Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)
факультет інформаційних технологій
(факультет)
Кафедра інформаційних технологій та комп'ютерної інженерії
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня магістра
(бакалавра, спеціаліста, магістра)

Студента Шлянчак Світлани Олександрівни
(ПІБ)
академічної групи 126м-22з-2
(шифр)
спеціальності 126 «Інформаційні системи та технології»
(код і назва спеціальності)
за освітньо-професійною програмою

«Інформаційні системи та технології»
(офіційна назва)
на тему Комплексна кваліфікаційна робота: Дослідження особливостей процесів використання графів знань у системах на базі графових нейронних мереж
(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	проф. Коротенко Г.М.			
розділів:				

Рецензент	доц. Ширін А.Л.			

Нормоконтролер	проф. Коротенко Г.М.			

Дніпро
2024

ЗАТВЕРДЖЕНО:
завідувач кафедри

___інформаційних технологій___
_____та комп'ютерної інженерії___
(повна назва)

_____ _Гнатушенко В.В._
(підпис) (прізвище, ініціали)

«_____»_____ 2024 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня _____магістр _____
(бакалавра, спеціаліста, магістра)

студенту _____Шлянчак С.О.____ **академічної групи** ___126м-22з-2___
(прізвище та ініціали) (шифр)

спеціальності _____126 «Інформаційні системи та технології»_____ _____

за освітньою-професійною програмою _____

«Інформаційні системи та технології»

на тему ____Комплексна кваліфікаційна робота: Дослідження особливостей процесів___
_____використання графів знань у системах на базі графових нейронних мереж_____

затверджену наказом ректора НТУ «Дніпровська політехніка» від 09.10.2021 р. № 1228-с

Розділ	Зміст	Термін виконання
Розділ 1	Аналіз стану області рішення задачі	15.10.2023 – 20.10.2023
Розділ 2	Вивчення особливостей процесів використання графів знань	21.10.2023 – 30.11.2023
Розділ 3	Виконання досліджень та оформлення кваліфікаційної роботи	01.12.2023 – 14.01.2024

Завдання видано _____ _____Коротенко Г.М._____
(підпис керівника) (прізвище, ініціали)

Дата видачі _____15.10.2023 р._____

Дата подання до екзаменаційної комісії ____17.01.2024 р.____

Прийнято до виконання _____ _____Шлянчак С.О._____
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: стор. 69, рис. 16, додатки 4, джерела 37.

Об'єкт дослідження: процес використання графів знань для навчання графових нейронних мереж.

Предмет дослідження: різноманіття архітектур графових нейронних мереж та спосіб використання графів знань для розв’язання задач ізоморфізму графів.

Мета кваліфікаційної роботи: дослідити особливості побудови різноманітних архітектур графових нейронних мереж, а також процес розв’язання задач ізоморфізму графів чи їх частин класичними методами та за допомогою графової нейронної мережі GIN.

Новизна отриманих результатів полягає у дослідженні впливу використання графових нейронних мереж на швидкість та точність розв’язання задач ізоморфізму графів та їх частин.

Пояснювальна записка містить огляд різних архітектур графових нейронних мереж, опис їхніх особливостей, опис класичного алгоритму WL-тест, опис графової нейронної мережі GIN та способу підготовки вхідних даних для її використання, а також спеціалізованого методу навчання graphcontrastivelearning для підвищення якості розв’язання задачі ізоморфізму графів.

Список ключових слів: граф знань, WL-тест, graphcontrastivelearning, GIN, embedding, GNN, ізоморфізм, підграф, граф.

ABSTRACT

Explanatory note: pages 69, figures 16, applications 4, sources 37.

Object of research: is the process of using knowledge graphs to train graph neural networks.

Subject of research: the variety of graph neural network architectures and how to use knowledge graphs to solve graph isomorphism problems.

Purpose of research: to investigate the features of building various architectures of graph neural networks, as well as the process of solving problems of isomorphism of graphs or their parts using classical methods and the graph neural network GIN.

The novelty of the obtained results lies in the study of the effect of using graph neural networks on the speed and accuracy of solving problems of isomorphism of graphs and their parts.

The explanatory note contains an overview of different architectures of graph neural networks, a description of their features, a description of the classical WL-test algorithm, a description of the GIN graph neural network and how to prepare input data for its use, as well as a specialized training method called graphcontrastivelearning to improve the quality of solving the graph isomorphism problem.

Key words: knowledge graph, WL-test, graphcontrastivelearning, GIN, embedding, GNN, isomorphism, subgraph, graph.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

WL-test – тест Вейсфлера-Лемана

GCL – Graph contrastive learning (контрастне навчання на графах)

GCN - Graph Convolutional Network (граф згорткової мережі)

GIN – Graph Isomorphism Network (мережа ізоморфізму графів)

GNN – Graph Neural Network (графова нейронна мережа)

Embedding - вбудування

Isomorphism – подоба форми

Subgraph - підграф

STGNN – Spatio-Temporal Graph Neural Networks (просторово-часові нейронні мережі

Graph - граф

ВСТУП

Актуальність роботи. Останнім часом широке розповсюдження отримали кілька принципово різних сімейств нейронних мереж, які використовують різні за змістом вхідні дані і ефективно розв’язують широкий спектр складних задач. Одним з новітніх напрямків науки про дані є поява та розвиток графових нейронних мереж. Спектр їхнього вжитку є доволі широким, починаючи від аналізу природніх та біологічних процесів і аж до дослідження процесів у соціумі. Протягом останнього часу з’явилися публікації компанії DeepMind, яка за допомогою графової нейронної мережі взялась прогнозувати новітні хімічні сполуки для створення мікроелектроніки.

Однією з важливих задач, яка виникає в біології, хімії, криміналістиці, соціології тощо, є задача ізоморфізму графів, яка дозволяє встановити ступінь схожості між собою речовин, ситуацій, обставин та інше. Класичних методів для розв’язку задачі ізоморфізму є не надто багато, бо існує метод, що точно та швидко здатен обробляти невеликі графи. Але цей метод не є ефективним для графів великого розміру. Тож розробка способу розв’язання задачі ізоморфізму великих графів є актуальною.

Об’єктом досліджень є процес використання графів знань для навчання графових нейронних мереж.

Предметом досліджень є різноманіття архітектур графових нейронних мереж та спосіб використання графів знань для розв’язання задач ізоморфізму графів.

Мета роботи – дослідити особливості побудови різноманітних архітектур графових нейронних мереж, а також процес розв’язання задач ізоморфізму графів чи їх частин класичними методами та за допомогою графової нейронної мережі GIN.

Пояснювальна записка містить огляд різних архітектур графових нейронних мереж, опис їхніх особливостей, опис класичного алгоритму WL-тест, опис графової нейронної мережі GIN та способу підготовки вхідних

даних для її використання, а також спеціалізованого методу навчання GraphContrastiveLearning для підвищення якості розв’язання задачі ізоморфізму графів.

Постановка задачі

Дослідити особливості різноманітних архітектур графових нейронних мереж. Проаналізувати графову нейронну мережу GIN та класичний метод перевірки ізоморфізму графів WL-test. Дослідити спосіб перетворення графів знань на джерело вхідних даних на GIN та спосіб покращення точності перевірки ізоморфізму графів за допомогою методу навчання GraphContrastiveLearning. Перевірити теоретичні припущення щодо використання GIN та GraphContrastiveLearning на практиці.

Висновок.

В цій атестаційній роботі буде досліджено розмай графових нейронних арзітектур, особливості архітектури GIN та класичний алгоритм WL-test для перевірки ізоморфізму графів, буде досліждено спосіб підготовки графів знань для використання GIN та спосіб покращення точності результатів шляхом застосування GraphContrastiveLearning.

РОЗДІЛ 1. АНАЛІЗ СТАНУ ОБЛАСТІ РІШЕННЯ ЗАДАЧІ

1.1.Знання у реальному світі та в системах штучного інтелекту

У реальному світі знання становлять основу нашого розуміння навколишнього середовища та взаємодії з ним. Вони представляють собою не лише факти та інформацію, але й розуміння, отримане через досвід, навчання та аналіз.

У філософії знання зазвичай визначають як виправдане істинне переконання. Це означає, що для того, щоб щось можна було вважати знанням, воно має бути вірогідним, воно має бути правдою, і має бути вагома причина або обґрунтування для того, щоб вважати це правдою. Філософи обговорювали та вдосконалювали це визначення протягом століть, і існує багато різних точок зору на те, що становить виправдання та істину.

З іншого боку, у контексті інформаційної науки та управління знання визначається як систематизована інформація, яка відображається у вигляді даних, фактів, правил, та окрім простого накопичення, включає в себе аналіз, інтерпретацію та контекстуалізацію для досягнення конкретних цілей або вирішення завдань. Знання у цьому контексті може включати не лише конкретні факти, а й взаємозв'язки між ними, що дозволяє використовувати його для прийняття рішень та розв'язання складних завдань.

Якщо говорити мовою епістемології[9], знання в системах штучного інтелекту стосуються обробленої інформації, яка використовується в процесі обґрунтування, який породжує переконання. Однак ці переконання не обов’язково мають бути правдивими, оскільки навчальні дані, які використовуються, представляють лише частину даних реального світу.

Знання у системах штучного інтелекту зазвичай представлені у евклідовому просторі, кожен елемент або атрибут знання має числове представлення, що дозволяє їх відображати як точки у просторі з фіксованою кількістю вимірів. Це дає можливість застосовувати різноманітні математичні методи для аналізу та обробки цих даних, зокрема методи класифікації,

кластеризації, регресії та інші алгоритми машинного навчання. Однак з розвитком штучного інтелекту зростає кількість систем, де знання генеруються з неевклідових областей, де застосовуються складніші моделі представлення даних, що потребують більш гнучких структур для відображення зв'язків та асоціацій між елементами інформації.

Це зумовило використання графових структур, оскільки вони дозволяють не лише уявно відобразити взаємозв'язки між елементами, а й аналізувати складні мережі взаємозв'язків та патернів. Графові моделі можуть ефективно репрезентувати навіть неструктуровані дані, забезпечуючи можливість аналізувати та використовувати їх для розуміння та прийняття рішень у системах штучного інтелекту. Це відкриває нові можливості для штучного інтелекту: від розвідування даних до рекомендаційних систем, а також у сферах, де важлива контекстуальна інтерпретація даних, таких як медицина, фінанси та багато інших галузей. Графові структури стають не тільки зручним інструментом в аналізі даних, але й потужним засобом розуміння та використання знань для прийняття важливих рішень у реальному часі.

1.2.Графові нейронні мережі

Графом називають структуру даних, яка моделює множину об'єктів (вершин) та їх зв'язків (ребер) (рис. 1.1). Велика виразність графів зумовила зростання досліджень з аналізу графових структур за допомогою методів машинного навчання. Вони використовуються у різних видах систем, включаючи соціальні мережі, фізичні системи, графи знань та інші. Як унікальна неевклідова структура даних для машинного навчання, аналіз графів фокусується на таких завданнях, як класифікація вузлів, прогнозування зв'язків і кластеризація.

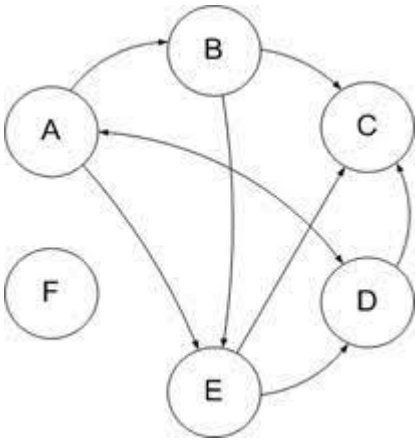

Рис. 1.1. Приклад графу

Графи можна класифікувати за різними критеріями, в залежності від їхньої структури, властивостей або застосувань. Графи зі складною структурою можуть надавати більше інформації про вузли та їхні зв'язки. Зазвичай виділяють наступні категорії:

- Орієнтовані та неорієнтовані графи - характеризуються направленням зв’язків графа. В орієнтованих графах кожне ребро має чітко визначений напрямок з однієї вершини до іншої, вказуючи на відносини або спрямованість взаємозв'язків між ними. У неорієнтованих графах ребра, зазвичай, не мають визначеного напрямку. Їх розглядають як неспрямовані зв'язки між двома вершинами.
- Однорідні та неоднорідні графи. В однорідних графах усі вершини та ребра мають однаковий тип, що означає, що всі елементи графа відносяться до однієї і тієї ж категорії або класу. В свою чергу ребра в однорідному графі мають однакові характеристики, які не відрізняються між собою за своєю природою або властивостями. У неоднорідних графах вершини та ребра можуть мати різні типи. Це означає, що вони можуть бути класифіковані чи розподілені на різні категорії, мають різні характеристики, які відрізняються за певними властивостями або призначеннями.
- Статичні та динамічні графи. Статичні графи мають сталу топологію та незмінні дані протягом усього часу їхнього існування. У таких графах не

відбуваються зміни в структурі вершин та ребер протягом аналізу чи використання графа. Динамічні графи, навпаки, зазнають змін у топології або вхідних даних з часом. Це означає, що структура графа може змінюватися через додавання або видалення вершин та ребер, а також через зміну характеристик або атрибутів цих елементів графа протягом певних проміжків часу.

Вищезазначені категорії є ортогональними, що дозволяє комбінувати їх характеристики між собою. Наприклад, може існувати граф, який одночасно є динамічним (змінюється з часом), орієнтованим (ребра мають напрямок), та неоднорідним (вершини та ребра мають різні типи).

Крім того, в сфері графів існує й інші типи, такі як гіперграфи, які дозволяють представляти більш складні взаємозв'язки між елементами, або підписані графи, де вершинам чи ребрам надаються додаткові атрибути або мітки, що розширює можливості аналізу та розуміння даних у графічній структурі. Комбінація цих різних типів дозволяє більш точно моделювати та аналізувати різноманітні системи та процеси у різних галузях.

