ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «МОДЕЛЮВАННЯ КОМП'ЮТЕРНОЇ МЕРЕЖІ НА ОСНОВІ ГРАФОВИХ НЕЙРОННИХ МЕРЕЖ»

на здобуття освітнього ступеня магістра
зі спеціальності 123 Комп'ютерної інженерії
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерні системи та мережі
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ *Владислав ПРИСЯЖНЮК*
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач(ка) вищої освіти гр. КСДМ-61
Владислав ПРИСЯЖНЮК
Ім'я, ПРІЗВИЩЕ

Керівник: Ярослав ТОРОШАНКО
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Рецензент: _____
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Комп'ютерної інженерії

Ступінь вищої освіти Магістр

Спеціальність 123 «Комп'ютерна інженерія»

Освітньо-професійна програма Комп'ютерні системи та мережі

ЗАТВЕРДЖУЮ
Завідувач кафедрою Наталія ЛАЩЕВСЬКА
_____Ім'я, ПРІЗВИЩЕ
«_____» _____ 2024р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Присяжнюк Владислав Дмитрович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Моделювання комп'ютерної мережі на основі графових нейронних мереж»
керівник кваліфікаційної роботи: Ярослав ТОРОШАНКО, канд.техн.наук, доцент ,
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «15» жовтня 2024р. №320
2. Строк подання кваліфікаційної роботи «29» грудня 2024р.
3. Вихідні дані до кваліфікаційної роботи: Науково-технічна література, методи моделювання комп'ютерних мереж за допомогою графових нейронних мереж, інтернет ресурси стосовно моделей машинного навчання
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
    1. Тенденції розвитку комп'ютерних мереж
    2. Основані принципи моделювання комп'ютерних мереж
    3. Комп'ютерне мережеве моделювання за допомогою графових нейронних мереж
5. Перелік ілюстративного матеріалу: *презентація*
    1. Мета, об'єкт, предмет дослідження
    2. Актуальність роботи
    3. Ілюстрація ролей в імітаційній комп'ютерній мережі, подібній до Інтернету
    4. Приклад аналізу першопричини (RCA)
    5. Загальний вигляд ітеративного процесу машинного навчання
    6. Приклад базової топології, що веде до трьох представлених дводольних графів
    7. Алгоритм 1 - Взаємодія між вбудовуваннями
    8. Візуальне представлення підходу A1
    9. Візуальне представлення підходу A2
    10. Порівняння ефективності варіантів абляції при збільшенні навчальної вибірки
    11. 2-компонентна t-SNE-візуалізація вкладених маршрутів
    12. Візуалізація t-SNE вбудованих з'єднань

13. Висновки

6. Дата видачі завдання «01» вересня 2024р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Збір даних	10.10-12.10.2024	
2	Обробка матеріалу	12.10-15.10.2024	
3	Розробка теоретичної частини	15.10-18.10.2024	
4	Тенденції розвитку комп’ютерних мереж	18.10-22.10.2024	
5	Основані принципи моделювання комп'ютерних мереж	22.10-26.10.2024	
6	Комп'ютерне мережеве моделювання за допомогою графових нейронних мереж	26.10-30.11.2024	
7	Висновки за результатами аналізу	30.11-02.12.2024	
8	Розробка презентації	02.12-08.12.2024	
9	Передзахист	09.12-11.12.2024	
10	Здача в деканат	20.12-29.12.2024	

Здобувач вищої освіти _____ Владислав ПРИСЯЖНЮК
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Ярослав ТОРОШАНКО
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістр: 81 стор., 18 рис., 5 табл., 45 джерел.

Мета роботи – підвищення ефективності обробки даних, оптимізація мережевих ресурсів та покращення якості послуг, які надаються користувачам.

Об'єкт дослідження – функціонування та способи моделювання комп'ютерних мереж з використанням графових нейронних мереж.

Предмет дослідження – алгоритми та методи, які застосовуються для побудовиі навчання графових нейронних мереж.

Короткий зміст роботи: в роботі представлено та оцінено передові методи моделювання комп'ютерних мереж за допомогою графових нейронних мереж, проведено аналіз першопричин. Розкрито основні поняття, що лежать в основі машинного навчання, включаючи останні розробки нейронних мереж, такі як згорткові нейронні мережі (CNN) та графові нейронні мережі (GNN). В роботі розглядається застосування узагальнених GNN для прогнозування продуктивності мережевих шляхів.

Запропоновано два різних підходи до вивчення представлень, зіставлених з вузлами мережі, а саме: один з вкладеннями вузлів n (A1) і один без них (A2). Після оптимізації гіперпараметрів було отримано модель машинного навчання, пристосованого для завдання прогнозування затримок на шляху з дуже хорошим узагальненням на невідомі топології мереж. У роботі також наведено дослідження абляції та додаткові візуалізації вивчених репрезентацій, намагаючись повністю охарактеризувати результати дослідження.

КЛЮЧОВІ СЛОВА: КОМП'ЮТЕРНІ МЕРЕЖІ, ГРАФОВІ НЕЙРОННІ МЕРЕЖІ, МОДЕЛЮВАННЯ, ОПТИМІЗАЦІЯ МЕРЕЖЕВИХ РЕСУРСІВ. АЛГОРИТМИ НАВЧАННЯ, МАРШРУТИЗАЦІЯ, ТОПОЛОГІЯ МЕРЕЖІ, АБЛЦІЯ, RCA, ROUTENET, CNN, GGN.

ABSTRACT

The text part of the qualification work for obtaining a master's degree: 81 pages, 18 figures, 5 tables, 45 sources.

Purpose of the work – increasing the efficiency of data processing, optimizing network resources and improving the quality of services provided to users.

Object of the research – functioning and methods of modeling computer networks using graph neural networks.

Summary of the work: the paper presents and evaluates advanced methods for modeling computer networks using graph neural networks, and analyzes the root causes. The basic concepts underlying machine learning are revealed, including the latest developments in neural networks, such as convolutional neural networks (CNNs) and graph neural networks (GNNs). This paper considers the application of generalized GNNs for predicting the performance of network paths.

Two different approaches to studying the representations mapped to network nodes are proposed, namely, one with n nodes (A1) and one without (A2). After optimizing the hyperparameters, a machine learning model adapted to the task of path delay prediction with very good generalization to unknown network topologies is obtained. The paper also presents an ablation study and additional visualizations of the learned representations in an attempt to fully characterize the results of the study.

KEYWORDS: COMPUTER NETWORKS, GRAPH NEURAL NETWORKS, MODELING, OPTIMIZATION OF NETWORK RESOURCES. LEARNING ALGORITHMS, ROUTING, NETWORK TOPOLOGY, ABLATION, RCA, ROUTENET, CNN, GGN

ЗМІСТ

ВСТУП

Інтернет став основним засобом зв'язку для багатьох організацій, людей і машин. Він працює завдяки федерації тисяч мережевих операторів, які співпрацюють, щоб забезпечити швидкий і стійкий зв'язок між будь-якими пристроями на землі. Інтернет багато в чому досягнув приголомшливого успіху, але він все ще стикається з серйозними проблемами. Зокрема, зі зростанням глобального попиту на Інтернет і пропозицією мультимедійних послуг операторам стає дедалі важче забезпечувати належну якість обслуговування споживачів. З іншого боку, складність мережі здебільшого прихована від кінцевих користувачів: на практиці усунення несправностей з'єднання залишається складним завданням.

Нещодавні досягнення в галузі машинного навчання показали, що підходи, засновані на даних, можуть допомогти зрозуміти складні системи, такі як Інтернет. У цій роботі показано, що ці підходи можна використовувати для моделювання великих комп'ютерних мереж, навіть коли наявних знань недостатньо.

У цій роботі продемонстровано, що статистичне навчання може допомогти моделювати великі комп'ютерні мережі зі складними взаємозалежностями, навіть з обмеженою інформацією про оцінювані мережі. Це твердження базується на застосуванні останніх досягнень в методах обробки зображень і графіків до комп'ютерних мереж, використовуючи дані, зібрані як з модельованих, так і з реальних мереж. Висновки роботи виступають за інтеграцію методів, що базуються на даних, у проектуванні та управлінні комп'ютерними мережами. Хоча ці мережі стають дедалі складнішими і важчими в управлінні, ці методи мають потенціал для покращення продуктивності мережі, одночасно зменшуючи ризики простоїв та операційні витрати - фундаментальні показники для мережевих операторів. З іншого боку, дослідження показують, що зовнішні спостерігачі можуть використовувати статистичне навчання для вивчення характеристик комп'ютерної мережі, незважаючи на те, що мають мало попередньої інформації про неї. Це дуже важливо для багатьох суб'єктів, включаючи споживачів, які бажають оцінити

мережевих операторів або сторонні дослідницькі кампанії. Хоча було зосереджено увагу на комп'ютерних мережах, можна вважати, що більшість результатів можуть бути застосовані до інших сфер взаємозалежних систем з обмеженими змінами. Наприклад, результати можуть бути застосовані до будь-яких кіберфізичних систем (розумних мереж або систем промислового управління), соціальних і транспортних мереж або хімічних і біологічних процесів.

РОЗДІЛ 1 ТЕНДЕНЦІЇ РОЗВИТКУ КОМП'ЮТЕРНИХ МЕРЕЖ

1.1 Сутність поняття комп'ютерних мереж

Ця робота присвячена комп'ютерним мережам, як це формально заперечує Ендрю С. Таненбаум у своїй книзі «Комп'ютерні мережі»: «сукупність автономних комп'ютерів, пов'язаних між собою єдиною технологією». Більш конкретно, ми зацікавлені у вивченні мереж, які називаються інтернет-мережами (одним з очевидних прикладів є Інтернет). Для простоти ми називаємо кожен комп'ютер у мережі вузлом, будь то персональний комп'ютер, мобільний телефон, пристрій, що носиться, виділений сервер, «розумний» автомобіль тощо. Два вузли, з'єднані зв'язком, можуть комунікувати, обмінюючись повідомленнями за допомогою спільної технології (наприклад, Bluetooth Low Energy (BLE) або Інтернет-протокол (IP) через Ethernet або Wi-Fi). Нарешті, шлях визначається як впорядкований список вузлів і зв'язків і описує маршрут між двома вузлами, які можуть бути не з'єднані безпосередньо. Вузли, канали та шляхи формують основні компоненти кожної комп'ютерної мережі (рис. 1.1). Завдяки своїй універсальності, ці поняття можна легко перенести на інші види мереж (наприклад, транспортні, соціальні, біологічні мережі).

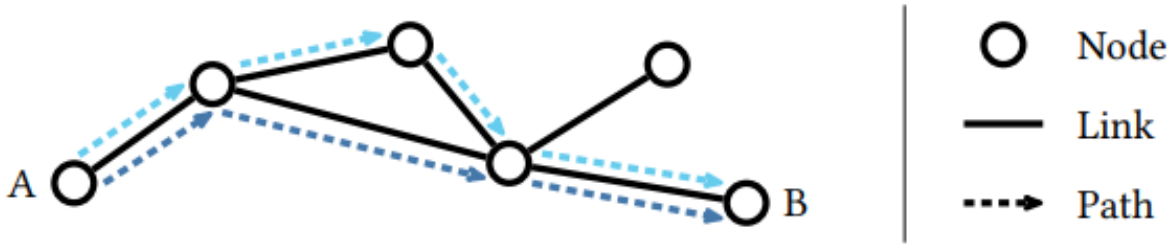


Рисунок 1.1 - Ілюстрація комп'ютерної мережі з двома можливими шляхами від вузла A до B

1.1.1 Корисні метрики

На практиці з'єднання комп'ютерних мереж характеризуються затримкою, джиттером, коефіцієнтом втрат і пропускною здатністю. Затримка - це середній час, необхідний для проходження повідомлення через канал, тоді як джиттер - це середнє відхилення від цієї затримки. Відношення втрачених повідомлень до загальної кількості заперечується коефіцієнтом втрат. Нарешті, пропускна здатність - це кількість інформації, яка може бути передана через канал за одиницю часу: вона зазвичай вимірюється в бітах за секунду.

Пропускну здатність не слід плутати з доступною пропускною здатністю, тобто залишковою пропускною здатністю каналу, коли він вже використовується. Важливо зазначити, що зазвичай неможливо виміряти безпосередньо цю пропускну здатність: натомість ми вимірюємо пропускну здатність або «корисну потужність» Це означає, що затримка може бути нижчою за пропускну здатність (наприклад, через нижчу доступну пропускну здатність, накладні витрати на протокол або ретрансляцію), а іноді й вищою (наприклад, через стиснення та буферизацію). Аналогічно, затримка в основному оцінюється за допомогою часу проходження в обидва кінці (RTT), більш простого підходу, ніж час проходження в один кінець, який вимагає точної синхронізації годинника. Вузли можуть бути охарактеризовані додатковою затримкою, яку вони додають при обробці повідомлень (змодельованою параметрами черги або буферизації).

1.1.2 Ролі цільових суб'єктів мережі

Рішення проблем комп'ютерних мереж значною мірою залежать від припущень, зроблених щодо цільових суб'єктів мереж. Чи призначене одне рішення для мережевих адміністраторів і потребує повних знань про мережеві

компоненти? Або ж воно розраховане на роботу з обмеженими знаннями, а тому орієнтоване на користувачів мережі? Для того, щоб порівняти припущення, зроблені в дослідженні, з попередніми роботами, було визначено загальний набір ролей на основі архітектури Інтернету (рисунок 1.2 для ілюстрації топології імітаційної мережі).

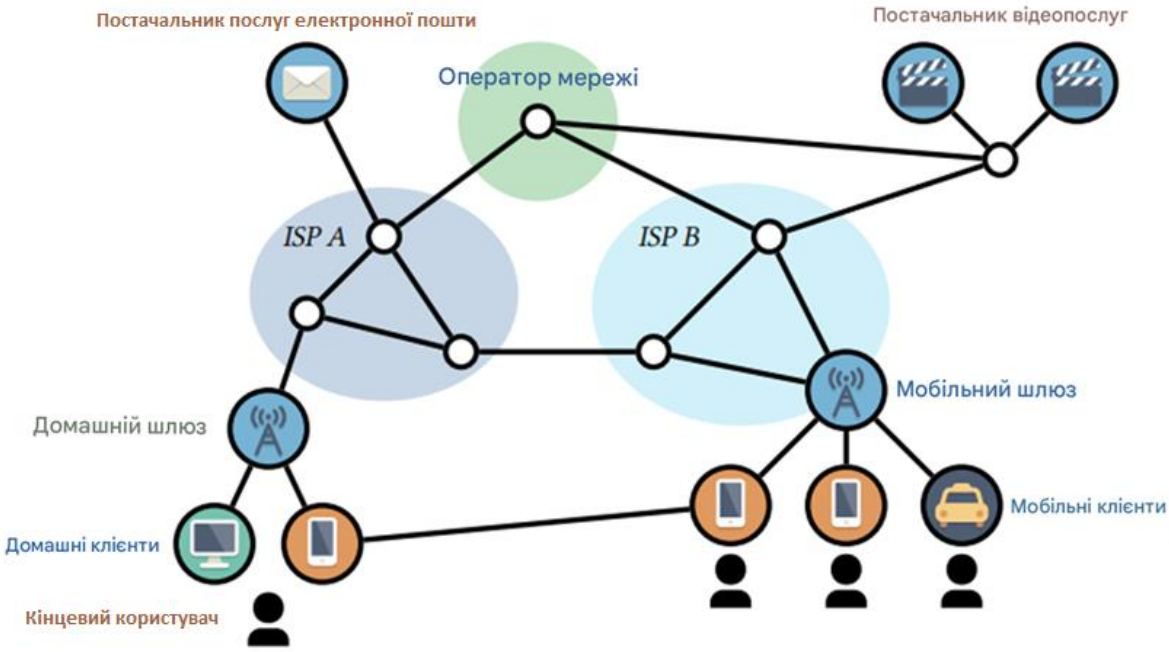


Рисунок 1.2 - Ілюстрація ролей в імітаційній комп'ютерній мережі, подібній до Інтернету

Кінцеві користувачі можуть використовувати свої клієнти для доступу до електронної пошти та відеосервісів через шлюз, наданий провайдером. Слід зауважити, що два клієнти можуть безпосередньо спілкуватися, наприклад, з BLE

Перш за все, кожен оператор мережі відповідає за обслуговування частини мережі. Можна припустити, що мережеві оператори мають повне знання властивостей вузлів і з’єднань у своїй зоні відповідальності. В Інтернеті можна приблизно відобразити мережевих операторів до автономних систем (AS).

Інтернет-провайдери (ISP) є особливим видом операторів: вони надають своїм клієнтам доступ до глобальної мережі.

Розрізняються наступні два типи клієнтів:

- постачальники послуг розгортають комп'ютери в мережі, які надають спеціальні послуги, такі як потокове аудіо або відео, завантаження, веб-сайти електронної комерції тощо. Розгорнуті комп'ютери зазвичай називають «серверами»;

- кінцеві користувачі, з іншого боку, використовують свої комп'ютери для доступу до вищезгаданих послуг або спілкування з іншими кінцевими користувачами. Їхні комп'ютери називаються «клієнтами», на відміну від серверів (не плутати з терміном «замовник»). Слід зазначити, що ця модель є спробою спростити та узагальнити інтернет-мережі, складні та децентралізовані системи.

Наприклад, великі постачальники послуг можуть виступати в ролі мережевих операторів, щоб оптимізувати свої витрати та якість послуг. І навпаки, інтернет-провайдери зазвичай надають інші корисні послуги, такі як поштові скриньки або хостинг веб-сайтів. Проте в даній роботі класифікація ролей відповідає існуючим мережевим моделям і дозволяє легко порівнювати рішення.

1.2 Зростаюча складність комп'ютерних мереж

Попит на швидший Інтернет за останні роки різко зріс у всьому світі. Згідно з «Фактами і цифрами» Міжнародного союзу електрозв'язку (МСЕ), у 2022 році понад 50% населення світу було підключено до Інтернету, порівняно з менш ніж 30% десять років тому. Зокрема, 69% людей у віці 15-24 років мають доступ до Інтернету. Кількість домогосподарств з широкосмуговим доступом також подвоїлася з 30% у 2012 році до 57% у 2022 році. Разом зі зростанням кількості підключених людей значно збільшилася і пропускна здатність, необхідна кожній людині. Така еволюція є природною з огляду на розвиток медіа-технологій, що дозволяють дешево захоплювати, обробляти і відображати контент високої роздільної здатності (наприклад, зображення, музику, відео, ігри, ...). Наприклад, за оцінками Cisco, кількість телевізорів з роздільною здатністю 4K подвоїлася між

2020 і 2022 роками. Ця тенденція продовжуватиметься, оскільки з часом все більше пристроїв також підключаються до Інтернету: прогнози показують, що понад 13 мільярдів пристроїв можуть бути підключені до Інтернету в 2025 року, включаючи пристрої, що потребують широкосмугового зв'язку, такі як автономні транспортні засоби або хмарні камери спостереження. Криза Covid-19 ще більше підвищила попит на широкосмугове відстеження через регіональні локдауни та більший інтерес до дистанційної роботи. (Наприклад, в Італії під час карантину 2020 року було використано на 44% більше відстеження, ніж за аналогічний період 2019 року).

Щоб задовольнити цей зростаючий попит, інтернет-оператори інвестували в розширення своїх мереж: за даними MCE, з 2019 року до середини 2020 року в усьому світі було додано 200 Тбіт/с додаткової пропускної здатності, що в цілому становить 700 Тбіт/с. Починаючи з 2018 року, середня світова пропускна здатність широкосмугового (мобільного) зв'язку для кінцевих користувачів зростала на 20% (або 27%) на рік і, ймовірно, досягне 110 Мбіт/с до 2025 року (або 44 Мбіт/с). Це означає, що розгортається все більше міжконтинентальних кабелів, локальних волокон і антен, але стратегія, яку використовують оператори для управління своїми мережами, також повинна еволюціонувати.

Однією з важливих змін є перехід до програмно-керованих мереж (SDN), де площина управління мережею відокремлена від площини передачі (даних): ця парадигма дозволяє операторам краще адаптуватися до динамічних вимог своїх клієнтів. За даними 40% опитаних операторів вже перейшли на такі мережі, і ще 55% операторів очікують розгортання протягом двох років. SDN теоретично дозволяє автоматизувати більшість завдань трасування, що дає змогу оптимізувати мережеві конгруенції, маршрутизацію та пірингові політики між операторами. У мережах 5G (які за задумом підтримують SDN) оператори заохочуються до розгортання мережевих функцій «на периферії», щоб уможливити нові варіанти використання, такі як хмарні ігри або великомасштабна телеметрична аналітика. На практиці це означає розгортання значних обчислювальних ресурсів у центрах обробки даних або навіть антен поблизу кінцевих користувачів. Інтернет-сервіси

та додатки також повинні адаптувати свою інфраструктуру для задоволення попиту, наприклад, шляхом розгортання розподілених систем у різних точках світу. Знову ж таки, незважаючи на широкий спектр хмарних сервісів, ці розподілені системи природно додають складності та нових залежностей у рівняння. Наприклад, Netflix розгорнула сервери доставки у понад 500 місцях, щоб задовольнити світовий попит на потокове відео, що дозволило сервісу стати найбільшим постачальником контенту у Франції у 2020 році, випередивши Google та його відеосервіс Youtube.

Однак усі ці додаткові складнощі можуть мати зворотний ефект і призвести до серйозних перебоїв у роботі. 17 липня 2020 року, намагаючись вирішити відносно легку проблему перевантаження мережі, команда інженерів Cloudflare оновила конфігурацію маршрутизаторів. Однак ця зміна конфігурації містила помилку, яка перенаправила всю трасування Cloudflare оновила конфігурацію маршрутизаторів. Однак ця зміна конфігурації містила помилку, яка перенаправила всю трасу Cloudare через одну мережеву точку: це призвело до глобального відключення більшості з 12 мільйонів веб-сайтів, які обслуговує компанія, приблизно на 30 хвилин. Автоматизовані системи також можуть давати ефектні збої: 30 серпня 2020 року CenturyLink/Level3 (один з найбільших мережевих операторів за версією CAIDA) спровокував невелику зміну для захисту клієнта від кібератаки. Ланцюжок автоматизованих подій змусив CenturyLink/Level3 повідомити інших мережевих операторів про величезну кількість змін маршрутизації, що призвело до відключення величезної частини Інтернету майже на п'ять годин. У цих прикладах зв'язок між першопричиною і симптомами, що спостерігалися, не був очевидним - навіть для інсайдерів - через численні залежності між компонентами мережі. Для зовнішніх спостерігачів, таких як дослідники або кінцеві користувачі, усунення несправностей і розуміння перебоїв в роботі Інтернету є ще більш складним завданням, оскільки більшість цих залежностей є прихованими (з технічних або комерційних причин). Хоча деяка загальнодоступна інформація доступна через точки обміну даними в Інтернеті

(IXP), більшу частину топології Інтернету і зв'язків між учасниками мережі важко зрозуміти.

1.3 Підходи до моделювання комп’ютерних мереж на основі даних

Програми для автоматизації, оптимізації та прийняття рішень широко використовуються операторами мереж у спробі забезпечити високу якість послуг для своїх клієнтів та зменшити операційні витрати. Історично засновані на статичних правилах прийняття рішень, розроблених експертами, ці програми можуть мати труднощі з розумінням складних залежностей сучасних комп'ютерних мереж. Проте зростає інтерес до статистичних і основаних на даних підходів до проектування та експлуатації мереж: алгоритми машинного навчання використовують великі обсяги даних, зібраних операторами, для створення релевантних моделей своїх мереж.

Проте застосування методів машинного навчання до комп'ютерних мереж та Інтернету залишається складним на практиці через великий масштаб керованих мереж. Ці мережі справді важко контролювати, щоб отримувати безперервні вимірювання та актуальну інформацію. Більше того, з нещодавнім переходом до туманних і периферійних обчислень, все більше управлінських рішень автоматично приймається обладнанням з обмеженими знаннями про всю мережу.

За останнє десятиліття прогрес в обчислювальній ефективності уможливив проривні рішення з нейронними мережами та моделями глибокого навчання. Одним з дуже популярних прикладів є AlphaGo від Google, перше програмне забезпечення, здатне обіграти професійних гравців. Інші приклади включають широкий спектр областей, від історичної теми аналізу зображень до синтезу мультимедійного контенту, моделей перекладу на основі даних або аналізу кіберфізичних систем.

Завдяки своїй виразності ці рішення виявилися кращими в моделюванні складних систем і залежностей, ніж більшість традиційних підходів. Зокрема, за наявності достатньої кількості даних моделі глибокого навчання здатні витягувати («вивчати») складні приховані взаємозв'язки у вхідних даних, що призводить до більш точних і узагальнюючих рішень.

1.3.1 Моделювання комп'ютерної мережі для швидкого прогнозування продуктивності

Через складні взаємозалежності між компонентами комп'ютерної мережі, зазвичай важко передбачити, як певна мережева конфігурація буде працювати на практиці. Це основна проблема для мережевих операторів, які бажають оптимізувати свої конфігурації без створення проблем: невеликі зміни в одній частині мережі можуть призвести до різких змін продуктивності в інших, здавалося б, не пов'язаних між собою частинах. Історично оператори використовували експертні знання та моделювання, але ці підходи можуть бути неточними і повільними. У контексті великомасштабних динамічних мереж сьогодні потрібні нові надійні та ефективні рішення.

