МАГІСТЕРСЬКА РОБОТА

МР. ІПм - 22.00.00.000 ПЗ

Група ІПм-23-3

Сенишин Артем

2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Сенишин Артем Юрійович

(прізвище, ім'я, по батькові)

УДК 004.942

(індекс)

МАГІСТЕРСЬКА РОБОТА

Моделі, методи та засоби масштабування розподіленого навчання графових нейронних мереж

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Сенишин А.Ю.

(підпис, ініціали та прізвище здобувача освітнього ступеня)

Науковий керівник Піх Володимир Ярославович, к.т.н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту

Завідувач кафедри

доц. Бандура В.В.

(посада) (підпис) (дата) (ініціали та прізвище)

Нормоконтроль

доц. Вовк Р.Б.

(посада) (підпис) (дата) (ініціали та прізвище)

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

Івано-Франківськ – 2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Освітній рівень магістр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**
доц. В.В. Бандура
“ 04 ” вересня 2024 р.

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Сенишин Артем Юрійович

(прізвище, ім’я, по-батькові)

1. Тема магістерської роботи “ Моделі, методи та засоби масштабування розподіленого навчання графових нейронних мереж ”

керівник проекту (роботи) Піх Володимир Ярославович, к.т.н., доцент

затверджені наказом закладу вищої освіти від “ 22 ” листопада 2024 р. № 781/7

2. Строк подання студентом проекту (роботи) 15 грудня 2024 р.

3. Вихідні дані до проекту (роботи) Теоретичні концепції та формальні моделі побудови та функціонування інформаційних технологій побудови нейронних мережі

4. Зміст розрахунково - пояснювальної записки(перелік питань, які потрібно розробити)

1. Дослідження основних аспектів аналізу даних на основі графічних нейронних мереж
2. Дослідження методів та алгоритмів графових нейронних мереж для розподіленого навчання
3. Підхід до розподіленого навчання нейронних мереж на основі розподіленої хеш-таблиці
4. Імплементація моделей та методів масштабування навчання графових нейронних мереж

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1. Представлення деяких типових графів (рис. 1.1)
2. Експоненціальне зростання популярних графових наборів даних з часом (рис. 1.2)
3 Ілюстрація процесу розбиття графа (рис. 1.4)
4. Федеративний процес навчання з вузлом, який підтримує глобальну модель (рис. 1.5)
5. Результати тесту Geekbench ML для різних пристроїв (рис. 1.6)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата
Перевірка на плагіат	доц., к.т.н. Вовк Р.Б.	

7. Дата видачі завдання 04 вересня 2024 р.

Керівник _____
(підпис)

Завдання прийняв до виконання_____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір і вивчення літератури по темі магістерської роботи	15.09.2024	виконано
2	Аналіз концепцій та алгоритмів предметної області	29.09.2024	виконано
3	Дослідження основних аспектів аналізу даних на основі графічних нейронних мереж	15.10.2024	виконано
4	Дослідження методів та алгоритмів графових нейронних мереж для розподіленого навчання	08.11.2024	виконано
5	Підхід до розподіленого навчання нейронних мереж на основі розподіленої хеш-таблиці	20.11.2024	виконано
6	Імплементація моделей та методів масштабування навчання графових нейронних мереж	01.12.2024	виконано
7	Затвердження пояснювальної записки роботи завідувачем кафедри	15.12.2024	виконано

Студент – магістр _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Магістерська робота: 82 с., 21 рис., 52 джерела.

Тема: Моделі, методи та засоби масштабування розподіленого навчання графових нейронних мереж

Об'єкт дослідження: процеси розподіленого навчання графових нейронних мереж у децентралізованих системах.

Мета роботи: розробка інноваційних методів і алгоритмів для масштабованого розподіленого навчання графових нейронних мереж на малопотужних пристроях із використанням розподіленої хеш-таблиці (DHT), що забезпечить ефективне використання ресурсів та збереження конфіденційності даних.

Предмет дослідження: методи і алгоритми оптимізації розподіленого навчання графових нейронних мереж у рівноправних мережах з використанням розподіленої хеш-таблиці.

Результати дослідження

В роботі запропоновано адаптивний метод балансування навантаження (ALBP) у системі GraphDHT для розподіленого навчання графових нейронних мереж на малопотужних пристроях.

Висновок

Розроблено алгоритм зваженого розбиття графів і агрегації моделей, який покращує ефективність навчання та знижує обчислювальні вимоги до пристроїв та представлено технологію розподіленого навчання з використанням DHT для GNN.

ГРАФОВІ НЕЙРОННІ МЕРЕЖІ (GNN), РОЗПОДІЛЕНЕ НАВЧАННЯ, АДАПТИВНЕ БАЛАНСУВАННЯ НАВАНТАЖЕННЯ (ALBP), РОЗПОДІЛЕНА ХЕШ-ТАБЛИЦЯ (DHT), РІВНОПРАВНІ МЕРЕЖІ (P2P), ДЕЦЕНТРАЛІЗОВАНІ СИСТЕМИ.

ABSTRACT

Master Thesis: 82 pp., 21 fig., 52 sources.

Thesis Subject: Topic: Models, methods and means of scaling distributed learning of graph neural networks

Research subject: processes of distributed learning of graph neural networks in decentralized systems.

The purpose of the work: development of innovative methods and algorithms for scalable distributed training of graph neural networks on low-power devices using a distributed hash table (DHT), which ensures efficient use of resources and preservation of data confidentiality.

Research subject: optimization methods and algorithms for distributed learning of graph neural networks in peer-to-peer networks using a distributed hash table.

Research results

In the paper, we propose an adaptive load balancing method (ALBP) in the GraphDHT system for distributed training of graph neural networks on low-power devices.

Conclusion

A weighted graph partitioning and model aggregation algorithm is developed, which increases the learning efficiency and reduces the computing requirements on devices, and a distributed learning technology using DHT for GNN is presented.

GRAPH NEURAL NETWORKS (GNN), DISTRIBUTED LEARNING, ADAPTIVE LOAD BALANCING (ALBP), DISTRIBUTED HASH TABLE (DHT), PEER-TO-PEER NETWORKS (P2P), DECENTRALIZED SYSTEMS.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

GNN - Graph Neural Network

GCN - Graph Convolutional Network

RGCN - Relational Graph Convolutional Network

VGAE - Variational Graph Auto-Encoder

GAT - Graph Attention Network

SGC - Simplified Graph Convolution

PPI - Protein-Protein Interaction (network)

OAG - Open Academic Graph

NeurIPS - Neural Information Processing Systems

ICLR - International Conference on Learning Representations

KDD - Knowledge Discovery and Data Mining

MLP - Multi-Layer Perceptron

DGL - Deep Graph Library

DHT - Distributed Hash Table

PyG - PyTorch Geometric

HPC - High-Performance Computing

GPU - Graphics Processing Unit

TPU - Tensor Processing Unit

SGD - Stochastic Gradient Descent

MPI - Message Passing Interface

OCL - Object Constraint Language

VAE - Variational Auto-Encoder

FL - Federated Learning

RL - Reinforcement Learning

AI - Artificial Intelligence

ВСТУП

Актуальність теми.

У сучасних умовах стрімкого зростання обсягів даних та ускладнення їхньої структури, розподілене навчання графових нейронних мереж (GNN) стає важливим інструментом для аналізу даних у формі графів. Такі мережі знаходять застосування в різних галузях, включаючи соціальні мережі, обробку природної мови, біоінформатику, хімію, фінансові операції та телекомунікації. Графові дані мають специфічну природу, де взаємозв'язки між об'єктами так само важливі, як і самі об'єкти. Однак, навчання на таких даних потребує значних обчислювальних ресурсів і складних алгоритмів, що ускладнює його реалізацію на пристроях з обмеженими можливостями, таких як смартфони або інші пристрої периферії.

З розвитком технологій Інтернету речей (IoT) та мереж периферійних обчислень виникає потреба в розподілених системах машинного навчання, які можуть ефективно працювати на кінцевих пристроях. При цьому одним із ключових викликів є забезпечення конфіденційності даних. У централізованих системах навчання передбачає передавання великих обсягів даних на центральний сервер, що може призводити до витоку або порушення конфіденційності чутливої інформації. Тому виникає необхідність у децентралізованих підходах, таких як федеративне навчання, де навчальні дані залишаються на пристроях користувачів, а моделі обмінюються лише зведеними результатами.

Розподілене навчання графових нейронних мереж на малопотужних пристроях стикається з додатковими проблемами, такими як адаптація навчання до обмежених ресурсів (обчислювальна потужність, енергоспоживання, пам'ять) і забезпечення надійного збереження конфіденційності даних. У цьому контексті актуальною стає розробка нових підходів до адаптивного балансування навантаження та оптимізації процесу навчання GNN у децентралізованих мережах.

Запропоновані у дослідженні методи, що базуються на адаптивному балансуванні навантаження та використанні технологій розподілених хеш-таблиць (DHT), дозволяють ефективно розподіляти обчислювальні завдання між кінцевими пристроями, враховуючи їхні можливості. Це особливо важливо в умовах зростаючих обсягів даних та необхідності обробки графів на малопотужних пристроях. Крім того, система на основі рівноправної мережі (P2P) дозволяє зменшити централізовані обчислювальні витрати та підвищити рівень безпеки.

Отже, актуальність даного дослідження полягає у вирішенні важливих сучасних проблем: зниження обчислювальних витрат у графових нейронних мережах, підвищення ефективності навчання на кінцевих пристроях, збереження конфіденційності даних та оптимізація процесу розподіленого навчання у децентралізованих середовищах.

Мета дослідження - розробка інноваційних методів і алгоритмів для масштабованого розподіленого навчання графових нейронних мереж на малопотужних пристроях із використанням розподіленої хеш-таблиці (DHT), що забезпечить ефективне використання ресурсів та збереження конфіденційності даних.

Об’єкт дослідження - процеси розподіленого навчання графових нейронних мереж у децентралізованих системах.

Предмет дослідження - методи і алгоритми оптимізації розподіленого навчання графових нейронних мереж у рівноправних мережах з використанням розподіленої хеш-таблиці.

Відповідно до мети роботи було сформовано наступні **задачі:**

- Проаналізувати існуючі методи розподілу даних і навчання графових нейронних мереж.
- Розробити адаптивний метод балансування навантаження (ALBP) для ефективного навчання GNN у децентралізованих мережах.
- Вивчити алгоритми розподілу графів і агрегації моделей для покращення обчислювальної ефективності GNN.

- Розробити механізми збереження конфіденційності даних під час розподіленого навчання у рівноправних мережах.
- Тестувати ефективність запропонованих методів на різних наборах даних і архітектурах GNN.

Методи дослідження.

Для досягнення поставленої мети було використано методи аналізу та синтезу алгоритмів розподілу даних, математичне моделювання процесів розподіленого навчання, а також експериментальне тестування запропонованих алгоритмів на реальних наборах даних і архітектурах графових нейронних мереж.

Наукова новизна отриманих результатів

Розроблено алгоритм зваженого розбиття графів і агрегації моделей, який покращує ефективність навчання та знижує обчислювальні вимоги до пристроїв та представлено технологію розподіленого навчання з використанням DHT для GNN, що забезпечує збереження конфіденційності в рівноправних мережах.

Практичне значення магістерської роботи

Результати дослідження можуть бути використані для впровадження у системи розподіленого навчання на основі графових нейронних мереж, зокрема для оптимізації обчислювальних процесів на кінцевих пристроях, таких як смартфони. Запропоновані методи можуть бути корисними у сферах, де важливе значення має як обробка великих обсягів графових даних, так і збереження конфіденційності, наприклад, в аналізі соціальних мереж, фінансових операціях, тощо.

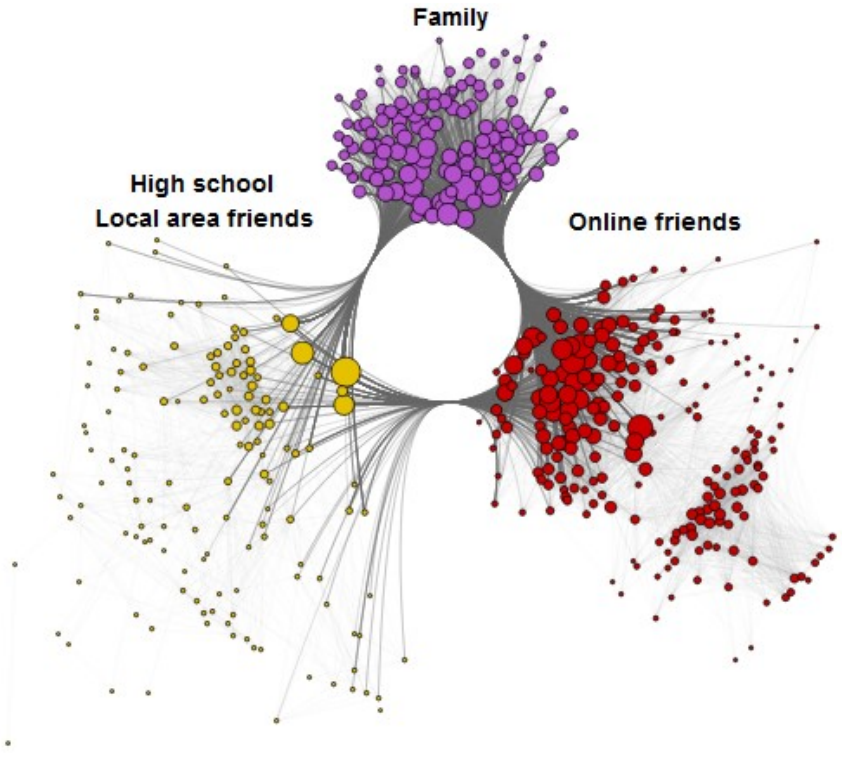
Структура магістерської роботи. Робота складається зі вступу, трьох розділів та висновків. Загальний обсяг роботи становить 82 сторінки, і містить 21 рисунок, перелік використаних джерел із 52 найменувань.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ОСНОВНИХ АСПЕКТІВ АНАЛІЗУ ДАНИХ НА ОСНОВІ ГРАФІЧНИХ НЕЙРОННИХ МЕРЕЖ

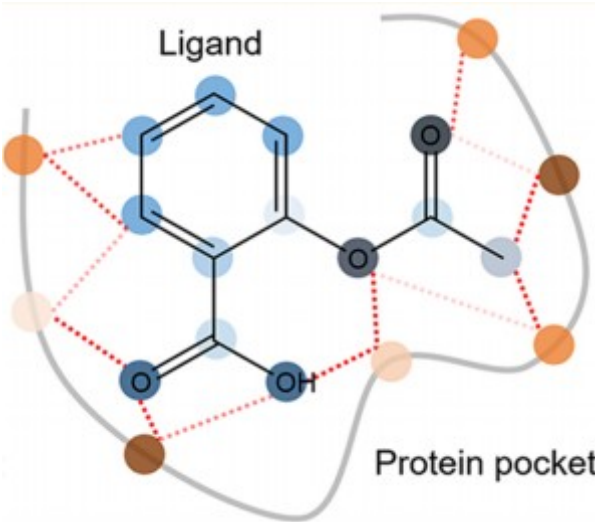
1.1. Основні аспекти функціонування графових нейронних мереж

Графові нейронні мережі (GNN) є типом моделі машинного навчання, що спеціалізується на аналізі даних, структурованих у вигляді мережі, таких як зв'язки в соціальних мережах або біологічних системах. Ці моделі можуть допомогти виявляти закономірності та робити прогнози в різних завданнях, але їх навчання на великомасштабних наборах даних може вимагати значних обчислювальних ресурсів та ретельного поводження з чутливими даними.

Графові нейронні мережі (GNN) — це тип нейронних мереж, які можуть обробляти й аналізувати дані, представлені у формі графіка. Граф — це тип структури даних, що складається з вершин і ребер. Графіки є природним представленням для широкого діапазону даних реального світу, таких як соціальні мережі, веб-графіки та біологічні мережі.



а) Графік соціальної мережі, що показує різні кластери вузлів



б) Граф молекулярної кишені білка

Рис. 1.1. Представлення деяких типових графів

Рисунок 1.1 а, б показує деякі ілюстрації таких графіків у опублікованих роботах [1, 2]. GNN призначені для вивчення складних шаблонів і структур у даних графа, що дозволяє ефективно вирішувати різні завдання, пов’язані з графом, включаючи класифікацію вузлів, прогнозування країв і кластеризацію графів. Рисунок 1.2 показує деякі з цих типових графових завдань, які виконує GNN: класифікація вузла в одній із попередньо визначених категорій, передбачення зв’язку між двома випадковими вузлами, а також міцність цього зв’язку, класифікація цілого нового графа на одна з попередньо визначених категорій, кластеризація підграфів у спільноти та виявлення того, чи є вузол аномальним у певному відношенні порівняно з його сусідами чи іншими вузлами в цьому графі.