Клас методів та підходів машинного навчання, які спеціалізуються на обробці графічних даних та структур, має назву графових нейронних мереж або GNN[3]. Ці методи використовують нейронні мережі для аналізу та роботи з інформацією, яка представлена у вигляді графа, де вершини відображають об'єкти, а ребра - їхні зв'язки. В останні роки GNN стали популярним інструментом у різних областях, від рекомендаційних систем до біології та фінансів, оскільки вони дозволяють ефективно використовувати та аналізувати графові структури для різноманітних завдань машинного навчання.

Вперше нейронні мережі були застосовані до орієнтованих ациклічних графів у роботі Спердуті та ін. [4], що стало поштовхом до ранніх досліджень GNN. Концепція графових нейронних мереж була вперше висвітлена в роботі Горі та інших у 2005 році [6]. Пізніше цей підхід був розглянутий в роботах Скарселлі у 2009 році [7] і Галліккіо у 2010 році [8]. Ці ранні дослідження

відносяться до категорії рекурентних графових нейронних мереж (RecGNN). У цих роботах вивчається представлення цільового вузла, у якому інформація розповсюджується через його сусідів у відповідності з ітеративним процесом до досягнення стабільної фіксованої точки. Попри успіх, універсальна ідея цих методів полягає в побудові систем переходу станів на графах та ітерації до збіжності, що обмежує їх розширення та здатність до представлення.

Натхненні успіхом згорткових нейронних мереж (CNN) в області комп'ютерного зору, паралельно розробляється велика кількість методів, які переосмислюють поняття згортки графічних даних. CNN здатні витягувати різномасштабні локалізовані просторові ознаки та компонувати їх для побудови високо виразних репрезентацій, що призвело до прориву майже у всіх сферах машинного навчання та розпочало нову еру глибокого навчання (LeCun et al., 2015) [37].

У сфері графових нейронних мереж ці підходи об'єднані під назвою згорткових графових нейронних мереж (ConvGNN, GCNN, GCN). ConvGNN поділяються на два основних напрямки: спектральні та просторові підходи. Перші дослідження спектральних ConvGNN були представлені в роботі Bruna et al. [5]., де вони розробили метод згортки графів на основі спектральної теорії графів. Дослідження просторових ConvGNN у свою чергу розпочалися набагато раніше. Розглядаючи взаємну залежність графів у 2009 році, Мікелі та його колеги [16] вперше проаналізували цю проблему за допомогою архітектурно складених нерекурсивних шарів, успадкувавши ідеї передачі повідомлень з RecGNN. Однак, незважаючи на важливий внесок цієї роботи в розвиток методів аналізу графів, її значущість пройшла поза увагою дослідників.

Наряду з RecGNN та ConvGNN, протягом останніх кількох років було розроблено численні альтернативні варіації графових нейронних мереж. Серед них особливо варто відзначити графові автокодери (GAE) та просторово-часові графові нейронні мережі (STGNN). Ці системи навчання можуть бути

побудовані на основі RecGNN, ConvGNN або інших нейронних архітектур для моделювання графів.

Сучасні графові нейронні мережі є ключовим інструментом у вирішенні навчальних завдань, які передбачають роботу з графовими даними, що містять багато інформації про зв'язки між елементами. Вони знаходять широке застосування в різних галузях, таких як моделювання фізичних систем, вивчення молекулярних відбитків, прогнозування білкових інтерфейсів і класифікація хвороб. Особливо важливою є їхня роль у навчанні на неструктурних даних, наприклад, у текстах та зображеннях. Зокрема задачі міркування на основі витягнутих структур, таких як дерева залежностей речень або графи сцен зображень, вимагає моделей з графовою структурою для ефективного розв'язання завдань.

1.3. **Різновиди архітектур графових нейронних мереж**

Як було визначено вище графові нейронні мережі - це клас методів машинного навчання, призначених для обробки графових даних, які моделюють зв'язки між об'єктами та базуються на передачі повідомлень між ними, що дозволяє ефективно представляти та аналізувати графові дані. Існують різні архітектури побудови графових нейронних мереж, такі як графові згорткові мережі (GCN), які агрегують інформацію з сусідніх вузлів, графові мережі уваги (GAT), які використовують механізм уваги для визначення ваги зв'язків, та графові рекурентні мережі (GRN або RecGNN), які передають повідомлення від одного вузла до іншого через ітеративний процес, а також інші архітектури, зокрема графові автокодери (GAE) та просторово-часові графові нейронні мережі (STGNN). Кожен із цих різновидів GNN має свої унікальні особливості та застосування, що робить їх ефективними для вирішення різних завдань в аналізі графових даних.

1.3.1. Рекурентні графові нейронні мережі

Рекурентні графові нейронні мережі були першими роботами з дослідження графових нейронних мереж. RecGNN застосовуються для вилучення високорівневих представлень вершин графа, використовуючи один і той самий набір параметрів рекурентно (рис 1.2.).

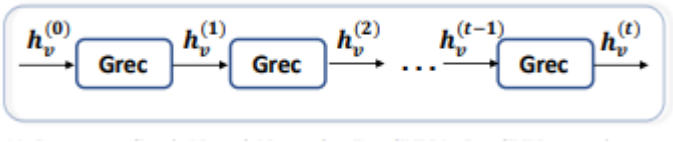

Рис 1.2. Рекурентна графова нейронна мережа

У кожному кроці рекурентного процесу кожен вузол враховує інформацію, отриману від сусідів, і оновлює своє внутрішнє представлення. Це оновлення може відбуватися за допомогою різних архітектур нейронних мереж, таких як LSTM (Long Short-Term Memory) або GRU (Gated Recurrent Unit), що дозволяє враховувати інформацію з попередніх кроків інтерації.

Основна мета RecGNN полягає в досягненні стійкої рівноваги, коли вузли графа перестають змінювати свої представлення на підставі отриманої інформації. Вона означає, що подальші ітерації обміну інформацією не призводять до зміни внутрішніх представлень вузлів, що вказує на збіжність процесу.

Перевагою RecGNN є те, що вони можуть використовувати рекурентні зв'язки для аналізу змін у графічній структурі з часом. Це корисно при моделюванні динамічних графів, де зв'язки еволюціонують зі зміною часу. Також рекурентні графові мережі можуть ефективно працювати з графами різних розмірів, після чого їхні рекурентні механізми можуть адаптуватися до кількості вузлів та ребер у графі.

З іншого боку RecGNN мають недоліки, одним з яких є велика кількість параметрів, що потрібно налаштувати під час процесу навчання. Це може призвести до складнощів у навчанні моделі, особливо при обробці великих

графів, які мають багато вузлів і зв'язків. Ще однією проблемою є висока обчислювальна складність рекурентних механізмів при роботі з великими графами. Розрахунки для кожного вузла ітеративно залежать від інших вузлів у графі, що може призвести до значного часу обчислення, особливо при великих масштабах.

1.3.2. Графові автокодери

Графові автокодери (Graph Autoencoders) - це клас методів у глибокому навчанні, спрямованих на використання нейронних мереж для створення векторних представлень графів чи їх елементів. Ці моделі базуються на концепції автокодерів, які вивчають представлення даних шляхом зменшення розмірності вхідних даних і їх відновлення до початкового вигляду. У контексті графів, графові автокодери вивчають компактне числове представлення графа чи його елементів (наприклад, вузлів або ребер) та його подальшу реконструкцію.

Зазвичай графовий автокодер складається з двох основних частин: енкодеру, який перетворює вхідний граф у меншорозмірний векторний простір, і декодеру, який відновлює вихідний граф з цього простору (рис 1.3.). Процес навчання полягає у мінімізації втрат під час реконструкції графа та одночасному забезпеченні, щоб отримані векторні представлення були інформативними та універсальними для вхідних графів.

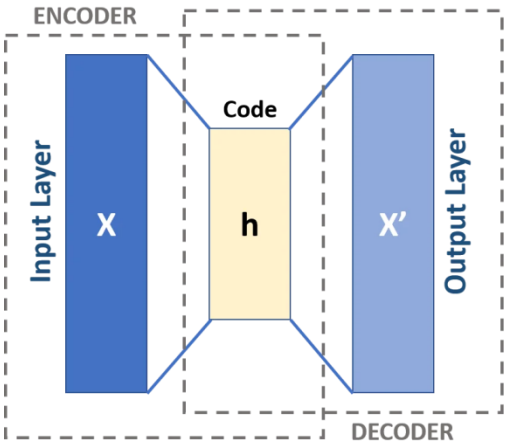

Рис 1.3. Алгоритм автокодера

Ці методи знаходять широке застосування в області визначення вузлів, класифікації та згорткового навчання на графах. Вони допомагають у вирішенні завдань, таких як прогнозування властивостей вузлів, аналіз графових структур та відносин між вузлами, знаходження аномалій у графах та багато інших завдань, пов'язаних з обробкою і аналізом графів.

Перевагою графових автокодерів є їх здатність до зведення графів до низькорозмірних представлень, що може бути корисним для роботи з великими графами та використання отриманих векторних представлень для подальших завдань машинного навчання. Однак, вони можуть мати обмежену здатність реконструювати деякі складні графові структури, особливо у випадку великих або динамічних графів.

1.3.3. Просторово-часові графові нейронні мережі

Просторово-часові графові нейронні мережі або STGNN спрямовані на аналіз просторово-часових графів, що використовуються при виконанні таких задач, як передбачення руху, маневрів водіїв та розпізнавання дій людини (рис 1.4.). Одним з головних аспектів STGNN є поєднання згорткових графових

механізмів для розуміння просторової структури графів з рекурентними механізмами для моделювання часової залежності.

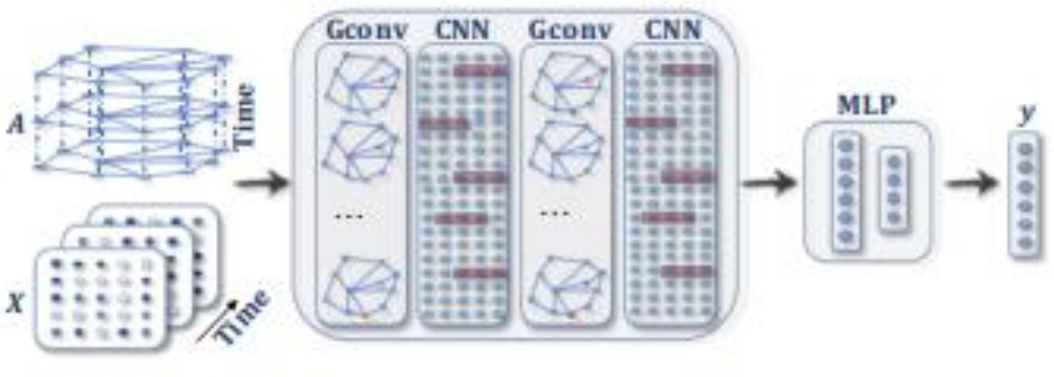

Рис 1.4. Просторово-часова графова нейронна мережа

Ці мережі можуть ефективно моделювати складні структури графів, що мають як просторові, так і часові залежності. Це робить їх універсальними для адаптації до реальних сценаріїв, де важливо враховувати динаміку змін. Основними перевагами таких мереж є здатність передбачати та моделювати поведінку графів у майбутньому, а також можливість адаптуватися до просторових та часових залежностей. Проте складність обробки та необхідність великих обсягів даних можуть бути основними недоліками при їх використанні.

1.3.4. Графові мережі уваги

Графова мережа уваги (Graph Attention Network) - це тип графової нейронної мережі, який використовує механізм уваги для аналізу графових даних (рис 1.5.). Основна ідея полягає у можливості фокусуватись на конкретних елементах графа, які є важливими для певних обчислень [11].

Механізм уваги в графових мережах може варіюватися, але загалом він передбачає визначення ваги для кожного зв'язку між вузлами або ваги самого

вузла на основі його важливості для певного контексту чи завдання[10]. Це дозволяє приділяти більше уваги важливим вузлам під час обчислень, удосконалюючи якість роботи моделі.

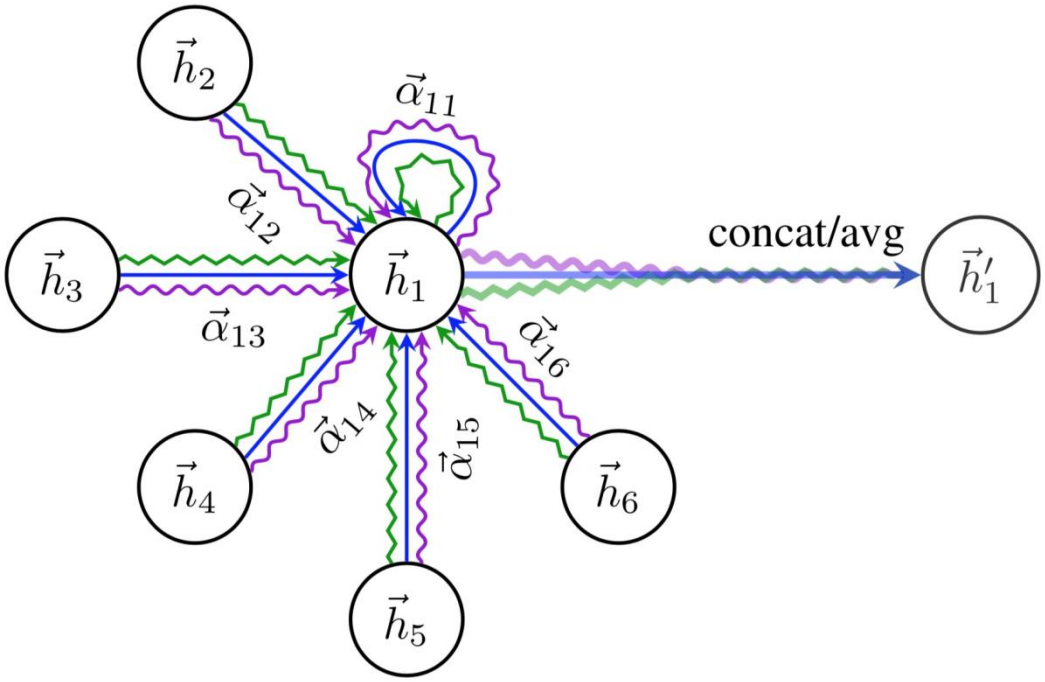

Рис. 1.5. Графова мережа уваги

Графові мережі уваги застосовуються в різних областях. В задачах аналізу текстів або перекладу мови, графові мережі уваги можуть визначати ваги для кожного слова в реченні, враховуючи контекст та важливість кожного слова для здійснення перекладу чи аналізу тексту. У випадку комп'ютерного зору або аналізу зображень, графові мережі уваги можуть використовуватися для сегментації об'єктів на зображенні, надаючи більше уваги деяким областям для точнішого виділення та розпізнавання об'єктів. У біоінформатиці, вони можуть застосовуватися для аналізу молекулярних структур чи білкових взаємодій, де важливо виділити ключові зв'язки для розуміння функцій біологічних об'єктів.

