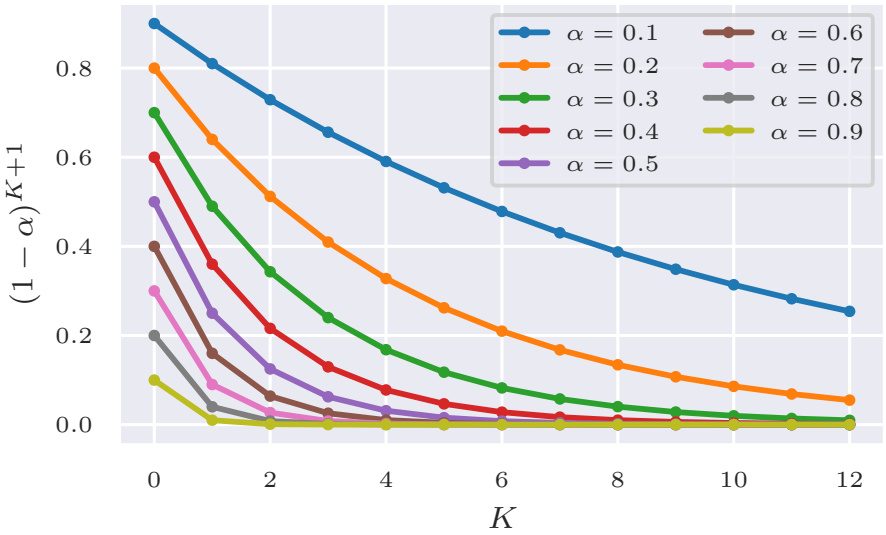


Figure 4.2: Maximal approximation error of each attention coefficient

4.5.2 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:

$$(\eta_v)_k = \beta_v(1 - \beta_v)^k > 0. \tag{4.26}$$

This leads to the important property:

$$\forall v \in V, \sum_{k=0}^{\infty}(\eta_v)_k = \sum_{k=0}^{\infty}\beta_v(1 - \beta_v)^k = \frac{\beta_v}{1 - (1 - \beta_v)} = 1. \tag{4.27}$$

Let $\eta_k = \{(\eta_v)_k\}_{v=1}^{|V|}$ be the attention decay vector at the k-th hop. With the property in (4.27), we can generalize equation (4.8) as follows:

$$\begin{cases} (\mathcal{A}^{(1)})^0 & = \boldsymbol{I} \\ \mathcal{A}_\eta & = \sum_{k=0}^{\infty}\eta_k(\mathcal{A}^{(1)})^k. \end{cases} \tag{4.28}$$

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 4.3.2, we can approximate $\mathcal{A}_\eta \boldsymbol{X'}$ with $\boldsymbol{Z}_\beta^{(k)}$ as follows:

$$\begin{cases} \boldsymbol{Z}_\beta^{(0)} & = \boldsymbol{X'} \\ \boldsymbol{Z}_\beta^{(k)} & = (\vec{1} - \beta)\mathcal{A}^{(1)}\boldsymbol{Z}_\beta^{(k-1)} + \beta\boldsymbol{Z}_\beta^{(0)},\ k = (1, K) \end{cases} \tag{4.29}$$

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

Proof of Proposition 2:

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

$$\begin{aligned}\boldsymbol{Z}_{\beta}^{(K)} = &\underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{K \text{ times}}\boldsymbol{X'} + \underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{K-1 \text{ times}}\beta\boldsymbol{X'} \\ &+ \underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{K-2 \text{ times}}\beta\boldsymbol{X'} + \dots + (\vec{1}-\beta)\mathcal{A}^{(1)}\beta\boldsymbol{X'} + \beta\boldsymbol{X'}\end{aligned} \tag{4.30}$$

We also have:

$$\begin{aligned}\mathcal{A}_{\eta}\boldsymbol{X'} &= \left(\sum_{k=0}^{\infty}\eta_k(\mathcal{A}^{(1)})^k\right)\boldsymbol{X'} \\ &= \eta_0\boldsymbol{X'} + \eta_1\mathcal{A}^{(1)}\boldsymbol{X'} + \eta_2(\mathcal{A}^{(1)})^2\boldsymbol{X'} + \dots \\ &= \beta\boldsymbol{X'} + \beta(\vec{1}-\beta)\mathcal{A}^{(1)}\boldsymbol{X'} + \beta(\vec{1}-\beta)^2(\mathcal{A}^{(1)})^2\boldsymbol{X'} + \dots\end{aligned} \tag{4.31}$$

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:

$$(\vec{1}-\beta)\mathcal{A}^{(1)} = \mathcal{A}^{(1)}(\vec{1}-\beta). \tag{4.32}$$

This leads to:

$$\begin{aligned}\underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{k \text{ times}}\beta\boldsymbol{X'} &= (\underbrace{(\vec{1}-\beta)\dots}_{k \text{ times}})(\underbrace{\mathcal{A}^{(1)}\dots}_{k \text{ times}})\beta\boldsymbol{X'} \\ &= (\vec{1}-\beta)^k(\mathcal{A}^{(1)})^k\beta\boldsymbol{X'} \\ &= \beta(\vec{1}-\beta)^k(\mathcal{A}^{(1)})^k\boldsymbol{X'}.\end{aligned} \tag{4.33}$$

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:

$$\lim_{k\to\infty}(1-\beta_v)^k = 0. \tag{4.34}$$

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

$$\lim_{k\to\infty}(\vec{1}-\beta)^k = \{\lim_{k\to\infty}(1-\beta_v)^k\}_{v=1}^{|V|} = \vec{0}. \tag{4.35}$$

With Lemma 1 and (4.35), we can conclude:

$$\begin{aligned}\lim_{k\to\infty}\underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\ \dots}_{k\text{ times}}\boldsymbol{X'} &= \lim_{k\to\infty}(\vec{1}-\beta)^k(\mathcal{A}^{(1)})^k\boldsymbol{X'}\\ &= (\lim_{k\to\infty}(\vec{1}-\beta)^k)(\lim_{k\to\infty}(\mathcal{A}^{(1)})^k\boldsymbol{X'})\\ &= \vec{0}(\lim_{k\to\infty}(\mathcal{A}^{(1)})^k\boldsymbol{X'})\\ &= O.\end{aligned} \tag{4.36}$$