У зв'язку з цим була запропонована парадигма Knowledge-Densed Networking (KDN). Вона пропонує використовувати комбіновані можливості SDN (полегшений збір метрик і управління мережею) і машинного навчання для ефективного проектування і управління великими і складними комп'ютерними мережами. У KDN мережі повинні моделюватися на основі відомої конфігурації мережі, щоб можна було застосовувати автоматизовані рішення. Однією з цілей такого моделювання є прогнозування продуктивності мережі з урахуванням змін конфігурації для прийняття найкращих рішень.

В роботі пропонується моделювання залежності між компонентами мережі у множинних двосторонніх графах. Було застосовано сучасні методи GGN до цієї

моделі залежностей і отримано прогнози з дуже хорошою точністю для великих мереж за кілька сотень мілісекунд (на відміну від інструментів моделювання, які вимагають кілька хвилин для тих же прогнозів). Зокрема, даний підхід може працювати з мережевими компонентами з різними політиками планування. Було показано, що його можна узагальнити на різні мережеві топології та конгруенції і дослідити його внутрішню роботу за допомогою візуалізації вивчених репрезентацій.

Користувачі широкосмугового і мобільного Інтернету регулярно стикаються з погіршенням продуктивності і проблемами підключення при доступі до послуг через Інтернет. Під час таких інцидентів кінцевим користувачам складно визначити першопричини спостережуваних симптомів: чи це пов'язано з поганим бездротовим покриттям? Простоєм сервісу? Неправильне налаштування пристрою? Користувачі, природно, звинувачують інтернет-провайдерів (ISP) у багатьох погіршеннях, оскільки саме вони відповідають за забезпечення якісного інтернет-з'єднання. Як наслідок, до їхніх служб підтримки часто звертаються з приводу симптомів, які насправді не спричинені провайдером. Це призводить до додаткових витрат на підтримку та погіршення відносин з клієнтами: У 2018 році обслуговування клієнтів у сфері телебачення та телекомунікацій було визнано одним з найгірших в Україні. Кінцеві користувачі та провайдери виграють від автоматизованого аналізу першопричин (RCA), здатного визначити найбільш ймовірні причини збоїв. Клієнти зможуть швидко виявляти збої, уникаючи найбільш трудомістких телефонних дзвінків, що призведе до більшої задоволеності (75% громадян України з молодших поколінь надають перевагу письмовому спілкуванню у відносинах з клієнтами); і навпаки, провайдери отримуватимуть звіти лише про першопричини, які, ймовірно, лежать в їхній зоні відповідальності. Постраждалі сторонні інтернет-сервіси також виграють від кращого клієнтського досвіду та автоматичних звітів, оскільки навіть невелике погіршення продуктивності може призвести до зниження доходів. У цьому контексті наявні знання різко скорочуються порівняно з попередньою проблемою: при проведенні RCA з точки зору одного користувача неможливо мати доступ до повної топології

та залежностей інтернет-сервісів. Щоб вирішити цю проблему, було запропоновано використовувати комбінацію краудсорсингових вимірювань і використання еталонних («опорних») вимірювальних серверів, розгорнутих у багатьох точках Інтернету: збираючи метрики з цих точок, показано, що можна виявити багато першопричин навіть без співпраці з провайдерами. Було використано зібраний набір даних з контрольованим і показують, що згорткова нейронна мережа (CNN) перевершує два альтернативні рішення, отримані на основі типових алгоритмів машинного навчання. Зокрема, представлені моделі CNN є розширюваними і підтримують динамічні сервери орієнтирів.

1.4 Використання веб-браузерів для збору мережевих метрик

Третій результат, який було представлено в цій роботі, пов'язаний зі збором метрик, важливих для RCA. У цій роботі підкреслено необхідність метрик якості обслуговування (QoE) в практичній структурі RCA, що сприяє розвитку інструментарію середовища виконання на пристроях кінцевого користувача в цьому відношенні. Зокрема, истверджується, що веб-браузери є хорошими платформами для збору метрик, які можуть забезпечити точні оцінки як QoE, так і мережевих метрик (метрик якості обслуговування (QoS)). Запропоновано набір методів, основаних на можливостях браузера, з урахуванням обмежень безпеки сучасних веб-браузерів.

В результаті аналізу метрик, використовуючи запропонований фреймворк, було виділено кілька цікавих висновків із зібраного набору даних, включаючи зміни QoE з плином часу, точне визначення диференціації трафіку або важливі зміни в інтернет-маршрутизації. Також запропоновано статистичний підхід до визначення несправностей і пов'язаних з ними першопричин, які спостерігались в наборі даних. Цей метод збору даних є необхідним для будь-якого рішення, основаного на даних: він дозволяє оцінити запропоновані методи RCA в Інтернеті

з реальними несправностями і погіршенням QoE. Оскільки даний набір даних значно відрізняється від попереднього набору даних, який було використано з контрольованими несправностями, було отримано відмінні результати, які показують, що типові базові лінії машинного навчання залишаються цікавими для проблеми RCA.

Хоча загальна продуктивність є нижчою, дане дослідження все ще демонструє інтерес статистичного навчання для RCA і піднімає низку цікавих питань та напрямків досліджень для подальшої роботи в цій галузі.

РОЗДІЛ 2 ОСНОВАНІ ПРИНЦИПИ МОДЕЛЮВАННЯ КОМП'ЮТЕРНИХ МЕРЕЖ

Моделювання комп'ютерної мережі має важливе значення для розуміння та прогнозування її роботи. Існує багато способів виконати це завдання, і в роботі представлено існуючі методи.

Під час дослідження було припущено, що задача розв’язується від імені оператора мережі, який має повну інформацію про мережу, яку необхідно змоделювати. У цьому випадку методи моделювання зазвичай використовуються під час проектування мережі. На противагу цьому, також обговорюються методи, що працюють з обмеженими знаннями: ці методи найкраще підходять під час експлуатації мережі, або коли необхідно вирішити проблему для не операторів (наприклад, для кінцевих користувачів).

2.1 Моделювання на основі повних знань про мережу

В роботі було представлено методи моделювання мережі, коли відома топологія мережі та більшість характеристик компонентів, які можна розділити на чотири широкі категорії:

- теорія черг;
- мережі Петрі;
- моделювання дискретних подій;
- емуляція мережі.

Нижче розглядається кожна з них по черзі.

2.1.1 Теорія черг

Вперше модель телекомунікаційної мережі з чергами була запропонована Агнером К. Ерлангом у 1917 році для оцінки середнього часу очікування на автоматичних телефонних станціях. Теорія масового обслуговування дозволяє аналітично передбачити очікувану поведінку черги відповідно до швидкості прибуття «клієнтів», розподілу часу обслуговування та кількості «серверів».

Кілька черг можуть бути об'єднані в мережі масового обслуговування. Ця теоретична основа була перенесена на комп'ютерні мережі Леонардом Клейнроком, що призвело до розвитку ARPANET (прабатька Інтернету). Однак було визнано, що аналітичні обчислення швидко стають нерозв'язними, особливо для політик черг, складніших, ніж «rst-in, rst-out». Зокрема, більшість мереж масового обслуговування припускають незалежність швидкості прибуття черг, що не відповідає дійсності. Нещодавні роботи покладаються на кореляцію та апроксимацію швидкостей для аналізу відносно невеликих мереж масового обслуговування.

2.1.2 Мережі Петрі

Введені в 1962 році, мережі Петрі доповнюють теорії автоматів і черг шляхом явного моделювання залежностей. Вони позначаються графом з двома типами вершин: місцями та переходами. Важливим обмеженням є те, що типи вершин чергуються у дводольному графі: жодне місце (або перехід) не пов'язане безпосередньо з іншим місцем (або переходом). Місця може містити будь-яку кількість токенів (наприклад, комунікаційних пакетів). Після заперечення, мережа Петрі може працювати за допомогою кільцевих переходів, що дозволяє токенам бути в боргу в мережі. Як і заперечував Карл Петрі, ці мережі не можуть точно

моделювати черги і затримки, присутні в комп'ютерних мережах. Тому велика кількість робіт присвячена вивченню часових мереж Петрі - варіанту, який краще підходить для цього випадку і має багато успішних застосувань.

Як додатковий варіант, були запропоновані мережі Петрі з чергами для моделювання затримок у реальних мережах. Зрештою, ці розширені мережі Петрі можуть бути перетворені на ланцюги Маркова, що дозволяє аналітично аналізувати властивості мережі подібно до мереж масового обслуговування, з аналогічними обмеженнями на пропускну здатність.

2.1.3 Моделювання дискретних подій

Реальні комп'ютерні мережі не можуть бути розв'язані аналітично через нерозв'язну кількість станів, в яких вони можуть перебувати. Проте, можна вдатися до моделювання, щоб передбачити продуктивність мережі. Зокрема, моделювання дискретних подій є практичним підходом, який застосовують найпопулярніші симулятори комп'ютерних мереж, серед яких NS-3 та OMNeT++.

У мережевих симуляторах події, такі як «повідомлення, отримане у вузлі x», обробляються одна за одною: симулятор переходить від події до події, оскільки не передбачається, що між послідовними подіями нічого не відбувається. Одночасних подій можна уникнути, використовуючи стохастичні тривалості, і стає можливим виміряти продуктивність комп'ютерної мережі в будь-який момент часу, навіть для складних мереж. Наприклад, NS-3 надає моделі для мережевих стеків Linux і для бездротового зв'язку, такого як Wi-Fi. Незважаючи на значну оптимізацію, симулятори залишаються обмеженими доступними обчислювальними потужностями; вартість моделювання зростає зі збільшенням розміру оцінюваної мережі та кількості подій, тобто кількості обмінюваних повідомлень. Крім того, слід бути обережним при встановленні параметрів моделювання, оскільки невеликі помилки можуть призвести до нереалістичних результатів.

OMNeT++, наприклад, використовувався для створення набору даних про продуктивність комп'ютерної мережі з урахуванням багатьох різних конфігурацій. Хоча використання цього симулятора на практиці є повільним (на кожну конфігурацію потрібно кілька хвилин обчислень), отриманий набір даних можна широко використовувати для розробки та перевірки інших підходів, що базуються на даних.

2.1.4 Емуляція

Більш сучасною альтернативою симуляції є емуляція: мережеві стеки реальних комп'ютерів використовуються для обміну реальними повідомленнями. Завданням емуляції є відтворення мережевої моделі шляхом емуляції характеристик вузлів і з'єднань. Наприклад, ядро Linux надає netem набір інструментів для додавання затримок, втрат або формування пропускної здатності до отриманих або надісланих повідомлень. Емулятор MiniNet використовує простори імен netem і процесів для емуляції довільної комп'ютерної мережі на одному хості. Останні системи, такі як Kollaps або AdvantEdge, можуть створювати мережі, що охоплюють декілька хостів, завдяки контейнерним технологіям та інструментарію Kubernetes. Емуляція бездротової мережі можлива за допомогою спеціальних тестових стендів, хоча в більшості випадків вона менш практична, ніж симуляція.

Емуляція має багато переваг: вона дозволяє оцінити існуючі додатки в середовищі, близькому до реальної мережі, з низькими накладними витратами, значно знижує обчислювальні витрати і забезпечує прийнятний рівень відтворюваності. Однак деякі обмеження залишаються.

По-перше, емульована мережа обмежена пропускною здатністю фізичної мережі між емуляційними комп'ютерами - наприклад, неможливо емулювати зв'язок із затримкою в 1 мс над фізичним зв'язком із затримкою в 5 мс. Крім того,

одночасна передача повідомлень може бути проблематичною при моделюванні обмеженої пропускної здатності каналу. Зовсім недавно в кількох роботах було запропоновано моделювати комп'ютерні мережі з використанням методів машинного навчання і, зокрема, нейронних мереж.

2.2 Моделювання на основі обмежених знань

Під час роботи мережі наявні знання про комп'ютерну мережу можуть бути набагато меншими, ніж під час її проектування. Це може бути пов'язано з експлуатаційними обмеженнями (наприклад, неможливістю контролювати кожне з’єднання мережі) або взаємодією між різними сферами відповідальності. В Інтернеті жоден провайдер не має повного знання про кожен компонент всієї мережі; навіть управління всіма характеристиками компонентів у власній AS, як відомо, є складною проблемою. Це ще більш складно для кінцевих користувачів, які можуть спостерігати за поведінкою лише з далекого краю комп'ютерної мережі.

Існує кілька робіт, які намагаються виявити більше інформації про мережу і вивести характеристики компонентів з глобальних спостережень. Наприклад, виявлення мережі за допомогою зондів. Однією з частих проблем при роботі з великими мережами є відсутність відомої топології. Основним рішенням для збору інформації про топологію мережі є надсилання зондів у різні вузли та аналіз відповідей на ці зонди. Наприклад, інструмент traceroute широко використовується для дослідження IP-мереж, використовуючи час життя пакетів (TTL). При проходженні проміжних вузлів на шляху від джерела трасування до місця призначення заголовок TTL IP-пакета зменшується на одиницю. Якщо він досягає нуля, вузол очікується, що відправник початкового пакета може надіслати спеціальну помилку протоколу обміну повідомленнями управління Інтернетом (ICMP) «TTL перевищено» назад відправнику початкового пакета.

Ідея трасування полягає в тому, щоб надсилати послідовні пакети зі збільшенням TTL: теоретично, джерело отримає одну помилку ICMP на проміжний вузол, що дозволить ідентифікувати зв'язки в досліджуваному шляху. Цей метод використовувався для повного виведення топології, але його точність може страждати від пропущених відповідей і добровільного заплутування з боку проміжних вузлів.

Такі методи залишаються активною областю досліджень, нещодавні роботи, наприклад, запропонували використовувати паралельні зонди та стохастичну вибірку для значного покращення ефективності заходів трасування.

Виявлення сервісних залежностей. Вузли обчислювальної мережі покладаються на інші вузли (сервери), що надають послуги. Тому для оператора мережі дуже важливо розуміти взаємозв'язки (залежності) між вузлами, щоб він міг оптимізувати топологію і характеристики своєї мережі. Іншим варіантом використання виявлення залежностей є аналіз першопричин збоїв, який полягає у визначенні походження несправностей, що виникають. Це не тривіальна задача, і було запропоновано декілька методів (таблиця 2.1), що використовують метадані повідомлень, трасування пакетів або контрольовані збурення. Наприклад, Sherlock збирає траси пакетів з проміжних вузлів і оцінює умовну ймовірність доступу до одного сервісу A (залежного) за короткий час до сервісу B (незалежного). Після базової інтерполяції ця умовна ймовірність дає ймовірність залежності між A і B, яка пізніше використовується для аналізу першопричини.

Таблиця 2.1 - Вибрані методи для виведення залежності від послуг

Reference	Year	Short description
Pinpoint [57]	2002	J2EE communication layer to add tracing in queries.
Tulip [58]	2003	Tracing log written in packets by traversed Internet routers.
Sherlock [59]	2007	Packet traces are extracted from routers and later correlated.
X-Trace [60]	2007	Tasks tagged using metadata.
Orion [61]	2008	Correlation of delays extracted from packet traces.
NetMedic [62]	2009	Instrumented Windows sockets and firewall.
WebProphet [63]	2010	Perturbated queries using a controlled proxy.

Мережева томографія Моше Варді (Moshe Y. Vardi) формалізував задачу мережевої томографії у 1996 році. Мета томографії полягає в тому, щоб оцінити набір параметрів x за набором спостережень y, знаючи матрицю маршрутизації A в наступному рівнянні:

$$y = A \cdot x \tag{2.1}$$

Наприклад, в задачі оцінки «Джерело-Призначення» оцінюється трафік для кожного шляху в мережі (x) (кожної пари джерело-призначення) на основі звітів про трафік в кожній ланці y. Це, по суті, обернена задача, яка може бути вирішена за допомогою ітераційних алгоритмів, таких як розв'язувачі за методом найменших квадратів. Інша проблема полягає в оцінці характеристик канального рівня, таких як затримка і втрати (x), на основі вимірювань наскрізного шляху (y). Для вирішення цієї проблеми можна використовувати багатоадресні зонди, але вони зазвичай не застосовуються в більшості реальних комп'ютерних мереж, де перевага надається одноадресним зондам.

При ідентифікації топології матриця маршрутизації A також може бути виведена з лісу можливих топологій. Однак у більшості випадків A є рангово-залежною, що призводить до множинних розв'язків.

Один із способів полегшити розв'язання такого роду рівнянь - визначити, які характеристики мережі можна оцінити («ідентифікувати»), і відповідно розмістити монітори в комп'ютерній мережі. Інші підходи використовують додаткову статистичну інформацію про з’єднання. Зовсім недавно було запропоновано кілька методів для врахування наслідків збоїв під час томографії.

2.3 Аналіз першопричин відмов в комп’ютерній мережі

Одним з відомих випадків використання моделей, які було описано в попередньому розділі, є здатність розуміти причини збоїв. Більш конкретно,

дослідження зосереджено на локалізації першопричин відмов (Root cause analysis – RCA).

Виявлення збоїв і RCA - це дві різні проблеми з різними рішеннями: методи виявлення використовують виявлення відхилень для виявлення аномалій, тоді як RCA намагається знайти джерело(а) аномалій після їх виявлення.

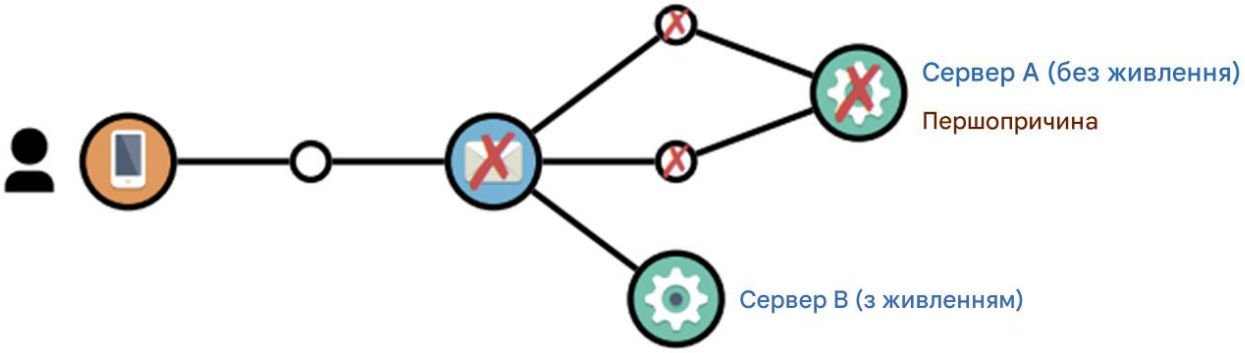


Рисунок 2.1 - Приклад аналізу першопричини (RCA): один кінцевий користувач не може отримати доступ до свого поштового сервісу через збій живлення «Сервера А» (першопричина). Інші компоненти, що покладаються на цей сервер, виявляються несправними (наприклад, служба електронної пошти)

Рисунок 2.1 ілюструє завдання RCA в простій ситуації. У цьому прикладі один користувач стикається з фатальними помилками при спробі отримати доступ до своєї електронної пошти. Цей користувач може припустити, що весь поштовий сервіс несправний, або що посилання на сервіс не працюють належним чином. Ідеальна система RCA мала б точно визначати причину збою (яку зазвичай називають «несправністю»). У цьому випадку причиною є відключення живлення критично важливого внутрішнього сервера, що робить поштову службу недоступною через ланцюжок залежностей.

На відміну від цього, детектори несправностей висвітлили б проблеми в усіх «перехрещених» компонентах (X), що ускладнило б пошук і усунення несправностей.

Тип методів RCA, які можна застосувати, значною мірою залежить від типу та обсягу наявної інформації; це, в свою чергу, залежить від того, як дані та знання отримуються на практиці. Інтернет-провайдери можуть використовувати свої

внутрішні знання або метрики SDN для правильної побудови графів залежностей між компонентами мережі. У більшості дата-центрів топологія комп'ютерної мережі є надлишковою та ієрархічною в Clos-подібних схемах (наприклад, у Facebook або Azure сервери з'єднані між собою за 3-шаровою схемою з надлишковими зв'язками між шарами).

Такий вибір дозволяє проводити подальші оптимізації та розподіляти RCA між компонентами мережі. Наприклад, маршрутизатори можуть голосувати за підозрілі несправні компоненти у своїй стійці. Без заглядання всередину AS або центрів обробки даних, кінцеві користувачі все ще можуть отримати певні знання про мережу з загальнодоступних даних.

Трасування маршрутів, загальнодоступні канали протоколу прикордонного шлюзу (BGP) та бази даних IXP були успішно використані для виявлення та локалізації збоїв. Пасивний моніторинг трасування також можливий за допомогою фонового випромінювання Інтернету: варіації трасування, що надходять до нерозподілених IP-прексів, можуть вказувати на серйозні глобальні перебої в роботі. Проте, зважаючи на дефіцит даних про внутрішню топологію та пірінг AS, для незернистого RCA необхідна більш тісна співпраця між кінцевими користувачами та провайдерами. Нижче розглянуто доступні рішення RCA відповідно до обсягу наявних знань про мережу.

2.3.1 RCA на основі повних знань про мережу

Багато методів RCA (таблиця 2.2) вимагають великих знань про аналізовану комп'ютерну мережу. Ранні спроби пропонували використовувати бінарну томографію мережі: при такому підході компонент мережі є або номінальним, або несправним. Мета полягає в тому, щоб визначити, які компоненти є несправними, на основі наскрізних вимірювань на мережевих шляхах. Це еквівалентно розв'язанню задачі про мінімальну кількість влучень, яка, як відомо, є NP-важкою.

Таким чином, евристики використовуються для отримання результату за практичний час. Як приклад, автори Tomo пропонують ітеративний алгоритм: поки існують непояснені відмови, Tomo обчислює оцінку для кожного з'єднання l як кількість непояснених відмов, які можна пояснити тим, що l несправна. Потім він вибирає з'єднання з найвищим показником і повторює цикл. Цей простий підхід не працює з динамічними мережами і перехідними відмовами, що робить його складним для використання в RCA на практиці.

Пізніші роботи моделюють ймовірності відмов замість двійкових значень . У цьому ж ключі для RCA використовували байєсівські мережі. Ці мережі - іноді відомі як мережі переконань - призначені для моделювання умовних ймовірностей залежних випадкових величин у спрямованому ациклічному графі. Однак точний висновок у байєсівських мережах також є NP-важкою проблемою: методи моделювання комп'ютерної мережі гірші за байєсівські мережі та наближення, які вони роблять. У першому прикладі, SCORE моделює спостереження та можливі причини у вигляді двостороннього графа. Потім вона використовує «бритву Оккама» для вибору найпростішого пояснення з набору можливих причин, використовуючи той самий метод підрахунку балів, що й Tomo. В іншому прикладі Шерлок створює детальну байєсівську мережу з виявлених залежностей . Потім він перевіряє кожну можливу першопричину, обчислюючи оцінку правдоподібності, припускаючи, що несправними є щонайбільше k компонентів. Це припущення є цілком обґрунтованим для $k \leq 3$ і зменшує складність для n компонент від $O(2^n)$ до $O(n^k)$. Нарешті, Gestalt - це гібридне рішення, яке класифікує і спирається на попередні роботи: воно оптимізує Шерлока, ігноруючи малоймовірні комбінації невдач, що дає кращі результати за частку повного часу дослідження.

Таблиця 2.2 - Огляд методів RCA, основаних на моделюванні залежностей

Method	Scale	Model	Exploration	Ranking
Shrink [94]	++	Bipartite	Complete (3 causes max)	Most probable
Tomo / NetDiagnoser [95]	++	Bipartite	Greedy while unexplained failures remain	Maximizes intersection with unexplained failures
Sherlock [59]	–	Multi level	Complete (3 causes max)	Most probable
NetMedic [62]	–	Multi level	N/A	Abnormality from known states
SCORE [96]	++	Bipartite	Greedy while unexplained failures remain	Prefers simpler explanations
Gestalt [97]	+	3 levels	Partial (3 causes max)	Most probable
Facebook [85]	++	Topology-based	N/A	Outlier detection
Kepler [90]	+ + +	BGP / IXP	N/A	Fraction of unreachable paths
007 [89]	++	Traceroute	Greedy with score threshold	Number of votes per link
SDNProbe [83]	++	Bipartite	Hopcroft-Karp	Hypothesis testing
Deepview [86]	++	Bipartite	Lasso Regression	Hypothesis testing
Zeno [84]	+	Provenance	N/A	Hypothesis testing

Представлені методи розроблені для різних масштабів (- мала мережа, + підприємство, ++ центр обробки даних/провайдер, + + + Інтернет) і пропонують кілька стратегій дослідження та ранжування. Відсортовано за датою.

2.3.2 RCA з обмеженими знаннями

Оператори мережі та кінцеві користувачі можуть не мати доступу до великої кількості інформації, необхідної для методів, які було описано в попередньому підрозділі. Однак вони можуть хотіти зрозуміти, чому комп'ютерна мережа «не працює» на їхньому боці, і чи можуть вони щось зробити, щоб вирішити проблему

самостійно. Перевірка швидкості Інтернету - це перший сервіс, до якого звертаються багато кінцевих користувачів, коли шукають першопричини: ці сервіси дозволяють швидко перевірити з'єднання з вимірювальними серверами, повертаючи середній час очікування і пропускну здатність. (Один лише Speedtest оголосив про понад 800 мільйонів тестів з 220 мільйонів унікальних пристроїв у 2020 році).