Основною перевагою графових нейронних мереж є наявність механізму гнучкої уваги, що дозволяє враховувати важливість кожного вузла в графі та динамічно адаптуватись до поточного контексту задачі[12]. Але, важливо зазначити, що у деяких випадках механізм уваги може призводити до нестабільності у великих графах. Також обробка та навчання моделі можуть вимагати значних обчислювальних ресурсів, що збільшує час роботи.

1.3.5. Графові згорткові мережі

Графова згортковка мережа (Graph Convolutional Network) - це тип графової нейронної мережі, яка використовує концепцію згорток, аналогічну до тих, що застосовуються у звичайних згорткових нейронних мережах (CNN) для обробки властивостей вузлів графа замість пікселів (рис 1.6).

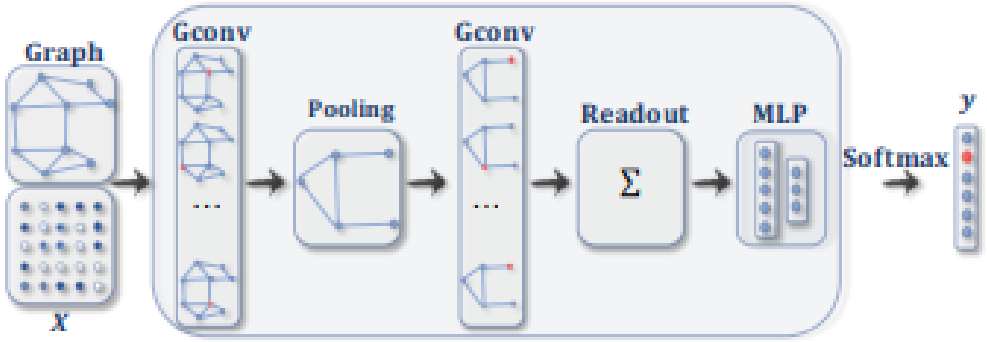

Рис 1.6. Графова згорткова мережа

Згорткові шари дозволяють виконувати операції згортки над сусідніми вузлами графа, використовуючи їхні властивості для отримання високорівневих представлень вершин. Це дозволяє враховувати важливі зв'язки між сусідніми вузлами та ефективно моделювати структуру графа, роблячи GCN центральною складовою для багатьох складних моделей GNN.

Графові згорткові мережі застосовуються у різних сферах для аналізу та обробки графових даних. GCN використовуються в рекомендаційних системах для покращення рекомендацій на основі взаємодії між користувачами та об'єктами. У біологічних дослідженнях вони використовуються для аналізу біохімічних мереж, дослідження взаємодії білків та метаболічних шляхів. Також знаходять застосування в комп'ютерному баченні для розпізнавання образів та аналізу їхніх зв'язків, у кібербезпеці для виявлення аномалій та кібератак, у молекулярному моделюванні для дослідження хімічних сполук та структур, а також на фінансових ринках для аналізу тенденцій та ризиків.

Однією з важливих переваг GCN є їх здатність до ефективного врахування структури графа під час обробки даних, а також їх універсальність[13]. Вони можуть застосовуватися до різноманітних типів графів, будучи гнучкими у використанні для вирішення різноманітних завдань, від класифікації вузлів до передбачення зв'язків між ними. Крім того, GCN можна розширювати для використання у складніших структурах GNN або комбінувати з іншими типами мереж для поліпшення результатів моделі.

Таким чином архітектура згорткових нейронних мереж стала популярною основою для розвитку графових нейронних мереж. Графи є складними структурами з вузлами та зв'язками, і GNN застосовують концепції згорток для аналізу цих структур. Операції згортки виявляють взаємозв'язків між вузлами у графі та виняткових властивостей, подібно тому, як CNN виявляють ознаки у регіонах зображення.

1.4. Методи обробки графів у графових нейроних мережах

Графові нейронні мережі використовують різні методи обробки графових даних, залежно від їх архітектури. У моделях графових нейронних мереж ці методи представлені модулями розповсюдження, вибірки та об'єднання. Модулі розповсюдження відповідають за передачу інформації між

вузлами графа. Модулі вибірки обирають підмножину вузлів або ребер графа для подальшої обробки. Модулі об'єднання, у свою чергу, займаються об'єднанням або агрегацією інформації з різних частин графа для отримання більш високорівневих або узагальнених представлень[36].

1.4.1. Модулі розповсюдження - оператори згортки

Найпоширенішими операторами розповсюдження, що використовуються у GNN, є оператори згортки. Основна ідея операторів згортки полягає у використанні концепції згорток з згорткових нейронних мереж. У графах, оператори згортки враховують сусідні вузли та їхні властивості для оновлення властивостей конкретного вузла, здійснюючи обмін інформацією між сусідами. Існують два основних підходи до застосування операторів згортки: спектральний та просторовий.

Спектральні підходи в графових нейронних мережах базуються на теорії спектральних властивостей графів. Основна ідея полягає в тому, щоб перетворити графові дані у спектральний простір та використовувати методи, подібні до тих, що використовуються у цифровій обробці сигналів[14].

У спектральних підходах графовий сигнал або властивість кожного вузла графа перетворюється в спектральну область через графове перетворення Фур'є. Після цього, замість звичайних операцій згортки, використовуються операції у спектральному просторі для обробки цього сигналу. Ці операції використовують фільтри частот для виявлення та аналізу структур на графі. Згортки в спектральних підходах обчислюються через перемноження вхідного сигналу на фільтр у спектральному просторі, отримуючи таким чином вихідний сигнал. Для створення фільтрів у спектральному просторі використовуються такі методи як спектральна мережа або ChebNet, які використовують різні підходи до згорток, використовуючи усічений розклад Чебишова або операції нормалізації для поліпшення результатів обробки графів.

Просторовий підхід, у свою чергу, полягає у прямому застосуванні згорток безпосередньо на графі без його перетворення, спираючись на його топологію. У просторовому підході згортка застосовується до кожного вузла графа, фільтруючи його властивості, використовуючи інформацію про його безпосередніх сусідів. Основною проблемою даного підходу при виборі операції згортки є те, що для кожного вузла потрібно враховувати різноманітність кількості та характеру його сусідів[15]. Фіксований розмір околиці для кожного вузла може призвести до неповної або надмірної інформації у результаті, оскільки різні вузли можуть мати різну кількість сусідів. Збереження локальної інваріантності - це ще один важливий аспект у просторових підходах. Результати згортки залежать від інформації, що отримується від сусідів, тому вони можуть бути чутливими до локальних змін у структурі графа. Це означає, що навіть незначні зміни в околиці конкретного вузла можуть вплинути на його оновлені ознаки, що може бути проблемою у виконанні точних та стабільних прогнозів у мережі.

1.4.2. Модулі розповсюдження - рекурентні оператори

Рекурентні методи графового аналізу зайняли важливе місце в початкових дослідженнях у галузі графових нейронних мереж. Основна ідея рекурентних операторів схожа з ідеєю операторів згортки та полягає в тому, щоб кожен вузол мережі оновлював свій внутрішній стан, використовуючи інформацію з його сусідів, у процесі ітеративних кроків. Одна з основних відмінностей між рекурентними операторами та згортковими полягає в призначенні ваг для шарів. У згорткових операторах кожен шар має власні ваги, тоді як у рекурентних операторах використовується однакова вага для всіх шарів. Рекурентні оператори у графових нейронних мережах використовують два основних підходи: методи на основі збіжності та методи на основі воріт.

Методи на основі збіжності орієнтовані на стабільність та збіжність прихованих станів в мережі. В таких методах важливо контролювати розмір та вплив кожного оновлення, аби уникнути нестабільності. Оновлення вузлів проводяться за допомогою функцій, які гарантують, що кожен новий стан вузла базується на попередньому стані та приходить до стабільної точки. Це може бути досягнуто шляхом обмежень на ваги, додаванням функцій активації для контролю інформаційного потоку та іншими методами, які гарантують стійкість та збіжність обчислень.

Методи на основі воріт використовують підходи, подібні до рекурентних нейронних мереж (RNN)[16], де кожний вузол може активувати або деактивувати частини інформації, яка проходить через нього. Ці методи використовують ворота для контролю потоку інформації через вузли, дозволяючи змінювати та регулювати вплив інформації з різних джерел на кожному кроці ітерації. Graph LSTM - це приклад такого методу, який використовує механізм воріт для роботи з графовими даними.

1.4.3. Модулі вибірки

Механізм вибірки у графових нейронних мережах використовується для подолання проблеми “вибуху сусідів”, коли великий обсяг інформації від сусідніх вузлів перевантажує обчислення. При агрегації повідомлень для кожного вузла графа від його сусідів, механізм вибірки обирає лише підмножину сусідніх вузлів для аналізу. Це дозволяє зменшити обсяг інформації, який обробляється, та уникнути вибуху обчислень при подальшому розповсюдженні інформації через граф. Загалом можна виділити наступні механізми вибірки у графових нейронних мережах: вибірка вузлів, вибірка шарів та вибірка підграфів.

Механізм вибірки вузлів фокусується на відборі обмеженої кількості сусідніх вузлів для кожного вузла у графі. Це дозволяє зберігати компактну

околицю визначення навколо кожної вершини, скорочуючи обсяг даних для подальшої обробки.

Пошарова вибірка відрізняється від вибірки сусідів для кожного вузла, оскільки зберігає невеликий набір вузлів для агрегації в кожному шарі графової нейронної мережі[17]. Цей підхід дозволяє контролювати кількість інформації, яка переноситься крізь кожен рівень, забезпечуючи зменшення обсягу обчислень та регулюючі розмір інформаційного потоку в процесі обробки даних у графі.

Механізм вибірки підграфів відрізняється від традиційної вибірки вершин і ребер, оскільки базується на обмеженні пошуку околиць в межах певних підграфів замість аналізу всього графа. Замість того, щоб враховувати повну топологію графа, цей підхід використовує обмежений набір підграфів для обмеження аналізу, що дозволяє зосередитись на конкретних частинках графа, що сппрощує та оптимізовує обробку великих мереж.

1.4.4. Модулі об’єднання

При побудові згорткових нейронних мереж у галузі комп’ютерного зору за шаром згортки зазвичай слідує шар об'єднання для отримання більш загальних характеристик. У контексті графових даних модулі об'єднання спрямовані на агрегацію властивостей та інформації з різних частин графа для отримання більш загального уявлення. Вони відіграють ключову роль у формуванні ієрархічної структури графів та вирішення задач класифікації та прогнозування. Процес агрегації може включати об'єднання інформації з різних вузлів або групування характеристик від різних частин графа. Це дозволяє виявити більш високорівневі або загальні ознаки, що можуть бути використані для подальшого аналізу та прийняття рішень. В графових нейронних мережах виділяють два основних підходи до об'єднання: пряме та ієрархічне об'єднання.

Прямі методи об'єднання зазвичай здійснюють агрегацію інформації на одному рівні графа, об'єднуючи характеристики безпосередньо з сусідніх вузлів для отримання загальних особливостей. Ці методи безпосередньо вивчають представлення графів з вузлів і не досліджують ієрархічні властивості структури графа[18].

У свою чергу, ієрархічні методи об'єднання оперують на різних рівнях графової структури, формуючи більш складні та абстрактні характеристики, шляхом об'єднання інформації з різних шарів або рівнів. Ці методи спроможні виявляти та аналізувати комплексні зв'язки між елементами графа на різних рівнях ієрархії. Шляхом поєднання даних з різних рівнів вони створюють більш абстрактні ознаки, які відображають загальні характеристики графової структури.

Таким чином обробка даних у графових нейронних мережах спирається на методи згортки графів, які використовуються для виділення ключових характеристик у даних, необхідних для подальшої обробки та аналізу. При побудові моделей графових нейронних мереж також використовуються методи для оптимізації процесу згортки з метою покращення точності та ефективності моделей.

1.5. Постановка задачі

Метою роботи є дослідження та обгрунтування архітектури нейронної мережі для розв’язку задачі перевірки ізоморфізму графів. Для досягнення поставленої мети необхідно:

- Дослідити архітектури графових нейронних мереж, що здатні розв’язувати подібну задачу;
- Обгрунтувати вибір тієї чи іншої архітектури;
- Побудувати та навчити нейронну мережу обраної архітектури та дослідити її властивості.

1.6. Висновок до розділу

У цьому розділі були проаналізовані основні архітектури та методи графових нейронних мереж. Аналіз показав широке використання архітектури графових згорткових мереж, які були натхненні архітектурою згорткових нейронних мереж з галузі комп’ютерного зору. Основним методом обробки графових даних у цій архітектурі є оператори згортки. За допомогою цієї архітектури розв’язується широкий клас задач, однак існують спеціалізовані архітектури, які також дозволяють розв’язувати окремі задачі. Через це виникає питання щодо доцільності використання згорткових мереж або спеціалізованих мереж задля розв’язку задач на графах.