Thus, Lemma 2 is proven. □

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

$$\begin{aligned}&\lim_{K\to\infty}\boldsymbol{Z}_\beta^{(K)}\\ &= \lim_{K\to\infty}\left[\underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{K\text{ times}}\boldsymbol{X'}\right]\\ &+ \lim_{K\to\infty}\left[\underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{K-1\text{ times}}\beta\boldsymbol{X'}+\cdots+(\vec{1}-\beta)\mathcal{A}^{(1)}\beta\boldsymbol{X'}+\beta\boldsymbol{X'}\right]\\ &= O+\lim_{K\to\infty}\left[\beta(\vec{1}-\beta)^{K-1}(\mathcal{A}^{(1)})^{K-1}\boldsymbol{X'}+\cdots+\beta(\vec{1}-\beta)\mathcal{A}^{(1)}\boldsymbol{X'}+\beta\boldsymbol{X'}\right]\\ &= \lim_{K\to\infty}\left[\eta_{K-1}(\mathcal{A}^{(1)})^{K-1}\boldsymbol{X'}+\cdots+\eta_1\mathcal{A}^{(1)}\boldsymbol{X'}+\eta_0\boldsymbol{X'}\right]\\ &= \lim_{K\to\infty}\left[\left(\sum_{k=0}^{K-1}\eta_k(\mathcal{A}^{(1)})^k\right)\right]\boldsymbol{X'}\\ &= \mathcal{A}_\eta\boldsymbol{X'}\quad\text{as}\quad\mathcal{A}_\eta=\sum_{k=0}^{\infty}\eta_k(\mathcal{A}^{(1)})^k\quad\text{in (4.28).}\end{aligned} \tag{4.37}$$

Thus, Proposition 2 is proven. □

Chapter 5

Experiments and Results

In this chapter, we establish comprehensive evaluations specifically tailored for assessing xNeuSM. Following this, we perform extensive experiments using six real-world datasets to thoroughly gauge the performance of xNeuSM. The principal aim is to methodically investigate the following research questions (RQs):

- **RQ1**: How does xNeuSM's performance compare to various baseline techniques in subgraph matching?
- **RQ2**: To what extent are the predictions generated by xNeuSM reliable?
- **RQ3**: What impact do changes in pattern size and density have on xNeuSM's performance?
- **RQ4**: How well does xNeuSM handle the task of matching explanation?
- **RQ5**: What is the specific contribution and influence of each element within xNeuSM?
- **RQ6**: Can xNeuSM effectively adapt to new, previously unseen graphs in inductive settings?
- **RQ7**: Does xNeuSM maintain its efficiency in scenarios involving directed subgraph matching and explanation?

Each research question is carefully crafted to explore various aspects of xNeuSM, encompassing its performance, scalability, interpretability, and adaptability under diverse conditions and scenarios. The experiments conducted intend to offer robust empirical evidence in answering these research inquiries, thereby substantiating the efficacy of our framework in the practical drug design process.

5.1 Experimental setup

In this section, we provide an elaborate overview of our experimental framework, covering the choice of datasets, baseline techniques, data pre-processing methods, and the assessment metrics used. This thorough description aims to clarify the details of our experimental configuration, promoting transparency and reproducibility, while offering insights into the fundamental components that define our evaluation approach.

5.1.1 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 [56] within graph mining research. The statistics for these datasets are summarized in Table 5.1. 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.

Table 5.1: 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

5.1.2 Baseline techniques

In this thesis, we conduct a comprehensive comparison of our proposed method against several state-of-the-art approaches, spanning both traditional exact subgraph matching methods and modern approximate techniques leveraging GNNs. Specifically, for exact subgraph matching, we benchmark our method against VF3 [32], TurboISO [12], CFL [13], CECI [14], QuickSI [11], DAF [15], and

GraphQL [18]. For modern GNN-based approaches, we compare against two state-of-the-art methods known for their strong performance including NeuralMatch [16] and DualMP [17].

5.1.3 Data preparation

Within our experimental setup, we employ separate training and testing datasets. The testing dataset comprises actual instances from the real world, while the training dataset is artificially generated to match the size and distribution of degrees found in the respective testing dataset.

For every graph present in the testing dataset $\mathcal{D}$, we generate 2000 queries, half of which align with the graphs isomorphically, and vice versa. These queries range in size from 2 to the size of the data graph, adhering to a distribution that is uniform. The average degrees of these queries follow a Normal distribution of node degree within the dataset $\mathcal{N}(\overline{deg}, \sigma^2_{\mathcal{D}}(deg))$.

For each actual dataset, we construct a synthetic training dataset mirroring the graph size and degree distribution. We follow the same procedure to generate 2000 queries for each target graph, mirroring the approach used in the testing set. The number of target graphs within these synthetic training datasets is four times larger than that in the real datasets.

5.1.4 Metrics

Subgraph matching task: In this task, we perform a comprehensive comparative analysis of our proposed approach against exact and approximate methods using various metrics to thoroughly evaluate its runtime and performance. The metrics employed in this assessment include:

- *Execution time*: This metric signifies the average processing time for a query (target graph, query pattern), excluding disk loading time.
- *ROC AUC*: Representing model performance, the ROC curve illustrates the True Positive Rate (TPR) versus the False Positive Rate (FPR) at different thresholds. The AUC measures overall performance based on the curve's area, ranging from 0 to 1, where higher values indicate superior performance.
- *PR AUC*: Utilizing Precision-Recall curves, this metric emphasizes Precision and Recall rates over TPR and FPR. It quantifies the area under the

Precision-Recall curve, offering a more precise assessment, especially for imbalanced datasets. Higher PR AUC values reflect better performance.

- *F1 score*: Calculated as the harmonic mean of Precision and Recall, the F1 score provides a balanced evaluation of the model's performance. Precision assesses accuracy in identifying positive instances, while Recall evaluates the ability to capture all actual positives.

Matching explanation task: To showcase the effectiveness of identifying mappings of isomorphic subgraphs, we retrieve the 1-hop attention matrix from the final GLeMA layer in the section involving inter-graph connections. Afterward, we prioritize the mappings of each query node within the transaction based on their individual attention scores. We utilize the following two metrics for assessment:

- *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 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.

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

5.1.5 Hyperparameters and Reproducibility

We provide our hyperparameter settings for training our xNeuSM in Table 5.2.

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.git`.

Table 5.2: 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

5.2 Subgraph Matching Results

To address the primary research question (**RQ1**), we extensively evaluate xNeuSM's performance holistically, specifically focusing on key aspects: (i) execution time and (ii) its efficacy in subgraph matching tasks. We gauge its performance using four key metrics: ROC AUC, PR AUC, F1 score, and accuracy.

Execution Time across real datasets. The outcomes, as shown in Figure 5.1, unequivocally indicate that our proposed solution, xNeuSM, outperforms all state-of-the-art exact and approximate methods in terms of execution time. This emphasizes the superior scalability of xNeuSM, enabling it to handle larger graphs efficiently.

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 $O(L_G(|E_\mathcal{T}| + |E_\mathcal{P}|) + |V_\mathcal{T}| \times |V_\mathcal{P}|)$ [16]. 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 graphs 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 [11], [12]. From the above analysis, we can conclude that our approach exhibits one of the lowest time complexities, resulting in faster runtimes compared to most 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 5.4).