GNN працюють, використовуючи механізм передачі повідомлень, який дозволяє вузлам обмінюватися інформацією зі своїми сусідами. Цей процес дозволяє GNN охоплювати як локальні, так і глобальні властивості вхідного графа, що в кінцевому підсумку призводить до насиченого та виразного представлення даних. В останні роки були запропоновані різні архітектури GNN, у тому числі Graph Convolutional Networks (GCNs), GraphSAGE і Graph

Attention Networks (GATs) [8], кожна з яких має свої унікальні переваги та застосування.

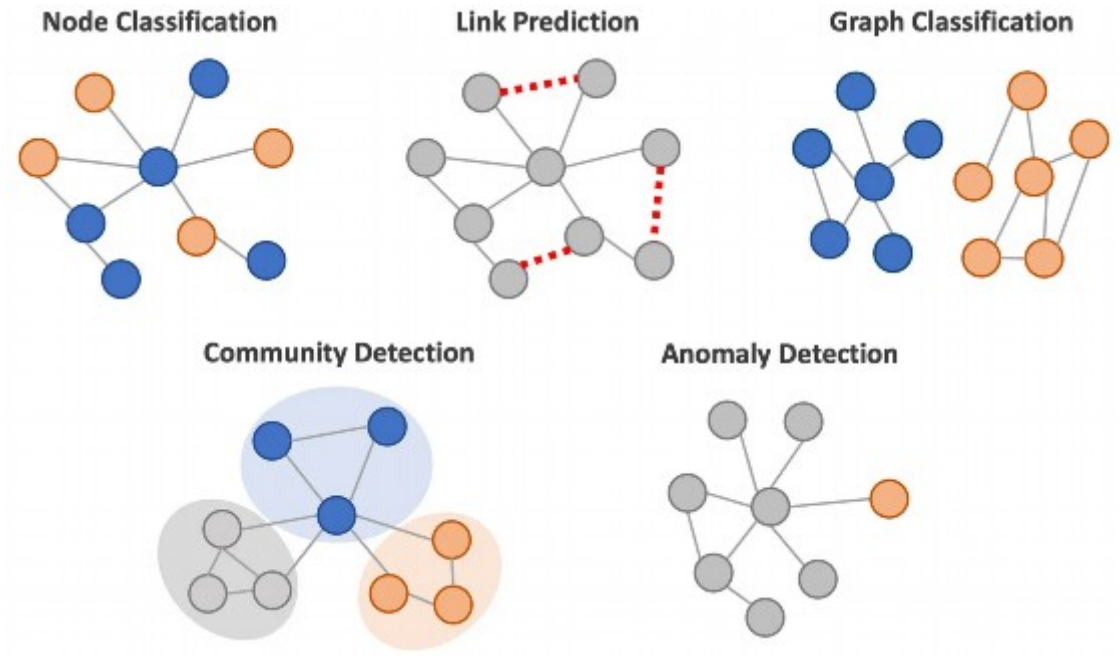


Рис. 1.2. Загальні операції GNN

Незважаючи на вражаючу продуктивність у багатьох завданнях, GNN представляють проблеми з точки зору обчислювальної складності та масштабованості, особливо при роботі з великомасштабними наборами даних графів. Як наслідок, розподілені методи навчання та ефективні методи обробки графів стали важливими для того, щоб GNN застосовувалися до реальних проблем. Це дослідження спрямоване на вирішення цих проблем шляхом розробки нового підходу до розподіленого навчання GNN у одноранговій мережі гетерогенних периферійних пристроїв, використовуючи потужність федеративного навчання та методів поділу графів.

1.1.1. Особливості розподіленого навчання

Збільшення розміру та складності графових наборів даних у поєднанні зі зростаючими вимогами до різноманітних додатків на основі графів посилили проблеми, пов’язані з навчанням GNN. На рисунку 1.3 показано експоненціальне зростання деяких розмірів популярних наборів даних

графіків з часом. У міру того, як масштаб графових даних розширюється, обчислювальні вимоги до навчання GNN збільшуються, перевищуючи потужність окремих машин і вимагаючи використання методів розподіленого навчання. У той час як розподілене навчання дозволяє обробляти більші графіки, воно також створює набір внутрішніх проблем, які необхідно вирішити, щоб забезпечити ефективне та ефективне навчання.

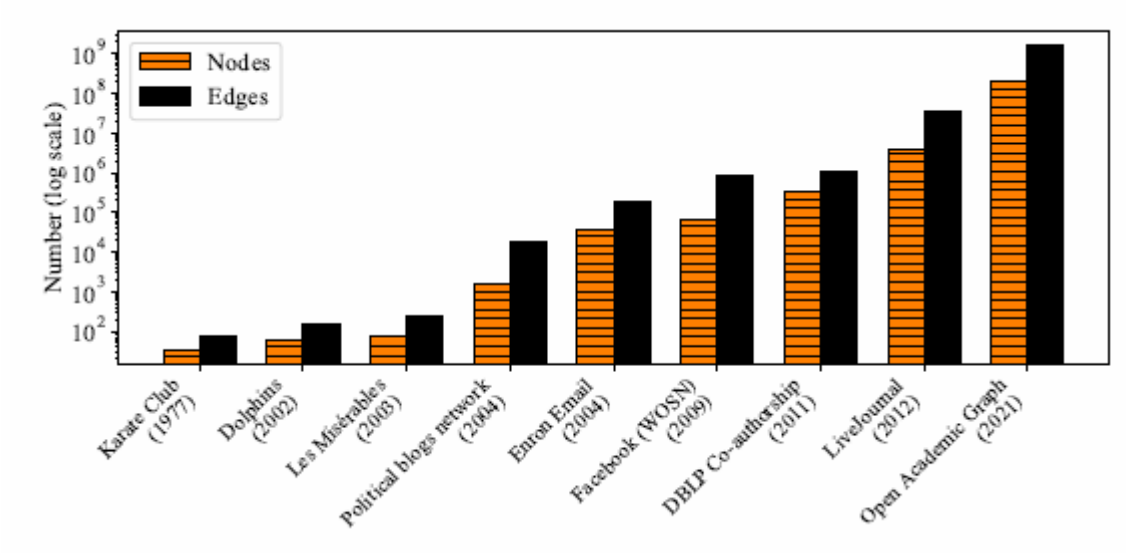


Рис. 1.2. Експоненціальне зростання популярних графових наборів даних з часом

Однією з головних проблем у розподіленому навчанні є необхідність розділити дані графа між декількома пристроями або вузлами, зберігаючи при цьому структурну цілісність вхідного графа. Рисунок 1.4 показує ілюстрацію поділу графа на чотири частини на основі простого алгоритму тісного зв’язку між вузлами [19]. Традиційні методи поділу графів, такі як Random Greedy Partitioning і Recursive Bisection [24], використовувалися для поділу графів на підграфи, але вони часто не враховують неоднорідність апаратних можливостей периферійних пристроїв [31]. Ця неоднорідність може призвести до незбалансованого робочого навантаження та неоптимальної ефективності навчання.

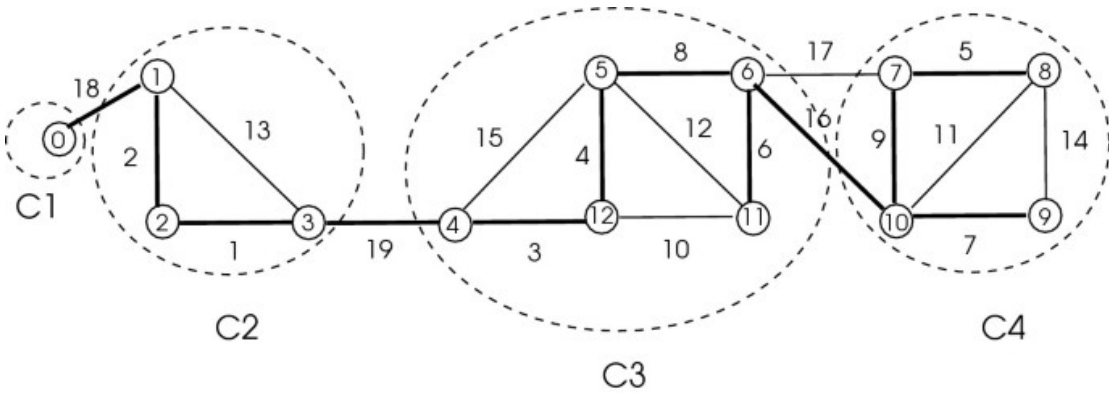


Рис. 1.4. Ілюстрація процесу розбиття графа

Інша проблема розподіленого навчання виникає через необхідність координувати та синхронізувати процес навчання на кількох пристроях. Ця координація має вирішальне значення для забезпечення узгоджених і точних оновлень моделі, але вона може потребувати ресурсів, що призведе до вузьких місць у спілкуванні та збільшення часу навчання. Крім того, проблеми конфіденційності, пов’язані з обміном необробленими даними між пристроями, викликали потребу в методах розподіленого навчання, що зберігають конфіденційність, таких як інтегроване навчання, що додає додатковий рівень складності процесу.

На додаток до обчислювальних проблем і проблем конфіденційності, пов’язаних із розподіленим навчанням GNN, існують значні регуляторні та геополітичні міркування, які слід брати до уваги. Оскільки конфіденційність даних стала глобальною проблемою, кілька країн запровадили суворі правила, що регулюють збір, зберігання та обробку даних своїх громадян. Наприклад, суворі закони про захист даних (наприклад, GDPR [1], PIPL [12] тощо) можуть обмежувати обмін особистою інформацією через національні кордони, що потребує розробки локалізованих моделей машинного навчання, які можна навчити та розгорнути без порушення таких правил. Крім того, геополітична напруженість між країнами може посилити занепокоєння щодо транскордонного обміну даними. У таких сценаріях уряди можуть накладати обмеження на передачу даних між своїми країнами, щоб гарантувати, що

інформація їхніх громадян використовується виключно для місцевих цілей і не передається потенційно ворогуючим націям [4].

У цьому дослідженні ми вирішуємо ці проблеми, розробляючи новий підхід до розподіленого навчання GNN, який використовує однорангову мережу гетерогенних периферійних пристроїв у безпосередній географічній близькості, з'єднаних через розподілену хеш-таблицю. У нашому методі використовується адаптивний алгоритм поділу графів і методи інтегрованого навчання, щоб забезпечити ефективне навчання із збереженням конфіденційності на різноманітних периферійних пристроях.

1.2. Розподілене (федеративне) навчання на кінцевих пристроях

Федеративне навчання - це підхід до машинного навчання, який дозволяє тренувати моделі на великих обсягах даних, розподілених по багатьох пристроях (наприклад, смартфонах, датчиках), без необхідності передавати ці дані на центральний сервер. Кожен пристрій здійснює локальне навчання моделі на своїх даних, а потім надсилає на центральний сервер лише оновлені моделі (а не самі дані). Центральний сервер об'єднує ці оновлення, створюючи покращену глобальну модель, яка потім розповсюджується назад на пристрої.

Федеративне навчання (FL) стало багатообіцяючою технікою для навчання моделям машинного навчання розподіленим способом із збереженням конфіденційності. У FL кілька периферійних пристроїв, таких як смартфони або пристрої IoT, спільно навчають спільну глобальну модель без необхідності обмінюватися вихідними даними. Натомість вони спільно використовують лише оновлення моделі або градієнти, які агрегуються на центральному сервері або іншому координаційному вузлі для оновлення глобальної моделі. Рисунок 1.5 показує звичайний процес об'єднаного навчання, що включає три завдання: ініціалізацію, агрегацію та оновлення моделі, що виконується центральним вузлом; локальне навчання моделі та

локальні оновлення, що виконуються вузлами периферійних пристроїв. Цей підхід зменшує проблеми конфіденційності щодо обміну даними, оскільки конфіденційна інформація залишається на локальних пристроях, а учасники обмінюються лише оновленнями моделі. Однак проблема, пов'язана з тим, що центральний вузол знає, який пристрій навчився якої моделі, все ще залишається.

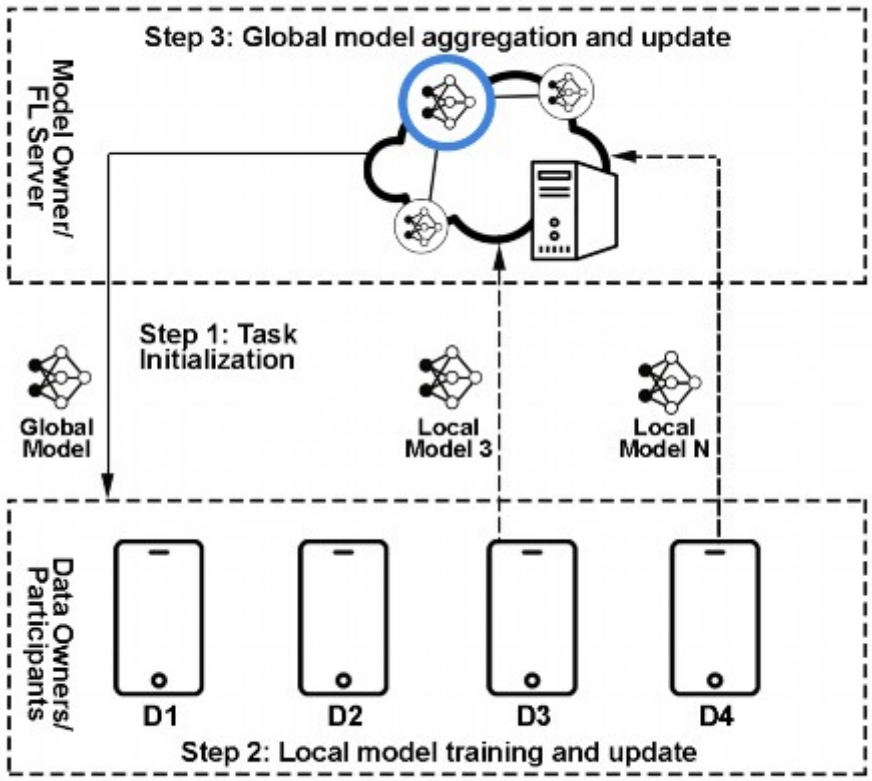


Рис. 1.5. Федеративний процес навчання з центральним вузлом, який підтримує глобальну модель, і периферійними пристроями, які виконують навчання локальної моделі.

Граничні пристрої все частіше використовуються в різних додатках, у тому числі в залучених GNN, завдяки їх повсюдності та здатності до локальних обчислень. Проте крайні пристрої демонструють значну неоднорідність з точки зору їх апаратних можливостей, підключення до

мережі та доступних ресурсів [17]. На рисунку 1.6 показано розподіл показників Geekbench Machine Learning для різних мобільних пристроїв [2].

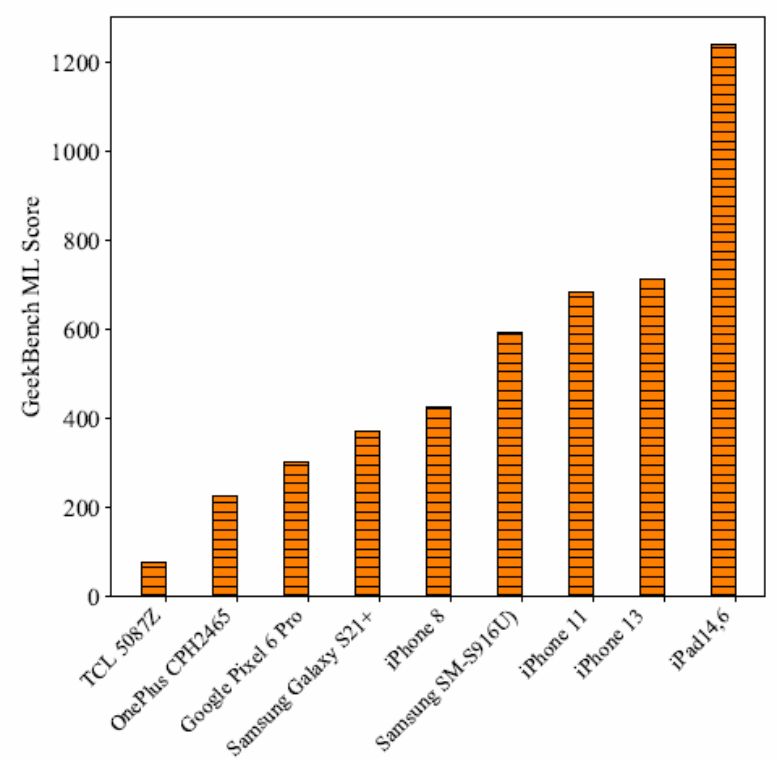


Рис. 1.6. Результати тесту Geekbench ML для різних пристроїв

Дані були взяті з найпопулярніших останніх контрольних тестів на квітень 2023 року без відбору, що ілюструє велику різноманітність і фрагментацію у використанні мобільного обладнання. Діапазон контрольних показників підкреслює значні відмінності в продуктивності машинного навчання на різних пристроях через їх апаратні можливості. Ця різноманітність створює унікальні проблеми під час їх включення у розподілені системи навчання, оскільки звичайні підходи можуть не врахувати ці відмінності. Крім того, зв’язок і синхронізація між периферійними пристроями в розподіленому середовищі може потребувати ресурсів, що потенційно може призвести до збільшення часу навчання та вузьких місць зв’язку.