РОЗДІЛ 2. ВИВЧЕННЯ ОСОБЛИВОСТЕЙ ПРОЦЕСІВ ВИКОРИСТАННЯ ГРАФІВ ЗНАНЬ

2.1. Проблема ізоморфізму графів та її розв’язок за допомогою нейронних мереж

Ізоморфізмом у контексті роботи з графовими структурами називають концепцію, що визначає, коли два графи мають однакову структуру, тобто якщо існує схожість відносно взаємозв'язків між вершинами та ребрами двох графів, за яким ми можемо зробити висновок, що ці графи фактично однакові. Іншими словами, якщо можливо знайти взаємно однозначне відображення між вершинами двох графів таким чином, що зберігається структура графа, то ці графи вважаються ізоморфними[19] (рис.2.1). Математично ізоморфізм визначається наступним чином: два графи H і G є ізоморфними тоді і тільки тоді, коли для будь-якої пари вузлів u і v з H, які є суміжними, існує перетворення f, де f(u) є суміжним до f(v) у G.

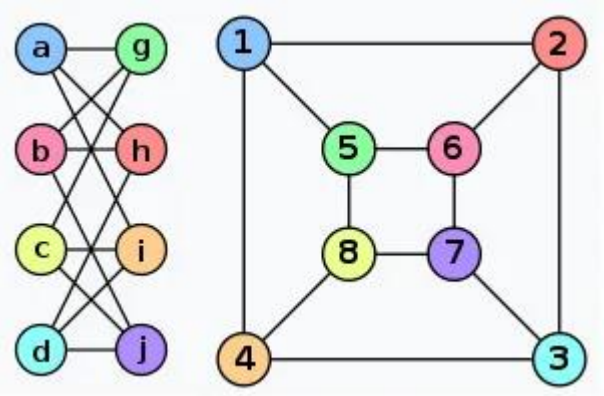

Рис. 2.1. Ізоморфні графи

Це важливий інструмент у теорії графів та застосовується у різних областях, включаючи комп'ютерні науки, хімію, біологію та соціальні науки. Основні цілі використання ізоморфізму включають:

- Виявлення структурної схожості: Через виявлення ізоморфізму можна порівнювати графи та встановлювати, наскільки вони схожі структурно. Це може бути корисним для пошуку схожих патернів або взаємозв'язків у різних графах.
- Аналіз мереж: Для великих мереж, таких як соціальні мережі, біологічні мережі або графи знань, важливо встановлювати схожість структури. Це може допомогти у виявленні груп, підграфів або важливих взаємозв'язків.
- Моделювання та прогнозування: У графових нейронних мережах аналіз ізоморфізму може використовуватись для розв'язання завдань класифікації, прогнозування або генерації графів, оскільки встановлення структурної схожості може бути ключем для прийняття рішень
- Перевірка унікальності: Визначення, чи є граф унікальним у контексті певного застосування, що є важливим для запобігання дублювання даних або виявлення унікальних властивостей графів.
- Кодування та зберігання: У деяких областях, де графи представляють складні системи або дані, ізоморфізм може використовуватись для кодування та зберігання графів, щоб зменшити їхню складність та ефективно використовувати пам'ять.

Проблема обчислення ізоморфізму графів є важливою та важкою, оскільки для неї досі не існує ефективного поліноміального алгоритму, який би працював для всіх випадків. Найкращі відомі алгоритми для розв'язання цієї проблеми мають експоненційну складність у гірших випадках, що означає, що час їх виконання зростає швидко з розміром графа[35]. Це робить проблему ізоморфізму графів важкою для розв'язання, особливо для великих графів, і робить важкою розробку алгоритму, що працював би швидко для будь-якого графа.

Найбільш ефективним алгоритмом обчислення ізоморфізму графів є WL-Test (Weisfeiler-Lehman Isomorphism Test)[1]. Цей тест базується на ідеї взаємного порівняння елементів графів та їх оточень, щоб визначити, чи є два

графи ізоморфними. Основний принцип WL-Test полягає у використанні міток, які призначаються вершинам графа. Процес WL-Test включає кілька ітерацій, кожна з яких має наступні кроки:

- Ініціалізація міток: Кожна вершина у графі отримує початкову мітку.
- Порівняння міток: Для кожної вершини визначається набір міток її сусідів та їх позиції. Ці набори порівнюються для кожної вершини, і якщо вони однакові, мітки вершини оновлюються.
- Оновлення міток: Після порівняння ідентичні набори міток вершин замінюються однією новою міткою. Цей процес повторюється для всіх вершин.
- Ітерації: Кроки 2 та 3 повторюються протягом кількох ітерацій. Кожна наступна ітерація використовує попередні результати порівняння міток.

Якщо після закінчення ітерацій у двох графів мітки вершин однакові, то ці графи вважаються ізоморфними. Як зазначалося вище, WL-тест ефективний для невеликих або помірно великих графів. Для великих графів він може стати обчислювально важким через зростання обчислювальної складності з кожною ітерацією[20].

Останні дослідження з графових нейронних мереж піднімають проблему використання GNN для вирішення задачі обчислення ізоморфності графів, адже основна стратегія алгоритмів вирішення проблеми включає у себе агрегацію інформації про сусідів вузлів графа, що також є концепцією роботи графових нейронних мереж. Одним із ключових напрямів розвитку, спрямованих на вирішення цих викликів, стала архітектура GIN (Graph Isomorphism Network).

Головна ідея полягає у тому, що GIN використовує мережу графових згорток, яка може виявляти ізоморфність графів, розпізнаючи їхні структурні властивості. Зокрема, GIN використовує функцію активації, яка перетворює інформацію з сусідніх вершин у новий вектор для кожної вершини. Ця функція є симетричною, що означає, що вона не залежить від порядку подання сусідніх вузлів. Основні кроки роботи мережі наступні:

- Ініціалізація функції вузла: як і в інших GNN, кожен вузол на графі ініціалізується вектором функції.
- Агрегація ознак: кожен вузол об’єднує вектори ознак своїх сусідніх вузлів. На відміну від інших GNN, GIN включають власні функції вузла в агрегацію.
- Трансформація ознак: агрегований вектор ознак перетворюється, як правило, з використанням лінійної трансформації з наступною нелінійною функцією активації. Перетворення містить параметр, який можна вивчати та який дозволяє моделі контролювати важливість власних функцій вузла порівняно з функціями його сусідів.
- Кроки 2 і 3 повторюються для певної кількості шарів. З кожним шаром вузли об’єднують і перетворюють об’єкти з дедалі більшої околиці.
- Зчитування: після останнього шару функція зчитування використовується для агрегування векторів ознак усіх вузлів на графі для отримання результату на рівні графа.

На підставі отриманих остаточних векторів модель виконує подальші порівняння, наприклад, за допомогою класифікатора, щоб визначити, чи є графи ізоморфними, чи відмінними.

Ще одним підходом до розв'язання задачі ізоморфізму є алгоритм графового контрастивного навчання з аугментаціями для класифікації графів. Цей метод полягає в порівнянні властивостей графів або їх елементів, використовуючи контрастні механізми.

Його сутність полягає в тому, що модель навчається відрізняти подібні графи від відмінних. Для цього вона працює з парою графів: один зразок вважається "позитивним" (граф, що розглядається), а інший - "негативним" (змінений або штучно створений граф). Модель намагається розрізнити, які елементи є спільними, а які - унікальні для кожного графа.

Це дозволяє навчальній моделі виявляти унікальні особливості графів і визначати, що є ключовими для вирішення задачі ізоморфізму. Аугментації[22] в цьому контексті можуть включати зміни в структурі графів,

додавання/видалення вузлів чи ребер, або будь-які трансформації, які допомагають моделі краще усвідомлювати різницю між графами.

Враховуючи наявність популярного і доволі ефективного WL-Test постає питання наскільки ефективне використання нейронних мереж для розв'язання проблеми ізоморфізму у великих графах. Для цього необхідно детальніше ознайомитись з WL-Test та архітектурою графових нейронних мереж GIN, а також з алгоритмом графового контрастивного навчання.

2.2. WL-Test алгоритм

Алгоритм Вайсфейлера-Лемана[2] — це проста комбінаторна техніка, яка в основному використовується для класифікації графів та інших реляційних структур. Його оригінальна версія була представлена в 1968 році з метою вивчення симетрій у високорегулярних графах. У контексті ізоморфізму графів алгоритм зазвичай застосовується до двох вхідних графів паралельно, щоб вирішити, чи є вони ізоморфними та часто називається процедурою уточнення кольорів графа (color refinement)[34].

Одновимірний алгоритм Вайсфейлера-Лемана, який також називають уточненням кольорів або наївною класифікацією вершин обчислює так зване стійке розфарбування множини вершин свого вхідного графа (рис). Основні кроки алгоритму наступні:

- Нехай G=(V,E) - вхідний граф, де V - множина вершин, E - множина ребер.
- На початку кожній вершині v ∈ V присвоюється початковий колір.
- На кожній ітерації для кожної вершини v, його сусідні вершини перераховуються, і їх мультинабір кольорів використовується для уточнення мультинабору кольорів вершини v (рис 2.).
- Процес уточнення повторюється, доки жодне перепризначення кольорів не буде вносити змін у класифікацію вершин, тобто $c_v^i = c_v^{i-1}$

Для визначення ізоморфізму двох графів отримані результати алгоритму порівнюються між собою. Якщо виявлені відмінності між розфарбуваннями, то ці графи вважаються неізоморфними.

Input: $G = (V, E, X_V)$

1. $c_v^0 \leftarrow \text{hash}(X_v)$ for all $v \in V$
2. **repeat**
3. $\quad c_v^\ell \leftarrow \text{hash}(c_v^{\ell-1}, \{\!\{c_w^{\ell-1} : w \in \mathcal{N}_G(v)\}\!\})\ \forall v \in V$
4. **until** $(c_v^\ell)_{v \in V} = (c_v^{\ell-1})_{v \in V}$
5. **return** $\{\!\{c_v^\ell : v \in V\}\!\}$

Рис. 2.2. Алгоритм уточнення кольору

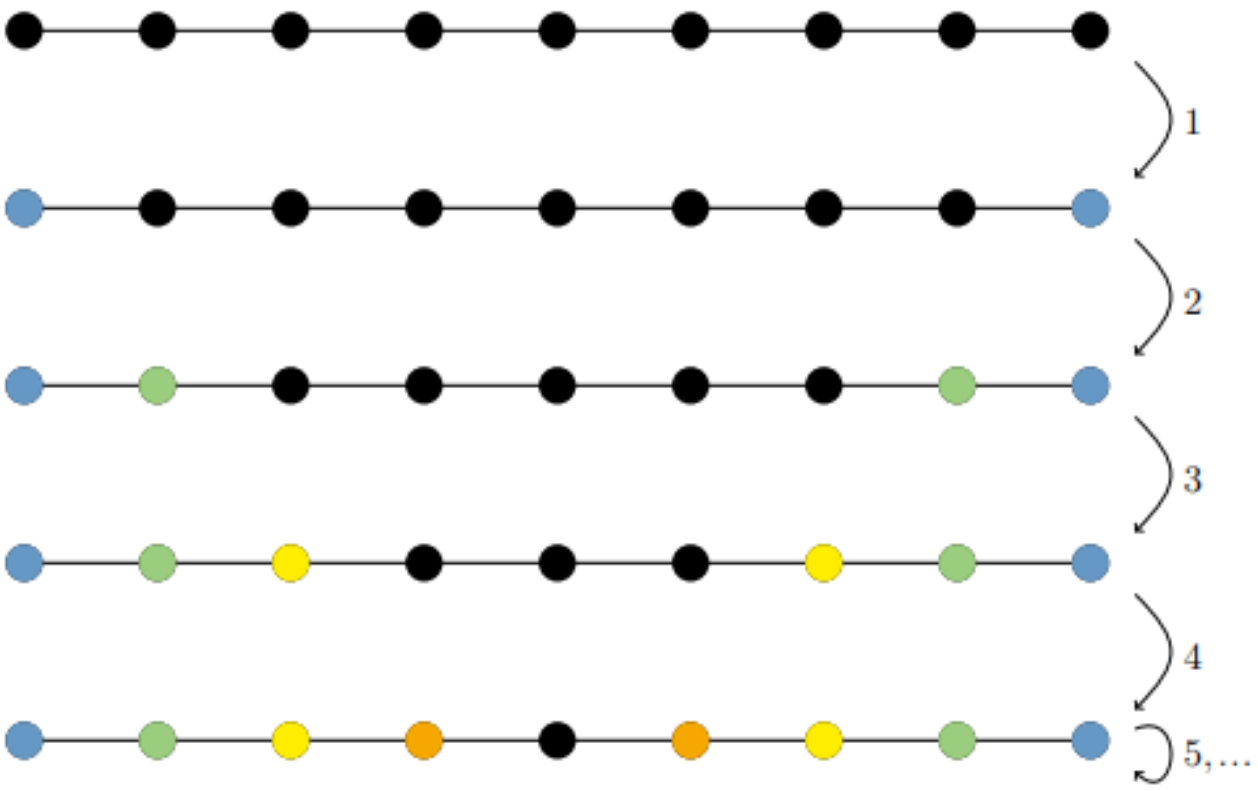

Рис. 2.3. Приклад роботи алгоритму уточнення кольорів

Оцінка часу виконання алгоритму уточнення кольорів залежить від розміру вхідного графа та його структури. Для графа з n вершинами та m ребрами, часова складність може бути у квадратичній або експоненціальній залежності від n та m. Зазвичай оцінка часу виконання алгоритму може бути описана як O((m+n) log n). У найкращому випадку, коли графи мають невеликі розміри та структуру, час виконання може бути близьким до лінійного. Однак для великих графів з великою кількістю вершин та складною структурою, час виконання може зростати експоненціально, що робить алгоритм непрактичним у деяких випадках.

2.3.GIN

Графова Мережа Ізоморфізму (GIN) - це архітектура графових нейронних мереж, яка була розроблена для розв'язання проблеми ізоморфізму графів. Одна з ключових особливостей GIN полягає в узагальненні WL-тесту. GIN використовує концепцію WL і робить його більш потужним, застосовуючи його для аналізу великих та складних наборів даних[23]. Таким чином, алгоритм може виявляти складніші патерни та зв'язки, ніж інші алгоритми.