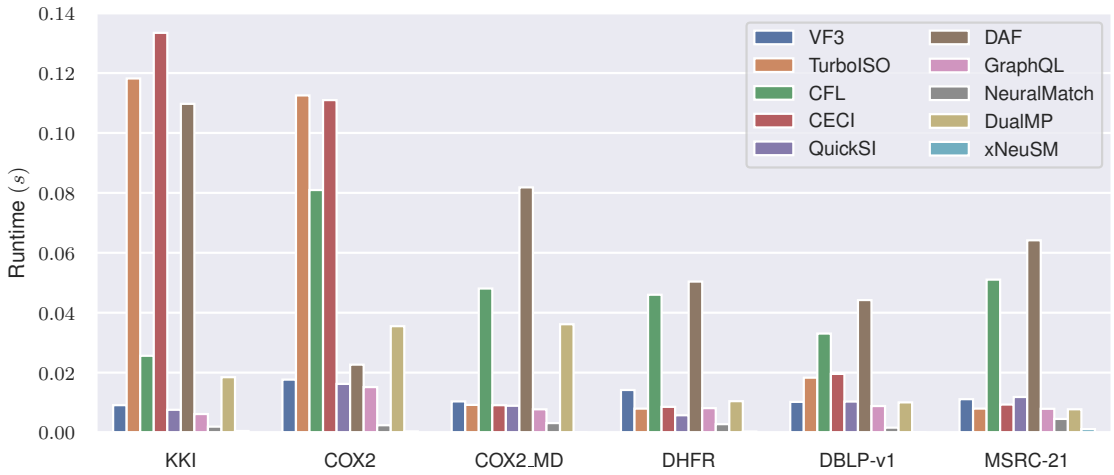


Figure 5.1: Execution time on subgraph matching task. Our proposed method, xNeuSM, is represented in light blue and demonstrates competitive performance with significantly lower runtime compared to most baseline methods.

Performance across real datasets. The benchmarking results of our xNeuSM and the approximate approach are displayed in Figure 5.2. The results undeniably confirm that our approach attains performance levels nearly equivalent to exact methods and surpasses the current state-of-the-art (SOTA) performance of the approximate method across all examined real datasets. This highlights the adaptability of our approach, rendering it suitable for various real-world applications like pattern matching in social networks, identifying compounds with specific activities, and beyond.

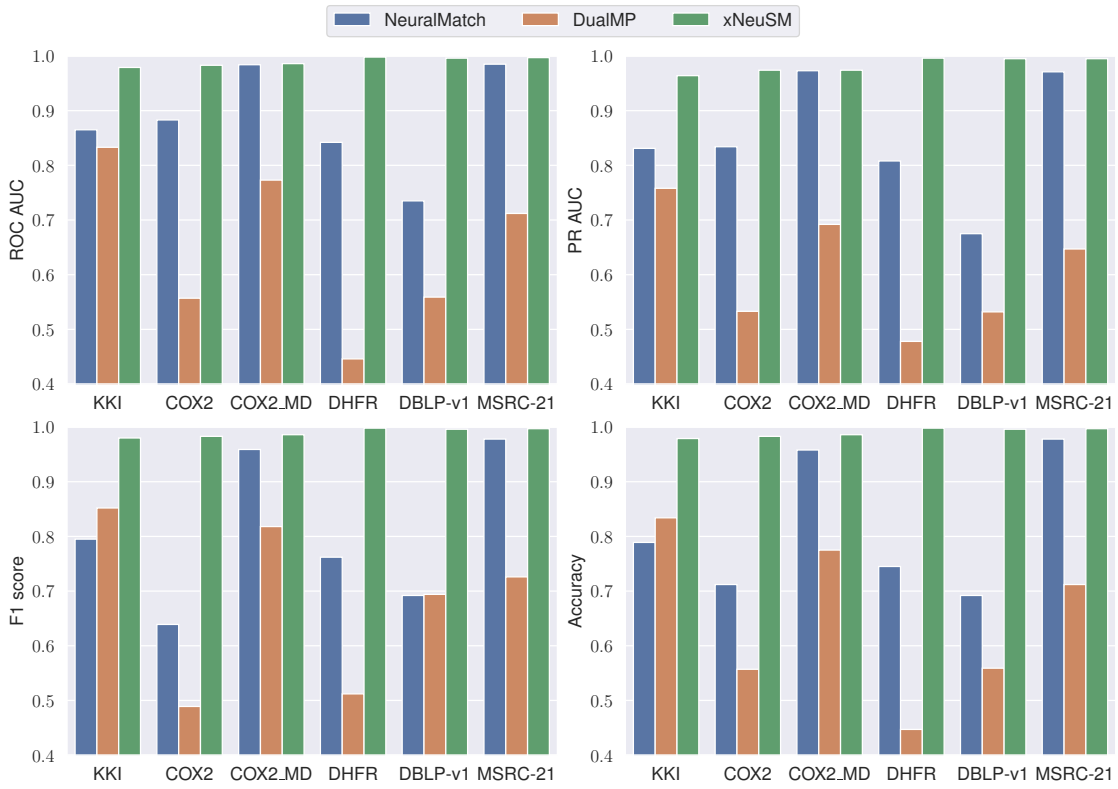


Figure 5.2: Performance comparison between xNeuSM and approximate methods. The evaluation metrics include ROC AUC (top-left), PR AUC (top-right), F1 score (bottom-left), and Accuracy (bottom-right). xNeuSM consistently achieves the highest scores across all datasets and metrics.

5.3 Confidence Analysis

In this section, we evaluate the model confidence in its predicted outputs to address **RQ2**. We use the predicted probability as a measure of confidence. Our analysis demonstrates that the model maintains high performance even as the output probability threshold for subgraph matching increases. The relationship between the confidence threshold and model performance is depicted in Figure 5.3. Notably, at a confidence threshold of 0.9, our model achieves over 90% across all evaluation metrics on the test datasets. This result highlights the robustness of xNeuSM in its predictions.