Неоднорідність пристроїв у FL створює проблеми в моделях синхронного навчання. Час навчання на раунд визначається

найповільнішими учасниками, що призводить до неефективності. Кінцеві пристрої з обмеженими ресурсами можуть збільшити кількість відсіву під час процесу навчання та вплинути на точність моделі. Традиційні підходи FL часто ігнорують різноманітні можливості пристроїв, що призводить до довшого часу навчання та погіршення кінцевої точності моделі.

Включення федеративного навчання та периферійних пристроїв у контексті навчання GNN дає можливість вирішити ці проблеми, зберігаючи конфіденційність даних і використовуючи можливості розподілених обчислень периферійних пристроїв. Це дослідження зосереджено на розробці нового підходу до розподіленого навчання GNN у одноранговій мережі різнорідних периферійних пристроїв із використанням розподіленої хеш-таблиці для ефективного зв'язку та координації. Застосовуючи методи адаптивного розподілу графів і стратегії об'єднаного навчання, ми прагнемо забезпечити ефективне навчання з збереженням конфіденційності на різноманітних периферійних пристроях.

1.3. Завдання дослідження в контексті розподіленого навчання

Однією з головних цілей дослідження в цьому дослідженні є розробка ефективного підходу до розділення графів, який враховує неоднорідність периферійних пристроїв у розподіленому навчальному середовищі та працює з мережею DHT (Distributed Hash Table) p2p, а також для кількох наборів даних графів і архітектур GNN. Традиційні методи поділу графів, такі як Random Greedy Partitioning і Recursive Bisection [24], широко використовувалися для поділу графів на підграфи для паралельної обробки. Однак ці методи часто не враховують різноманітні апаратні можливості, обчислювальні ресурси та підключення до мережі крайніх пристроїв, що призводить до незбалансованого робочого навантаження та неоптимальної продуктивності навчання [31].

Щоб вирішити цю проблему, пропоноване дослідження має на меті розробити та реалізувати, а також дослідити існуючі алгоритми поділу графів, які враховують унікальні характеристики периферійних пристроїв, такі як їхні доступні ресурси та апаратні можливості (можливість обробки та об'єм пам'яті). Оптимізуючи процес поділу з урахуванням неоднорідності пристроїв, ми прагнемо створити більш збалансований розподіл даних графіка, забезпечуючи більш ефективне навчання та краще використання доступних ресурсів.

Крім того, запропонований підхід до розділення буде розроблено для підтримки структурної цілісності вхідного графа, забезпечуючи збереження зв'язків між вузлами під час процесу розділення. Це важливо для ефективного навчання GNN, як процес навчання ґрунтується на виявленні складних моделей і залежностей, присутніх у вхідних даних графа. Зрештою, наша мета полягає в тому, щоб розробити або протестувати методи поділу графів, які максимізують ефективність навчання та продуктивність у розподіленому середовищі, одночасно враховуючи різноманітні характеристики периферійних пристроїв.

1.3.1. Децентралізована мережа P2P для розподіленого навчання GNN

Ще однією ключовою метою цього дослідження є створення децентралізованої одноранговой (P2P) мережевої інфраструктури для ефективного та масштабованого розподіленого навчання GNN. Традиційні централізовані системи можуть зіткнутися зі значними вузькими місцями зв'язку та синхронізації , що обмежує їх масштабованість і адаптивність у великомасштабних програмах. З іншого боку, децентралізована мережа P2P може запропонувати покращену продуктивність, відмовостійкість і балансування навантаження, що робить її більш придатною для вирішення складних завдань розподіленого навчання GNN на периферійних пристроях.

У цьому дослідженні ми прагнемо дослідити та реалізувати мережу P2P на основі технології розподіленої хеш-таблиці (DHT), яка забезпечує

високомасштабований та ефективний метод організації та доступу до даних у децентралізований спосіб. Запропонована мережа P2P використовуватиме сильні сторони DHT, такі як їх самоорганізація та розподілена природа, щоб полегшити ефективний зв'язок і координацію між периферійними пристроями під час навчання GNN. Крім того, наша мета полягає в тому, щоб створити P2P-мережу, яка може легко впоратися з динамічною природою периферійних пристроїв, оскільки вони можуть часто приєднуватися або залишати мережу, зазнавати коливань ресурсів або стикатися з проблемами підключення. Децентралізована P2P-мережа буде розроблена таким чином, щоб адаптуватися до цих змін, забезпечуючи ефективність і надійність розподіленого процесу навчання GNN в умовах, що змінюються в мережі.

Ми спираємося на роботу [12], створюючи кільце P2P із географічно близьких пристроїв, розділяючи графи в головному вузлі, навчаючи та агрегуючи архітектури GNN спеціально, і показуючи, що масштабування працює для того самого. Впроваджуючи децентралізовану мережу P2P для розподіленого навчання GNN, наше дослідження спрямоване на подолання обмежень централізованих систем і підвищення масштабованості, адаптивності та продуктивності навчання GNN на крайніх пристроях.

1.3.2. Федеративне усереднення та збереження конфіденційності

Важливою метою цього дослідження є розробка механізму об'єднаного усереднення, який сприяє ефективному об'єднанню вивчених параметрів з окремих моделей GNN, зберігаючи конфіденційність. Інтегроване навчання стало багатообіцяючим підходом до навчання моделей машинного навчання на периферійних пристроях, що дозволяє їм спільно навчатися з децентралізованих джерел даних без необхідності обміну необробленими даними, тим самим захищаючи конфіденційність користувачів.

У нашому дослідженні ми прагнемо розробити та реалізувати об'єднаний алгоритм усереднення, який ефективно об'єднує параметри моделі GNN з кількох периферійних пристроїв у єдину глобальну модель. Це

виявилося простим, але ефективним методом об’єднаної агрегації. Цей процес має бути ефективним і надійним, гарантуючи, що зведена модель точно фіксує інформацію, отриману кожним окремим пристроєм під час процесу навчання.

Крім того, нашою метою є розробка об’єднаного механізму усереднення, який враховує неоднорідність периферійних пристроїв, оскільки вони можуть мати різні апаратні можливості, обчислювальні ресурси та кількість локальних даних. Запропонований алгоритм повинен мати можливість адаптуватися до цих варіацій, гарантуючи, що глобальна модель точно представляє колективне навчання всіх пристроїв-учасників.

Інший важливий аспект нашої дослідницької цілі полягає в тому, щоб гарантувати, що процес федеративного навчання зберігає конфіденційність даних, які використовуються для навчання GNN. Алгоритм має запобігати будь-якому потенційному витоку конфіденційної інформації під час процесу агрегації моделі, водночас забезпечуючи ефективне навчання з децентралізованих джерел даних.

Досягнувши цих цілей, наше дослідження має на меті розробити об’єднаний механізм усереднення, який забезпечує ефективне, надійне агрегування параметрів моделі GNN із збереженням конфіденційності, що веде до покращення продуктивності та застосовності навчання GNN у розподілених налаштуваннях на різнорідних периферійних пристроях.

1.4. Дослідження підходу адаптивного балансування навантаження за допомогою партиціонування (ALBP)

Наше дослідження представляє підхід Adaptive Load-Balanced Partitioning (ALBP) як новий внесок у сферу федеративного навчання розподілених графів. На відміну від існуючих методів, які зосереджені головним чином на відборі учасників на основі порогового значення [6], репутації [39], розподілу ймовірностей [36] навчання з підкріпленням [35], і

кластеризації [25], наш підхід наголошує на оптимізації використання кожного ресурсу.

Adaptive Load-Balanced Partitioning (ALBP) – це стратегія, яка використовується для розподілу навантаження в системах, що обробляють великі обсяги даних. Цей підхід дозволяє динамічно розподіляти завдання між різними ресурсами (наприклад, серверами, процесорами) таким чином, щоб збалансувати навантаження та оптимізувати загальну продуктивність системи.

Алгоритм роботи ALBP:

- Розбиття на частини: Великий обсяг даних або завдання розбивається на менші, більш керовані частини.

- Динамічний розподіл: Ці частини динамічно розподіляються між доступними ресурсами. Розподіл здійснюється з урахуванням таких факторів:

- Поточне навантаження: Системи ALBP постійно відстежують навантаження на кожному ресурсі.

- Продуктивність ресурсів: Враховується швидкість обробки даних різними ресурсами.

- Характеристика даних: Деякі частини даних можуть бути більш обчислювально складними, ніж інші.

- Адаптація: Системи ALBP постійно адаптуються до змінних умов. Якщо навантаження на якомусь ресурсі зростає, частина завдань перерозподіляється на інші ресурси.

Нехай маємо певний веб-сервіс, який обробляє велику кількість запитів користувачів. За допомогою ALBP можна розподілити обробку цих запитів між декількома серверами. Якщо навантаження на один з серверів зростає, частина запитів автоматично перенаправляється на інші сервери, забезпечуючи стабільну роботу сервісу.

ALBP - це стратегія, яка дозволяє динамічно розподіляти обчислювальне навантаження між різними ресурсами, забезпечуючи оптимальну продуктивність системи.

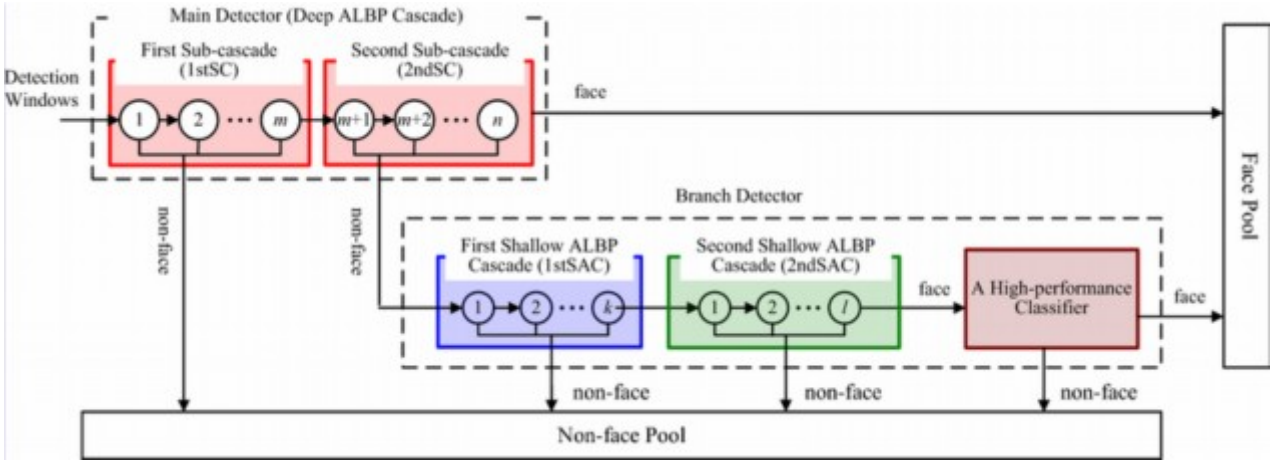


Рис. 1.7. Типова архітектур системи з ALBP

На рисунку 1.7 представлено типову архітектуру системи з ALBP:

- Центральний контролер:

Відстежує поточне навантаження на кожному ресурсі.

Аналізує характеристики даних та завдання.

Приймає рішення про розподіл завдань.

- Ресурси:

Можуть бути фізичними серверами, віртуальними машинами, контейнерами тощо.

Виконують обчислювальні завдання.

- Мережа:

Забезпечує зв'язок між центральним контролером та ресурсами.

Існує багато різних алгоритмів, які використовуються для реалізації ALBP. Деякі з найпоширеніших:

- Round-Robin: Завдання розподіляються по черзі між ресурсами.

- Найкоротша робота спочатку (Shortest Job First): Завдання з найменшим часом виконання розподіляються спочатку.

- Найбільш вільний ресурс (Least Loaded): Завдання розподіляються на ресурс з найменшим навантаженням.

- Алгоритми, засновані на машинному навчанні: Використовують машинне навчання для прогнозування навантаження та оптимізації розподілу.

ALBP є потужним інструментом для оптимізації роботи розподілених систем. Вибір конкретного алгоритму ALBP залежить від специфічних вимог системи та характеру обчислювальних завдань.

Гетерогенність даних у FL, де дані не є IID (незалежні та однаково розподілені) між пристроями, створює ще одну проблему. Місцеві набори даних можуть не відображати розподіл населення, що призводить до упередженого оновлення моделі та зниження точності моделі. Крім того, пристрої з нижчими можливостями менш імовірно будуть обрані для навчання, що призводить до упередженості вибору, яка може ще більше погіршити точність моделі.

Ми прагнемо включити всі периферійні пристрої в процес навчання замість відбору учасників, і, отже, незалежно від рівня їх продуктивності, шляхом розподілу розділів набору даних графа відповідно до їхніх обчислювальних можливостей. Ця стратегія гарантує, що навіть пристрої з нижчою продуктивністю не виключаються, а натомість отримують відповідний розділ, який дозволяє їм ефективно брати участь у загальному часовому проміжку навчання.

Наш підхід вирізняється своєю здатністю надавати зважене значення окремим моделям на основі вихідного співвідношення розподілу під час процесу агрегування моделей. Цей аспект GraphDHT (ALBP) відрізняє його від інших методів, які в основному зосереджені на оптимізації відбору учасників, як згадувалося вище, і який (GraphDHT) розглядає вибір найбільш підходящих учасників для тренувального раунду, а потім використовує звичайний розподіл для розподіленого об’єднаного навчання. Навпаки , наш метод зосереджений на максимізації корисності кожного ресурсу шляхом адаптації стратегії розподілу до унікальних можливостей кожного пристрою, забезпечуючи більш збалансоване та ефективне використання обчислювальних ресурсів у мережі.

Підхід ALBP відрізняється від традиційних методів, зосереджених на відборі учасників у федеративному навчанні, до подібної мети. Він пропонує

більш інклюзивний та ефективний спосіб використовувати весь потенціал гетерогенних периферійних пристроїв у розподіленому навчанні GNN, що робить його вагомим доповненням до зростаючої кількості досліджень у цій галузі.

Висновки до розділу

У висновках до розділу, що охоплює дослідження основних аспектів аналізу даних на основі графічних нейронних мереж (GNN), слід підкреслити наступні ключові моменти.

Графічні нейронні мережі відіграють важливу роль у сучасному аналізі даних, оскільки дозволяють ефективно працювати з неструктурованими та зв'язаними даними. Вивчені особливості GNN забезпечують глибше розуміння їхнього функціонування та ключових механізмів навчання.

Розподілене (федеративне) навчання на кінцевих пристроях значно підвищує можливості використання GNN в умовах обмежених обчислювальних ресурсів, зменшуючи потребу в централізованій обробці даних. Це також сприяє збереженню конфіденційності даних, що є особливо важливим у сучасних умовах підвищених вимог до безпеки.

Завдання децентралізованої мережі P2P для розподіленого навчання графових нейронних мереж було розглянуто у контексті його впливу на ефективність та масштабованість таких систем. Це дозволяє забезпечити адаптивне балансування навантаження та оптимізацію ресурсів.

Федеративне усереднення і застосування підходів до збереження конфіденційності даних підтвердили свою ефективність у контексті навчання GNN, забезпечуючи високу продуктивність моделей та зниження ризиків порушення конфіденційності.

Адаптивне балансування навантаження через партиціонування (ALBP) дозволяє більш раціонально використовувати ресурси, забезпечуючи

оптимізацію процесів навчання та обробки даних у розподілених середовищах.