В роботі представлено два сімейства методів у цьому просторі, перше з яких базується на попередньо встановлених правилах, а друге - на краудсорсингу.

Діагностування з використанням попередньо встановлених правил - це типовий підхід до визначення причин збоїв. Інтуїтивно зрозуміло, що автоматизовані тести запускаються послідовно, і кожен тест пов'язаний з потенційною першопричиною. Один невдалий тест (або група невдалих тестів) може вказати кінцевому користувачеві на першопричину, що пояснює несправність. Тести слідують встановленим правилам: ці правила зазвичай визначаються експертами в галузі мережевих технологій, в деяких пропозиціях їм допомагають алгоритми машинного навчання.

Більшість інтернет-провайдерів надають методи самодіагностики або через спеціальний сервіс на своєму домашньому шлюзі, або у вигляді покрокової документації на своїх веб-сайтах. Французький провайдер Bouygues Telecom надає набір «Діагностичних» сторінок на веб-сервері свого шлюзу з актуальною інформацією про затримки трасування, ICMP, протоколу управління транспортом (TCP) та служби доменних імен (DNS). Однак ці сервіси найкраще підходять для технічно підкованих кінцевих користувачів, які знають, де їх знайти і як інтерпретувати результати; і націлені лише на часті проблеми і сценарії неправильного з'єднання в домашніх мережах. У літературі, Netalyzr, Fathom і FireLog запропонували діагностичні інструменти, які також використовують правила і базуються на розширеннях веб-браузерів. Веб-браузери можуть працювати на більшості типів комп'ютерів (включно з мобільними платформами), що дозволяє більшості кінцевих користувачів виконувати надійний RCA. Однак сучасні браузери мають технічні обмеження та обмеження безпеки, які

перешкоджають виконанню деяких поширених правил (наприклад, вимірювання низькорівневої статистики мережевих комунікацій).

Інший підхід до аналізу першопричин полягає в об'єднанні даних, зібраних з різних точок спостереження, і виявленні розбіжностей за допомогою статистичних методів. Це основна ідея веб-сайту DownDetector, який моніторить соціальні мережі («натовп») на предмет скарг на веб-сервіси. Коли за певний проміжок часу досягається певний поріг скарг, сервіс позначається як «деградований» або «непрацюючий». Одна з перших пропозицій, PeerPressure, дозволяє обмінюватися ключами конфігурації реєстру Windows між кінцевими користувачами. Потім вона оцінює ймовірність того, що кожен ключ конфігурації є першопричиною конкретної несправності, використовуючи просту байєсівську оцінку. Статистичне агрегування також може бути виконане для метрик комп'ютерних мереж, щоб виявити і точно визначити першопричини. У цьому випадку метрики часто обмінюються між кінцевими користувачами в тих самих географічних або топологічних регіонах за допомогою центрального або розподіленого сховища.

Розподілені хеш-таблиці (Distributed Hash Tables, DHT), популярні в однорангових комунікаціях, дозволяють кінцевим користувачам отримувати доступ до інформації, проіндексованої за допомогою «хешів» і розміщеної на хостингу інших кінцевих користувачів. Наприклад, DYSWIS публікує метрики на різних рівнях деталізації в DHT, від рівня домашньої мережі до рівня AS. Це дозволяє кінцевим користувачам порівнювати свої показники з показниками сусідів. Історичні дані також можуть бути використані, щоб отримати більше знань про конкретну несправність. Кінцеві користувачі можуть регулярно вимірювати відповідні метрики і будувати історичні часові ряди цих метрик. Припускається, що якщо розподіл часового ряду змінюється, то варто звернути увагу на відстежувані метрики для виявлення потенційних першопричин. Ця проблема формулюється як виявлення та локалізація точок змін і може бути вирішена статистично за допомогою CUSUM (CUmulative SUM control chart). Наприклад, FChains та Callegari та ін. використовують CUSUM для виявлення та локалізації збоїв у хмарних системах та веб-браузерах відповідно. У LOUD залежності між

часовими рядами виводяться за допомогою тесту причинності Грейнджера; автори продемонстрували хорошу локалізацію першопричин, знайшовши центральні вузли на виведеному графіку залежностей. Ці методи краудсорсингу дають грубіший RCA, ніж методи, описані в попередньому підрозділі, але їх можна легко масштабувати до тисяч і навіть мільйонів точок спостереження. Вони широко використовуються для порівняння роботи провайдерів та виявлення масштабних відключень, без потреби в центральних службах.

Однак вони мають низку важливих обмежень. Їх точність значною мірою залежить від даних, якими обмінюються однорангові пристрої (або які зберігаються в локальних часових рядах): якщо одноранговий пристрій неправильно ідентифікований в топології мережі, його дані можуть бути неправильно проаналізовані. Гірше того, однорангові користувачі можуть змінювати спільні дані, щоб зробити RCA марним, і навіть можуть шпигувати за важливою інформацією про усунення несправностей. Часові ряди вимагають регулярних вимірювань у часі і є чутливими до динамічності мережі - це особливо актуально для мобільних кінцевих користувачів.

Багато з представлених підходів покладаються на наявну базову істину для деяких спостережуваних збоїв: ці приклади дозволяють розробляти та оцінювати метод RCA економно. Це особливо актуально для контрольованого машинного навчання, як описано в наступному розділі. Хоча істинну інформацію легко отримати в більшості контрольованих середовищ (наприклад, в лабораторіях або мережах центрів обробки даних), отримати надійну інформацію в масштабах Інтернету набагато складніше: мережеві оператори часто приховують першопричини з комерційних міркувань або з міркувань безпеки. Публічні сторінки статусів, списки розсилки мережевих експертів і соціальні мережі іноді містять відповідну інформацію, але збір цих даних часто є ручним і трудомістким завданням.

2.4 Машинне навчання

У цьому розділі описано загальну методологію підходів машинного навчання разом з деякими практичними прикладами.

2.4.1 Методологія машинного навчання

Машинне навчання позначає широкий спектр методів у ще ширшій галузі штучного інтелекту (ШІ). На високому рівні алгоритм машинного навчання може бути використаний для побудови прогностичних моделей, пов'язаних з деяким набором даних. Метою будь-якого такого алгоритму є автоматичне покращення побудованих моделей відповідно до певного показника ефективності. Після початкової фази навчання (рис. 2.2) модель можна використовувати для вирішення завдання на основі нових даних (це фаза виведення).

Алгоритми машинного навчання потребують даних як під час навчання, так і під час виведення. Дані зберігаються у вигляді зразків у наборі даних, який зазвичай поділяється на наступні набори (рисунок 2.2):

- навчальний набір: зазвичай найбільший, він використовується на етапі навчання. На практиці кожен зразок з цього набору зчитується і використовується алгоритмом послідовно або кілька разів;

- перевірочний набір: цей набір використовується для підтвердження того, що навчена модель здатна добре працювати на невідомих даних. Можна вибрати гіперпараметри моделі таким чином, щоб вона показала найкращі результати на перевірочному наборі. Якщо модель добре працює на навчальній множині, але погано на перевірочній, це означає, що модель перевантажена: вона спеціалізується на навчальній множині і не здатна узагальнювати більшу кількість даних;

- тестова множина (також називається оціночною множиною): після вибору найкращих гіперпараметрів - завдяки валідаційній множині - і відповідного навчання моделі, можна оцінити модель, використовуючи тестову множину. Оскільки цей набір не використовувався під час розробки та навчання моделі, він повинен найкраще відображати поведінку моделі на «свіжих» даних. У задачах машинного навчання цей набір не розкривається учасникам, щоб можна було об'єктивно порівняти моделі.

Поширеним припущенням є те, що всі набори мають однаковий розподіл ймовірностей, але на практиці цього не можна гарантувати: нові дані можуть зміщуватися до значень, не досліджених під час навчання. На щастя, сучасні алгоритми здатні послабити це припущення: цю можливість використано у Розділі 3, де модель машинного навчання навчається на двох топологіях мережі, а потім перевіряється і тестується на двох інших, ненавчених мережевих топологіях. В літературі існує велика плутанина між валідацією та тестовими наборами, ймовірно, тому, що тестові дані можуть легко стати валідаційними, якщо їх використовувати під час побудови моделі.

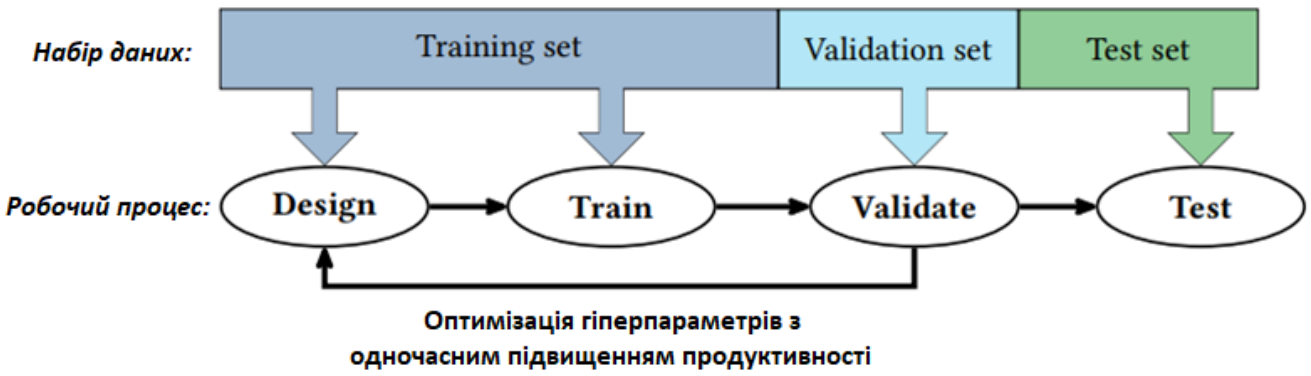


Рисунок 2.2 - Загальний вигляд ітеративного процесу машинного навчання. Кожен крок повинен використовувати правильний набір вхідних даних, щоб уникнути перебору

2.4.2 Визначення продуктивності моделі

Вибір конкретної метрики продуктивності для алгоритму машинного навчання визначається завданням, яке має вирішувати навчена модель. Розрізняють два основні класи завдань:

- контрольоване;

- неконтрольоване навчання.

У контрольованому навчанні кожен зразок у наборі даних асоціюється з міткою. Ця мітка представляє очікуваний результат, який модель повинна видати на вході зразка (також званий ознаками).

Продуктивність можна визначити як середню похибку між очікуваною міткою (з набору даних) і прогнозованою (з результатів роботи моделі):

- у задачах регресії міткою є числове значення: завдання полягає в тому, щоб передбачити це значення за деякими вхідними даними (наприклад, «передбачити затримку в каналі зв'язку за заданими характеристиками мережі»). Поширеними метриками помилок є середня абсолютна помилка (MAE) і середня квадратична помилка (MSE);

- у задачах класифікації мітки - це категоріальні класи, які модель повинна передбачити для кожної вибірки (наприклад, «передбачити, чи є даний зв'язок номінальним (клас 1) або несправним (клас 2) за заданими характеристиками мережі»). У цьому випадку похибка продуктивності часто вимірюється перехресною ентропією, метрикою пригадування або точністю.

Мітки можуть бути недоступні. Це може бути пов'язано з тим, що маркування кожного зразка є надто дорогим, або з тим, що деяка важлива інформація не може бути зібрана. Ця проблема вирішується за допомогою неконтрольованих підходів, які намагаються виявити закономірності в навчальних даних.

Кластеризація даних і виявлення аномалій є двома прикладами неконтрольованих завдань. За відсутності мітки важко стверджувати про продуктивність неконтрольованих алгоритмів. У більшості випадків невелика

кількість зразків маркується у валідаційних і тестових наборах, що дає змогу проводити емпіричну оцінку. Часто це ручний процес, що передбачає трудомісткий збір інформації та перехресні посилання, як підкреслено в підрозділі 2.3.2 для RCA в комп'ютерних мережах. Нижче наведено кілька прикладів алгоритмів машинного навчання, які були успішно застосовані до комп'ютерних мереж в останніх дослідженнях.

2.4.3 Наївні класифікатори Байєса

Наївний Байєс є одним з найпростіших сімейств класифікаторів, заснованих на теоремі Байєса. Кожна вибірка в наборі даних асоціюється з ознаками $x = (x_i)_{i\in\{1,\dots,n\}}$ та класом C_k серед K класів. Для кожного класу k ми хочемо оцінити умовну ймовірність того, що вибірка належить до класу k, враховуючи її ознаки x. Використовуючи теорему Байєса:

$$P(C_k|x) = \frac{P(C_k)\cdot P(x|C_k)}{P_{(x)}} \tag{2.2}$$

Тепер, що стосується наївної частини, ми припускаємо, що ознаки x_i є взаємно незалежними. (Це припущення часто не перевіряється, але наївні байєсівські класифікатори залишаються напрочуд стійкими на практиці). Порівнюючи класи, ми ігноруємо знаменник у рівнянні 2.2, оскільки він не залежить від C_k. Тепер ми можемо оцінити найбільш ймовірний клас $\tilde{k}$ такий як:

$$\tilde{k} = \underset{k\in\{1,\dots,K\}}{\operatorname{argmax}} P(C_k)\prod_{i=1}^{n} P(x_i|C_k) \tag{2.3}$$

Попередню ймовірність для класу C_k можна емпірично оцінити як відношення вибірок, що мають цей клас у навчальних даних. Розподіл ймовірностей ознак зазвичай моделюється за допомогою нормального розподілу, але можна побудувати більш виразні моделі за допомогою оцінки щільності ядра (KDE). Маючи набір можливих значень ознак для класу C_k, KDE використовує зважену суму ядер для оцінки розподілу спостережуваних значень.

Ядра - це невід'ємні функції, які локально оцінюють розподіл ймовірностей; на практиці часто використовують стандартну функцію нормальної щільності. Наївні байєсівські класифікатори зазвичай потребують невеликої кількості навіть у KDE; це означає, що їх можна навчати і зберігати дуже ефективно.

У комп'ютерних мережах їх використовували для виявлення несправностей і навіть для мережевої томографії за допомогою beCAUSe.

2.4.4 Навчання на основі дерев рішень

Для побудови моделей дерев рішень було запропоновано багато алгоритмів машинного навчання. Ці моделі складаються з послідовності тестів («гілок»), які ведуть до остаточних прогнозів («листя»). Модель дерева рішень можна застосувати до вибірки, слідуючи гілками відповідно до результатів тестів: на кожній гілці перевіряється значення однієї ознаки вибірки. Наприклад, один тест A може перевіряти, чи є значення ознаки меншим за 10; якщо це так, то наступним тестом буде B, інакше - C. У нижній частині дерева листки містять клас або числове значення для задач класифікації або регресії, відповідно.

Основною перевагою дерев рішень є їхня інтерпретованість: за тестами, зробленими в гілках, легко зрозуміти причину, що лежить в основі того чи іншого прогнозу. Оскільки побудова оптимального дерева рішень, як відомо, є NP-повною, більшість алгоритмів навчання рекурсивно знаходять найкращий тест або найкраще розбиття: на кожній гілці алгоритм оцінює, який тест був би найкращим для розбиття навчального набору даних між різними прогнозами.

Якість тесту оцінюється за допомогою функції втрат на створених дочірніх гілках: для кожного класу C_k на дочірній гілці i, нехай $P_i(C_k)$ є часткою зразків, що мають клас C_k як мітку. Втрати в i можна оцінити за двома популярними формулами:

$$H_i = 1 - \sum_{C_k} P_i(C_k)^2$$

$$H_i = -\sum_{C_k} P_i(C_k) \log(P_i(C_k)) \quad \text{(ентропія)}$$

Аналогічно можна використовувати MAE та MSE як функції втрат для дерев регресії. Можна розбивати набір даних до тих пір, поки всі листки не будуть містити лише один клас (або одне значення); однак це може призвести до дуже великих дерев і серйозних перекриттів. Більшість алгоритмів пропонують критерії зупинки, такі як «мінімальна кількість навчальних вибірок на листок» або «максимальна глибина дерева», щоб уникнути цієї проблеми. Серед алгоритмів навчання, C4.5 та CART широко використовуються в літературі про комп'ютерні мережі. Наприклад, Ebay та Microsoft використовують дерева рішень для пошуку збоїв у своїх центрах обробки даних; проблеми мобільного потокового передавання діагностуються за допомогою C4.5; NetPrints пропонує усунення несправностей конгруентності за допомогою дерев, що інтерпретуються, а алгоритми контролю перевантажень TCP автоматично виводяться за допомогою дерев рішень.

2.4.5 Ансамблеве навчання

Для покращення прогнозів машинного навчання можна об'єднати декілька моделей. Для задач класифікації можна взяти клас, передбачений більшістю моделей, тоді як у задачах регресії можна використовувати середнє передбачення. Мабуть, найпопулярнішим прикладом ансамблевого навчання є підхід випадкових лісів.

У випадкових лісах багато простих дерев рішень навчаються на випадкових підмножинах навчального набору даних; під час виведення на виході обирається більшість прогнозів серед усіх маленьких дерев. Цей процес називається «пакуванням» (від англ. «bootstrap aggregating») і може бути застосований до інших сімейств моделей.

«Бустінг» є альтернативним ансамблевим методом: нові моделі послідовно навчаються на вибірках з невірними прогнозами попередніх моделей. Якщо функція ефективності є диференційованою, градієнтний бустінг можна використовувати для оптимізації побудови нових моделей.

Нарешті, «Stacking» використовує «мета-модель», навчену об'єднувати результати інших моделей у кінцевий прогноз. Ці прості підходи допомагають зменшити кількість помилок і напрочуд добре працюють на практиці: ансамблеві методи зазвичай демонструють найкращі результати в задачах машинного навчання.

Ця тенденція була підтверджена в багатьох недавніх проблемах комп'ютерних мереж, включаючи виявлення та локалізацію несправностей. Емпірично, вихід ансамблю повинен бути принаймні настільки ж ефективним, як і вихід найкращої моделі для даної вибірки: помилки окремих моделей в середньому компенсуються.

Одним з основних недоліків є те, що важче зрозуміти прогноз ансамблевої моделі (порівняно з деревами рішень).

2.4.6 Артикуляційні нейронні мережі

Однією з перших моделей взаємодії нейронів мозку є перцептрон Франка Розенблата. Перцептрон можна описати як бінарний лінійний класифікатор: він ітеративно намагається визначити параметри гіперплощини, що дозволяє розділити два класи. Для n вхідних ознак ця гіперплощина визначається вагами $w \in R^n$ та зміщенням $b \in R$ таким чином, що передбачуваний клас $\widetilde{y_i}$ вибірки з ознаками x_i має вигляд:

$$\widetilde{y_i} = \begin{cases} 1 & \text{якщо } w \cdot x_i + b > 0, \\ 0 & \text{інакше} \end{cases} \tag{2.4}$$

Ваги ітеративно оновлюються для кожної вибірки і в навчальному наборі даних, як показано в рівнянні 2.5. Важливим параметром процесу навчання є швидкість навчання η ∈]0, 1]: велика швидкість навчання прискорює навчання, але робить ваги менш стабільними.

$$\forall j \in \{1, \dots, n\}: w_j' = w_j + \eta(y_i - \widehat{y_i})x_{i,j} \quad \text{та } b' = b + \eta(y_i - \widehat{y_i}) \qquad (2.5)$$

Доведено, що персептрони збігаються, якщо навчальна множина є лінійно роздільною. Однак більшість наборів даних не є такими: це призвело до розробки більш складних моделей, таких як машини опорних векторів (SVM) і багатошарові персептрони (MLP), останні більш відомі як «повністю з'єднані нейронні мережі».

У MLP використовується декілька персептронів з нелінійними функціями активації для створення послідовності шарів будь-якої розмірності. Перший шар відповідає ознакам зразка, тоді як останній шар містить вихідні дані моделі. Проміжні шари часто називають «прихованими шарами» і вони можуть бути будь-якого розміру - кількість шарів та їхні розміри стають основними гіперпараметрами MLP. Функції активації для прихованих шарів можуть бути такими ж простими, як і для прямокутної лінійної одиниці (ReLU): f (x) = max (0, x). На відміну від цього, оскільки останній шар безпосередньо відображається на вихід(и) моделі, його функція активації залежить від задачі: для задач класифікації з K класами часто використовується функція softmax для нормалізації K виходів до розподілу ймовірностей.

$$\sigma: \mathbb{R}^K \mapsto \mathbb{R}^K: \sigma(x)_i = \frac{e^{x_i}}{\sum_{j=0}^{K} e^{x_j}} \qquad \text{(активація Softmax)}$$

Перцептрон передає інформацію з шару L_t розміром u до шару L_{t+1} розміром v за допомогою матриці ваг («параметрів») $w^t \epsilon \mathbb{R}^{u \times u}$. Порівняно з рівнянням 2.4, замість точкового добутку між входами персептрона та w^t використовується множення матриці (рисунок 2.3 для ілюстрації нейронної мережі з одним прихованим шаром). Процес навчання схожий, хоча і з набагато більшою кількістю ваг для навчання: для кожної навчальної вибірки алгоритм повинен обчислити градієнт кожного значення ваги по відношенню до функції втрат задачі.

Метод зворотного поширення оптимізує цей процес: він починається з обчислення градієнтів останнього персептрона а потім за ланцюговим правилом повертається до першого. Потім ваги оновлюються відповідно до їх градієнтів та обраної швидкості навчання η - процес, відомий як «Стохастичний градієнтний спуск». (Використання повного набору даних у кожній ітерації є недоцільним, тому замість цього вибираються випадкові вибірки, що робить процес «стохастичним»). Інші методи оптимізації та динамічні планувальники швидкості навчання були запропоновані для прискорення процесу навчання та уникнення «локальних мінімумів втрат». Спочатку розроблені для роботи на дорогому спеціальному обладнанні з використанням потенціометрів як ваг і двигунів для оновлення, артикулярні нейронні мережі тепер можуть працювати на товарних процесорах (CPU) і графічних процесорах (GPU). Зараз ми розглянемо кілька варіантів MLP та їх застосування.

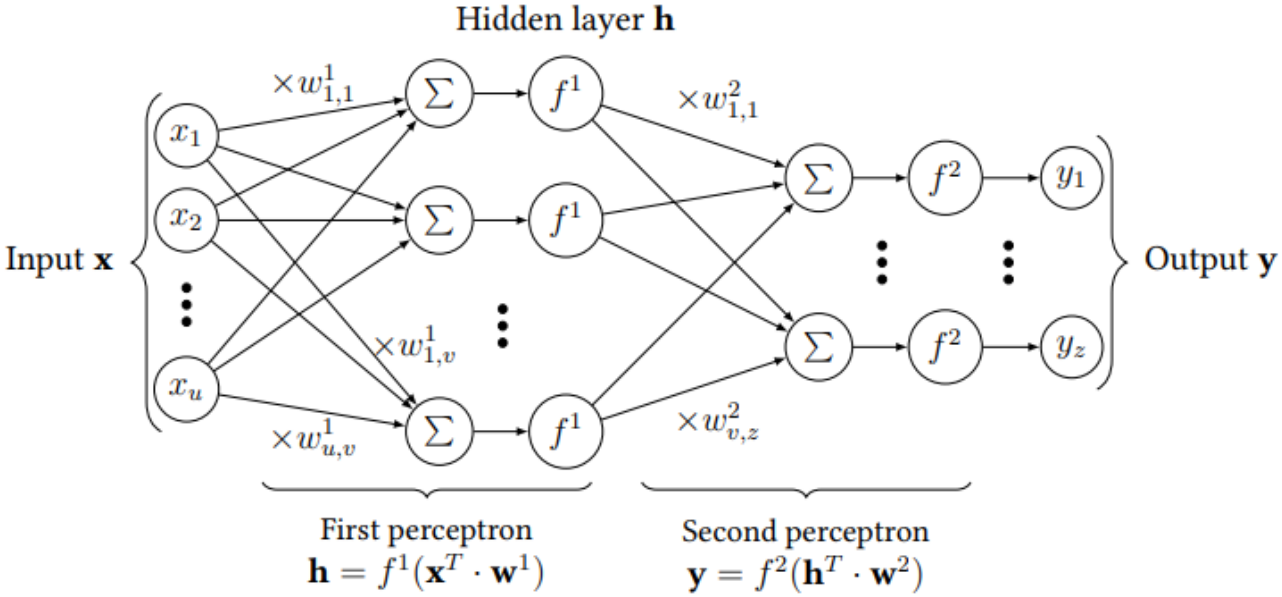


Рисунок 2.3 - Простий MLP з одним прихованим шаром. Значення поширюються через нейронну мережу за допомогою матричних множень між входами (x та h) та вагами (w^k). Функції активації позначаються f^k

З ростом доступних обчислювальних потужностей стало можливим навчати нейронні мережі на даних високої розмірності. Зображення є прикладом таких

даних: легко зіставити зображення розміром n × m пікселів з c колірними каналами з вхідним вектором розміром n × m × c.

Згорткові нейронні мережі (CNN) спочатку були запропоновані для розпізнавання написаних чисел на невеликих зображеннях: за допомогою декількох згорткових шарів CNN можуть розпізнавати закономірності на вхідному зображенні. Один згортковий шар працює шляхом згортки невеликих літер (або ядер) у двовимірному просторі зображення. Основна перевага цього підходу полягає в тому, що він вимагає набагато менше параметрів, ніж повністю з'єднані шари, що зменшує перекриття. Об'єднання шарів також може додатково зменшити кількість параметрів у прихованих шарах шляхом «злиття сусідніх пікселів» за допомогою наприклад, середнє або максимальне значення. Оскільки одні й ті самі літери застосовуються кілька разів для покриття зображення, ШНМ є більш стійкими до перекладів і краще виявляють закономірності.