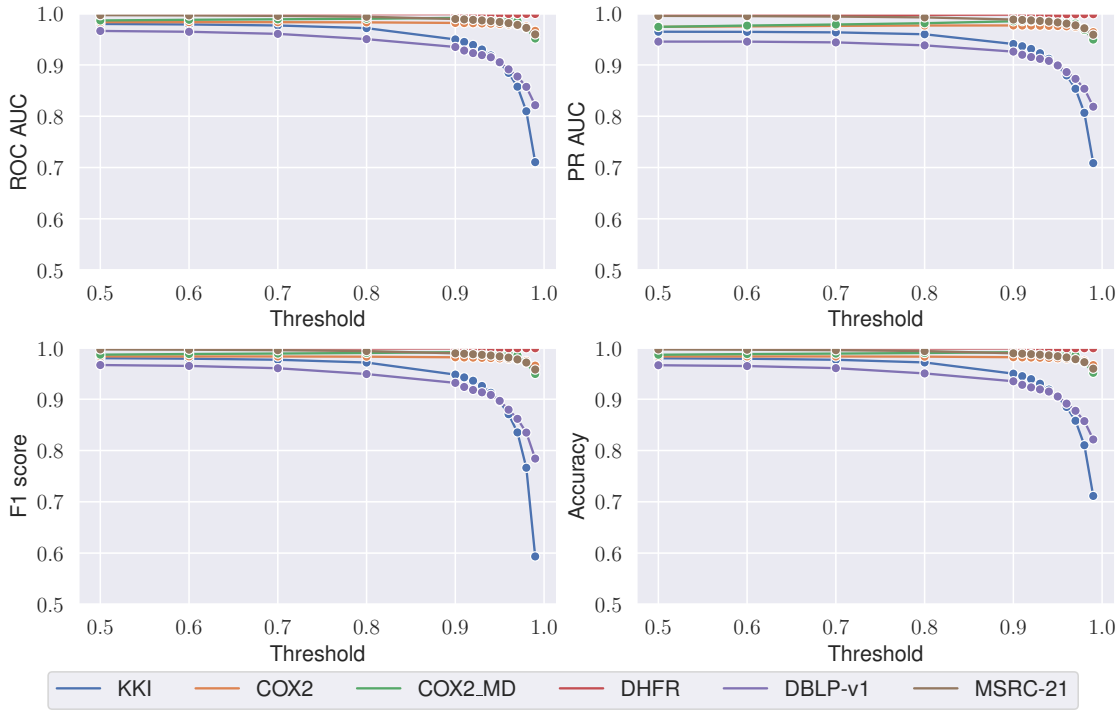


Figure 5.3: Relation between confidence threshold and model performance. The four subplots depict key evaluation metrics: RROC AUC (top-left), PR AUC (top-right), F1 score (bottom-left), and Accuracy (bottom-right). As the confidence threshold increases, xNeuSM maintains high performance across all metrics. Notably, for thresholds up to 0.9, performance remains stable, demonstrating the robustness of the model's predictions.

5.4 Scalability

To evaluate the scalability of xNeuSM and address research question **RQ3**, we assessed the performance of all techniques across various real datasets with different graph density levels, spanning from sparse to dense graphs.

- Vary $D(P)$: We segregated queries into two subsets based on their average degree. The "dense" subset encompassed queries with a degree of three or higher, while the "sparse" subset included queries with a degree less than three.
- Vary $|V_P|$: We categorized the query set into four groups, distinguished by query size thresholds: $|V_P| \leq 20$, $20 < |V_P| \leq 40$, $40 < |V_P| \leq 60$, and $60 < |V_P|$.

Our findings, presented in Figure 5.4, showcase runtime data on a logarithmic scale. The results indicate that exact methods encounter a considerable rise in runtime as the number of query nodes increases. Certain methods, like CECI or CFL, demonstrate a heightened sensitivity to the number of query nodes. Conversely, although experiencing increased time requirements, our methods display relatively modest increments due to the parallelizability of all operations using GPU. In short, our approach proves to be more efficient in large-scale settings.

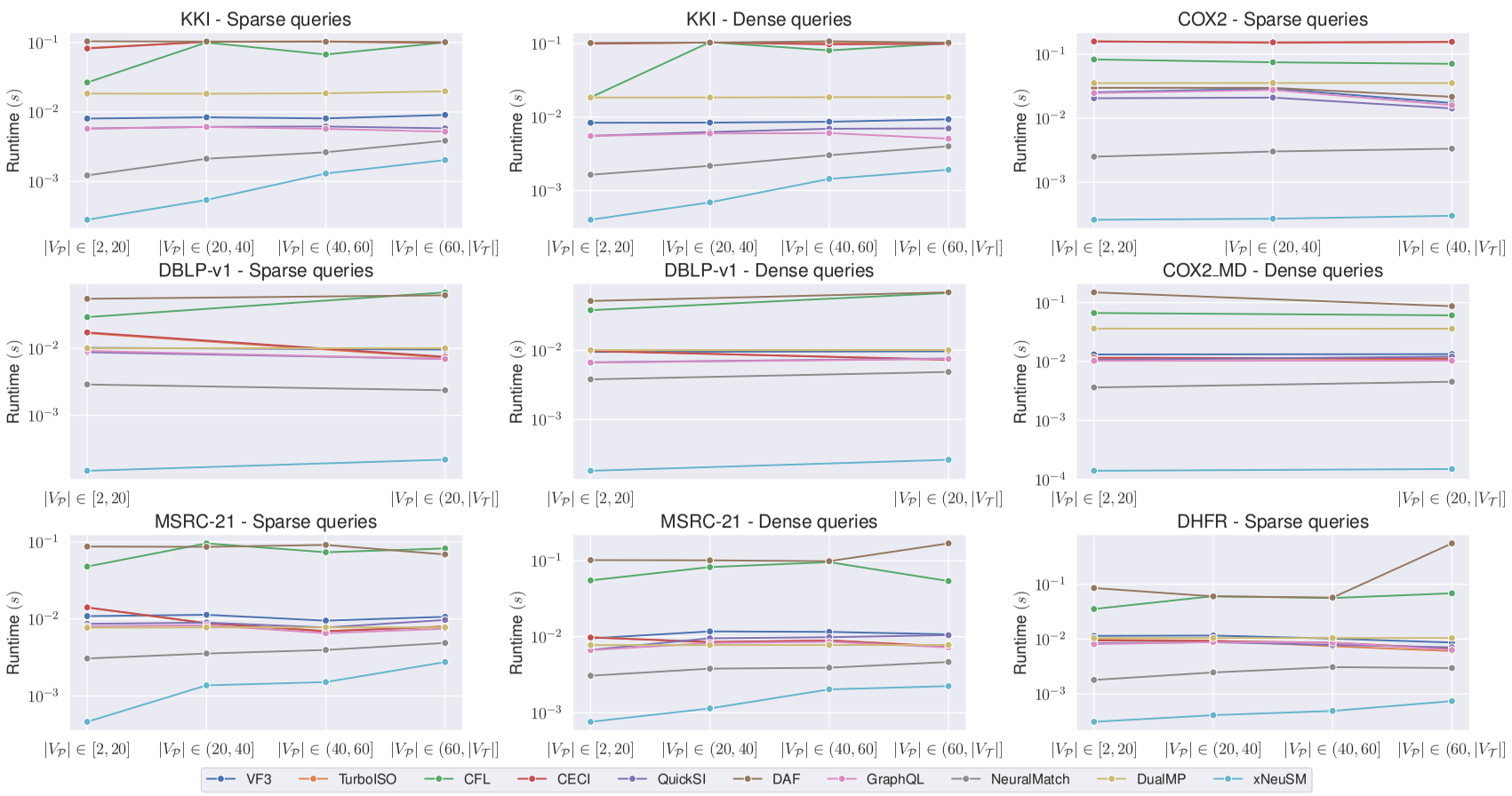


Figure 5.4: Comparison of runtimes between subgraph matching methods across varying query sizes, densities, and different datasets. The y-axis represents runtime (log scale), demonstrating the scalability of each method. xNeuSM (cyan) consistently achieves the lowest runtime across all settings, highlighting its efficiency in handling both sparse and dense query graphs compared to other approximate and exact baselines.