Узагальнюючи, дослідження в даному розділі підтвердило значущість розвитку графових нейронних мереж у поєднанні з методами розподіленого навчання та збереження конфіденційності для підвищення ефективності обробки великих масивів даних у різних середовищах.

РОЗДІЛ 2. ДОСЛІДЖЕННЯ МЕТОДІВ ТА АЛГОРИТМІВ ГРАФОВИХ НЕЙРОННИХ МЕРЕЖ ДЛЯ РОЗПОДІЛЕНОГО НАВЧАННЯ

2.1. Представлення концепції розбиття графів

Розбиття графів є критичним кроком у розробці системи навчання розподілених графів. Воно передбачає декомпозицію вхідного графа на менші, керовані підграфи, які можуть оброблятися незалежно й одночасно різними вузлами в розподіленій мережі. Основна мета розбиття графів полягає в тому, щоб мінімізувати накладні витрати на зв'язок між вузлами, зберігаючи збалансоване робоче навантаження між ними, тому що розмір наборів даних графів може бути величезним [43]. Цей процес безпосередньо впливає на загальну продуктивність, ефективність і масштабованість системи. У цьому розділі ми обговоримо методи поділу графів, які використовуються в нашій системі розподіленого навчання, і їх вплив на продуктивність системи.

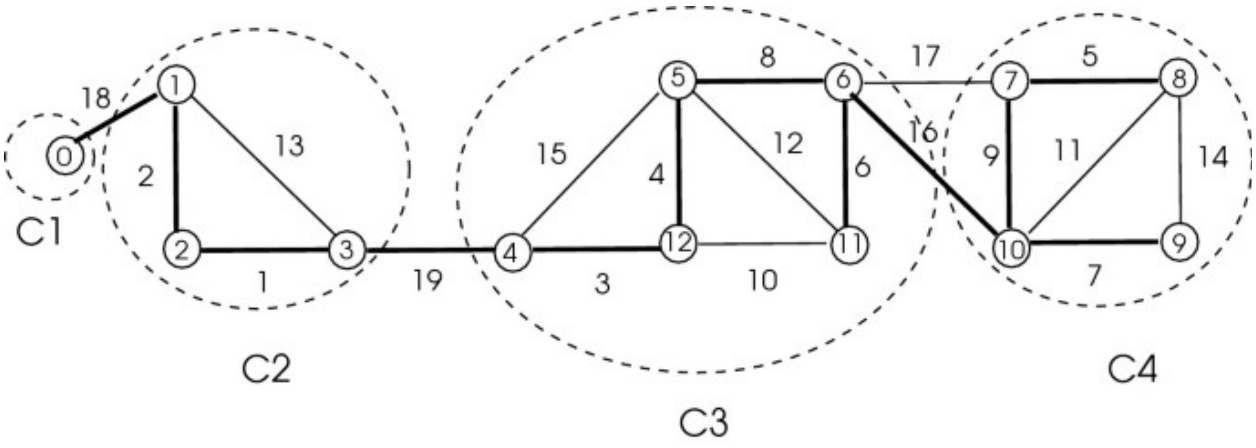


Рис. 2.1. Представлення процесу розбиття графа

Рисунок 2.1 ілюструє простий процес розбиття графа. Вхідний граф розкладається на менші підграфи, кожен з яких призначений іншому вузлу розподіленої мережі. Розбиття має на меті мінімізувати кількість вирізаних ребер і збалансувати робоче навантаження між вузлами.

Вибір методів поділу графів відіграє вирішальну роль у визначенні ефективності системи навчання розподілених графів. У цій роботі ми досліджуємо кілька стратегій поділу графів, включаючи розділення з адаптивним балансуванням навантаження (ALBP), розділення METIS та інші. У наступних підрозділах надається огляд цих методів і досліджуються їхні переваги та недоліки в контексті нашої системи.

2.1.1. Огляд методів розбиття графів

Методи розбиття графів можна розділити на два типи: евристичні методи та багаторівневі методи. Евристичні методи [23], такі як спектральне розбиття та алгоритм Кернігана-Ліна [44], базуються на оптимізації конкретних цілей, таких як мінімізація кількості розрізів по краях або максимізація модульності. Однак ці методи можуть бути дорогими з точки зору обчислень, особливо для великих графіків.

Багаторівневі методи [42], з іншого боку, більш масштабовані та ефективні. Вони складаються з фази укрупнення, фази поділу та фази розукрупнення. Під час фази укрупнення вхідний граф зменшується до меншого графа шляхом згортання вузлів і ребер. Фаза розбиття застосовує алгоритм розбиття до меншого графа, і, нарешті, фаза розгрублення уточнює розбиття на вихідному графі. Одним із широко використовуваних методів багаторівневого розділення є METIS [3], який використовує комбінацію евристик і багаторівневих схем для створення високоякісних розділів.

У контексті навчання розподіленого графа на гетерогенних периферійних пристроях техніка розділення повинна бути зосереджена не лише на мінімізації накладних витрат на зв’язок, але також враховувати апаратні можливості та обмеження ресурсів окремих пристроїв. Тому нам потрібен метод розподілу, який міг би адаптуватися до змінних ресурсів і потужності обробки периферійних пристроїв, зберігаючи бажаний баланс навантаження та ефективність зв’язку.

Враховуючи вимоги нашого конкретного випадку, ми дослідимо адаптивне розподілення балансування навантаження (ALBP) і більш традиційні методи розділення METIS у наступних розділах. Ці методи будуть оцінюватися на основі їх здатності вирішувати проблеми, пов'язані з різнорідними граничними пристроями в розподіленій системі навчання графів.

2.1.2. Адаптивне розподілення балансування навантаження (ALBP)

Adaptive Load Balancing Partitioning (ALBP) — це метод розділення графів, спеціально розроблений для розподіленого навчання графів на гетерогенних периферійних пристроях. Основною метою ALBP є забезпечення того, щоб розділені підграфи розподілялися між пристроями відповідно до їх відносних апаратних можливостей, таким чином досягаючи збалансованого робочого навантаження та ефективного використання ресурсів.

У нашому проекті ми спочатку обчислюємо відносні відсотки можливостей обладнання для всіх пристроїв у мережі. Ці відсотки розраховуються на основі таких факторів, як обчислювальна потужність, ємність пам'яті та пропускна здатність мережі, які впливають на загальну продуктивність пристроїв під час процесу навчання графіка. Сума цих відсотків для всіх пристроїв дорівнює одиниці.

Далі ми розділяємо вхідний графік відповідно до обчислених відносних відсоткових можливостей обладнання. Наприклад, якщо два пристрої мають відносні можливості 0,4 і 0,6, граф буде розділено на два підграфи з 40% і 60% вузлів і ребер відповідно. Це розділення гарантує, що кожен пристрій отримує робоче навантаження, пропорційне його апаратним можливостям, таким чином сприяючи ефективній паралельній обробці та мінімізуючи загальний час навчання.

Рисунок 2.2 ілюструє цей дизайн розділу адаптивного балансування навантаження. H представляє головний вузол, а C1-C5 — це п'ять дочірніх

вузлів, з’єднаних на кільці накладання кондитерського DHT. Розподіл коефіцієнтів апаратних засобів розраховується головою та відображається в таблиці над головою. Граф має відповідні пропорції, і локальні дочірні моделі навчаються на цьому графіку.

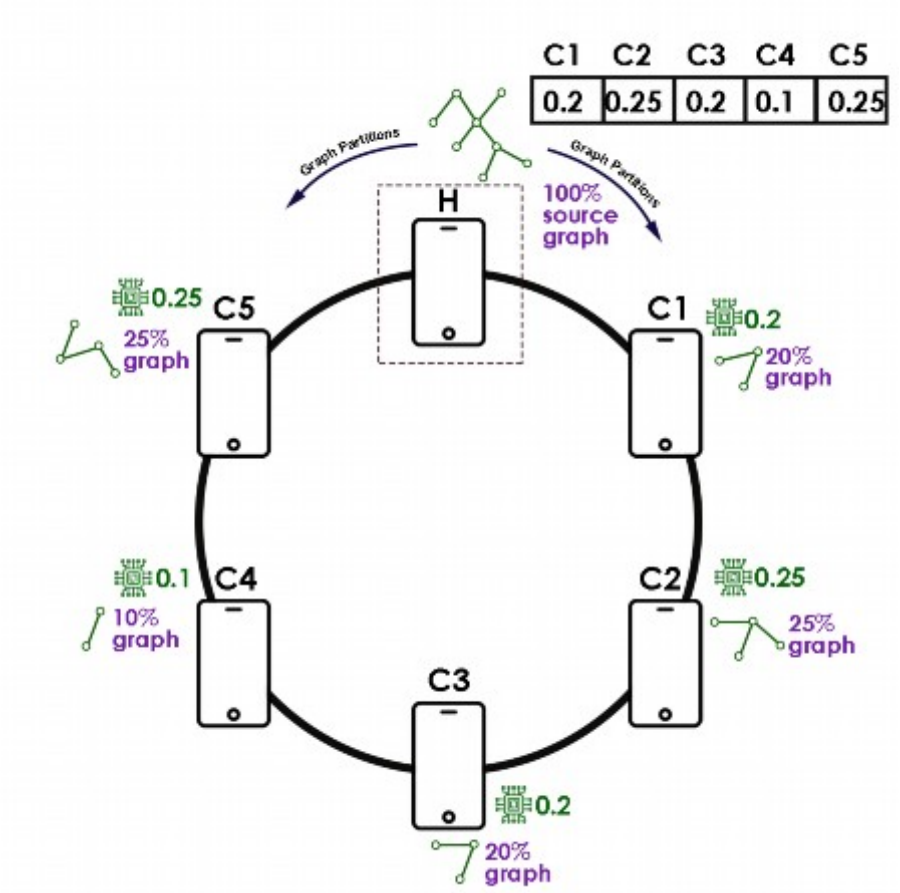


Рис. 2.2. Адаптивне розподілення балансування навантаження

У системі об’єднаного навчання, де ми об’єднуємо моделі шляхом усереднення вивчених параметрів окремих локальних моделей, вигідно розглядати початкові коефіцієнти розподілу як ваги для зваженого усереднення, на відміну від використання звичайного усереднення. Обґрунтування цього підходу полягає в тому, що моделі, які навчалися на значно більших піднаборах даних, мають більше значення в глобальній моделі порівняно з тими, які отримали менший піднабір даних. У наших експериментах ми досліджуємо як традиційні, так і зважені методи

усереднення, щоб перевірити цю гіпотезу та оцінити їхній вплив на загальну продуктивність системи навчання розподілених графів.

Використовуючи ALBP, ми прагнемо вирішити проблеми, пов’язані з балансуванням навантаження та повторним обмеженням джерела в системі навчання з розподіленими графами, що забезпечує ефективне та масштабоване навчання на гетерогенних граничних пристроях.

2.1.3 Алгоритм розбиття графів METIS

METIS — добре відома техніка розбиття графів, яка широко використовується в різних областях завдяки своїй ефективності та ефективності. Він заснований на парадигмі багаторівневого розбиття, яка здатна швидко розбивати великі графи, створюючи високоякісні розбиття. Багаторівнева парадигма складається з трьох основних фаз: укрупнення, початкове розділення та розукрупнення. На етапі укрупнення вихідний граф рекурсивно скорочується до меншого графа шляхом згортання вершин і ребер. Потім менший граф поділяється за допомогою простого алгоритму поділу під час початкової фази поділу. Нарешті, під час фази розгрублення розділ послідовно проектується назад до початкового графа, уточнюючи його на кожному рівні.

Ключові етапи алгоритму METIS:

- Ініціалізація: Граф розбивається на випадкові підграфи.

- Ітеративне покращення:

- Вибір вершини: Вибирається вершина, яка буде переміщена в інший підграф.

- Обчислення вартості переміщення: Розраховується, як зміниться загальна вартість розбиття, якщо перемістити цю вершину.

- Переміщення вершини: Якщо переміщення зменшує вартість, вершина переміщається.

- Завершення: Процес повторюється доти, доки не буде досягнуто заданого критерію зупинки (наприклад, максимальна кількість ітерацій або мінімальне змінення вартості).

METIS є потужним інструментом для розбиття графів, який знаходить широке застосування в різних областях. Завдяки своїй ефективності та гнучкості, він став стандартом де-факто для багатьох задач, пов'язаних з розбиттям графів.

METIS має відношення до архітектури нашої системи, оскільки він спрямований на мінімізацію країв, тобто загальну кількість ребер, що перетинають межі розділів. Зводячи до мінімуму зрізи, накладні витрати на зв’язок між пристроями під час об’єднаного процесу навчання можна зменшити, що призведе до більш ефективної системи. Крім того, METIS має високу масштабованість, що робить його придатним для великомасштабних сценаріїв поділу графів, які можуть зустрітися в нашій системі федеративного навчання на основі периферійних пристроїв.

У нашій реалізації ми адаптуємо METIS для роботи в нашій системі без зміни основного алгоритму. Замість цього ми зосереджуємось на інтеграції METIS у загальний робочий процес розділення графів, забезпечуючи безперебійну взаємодію між методом розділення та процесом федеративного навчання. Впроваджуючи та порівнюючи METIS і ALBP, два принципово різні підходи до розділення графів, ми прагнемо отримати краще розуміння та переваги, пов’язані з кожною технікою в контексті архітектури нашої системи.

Наша мета полягає в тому, щоб оцінити продуктивність методів розподілу адаптивного балансування навантаження (ALBP) і методів розподілу METIS з точки зору їхнього впливу на ефективність, комунікаційні витрати та результати навчання нашої інтегрованої системи навчання на основі периферійних пристроїв. Порівнюючи ці дві різні методи поділу, ми прагнемо визначити найбільш прийнятний підхід для нашої конкретної програми та надати цінну інформацію для майбутніх досліджень у цій галузі.

2.2. Архітектура та особливості графічних нейронних мереж

У цьому розділі приведено дизайн графових нейронних мереж, які ми націлюємо в нашій системі. GNN складаються з чистих згорткових шарів графів і згорткових шарів графів, інтегрованих із більш звичайними шарами, такими як CNN і LSTM. Наші GNN добре працюють як з класифікацією, так і з регресією, оскільки ми реалізуємо мережу для обох.

2.2.1. Архітектура графових нейронних мереж GNN

Графові нейронні мережі (GNN) — це клас моделей глибокого навчання, спеціально розроблених для обробки графоструктурованих даних. Останніми роками вони привернули значну увагу завдяки своїй здатності вивчати складні закономірності та зв’язки, наявні в графічних даних, перевершуючи традиційні методи машинного навчання в різних завданнях на основі графів. Основна ідея GNN полягає в тому, щоб використовувати структуру локального графа та розповсюджувати інформацію через мережу, дозволяючи моделі вивчати значущі вбудовування вузлів, які фіксують як локальні, так і глобальні властивості графа.

Архітектура GNN зазвичай складається з декількох згорткових шарів графа, які слідують за ними за допомогою функції зчитування, яка агрегує інформацію між вузлами для створення кінцевого результату. Типову архітектуру GNN можна показано на рисунку 2.3. Кожен згортковий рівень графа відповідає за вивчення нелінійної функції, яка відображає характеристики вхідного вузла та інформацію про їх сусідство з представленням вищого рівня. Це досягається за допомогою механізму передачі повідомлень, який дозволяє вузлам спілкуватися та обмінюватися інформацією зі своїми сусідами [28]. Отримані вбудовування вузлів потім використовуються як вхідні дані для наступного згорткового рівня графа, дозволяючи моделі охоплювати більш складні шаблони в даних графа під час проходження шарами [32].