Основним принципом роботи GIN є моделювання ін'єктивних мультимножинних функцій для агрегації інформації від сусідів вузла. Це досягається за допомогою теорії "глибоких мультимножин", що дозволяє параметризувати універсальні мультимножинні функції за допомогою нейронних мереж. Однією з ключових властивостей GIN є можливість використання суматорів для представлення універсальних функцій над мультимножинами[33]. Можна виділити наступні кроки роботи даної мережі:

- Ініціалізація функції вузла: як і в інших GNN, кожен вузол на графі ініціалізується вектором функції.
- Агрегація ознак: кожен вузол об’єднує вектори ознак своїх сусідніх вузлів. На відміну від інших GNN, GIN включають власні функції вузла в агрегацію.

- Трансформація ознак: агрегований вектор ознак потім перетворюється, як правило, з використанням лінійної трансформації з наступною нелінійною функцією активації. Перетворення містить параметр, який можна вивчати, який дозволяє моделі контролювати важливість власних функцій вузла порівняно з функціями його сусідів.
- Кроки 2 і 3 повторюються для певної кількості шарів. З кожним шаром вузли об’єднують і перетворюють об’єкти з дедалі більшої околиці.
- Зчитування: після останнього шару функція зчитування використовується для агрегування векторів ознак усіх вузлів на графіку для отримання результату на рівні графіка (рис. 2.4).

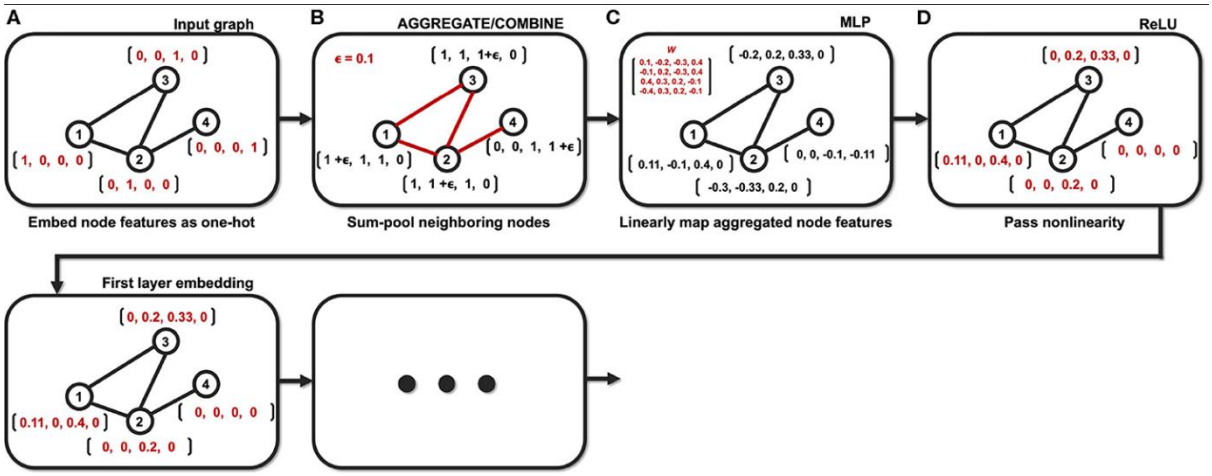

Рис. 2.4. Приклад роботи GIN для графа з 4 вершинами

Однією з ключових складових цієї архітектури є вбудовування вузлів графа. Вбудовування вузлів у графових ізоморфних мережах - це процес перетворення кожного вузла графа у векторний або числовий простір таким чином, щоб зберегти його структурні та контекстуальні властивості[24]. Основна мета полягає в тому, щоб кожен вузол отримав векторне представлення, яке враховує його роль та взаємозв'язки з іншими вузлами у графі.

Процес вбудовування вузлів у графових мережах включає наступні етапи:

- Підготовка вхідних даних: Граф подається на вхід, де кожен вузол може мати різні характеристики, такі як властивості вузла, зв'язки з іншими вузлами, мітки тощо. Ці дані можна представити у вигляді матриці або тензора.
- Створення вбудування вузлів: Нейронна мережа, зазвичай заснована на конволюційних, рекурентних або трансформерних архітектурах, отримує на вхід інформацію про граф та генерує вбудування (embedding) для кожного вузла. Це вбудування може бути вектором фіксованої довжини, що представляє вузол у числовому просторі.
- Оновлення вбудувань: У графових ізоморфних мережах оновлення вбудувань вузлів зазвичай відбувається через кілька етапів або шарів. Оновлення вбудувань полягає в тому, щоб кожен вузол мережі отримав оновлене числове або векторне представлення, яке враховує не лише його власні характеристики, а й інформацію про його сусідів та контекст.
- Представлення графу: В результаті процесу вбудовування отримуємо векторні представлення всіх вузлів графа (рис).

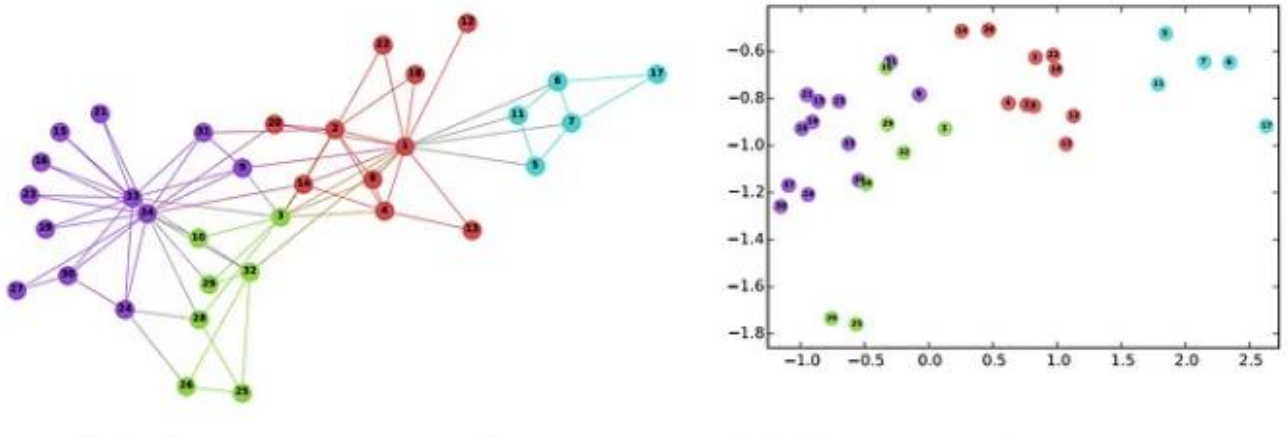

Рис. 2.5. Приклад алгоритму вбудовування вузлів

Додатково, GIN здатна оновлювати вбудовування вузлів за допомогою мультишарових перцептронів, які дозволяють моделювати складні функції для агрегації інформації від сусідніх вузлів та самого вузла. Ця архітектура може представляти універсальні функції над вузлом та мультимножиною його сусідів, що дозволяє задовольняти умовам ін'єктивності та максимальної дискримінативної потужності серед інших графових нейронних мереж[32].

Вбудовані вузли, отримані GIN, можна безпосередньо використовувати для таких завдань, як класифікація вузлів і прогнозування зв’язків. Для завдань класифікації графів у архітектурі використовується функція «зчитування», яка, враховуючи вбудовування окремих вузлів, виробляє вбудовування всього графу.

Важливим аспектом зчитування на рівні графа є те, що представлення вузлів, які відповідають структурам піддерева, стають більш уточненими та глобальними зі збільшенням кількості ітерацій[25]. Достатня кількість ітерацій є ключем до досягнення хорошої дискримінаційної здатності.

GIN має переваги порівняно з класичним WL-тестом. Щодо часової оцінки, GIN може працювати відносно ефективно для великих графів завдяки здатності використовувати механізми швидкого та ефективного агрегування інформації про сусідні вузли.

2.4. Графове контрастивне навчання (graphCL)

Графове контрастивне навчання - це метод класифікації, який використовується для роботи з графовими даними. Його основна ідея полягає в тому, щоб навчити модель розрізняти корисні зв'язки у графах від некорисних шляхом порівняння подібних та відмінних елементів у графах за допомогою контрастивних механізмів. Даний метод був інспірований використанням контрастивних механізмів у обробці зображень, основна ідея

яких полягає у тому, щоб уявлення узгоджувалися одне з одним за належних перетворень[31].

Аугментація даних спрямована на створення нових і реалістично раціональних даних шляхом застосування певних перетворень без впливу на семантику[26]. Наприклад, при класифікації зображень програми обертання та обрізання враховують умову, що люди отримують однакові семантичні знання з повернутого зображення або його локальних фрагментів. Попри універсальність методів аугментації для зображень, графові дані не мають однорідної структури, тому для різних категорій графів існують різні аугментації, що можуть бути більш ефективними у різних випадках. Загально можна виділити чотири категорії аугментації графових даних:

- Вилучення вузла. При вилученні вузла графа випадково відкидається певна частина вершин разом із їхніми зв’язками. Операція передбачає, що відсутня частина вершин не вплине на семантичне значення графа. Ймовірність випадкового вилучення кожного вузла відповідає типовому незалежному та рівномірному розподілу (або будь-якому іншому розподілу).
- Пертурбація границь. Ця операція передбачає зміну зв’язків вершин графу шляхом додавання або вилучення певного співвідношення ребер. Це означає, що семантичне значення графу має певну стійкість до варіацій шаблону зв’язності країв.
- Маскування атрибутів. Маскування атрибутів спонукає моделі відновлювати замасковані атрибути вершин, використовуючи їх контекстну інформацію, тобто атрибути, що залишилися. Основне припущення полягає в тому, що відсутність часткових атрибутів вершин не сильно впливає на прогнози моделі.
- Будування підграфу. Підграф будується за допомогою випадкового блукання. Це передбачає, що семантика графу може бути значною мірою збережена в його локальній (частковій) структурі.

Модифікований граф надалі використовується в алгоритмі графового контрастного навчання. У контрастному навчанні графів попереднє навчання виконується шляхом максимізації узгодженості між двома аугментованими видами одного графа через контрастні втрати в латентному просторі (рис. 2.6). Структура алгоритму складається з наступних чотирьох основних компонентів:

- Аугментація даних графа. Дані графа піддаються аугментації, щоб отримати два корельовані погляди Gˆi, Gˆj.
- Кодер на основі GNN. Кодер на основі GNN виділяє вектори представлення на рівні графа hi, hj для аугментованих графів Gˆi, Gˆj. Алгоритм навчання не накладає жодних обмежень на архітектуру GNN, на основі якої будується кодер.
- Проекційна головка. Нелінійне перетворення під назвою “проекційна головка” відображає доповнені представлення в інший латентний простір, де обчислюється контрастна втрата zi, zj. У графовому контрастному навчанні для отримання zi, zj використовується двошаровий персептрон (MLP).
- Функція контрастних втрат. Функція контрастних втрат визначається для забезпечення максимальної узгодженості між позитивними парами zi, zj порівняно з негативними парами. Прикладом такої функції може бути функція нормалізованої перехресної втрати ентропії з температурним масштабом (NT-Xent).

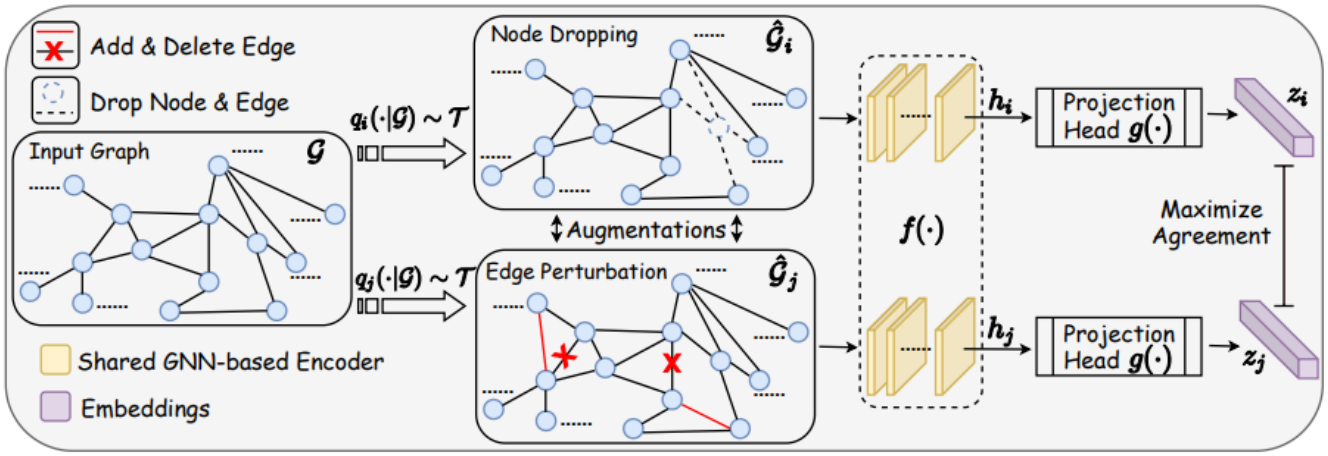

Рис 2.6. Алгоритм графового контрастного навчання

Під час попереднього навчання GNN випадково обирається вибірка з N графів, які обробляються за допомогою контрастного навчання, у результаті чого отримують 2N аугментованих графів і відповідні контрастні втрати для оптимізації. Негативні пари не відбираються явно, а генеруються з інших N − 1 аугментованих графів у межах тієї ж вибірки. Після цього обчислюється загальна функція втрат для всіх позитивних пар з вибірки, яка використовується для зміни моделі[27].

Особливістю даного алгоритму є те, що аугментація даних графа має вирішальне значення. Без будь-якої аугментації графа порівняльне навчання не є корисним і часто гірше порівняно з навчанням на оригіналі. Без аугментації graphCL просто порівнює дві оригінальні вибірки як негативну пару (при цьому втрата позитивної пари стає нульовою), що призводить до рівномірного відштовхування всіх представлень графів одне від одного, що є неінтуїтивним для класифікації. Важливо, що коли застосовуються відповідні аугментації, впроваджуються відповідні пріоритети розподілу даних, змушуючи модель вивчати уявлення, інваріантні до бажаних змін, завдяки максимальному узгодженню між графіком і його доповненням.