На рисунку 2.4 показано приклад імітаційного ШНМ із вхідними даними (ліворуч) у вигляді матриці 4×4. На практиці для зображень використовується двовимірний вхід; третій вимір також можна використовувати для представлення кольорів (наприклад, для червоного, зеленого та синього каналів). Під час згортки 2 × 2 ми послідовно обчислюємо добутки точок між 2 × 2 «вікнами» над вхідними даними та параметризованим ядром $K = \begin{pmatrix} 1 & 0 \\ 1 & -1 \end{pmatrix}$, яке було вивчено на етапі навчання. Як показано на рисунку, лівий верхній вихід обчислюється за допомогою верхнього лівого вікна на вході (рівняння 2.6). Потім обчислюється верхній центральний вихід (рівняння 2.7) шляхом «зсуву» вікна на одну клітинку вправо: це згортка з вікнами, що перекриваються, що призводить до виходу розміром 3 × 3.

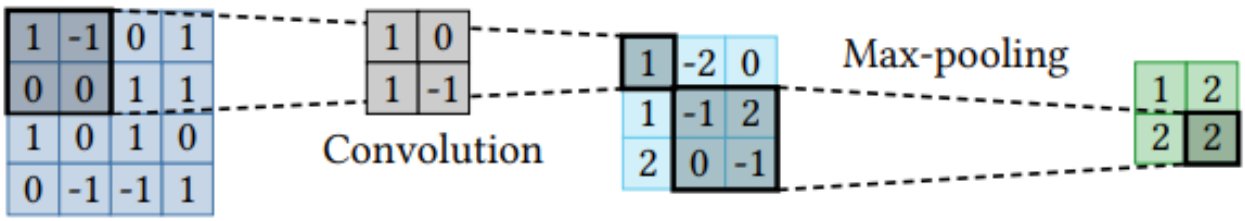


Рисунок 2.4 - Приклад згортки 2 × 2 з подальшим максимальним об'єднанням 2 × 2. Ядра згортки та об'єднання «ковзають» по входах для побудови виходів зліва направо, зверху вниз

$$vec\left(\begin{pmatrix}1 & -1\\ 0 & 0\end{pmatrix}\right)\cdot vec\,(K) = 1\ \times 1 + (-1)\times 0 + 0\times 1 + 0\times(-1) = 1 \quad (2.6)$$

$$vec\left(\begin{pmatrix}-1 & 0\\ 0 & 1\end{pmatrix}\right)\cdot vec\,(K) = (-1)\times 1 + 0\times 0 + 0\times 1 + 1\times(-1) = -2 \quad (2.7)$$

Об'єднання операцій допомагає зменшити кількість вимірів у CNN: на рисунку 2.4 застосовується операція 2×2 «Max-pooling», використовуючи аналогічне ковзне вікно. Наприклад, вихідні дані внизу праворуч обчислюються за максимальним значенням у правому нижньому вікні: $\left(\begin{pmatrix}-1 & 2\\ 0 & -1\end{pmatrix}\right)= 2$.

У той час як перші шари згортки виявляють прості патерни (наприклад, горизонтальні або вертикальні лінії), глибші шари активуються дедалі складнішими патернами (колами або навіть людськими обличчями). Варто зазначити, що найефективніші ШНМ є надзвичайно глибокими: модель ResNet перемогла в конкурсі ILSVRC2015 з розпізнавання зображень, маючи понад 150 прихованих шарів.

У задачах класифікації приховані шари зазвичай є просто повністю зв'язними шарами, навченими на конкретному наборі даних. Це призводить до цікавої властивості: зазвичай можна змінити класи, які може передбачити модель, лише перенавчивши останні шари мережі. Інтуїтивно зрозуміло, що згорткові шари виділяють релевантні ознаки з вхідних даних, тоді як повнозв'язні шари спеціалізуються на прогнозуванні певного класу.

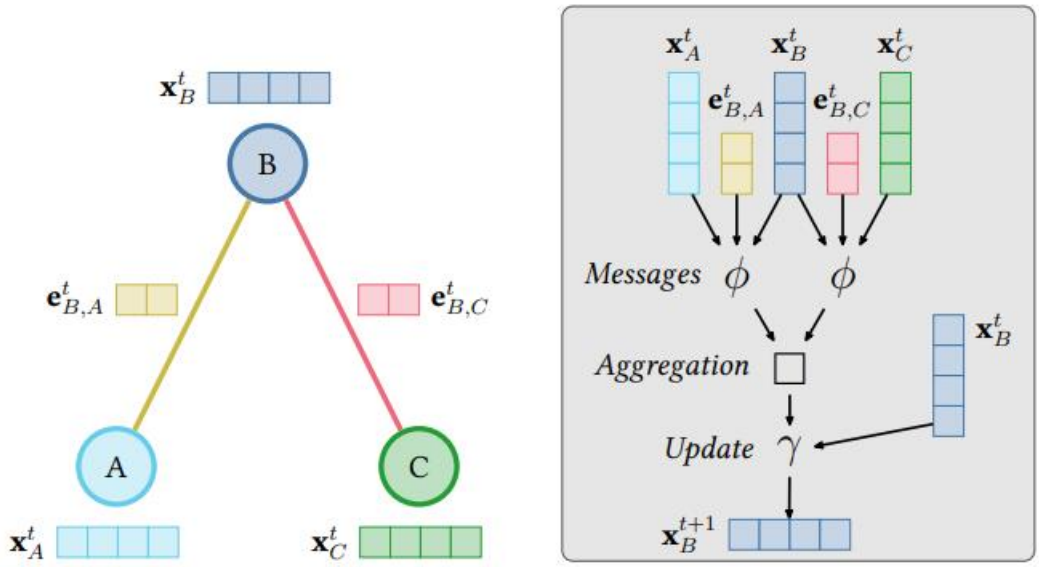


Рисунок 2.5 - Зліва: простий граф з відповідними вкладеннями. Праворуч: процедура оновлення представлення B з урахуванням його сусідів A та C, використовуючи передачу повідомлень GNN (рівняння 2.8)

Рекурентні нейронні мережі (RNN) найкраще підходять, коли вхідні дані моделі складаються з послідовності значень. Це, наприклад, стосується часових рядів, аудіозаписів або текстових речень. Порівняно з попередніми архітектурами, рекурентні шари підтримують внутрішній стан; це важливо для запам'ятовування контексту між різними значеннями у вхідній послідовності. Довга короткочасна пам'ять (LSTM) є найпоширенішим рекурентним шаром. Він успішно застосовується в автоматизованому перекладі, розпізнаванні та синтезі мови, прогнозуванні часових рядів, виявленні аномалій тощо. Поєднуючи найкраще з обох світів, шари ConvLSTM додають згортки до ШНМ, покращуючи підтримку більш складних вхідних даних, таких як послідовності радіолокаційних зондів або фільмів.

Нещодавно з'явилися графові нейронні мережі (GNN), які повністю використовують топологію вхідних даних у вигляді графів. Це сімейство особливо цікаве для моделювання фізичних мереж (міських доріг, електромереж, молекул тощо), а також мереж залежностей (наприклад, соціальних графів). Звичайно,

комп'ютерні мережі можна легко відобразити на граф: GNN використовувалися для оптимізації маршрутизації, виявлення крайових аномалій та моделювання SDN.

У багатьох популярних GNN для поширення інформаційного сигналу між вершинами використовується передача повідомлень. На кожному рівні t кожна вершина графа GNN асоціюється з прихованими станами $x_i^t \epsilon \mathbb{R}^u$ (які також ϕ вкладеннями $e_{i,j}^t \epsilon \mathbb{R}^v$. Для кожного ребра та вершини обчислюються вкладення для наступного шару, використовуючи попередні вкладення та вкладення сусідів.

На рисунку 2.5 показано, як можна оновити вкладення графа з 3-ма вершинами. Для кожного сусіда цільової вершини (тут A і C) ми обчислюємо повідомлення за допомогою . Потім ми агрегуємо повідомлення □ з і оновлюємо представлення B за допомогою γ. У більш загальному випадку, вкладення оновлюються за допомогою рівняння 2.8, де

- ϕ: $\mathbb{R}^{u \times u \times v} \mapsto \mathbb{R}^w$ - функція повідомлення;

- γ: $\mathbb{R}^{u \times w} \mapsto \mathbb{R}^u$- функція оновлення вершини;

- □ функція агрегації, яка може приймати будь-яку кількість аргументів (наприклад, сума, середнє);

- $N(i)$ повертає набір індексів сусідів вершини i;

- γ і ϕ - зазвичай (прості) нейронні мережі.

$$x_i^{t+1} = \gamma\left(x_i^t, \square_{j \epsilon N(i)} \phi\left(x_i^t, x_j^t, e_{i,j}^t\right)\right) \quad (2.8)$$

Цей метод передачі повідомлень може бути адаптований до будь-якої топології графа завдяки оператору агрегування. (Зауважимо, що використовувана топологія GNN є похідною від топології досліджуваної мережі, але не обов'язково є такою ж). Навчені ваги заперечуються тільки в γ і ϕ: будь-який існуючий алгоритм навчання може бути використаний для оптимізації цих ваг. Знову ж таки, повністю з'єднані шари можуть бути використані для зчитування інформації з побудованих вкладок і виведення запитуваного прогнозу на рівні ребер, вершин або графа.

На цьому етапі було досліджено моделювання комп'ютерних мереж, проведено аналіз першопричин, а також проблем і методів машинного навчання.

Існує багато пропозицій щодо моделювання комп'ютерних мереж, які було класифікувано в залежності від кількості доступної інформації.

У цій роботі зазначено, що статистичне (машинне) навчання може бути дуже корисним у моделюванні великих комп'ютерних мереж зі складними взаємозалежностями. На підтвердження цього підходу запропоновано три розділи, кожен з яких розглядає проблему моделювання мереж з різних точок зору.

У розділі 3 проведено того, як GNN можна використовувати для оцінки продуктивності мережевих шляхів. На сьогоднішній день імітаційне моделювання та емуляція є найбільш практичними підходами для оцінки продуктивності мережі, знаючи характеристики її компонентів. Однак ці методи є дорогими і не можуть масштабуватися до великих мереж. На противагу цьому, це дослідження дозволяє точно моделювати мережі з десятками вузлів всього за кілька мілісекунд.

Було використано кілька моделей машинного навчання (наївний Байєс і класифікатор випадкових лісів, а також оригінальну модель CNN: DiagNet) на цьому наборі даних, намагаючись змоделювати великомасштабну комп'ютерну мережу з точки зору кінцевих користувачів. Було застосовано ці моделі для RCA: завдяки краудсорсингу активних вимірювань від багатьох кінцевих користувачів з різних точок спостереження, вони забезпечують автоматизований висновок залежностей і діагностику без необхідності співпраці з провайдерами. У порівнянні зі звичайною томографією мережі, представлені рішення не потребують точної інформації про топологію мережі: вони використовують різноманітність точок спостереження і виводять комбінації розташування і сімейства несправностей в якості ймовірних першопричин. Було розробленро рішення так, щоб їх можна було розширювати, і показати перевагу DiagNet CNN в його здатності приймати нові функції, невидимі під час навчання.

РОЗДІЛ 3 КОМП'ЮТЕРНЕ МЕРЕЖЕВЕ МОДЕЛЮВАННЯ ЗА ДОПОМОГОЮ ГРАФОВИХ НЕЙРОННИХ МЕРЕЖ

Мережеві оператори проектують комп'ютерні мережі, щоб забезпечити якісний зв'язок для своїх клієнтів. Таким чином, їм потрібні надійні методи прогнозування очікуваної продуктивності ще до розгортання фізичних каналів і пристроїв мережі або модифікації конфігурацій програмно-розширених мереж (SDN).

Існує два сімейства підходів до моделювання комп'ютерної мережі. Емуляція мережі намагається відтворити модельовану топологію мережі на існуючій мережі. За наявності достатніх апаратних ресурсів емуляція може бути дуже корисною для оцінки того, як змодельована мережа буде працювати на практиці. Методи емуляції є більш гнучкими, дозволяючи моделювати мережі з довільними характеристиками; це дуже зручно, коли платформа емуляції недостатньо велика. Однак, враховуючи складність взаємодії компонентів мережі, інструменти імітації часто вимагають значних ресурсів або вимагають значного спрощення мережевої моделі. Це є проблематичним у таких випадках контексті великих динамічних мереж, де трафік постійно змінюється з часом: оператори повинні переглядати і адаптувати топологію мережі відповідно до цих змін, прагнучи при цьому мінімізувати вплив на мережу. Проектувальники мереж також повинні мати можливість тестувати альтернативні конфігурації мережі за короткий час. Таким чином, потрібен ефективний інструмент мережевого моделювання, щоб швидко оцінити, як оновлена топологія мережі буде працювати з реальними характеристиками мережі.

У цій кваліфікаційній магістерській роботі було досліджено, як можна передбачити детальну продуктивність великої мережевої моделі, не покладаючись на дорогі симуляції, а саме, як графові нейронні мережі (GNN) можуть бути навчені робити прогнози продуктивності за частку часу, необхідного для імітаційного моделювання. В роботі надано опис раннього рішення GNN, RouteNet, пояснено,

як було розширено початковий підхід RouteNet, щоб відобразити всі компоненти комп'ютерної мережі в GNN та представлено результати дослідження.

Застосування першого підходу (A1) призвело до середньої відносної похибки лише 1,53% для наданого набору оціночних даних. Використовуючи другий підхід (A2), було отримали нижчу відносну похибку лише 1,15%. У роботі також наведено дослідження абляції та додаткові візуалізації вивчених репрезентацій, намагаючись повністю охарактеризувати результати дослідження.

3.1 Особливості застосування графових нейронних мереж (GNN)

Графові нейронні мережі є особливо перспективними для моделювання комп'ютерних мереж. Це нещодавнє сімейство моделей машинного навчання добре підходить для відображення складних взаємодій між компонентами мережі. Насправді, воно успішно використовується в широкому спектрі додатків, від аналізу невідомих хімічних сполук до прогнозування траєкторії руху в дорожніх мережах. Однією з важливих особливостей GNN є те, що вони можуть бути адаптовані до будь-якої топології мережі після початкової фази навчання.

В цій роботі основна увага приділялась вирішенню завдання, яке полягало в тому, щоб створити модель GNN, яка, знаючи ряд характеристик мережі (тобто топологію, пропускну здатність каналів, тип обслуговування (ToS) і трасування, а також політику черги/планування вузлів), могла б прогнозувати продуктивність каналу (рис. 3.1). Зокрема, побудувати модель, яка прогнозує середню затримку на кожному шляху, враховуючи затримки в чергах на проміжних вузлах.

Для вирішення поставленого завдання було сформульовано декілька правил: рішення повинні базуватися на артикуляційних нейронних мережах, навчених «з нуля» без використання методів мережевого моделювання.

Було використано тестовий набір даних з відсутніми метриками ефективності шляху. Цей набір даних базується на невідомій топології, яка відсутня у наданих

навчальних та валідаційних наборах даних. Такий вибір дозволяє оцінити, як рішення узагальнюють різні топології без перенавчання, що є критично важливою властивістю для використання рішень на практиці. Використовуючи дані, свої кожна команда повинна була представити відсутні затримки на шляху на спеціальному сайті; потім сайт обчислював середню абсолютну відсоткову похибку (MAPE) за секретними, очікуваними показниками (рівняння 3.1). (Слід зауважити, що ця функція оцінки є дійсною, оскільки очікувані затримки завжди позитивні).

MAPE = 1 n Xn i=1 expectedi - predictedi expectedi (3.1)

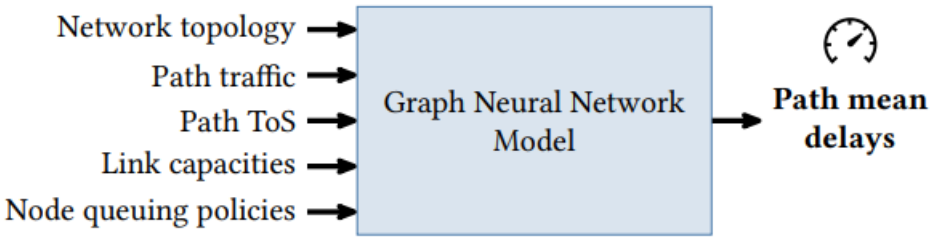


Рисунок 3.1 - Ілюстрація завдання виклику GNN: передбачити затримки для мережевих шляхів із заданої конфігурації

3.2 Базова лінія RouteNet

RouteNet - це мережа передачі повідомлень GNN, яка підтримує два типи вбудовувань: для шляхів і для з'єднань. Знаючи характеристики мережі, RouteNet може передбачити середні затримки на шляху, джиттер і втрати - важливі метрики для оцінки продуктивності мережі - що робить його хорошою базовою лінією.

Топологія комп'ютерної мережі відображається на GNN наступним чином. Спочатку створюється одна вершина типу «link» для кожного шляху в топології мережі. Потім для кожного шляху додається одна вершина типу «шлях», яка зв'язується з усіма вершинами типу «link», що є частиною шляху. При такому

підході результуюча GNN є, по суті, двостороннім графом, що підкреслює кругові залежності між з’єднаннями і шляхами.

Вбудування ініціалізуються відповідними характеристиками шляхів і з’єднань. Потім відбувається ітеративний обмін повідомленнями для оновлення цих вбудовувань: спочатку, вбудовування шляху оновлюється з впорядкованих вбудовувань його з’єднань-членів. (Внутрішньо, ця функція першого оновлення використовує RNN для врахування впорядкованості шляхів). Потім RouteNet робить протилежне: кожне вбудовування з’єднання оновлюється шляхом агрегування вбудовувань всіх шляхів, які використовують ці з’єднання. Для відображення зростаючого навантаження на з’єднання, це об'єднання виконується за допомогою функції глобальної суми, а потім за допомогою простого персептрона. Ці два циклічні оновлення виконуються $T = 8$ разів для кожної вибірки, як під час навчання, так і під час виведення.

Враховуючи складні взаємозалежності між шляхами та з’єднаннями, цей метод імітує наближену ітерацію з нульовою точкою. GNN, що передають повідомлення, відрізняються від типових шарів нейронних мереж, але підхід насправді схожий: послідовні нелінійні операції призводять до нових прихованих станів для шляхів і з’єднань. Приховані стани шляхів потім обробляються в стандартному MLP для зчитування очікуваних показників продуктивності (наприклад, середньої затримки на шляху).

RouteNet підтримує довільну топологію мережі і може добре узагальнюватися на інші топології.

В роботі запропоновано два різних підходи до вивчення представлень, зіставлених з вузлами мережі. У першому підході (A1) було додано спеціальні вбудовування для вузлів. Ці нові вбудовування сприяють оновленню вбудовувань з’єднань та шляхів у покращеній версії базового алгоритму.

Другий підхід (A2) вбудовує об'єкти вузлів у об'єкти з’єднань. Незважаючи на те, що він простіший, цей альтернативний підхід ще більше зменшує середні помилки. Цей розділ починається з того, що є спільного між цими двома підходами,

пояснюючи, як моделюються і поширюються взаємозалежності між шляхами, з’єднаннями і вузлами.

3.3 Передача повідомлень на двосторонніх графах для моделювання залежностей

RouteNet моделює залежності між з’єднаннями та шляхами за допомогою двосторонніх графів. Спрямовані дводольні графи, що з'єднують типи *A* і *B*, позначаються $G_{A,B}$; як приклад, граф, що моделює залежності між шляхами і вузлами в RouteNet, має вигляд $G_{p,l}$. Оскільки ми хочемо додати вбудовування для вузлів мережі (підхід A1), нам потрібні додаткові графи залежностей для моделювання взаємозалежностей «вузол-шлях» та «вузол-з’єднання». Тому ми побудуємо Gp,l, $G_{l,n}$ і $G_{p,n,}$ три двосторонні графи наступним чином:

- $G_{p,l}$ з'єднує кожен шлях зі з’єднаннями, які він використовує, це, по суті, дводольний граф, запропонований в RouteNet;

- Аналогічно, $G_{p,n}$ з'єднує кожен шлях з вузлами, через які він проходить;

- Нарешті, $G_{l,n}$ з'єднує кожне з’єднання з його початковим і кінцевим вузлами.

Для ілюстрації показано топологію імітаційної комп'ютернгої мережі на рисунку 3.2 разом з відповідними трьома дводольними графами. Для наочності, ця топологія мережі містить лише три вузли (A, B і C), чотири з’єднання (від 1 до 4) і шість шляхів (AB, AC і тощо). Маршрутизація у цій топології є тривіальною, оскільки всі шляхи повинні проходити через вузол A. Наприклад, шлях BC складається з шляхів 2 і 3: $G_{p,l}$ має ребро між вершинами «BC» і «2», а також ребро між «BC» і «3». З’єднання 2 починається у вершині B і закінчується у вершині A, отже $G_{l,n}$ з'єднує «2» з «B», а також «2» з «A». На завершення ілюстрації слід зауважити, що повідомлення на шляху BC маршрутизуються як B → A → C. Тому в $G_{p,n}$ додаються три відповідні ребра: (BC, A); (BC, B) і (BC, C).

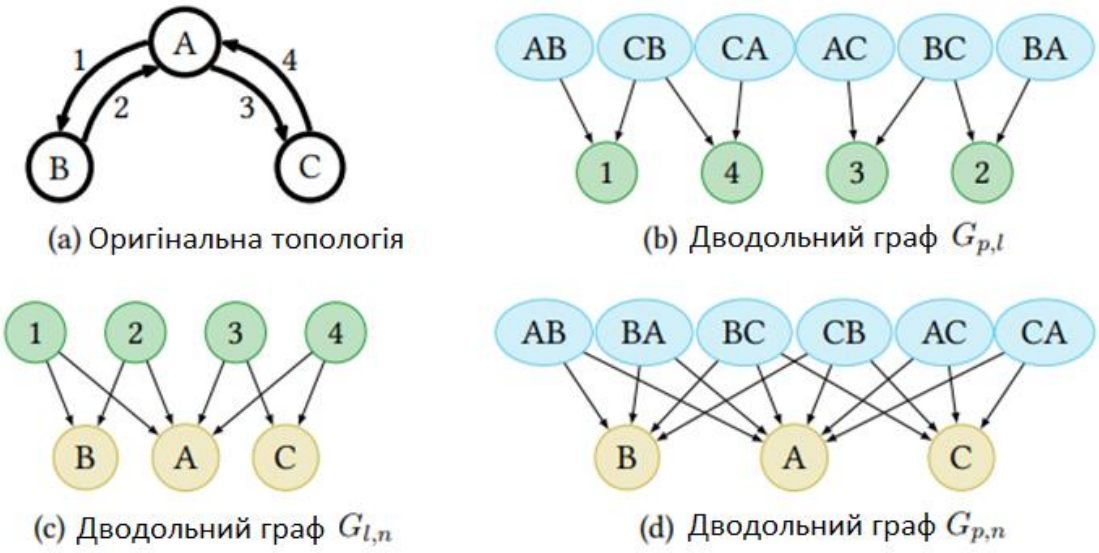


Рисунок 3.2 - Приклад базової топології, що веде до трьох представлених дводольних графів. Вузли позначаються однією літерою, з'єднання - числом, шляхи - двома літерами (XY означає шлях від X до Y)

Позначимо $N_{A,B}(i)$ як прямих сусідів вершини i у дводольному графі $G_{A,B}$. Більш формально, $N_{A,B}(i)$ = {j ∈ вершини ($G_{A,B}$) | (j, i) ∈ ребра ($G_{A,B}$) }. І навпаки, ми визначимо $\widetilde{N}_{A,B}(i)$ як індекси j вершин, де (i, j) є ребром у ($G_{A,B}$). Ця функція N (або $\widetilde{N}$) може бути використана для негайного визначення залежностей. З попереднього параграфу ми маємо $N_{p,l}(3)$ = {AC, BC}: це означає, з'єднання 3 підтримує обидва шляхи AC і BC. І навпаки, продуктивність шляху BC залежить від з'єднань 2 і 3, отже, $\widetilde{N}_{p,l}$ (BC) = (2, 3). Слід зауважити, що цей останній список сусідів може бути впорядкований для отримання додаткової інформації про домен, оскільки з'єднання впорядковані в межах кожного шляху. Вершина A є частиною кожного з'єднання: $N_{l,n}$ (A) = {1, 2, 3, 4}. Нарешті, ми також можемо впорядкувати сусідів на основі залежностей шлях-вузол, наприклад, $\widetilde{N}_{p,n}$ (BC) = (B, A, C).

У рівнянні 3.2 наведено спрощену версію формули передавання повідомлень. Ця формула описує, як вкладення x^{t+1} оновлюються на основі попереднього значення x^t та сукупності вкладень сусідів. У порівнянні з оригінальною формулою передачі повідомлень, ми опускаємо аргументи x^t_j та $e^t_{i,j}$ у ϕ, щоб зменшити її складність, оскільки в нашому підході ребра GNN не мають вкладень.

$$x_i^{t+1} = \gamma(x_i^t, \square_{j \in N_{A,B}} \phi(x_j^t)) \quad (3.2)$$

Далі ми будемо використовувати дві різні функції агрегування □: одну для невпорядкованих вхідних даних, а іншу, яка краще підходить для впорядкованих вхідних даних. У типових GNN сусіди не впорядковані: функція агрегування може бути простою, як сума вкладених вхідних даних. Деякі пропозиції нормалізують цю суму, наприклад, шляхом ділення на степені задіяних вершин - це допомагає, коли деякі вершини мають набагато більше сусідів, ніж інші. Ми використовуємо інший підхід і пропонуємо декілька функцій агрегації, інваріантних до перестановок (наприклад, сума, середнє або максимум), які об'єднуються за допомогою конкатенації (позначимо ‖).