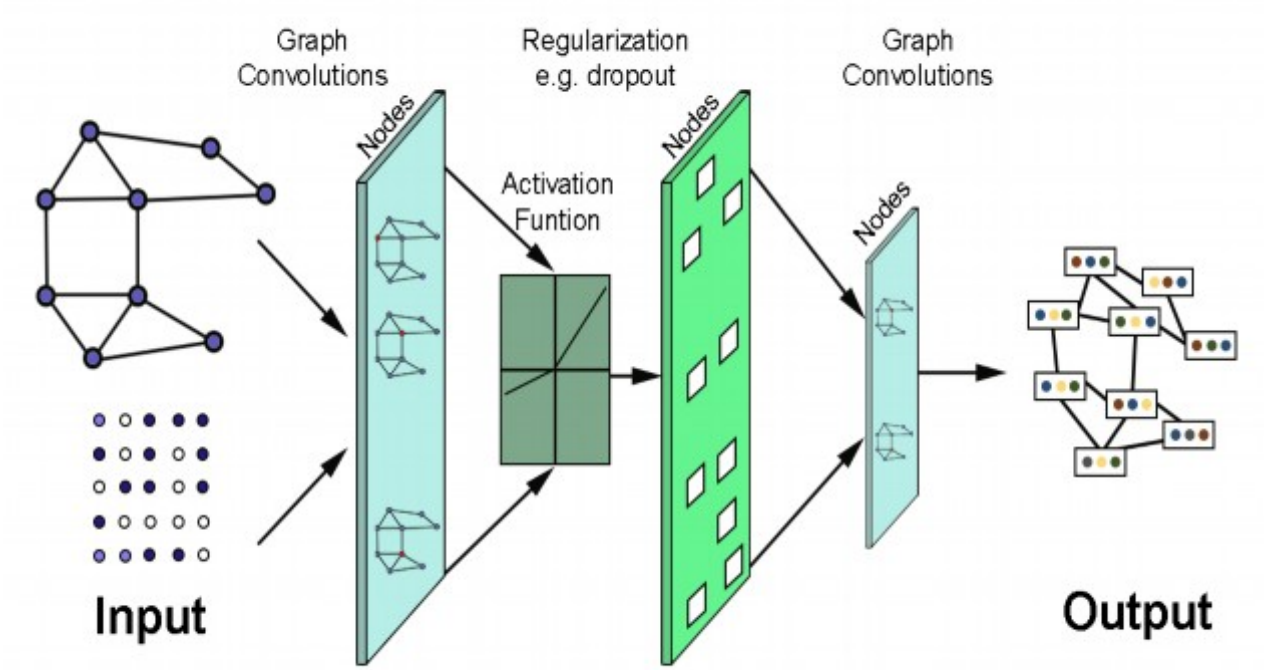


Рис. 2.3. Типова архітектура GNN, що демонструє кілька згорткових шарів графа та функцію зчитування, яка агрегує інформацію про вузли для генерації кінцевого результату

Функція зчитування застосовується після останнього шару згортки графа, щоб агрегувати вбудовані вузли та отримати бажаний результат. Це можуть бути передбачення на рівні вузла, наприклад класифікація вузла, або передбачення на рівні графіка, наприклад завдання класифікації графа або регресії. Вибір функції зчитування залежить від конкретної проблеми та бажаного формату виведення.

GNN надають потужну та гнучку структуру для вивчення представлень графоструктурованих даних. Їхня архітектура, що складається зі згорткових шарів графа та функції зчитування , дозволяє їм фіксувати складні шаблони та зв’язки, присутні у вхідному графі, зберігаючи масштабованість та інтерпретацію [7]. У контексті нашої об’єднаної системи навчання GNN пропонують багатообіцяючий підхід до навчання на основі графоструктурованих даних, розподілених між периферійними пристроями.

2.2.2. Граф згорткових шарів

Графові згорткові шари (GCL) утворюють блоки графових нейронних мереж (GNN) і відповідають за вивчення представлень вузлів у графі шляхом агрегування інформації з їх локальних околиць. Ці шари мають на меті узагальнити операцію згортання від звичайних структур, подібних до сітки, таких як зображення, до нерегулярних даних, структурованих на графах.

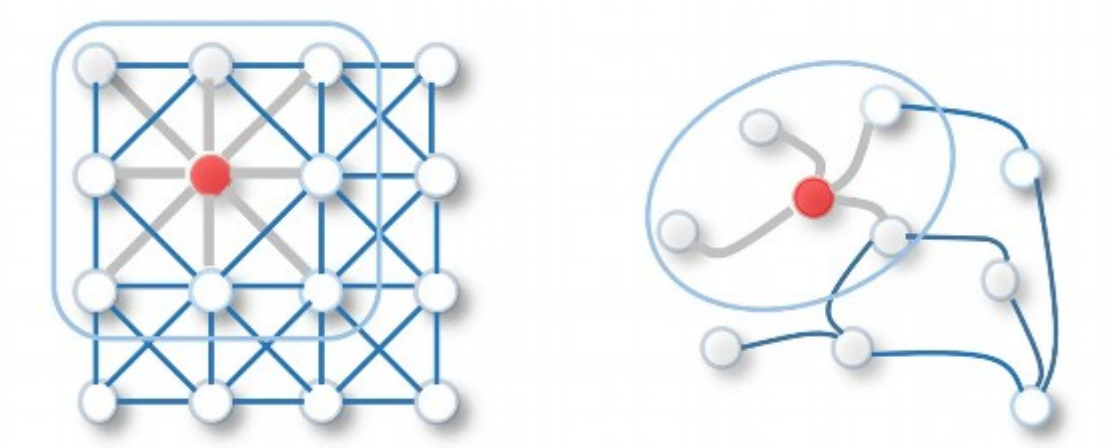


Рис. 2.4. Перетворення згортки вузла та векторів ознак його сусідніх вузлів на прикладах графів.

Просту репрезентативну операцію згортки графіка можна побачити на рисунку 2.4. GCL оновлює вектор ознак вузла, агрегуючи вектори ознак сусідніх вузлів і застосовуючи нелінійне перетворення. Агрегацію можна здійснити за допомогою різних стратегій, таких як сума, середнє або максимальне об’єднання. Вибір функції агрегації залежить від конкретної проблеми та бажаних властивостей вбудовування вузлів.

У типовому GCL оновлений вектор ознак для вузла i обчислюється наступним чином:

1. Агрегувати вектори ознак сусідів вузла i: для кожного сусіднього вузла j повідомлення обчислюється як функція його вектора ознак і атрибутів краю (якщо доступно). Потім повідомлення від усіх сусідів агрегуються за допомогою вибраної функції агрегування.

2. Оновити вектор ознак вузла: агреговане повідомлення об’єднується з поточним вектором ознак вузла і за допомогою вагової матриці, що вивчається. Потім для отримання оновленого вектора ознак застосовується нелінійна функція активації, наприклад ReLU або sigmoid.

Процес повторюється для всіх вузлів у графі, а отримані вбудовування вузлів використовуються як вхідні дані для наступного GCL або функції зчитування, залежно від архітектури GNN.

GCL відіграють вирішальну роль у GNN, дозволяючи моделі вивчати значущі представлення вузлів , які фіксують як локальні, так і глобальні властивості графа. Їхня здатність поширювати інформацію по графу дозволяє GNN ефективно обробляти дані, структуровані на графах, і досягати чудової продуктивності в різних графових завданнях.

2.2.3. Графічна згортка з інтеграцією LSTM

У цій роботі ми представляємо архітектуру, яка поєднує графові згорткові мережі (GCN) із шарами довгострокової короткочасної пам’яті (LSTM) для ефективної обробки графоструктурованих даних із послідовними або часовими характеристиками. Інтеграція LSTM у GCN дозволяє моделі охоплювати як топологічну структуру графа, так і часові залежності, наявні в атрибутах вузла та краю, що особливо корисно для різних завдань регресії.

Графічно структуровані дані з послідовними або часовими аспектами можна знайти в різних областях застосування, таких як соціальні мережі з еволюцією взаємодії користувачів [13], мережі цитування з публікаціями з мітками часу і мережі трафіку з даними датчиків у реальному часі.

Включення тимчасової інформації в таких випадках може призвести до більш точних і надійних прогнозів.

Запропонована архітектура складається з таких компонентів:

1. Згорткові шари графа (GCL): ці рівні відповідають за обробку даних, структурованих у графі, і представлення вузлів навчання, які фіксують локальні та глобальні властивості графа. GCL виконують агрегацію функцій

сусідніх вузлів і застосовують нелінійні перетворення для оновлення вбудованих вузлів.

2. Шари LSTM: після GCL оновлені вбудовані вузли надходять у шари LSTM, які спеціально розроблені для обробки послідовних даних. Рівні LSTM фіксують тимчасові залежності в атрибутах вузла, дозволяючи моделі вивчати складні шаблони та довгострокові залежності між кроками часу.

3. Шари зчитування та регресії: вихідні дані шарів LSTM потім пропускаються через функцію зчитування для отримання представлення на рівні графіка. Це представлення згодом використовується як вхідні дані для одного або кількох повністю пов’язаних рівнів, за якими слідує остаточний рівень регресії, який передбачає цільові значення.

Завдяки інтеграції шарів LSTM в архітектуру на основі GCN наша модель може ефективно обробляти структуровані на графах дані з часовими характеристиками, що забезпечує кращу продуктивність у різних завданнях регресії. У розділі про реалізацію ми демонструємо ефективність запропонованої архітектури в задачі прогнозування трафіку, демонструючи переваги поєднання згорткових шарів графа з шарами LSTM для обробки складних графоструктурованих даних.

2.3. Підхід до розподіленого навчання графових нейронних мереж з використанням розподіленої хеш-таблиці

Щоб забезпечити масштабований та ефективний підхід до розподіленого навчання для графових нейронних мереж, ми використовуємо розподілену хеш-таблицю (DHT) як базову мережеву інфраструктуру. DHT — це клас децентралізованих розподілених систем, які надають службу пошуку, подібну до хеш-таблиці, що дозволяє ефективно зберігати та отримувати пари ключ-значення. У контексті нашого дослідження DHT-мережі сприяють розподілу розбиття графів, навчанню моделі та зв’язку між гетерогенними граничними пристроями. У цьому розділі ми обговорюємо

конкретну мережеву архітектуру DHT, що використовується в нашій системі, яка базується на Pastry DHT і системі публікації/підписки Scribe, а також відповідні механізми відновлення після збою.

2.3.1. Алгоритм побудови розподілених хеш таблиць Pasty DHT

Мережа Pastry DHT, яка є основою нашої системи розподіленого навчання, є високомасштабованою та стійкою архітектурою, розробленою для ефективної маршрутизації повідомлень і розподілу даних між вузлами. У контексті нашого дослідження Pastry забезпечує ефективну координацію навчання нейронної мережі розподіленого графа та розподіл даних між різнорідними граничними пристроями.

Pastry організовує вузли в кільцеву структуру, призначаючи кожному вузлу унікальний ідентифікатор під назвою nodeId, як показано на рисунку 2.5. NodeId отримується з узгодженої функції хешування, що забезпечує збалансований розподіл вузлів у мережі.

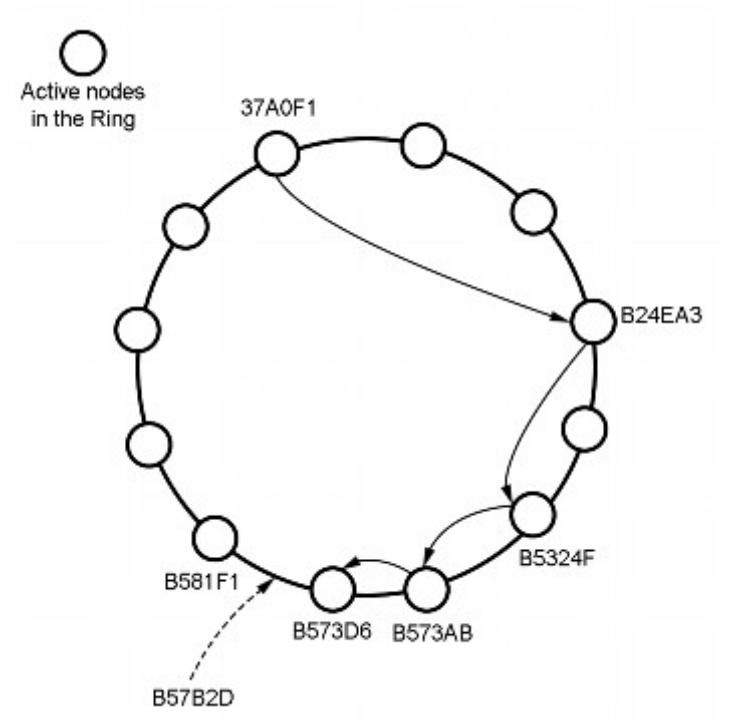


Рис. 2.5. Представлення кільця Pastry та маршрутизація повідомлень між двома вузлами

У цьому прикладі вузол Pastry 37A0F1 направляє повідомлення до вузла з LID, чисельно найближчим до даного ключа B57B2D. На рисунку 2.5 показано використання префіксної маршрутизації, де вузол p пересилає повідомлення до вузла q, який спільно використовує однозначний (або b біт) довший префікс з ключем k порівняно з LID p. Протокол Pastry забезпечує ефективний пошук пар ключ-значення гібридним способом, використовуючи як деревоподібні, так і кільцеподібні структури.

Мережа Pastry DHT використовує високоефективний механізм маршрутизації, який використовує як простір ідентифікатора вузла, так і топологію мережі, щоб мінімізувати затримку зв'язку. На рисунку 2.5 також показано механізм маршрутизації. В основі цього механізму знаходяться таблиці маршрутизації та листові набори, які підтримуються кожним вузлом. Таблиця маршрутизації організована в ієрархічну структуру з рядками, що відповідають довжині спільного префікса ідентифікаторів вузлів, і стовпцями, що відповідають можливим значенням наступної цифри в ідентифікаторі вузла. Кожен запис у таблиці маршрутизації містить інформацію про вузол, ідентифікатор якого має відповідну довжину префікса та має вказану наступну цифру. Ця структура дозволяє приймати швидкі та точні рішення щодо маршрутизації на основі ідентифікатора вузла призначення.

На додаток до таблиці маршрутизації, кожен вузол підтримує кінцевий набір, що складається з чисельно найближчих вузлів, як менших, так і більших, у просторі ID. Листовий набір сприяє ефективній маршрутизації на останніх етапах процесу доставки повідомлення, гарантуючи, що повідомлення досягне цільового вузла або його безпосереднього сусіда лише за кілька стрибків. Коли вузол отримує повідомлення, він звертається до своєї таблиці маршрутизації та листового набору, щоб переслати повідомлення до наступного вузла, чисельно ближчого до адресата. Цей процес повторюється ітеративно, доки повідомлення не досягне своєї мети або вузла без ближчих сусідів. Комбінація таблиці маршрутизації та

листового набору дозволяє Pastry маршрутизувати повідомлення в O(log N) переходах, де N — загальна кількість вузлів у мережі, що створює масштабовану та ефективну комунікаційну інфраструктуру для нашої системи навчання нейронної мережі з розподіленими графами.

У нашій системі мережа Pastry DHT відповідає за керування розподілом розділів графів між периферійними пристроями, а також за обробку зв'язку та агрегації оновлень моделі протягом усього процесу об'єднаного навчання. Використовуючи ефективний механізм маршрутизації Pastry, наша система може динамічно адаптуватися до додавання або видалення вузлів, забезпечуючи надійність і масштабованість в умовах різноманітних умов мережі та можливостей пристроїв.

2.3.2. Децентралізована система Scribe

Scribe – це децентралізована система публікації/підписки, побудована на основі мережі Pastry DHT [14]. Він забезпечує масштабовану та відмовостійку комунікаційну інфраструктуру для нашої системи навчання нейронної мережі з розподіленими графами. У Scribe вузли можуть діяти як видавці, передплатники або обидва, забезпечуючи гнучку модель зв'язку, яка підтримує різні сценарії навчання та конфігурації мережі.

Scribe організовує теми в ієрархічній структурі, і кожна тема пов'язана з унікальним ідентифікатором. Цей ідентифікатор виводиться з хешу назви теми та відображається на вузлі в кільці Pastry DHT, який служить точкою зустрічі для теми. Цей вузол, відомий як корінь теми, відповідає за керування деревом багатоадресної розсилки та пересилання повідомлень передплатникам теми.

Коли вузол бажає підписатися на тему, він надсилає повідомлення про приєднання до кореня теми. Потім корінь оновлює дерево багатоадресної передачі, додаючи нового абонента до відповідної гілки, гарантуючи, що дерево залишається збалансованим і ефективним. Після підписки вузол може отримувати повідомлення, опубліковані в цій темі видавцями. Щоб

опублікувати повідомлення, вузол надсилає повідомлення до кореня теми, який потім розповсюджує повідомлення всім передплатникам за допомогою дерева багатоадресної розсилки. Цей процес забезпечує ефективну та надійну доставку повідомлень, мінімізуючи мережевий трафік і затримку.

У контексті нашої системи навчання нейронної мережі з розподіленим графом Scribe забезпечує ефективний зв'язок між різнорідними периферійними пристроями. Пристрої можуть підписуватися на теми, що відповідають різним аспектам процесу навчання, таким як оновлення моделі, обмін параметрами та сигнали синхронізації. Модель публікації/підписки спрощує процес комунікації та дозволяє системі динамічно адаптуватися до змін у топології мережі або навчальному навантаженні. Використовуючи масштабованість і відмовостійкість Scribe, наша система може ефективно координувати процес навчання на великій кількості периферійних пристроїв, забезпечуючи своєчасне й точне оновлення моделі та сприяючи ефективному навчанню графових нейронних мереж.