Наступною особливістю є використання комбінованих типів аугментацій при навчанні. Застосування пар одного типу зазвичай не

призводить до найкращої продуктивності (за винятком вилучення вузла), порівняно з парами аугментацій різних типів. Подібні спостереження були зроблені при навчанні візуальним представленням, де створення різних аугментацій дозволяє вивчення особливостей, які перетворюються на низькорівневі "скорочення", зробивши особливості більш універсальними. Використання пар графів з різними типами аугментацій у графовому контрастному навчанні зумовлює уповільнення спадання функції втрат у порівнянні з парами того самого типу, коли процедура оптимізації залишається незмінною. Таким чином створення аугментованих пар різних типів відповідає «складнішому» завданню контрастного прогнозування[28].

Також важливо враховувати, що вибір аугментацій напряму залежить від структури графових даних. Так, наприклад пертурбація границь показує ефективні результати при використанні у соціальних мережах, але погіршує роботу біологічних мереж. У порівнянні з соціальними мережами «семантеми» деяких даних біомолекул більш чутливі до окремих границь. Наприклад зміна молекул NCI1 відповідає видаленню або додаванню ковалентного зв’язків, що може різко змінити ідентичність і навіть дійсність сполуки. Таким чином пертурбація границь для хімічних сполук концептуально несумісна зі знаннями предметної області та емпірично некорисна для продуктивності навчання.

Загалом, дослідження довели, що використання маскування атрибутів графів сприяє покращенню продуктивності, особливо у щільних графах. За результатами досліджень на різних наборах даних соціальних мереж, таких як COLLAB та RDT-B, підтверджено, що маскування атрибутів графів може забезпечити покращення від 0,17% до 5,12% залежно від щільності графів. Також виявлено, що застосування меншого маскування до розрідженого графа не сприяє покращенню, в той час, як більше маскування у щільних графах може мати позитивний ефект (рис 2.7).

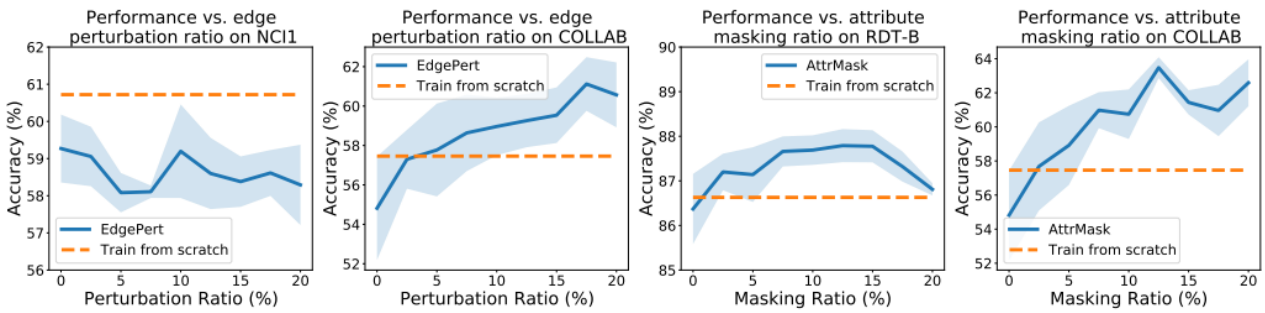

Рис. 2.7. Показники продуктивності відносно різного типу аугментацій

З іншого боку, спостереження щодо аугментацій вилучення вузлів та підграфів виявили їх важливість при обробці даних графа. При відкиданні вузлів підкреслюється пріоритет, що відсутність певних вершин (наприклад, деяких атомів водню в хімічних сполуках або крайніх користувачів для соціальних мереж) не змінює семантичну інформацію, що впливає на наше пізнання досліджуваних даних. Попередні дослідження показали узгодженість локальної і глобальної інформації при вилученні підграфів, а також ефективність даного типу аугментації при репрезентативному навчанні. У контексті контрастивного навчання дослідження показали, що вилучення підграфів призводить до підвищення продуктивності у щільних графах соціальних мереж, таких як COLLAB[29]. З іншого боку, для графів з меншою щільністю, таких як PROTEINS, вилучення підграфів не обов’язково впливає на ппродуктивність (рис. 2.8).

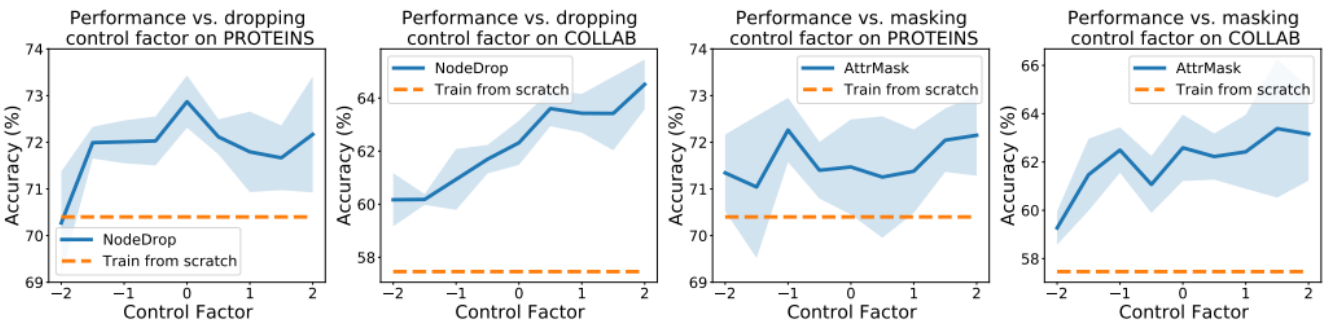

Рис. 2.8. Порівняння продуктивності аугментації вилучення вузлів з маскуванням атрибутів

Таким чином графове контрастивне навчання є ефективним методом у сфері аналізу та класифікації графових даних. Однак, важливо зазначити, що успішність цього підходу значно залежить від правильного вибору та розробки аугментаційних стратегій. Використання різноманітних аугментаційних методів, таких як маскування атрибутів, видалення вузлів або підграфів, перетворення ребер та інших, може значно покращити репрезентації графів, роблячи їх більш універсальними та загальними. Зокрема, визначення оптимальних стратегій аугментацій, які ефективно застосовуються до конкретних типів графів, є ключовим аспектом у розвитку цього методу. Додатково, розробка нових підходів до контрастивного навчання, які будуть більш узгодженими з особливостями графових даних та будуть враховувати їхні унікальні властивості, може виявитися продуктивним напрямком для майбутніх досліджень[30].

Відповідно часова оцінка графового контрастного навчання зазвичай залежить від декількох факторів, таких як розмір графів, кількість доповнень та складність використаних алгоритмів. Точні часові рамки можуть суттєво відрізнятися в залежності від конкретної реалізації цього підходу, обсягу та типу даних, а також вибраного методу контрастного навчання.

2.5. Висновок до розділу 2

В цьому розділі було розглянуто проблему ізоморфізму графів. Були описані класичний алгоритм WL-тест, який добре показав себе для встановлення графу ізоморфізму невеликих за розміром графів. Також було розглянуто архітектуру та особливості графової нейронної мережі GIN, яку було створено спеціально для розв’язання задач ізоморфізму графів великого

розміру. Для підвищення якості машинного навчання на графах, ізоморфізм яких треба встановити, було розглянуто алгоритм Graph Contrastive Learning, що дозволяє підвищити точність виявлення ізоморфізму підграфів.

РОЗДІЛ 3. ДОСЛІДЖЕННЯ ТА ПОРІВНЯННЯ ПРОЯВІВ ІЗОМОРФІЗМУ ГРАФІВ ЗНАНЬ В ПРОЦЕСІ НАВЧАННЯ

3.1.Умови випробувань

Як зазначалося у попередньому розділі, проблема ізоморфізму має сенс для графів різного призначення. Це може бути з’ясування схожості молекулярної будови органічних та неорганічних речей, ситуаційний аналіз в системах прийняття рішень, пошук повторюваних частин графів знань тощо. В цій роботі в якості вхідного графу буде використовуватись граф молекулярної структури протеїнів. Ізоморфізм був перевірений двома методами: класичним WL-test та за допомогою графової нейронної мережі за архітектурою GIN.

3.2. Порівняння структури протеїнів

Задачі ізоморфізму графів використовуються у дослідженні протеїнів із кількох причин:

- Аналіз Структурної Схожості: Графи часто використовуються для представлення молекулярних структур, включаючи структури протеїнів. Задачі ізоморфізму дозволяють ідентифікувати схожість між цими структурами, що може вказувати на функціональну або еволюційну пов'язаність між різними протеїнами.
- Прогнозування Функцій Протеїнів: Встановлення схожості між структурами протеїнів може допомогти у прогнозуванні їхніх функцій. Наприклад, якщо структура невідомого протеїну схожа на відому структуру з відомою функцією, можна зробити припущення про функції цього невідомого протеїну.
- Виявлення Гомологічних Відносин: Ізоморфізм графів може допомогти виявити гомологічні відносини між протеїнами, що є ключовими для розуміння еволюційних шляхів та взаємозв'язків між різними організмами.

- Дослідження Біологічних Мереж: Протеїни часто взаємодіють один з одним, утворюючи складні мережі взаємодій. Аналіз ізоморфізму графів може допомогти у визначенні патернів у цих мережах, що може вказувати на біологічно значущі зв'язки.
- Ефективність Пошуку та Аналізу: Використання алгоритмів ізоморфізму графів може значно підвищити ефективність пошуку та аналізу біологічних даних, дозволяючи обробляти великі набори даних, що часто зустрічаються у біоінформатиці.

Ці застосування ізоморфізму графів відкривають нові перспективи для досліджень у біології та медицині, забезпечуючи глибше розуміння біологічних процесів та сприяючи відкриттю нових ліків і терапевтичних стратегій.

3.3. Датасет

В якості джерела вхідних даних був використаний датасет PROTEINS. Для зручності була використана обмежена версія цього датасету, яка включає 1112 записів про різні протеїни. Структура датасету, що описує відповідні графи, представлена у вигляді кількох файлів. Серед них є файли з мітками протеїнів, кожен протеїн це молекула, яка уособлює окремий граф. Таким чином датасет містить файл зі списком графів, зі списком індикаторів графів, зі списком вузлів, що належать графам, зі списком атрибутів, що належать графам та списком ребер, що поєднують вузли у графах. Тобто, фактично, інформацію про протеїни треба було перетворити у тензорний вигляд кількох наборів списків. Підготована для навчання інформація виглядає як список тензорів для графів кожного з протеїнів та список міток, якими ці протеїни позначаються. В лістингу наведено код, що створює датасет для навчання.

Лістинг 3.1

```
import torch

import ast


class Molecule:

    def __init__(self, graph, nodes):

        self.graph = graph.to('cuda')

        self.nodes = nodes.to('cuda')


def load_data():

    data_num = 1112


    graph_labels = open('graph_labels.txt')

    node_labels = open('node_labels.txt')

    node_attributes = open('node_attributes.txt')

    graph_indicators = open('graph_indicators.txt')

    edges = open('graph_edges.txt')


    t0, t1 = 1, 1

    graphs = []

    graph_size_list = []


    for i in range(data_num):

        graph_size = 1

        while True:

            t1 = int(graph_indicators.readline())

            if t0 == t1:

                graph_size += 1

                t0 = t1

            else:

                t0 = t1
```

```
            break
    graphs.append(torch.zeros(graph_size, graph_size))
    graph_size_list.append(graph_size)

s1 = s2 = 0
for i in range(data_num):
    s1 += graph_size_list[i]
    while True:
        edge = literal_eval(edges.readline())
        if edge[0] <= s1 & edge[1] <= s1:
            graphs[i][edge[0] - s2][edge[1] - s2] = 1.0
            graphs[i][edge[1] - s2][edge[0] - s2] = 1.0
        else:
            break
    s2 = s1

node_dim = 4
node_list = []
for i in range(data_num):
    nodes = torch.zeros(graph_size_list[i], node_dim)
    for j in range(graph_size_list[i]):
        node_label = int(node_labels.readline())
        node_attribute = float(node_attributes.readline())
        node_val =  1.0
        if (node_label == 0):
            nodes[j] = torch.tensor([node_val, 0.0, 0.0, node_attribute])
        elif (node_label == 1):
            nodes[j] = torch.tensor([0.0, node_val, 0.0, node_attribute])
        else:
            nodes[j] = torch.tensor([0.0, 0.0, node_val, node_attribute])
    node_list.append(nodes)
```

```
    molecule = []
    labels = []
    for i in range(data_num):
        molecule.append(Molecule(graphs[i], node_list[i]))
        labels.append(torch.tensor([float(graph_labels.readline()) - 1.0]).to('cuda'))

    graph_labels.close()
    node_labels.close()
    node_attributes.close()
    graph_indicators.close()
    edges.close()

    return molecule, labels

m, l = load_data()
```

3.4. WL-test

Як зазначалося у розділі 2.2, WL-test фактично є алгоритмом, заснованим на розфарбуванні графу. Розфарбування графів на приктиці може виконуватись кількома різними способами і найскладніший випадок, коли треба розфарбувати великий неорієнтований незважений граф. Річ у тім, що наявність орієнтованих ребер та ваг дозволяє доволі просто визначити просторову орієнтацію графа і це суттєво спрощує алгоритм. Тому для розфарбування неорієнтованих графів, часто використовують комбінаторні алгоритми такі, наприклад, як еволюційний алгоритм тощо. Для розв’язку задачі WL-test, часто використовують алгоритм заснований на хешуванні вузлів та ребер, найпростішою версію якого наведено у лістингу 3.2.

Таким чином, щоб з’ясувати факт ізоморфізму графів, треба просто порівняти результати розрахунку хешів цих графів.