5.5 Matching Explanation

Quantitative analysis. To address **RQ4**, we conducted experiments focusing on the matching explanation task. It is essential to note that this task was exclusively applied to known isomorphism pairs of (pattern, target). We deliberately excluded non-isomorphic cases from our testing as they might not reflect real-world scenarios. Once the isomorphism was established, we proceeded with subgraph mapping. In this task, we computed attention scores for all transaction nodes relative to 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 efficacy of subgraph alignment was evaluated and is presented in Table 5.3.

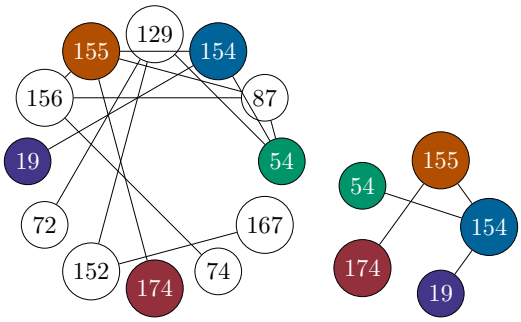
Table 5.3 showcases the superior performance of our method in the subgraph alignment task across various datasets such as KKI, and COX2_MD. However, dealing with a limited set of node labels presents increased challenges in this task, potentially complicating the retrieval of accurate subgraph mappings. This complexity arises from the exponential growth in the potential number of mappings.

Table 5.3: Performance of in subgraph aligning task

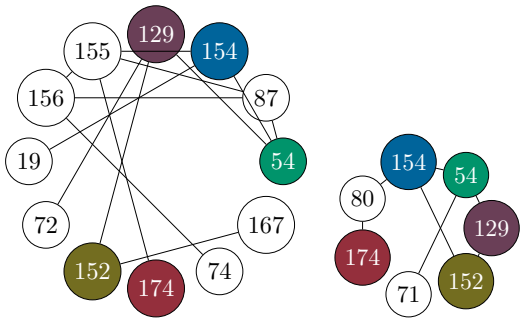
Dataset	Top-1↑	Top-5↑	Top-10↑	MRR↑
KKI	0.9978	0.9999	0.9999	0.9987
COX2	0.2513	0.6259	0.8395	0.4273
COX2_MD	0.9481	0.9828	0.9881	0.9630
DHFR	0.9999	0.9999	0.9999	0.9999
DBLP-v1	0.9994	0.9999	0.9999	0.9996
MSRC-21	0.9994	0.9999	0.9999	0.9999

Qualitative analysis. For the qualitative assessment of the matching explanation task, we have created visual representations for two instances from the KKI dataset. Specifically, Figure 5.5a showcases an isomorphic scenario, while Figure 5.5b depicts a non-isomorphic case. Each node within these figures is denoted by a numerical label, and nodes predicted to align are uniformly color-coded. Our model excels in the isomorphic case, accurately predicting all node mappings. In contrast, in the non-isomorphic case shown in Figure 5.5b, the model generates potential mappings for all subgraphs within the pattern. For example, the subgraph $154 - 54 - 129 - 152$ can be altered into an isomorphic subgraph of the

target within the pattern by removing the edge $154 - 152$.



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



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

Figure 5.5: Examples of isomorphism and non-isomorphism cases resulted from our model in the KKI dataset

5.6 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 experiment with this setting using the KKI, COX2, and DBLP-v1 datasets against several variants of xNeuSM.

- **Model Architecture:**
  - `cross-only`: This configuration exclusively employs interconnections 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.
- **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 Graph Attention Network 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 outcomes detailed in Table 5.4, 5.5, and 5.6 offer compelling support for the effectiveness of our proposed model design, demonstrating an optimal balance between performance and computational efficiency. These results underscore the significance of incorporating intra-connections to achieve exceptional performance levels. The combination of inter- and intra-connections significantly enhances the model's capability to discern unaligned nodes, resulting in superior performance

compared to utilizing either intra- or inter-connections in isolation. Moreover, increasing the number of hops allows the model to capture the global structure of graphs more effectively, contributing to enhanced performance. However, a continuous rise in the number of hops leads to a slowdown in the model's speed. The interleaved-hop strategy emerges as the most viable option, maintaining high performance while reducing computational time. Regarding the number of attention heads, a larger number leads to longer runtimes and a higher risk of overfitting. We observe that in three datasets—KKI, COX2, and DBLP-v1—the model appears to overfit when the number of attention heads increases. This suggests that careful hyperparameter tuning is necessary when applying xNeuSM to real-world problems.

Table 5.4: Impact of xNeuSM components on KKI dataset

Model	**Time↓**	**ROC↑**	**PR↑**	**F1↑**	**Acc↑**	**MRR↑**
Cross-only 1-hop	0.56	0.979	0.959	0.979	0.979	**0.999**
Cross-only increasing-hop	0.60	0.977	0.956	0.977	0.977	0.996
Cross-only interleaved-hop	0.42	0.978	0.958	0.978	0.978	0.997
Intra-only 1-hop	0.49	0.611	0.578	0.485	0.612	–
Intra-only increasing-hop	0.40	0.628	0.593	0.515	0.628	–
Intra-only interleaved-hop	0.42	0.669	0.626	0.602	0.670	–
Both 1-hop	0.62	0.968	0.939	0.968	0.968	**0.999**
Both increasing-hop	0.70	**0.980**	0.963	**0.980**	**0.980**	**0.999**
Both interleaved-hop	0.51	0.979	**0.964**	**0.980**	0.979	0.998
With 2 attention heads	0.86	0.956	0.935	0.954	0.957	0.998
With 4 attention heads	1.19	0.938	0.923	0.936	0.937	**0.999**