2.3.3. Механізми відновлення навчання мережі після збою

У системі навчання нейронної мережі з розподіленим графом, як наша, відновлення після збоїв має вирішальне значення для підтримки стійкості та надійності системи, особливо під час роботи з неоднорідними периферійними пристроями, які можуть мати переривчасте підключення, апаратні збої або інші проблеми. Наша система використовує внутрішні механізми відмовостійкості мережі Pastry DHT і системи публікації/підписки Scribe, щоб забезпечити ефективне та надійне відновлення після збоїв.

Мережа Pastry DHT використовує декілька механізмів для обробки збоїв вузлів, гарантуючи, що система залишається працездатною навіть у разі збоїв кількох вузлів. По-перше, він підтримує кілька записів таблиці маршрутизації для кожного значення цифри, забезпечуючи альтернативні шляхи на випадок збою вузла. Ця надлишковість змушує мережу маршрутизувати повідомлення навколо несправних вузлів, зберігаючи

з’єднання, навіть якщо деякі вузли недоступні. По-друге, Pastry використовує листовий набір, який є набором чисельно найближчих вузлів до певного вузла. Кінцевий набір надає додаткові шляхи резервного копіювання та постійно оновлюється відповідно до змін у топології мережі.

У разі збою вузла мережа Pastry DHT виявляє збій і відповідно оновлює таблиці маршрутизації та листові набори постраждалих вузлів. Цей процес виконується проактивним способом, гарантуючи, що мережа залишається стійкою та адаптованою до змін у топології.

Система публікації/підписки Scribe також надає механізми відмовостійкості для обробки збоїв у дереві багатоадресної розсилки. Коли виявляється збій вузла, корінь теми реорганізовує дерево багатоадресної розсилки, щоб виключити несправний вузол, і перепризначає передплатників до решти гілок. Це гарантує, що дерево залишається збалансованим і ефективним, а збій вузла не впливає на доставку повідомлень передплатникам. У нашій системі навчання нейронної мережі з розподіленими графами ці механізми відновлення після збоїв забезпечують стійкість і надійність процесу навчання. Коли пристрій зазнає збою, система може автоматично адаптуватися та продовжити процес навчання з мінімальним впливом на загальну продуктивність. Ця відмовостійкість особливо корисна в середовищі периферійних пристроїв, де пристрої можуть мати різний рівень надійності та доступності. Використовуючи внутрішні механізми відмовостійкості мережі Pastry DHT і системи публікації/підписки Scribe, наша система забезпечує стійку та ефективну інфраструктуру для навчання нейронної мережі розподілених графів.

2.4. Представлення робочого процесу навчання на основі розподіленої хеш-таблиці

Робочий процес складається з декількох етапів і візуально представлений на рисунку 2.6.

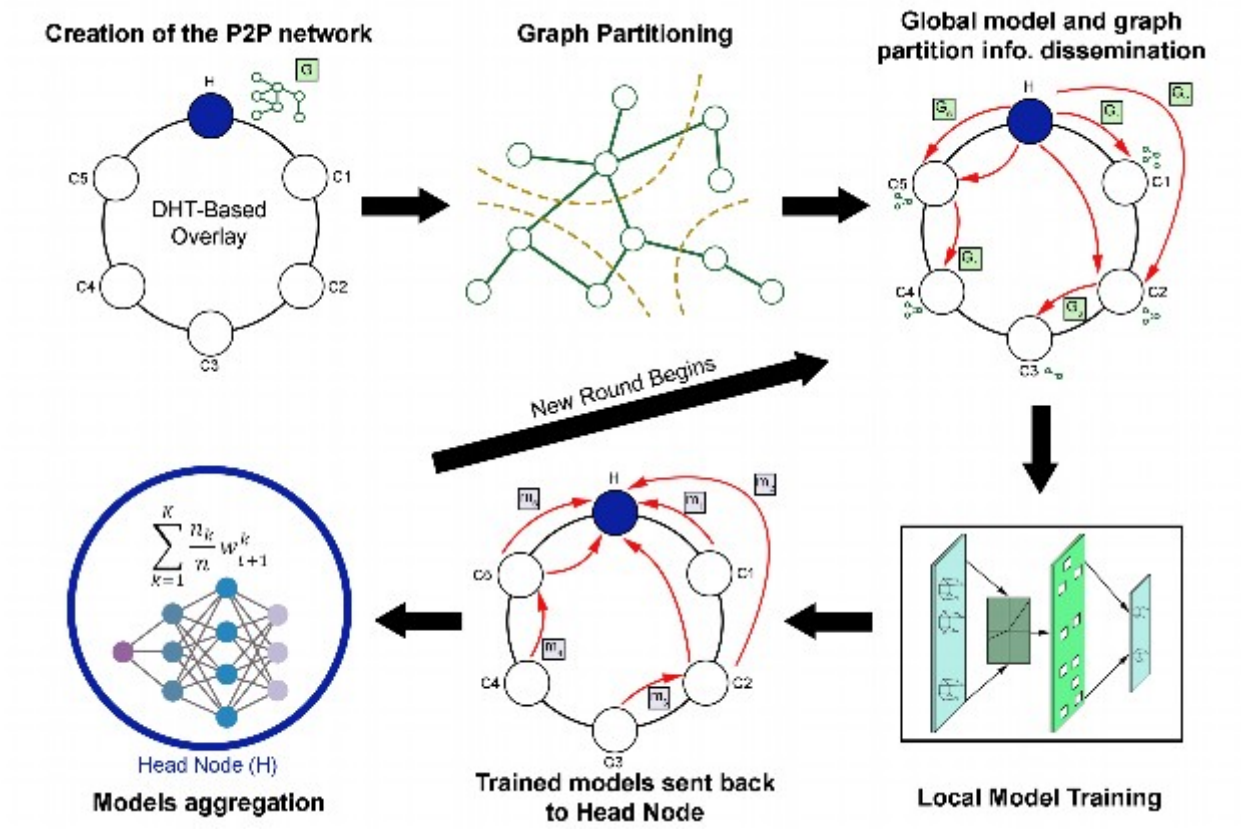


Рис. 2.6. Робочий процес системи GraphDHT

На рисунку 2.6 показано робочий процес який складається з наступних етапів:

а) Ініціалізація мережі Pastry DHT;

б) Сегментація графа;

в) Розповсюдження моделі та розділу;

г) Місцева модель навчання;

д) Передача моделі до головного вузла;

е) Агрегація глобальної моделі.

2.4.1. Ініціалізація та розбиття графа

Фаза ініціалізації та розділення графів нашої системи закладає основу для ефективного навчання нейронної мережі розподіленого графа на гетерогенних граничних пристроях. У цьому розділі ми окреслюємо етапи цієї критичної фази, готуючи основу для ефективного та масштабованого навчання моделі.

1. Ініціалізація: спочатку кожен периферійний пристрій у мережі Pastry DHT налаштовано за допомогою відповідного програмного забезпечення, включаючи код DHT (Java) і глибокого навчання (Python). Ці пристрої утворюють вузли мережі DHT, готові брати участь у навчальному процесі. Географічно релевантні вузли (наприклад, ті, що знаходяться лише в США або ті, що лише в організації) стають частиною накладеної мережі P2P. Випадковий вузол позначається як головний вузол, відповідальний за координацію процесу навчання та керування розділами даних.

2. Завантаження набору даних: головний вузол завантажує повний набір даних графа, включаючи функції вузла, з'єднання країв і мітки. Цей набір даних попередньо обробляється, щоб гарантувати, що він має відповідний формат для навчання нейронної мережі графів.

3. Оцінка можливостей пристрою: головний вузол спілкується з усіма залученими пристроями, щоб оцінити їх апаратні можливості. Ця інформація є важливою для визначення оптимальної стратегії розподілу для набору даних графа, враховуючи гетерогенну природу пристроїв. Головний вузол обчислює відносний розподіл апаратних можливостей у відсотках кожного пристрою, який використовуватиметься для розподілу адаптивного балансування навантаження. Для цієї оцінки можна використовувати будь-який настроюваний алгоритм. Для GraphDHT ми взяли кількість процесорів і розмір пам'яті як орієнтир.

4. Поділ графа: на основі обраного методу поділу (розбивання з адаптивним балансуванням навантаження або поділ METIS) головний вузол ділить набір даних графа на підграфи, гарантуючи, що кожен пристрій отримує розділ відповідного розміру на основі його апаратних можливостей. У разі адаптивного балансування навантаження розміри розділів визначаються відносними відсотками апаратних можливостей, обчисленими раніше для розбиття METIS, головний вузол запускає алгоритм METIS для отримання оптимального розбиття графа з урахуванням кількості пристроїв та їхніх можливостей.

5. Розподіл даних: після того як набір даних графа розділено, головний вузол надсилає кожен розділ на відповідний пристрій. Цей крок передбачає передачу характеристик вузла, з’єднань країв і міток, пов’язаних з кожним підграфом. Пристрої зберігають відповідні розділи локально та готуються до етапу навчання моделі. Якщо конфіденційність набору даних не викликає занепокоєння, ми можемо зберігати весь набір даних на кожному з периферійних пристроїв і надсилати інформацію про розділ на кожен окремий вузол, щоб він міг розділяти та використовувати власний ресурс. Це значно мінімізує витрати на зв'язок.

Після завершення фази ініціалізації та розділення графа система тепер готова розпочати навчання нейронної мережі розподіленого графа. Ретельно розділяючи набір даних графа та розподіляючи його між різнорідними периферійними пристроями, система забезпечує ефективний і масштабований процес навчання, який використовує можливості кожного пристрою-учасника.

2.4.2. Фаза федеративного процесу навчання

Після фази ініціалізації та поділу графа починається об’єднаний процес навчання. У цьому розділі детально описуються етапи процесу об’єднаного навчання, зосереджуючись на навчанні розподіленої моделі, агрегації моделей і синхронізації між різнорідними периферійними пристроями.

1. Навчання локальної моделі: коли кожен крайовий пристрій отримав свій розділ набору даних графа, вони переходять до самостійного навчання своїх локальних моделей. Використовуючи виділений підграф, пристрої виконують алгоритм навчання нейронної мережі графа, оновлюючи параметри моделі на основі свого конкретного розділу даних. Процес локального навчання може відрізнятися за тривалістю залежно від апаратних можливостей кожного пристрою, а також від складності архітектури GNN.

2. Агрегація моделі: після завершення циклу навчання на всіх пристроях головний вузол ініціює процес агрегації моделі. Кожен крайовий

пристрій надсилає параметри своєї локальної моделі головному вузлу, який відповідає за агрегування цих параметрів для формування глобальної моделі. У нашій системі розглядаються два методи агрегування: звичайне усереднення та зважене усереднення з використанням початкових коефіцієнтів розподілу. Цей вибір дозволяє нам досліджувати вплив різних стратегій агрегації на продуктивність моделі.

3. Оновлення глобальної моделі: коли головний вузол агрегує локальні параметри моделі, він оновлює глобальну модель новими обчисленими параметрами. Цей крок гарантує, що знання, отримані від локального навчання кожного пристрою, інтегруються в єдину уніфіковану модель, яка представляє весь набір даних графіка.

4. Синхронізація моделі: після оновлення глобальної моделі головний вузол розповсюджує оновлені параметри на всі периферійні пристрої, що беруть участь. Кожен пристрій замінює локальні параметри моделі параметрами глобальної моделі, гарантуючи, що всі пристрої починають наступний раунд навчання з узгодженою та актуальною моделлю.

5. Ітеративний процес: процес об’єднаного навчання продовжується ітеративно, при цьому кожен раунд включає локальне навчання моделі, агрегацію моделі, глобальне оновлення моделі та синхронізацію моделі. Процес повторюється, доки не буде виконано визначений критерій зупинки, наприклад досягнуто попередньо визначеної кількості тренувальних раундів або досягнуто цільової метрики продуктивності моделі.

Застосовуючи об’єднаний процес навчання, наша система дає змогу навчатися роботі нейронної мережі з розподіленим графом на гетерогенних периферійних пристроях, зберігаючи при цьому глобальну модель, яка консолідує навчання з усіх пристроїв-учасників. Цей підхід не тільки використовує обчислювальні ресурси кожного пристрою, але й забезпечує масштабований та ефективний процес навчання, який поважає обмеження та можливості основного апаратного забезпечення.

2.4.3. Агрегація та оцінка моделі

Після процесу об’єднаного навчання фокус зміщується на агрегацію та оцінку глобальної моделі на тестовому наборі даних. У цьому розділі докладно розглядаються етапи агрегування та оцінки моделі, підкреслюються методологічні міркування та застосовані показники ефективності.

1. Остаточне агрегування моделі: наприкінці процесу об’єднаного навчання головний вузол виконує остаточне агрегування моделі шляхом інтеграції локальних параметрів моделі з усіх периферійних пристроїв, що беруть участь. Метод агрегування, або звичайне усереднення, або зважене усереднення з використанням початкових коефіцієнтів розподілу, залежить від обраного підходу, як описано в розділі 2.4.2.

2. Підготовка до оцінки моделі: Отримавши остаточну глобальну модель, необхідно оцінити її продуктивність на тестовому наборі даних, отриманому з процесу навчання. Цей набір даних має бути репрезентативним для базового розподілу даних на графіку та достатньо великим, щоб забезпечити суттєве уявлення про можливості узагальнення моделі.

3. Метрики оцінки: для оцінки ефективності глобальної моделі використовуються різні метрики залежно від характеру проблеми, що розглядається. Для класифікації завдань, загальні показники включають точність, точність, запам’ятовування та оцінку F1. У випадку задач регресії використовуються такі показники, як середня абсолютна помилка (MAE), середньоквадратична помилка (RMSE) і коефіцієнт детермінації (R 2).

4. Оцінка моделі: глобальна модель застосовується до тестового набору даних, створюючи прогнози, які можна порівняти з фактичними цільовими значеннями. Обчислюючи показники оцінки цих прогнозів, ми можемо кількісно оцінити продуктивність моделі та оцінити її здатність узагальнювати нові, невідомі дані. Цей крок також дозволяє ідентифікувати потенційні області для вдосконалення, інформуючи про подальші вдосконалення архітектури моделі або процесу навчання.

5. Порівняння методів агрегування: щоб підтвердити гіпотезу щодо впливу методів агрегування на продуктивність глобальної моделі, порівнюються результати оцінювання для традиційних стратегій усереднення та зваженого усереднення. Цей аналіз дозволяє нам зробити висновки щодо придатності кожного підходу в контексті нашої об’єднаної системи навчання та конкретних характеристик набору даних графів.

Ретельно агрегуючи та оцінюючи глобальну модель, ми забезпечуємо всебічне розуміння її ефективності на невидимих даних, надаючи цінну інформацію про ефективність пропонованої об’єднаної системи навчання в навчанні графових нейронних мереж на гетерогенних граничних пристроях. Цей процес оцінювання також слугує ключовим механізмом зворотного зв’язку, керуючи подальшими вдосконаленнями та вдосконаленнями дизайну та впровадження системи.

Висновки до розділу

У висновках до розділу, що охоплює дослідження методів та алгоритмів графових нейронних мереж для розподіленого навчання, можна виділити наступні ключові аспекти.

Розбиття графів є фундаментальною концепцією для ефективної роботи графових нейронних мереж (GNN) у розподілених середовищах. Різні методи розбиття, включаючи адаптивне розподілення балансування навантаження (ALBP) та алгоритм METIS, дозволяють забезпечити оптимальне розподілення даних і зменшити навантаження на обчислювальні ресурси.

Архітектура графових нейронних мереж (GNN) детально розглянута в контексті їхніх основних елементів, таких як графові згорткові шари та інтеграція LSTM. Ці компоненти забезпечують здатність GNN моделювати складні взаємозв'язки між вузлами графів та покращувати ефективність навчання.

Розподілене навчання графових нейронних мереж із застосуванням розподіленої хеш-таблиці (DHT) демонструє значний потенціал для масштабованого навчання. Алгоритми побудови розподілених хеш-таблиць, такі як Pasty DHT, та децентралізована система Scribe дозволяють ефективно управляти процесом розподілення даних і забезпечують надійність системи у випадку збоїв.

Механізми відновлення після збоїв є важливим елементом у забезпеченні стабільності систем розподіленого навчання. Це дозволяє забезпечити безперервність процесу навчання графових нейронних мереж, навіть у випадках тимчасових проблем із мережею або обчислювальними ресурсами.