Лістинг 3.2.

```
class GraphObject:
    def __init__(self, graph, nodes):
        self.graph = graph  # Тензор PyTorch матриці суміжності
        self.nodes = nodes  # Список тензорів PyTorch для кожного вузла


def weisfeiler_lehman_graph_hash(molecule, iterations=3):
    adjacency_matrix = graph_object.graph
    node_features = graph_object.nodes
    num_nodes = adjacency_matrix.size(0)
    # Ініціалізація міток для кожної вершини
    labels = {node: hash((tuple(node_features[node][:3].tolist()), node_features[node][3].item())) for node in range(num_nodes)}
    # WL ітеративний процес
    for _ in range(iterations):
        new_labels = {}
        for node in range(num_nodes):
            neighbors = adjacency_matrix[node].nonzero().view(-1)
            neighbors_labels = [labels[neighbor.item()] for neighbor in neighbors]
            aggregated_label = hash((labels[node], tuple(sorted(neighbors_labels))))
            new_labels[node] = aggregated_label
        labels = new_labels
    # Обчислення кінцевого хешу
    label_counter = Counter(labels.values())
    graph_hash = hash(frozenset(label_counter.items()))
    return graph_hash
```

За допомогою цієї функції були перевірені всі молекули з наявного датасету. Для цього датасету такий метод показав 100% правильний результат,

однак для більшого датасету цей показник може бути гіршим. Погіршення може бути пояснено промахами хеш-функції, які можуть статися у випадку збільшенні кількості вхідних даних.

3.5. Нейронна мережа

Після виконання перевірки ізоморфізму за допомогою стандартного WL-test була побудована графова нейронна мережа архітектури GIN, код якої наведено у лістингу 3.4.

Лістинг 3.4

```
import torch


class GraphIsomorphismNetwork:
  def __init__(self, node_dim, update_loop_size):
    self.node_dim = node_dim
    self.update_loop_size = update_loop_size


  def mlp(self, molecule):
    return molecule.nodes + torch.mm(molecule.graph, molecule.nodes)


  def readout(self, molecule):
    return molecule.nodes.sum(dim=0)


  def predict(self, molecule):
    nodes = molecule.nodes
    for _ in range(self.update_loop_size):
      nodes = self.mlp(nodes)
    return torch.cat([self.readout(nodes) for _ in range(self.update_loop_size)])
```

Датасет, що був отриманий на початковому етапі, був використаний для навчання та перевірки цієї нейромережі. Оскільки датасет містить всього 1112 записів, що є доволі малою кількістю для якісного навчання, 1000 записів з нього було використано в якості навчальної вибірки та 112 в якості тестової. Результати обрахунку функції втрат під час навчання та тестування наведено на рисунку 3.1.

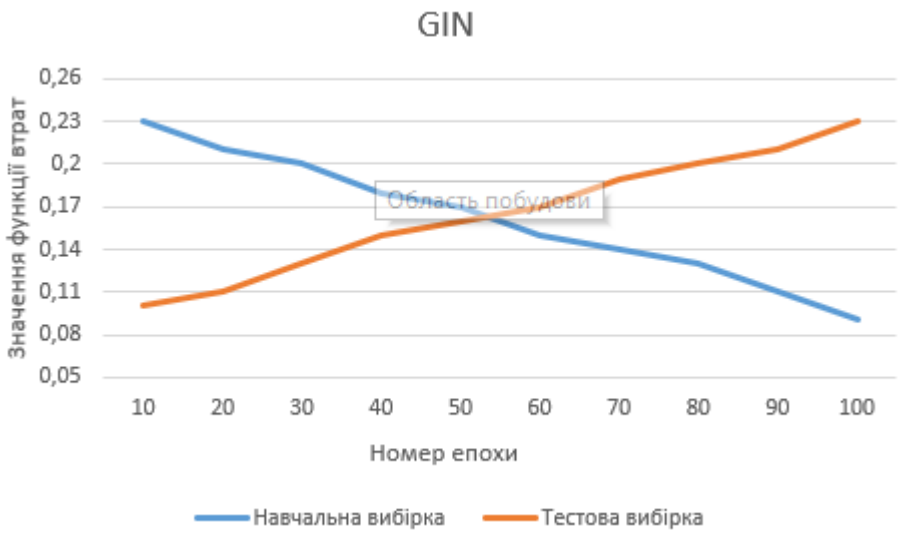

Рисунок 3.1. Функція втрат для GIN

За результатами тестування та перевірки видно, що мережа перенавчилася, тому під час тестування функція втрат росте. Такий ефект часто спостерігається у випадку коли вхідних даних для створення навчальної валідаційної та тестової вибірки було замало.

3.6. Контрастне навчання на графах

За результатами вивчення теоретичного матеріалу, що був викладений у розділі 2.4, було зроблено припущення, що використання контрастного навчання у певних випадках може поліпшити результати аналізу графів та підграфів на предмет ізоморфізму. Для перевірки цього припущення була збудована мережа Graph Contrastive Learning, код якої наведено у лістингу 3.5.

Лістинг 3.5

```
import torch
import torch.nn as nn
import torch.nn.functional as F


class GraphContrastiveLearning(nn.Module):
    def __init__(self, in_features, hidden_features, out_features):
        super(GraphContrastiveLearning, self).__init__()

        self.encoder = nn.Sequential(
            nn.Linear(in_features, hidden_features),
            nn.ReLU(),
            nn.Linear(hidden_features, out_features)
        )

        self.positive_edge_weight = nn.Parameter(torch.ones(1))
        self.negative_edge_weight = nn.Parameter(torch.ones(1))

    def forward(self, graph):
        x = self.encoder(graph.x)

        positive_edges = graph.edges[graph.edges.t() == 1]
        negative_edges = graph.edges[graph.edges.t() == 0]

        positive_scores = torch.mm(x[positive_edges[0]], x[positive_edges[1]])
        negative_scores = torch.mm(x[negative_edges[0]], x[negative_edges[1]])

        loss = self.positive_edge_weight * F.log_softmax(positive_scores) \
               + self.negative_edge_weight * F.log_softmax(-negative_scores)
```

return loss

Мережа була навчана та протестована на тому самому датасеті, що й графова мережа GIN. Результати обрахунку функції втрат наведено а рисунку 3.2.

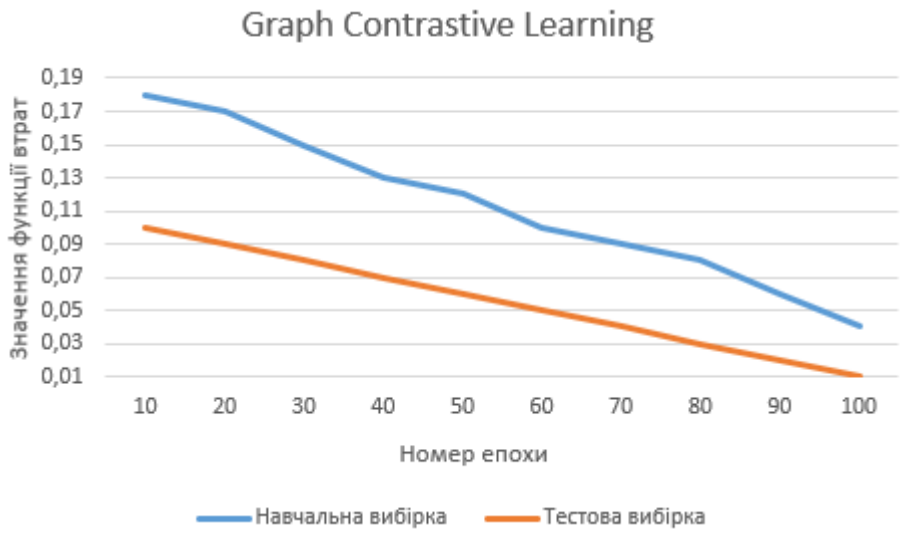

Рисунок 3.2. Функція втрат мережі Graph Contrastive Learning

За результатами навчання та тестування можна зробити висновок, що саме для цього набору даних, мережа контрастного навчання виявилася ефективнішою за GIN, оскільки попри малий обсяг вхідних даних, мережа не перенавчилася.

3.7. Висновок до розділу

Під час виконання практичних випробувань з використанням набору даних PROTEINS, був використаний класичний алгоритм WL-test, а також нейромережеві методи становлення факту ізоморфізму графів GIN та Graph Contrastive Learning застосований до базової GNN. За результатами випробувань нейромережевих методів, GIN показав гірші від інших методів результати тестування, оскільки для цього типу нейронної мережі, вхідних даних для тестування виявилося замало. Contrastive Learning показав порівняно з GIN гарні результати, оскільки ефекту перенавчання не

спостерігалося. Однак, класичний метод WL-test, де використовувалось хешування вмісту графів для цього датасету показав найкращий результат. Це можна пояснити малими розмірами молекул (графи були невеликими за розмірами), та малим розміром датасету, що унеможливило колізії хеш-функції. Однак для графів великого та надвеликого розміру, таке співвідношення результатів може бути змінене в бік переваги нейромережевих методів. Однак це може залежати від характеру вхідних даних.

ВИСНОВОК

В цій атестаційній роботі був виконаний огляд розмаю графових нейронних мереж та була розглянута задача ізоморфізму графів та її розв’язок різними способами. Зокрема, були розглянуті класичний алгоритм WL-test, графова нейронна мережа GIN та графова мережа Graph Contrastive Learning, що реалізує метод контрастного навчання. За допомогою розглянутих методів був проаналізований фрагмент набору даних PROTEINS. В результаті перевірки були з’ясовано, що через відносно невеликий обсяг вхідних даних, класичний алгоритм показав кращі результати, ніж нейромережеві методи, а з двох нейромережевих методів кращим виявився Graph Contrastive Learning, однак ці результати дійсні тільки для цього набору даних. Результати виконання атестаційної роботи можуть бути застосовані під час дослідження біологічних та хімічних сполук, розумової діяльності людини, соціальних та економічних явищ тощо.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. A. Alzaga, R. Iglesias, and R. Pignol. Spectra of symmetric powers of graphs and the weisfeiler-lehman refinements. J. Comb. Theory, Ser. B, 100(6):671–682, 2010.
2. https://davidbieber.com/post/2019-05-10-weisfeiler-lehman-isomorphism-test/
3. K. Audenaert, C. Godsil, G. Royle, and T. Rudolph. Symmetric squares of graphs. J. Comb. Theory Ser. B, 97(1):74–90, 2007.
4. C. J. Colbourn and K. S. Booth. Linear time automorphism algorithms for trees, interval graphs, and planar graphs. SIAM J. Comput., 10(1):203–225, 1981.
5. M. F¨urer. A counterexample in graph isomorphism testing. Technical report, 1987.
6. J. K¨obler, U. Sch¨oning, and J. Tor´an. The Graph Isomorphism Problem, Its Structural Complexity. Birkh¨auser, 1993.
7. O. Pikhurko and O. Verbitsky. Logical complexity of graphs: a survey. 2010.
8. B. Weisfeiler, editor. On construction and identification of graphs. Lecture Notes in Mathematics, Vol. 558. Springer-Verlag, Berlin, 1976. With contributions by A. Lehman, G. M. Adelson-Velsky, V. Arlazarov, I. Faragev, A. Uskov, I. Zuev, M. Rosenfeld and B. Weisfeiler.
9. P. Baldi and G. Pollastri, "The principled design of large-scalerecursive neural network architectures-dag-RNNs and the protein structureprediction problem", *J. Mach. Learn. Res.*, vol. 4, pp. 575-602, 2003.
10. E. Francesconi, P. Frasconi, M. Gori, S. Marinai, J. Sheng, G. Soda, et al., "Logo recognition by recursive neural networks" in Lecture Notes in Computer Science — Graphics Recognition, Germany, Berlin:Springer-Verlag, 1997.

11. M. Bianchini, M. Gori and F. Scarselli, "Processing directed acyclic graphs withrecursive neural networks", *IEEE Trans. Neural Netw.*, vol. 12, no. 6, pp. 1464-1470, Nov. 2001.
12. . Schmitt and C. Goller, "Relating chemical structureto activity: An application of the neural folding architecture", *Proc. Workshop Fuzzy-Neuro Syst./Conf. Eng. Appl. Neural Netw.*, pp. 170-177, 1998.
13. B. Hammer and J. Jain, "Neural methods for non-standard data", *Proc. 12th Eur. Symp. Artif. Neural Netw.*, pp. 281-292, 2004.
14. P. Kaluzny, "Counting stable equilibriaof cellular neural networks-A graph theoretic approach", *Proc. Int. Workshop Cellular Neural Netw. Appl.*, pp. 112-116, 1992.
15. F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner and G. Monfardini, "Computation capabilities of graph neuralnetworks", *IEEE Trans. Neural Netw.*, vol. 20, no. 1, Jan. 2009.
16. F. Pineda, "Generalization of back-propagationto recurrent neural networks", *Phys. Rev. Lett.*, vol. 59, pp. 2229-2232, 1987.
17. М. Рідміллер і Х. Браун, "Прямий адаптивний метод для швидшого зворотного навчання: алгоритм rprop", *Proc. IEEE Int. конф. Нейронна мережа.* , стор. 586-591, 1993.
18. G. Monfardini, V. Di Massa, F. Scarselli and M. Gori, "Graph neural networks for object localization", *Proc. 17th Eur. Conf. Artif. Intell.*, pp. 665-670, 2006-Aug.
19. F. Scarselli and G. Monfardini, *The GNN Toolbox*, [online] Available: http://airgroup.dii.unisi.it/projects/GraphNeuralNetwork/download.htm.
20. S. Muggleton, "Machine learning for systemsbiology", Proc. 15th Int. Conf. Inductive *Logic Programm.*, pp. 416-423, 2005-Aug.-10 – 13.
21. Akoglu L, Tong H, Koutra D. Graph based anomaly detection and description: a survey. Data Min Knowl Discov. 2015;29(3):626–88.
22. Hamilton WL, Ying R, Leskovec J. Representation learning on graphs: methods and applications. 2017. arXiv preprint arXiv:1709.05584