Table 5.5: Impact of xNeuSM components on COX2 dataset

Model	**Time↓**	**ROC↑**	**PR↑**	**F1↑**	**Acc↑**	**MRR↑**
Cross-only 1-hop	0.12	0.967	0.946	0.968	0.967	0.306
Cross-only increasing-hop	0.12	0.962	0.931	0.964	0.962	0.191
Cross-only interleaved-hop	0.11	0.974	0.953	0.974	0.974	0.203
Intra-only 1-hop	0.11	0.472	0.498	0.007	0.472	–
Intra-only increasing-hop	0.11	0.457	0.499	0.003	0.458	–
Intra-only interleaved-hop	0.12	0.491	0.495	0.180	0.491	–
Both 1-hop	0.12	0.962	0.950	0.961	0.962	0.177
Both increasing-hop	0.39	0.972	0.961	0.972	0.972	0.298
Both interleaved-hop	0.38	**0.983**	**0.974**	**0.984**	**0.983**	**0.427**
With 2 attention heads	0.49	0.954	0.925	0.955	0.954	0.298
With 4 attention heads	0.63	0.907	0.892	0.900	0.907	0.215

Table 5.6: Impact of xNeuSM components on DBLP-v1 dataset

Model	**Time↓**	**ROC↑**	**PR↑**	**F1↑**	**Acc↑**	**MRR↑**
Cross-only 1-hop	0.07	**0.996**	**0.995**	0.996	**0.996**	0.996
Cross-only increasing-hop	0.09	0.980	0.964	0.981	0.980	0.983
Cross-only interleaved-hop	0.08	0.980	0.963	0.981	0.980	0.985
Intra-only 1-hop	0.09	0.640	0.598	0.579	0.640	–
Intra-only increasing-hop	0.13	0.643	0.593	0.634	0.643	–
Intra-only interleaved-hop	0.09	0.618	0.576	0.573	0.618	–
Both 1-hop	0.09	**0.996**	0.992	0.996	**0.996**	0.995
Both increasing-hop	0.10	0.918	0.910	0.912	0.918	0.989
Both interleaved-hop	0.13	**0.996**	**0.995**	**0.997**	**0.996**	**0.999**
With 2 attention heads	0.17	0.976	0.964	0.975	0.976	**0.999**
With 4 attention heads	0.28	0.986	0.984	0.985	0.986	**0.999**

5.7 Generalisation Analysis

In this section, we conduct experiments to showcase xNeuSM's ability to generalize in out-of-distribution scenarios, addressing research question **RQ6**. In these scenarios, we deploy the model trained on one dataset to test it on others from the same set used in previous experiments. The test results are outlined in Table 5.7. Upon analysis of Table 5.7, several observations emerge:

- 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, COX2_MD).
- 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.

Table 5.7: 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**

5.8 Directed Subgraph Matching and Explanation

To address the final inquiry (**RQ7**), we evaluated the performance of our xNeuSM on directed graphs using the same datasets employed in previous experiments. For this assessment, we transformed all edges in these datasets into directed edges, assigning the tail node as the one with a smaller label and the head node as the one with a higher label. We then present the outcomes of this evaluation concerning both subgraph matching and matching explanation tasks in Table 5.8. The findings displayed in Table 5.8 affirm the efficacy of our proposed approach, indicating its effectiveness irrespective of whether the graph is directed or undirected.

Table 5.8: Performance of xNeuSM in directed subgraph matching and matching explanation. The results demonstrate that our method remains effective regardless of whether the graph is directed or undirected.

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

Chapter 6

Conclusion

6.1 Summary

This study introduced xNeuSM, a pioneering framework designed to enhance the interpretability of neural-based subgraph matching while surpassing existing algorithmic performance. Contributions encompass Graph Learnable Multi-hop Attention Networks and a multi-task learning framework for concurrent optimization of subgraph matching and explanation tasks. Theoretical justifications were provided, analyzing the approximation error of multi-hop attention and validating node-specific attention decays.

Extensive experiments conducted on real-world datasets (focusing on chemistry and bioinformatics) illustrated that xNeuSM significantly outperforms current techniques in both runtime and accuracy when it comes to subgraph matching. Its proficiency extends to accurately identifying node correspondences, evident from the matching explanation results. Additional studies isolating specific components in xNeuSM's architecture verified their effectiveness. This research delved into assessing xNeuSM's adaptability, scalability, and suitability for directed graphs and inductive settings. Overall, xNeuSM effectively balances enhanced performance and interpretability, positioning it as a pragmatic solution for diverse real-world tasks involving large graph subgraph matching and pattern analysis, benefitting the process of drug design.

6.2 Future Developments

Future directions for the development of xNeuSM involve enhancing its capabilities for inexact subgraph matching by incorporating techniques that account for structural variability and noise in real-world graphs. This will involve designing sophisticated algorithms to better align subgraphs that may differ in size, topology, or attributes. To augment its representational power, future work will integrate advanced GNN modules such as message-passing layers that encode higher-order interactions, attention mechanisms for identifying key substructures, and neural operators for dynamic graph embeddings. Additionally, xNeuSM's scope will be expanded to tackle domain-specific challenges in drug discovery, such as predicting drug-target interactions and modeling molecular pathways, as well as network alignment tasks like detecting isomorphic mappings across biological and social networks. These improvements will prioritize interpretability, allowing users to trace and understand model predictions while ensuring scalability to handle large, complex graphs. Ultimately, these efforts aim to establish xNeuSM as a comprehensive, efficient, and interpretable framework for addressing a wide spectrum of graph-related problems in both scientific and industrial domains.

References

[1] D. Q. Nguyen, T. Toan Nguyen, J. Jo, F. Poux, S. Anirban, and T. T. Quan, "Explainable Neural Subgraph Matching With Learnable Multi-Hop Attention," *IEEE Access*, vol. 12, pp. 130 474–130 492, 2024. DOI: 10.1109/ACCESS.2024.3458050.

[2] A. Droschinsky, L. Humbeck, O. Koch, N. M. Kriege, P. Mutzel, and T. Schäfer, "Graph-based methods for rational drug design," in *Algorithms for Big Data: DFG Priority Program 1736*. Cham: Springer Nature Switzerland, 2022, pp. 76–96, ISBN: 978-3-031-21534-6. DOI: 10.1007/978-3-031-21534-6_5.

[3] 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," *The VLDB Journal*, vol. 29, no. 2, pp. 595–618, 2020, ISSN: 0949-877X. DOI: 10.1007/s00778-019-00548-x.

[4] S. P. Nandanoori, S. Guan, S. Kundu, *et al.*, "Graph Neural Network and Koopman Models for Learning Networked Dynamics: A Comparative Study on Power Grid Transients Prediction," *IEEE Access*, vol. 10, pp. 32 337–32 349, 2022. DOI: 10.1109/ACCESS.2022.3160710.

[5] W. Fan, "Graph pattern matching revised for social network analysis," in *Proceedings of the 15th International Conference on Database Theory*, ser. ICDT '12, Berlin, Germany: Association for Computing Machinery, 2012, pp. 8–21, ISBN: 9781450307918. DOI: 10.1145/2274576.2274578.