Agg : X = {$x_1, \ldots, x_n$} → (avg (X) ‖ sum (X) ‖ min (X) ‖ max (X) ‖ var(X)) (Невпорядкована агрегація).

Зважаючи на це, можна забезпечити впорядкування сусідів для багатшої агрегації. Наприклад, сусідні вершини можна впорядкувати за такими властивостями, як відстань або хімічна взаємодія. У нашому випадку з'єднання природно впорядковуються за їхнім положенням на шляху. Ми застосовуємо підхід, схожий на RouteNet, і використовуємо RNN для агрегації впорядкованих послідовностей вкладень.

OrdAgg : X = ($x_1, \ldots, x_n$) → RNN (X) (Впорядкована агрегація)

У решті частини цього розділу γ і ϕ є одношаровими персептронами, за якими слідує нелінійна функція активації. Ми позначимо стани вкладень для шляхів, з'єднань та вузлів як p, l та n відповідно. Інтуїтивно зрозуміло, що останні вкладення n є важливими для моделювання навантаження на вузли на основі трафіку шляхів та типу обслуговування (ToS). Однак, вони вводять значно більше параметрів для вивчення і можуть бути непотрібними, якщо немає взаємодії між усіма чергами на одному вузлі. У наступних підрозділах ми досліджуємо два підходи: один з вкладеннями вузлів n (A1) і один без них (A2).

Тепер застосувати визначення шарів передачі повідомлень до нашої задачі (рівняння 3.2). Спираючись попередній приклад, ми оновлюємо вбудовування

з’єднань із залежних вбудовувань шляхів: іншими словами, обчислюємо l^{t+1} з попереднього значення l^t і відповідного p^t. Цю операцію проілюстровано на рисунку 3.3 для «з’єднання n.3». Спочатку знаходимо шляхи, які підтримує з’єднання n.3: з попереднього пункту ми знаємо, що це шляхи AC та BC. Для кожного знайденого шляху ми обчислюємо повідомлення за допомогою функції ϕ; потім ми об'єднуємо всі ці повідомлення в один вектор за допомогою функції агрегації. Нарешті, ми використовуємо функцію оновлення γ для обчислення нового вбудовування на основі попереднього значення та результату агрегування.

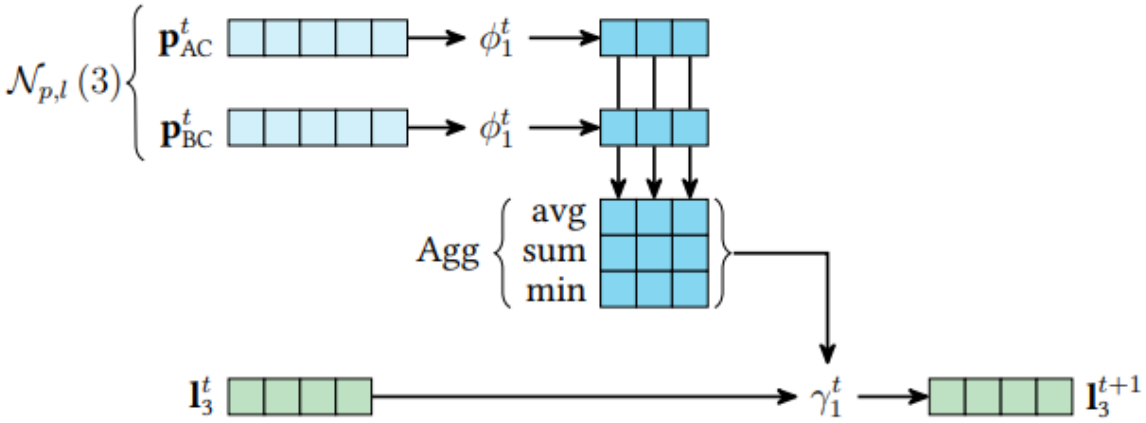


Рисунок 3.3 - Оновлення вбудовування зв'язку n.3 з прихованими станами сусіднього шляху (з імітаційної топології). Ми бачимо, як генеруються та агрегуються «повідомлення» для оновлення вбудовування з’єднань. Це еквівалентно

$$I_3^{t+1} = \Upsilon_1^t(I_3^t, \mathrm{Agg}_{j \epsilon N_{p,l}(3)} \phi_1^t(p_j^t))$$

3.4 Підходи для вивчення представлень

3.4.1 Підхід A1: спеціалізоване представлення з вбудовуванням вузлів

У цьому першому підході запропоновано додати третій тип вбудовування для вузлів. Таким чином, необхідно вирішити залежності між трьома типами вкладень: для шляхів, з'єднань та вершин. Запррпоновано вирішувати взаємозалежності між компонентами мережі шляхом поширення інформаційного сигналу від вбудовувань шляхів до вбудовувань з'єднань до вбудовувань вузлів. Ця стратегія безпосередньо випливає із взаємозалежностей, які можна інтуїтивно очікувати між цими компонентами мережі: трасування шляхів має прямий вплив на з'єднання, які, в свою чергу, мають прямий вплив на вузли. Однак перевантажені вузли можуть обмежити пропускну здатність каналів, що згодом вплине на показники продуктивності шляхів. Таким чином, важливо також поширювати залежності у зворотному напрямку: від вузлів до каналів до шляхів.

Взаємодію між вбудовуваннями показано на рисунку 3.4, а детальний алгоритм доступний в Алгоритмі 1. По-перше, ми ініціалізуємо вбудовування відповідними властивостями: вбудовування шляху ініціалізуються властивостями шляху разом із додатковим відступом, щоб відповідати розміру вбудовування. Та ж операція застосовується для з'єднань та вузлів - це відповідає кроку 1 на рисунку 3.4 та рядкам 1 по 3 в Алгоритмі 1. Потім ми виконуємо перший рівень передавання повідомлень, щоб оновити вбудовування з'єднань, використовуючи дводольний граф $G_{p,l}$ і пов'язані з ним залежності $N_{p,l}$ (крок 2, рядок 5).

Аналогічним чином ми оновлюємо вставки вузлів на кроці 3 (рядок 6), використовуючи $G_{l,n}$ та повідомлення від щойно оновлених вставок з'єднань. Потім ми поширюємо інформаційний сигнал «назад» від вузлів до з'єднань до шляхів, використовуючи зворотні залежності $\widetilde{N}_{l,n}$ та $\widetilde{N}_{p,l}$. Тобто відповідає крокам 4 і 5, рядкам 7 і 8. Для останнього повідомлення, що передається від з з'єднань до

шляхів, ми використовуємо впорядкування з'єднань всередині шляхів і використовуємо дві агрегації (рядок 9).

Algorithm 1: High-level pseudo-code of proposed GNN approach A1

Input: Features $\mathbf{F}_p, \mathbf{F}_l, \mathbf{F}_n$; neighbors $\mathcal{N}$ and $\tilde{\mathcal{N}}$; trainable functions ϕ, γ and FC.
Output: Readout performances of every path

▷ *State initialization, padding with zeroes*

1 $\forall i \in \mathbf{p} : \mathbf{p}_i \leftarrow [\mathbf{F}_{p,i}, 0, \ldots, 0]$
2 $\forall i \in \mathbf{l} : \mathbf{l}_i \leftarrow [\mathbf{F}_{l,i}, 0, \ldots, 0]$
3 $\forall i \in \mathbf{n} : \mathbf{n}_i \leftarrow [\mathbf{F}_{n,i}, 0, \ldots, 0]$
4 **for** $t \leftarrow 1$ **to** T **do**
▷ *Update link embeddings from path embeddings*
5 $\forall i \in \mathbf{l} : \mathbf{l}_i \leftarrow \gamma_1^t \left(\mathbf{l}_i, \mathrm{Agg}_{j \in \mathcal{N}_{p,l}(i)} \phi_1^t \left(\mathbf{p}_j\right)\right)$
▷ *Update node embeddings from link embeddings*
6 $\forall i \in \mathbf{n} : \mathbf{n}_i \leftarrow \gamma_2^t \left(\mathbf{n}_i, \mathrm{Agg}_{j \in \mathcal{N}_{l,n}(i)} \phi_2^t \left(\mathbf{l}_j\right)\right)$
▷ *Update link embeddings from node embeddings ("backward operation")*
7 $\forall i \in \mathbf{l} : \mathbf{l}_i \leftarrow \gamma_3^t \left(\mathbf{l}_i, \mathrm{Agg}_{j \in \tilde{\mathcal{N}}_{l,n}(i)} \phi_3^t \left(\mathbf{n}_j\right)\right)$
▷ *Update path embeddings from link embeddings, two aggregations*
8 $\forall i \in \mathbf{p} : \mathbf{p}'_i \leftarrow \gamma_4^t \left(\mathbf{p}_i, \mathrm{Agg}_{j \in \tilde{\mathcal{N}}_{p,l}(i)} \phi_4^t \left(\mathbf{l}_j\right)\right), \mathbf{p}''_i \leftarrow \gamma_5^t \left(\mathbf{p}_i, \mathrm{OrdAgg}^t_{j \in \tilde{\mathcal{N}}_{p,l}(i)} \phi_5^t \left(\mathbf{l}_j\right)\right)$
9 $\mathbf{p} \leftarrow FC_0 \left(\mathbf{p}' \parallel \mathbf{p}''\right)$

▷ *Bonus message passing from node to path embeddings*

10 $\forall i \in \mathbf{p} : \mathbf{p}'_i \leftarrow \gamma_6^{T+1} \left(\mathbf{p}_i, \mathrm{OrdAgg}^{T+1}_{j \in \tilde{\mathcal{N}}_{p,n}(i)} \phi_6^{T+1} \left(\mathbf{n}_j\right)\right)$

▷ *Start readout from path embeddings and include features back*

11 $\mathbf{r}_0 \leftarrow \left(\mathbf{p} \parallel \mathbf{p}' \parallel \mathbf{F}_p\right)$
12 $\mathbf{r}_1 \leftarrow \mathrm{Activation}\left(FC_1\left(\mathbf{r}_0\right)\right)$
13 $\mathbf{r}_2 \leftarrow \mathrm{Activation}\left(FC_2\left(\mathbf{r}_1\right)\right)$
14 $\mathbf{r}_3 \leftarrow \mathrm{Activation}\left(FC_3\left(\mathbf{r}_2\right)\right)$
15 **return** $FC_4\left(\mathbf{r}_3\right)$

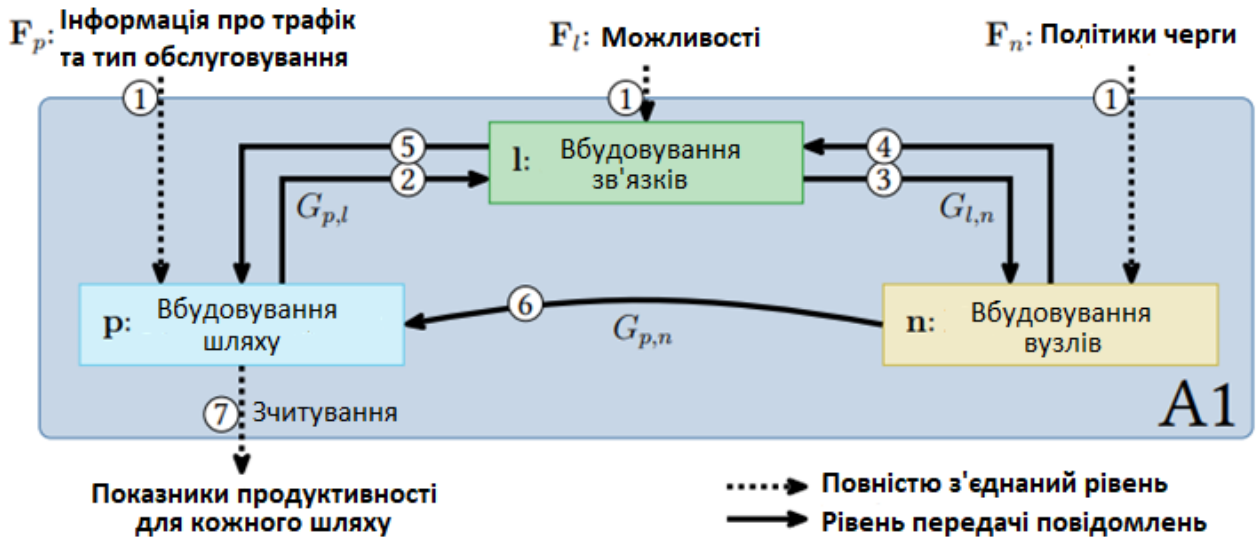


Рисунок 3.4 - Візуальне представлення підходу A1, запропонованого для задачі GNN. Спочатку вбудовування ініціалізуються з мережевих функцій 1 . Використовуючи операції передачі повідомлень над двосторонніми графами GNN $G_{p,l}$ та $G_{l,n}$, вбудовування оновлюються за допомогою кроків 2-5, які повторюються T разів. Вкладення шляхів оновлюються за допомогою $G_{p,n}$ 6, і ми зчитуємо прогнозовану середню затримку для кожного шляху 7, використовуючи MLP

На цьому етапі всі вбудовування були оновлені принаймні один раз. Однак, лише прямі залежності були змодельовані та поширені чотирма послідовними рівнями проходження повідомлень. Щоб виявити потенційні циклічні та непрямі залежності, ми використовуємо ту ж стратегію, що і RouteNet, яка вирішує цю проблему шляхом повторення циклу проходження повідомлень T разів (рядки з 5 по 9). Кожна нова ітерація поширює взаємозалежності на один крок далі, що дозволяє алгоритму виявити нетривіальні залежності між компонентами мережі. Цей підхід раніше використовувався в GNN, що передають повідомлення, як ітераційна апроксимація з однією точкою. Треба звернути увагу на вибрані індекси та експоненти у формулах Алгоритму 1: як приклад у рядку 5, γ_1^t і ϕ_1^t відповідно позначають функції оновлення (відповідно, повідомлення), індексовані 1 на ітерації t. Це означає, що кожна ітерація («шар GNN») має свій власний набір функцій, що навчаються, тобто параметрів, що навчаються, які дозволяють

вбудовувати збіжність. Хоча RouteNet використовував T = 8, ми виявили, що T = 3 є хорошим компромісом між точністю та обчислювальною складністю. Маючи в середньому трьох сусідів, один компонент мережі може досягати $3^{4\times T}$ інших компонентів (тобто більш ніж достатньо для збіжності вбудовувань). (Для наочності ми опускаємо індекси вкладеності в алгоритмі, оскільки під час оновлень ми перезаписуємо попередні значення вкладеності).

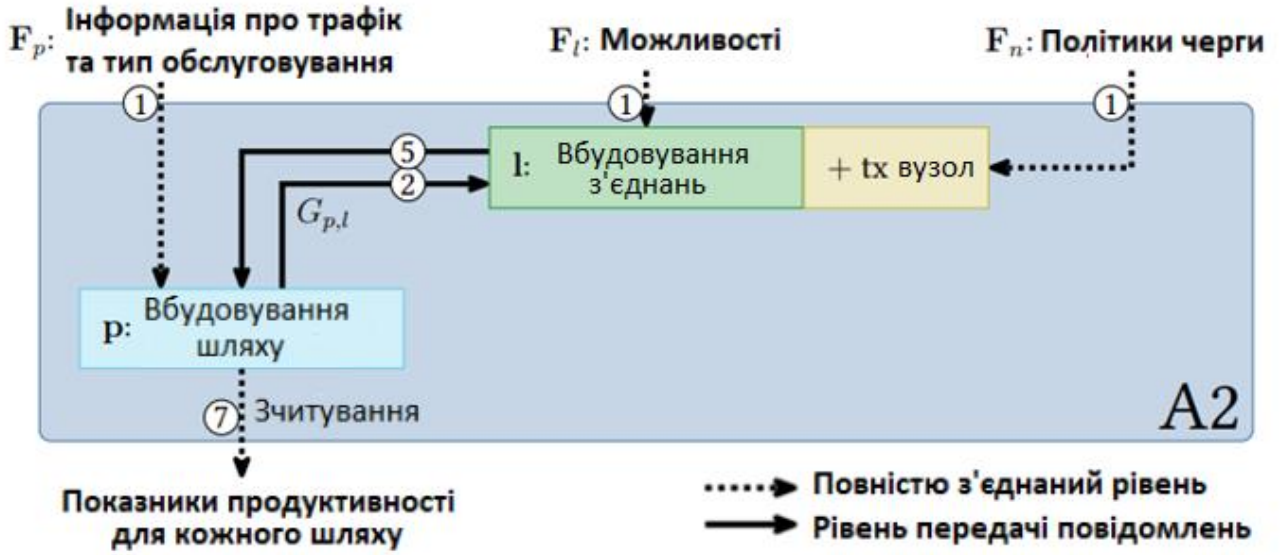


Рисунок 3.5 - Візуальне представлення підходу A2, досліджуваного в задачі GNN. У порівнянні з підходом A1 (рисунок 3.4), вбудовування з'єднань доповнюється функціями передавального вузла F_n. Кроки 3, 4, 6 у цьому підході не потрібні

Після виконання цих T ітерацій ми використовуємо третій дводольний граф $G_{p,n}$ для обчислення альтернативних вкладень p' з залежних вершин на кроці 6, рядок 10. Ця остання операція передачі повідомлення знову використовує впорядкування вершин всередині кожного шляху завдяки функції агрегації OrdAgg. Ми припускаємо, що ця остання операція дозволяє моделі використовувати непряму залежність між шляхами і вузлами без використання з'єднань як посередників.

Нарешті, настає черга «Зчитування» нашої GNN (крок 7, рядок 11). Ми пам'ятаємо, що ми хочемо отримати метрики продуктивності зі шляхів, тому нам потрібно зчитати інформацію безпосередньо з вкладень шляхів p і p'.

Характеристики шляху F_p можуть бути розмиті послідовними операціями передачі повідомлень. Щоб гарантувати, що вони безпосередньо відображаються на вихід моделі, ми додаємо їх до входів зчитування (рядок 11). Подібно до зчитування RouteNet, ми використовуємо MLP з трьома прихованими шарами (позначений *FC* для «повністю з'єднаний») і нелінійними функціями активації (рядки з 11 по 15). Варто зазначити, що цей повний алгоритм застосовується як під час навчання, так і під час виведення, для кожної вибірки даних.

3.4.2 Підхід A2: представлення вузлів через вбудовування з’єднань

Другий підхід є спрощеною версією підходу A1, де вкладення вузлів не моделюються в явному вигляді, а «підмішуються» до вкладень з’єднань. У початковій моделі мережі кожен зв'язок і асоціюється з вузлом-передавачем tx(i) та вузлом-приймачем rx(i). Таким чином, ми ініціалізуємо вкладення з’єднань відповідними характеристиками з’єднання та характеристиками передавального вузла. (Ми припускаємо, що мережеві черги моделюються лише для вихідних повідомлень, тому ми ігноруємо властивості вузла-одержувача у цьому підході).

$$\forall_i \in 1: 1_i \leftarrow \left[F_{l,i}, F_{n,tx(i)}, 0, \dots, 0\right]$$

(Замінює рядок 2 в Алгоритмі 1)

Слід зауважити, що тут ми фактично моделюємо один набір черг для кожного з’єднання. Це відрізняється від підходу A1, де черги розподілялися між усіма вихідними зв'язками кожного вузла завдяки спеціальним вбудовуванням у вузли.

3.5 Представлення наборів даних та результати випробувань

Дотримуючись стандартної методології методів машинного навчання, ми мали доступ до трьох різних наборів даних: один для навчання, один для перевірки і один для оцінювання. У цих наборах даних кожна вибірка містить результати однієї симуляції мережі OMNeT++ за відомою топологією з певними характеристиками компонентів. Навчальний набір даних містить зразки двох різних топологій: один з історичної мережі Національного наукового фонду (NSFNET) з 14 вузлами і 21 зв'язком, а інший - з більшої гігабітної європейської мережі Advanced Network Technology 2 (GÉANT2), що містить 24 вузли і 37 з'єднань. Топологія Німецької магістральної мережі (GBN) (17 вузлів і 26 з'єднань) використовується в апробаційному наборі даних. Нарешті, оціночний набір містить зразки з невідомою топологією, що мають 19 вузлів і 31 з'єднання.

В імітаційній моделі, що використовується для створення зразків, канали мають обмежену пропускну здатність, але нульову затримку, і вони не втрачають повідомлення. Однак OMNeT++ імітує черги повідомлень на кожному вузлі: вихідні повідомлення затримуються до тих пір, поки не буде достатньо пропускної здатності каналу, щоб відправити їх на наступний вузол. Коли пакет потрапляє в повну чергу або затримується більше, ніж на 20 одиниць часу моделювання, він відкидається. Звичайно, реальні комп'ютерні мережі мають з'єднання з позитивною затримкою. Тим не менш, підхід, застосований у цій задачі, цікавий тим, що він може досить точно імітувати проблеми буферизації та перевантажень.

Кожен зразок містить наступну інформацію:

- топологія мережі, включаючи інформацію про маршрутизацію шляхів;

- F_p, характеристики, пов'язані з кожним шляхом, включаючи кількість згенерованого трафіку і ToS;

- F_l, відповідні характеристики для з'єднань (на практиці єдиною характеристикою була пропускна здатність з'єднання);

- І F_n - характеристики вузлів (тобто політики черг і необов'язкові ваги черг).

Одним з важливих параметрів моделювання є матриця трафіку, що представляє частоту повідомлень на кожному шляху мережі. Ця частота визначається параметрами λ пуассонівських процесів - очікуваною кількістю повідомлень, що надходять за одиницю часу. (Розміри повідомлень випадково вибираються з бімодального розподілу.) У кожній вибірці є один шлях для кожної пари вузлів джерело-одержувач (наприклад, 14 × 13 шляхів для NSFNET). Крім того, кожен шлях має випадково вибраний ToS (0, 1 або 2, що поєднується з політикою черги у вузлах). Оскільки вузли не з'єднані безпосередньо з кожним іншим вузлом, також передбачена матриця маршрутизації: вона використовується проміжними вузлами для пересилання повідомлення до кінцевого пункту призначення через певні канали зв'язку. Матриці трафіку та маршрутизації генеруються випадковим чином для кожної вибірки, що дає багато унікальних мережевих конфігурацій.

Порівняно з початковим дослідженням RouteNet, було введено політику ToS шляху та політику черги вузлів. У симуляторі вузли вибирають наступні повідомлення, які вони надсилають до кожного каналу, з трьох різних черг rst-in rst-out. Використовуються наступні політики:

- повідомлення вибираються з черг з найвищими пріоритетами за допомогою політики суворого пріоритету;

- за допомогою зваженої справедливої черги (Weighted Fair Queuing, WFQ), чергам надається частина пропускної здатності каналу;

- нарешті, пропонується децит-раунд-робін (Decit Round Robin, DRR) як обчислювально ефективне наближення до зваженої справедливої черги.

Ці три запропоновані політики відповідають широко використовуваним політикам планування, які, наприклад, доступні в ядрі Linux. Завдяки різноманітності стратегій розподілу трафіку і черг, запропонована імітаційна модель може відтворювати поведінку реальних маршрутизаторів комп'ютерних мереж. Для кожної згенерованої мережевої конфігурації симуляція OMNeT++ вимірює показники продуктивності на кожному шляху, включаючи середню пропускну здатність, затримку, джиттер і рівень втрат. Симуляція конфігурації

зупиняється, коли ці показники досягають стаціонарного стану: на практиці одна вибірка набору даних займає від 30 секунд до більш ніж десяти хвилин часу моделювання - відповідно до наданих даних.

Після налаштування гіперпараметрів ми отримали від нашої найкращої моделі (на основі A1) MAPE 1,66% на оціночному наборі даних після 750 000 навчальних вибірок. Ми деталізуємо обрані гіперпараметри в таблиці 3.1. Ми хотіли б підкреслити, що кількість вбудовувань значно більша, ніж оригінальних вбудовувань RouteNet (400 проти 8). Це пояснюється тим, що призводить до більш виразних моделей, але також вимагає більше обчислень і обсягу пам'яті: в той час як RouteNet потребує менше дня для навчання, наша найкраща модель потребувала близько чотирьох днів.

Таблиця 3.1 - Гіперпараметри, що призводять до найкращої моделі

Batch size	8
Embeddings size	400 for paths, links and nodes (**p**, **l**, **n**)
RNN size	same as embeddings (400)
FC_1 size	512
FC_2 size	256
FC_3 size	256
T	3
Activation	Leaky Rectified Linear Unit
Total params	11 465 185

Таблиця 3.2 - Зведення гіперпараметрів, використаних у моделях для нашого найкращого представлення

Approach	Loss function	Hidden state size	FC_1 size	FC_2 size	FC_3 size	T
A1	MAPE	400	512	256	256	3
A1	MSE	400	512	256	256	3
A1	MAPE	300	128	128	256	3
A1	MSE	300	128	128	256	3

Після успішних підходів з машинного навчання було застосовано різні ансамблеві методи для подальшого покращення нашого результату. У задачі GNN найуспішнішим підходом виявилося усереднення ансамблю. Ми тренували

діерентні моделі з дещо зміненими гіперпараметрами. Хоча кожна з цих моделей мала відсоток помилок, вищий за 1,66% на оціночному наборі, усереднення їхніх результатів фактично призвело до меншої помилки. Цього можна було очікувати: об'єднання кількох моделей зазвичай призводить до зменшення відхилення та дисперсії між результатами.