Робочий процес розподіленого навчання включає етапи ініціалізації, розбиття графа, федеративне навчання та агрегацію моделей. Кожен з цих етапів має важливе значення для забезпечення ефективної роботи GNN у розподілених середовищах та підвищення якості моделі.

Узагальнюючи, досліджені методи та алгоритми довели свою ефективність для розподіленого навчання графових нейронних мереж. Використання технологій, таких як розподілена хеш-таблиця та федеративне навчання, забезпечує масштабованість, надійність та продуктивність у великих розподілених системах.

РОЗДІЛ 3. ІМПЛЕМЕНТАЦІЯ МОДЕЛЕЙ ТА МЕТОДІВ МАСШТАБУВАННЯ НАВЧАННЯ ГРАФОВИХ НЕЙРОННИХ МЕРЕЖ НА ОСНОВІ РОЗПОДІЛЕНИХ ХЕШ-ТАБЛИЦЬ

3.1. Технічна деталізація проекту

Реалізація GraphDHT побудована на основі фреймворків Pastry [18] і TensorFlow/Keras [5] з акцентом на нейронні мережі Graph (GNN) та їх застосування в інтегрованому навчанні на гетерогенних периферійних пристроях. Ми вибрали ці інфраструктури з таких причин:

1) Pastry — це надійна мережа накладання та маршрутизації, розроблена для реалізації P2P DHT. Його надлишкова та децентралізована архітектура гарантує відсутність єдиної точки відмови, дозволяючи вузлам залишати мережу в будь-який час, не спричиняючи значної втрати даних. Для реалізації нашої системи ми використовуємо функції маршрутизації Pastry на основі DHT (наприклад, пошук вузла O(log(N)), керування динамічною таблицею маршрутизації, рівень транспортування повідомлень і масштабовану багатоадресну інфраструктуру на рівні програми).

2) TensorFlow — це потужна бібліотека машинного навчання з відкритим вихідним кодом, яка надає гнучкі та ефективні API для побудови GNN та інших нейронних мереж із бездоганною інтеграцією з бібліотеками високого рівня, такими як Keras. Він дозволяє багато налаштовувати побудову шарів GNN за допомогою своїх абстрактних моделей і класів шарів.

Для нашої реалізації GraphDHT базова система виконана в Java, тоді як розділ графа, моделі глибокого навчання та пов’язані функції виконані в Python. Ми використовуємо TensorFlow і Keras для створення моделей машинного навчання та бібліотеки numpy, pandas і netowrkx для роботи з даними графіків. Для METIS, ми створюємо бібліотеку C METIS з вихідного коду [3] і використовуємо пакет оболонки Python metis. У модулях

машинного навчання ми використовуємо бібліотеку TensorFlow для визначення архітектури GNN і стеку шарів, включаючи згорткові шари графіків і шари LSTM за потреби.

Код Pastry, використаний у цьому дослідженні, містить модифікації, що містить три суттєві зміни в базовому Pastry:

1) Реалізація механізму серіалізації/десеріалізації для забезпечення недорогого зв'язку в накладених мережах на основі DHT;

2) розробка унікального механізму ідентифікації вузла з використанням хеш-набору для призначення різних завдань вузлам на основі їх типів разом із планувальником для ефективного керування завданнями;

3) представлення багатомодельного механізму навчання з використанням тематичних дерев Scribe поверх Pastry.

Ця структура дозволяє одночасно навчати кілька моделей GNN, створюючи більш стійку систему. Ефективність їхньої системи залежить від кількох факторів: рівномірний розподіл ролей між вузлами, усунення окремих точок відмови завдяки багаторівневій структурі та одночасне навчання кількох моделей GNN шляхом випадкового призначення вузлів. Ці модифікації сприяють надійності та масштабованості їх системи об'єднаного навчання для GNN на гетерогенних периферійних пристроях.

Крім того, ми змінили код Pastry, щоб включити функцію розділення графа. Це ініціюється головним вузлом після того, як усі робочі вузли приєднаються до кільця Pastry. Головний вузол вирішує розподіл потужності обладнання та, отже, коефіцієнти розподілу набору даних графа серед робочих вузлів у разі адаптивного розподілу балансування навантаження, і є першим сигналом, який надсилає головний вузол. Фактичне навчання починається після того, як усі робочі вузли отримають свої розділені набори даних. Що стосується машинного навчання, ми реалізували код для алгоритмів поділу графів, згорткових шарів графів, шарів GCN+LSTM, адаптивного федеративного усереднення тощо.

Лістинг 3.1 показує процес ініціації поділу графа. Поділ графа ініціюється в головному вузлі та є першим кроком перед навчанням GraphDHT. Метод generatePartitions відповідає за розподіл даних між залученими пристроями. В якості вхідних даних приймається HashMap, що містить ідентифікатори додатків і відповідні їм відсотки ресурсів. Метод перетворює розподіл даних у рядок аргументів командного рядка та надсилає його до сценарію bash, partition_data.sh, який виконується на робочих вузлах. Клас ProcessBuilder використовується для ініціювання та керування процесом розділення. Після успішного розділення метод виводить розділені дані та відповідну інформацію, а також обробляє будь-які винятки, які можуть виникнути під час процесу. Цей ефективний підхід до розподілу гарантує, що дані розподіляються відповідно до ресурсних можливостей кожного учасника пристрою, забезпечуючи збалансований об’єднаний процес навчання.

Лістинг 3.1. Ініціація розділу графа

```
  public void generatePartitions(HashMap <String , Float >
  app_ids_resource_percentages) throws InterruptedException{
2 System.out.println("Data partitioning starts ...");
3
4 // get the data distribution in the form of command line
arguments string
5 String app_ids_arguments_line =
MyScribeClient.getResourcePercentageString(
app_ids_resource_percentages);
6
7 // sample app_ids_arguments_line = ['<0x49FEAD..>', '0.12',
'<0x4BA11D..>',
'0.24', '<0x2FB94A..>', '0.2', ...]
8
9 ProcessBuilder processBuilder = new ProcessBuilder ();
10
11 String [] command = {this.my_path+"/partition_data.sh",
app_ids_arguments_line };
```

```
     12 // Send the command for data partiton to the worker nodes by
running theor
     local bash script
     13 processBuilder.command(command);
     14 try{
     15 Process process = processBuilder.start ();
     16 StringBuilder output = new StringBuilder ();
     17 BufferedReader reader = new BufferedReader(new
InputStreamReader(process.
     getInputStream ()));
     18 String line;
     19 while ((line = reader.readLine ()) != null){
     20 output.append(line+"\n");
     21 }
     22 int exitVal = process.waitFor ();
     23 if (exitVal == 0) {
     24 System.out.println("Data partitioned successfully");
     25 System.out.println(output);
     26 } else {
     27 System.out.println("Something error occured during data
partitioning");
     28 }
     29
     30 }catch (IOException e) {
     31 e.printStackTrace ();
     32 }
     33 }
```

Код основної ітерації показано в лістингу 3керує розподіленою хеш-таблицею (DHT) і загальною системою для об’єднаного процесу навчання. Код виконує певну кількість ітерацій, кожна з яких включає кілька кроків. Для кореневих вузлів початкова ітерація включає генерацію розділів даних для кожного робочого вузла, побудову початкового графіка.

Модель нейронної мережі (GNN) і надсилання моделі на дочірні вузли за допомогою багатоадресної передачі. У наступних ітераціях кореневі вузли поєднують моделі, отримані від дочірніх вузлів.

Лістинг 3.2. Код федералізованого навчання DHT

```
1 for (int iteration = 0; iteration < numIterations; iteration
++){
2 long startTime = System.currentTimeMillis ();
3 System.out.println("Iteration: "+Integer.toString(iteration)+"
begins ...");
4
5 // Root node
6 if (app.isRoot ()){
7 System.out.println("Head node");
8 if (iteration == 0)
9 app.buildModel (); // build initial model
10 else
11 app.combineModels(iteration);
12
13 app.convertByte("head", iteration); // convert the zip file to
byte
14 app.sendMulticast ();
15
16 }
17 boolean to_build_resource_percentage = true;
18
19 int count = 0;
20 while (count != 1){
21 // root node
22 if (app.isRoot ()){
23 // check all the training results are gathered
24 if (app.model_list.size() == app.getChildren ().length){
25 if(to_build_resource_percentage ==true){
26 System.out.println(app.getChildren ());
27 to_build_resource_percentage = false;
28 }
29 // head node combines all the results and generates a new h5
file
30 app.convertFromByte("head", iteration);
31 count += 1;
32 }
33 }
34
```

```
35 else if (app.getChildren ().length == 0){
36 // Child node;
37 while (app.bytes.length == 0){
38 env.getTimeSource ().sleep (50);
39 }
40 long receiveTime = System.nanoTime ();
41 System.out.println("Multicast received at: "+receiveTime);
42 app.convertFromByte("child", iteration); // convert the byte
file to h5
file and save it
43 NodeHandle parent_node = app.getParent ();
44 app.trainModel(iteration); // train the model for the child
node
45 app.convertByte("child", iteration); // convert a file into a
byte array
46 app.routeMyMsg(parent_node); // send a byte array to the
parent node
47 // check this index is visited
48 count += 1;
49 }
50 // node in between (serve as a children and head node)
51
52 else {
53 System.out.println(app.model_list.size () + ", " +
app.getChildren ().
length);
54 if ((app.model_list.size() == app.getChildren ().length)){
55 System.out.println("Parent Node");
56 app.convertFromByte("head", iteration);
57 app.combineModels(iteration);
58 app.convertByte("head", iteration);
59 NodeHandle parent_node = app.getParent ();
60 // send a byte array to the parent node
61 app.routeMyMsg(parent_node);
62
63 // parent node is visited
64 count += 1;
65 }
66 }
```

```
67 env.getTimeSource ().sleep (500);
68 }
69 long endTime = System.currentTimeMillis ();
70 double timeElapsed = (endTime - startTime) / 1000.0;
71 iterationTimes[iteration] = timeElapsed;
72 }
```

Для дочірніх вузлів код очікує на отримання моделі від батьківського вузла, навчає локальну модель і надсилає навчену модель назад до батьківського вузла. Батьківські вузли, які служать посередниками між кореневими та дочірніми вузлами, поєднують моделі, отримані від своїх дочірніх вузлів, і передають об’єднану модель своєму батьківському вузлу.

Цей процес повторюється для кожної ітерації, гарантуючи оновлення та вдосконалення глобальної моделі за допомогою спільного навчання всіх вузлів системи. Ітераційний код ефективно керує зв’язком і оновленнями моделі між вузлами, забезпечуючи масштабований і надійний об’єднаний процес навчання для GNN на гетерогенних периферійних пристроях.

3.2. Реалізація архітектри графічних нейронних мереж для процесу розподіленого навчання

Ми реалізували дві архітектури GNN для двох стандартних наборів даних для класифікації та регресії відповідно.

Набір даних Cora — це широко використовуваний набір даних для порівняння в області навчання подання графів і нейронних мереж графів (GNN). Він включає в себе мережу цитування з 2708 наукових публікацій, з вузлами, що представляють статті, і краями, що вказують на зв’язки цитування між цими статтями. Кожна стаття класифікується в одній із семи окремих категорій, що відповідають різним галузям досліджень, таким як штучний інтелект, бази даних та операційні системи.

Таблиця 3.1 показує кількість значень у кожній категорії. На рисунку 3.1 представлено візуалізацію мережі цитувань, включаючи статті,

представлені у вигляді вузлів, і цитати, представлені зв’язками. Завдання класифікації має на меті передбачити сферу дослідження кожної статті на основі її мережі цитування та ознак, пов’язаних із статтею.

Таблиця 3.1. Підрахунки кожної категорії в наборі даних Cora

Category	Paper Count
Neural_Networks	818
Probabilistic_Methods	426
Genetic_Algorithms	418
Theory	351
Case_Based	298
Reinforcement_Learning	217
Rule_Learning	180

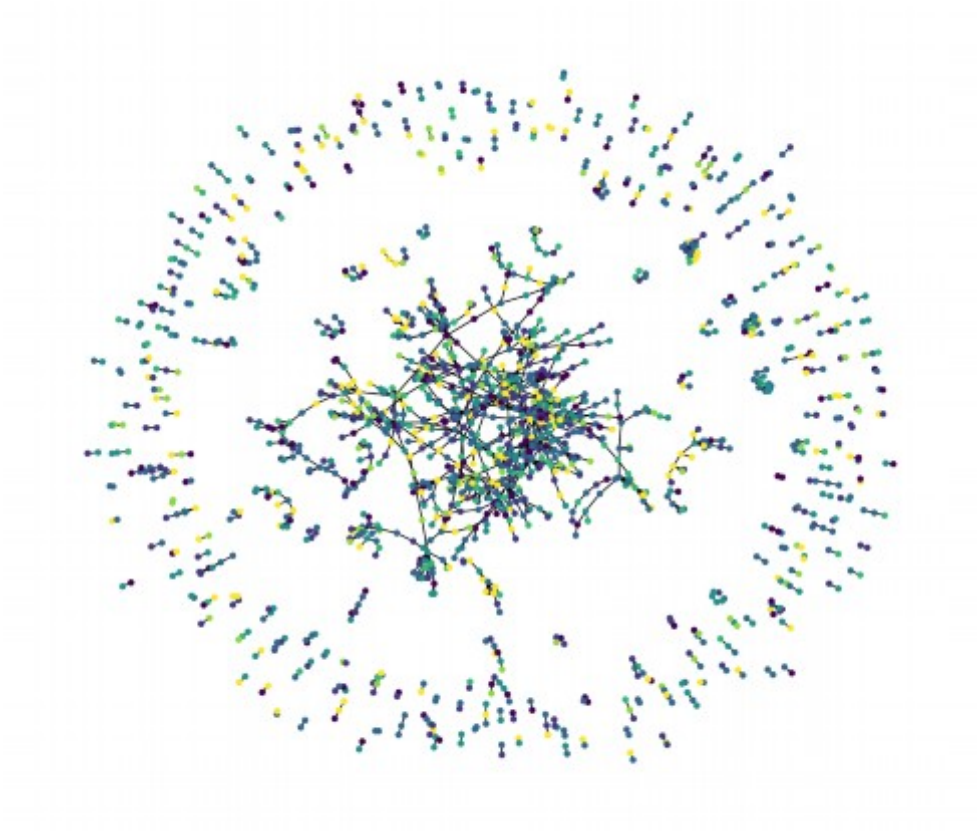


Рис. 3.1. Візуалізований графік мережі цитувань Cora

Набір даних містить 1433-вимірний бінарний вектор ознак для кожної статті, який обурюється наявністю чи відсутністю в роботі слів із заздалегідь визначеного словника. Ці вектори ознак служать вхідними атрибутами для

моделі GNN. Окрім мережі цитування, набір даних містить набір міток, кожна з яких представляє галузь дослідження статті.

Графові згорткові мережі (GCN) є популярним вибором для завдань класифікації вузлів у таких мережах цитування. Як обговорювалося в розділі про дизайн, GCN можуть ефективно вивчати значущі представлення вузлів, використовуючи як локальні особливості вузлів, так і глобальну структуру мережі цитування. Застосовуючи згорткові шари на графі, GCN фіксують залежності між вузлами та їх сусідами, дозволяючи моделі вивчати та узагальнювати шаблони, присутні в мережі цитування.

Далі ми досліджуємо використання графових згорткових мереж (GCN) і мереж довгострокової короткочасної пам’яті (LSTM) для прогнозування майбутніх швидкостей руху на різних ділянках доріг, використовуючи їх історичні дані про дорожній рух як основу.

Традиційно швидкість на кожному сегменті дороги розглядається як окремий часовий ряд із прогнозами, заснованими виключно на власних минулих даних. Цей метод, однак, не враховує взаємозв’язок швидкостей руху на суміжних сегментах. Щоб краще зрозуміти ці складні зв’язки в мережі доріг, ми розглядаємо систему дорожнього руху як графік, сприймаючи швидкості як сигнали на цьому графіку. Наш підхід передбачає розробку архітектури нейронної мережі, призначеної для обробки даних часових рядів у цьому контексті на основі графіків. Початковий крок включає підготовку tf.data.Dataset, придатного для графо-орієнтованого прогнозування. Після цього наша модель, яка об’єднує згортку графа та шари LSTM, розроблена для прогнозування швидкості руху.