[6] 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. 168 344–168 360, 2020. DOI: 10.1109/ACCESS.2020.3022850.

[7] Y. Zhang, Q. Yao, L. Yue, *et al.*, "Emerging drug interaction prediction enabled by a flow-based graph neural network with biomedical network," *Nature Computational Science*, vol. 3, no. 12, pp. 1023–1033, 2023, ISSN: 2662-8457. DOI: 10.1038/s43588-023-00558-4.

[8] O. Zhang, T. Wang, G. Weng, *et al.*, "Learning on topological surface and geometric structure for 3d molecular generation," *Nature Computational Science*, vol. 3, no. 10, pp. 849–859, 2023, ISSN: 2662-8457. DOI: 10.1038/s43588-023-00530-2.

[9] I. Roy, V. S. B. R. Velugoti, S. Chakrabarti, and A. De, "Interpretable Neural Subgraph Matching for Graph Retrieval," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, Virtual Event, Jun. 2022, pp. 8115–8123. DOI: 10.1609/aaai.v36i7.20784.

[10] 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. 34 868–34 881, 2021. DOI: 10.1109/ACCESS.2021.3060385.

[11] H. Shang, Y. Zhang, X. Lin, and J. X. Yu, "Taming verification hardness: An efficient algorithm for testing subgraph isomorphism," *Proc. VLDB Endow.*, vol. 1, no. 1, pp. 364–375, Aug. 2008, ISSN: 2150-8097. DOI: 10.14778/1453856.1453899.

[12] W.-S. Han, J. Lee, and J.-H. Lee, "TurboISO: Towards ultrafast and robust subgraph isomorphism search in large graph databases," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '13, New York, NY, USA: Association for Computing Machinery, 2013, pp. 337–348, ISBN: 9781450320375. DOI: 10.1145/2463676.2465300.

[13] F. Bi, L. Chang, X. Lin, L. Qin, and W. Zhang, "Efficient subgraph matching by postponing cartesian products," in *Proceedings of the 2016 International Conference on Management of Data*, ser. SIGMOD '16, San Francisco, CA, USA: Association for Computing Machinery, 2016, pp. 1199–1214, ISBN: 9781450335317. DOI: 10.1145/2882903.2915236.

[14] B. Bhattarai, H. Liu, and H. H. Huang, "Ceci: Compact embedding cluster index for scalable subgraph matching," in *Proceedings of the 2019 International Conference on Management of Data*, ser. SIGMOD '19, Amsterdam, Netherlands: Association for Computing Machinery, 2019, pp. 1447–1462, ISBN: 9781450356435. DOI: 10.1145/3299869.3300086.

[15] 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 *Proceedings of the 2019 International Conference on Management of Data*, ser. SIGMOD '19, Amsterdam, Netherlands: Association for Computing Machinery, 2019, pp. 1429–1446, ISBN: 9781450356435. DOI: 10.1145/3299869.3319880.

[16] R. Ying, Z. Lou, J. You, C. Wen, A. Canedo, and J. Leskovec, "Neural Subgraph Matching," 2020. arXiv: 2007.03092. [Online]. Available: https://arxiv.org/abs/2007.03092.

[17] X. Liu and Y. Song, "Graph Convolutional Networks with Dual Message Passing for Subgraph Isomorphism Counting and Matching," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, Virtual Event, Jun. 2022, pp. 7594–7602. DOI: 10.1609/aaai.v36i7.20725.

[18] H. He and A. K. Singh, "Graphs-at-a-time: Query language and access methods for graph databases," in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '08, Vancouver, Canada: Association for Computing Machinery, 2008, pp. 405–418, ISBN: 9781605581026. DOI: 10.1145/1376616.1376660.

[19] K. Xu, L. Wang, M. Yu, *et al.*, "Cross-lingual Knowledge Graph Alignment via Graph Matching Neural Network," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, A. Korhonen, D. Traum, and L. Màrquez, Eds., Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 3156–3161. DOI: 10.18653/v1/P19-1304.

[20] J. Xu, T. L. Wickramarathne, and N. V. Chawla, "Representing higher-order dependencies in networks," *Science Advances*, vol. 2, no. 5, e1600028, 2016. DOI: 10.1126/sciadv.1600028.

[21] G. Wang, R. Ying, J. Huang, and J. Leskovec, "Multi-hop Attention Graph Neural Networks," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, Z.-H. Zhou, Ed., Main Track, Virtual Event, Montreal, Canada: International Joint Conferences on Artificial Intelligence Organization, Aug. 2021, pp. 3089–3096. DOI: 10.24963/ijcai.2021/425.

[22] M. M. SysŁ;o, "The subgraph isomorphism problem for outerplanar graphs," *Theoretical Computer Science*, vol. 17, no. 1, pp. 91–97, 1982, ISSN: 0304-3975. DOI: 10.1016/0304-3975(82)90133-5.

[23] H. Shiokawa, Y. Naoi, and S. Matsugu, "Efficient correlated subgraph searches for ai-powered drug discovery," in *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, K. Larson, Ed., Main Track, Jeju, Korea: International Joint Conferences on Artificial Intelligence Organization, Aug. 2024, pp. 2351–2361. DOI: 10.24963/ijcai.2024/260.

[24] Y. Wang, Y. Xia, J. Yan, Y. Yuan, H.-B. Shen, and X. Pan, "Zerobind: A protein-specific zero-shot predictor with subgraph matching for drug-target interactions," *Nature Communications*, vol. 14, no. 1, p. 7861, 2023, ISSN: 2041-1723. DOI: 10.1038/s41467-023-43597-1.

[25] D. Rose, O. Wieder, T. Seidel, and T. Langer, "Pharmacomatch: Efficient 3d pharmacophore screening via neural subgraph matching," in *Proceedings of the 13th International Conference on Learning Representations*, Expo Drive, Singapore, Apr. 2025.

[26] Y. Ye, X. Lian, and M. Chen, "Efficient exact subgraph matching via gnn-based path dominance embedding," *Proc. VLDB Endow.*, vol. 17, no. 7, pp. 1628–1641, Mar. 2024, ISSN: 2150-8097. DOI: 10.14778/3654621.3654630.

[27] H. Du, Q. Yao, J. Zhang, Y. Liu, and Z. Wang, "Customized subgraph selection and encoding for drug-drug interaction prediction," in *Advances in Neural Information Processing Systems*, A. Globerson, L. Mackey, D. Belgrave, *et al.*, Eds., vol. 37, Vancouver, Canada: Curran Associates, Inc., 2024, pp. 109 582–109 608.