Найкраще рішення використовувало середнє гармонійне значення результатів чотирьох моделей, що призвело до MAPE 1,53% (таблиця 3.2). Перевага цього середньоквадратичного значення полягає в тому, що воно надає перевагу малим значенням - так само, як і MAPE, - зменшуючи похибку в наших прогнозах затримок. Ми також спробували стекування як інший ансамблевий метод, але без особливого успіху.

3.6 Методологія та деталі реалізації

Під час дослідження було випробуванго багато ідей та протестовано їх на наборі даних для валідації, спочатку додавши вбудовування вузлів і кілька додаткових шарів передачі повідомлень, потім спробувавши різні GNN і поступово збільшуючи розміри вбудовування і прихованих шарів. Як і в багатьох інших задачах, цей інкрементальний процес ґрунтувався на методі спроб і помилок: було відкидані ідеї, які не давали кращої моделі після навчання, що давало кращий результат на валідаційному наборі даних. Звичайно, через обчислювальні та часові обмеження неможливо було випробувати кожну комбінацію варіантів. Гіперпараметри були обрані за допомогою інтуїції, знань предметної області та мікробенчмарків в інкрементальному процесі.

3.6.1 Реалізація за допомогою PyTorch

Було реалізовано представлений алгоритм з нуля за допомогою PyTorch 1.4 та модуля PyTorch Geometric. Цей модуль містить численні примітиви для передачі повідомлень GNN: зокрема, він надає хороші абстракції та функції агрегації графів з GPU-прискоренням. Для повного використання можливостей PyTorch Geometric нам довелося використовувати єдиний тензор, що містить усі вбудовування для шляхів, з'єднань і вузлів: це вимагало деяких додаткових операцій заповнення і конкатенації порівняно з Алгоритмом 1. Даний алгоритм реалізовано менш ніж у 500 рядках коду на python, причому вдвічі більше для попередньої обробки та оцінки даних.

Вхідні дані мають значні відмінності між собою: наприклад, траєкторія шляху варіюється від 40 до 2000, тоді як пропускна здатність з'єднань варіюється від 10 000 до 100 000. Тому ми стандартизуємо безперервні характеристики шляхом видалення середнього значення та масштабування до одиничної дисперсії - ця класична попередня обробка сама по собі дозволила нам значно покращити базову лінію RouteNet. Крім того, характеристики вузлів мають лише обмежений набір значень: є лише три можливі політики черговості, а для двох з них - по 5 комбінацій ваг, що в сумі дає 11 можливих комбінацій. Ми кодуємо ці 11 комбінацій у 4-вимірне вбудовування (не плутати з вбудовуваннями компонентів мережі p, l, n): це «вхідне вбудовування» відповідає виходу одношарового персептрона, оптимізованого під час процесу навчання. Для шляхів ToS ми застосуємо інше вхідне вбудовування, перетворивши три можливі значення у розмірність розміру 4.

Було запущено алгоритм на внутрішній обчислювальній сітці з графічними процесорами Nvidia Tesla M60, P100 та V100. Кожен з цих графічних процесорів містить від 2048 до 5120 ядер CUDA та 8 або 16 ГБ пам'яті. Під час пошуку гіперпараметрів було обмежено навчання до 200 000 навчальних вибірок. Це відповідає половині наданих навчальних вибірок, але дозволяє підтримувати розумну тривалість навчання на спільній обчислювальній сітці (в середньому

менше одного дня). Було сфокусовано на найкращій архітектурі, і натренувано її на 750 000 зразків. З обраними гіперпараметрами це зайняло близько чотирьох навчальних днів зі швидкістю 2,7 вибірок на секунду на Tesla M60 (для порівняння: 4,8 вибірок на секунду на P100).

3.6.2 Методи для оптимальної стратегії навчання

Щоб отримати хороші моделі від алгоритму машинного навчання, необхідно забезпечити хорошу стратегію навчання. Розглянемо два методи, які дозволили значно покращити результат:

- використання логарифму у функції втрат.

При цьому використовується MAPE-похибка для оцінки рішень за очікуваними середніми затримками. Ця метрика надає перевагу невеликим прогнозам: передбачення 1 при очікуваному результаті 5 призводить до помилки 80%, тоді як передбачення 5 при очікуваному результаті 1 призводить до помилки 400%. Здавалося б, важливо навчити моделі на цій функції втрат, але це передбачає деякі практичні міркування. По-перше, неможливо нормалізувати або стандартизувати очікуваний результат моделі (нульове значення призведе до ділення на нуль, а від'ємні значення не мають сенсу для MAPE). По-друге, очікувані затримки варіюються від 0,0075 до 20 одиниць часу моделювання, причому більшість значень не перевищують 1. Для забезпечення числової стабільності (і оскільки в цій задачі нас цікавлять відносні затримки) ми навчали наші моделі за логарифмом розподілу очікуваних середніх затримок: $d_i' = \log(d_i + 1) > 0$, де d_i - фактична затримка, яку потрібно спрогнозувати, а d_i' - перетворене значення, яке ми використовуємо для навчання.

Розглядаємо ефект цього підходу на рисунку 3.6, де можна порівняти прогнози двох моделей, побудованих на основі того самого алгоритму (A1).

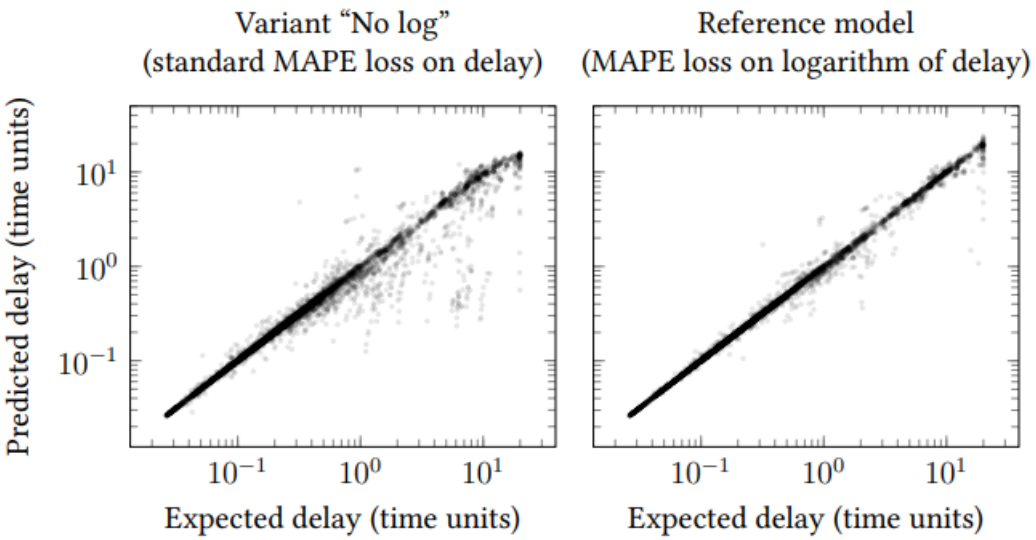


Рисунок 3.6 - Прогнозована та очікувана затримка для деяких шляхів (кожна точка - прогноз)

Перша модель (ліворуч) була навчена з використанням оригінальних затримок, тоді як друга (праворуч) використовувала логарифмічний метод, який ми щойно детально описали. Ми можемо чітко бачити, що ліва модель зміщена в бік прогнозування низьких затримок; це не дивно з огляду на визначення втрат MAPE. Застосування логарифмічної функції до втрат (правий графік) суттєво зменшило це небажане зміщення, оскільки «надмірні» прогнози менш забраковані: у попередньому прикладі, передбачення 5 замість 1 призвело б до втрат 159% (замість 400%).

3.7 Циклічне планування швидкості навчання

Швидкість навчання процедури навчання є основним фактором, який гарантує, що моделі дійсно збігаються до кращих параметрів. Занадто низька швидкість уповільнює навчання більше, ніж потрібно, тоді як вища швидкість може призвести до нестабільності результатів. Щоб уникнути цих проблем, ми використали оптимізатор Адама з параметрами за замовчуванням. Якщо коротко,

цей оптимізатор працює, підтримуючи адаптивну швидкість навчання для кожного параметра, що навчається, оновлюючи цю швидкість навчання відповідно до спостережуваної дисперсії та узгодженого розміру кроку α . Однією з популярних стратегій є зміна швидкості навчання (відповідно, α) в повторюваних циклах; спочатку збільшуючи її до певного порогу, а потім зменшуючи її протягом навчальних партій. З цією метою було використанго дві політики планування, що надаються PyTorch:

- від вибірки 0 до 500 000 α циклічно змінюється від мінімального «базового» значення 1×10^{-8} до максимального 1×10^{-3} , зі змінами кожні 128 вибірок. Фаза зростання охоплює 1280 вибірок, тоді як фаза зменшення триває понад 102 400 вибірок з використанням експоненціального коефіцієнта спаду 0,999;

- починаючи з вибірки 500 000 і далі: використовуються ті самі гіперпараметри, за винятком базового значення 1×10^{-10} і максимального значення 1×10^{-4} .

На рисунку 3.7 зображено еволюцію α та спостережуваних втрат у процесі навчання.

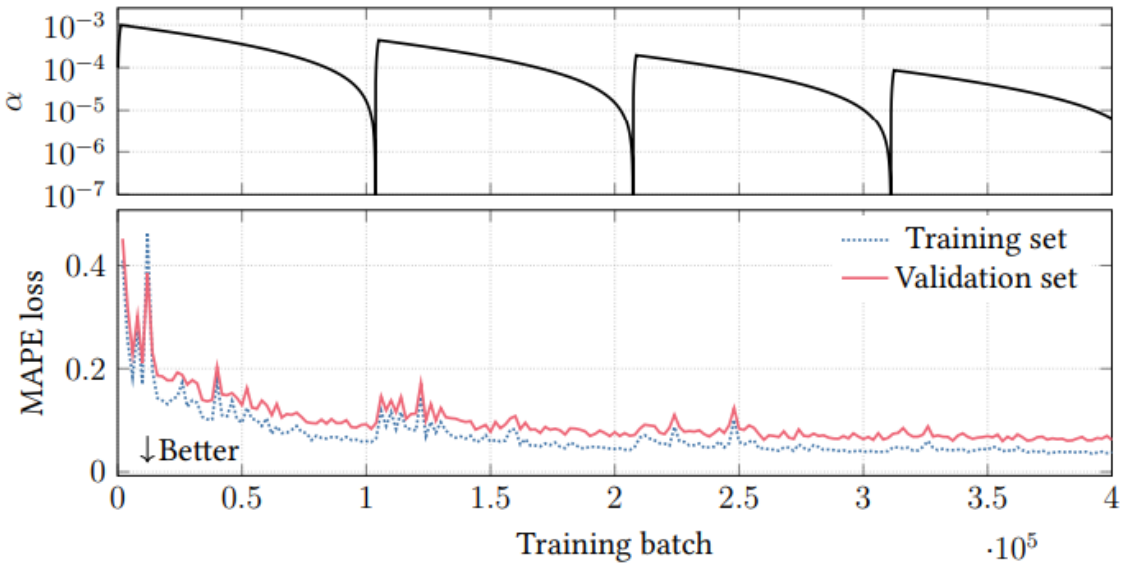


Рисунок 3.7 - Значення α Адама та втрати MAPE під час перших партій навчання. Було використано планувальник циклічної швидкості навчання PyTorch

Обрана політика планування дозволила нашим моделям досягти нижчих значень похибки порівняно з монотонно спадними планувальниками. Інтуїтивно зрозуміло, що ця політика уникає локальних мінімумів, «перестрибуючи» в недосліджені області простору параметрів. Це добре видно в районі партії 100 000: після повільного зменшення швидкість навчання стрибає назад до приблизно 5×10^{-4} . Потім втрати при навчанні зростають приблизно до 50 000 партій, перш ніж досягти нижчих значень. Зауважимо, що втрати на валідаційному наборі даних слідують за втратами на навчальному наборі даних з розумним інтервалом, не показуючи жодних ознак перенавчання.

3.8 Аналіз абляції

Представлені підходи GNN включають декілька внесків. Для кращого розуміння ефекту кожного з цих внесків у цьому розділі ми наводимо аналіз абляції. Метод полягає в наступному: по-перше, ми створюємо варіанти найскладнішого підходу A1, в кожному з яких один компонент вимкнено або дещо змінено. Потім ми навчаємо ці варіанти зі схожими гіперпараметрами і порівнюємо їхні результати з еталонною моделлю, основаною на A1. Ми також порівнюємо варіанти з альтернативним підходом A2.

Варіанти:

- Baseline8 та Baseline400: Ці варіанти базуються на наданій моделі RouteNet. Ми повторно реалізували RouteNet у нашому фреймворку PyTorch і використали ті ж самі гіперпараметри навчання, що і в нашій еталонній моделі. (Оптимізатор Adam за втратами MAPE із застосуванням логарифму, з використанням циклічного планування швидкості навчання). В той час як модель RouteNet спочатку використовувала вкраплення розміром 8 (Baseline8), ми також реалізували варіант Baseline400 з вкрапленнями розміром 400. У варіанті Baseline400 гіперпараметри для шарів зчитування відповідають гіперпараметрам A1 (таблиця 3.1).

- No log: Простий варіант, який не використовує логарифмічні цілі у функції втрат. Ми все ще стандартизуємо функції, але використовуємо «сирі» цільові затримки від 0 до 20 одиниць часу.

- Sum agg: Щоб оцінити ефект запропонованої нами невпорядкованої агрегації «Agg», ми замінили її простішою версією, яка обчислює лише глобальну суму її входів. Слід зауважити, що в цьому варіанті сума застосовується до 400 вимірів вхідних вкладень. В еталонній моделі вкладиші спочатку зводяться до розмірності 80 за допомогою персептрона щоб дозволити конкатенацію 5 операторів. За допомогою «Sum agg» цей персептрон зберігається як додатковий прихований шар розміром 400. Таким чином, цей варіант має трохи більше параметрів, ніж еталонна модель.

- No path-node: У цьому варіанті ми просто видаляємо крок пересилання повідомлень з вузлів на шляхи (крок 6, рядок 10). Оскільки відповідні функції передачі повідомлень не потрібно вивчати, загальний розмір моделі зменшується.

Таблиця 3.3 - Порівняння скорочених варіантів на оціночному наборі даних після навчання на одному мільйоні вибірок. Оцінки MAPE наведено для повного набору даних (стовпчик «всі») і лише для великих очікуваних затримок (затримки понад 5 одиниць часу, стовпчик «>5»)

Variant	# Parameters	MAPE (all)	MAPE (>5)	MSE	R^2
Reference A1	11 465 185	1.63%	4.23%	0.078	0.983
Approach A2	7 702 305	1.15%	3.63%	0.072	0.984
Baseline8	13 089	7.37%	27.1%	0.840	0.817
Baseline400	2 095 561	4.81%	20.8%	0.808	0.824
No log	11 465 185	1.61%	4.36%	0.134	0.971
Sum agg	13 005 025	1.52%	4.14%	0.082	0.982
No path-node	9 817 185	3.38%	4.83%	0.165	0.964

3.9 Результати дослідження

Було протестувано кожен варіант на більш ніж одному мільйоні зразків (на 250 000 більше, ніж під час конкурсу). Під час цього аналізу було протестовано кожний варіант з підходів, маючи доступ до очікуваних затримок в оціночному наборі даних, де була можливість оцінити варіанти на цьому наборі даних.

У таблиці 3.3 наведено MAPE, MSE та коефіцієнт детермінації R^2. Ця остання міра показує, яка частина дисперсії в прогнозі пояснюється моделлю: модель, яка повертає постійну затримку в русі, може мати хороші MAPE і MSE, але все одно мати максимальний R^2, що дорівнює нулю. І навпаки, модель, яка ідеально відповідає даним, матиме коефіцієнт детермінації, що дорівнює одиниці.

Базову лінію можна покращити за допомогою оптимізованої стратегії навчання. Завдяки стандартизації функцій та використанню вдосконаленого оптимізатора і планувальника, нам вдалося отримати MAPE лише 7,37% для базової лінії RouteNet. Цей результат є досить вражаючим, оскільки ця базова лінія не може використовувати можливості вузлів (політики черг та ваги). Додавання більшої кількості параметрів до RouteNet прискорює навчання (рисунок 3.8) і призводить до ще кращих результатів (4,81%).

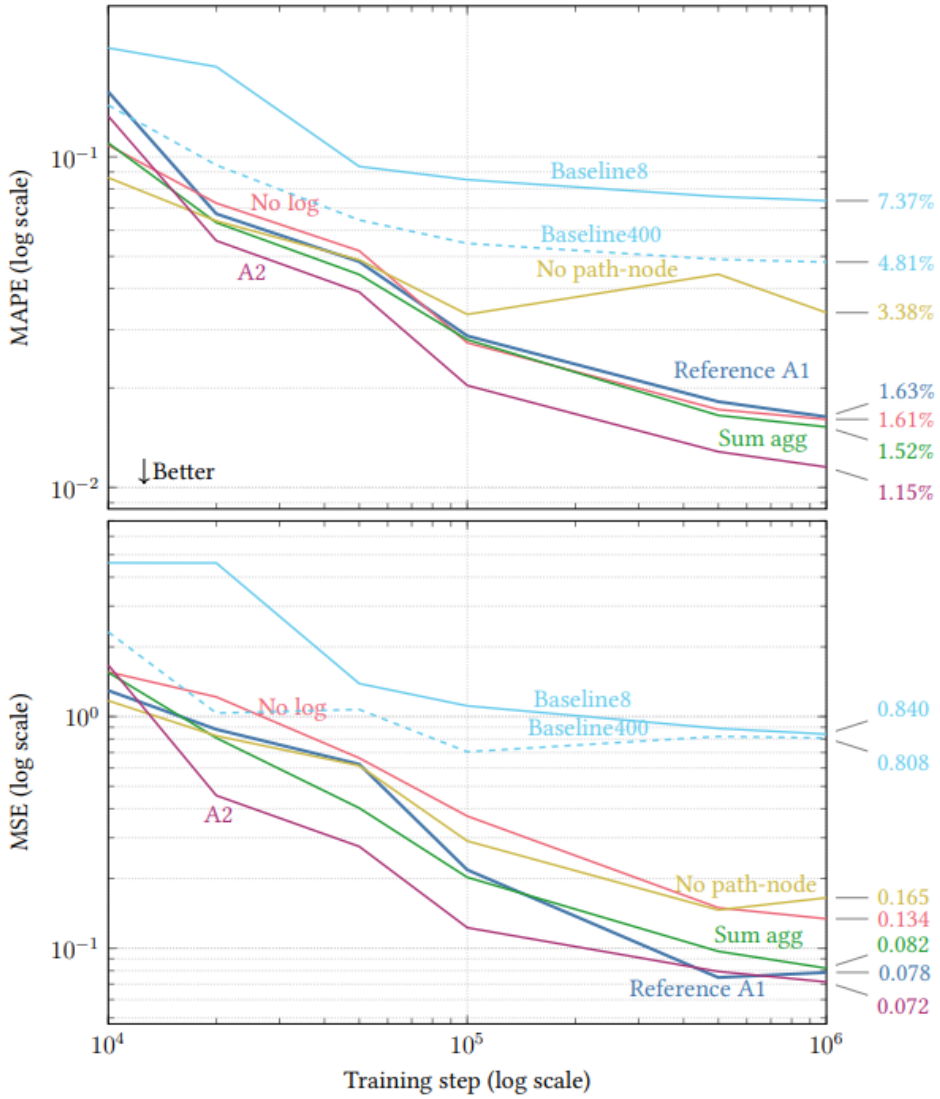


Рисунок 3.8 - Порівняння ефективності варіантів абляції при збільшенні навчальної вибірки

Простіші агрегації можуть бути кращими на практиці. Варіант «Sum agg» залишається на рівні з нашим еталоном як за показниками MAPE, так і за показниками MSE. Це вказує на те, що одиночні агрегації працюють краще, ніж запропонована нами складена агрегація з 5 глобальними функціями (Agg). Насправді, це не дивно для цього набору даних: затримки є адитивними по з'єднанням, отже, суми є достатніми. Дуже ймовірно, що якби ми хотіли обчислити інші метрики (наприклад, коефіцієнт втрат), то краще підійшли б інші функції агрегування. Більше того, у цьому варіанті можна підсумовувати повні вклади

розміром 400. У нашій еталонній моделі підсумовується лише одна п'ята частина вкладень (80) за рахунок обчислення інших функцій агрегування (avg, min, ...).

«Логарифмічні затримки» призводять до менш зміщених результатів. Під час оптимізації нашої еталонної моделі ми виявили, що використання логарифму очікуваної затримки призводить до кращих оцінок MAPE. Однак це не зовсім зрозуміло з нашого дослідження абляції: варіант «Без логарифма» насправді має дещо кращий показник MAPE (0,02% різниця). Дивлячись на MSE та R^2, ми бачимо, що наша еталонна модель все ще працює краще. Це підтверджує, що втрата MAPE стимулює моделі надавати перевагу низьким затримкам: незважаючи на дещо гірший показник MAPE в цілому, логарифми допомагають забезпечити кращі MSE і R^2 разом з кращим MAPE для високих очікуваних затримок на шляху.

Просто вилучення прямої передачі повідомлень між вузлами шляху у варіанті «Без вузла шляху» має величезний вплив на фінальні результати. Це чітко підтверджує необхідність такого прямого прошарку між вузлами та шляхами в нашій еталонній моделі. Однак, видалення вкладень вузлів і пов'язаних з ними операцій передачі повідомлень (підхід A2) насправді дає кращі результати в нашому аналізі, як для для MAPE, MSE та R^2 . Ця тенденція помітна на початку процесу навчання (після 20 000 вибірок, рисунок 3.8), що пояснюється такими чинниками:

- симуляція, яка використовувалася для генерації набору даних, моделювала один набір черг для кожного вихідного порту. Вкладення вузлів були введені для того, щоб дозволити моделі зберігати певне з’єднання між чергами вузла, наприклад, через спільний ресурс (процесор, пам'ять тощо). На рисунку 3.9 показано ілюстрацію стандартного маршрутизатора з вхідною та вихідною чергами: у деяких випадках пакети можуть затримуватися, незважаючи на порожні вихідні черги. У цьому наборі даних цей додатковий статус на рівні вузла насправді не потрібен, оскільки черги моделюються лише для вихідних портів. Це, безумовно, робить підхід A2 більш релевантним.

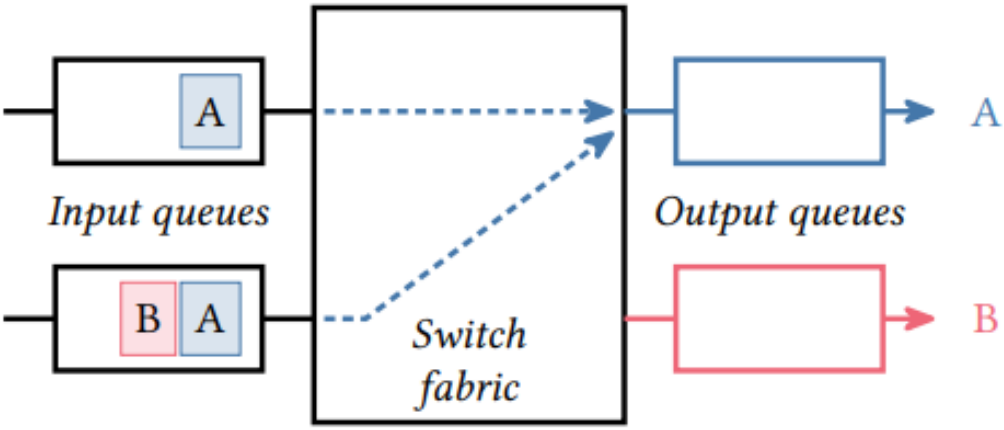


Рисунок 3.9 - Зображення реалістичного маршрутизатора з вхідною та вихідною чергами. У цьому прикладі існує певна суперечка між двома пакетами, що направляються на вихідний канал «А». Крім того, незважаючи на порожні вихідні черги, пакет до каналу «В» заблокований у другій вхідній черзі «в початок рядка»

- оновлення в А2 виконуються лише між вбудовуваннями шляхів та з’єднань. Оновлень через передачу повідомлень менше, ніж у еталонній моделі, і немає «проміжних» оновлень між шляхами та вузлами. Як підкреслюється низькою продуктивністю «Без вузла шляху», прямі оновлення є дуже важливими. Подальша робота може бути зосереджена на функціях навчання, які приймають на вхід декілька типів вбудовувань одночасно; наприклад, обчислюючи вбудовування вузлів з пов'язаних вбудовувань з’єднань та вбудовувань шляхів одночасно.