23. Cui P, Wang X, Pei J, Zhu W. A survey on network embedding. TKDE. 2018
24. Goyal P, Ferrara E. Graph embedding techniques, applications, and performance: a survey. Knowl Based Syst. 2018;151:78–94
25. Kipf TN, Welling M. Variational graph auto-encoders. 2016. arXiv preprint arXiv:1611.07308
26. You J, Ying R, Ren X, Hamilton WL, Leskovec J. Graphrnn: a deep generative model for graphs. 2018. arXiv preprint arXiv:1802.08773
27. Velickovic P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y. Graph attention networks. 2017. arXiv preprint arXiv:1710.10903
28. Zhou J, Cui G, Zhang Z, Yang C, Liu Z, Sun M. Graph neural networks: a review of methods and applications. 2018. arXiv preprint arXiv:1812.08434.
29. Wu Z, Pan S, Chen F, Long G, Zhang C, Yu PS. A comprehensive survey on graph neural networks. 2019. arXiv preprint arXiv:1901.00596
30. Li R, Wang S, Zhu F, Huang J. Adaptive graph convolutional neural networks. In: Thirty-second AAAI conference on artifcial intelligence. 2018
31. Yuning You, Tianlong Chen, Zhangyang Wang, and Yang Shen. L2 -gcn: Layer-wise and learned efficient training of graph convolutional networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 2127–2135, 2020.
32. Meng Liu, Hongyang Gao, and Shuiwang Ji. Towards deeper graph neural networks. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 338–348, 2020
33. Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen4, Zhangyang Wang, Yang Shen “Graph Contrastive Learning with Augmentations” 2021
34. Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Pre-training graph neural networks. arXiv preprint arXiv:1905.12265, 2019

35. Dumitru Erhan, Pierre-Antoine Manzagol, Yoshua Bengio, Samy Bengio, and Pascal Vincent. The difficulty of training deep architectures and the effect of unsupervised pre-training. In Artificial Intelligence and Statistics, pages 153–160, 2009.
36. Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. arXiv preprint arXiv:2005.00687, 2020.
37. https://www.researchgate.net/publication/277411157_Deep_Learning

ДОДАТОК А

Відомість матеріалів кваліфікаційної роботи

		Позначення	Найменування	Кільк. аркушів	Примітка
	1				
	2		Документація		
	3				
	4	**ІТКІ.ДП 18.01.ДА.ПЗ**	Пояснювальна записка	69	
	5				
	6		Диск CD-R з презентацією	1	

					ІТКІ.ДП 18.01.ДА.ПЗ			
Зм	Лист	№ докум.	Підпис	Дата				
Розроб.		Шлянчак			**Матеріали кваліфі-каційної роботи**	Літ.	Аркуш	Аркушів
Керівник		Коротенко					1	1
Рецензент		Ширін				НТУ «ДП» 8; 126м-22з-2		
Н.контр.		Коротенко						
Зав. каф.		Гнатушенко						

ДОДАТОК Б

КОД ПРОГРАМИ

data.py

```
import torch
import ast

class Molecule:
    def __init__(self, graph, nodes):
        self.graph = graph.to('cuda')
        self.nodes = nodes.to('cuda')

def load_data():
    data_num = 1112

    graph_labels = open('graph_labels.txt')
    node_labels = open('node_labels.txt')
    node_attributes = open('node_attributes.txt')
    graph_indicators = open('graph_indicators.txt')
    edges = open('graph_edges.txt')

    t0, t1 = 1, 1
    graphs = []
    graph_size_list = []

    for i in range(data_num):
        graph_size = 1
        while True:
            t1 = int(graph_indicators.readline())
            if t0 == t1:
                graph_size += 1
                t0 = t1
            else:
                t0 = t1
                break
        graphs.append(torch.zeros(graph_size, graph_size))
        graph_size_list.append(graph_size)
```

```
s1 = s2 = 0
for i in range(data_num):
    s1 += graph_size_list[i]
    while True:
        edge = literal_eval(edges.readline())
        if edge[0] <= s1 & edge[1] <= s1:
            graphs[i][edge[0] - s2][edge[1] - s2] = 1.0
            graphs[i][edge[1] - s2][edge[0] - s2] = 1.0
        else:
            break
    s2 = s1

node_dim = 4
node_list = []
for i in range(data_num):
    nodes = torch.zeros(graph_size_list[i], node_dim)
    for j in range(graph_size_list[i]):
        node_label = int(node_labels.readline())
        node_attribute = float(node_attributes.readline())
        node_val =  1.0
        if (node_label == 0):
            nodes[j] = torch.tensor([node_val, 0.0, 0.0, node_attribute])
        elif (node_label == 1):
            nodes[j] = torch.tensor([0.0, node_val, 0.0, node_attribute])
        else:
            nodes[j] = torch.tensor([0.0, 0.0, node_val, node_attribute])
    node_list.append(nodes)



molecule = []
labels = []
for i in range(data_num):
    molecule.append(Molecule(graphs[i], node_list[i]))
    labels.append(torch.tensor([float(graph_labels.readline()) - 1.0]).to('cuda'))

graph_labels.close()
node_labels.close()
node_attributes.close()
```

```
    graph_indicators.close()
    edges.close()

    return molecule, labels

m, l = load_data()
```

GCL.py

```
import torch
import torch.nn as nn
import torch.nn.functional as F

class GraphContrastiveLearning(nn.Module):
    def __init__(self, in_features, hidden_features, out_features):
        super(GraphContrastiveLearning, self).__init__()

        self.encoder = nn.Sequential(
            nn.Linear(in_features, hidden_features),
            nn.ReLU(),
            nn.Linear(hidden_features, out_features)
        )

        self.positive_edge_weight = nn.Parameter(torch.ones(1))
        self.negative_edge_weight = nn.Parameter(torch.ones(1))

    def forward(self, graph):
        x = self.encoder(graph.x)

        positive_edges = graph.edges[graph.edges.t() == 1]
        negative_edges = graph.edges[graph.edges.t() == 0]

        positive_scores = torch.mm(x[positive_edges[0]], x[positive_edges[1]])
        negative_scores = torch.mm(x[negative_edges[0]], x[negative_edges[1]])

        loss = self.positive_edge_weight * F.log_softmax(positive_scores) \
               + self.negative_edge_weight * F.log_softmax(-negative_scores)
```

```
        return loss
```

GNN.py

```
import torch

class GraphIsomorphismNetwork:
    def __init__(self, node_dim, update_loop_size):
        self.node_dim = node_dim
        self.update_loop_size = update_loop_size

    def mlp(self, molecule):
        return molecule.nodes + torch.mm(molecule.graph, molecule.nodes)

    def readout(self, molecule):
        return molecule.nodes.sum(dim=0)

    def predict(self, molecule):
        nodes = molecule.nodes
        for _ in range(self.update_loop_size):
            nodes = self.mlp(nodes)
        return torch.cat([self.readout(nodes) for _ in range(self.update_loop_size)])
```

WL.py

```
class GraphObject:
    def __init__(self, graph, nodes):
        self.graph = graph  # Тензор PyTorch матриці суміжності
        self.nodes = nodes  # Список тензорів PyTorch для кожного вузла

def weisfeiler_lehman_graph_hash(molecule, iterations=3):
    adjacency_matrix = graph_object.graph
    node_features = graph_object.nodes
    num_nodes = adjacency_matrix.size(0)
    # Ініціалізація міток для кожної вершини
    labels = {node: hash((tuple(node_features[node][:3].tolist()), node_features[node][3].item())) for node in range(num_nodes)}
    # WL ітеративний процес
    for _ in range(iterations):
        new_labels = {}
```

```
    for node in range(num_nodes):
        neighbors = adjacency_matrix[node].nonzero().view(-1)
        neighbors_labels = [labels[neighbor.item()] for neighbor in neighbors]
        aggregated_label = hash((labels[node], tuple(sorted(neighbors_labels))))
        new_labels[node] = aggregated_label
    labels = new_labels
# Обчислення кінцевого хешу
label_counter = Counter(labels.values())
graph_hash = hash(frozenset(label_counter.items()))
return graph_hash
```

ДОДАТОК В

ВІДГУК
на комплексну кваліфікаційну роботу рівня магістра «Дослідження особливостей процесів використання графів знань у системах на базі графових нейронних мереж» студента групи 126м-22з-2 Шлянчак Світлани Олександрівни

1 Мета даної кваліфікаційної роботи – дослідити особливості побудови різноманітних архітектур графових нейронних мереж, а також процес розв’язання задач ізоморфізму графів чи їх частин класичними методами та за допомогою графової нейронної мережі GIN.

2 Обрана тема актуальна тому, що однією з важливих задач, яка виникає в біології, хімії, криміналістиці, соціології тощо, є задача ізоморфізму графів, яка дозволяє встановити ступінь схожості між собою речовин, ситуацій, обставин та інше. Класичних методів для розв’язку задачі ізоморфізму є не надто багато, бо існує метод, що точно та швидко здатен обробляти невеликі графи. Але цей метод не є ефективним для графів великого розміру. Тож розробка способу розв’язання задачі ізоморфізму великих графів є актуальною..

3 Тема кваліфікаційної роботи відповідного рівня безпосередньо пов'язана з об'єктом діяльності магістра спеціальності 126 «Інформаційні системи та технології» галузі знань 12 «Інформаційні технології» – створення інформаційних технологій різного призначення і застосування.

4 Явища і процеси, що досліджуються в даній кваліфікаційній роботі і обрані для моделювання, оцінювання та реалізації – віднесені в освітньо-кваліфікаційній характеристиці магістрів до класу дослідних та евристичних, рішення яких заснована на знаково-понятійних уміннях.

5 Робота складається з трьох розділів. Перший розділ присвячений аналізу теми дослідження та постановці задачі щодо дослідження особливості побудови різноманітних архітектур графових нейронних мереж. У другому розділі наведено проектну складову вирішення завдання. Третій розділ присвячено дослідженню та порівнянню проявів ізоморфізму графів знань в процесі навчання. Оригінальність отриманих в роботі наукових результатів та їх наукова новизна полягають у наступному:

– досліджено формування та побудову архітектури графових нейронних мереж, що здатні розв’язувати обрану задачу;

– обгрунтовано вибір необхідної архітектури;

– побудувано та виконано навчання нейронної мережі обраної архітектури та досліджені її властивості.

6 Практичне значення результатів роботи полягає в обранні потрібних для вирішення поставленої задачі алгоритмів на основі порівняння групи найбільш відомих.

7 Практичні результати кваліфікаційної роботи отримані із застосуванням відповідних технічних і програмних засобів, мови Python, а

також програмних продуктів MS Word і MS PowerPoint на інформаційно-технологічній платформі Windows.

8 Оформлення графічних матеріалів до кваліфікаційної роботи рівня магістр виконано на сучасному рівні і відповідає вимогам, що пред'являються до рівня виконання робіт даної кваліфікації.

9 Ступінь самостійності виконання кваліфікаційної роботи достатньо висока.

10 В ході виконання роботи автором було виконано достатньо детальний огляд та порівняння найбільш відомих графових моделей та широкого спектру методів обробки графів у графових нейронних мережах. Також, на основі порівняння обраних графової нейронної мережі GIN та графової мережи Graph Contrastive Learning (GCL) в процесі їхнього навчання було з’ясовано, що GCL є більш результативною. Таким чином, отримані результати можуть бути застосовані під час дослідження біологічних та хімічних сполук, розумової діяльності людини, соціальних та економічних явищ тощо. Деякі дискусійні положення та недоліки, які мають місце в роботі:

а) недостатньо повно описано кінцеве практичне застосування отриманих результатів;

б) дощо спрощено описано взаємозв’язок досліджуваних графових моделей.

Незважаючи на вищевказані зауваження, кваліфікаційна робота Шлянчак Світлани Олександрівни цілком відповідає вимогам, що пред’являються до робіт другого (магістерського) рівня спеціальності 126 Інформаційні системи та технології і, в цілому, заслуговує оцінки «___відмінно (96 балів)____» при відповідному захисті та присвоєння здобувачу відповідної кваліфікації.

Керівник кваліфікаційної роботи,
проф. кафедри ІТКІ, д.т.н. Г.М. Коротенко

ДОДАТОК Г

РЕЦЕНЗІЯ
на комплексну кваліфікаційну роботу рівня магістра
«Дослідження особливостей процесів використання графів знань у системах на базі графових нейронних мереж»
студента групи 126м-22з-2 Шлянчак Світлани Олександрівни

Розглянута робота присвячена дослідженню особливостей побудови різноманітних архітектур графових нейронних мереж, а також процес розв'язання задач ізоморфізму графів чи їх частин класичними методами та за допомогою графової нейронної мережі GIN.

Завдання і зміст кваліфікаційної роботи відповідає головній цілі - перевірці знань і ступеня підготовленості студента за фахом 126 «Інформаційні системи та технології» галузі знань 12 «Інформаційні технології».

Зміст пояснювальної записки кваліфікаційної роботи відповідає необхідним критеріям та затвердженій темі.

Актуальність обраної теми обумовлена тим, що дослідження ефективності застосування інформаційних технологій при рішення задач на основі запровадження єлементів штучного інтелекту включає також і задачу ізоморфізму графів.

Повнота і глибина вирішення задач, поставлених в завданні на кваліфікаційну роботу є достатньою.

Оформлення пояснювальної записки кваліфікаційної роботи виконано в повній відповідності з діючими стандартами і нормативними вимогами.

Наукова новизна результатів кваліфікаційної роботи полягає у дослідженні впливу використання графових нейронних мереж на швидкість та точність розв'язання задач ізоморфізму графів та їх частин.

Практичне значення результатів роботи полягає у порівнянні особливості побудови різноманітних архітектур графових нейронних мереж, а також вивченні процесів розв'язання задач ізоморфізму графів чи їх частин класичними методами та за допомогою графової нейронної мережі GIN..

До числа загальних зауважень і недоліків роботи слід віднести:

1) не до кінця розкриті архітектурні особливості побудови графових нейронних мереж;

2) дещо спрощений опис кінцевих результатів роботи.

Однак, зазначені зауваження не здійснюють істотного впливу на підсумкові результати кваліфікаційної роботи і не знижують її безумовну практичну та наукову цінність.

Таким чином, слід зробити висновок, що кваліфікаційна робота в цілому

заслуговує оцінки «_____», а її виконавець присвоєння відповідної кваліфікації.

Рецензент, доцент кафедри програмного
забезпечення комп'ютерних систем НТУ
«ДП», к.т.н.. А.Л. Ширін