[28] R. Alizadehsani, S. S. Oyelere, S. Hussain, *et al.*, "Explainable artificial intelligence for drug discovery and development: A comprehensive survey," *IEEE Access*, vol. 12, pp. 35 796–35 812, 2024. DOI: 10.1109/ACCESS.2024.3373195.

[29] A. V. Geevarghese, "Explainable artificial intelligence in drug discovery," in *Explainable AI in Health Informatics*. Singapore: Springer Nature Singapore, 2024, pp. 113–134, ISBN: 978-981-97-3705-5. DOI: 10.1007/978-981-97-3705-5_6.

[30] 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 *Proceedings of the 2021 International Conference on Management of Data*, ser. SIGMOD '21, Virtual Event, China: Association for Computing Machinery, 2021, pp. 925–937, ISBN: 9781450383431. DOI: 10.1145/3448016.3457265.

[31] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. USA: W. H. Freeman & Co., 1990, ISBN: 0716710455.

[32] 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: Springer International Publishing, 2017, pp. 128–139, ISBN: 978-3-319-58961-9. DOI: 10.1007/978-3-319-58961-9_12.

[33] L. Cordella, P. Foggia, C. Sansone, and M. Vento, "A (sub)graph isomorphism algorithm for matching large graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 10, pp. 1367–1372, 2004. DOI: 10.1109/TPAMI.2004.75.

[34] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009. DOI: 10.1109/TNN.2008.2005605.

[35] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *Proceedings of the 5th International Conference on Learning Representations*, Toulon, France, Apr. 2017.

[36] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17, Long Beach, CA, USA: Curran Associates Inc., 2017, pp. 1025–1035, ISBN: 9781510860964.

[37] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How Powerful are Graph Neural Networks?" In *Proceedings of the 7th International Conference on Learning Representations*, New Orleans, LA, USA, May 2019.

[38] Y. Bai, H. Ding, Y. Sun, and W. Wang, "Convolutional set matching for graph similarity," in *NeurIPS 2018 Relational Representation Learning Workshop*, ser. NIPS'18, Montréal, Canada: Curran Associates Inc., 2018.

[39] Z. Zhang and W. S. Lee, "Deep Graphical Feature Learning for the Feature Matching Problem," in *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea, 2019, pp. 5086–5095. DOI: 10.1109/ICCV.2019.00519.

[40] H. Yuan, J. Tang, X. Hu, and S. Ji, "XGNN: Towards Model-Level Explanations of Graph Neural Networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '20, Virtual Event, CA, USA: Association for Computing Machinery, 2020, pp. 430–438, ISBN: 9781450379984. DOI: 10.1145/3394486.3403085.

[41] M. Vu and M. T. Thai, "PGM-Explainer: Probabilistic Graphical Model Explanations for Graph Neural Networks," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Virtual Event: Curran Associates, Inc., 2020, pp. 12 225–12 235.

[42] F. Wu, S. Li, X. Jin, *et al.*, "Rethinking Explaining Graph Neural Networks via Non-parametric Subgraph Matching," in *Proceedings of the 40th International Conference on Machine Learning*, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., ser. ICML'23, vol. 202, Honolulu, HI, USA: PMLR, Jul. 2023, pp. 37 511–37 523.

[43] D. Yang, Y. Ge, T. Nguyen, D. Molitor, J. D. Moorman, and A. L. Bertozzi, "Structural equivalence in subgraph matching," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 4, pp. 1846–1862, 2023. DOI: 10.1109/TNSE.2023.3236028.

[44] S. Sun and Q. Luo, "Subgraph matching with effective matching order and indexing," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 1, pp. 491–505, 2022. DOI: 10.1109/TKDE.2020.2980257.

[45] Y. Sun, G. Li, J. Du, B. Ning, and H. Chen, "A subgraph matching algorithm based on subgraph index for knowledge graph," *Front. Comput. Sci.*, vol. 16, no. 3, Jun. 2022, ISSN: 2095-2228. DOI: 10.1007/s11704-020-0360-y.

[46] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proceedings of the 6th International Conference on Learning Representations*, Vancouver, Canada, Apr. 2018.

[47] L. Ruiz, F. Gama, and A. Ribeiro, "Gated graph recurrent neural networks," *Trans. Sig. Proc.*, vol. 68, pp. 6303–6318, Jan. 2020, ISSN: 1053-587X. DOI: 10.1109/TSP.2020.3033962.

[48] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning*, ser. ICML'17, vol. 70, Sydney, NSW, Australia: JMLR.org, 2017, pp. 1263–1272.

[49] D. L. Sussman, Y. Park, C. E. Priebe, and V. Lyzinski, "Matched filters for noisy induced subgraph detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 11, pp. 2887–2900, 2020. DOI: 10.1109/TPAMI.2019.2914651.

[50] J. Jiménez-Luna, F. Grisoni, and G. Schneider, "Drug discovery with explainable artificial intelligence," *Nature Machine Intelligence*, vol. 2, no. 10, pp. 573–584, 2020, ISSN: 2522-5839. DOI: 10.1038/s42256-020-00236-4.

[51] 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," *Journal of Chemical Information and Modeling*, vol. 59, no. 9, pp. 3981–3988, 2019, PMID: 31443612. DOI: 10.1021/acs.jcim.9b00387.

[52] T. Luong, H. Pham, and C. D. Manning, "Effective Approaches to Attention-based Neural Machine Translation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 1412–1421. DOI: 10.18653/v1/D15-1166.

[53] J. Gasteiger, A. Bojchevski, and S. Günnemann, "Predict then Propagate: Graph Neural Networks meet Personalized PageRank," in *Proceedings of the 7th International Conference on Learning Representations*, New Orleans, LA, USA, May 2019.

[54] P. Lofgren, "Efficient Algorithms for Personalized PageRank," PhD thesis, Stanford University, 2015.

[55] K. Cho, B. van Merriënboer, C. Gulcehre, *et al.*, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, A. Moschitti, B. Pang, and W. Daelemans, Eds., Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. DOI: 10.3115/v1/D14-1179.

[56] 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 *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, Virtual Event, 2020. [Online]. Available: www.graphlearning.io.

Vita

General information

Full name: NGUYEN QUANG DUC
Date of birth: 24/12/2000 **Place of birth:** An Giang
Address: 287 Truong Chinh St., My Phuoc Ward, Long Xuyen City, An Giang

Qualification

No.	Years	Academic institutions	Major/ Specialty	Degree
1	2018-2022	Ho Chi Minh City University of Technology - VNU-HCMC	Computer Science	Bachelor
1	2023-2025	Ho Chi Minh City University of Technology - VNU-HCMC	Computer Science	Master

Professional Experience

No.	Years	Institutions	Address	Position
1	2023-now	Ho Chi Minh City University of Technology District 10, HCMC	268 Ly Thuong Kiet St., Ward 14,	Teaching Assistant