3.10 Візуалізація вивчених представлень

Прогнозована затримка кожного шляху витягується з відповідного вбудовування шляху за допомогою зчитування повністю з'єднаних шарів. Таким чином, цікаво проаналізувати зміст цих вивчених представлень, щоб перевірити здатність моделі витягувати відповідні характеристики для нашої конкретної задачі. Щоб зробити це у зрозумілий спосіб, ми спроектуємо високорозмірні

вбудовування (тобто 400 вимірів) до інтерпретованого двовимірного простору за допомогою t-розподіленого вбудовування стохастичних сусідів (t-SNE), техніки зменшення розмірності, яка зазвичай використовується для візуалізації високорозмірних зображень. У літературі ця техніка, наприклад, використовувалася для візуалізації вбудовування екранів відеоігор. Далі ми використовуємо найкращу модель, яку ми отримали на даний момент (з використанням підходу A2).

На рисунку 3.10 показано результат проекції t-SNE на валідаційні вибірки з різними кольорами для відповідних ознак у випадку проблеми прогнозування затримок (кожна точка представляє шлях).

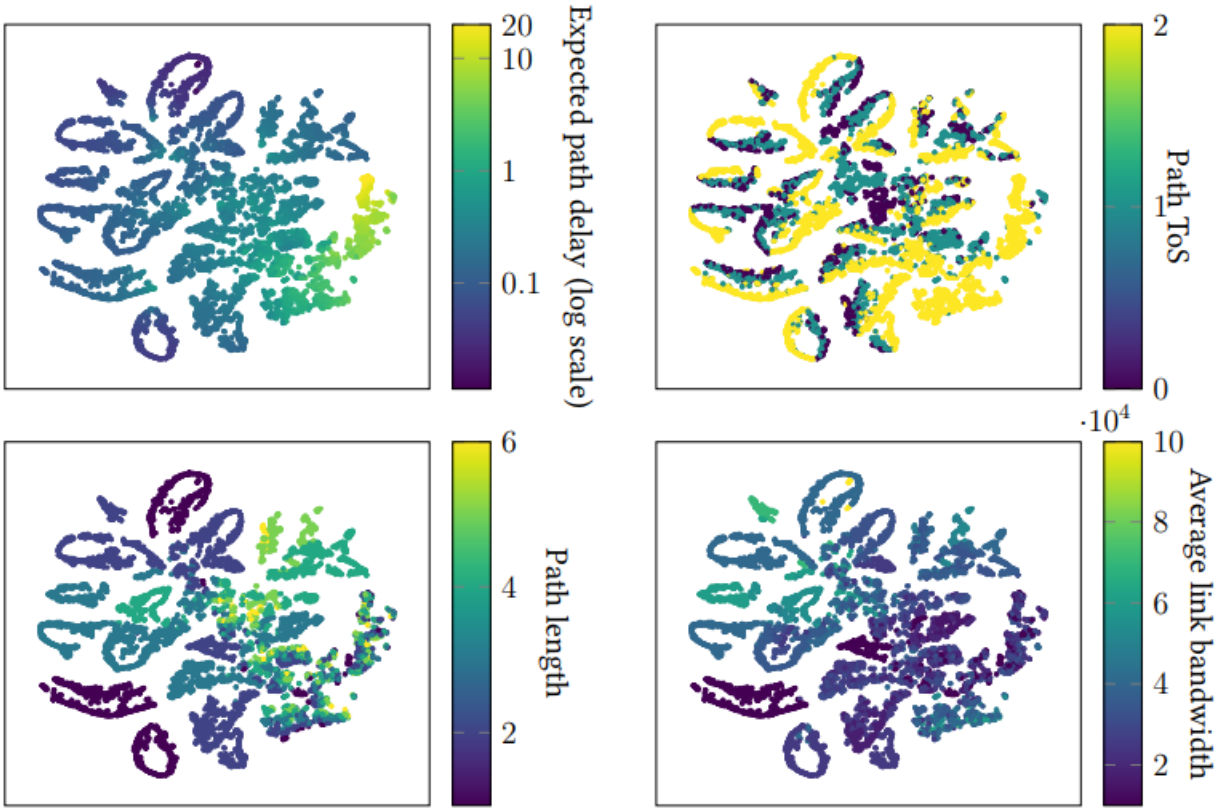


Рисунок 3.10 - 2-компонентна t-SNE-візуалізація вкладених маршрутів, забарвлена відповідно до очікуваної затримки на маршруті (вгорі ліворуч), типу обслуговування на маршруті (вгорі праворуч), довжини маршруту в кількості каналів (внизу ліворуч) та середньої пропускної здатності каналів у маршрутах (внизу праворуч). Враховуючи видимі кластери, можна сказати, що ці характеристики правильно вбудовуються у вбудовування

На верхньому лівому графіку ми розфарбували шляхи відповідно до очікуваної середньої затримки, і ми можемо чітко бачити плавний градієнт кольору між низькими і високими затримками, натякаючи на те, що вбудовування містять достатньо інформації, щоб дати хороші прогнози затримок. Верхній правий графік натякає на те, що шляхи з високими затримками частково корелюють зі шляхами, що мають ToS 2. Це не дивно: цей ToS має найменшу вагу для політик черги WFQ і DRR у наданому наборі даних, отже, шляхи з ToS 2 мають менший пріоритет, ніж інші шляхи. На нижніх графіках показано, що приховані стани також містять інформацію про зв'язки, що складають кожен шлях. Ми можемо чітко бачити, що шляхи згруповані за довжиною на лівому нижньому графіку і за середнім значенням пропускної здатності каналів на правому графіку, що вказує на те, що шари передачі повідомлень є дуже корисними для вилучення важливих сигналів. Варто зазначити, що групи демонструють чіткий розподіл значень ToS.

Результати аналогічні для вбудовування каналів, як показано на рисунку 3.11.

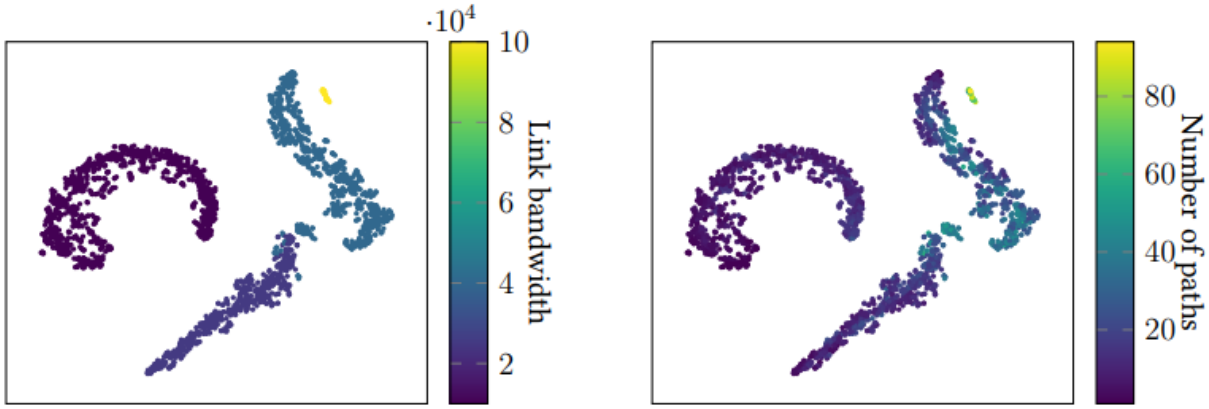


Рисунок 3.11 - Візуалізація t-SNE вбудованих з’єднань, розфарбованих відповідно до їхньої пропускної здатності (ліворуч) та кількості шляхів, які використовують кожне з’єднання (праворуч).

На лівому графіку ми бачимо, що інформація про пропускну здатність з’єднань зберігається, незважаючи на численні оновлення та об'єднання повідомлень. На правій діаграмі ми зобразили кількість шляхів, що проходять через кожне з’єднання. Ми бачимо, що з’єднання правильно розрізняються на низько- та високошвидкісні, як і очікувалося для задачі оцінки затримки на шляху.

ВИСНОВКИ

У цій роботі представлено та оцінено передові методи моделювання комп'ютерних мереж за допомогою графових нейронних мереж (ГНМ). Зокрема, запропоновано нову архітектуру моделі (A1), яка використовує кілька двосторонніх графів у шарах передачі повідомлень для моделювання взаємозалежностей компонентів у знаннях комп'ютерної мережі. Після оптимізації гіперпараметрів було отримано модель машинного навчання, пристосованого для завдання прогнозування затримок на шляху з дуже хорошим узагальненням на невідомі топології мереж.

У дослідженні абляції підкреслено, що простіший підхід (A2) насправді може дати кращі результати, ніж еталонна архітектура (A1) на оціночному наборі даних. Основна відмінність між двома підходами полягає в тому, що A2 не моделює залежності між мережевими чергами на одному вузлі в явному вигляді. Хоча ці залежності насправді не були потрібні в даних моделювання, які було використано для оцінки, очікується, що A1 буде працювати краще на більш реалістичних розгортаннях зі спільним використанням ресурсів між мережевими чергами. Більше того, хоча в цій конкретній проблемі є ще багато невідомих, слід зауважити, що рівні передачі повідомлень все ще можна покращити, щоб обчислювати вбудовування більш безпосередньо, тобто уникаючи проміжних етапів передачі повідомлень, як в представленій архітектурі. Тим не менш, слід зазначити, що результати дослідження можуть бути застосовані для вирішення широкого спектру проблем мережевого моделювання.

Прогнозування затримок трафіку є лише прикладом того, що можна передбачити за частку обчислювального часу, необхідного мережевим симуляторам: представлені моделі приблизно потребують 500 мс для обчислення затримок по всій топології, тоді як симулятори потребують десятки хвилин. Таким чином, запропоновані в цій роботі методи можуть бути інтегровані в мережеві

симулятори (і емулятори) для більш економних обчислень, але ціною певної втрати точності прогнозування.

Аналіз також висвітлює чіткі переваги досліджень абляції. При розробці алгоритмів машинного навчання часто додають більше компонентів, ніж насправді потрібно: вилучення одного внеску за раз може призвести до несподіваних висновків. Звичайно, ці дослідження можуть показати результати лише на конкретному наборі даних: не існує «найкращої» архітектури нейронної мережі, яку можна пристосувати до кожної проблеми.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ITU-D. « Measuring digital development - Facts and figures ». In: ITU Publications (2022).
2. Cisco. Cisco Annual Internet Report (2018-2023). Tech. rep. 2020. doi: 10 . 1016 / s1361 -3723(20)30026-9.
3. Timm Böttger, Felix Cuadrado, Gareth Tyson, Ignacio Castro, and Steve Uhlig. « Open connect everywhere: A glimpse at the internet ecosystem through the lens of the net-ix CDN». In: SIGCOMM CCR 48.1 (2018), pp. 28–34. doi: 10.1145/3211852.3211857.
4. ARCEP. L'état d'internet en France. Tech. rep. 2023.
5. John Graham-Cumming. Cloud-are outage on July 17, 2024. 2024. url: https : / / blog. cloudflare.com/cloudflare-outage-on-july-17-2020/ (visited on 03/31/2021).
6. CAIDA. ASRank. 2021. url: https://asrank.caida.org/ (visited on 03/31/2021).
7. Angelique Medina. CenturyLink/Level3 Outage Analysis. 2020. url: https://blog.thousandeyes.com/centurylink-level-3-outage-analysis/ (visited on 03/31/2021).
8. Kevin Vermeulen. « lmproved algorithms for capturing Internet maps ». PhD thesis. 2020.
9. BBC News. Google achieves AI 'breakthrough' by beating Go champion. 2016. url: https://www.bbc.com/news/technology-35420579 (visited on 03/31/2021).
10. Albert Mestres, Alberto Rodriguez-Natal, Josep Carner, Pere Barlet-Ros, Eduard Alarcn, Marc Sol, Victor Munts-Mulero, David Meyer, Sharon Barkai, Mike J Hibbett, Giovani Estrada, Khaldun Ma 'ruf, Florin Coras, Vina Ermagan, Hugo Latapie, Chris Cassar, John Evans, Fabio Maino, Jean Walrand, Albert Cabellos, Fernando Ramos, Eduard Alarcón, Marc Solé, and Victor MuntésMulero. « Knowledge-Dened Networking ». In: SIGCOMM CCR 47.3 (2017), pp. 2–10.
11. Krzysztof Rusek, José Suárez-Varela, Albert Mestres, Pere Barlet-Ros, and Albert CabellosAparicio. « Unveiling the potential of Graph Neural Networks for network modeling and optimization in SDN ». In: SOSR. Jan. 2019. isbn: 978-1-4503-6710-3. doi: 10.1145/3314148.3314357.
12. ITU. ITU AI/ML in 5G Challenge. 2020. url: https://www.itu.int/en/ITU- T/AI/ challenge/2020/Pages/default.aspx (visited on 02/15/2021).
13. Bruce Temkin. 2018 Temkin Experience Ratins, U.S. Tech. rep. 2018.
14. Hiver. Customer Support Through The Eyes of Consumers in 2020. 2020.153
15. Diana Zeaiter Joumblatt, Jaideep Chandrashekar, Branislav Kevton, and Renata Teixeira. « Predicting User Dissatisfaction with Internet Application Performance at End-Hosts ». In: INFOCOM. 2013.
16. Hyunwoo Nam, Kyung Hwa Kim, and Henning Schulzrinne. « QoE matters more than QoS: Why people stop watching cat videos ». In: INFOCOM. 2016. isbn: 978-1-4673-9953-1. doi: 10.1109/INFOCOM.2016.7524426.

17. Andrew S Tanenbaum and David J. Wetherall. Computer Networks. 5th. Computer Networks. Prentice Hall PTR, 2010. isbn: 978-0-13-066102-9.
18. Vangie Beal. WiFi Denition & Meaning. 2021. url: https : / / www . webopedia . com /definitions/wifi/ (visited on 04/02/2021).
19. John Hawkinson and Tony Bates. RFC 1930: Guidelines for creation, selection, and registration of an Autonomous System (AS). 1996.
20. Ramon Puigjaner. « Performance Modelling of Computer Networks ». In: LANC. 2003. isbn: 1-58113-789-3.
21. Gabriel A. Wainer. Discrete-event modeling and simulation: a practitioner's approach. 2009. isbn: 978-1-4200-5336-4.
22. Francisco J. Suárez, Pelayo Nuño, Juan C. Granda, and Daniel F. García. « Computer networks performance modeling and simulation ». In: Modeling and Simulation of Computer Networks and Systems: Methodologies and Applications. Ed. by Mohammad S. Obaidat Zarai, Petros Nicopolitidis, and Faouzi. Morgan Kaufmann, 2015, pp. 187–223. isbn: 978-0-12-801158-4. doi: 10. 1016/B978-0-12-800887-4.00007-9.
23. Robert B Cooper. Queuing Theory. 2nd. 1981.
24. Chee-Hock Ng and Boon-Hee Soong. Queueing modelling fundamentals: With applications in communication networks. 2nd. 2008.
25. B. Filipowicz and J. Kwiecień. « Queueing systems and networks. Models and applications ». In: Bulletin of the Polish Academy of Sciences: Technical Sciences 56.4 (2008), pp. 379–390.
26. Leonard Kleinrock. Queueing systems, volume 2: Computer applications. 1976.
27. Anthony Ephremides. « Limitations of Queueing Models in Communication Networks ». In: Performance Limits in Communication Theory and Practice. 1988, pp. 143–153.
28. Martin Reiser. « A Queueing Network Analysis of Computer Communication Networks with Window Flow Control ». In: IEEE Transactions on Communications 27.8 (1979), pp. 1199–1209. doi: 10.1109/TCOM.1979.1094531.154
29. Anthony Ephremides and Bruce Hajek. « Information theory and communication networks: An unconsummated union ». In: IEEE Transactions on Information Theory 44.6 (1998), pp. 2416–2434. doi: 10.1109/18.720543.
30. Carolina Osorio and Michel Bierlaire. « An analytic nite capacity queueing network model capturing the propagation of congestion and blocking ». In: European Journal of Operational Research 196.3 (2009), pp. 996–1007. doi: 10.1016/j.ejor.2008.04.035.
31. Anthony Ebert, Paul Wu, Kerrie Mengersen, and Fabrizio Ruggeri. « Computationally ecient simulation of queues: The r package queuecomputer ». In: CoRR (2019). doi: 10.18637/jss.v095.i05. url: https://arxiv.org/abs/1703.02151.
32. Carl Adam Petri and Wolfgang Reisig. « Petri net ». In: Scholarpedia 3.4 (2008), p. 6477.
33. Jacium Wang. Timed petri nets: Theory and application. Springer, 1998. isbn: 978-1-4615-5537-7.

34. Jonathan Billington and Michel Diaz, eds. Application of Petri Nets to Communication Networks Springer, 1999. isbn: 978-3-540-65870-2.

35. F. Bause. « Queueing Petri Nets a formalism for the combined qualitative and quantitative analysis of systems ». In: PNPM. 1993. isbn: 0-8186-4250-5. doi: 10 . 1109 / PNPM . 1993. 393439.

36. E. Rodríguez, J. L. Arqués, R. Rodríguez, M. Nuñez, M. Medina, T. L. Talarico, I. A. Casas, T. C.

37. Chung, W. J. Dobrogosz, L. Axelsson, S. E. Lindgren, W. J. Dobrogosz, Leila Kerkeni, Paula Ruano, Lismet Lazo Delgado, Sergio Picco, Liliana Villegas, Franco Tonelli, Mario Merlo, Javier Rigau, Dario Diaz, and Martin Masuelli. « On the Use of Queueing Petri Nets for Modeling and Performance Analysis of Distributed Systems ». In: Petri Net, Theory and Applications. Ed. By Vedran Kordic. Intech Open, 2008, pp. 534–566. isbn: 978-3-902613-12-7.

38. Jerry Banks, John Carson II, Barry Nelson, and David Nicol. Discrete-Event System Simulation. 5th. Pearson, 2010. isbn: 978-0-13-606212-7.

39. University of Washington NS-3 Consortium. ns-3 Network Simulator. 2021. url: https://www.nsnam.org (visited on 01/22/2021).

40. András Varga and Rudolf Hornig. « An Overview of the OMNeT++ Simulation Environment ». In: SIMUTools. 2008. isbn: 978-963-9799-20-2.

41. Sally Floyd and Eddie Kohler. « Internet research needs better models ». In: SIGCOMM CCR 33.1 (2003), pp. 29–34. doi: 10.1145/774763.774767.

42. Bob Lantz, Brandon Heller, and Nick McKeown. « A network in a laptop ». In: HotNets. 2010. isbn: 978-1-4503-0409-2. doi: 10.1145/1868447.1868466.155

43. Paulo Gouveia, João Neves, Carlos Segarra, Luca Liechti, Shady Issa, Valerio Schiavoni, and Miguel Matos. « Kollaps: Decentralized and Dynamic Topology Emulation ». In: EuroSys. 2020.

44. InterDigital. AdvantEdge mobile edge emulation platform. 2021. url: https://github.com/ InterDigitalInc/AdvantEDGE (visited on 01/22/2021).

45. The Kubernetes Authors. Kubernetes. 2021. url: https : / / kubernetes . io (visited on 01/22/2021).

Додаток А

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра комп’ютерної інженерії

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття освітнього ступеня магістра

НА ТЕМУ: «МОДЕЛЮВАННЯ КОМП’ЮТЕРНОЇ МЕРЕЖІ НА ОСНОВІ ГРАФОВИХ НЕЙРОННИХ МЕРЕЖ»

Виконав студент групи КСДМ-61
Присяжнюк Владислав Дмитрович
Керівник дипломної роботи:
Торошанко Ярослав Іванович
канд.техн.наук, доцент

Київ – 2025

МЕТА, ОБ’ЄКТА ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи - підвищення ефективності обробки даних, оптимізація мережевих ресурсів та покращення якості послуг, які надаються користувачам

Об’єкт дослідження - функціонування та способи моделювання комп'ютерних мереж з використанням графових нейронних мереж

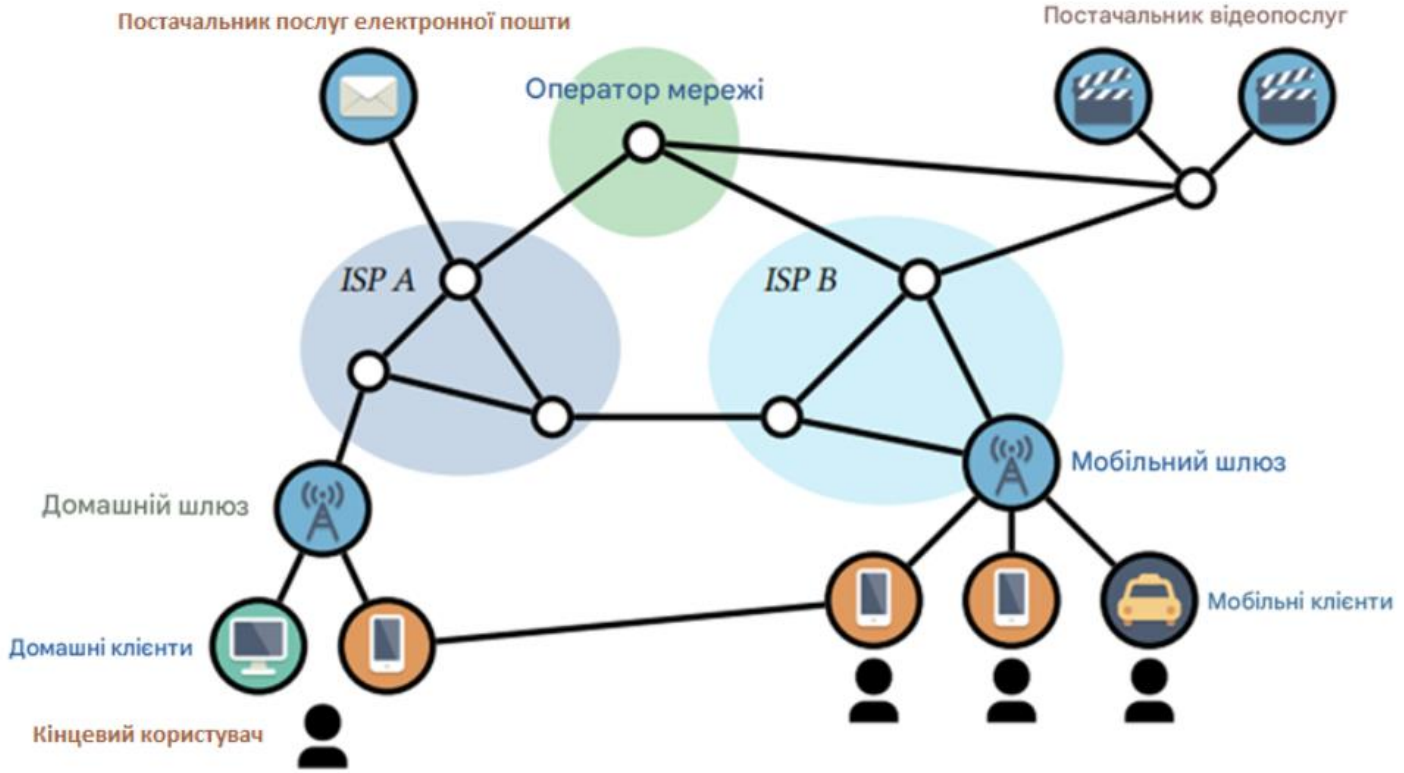
Предмет дослідження - алгоритми та методи, які застосовуються для побудови і навчання графових нейронних мереж

Актуальність роботи

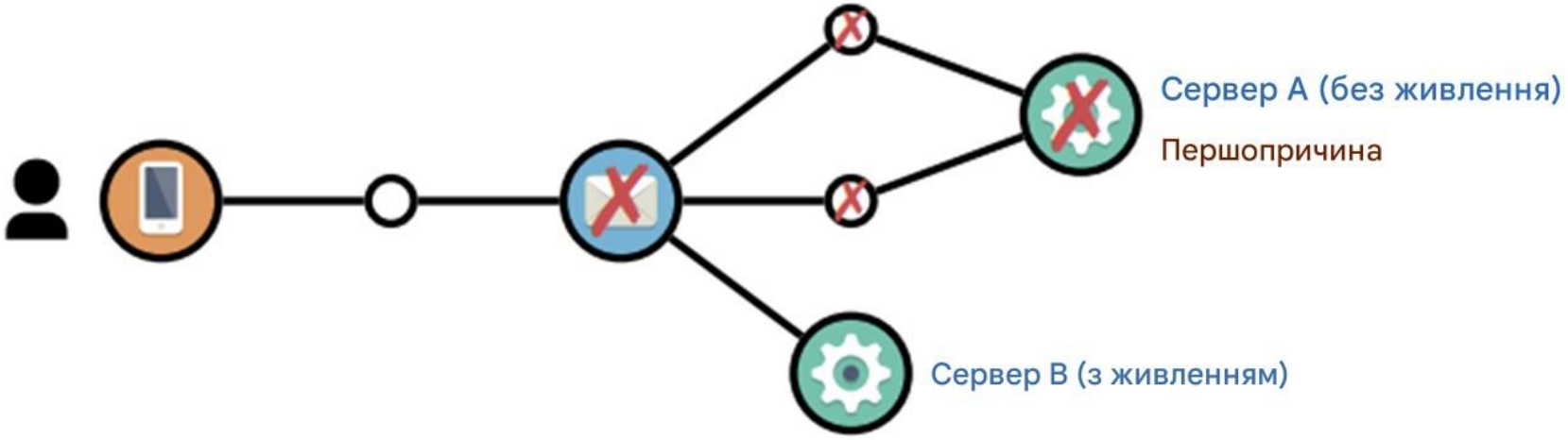
Актуальність даної роботи зумовлена стрімким розвитком технологій, пов'язаних з комп'ютерними мережами, а також зростанням обсягу даних, які потребують ефективної обробки. Використання графових нейронних мереж для моделювання мережевих структур дозволяє більш точно відображати взаємозв'язки між елементами мережі і знаходити оптимальні рішення для складних задач.

Крім того, з огляду на зростаючу потребу в забезпеченні безпеки та надійності комп'ютерних мереж, дослідження нових підходів до їх моделювання є надзвичайно важливим для науки і практики в галузі інформаційних технологій.

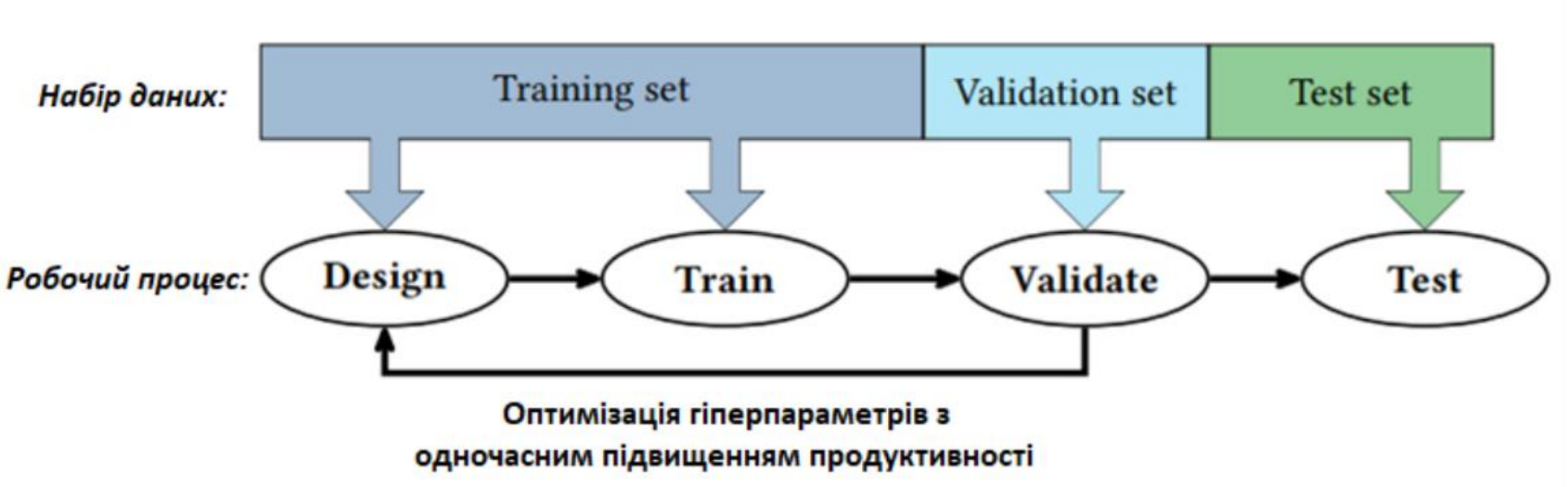
Ілюстрація ролей в імітаційній комп'ютерній мережі, подібній до Інтернету



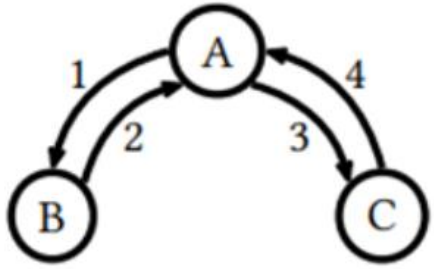
Приклад аналізу першопричини (RCA)



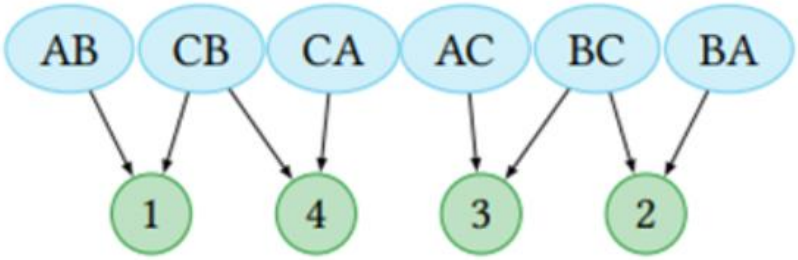
Загальний вигляд ітеративного процесу машинного навчання



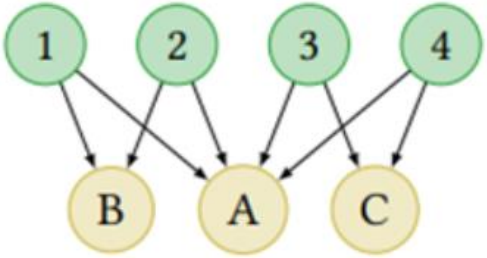
Приклад базової топології, що веде до трьох представлених дводольних графів



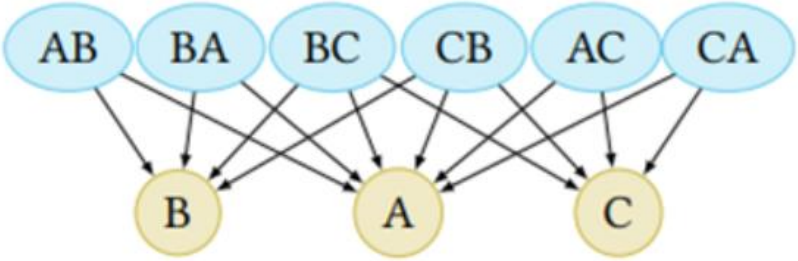
(a) Оригінальна топологія



(b) Дводольний граф $G_{p,l}$



(c) Дводольний граф $G_{l,n}$



(d) Дводольний граф $G_{p,n}$

Algorithm 1: High-level pseudo-code of proposed GNN approach A1

Input: Features $\mathbf{F}_p, \mathbf{F}_l, \mathbf{F}_n$; neighbors $\mathcal{N}$ and $\tilde{\mathcal{N}}$; trainable functions ϕ, γ and FC.
Output: Readout performances of every path

▷ *State initialization, padding with zeroes*

$\forall i \in \mathbf{p} : \mathbf{p}_i \leftarrow [\mathbf{F}_{p,i}, 0, \ldots, 0]$

$\forall i \in \mathbf{l} : \mathbf{l}_i \leftarrow [\mathbf{F}_{l,i}, 0, \ldots, 0]$

$\forall i \in \mathbf{n} : \mathbf{n}_i \leftarrow [\mathbf{F}_{n,i}, 0, \ldots, 0]$

for $t \leftarrow 1$ **to** T **do**

▷ *Update link embeddings from path embeddings*

$\forall i \in \mathbf{l} : \mathbf{l}_i \leftarrow \gamma_1^t \left(\mathbf{l}_i, \mathrm{Agg}_{j \in \mathcal{N}_{p,l}(i)} \, \phi_1^t \left(\mathbf{p}_j \right) \right)$

▷ *Update node embeddings from link embeddings*

$\forall i \in \mathbf{n} : \mathbf{n}_i \leftarrow \gamma_2^t \left(\mathbf{n}_i, \mathrm{Agg}_{j \in \mathcal{N}_{l,n}(i)} \, \phi_2^t \left(\mathbf{l}_j \right) \right)$

▷ *Update link embeddings from node embeddings ("backward operation")*

$\forall i \in \mathbf{l} : \mathbf{l}_i \leftarrow \gamma_3^t \left(\mathbf{l}_i, \mathrm{Agg}_{j \in \tilde{\mathcal{N}}_{l,n}(i)} \, \phi_3^t \left(\mathbf{n}_j \right) \right)$

▷ *Update path embeddings from link embeddings, two aggregations*

$\forall i \in \mathbf{p} : \mathbf{p}'_i \leftarrow \gamma_4^t \left(\mathbf{p}_i, \mathrm{Agg}_{j \in \tilde{\mathcal{N}}_{p,l}(i)} \, \phi_4^t \left(\mathbf{l}_j \right) \right), \mathbf{p}''_i \leftarrow \gamma_5^t \left(\mathbf{p}_i, \mathrm{OrdAgg}^t_{j \in \tilde{\mathcal{N}}_{p,l}(i)} \, \phi_5^t \left(\mathbf{l}_j \right) \right)$

$\mathbf{p} \leftarrow FC_0 \left(\mathbf{p}' \parallel \mathbf{p}'' \right)$

▷ *Bonus message passing from node to path embeddings*

$\forall i \in \mathbf{p} : \mathbf{p}'_i \leftarrow \gamma_6^{T+1} \left(\mathbf{p}_i, \mathrm{OrdAgg}^{T+1}_{j \in \tilde{\mathcal{N}}_{p,n}(i)} \, \phi_6^{T+1} \left(\mathbf{n}_j \right) \right)$

▷ *Start readout from path embeddings and include features back*

$\mathbf{r}_0 \leftarrow \left(\mathbf{p} \parallel \mathbf{p}' \parallel \mathbf{F}_p \right)$

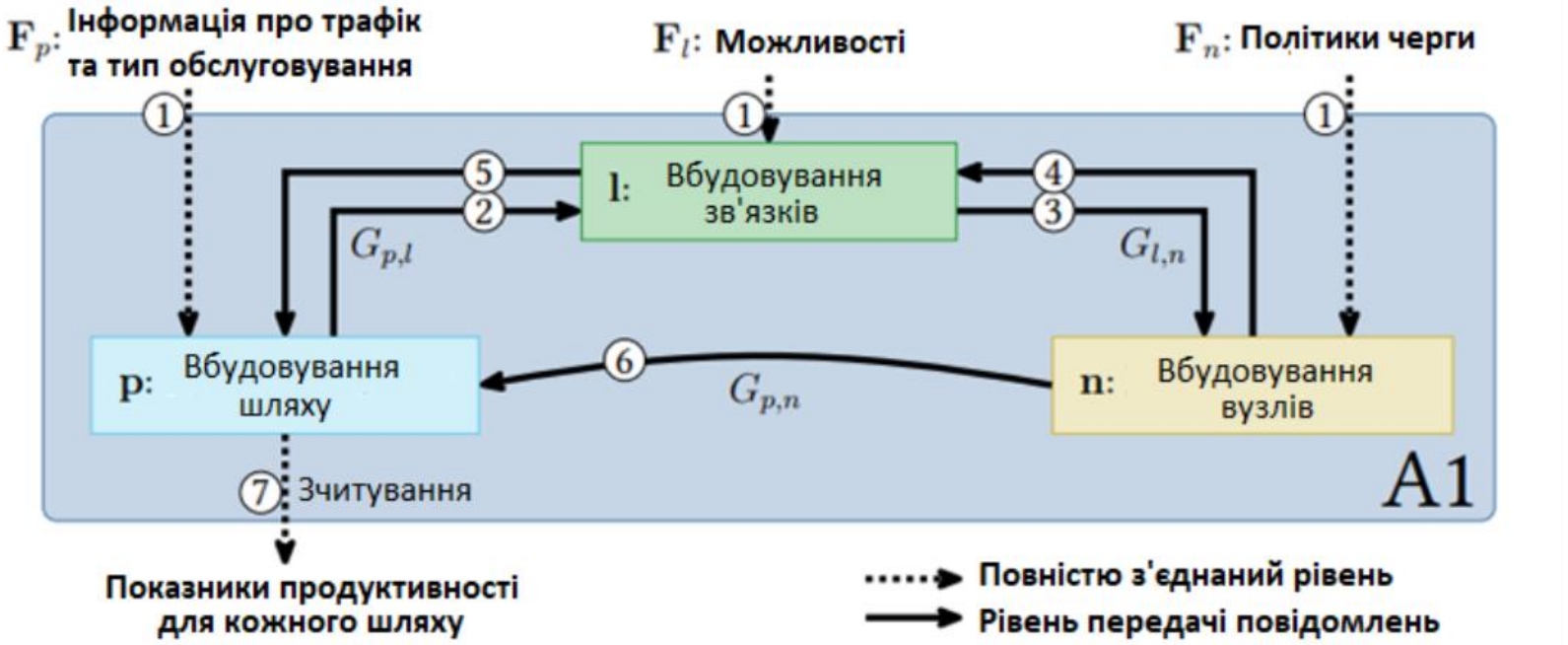
$\mathbf{r}_1 \leftarrow \mathrm{Activation} \left(FC_1 \left(\mathbf{r}_0 \right) \right)$

$\mathbf{r}_2 \leftarrow \mathrm{Activation} \left(FC_2 \left(\mathbf{r}_1 \right) \right)$

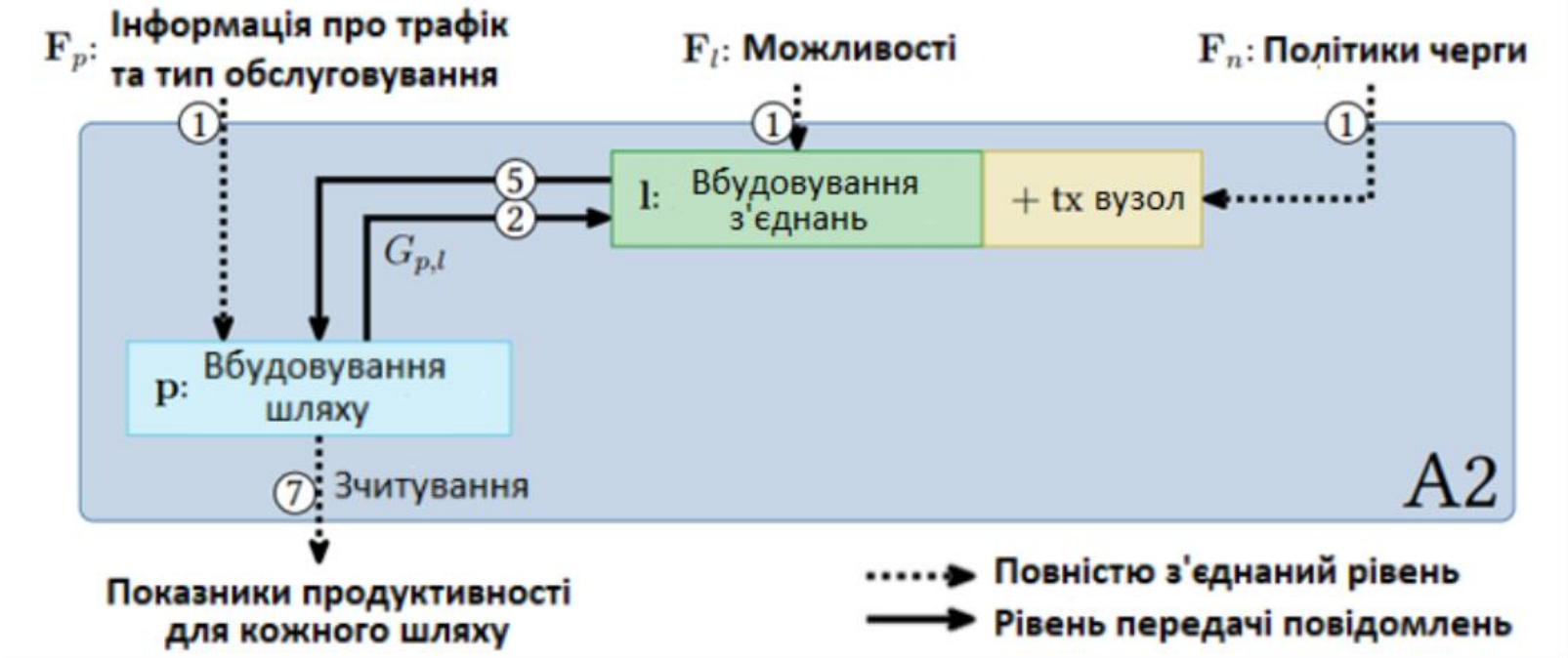
$\mathbf{r}_3 \leftarrow \mathrm{Activation} \left(FC_3 \left(\mathbf{r}_2 \right) \right)$

return $FC_4 \left(\mathbf{r}_3 \right)$

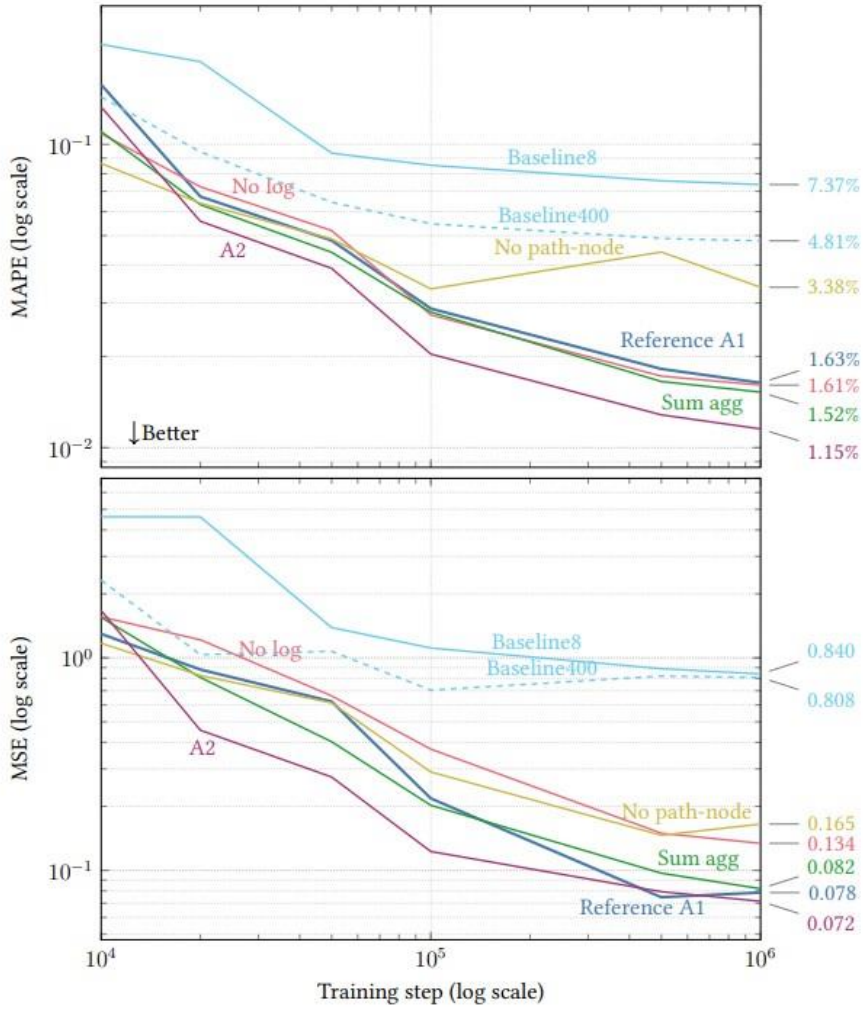
Візуальне представлення підходу A1



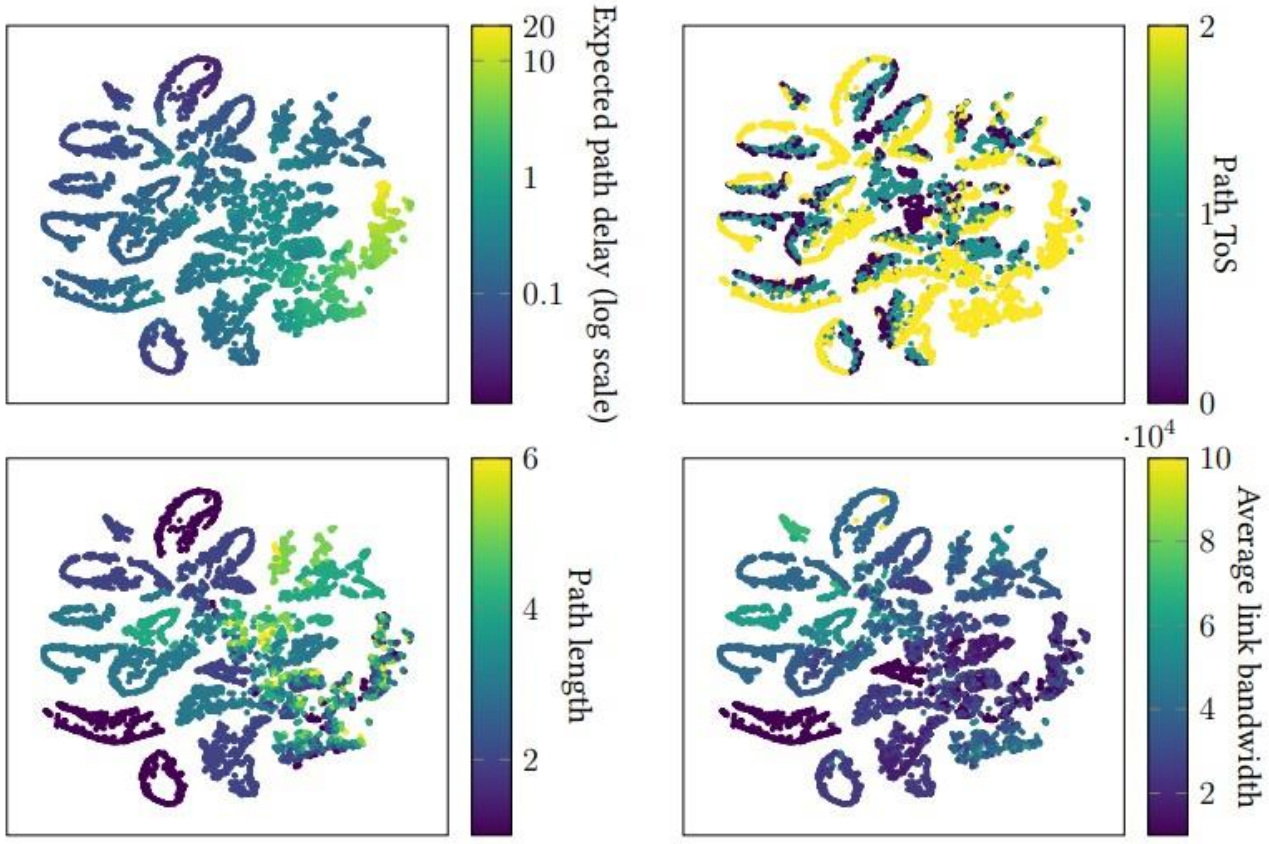
Візуальне представлення підходу A2



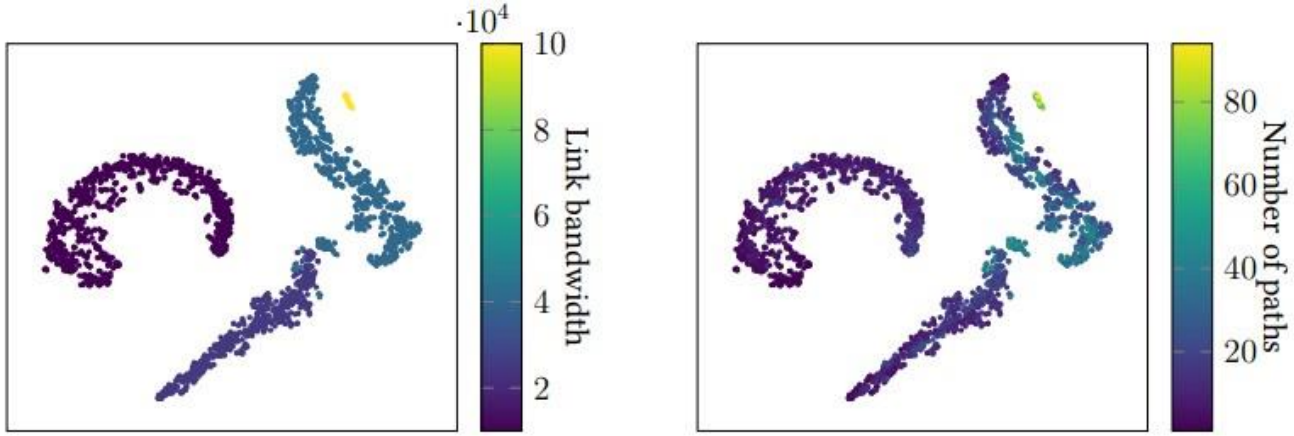
Порівняння ефективності варіантів абляції при збільшенні навчальної вибірки



2-компонентна t-SNE-візуалізація вкладених маршрутів



Візуалізація t-SNE вбудованих з’єднань



Висновки

У цій роботі представлено та оцінено передові методи моделювання комп'ютерних мереж за допомогою графових нейронних мереж. Зокрема, запропоновано нову архітектуру моделі (A1), яка використовує кілька двосторонніх графів у шарах передачі повідомлень для моделювання взаємозалежностей компонентів у знаннях комп'ютерної мережі. Після оптимізації гіперпараметрів було отримано модель машинного навчання, пристосованого для завдання прогнозування затримок на шляху з дуже хорошим узагальненням на невідомі топології мереж.

Прогнозування затримок трафіку є лише прикладом того, що можна передбачити за частку обчислювального часу, необхідного мережевим симуляторам: представлені моделі приблизно потребують 500 мс для обчислення затримок по всій топології, тоді як симулятори потребують десятки хвилин. Таким чином, запропоновані в цій роботі методи можуть бути інтегровані в мережеві симулятори (і емулятори) для більш економних обчислень, але ціною певної втрати точності прогнозування.