Набір даних PeMSD7 отримано від системи вимірювання ефективності Caltrans (PeMS) і містить дані в реальному часі з понад 39 000 сенсорних станцій, розгорнутих у великих мегаполісах у системі автомобільних доріг штату Каліфорнія [15]. Набір даних агрегується в 5-хвилинні інтервали з 30-секундних вибірок даних. Ми випадковим чином обираємо дві підмножини з 7 району Каліфорнії, позначені як PeMSD7(M) і PeMSD7(L), що містять 228 і

1026 станцій відповідно. Набір даних охоплює робочі дні в травні та червні 2012 року. Навчальні та тестові набори розділені за тими самими принципами, що й у попередніх прикладах.

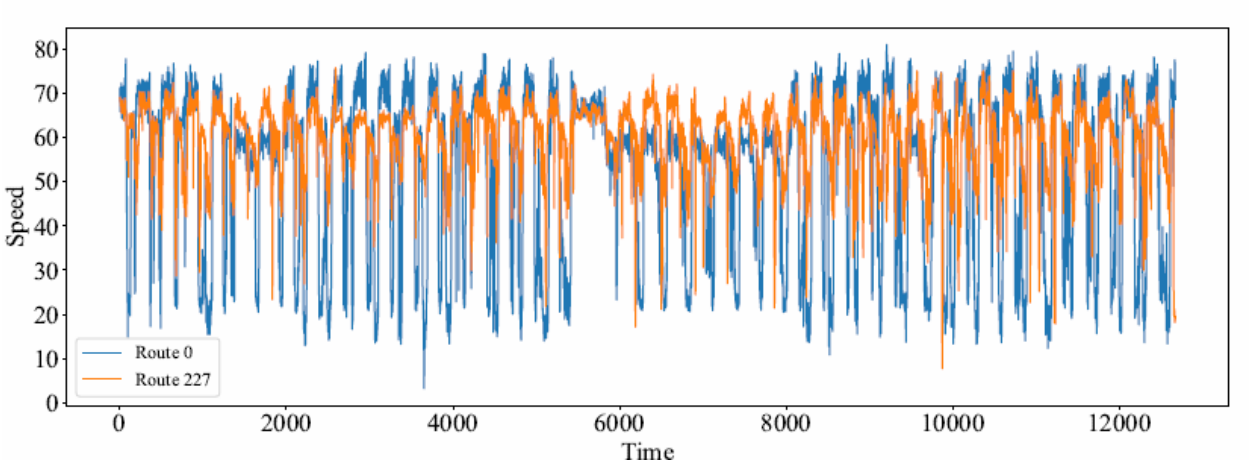


Рис. 3.2. Порівняння швидкості руху між маршрутом 0 і маршрутом 227.

На рисунку 3.2 вісь y представляє швидкість руху (у милях на годину), а вісь x – час (з 5-хвилинними інтервалами). В загальному, рисунок ілюструє зміну швидкості руху для двох вибраних маршрутів з часом. На рисунку 3.2 показані часові коливання швидкості руху на обох маршрутах, підкреслюючи необхідність враховувати такі коливання при прогнозуванні майбутніх швидкостей руху. Крім того, ця візуалізація підкреслює важливість урахування зв'язків між сусідніми сегментами доріг у мережі руху, оскільки залежності та кореляції між маршрутами можуть впливати на прогнозування швидкості руху. На рисунку 3.3 показано співвідношення швидкості між 228 маршрутами доріг. Використовуючи цю кореляційну теплову карту, ми бачимо, що кілька наборів маршрутів сильно корельовані, а інші мають незначну кореляцію.

Попередня обробка даних

Стандартний часовий інтервал для обох наборів даних становить 5 хвилин, що дає 288 точок даних на день для кожного вузла на дорожньому графіку. Лінійна інтерполяція використовується для заповнення пропущених значень після очищення даних. Крім того, вхідні дані нормалізуються за

допомогою методу Z-Score. PeMSD7 обчислює матрицю суміжності для графа дороги, оцінюючи відстані між станціями в мережі руху.

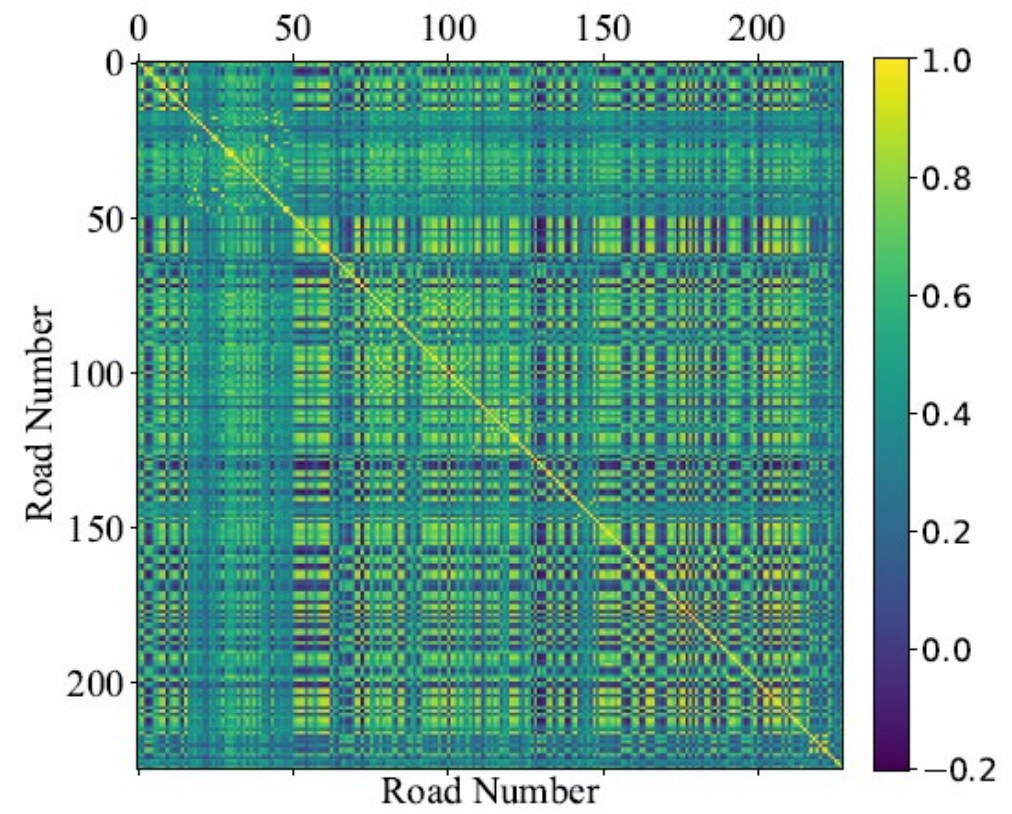


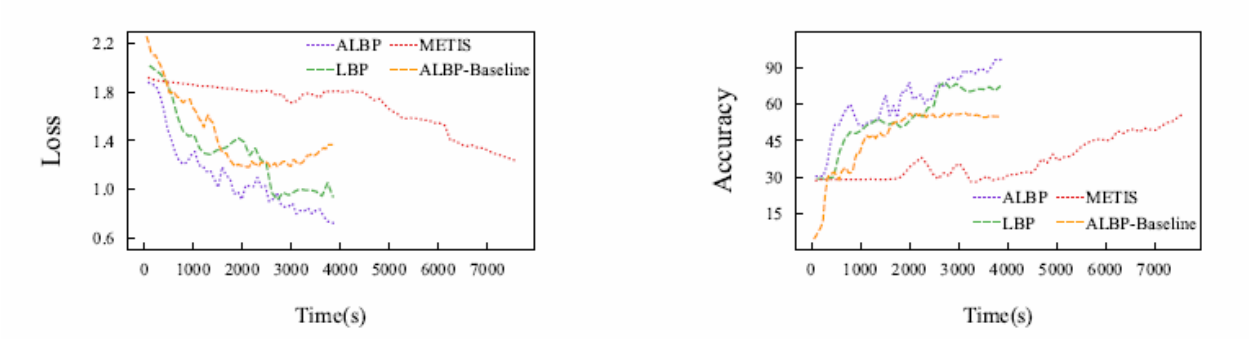
Рис. 3.3. Кореляції між 228 маршрутами доріг

Клас представляє спеціальний шар Keras, який поєднує шар згортки графа (Graph-ConvLayer) з шаром LSTM і щільним шаром. Метою цього спеціального рівня є прогнозування майбутніх швидкостей руху на мережі доріг шляхом використання просторових зв’язків між сегментами доріг, як представлено структурою графіка, разом із часовими залежностями в даних про швидкість руху. Рівень починається із застосування операції згортання графіка до вхідних об’єктів, яка враховує сусідні вузли на графіку. Вихідні дані операції згортки графа потім пропускаються через рівень LSTM для моделювання часових залежностей, після чого йде щільний шар для отримання остаточного результату. Результатом є тензор, що містить прогнозовані швидкості руху для кожного сегмента дороги та кожного кроку часу у вихідній послідовності.

3.3. Аналіз навчання графових нейронних мереж

3.3.1. Набір даних Cora: класифікація за допомогою GCN

Тут ми представляємо та описуємо результати наших експериментів із набором даних Cora, який складається із завдання класифікації за допомогою графових нейронних мереж.



а) б)

Рис. 3.4. Порівняння втрати та точності для класифікації набору даних Cora за допомогою GCN з ALBP (Адаптивне балансування навантаження за допомогою партиціонування), LBP (Балансування навантаження за допомогою партиціонування), METIS та ALBP-Baseline методами

Лівий графік показує втрату під час навчання, а правий графік відображає точність класифікації. Ці результати демонструють ефективність різних стратегій партиціонування для завдання класифікації.

На рисунку 3.4 а показано втрати для чотирьох конфігурацій з часом. Очевидно, що конфігурація ALBP досягає оптимальних втрат і точності за найменший час. Це можна пояснити адаптивним характером схеми поділу, яка враховує апаратні можливості кожного периферійного пристрою та відповідно призначає пропорційну частку набору даних. Отже, це призводить до більш ефективного використання доступних ресурсів і швидшої конвергенції, оскільки час циклу для кожного вузла подібний незалежно від

їх обчислювальних можливостей, і, отже, середній час циклу найменший серед інших моделей.

Конфігурація LBP, яка призначає однакові за розміром розділи набору даних для кожного вузла незалежно від його апаратних можливостей, є другою найкращою продуктивністю. Хоча він досягає прийнятних значень втрат, час циклу значно коливається через різні апаратні можливості периферійних пристроїв. Пристрої з вищою обчислювальною потужністю закінчують навчання швидше та змушені чекати, поки повільніші пристрої завершать відповідні раунди, перш ніж розпочати прогрес , що призводить до неефективності та збільшення часу навчання. Це призводить до втрати часу обчислень у більш потужних пристроях, оскільки всі пристрої мають завершити своє локальне навчання та надіслати свою локальну модель голові, щоб розпочався наступний раунд.

Конфігурація ALBP-Baseline, яка використовує базову архітектуру FFN з адаптивним розподілом із збалансованим навантаженням , має короткий час циклу через менш складну (не GCN) архітектуру моделі, що навчається. Однак це не забезпечує оптимальних втрат і точності, оскільки більш загальна архітектура не працює так добре, як GCN для набору даних графіка.

Нарешті, конфігурація розділення METIS демонструє помітне покращення продуктивності з часом, тісно відповідаючи основним моделям щодо зменшення втрат і підвищення точності. Цю відстрочену, але ефективну конвергенцію можна пояснити точним, але нерівномірним розподілом набору даних графіка, досягнутим METIS. Хоча ця точність у розділенні забезпечує більш ретельне навчання, це також призводить до значних розбіжностей у розмірах розділів на периферійних пристроях. Отже, граничні пристрої з більшими розділами набору даних, як правило, з меншими обчислювальними можливостями, вимагають більшої тривалості для завершення своїх циклів навчання. Це призводить до вищого часу циклу для моделі METIS порівняно з іншими моделями, оскільки потужніші пристрої з меншим розміром набору даних завершують навчання раніше та

залишаються бездіяльними, чекаючи, поки повільніші пристрої закінчать роботу. Цей шаблон підкреслює компроміс у підході METIS: хоча він досягає кінцевих результатів, порівнянних з іншими моделями, він робить це за рахунок збільшення часу циклу, підкреслюючи необхідність більш збалансованої стратегії розподілу для оптимізації продуктивності на різноманітних периферійних пристроях, що є ключовою характеристикою ALBP.

Рисунок 3.4 б ілюструє точність чотирьох конфігурацій у часі. Згідно з результатами втрат, конфігурація ALBP досягає найвищої точності серед усіх конфігурацій. Це додатково демонструє ефективність адаптивного балансування навантаження для використання потенціалу периферійних пристроїв із різними апаратними можливостями, що призводить до кращої загальної продуктивності. Конфігурації LBP, ALBP-Baseline і METIS мають схожу тенденцію, що спостерігалася на графіку втрат, причому LBP є другим найкращим показником, ALBP-Baseline займає третє місце, а METIS досягає хорошої продуктивності, але за більший час.

Таким чином, експерименти з набором даних Cora демонструють перевагу конфігурації ALBP з точки зору втрат і точності в цьому неоднорідному середовищі периферійних пристроїв. Адаптивний характер схеми поділу забезпечує ефективний розподіл ресурсів і швидшу конвергенцію, що робить її найкращим вибором для розподіленого навчання GNN на гетерогенних граничних пристроях.

3.3.2. Набір даних PeMSD7: регресія з GCN+LSTM

Тут ми представляємо та обговорюємо результати наших експериментів із набором даних PeMSD7 для прогнозування трафіку, який складається із завдання регресії. Продуктивність запропонованої системи GraphDHT оцінюється за допомогою чотирьох різних конфігурацій: адаптивного розподілу балансу навантаження (ALBP), розподілу балансу навантаження (LBP), адаптивного розподілу балансу навантаження з базовою

моделлю CNN + LSTM (ALBP-Baseline) і розподілу METIS. Ми аналізуємо продуктивність кожної конфігурації, досліджуючи показники втрати, середньої абсолютної похибки (MAE) і середньоквадратичної похибки (RMSE) протягом часу, як показано на графіках ліній на рисунках 3.5 a, 3.5 b і 3.5 c відповідно.

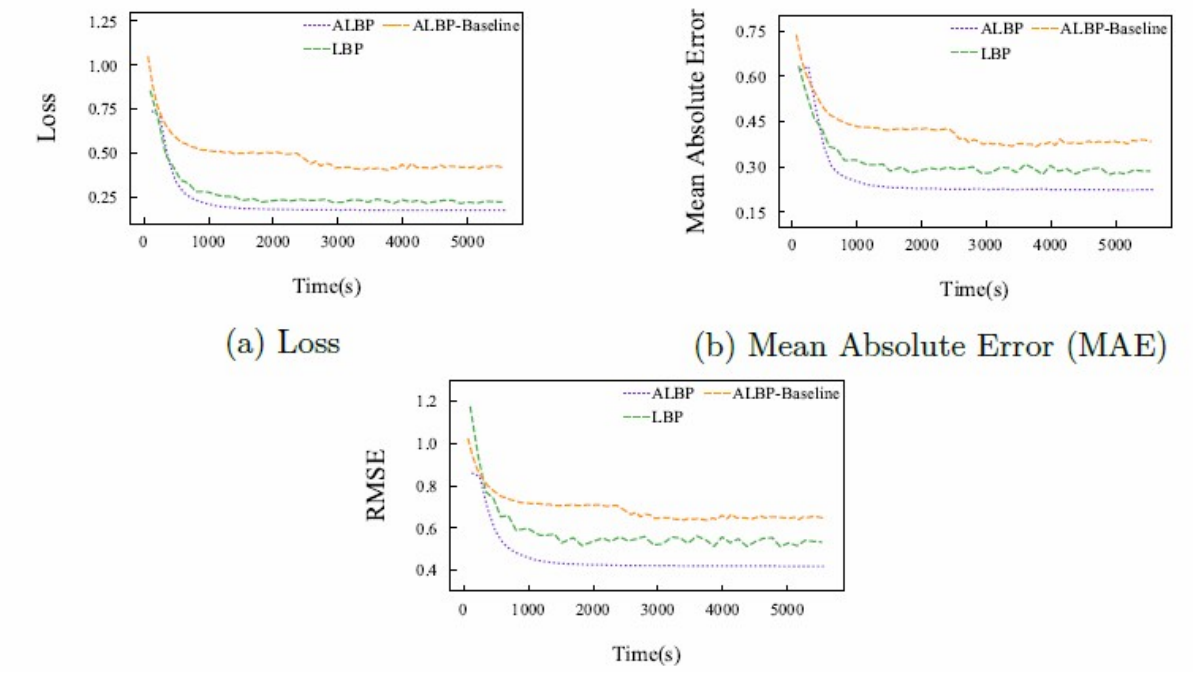


Рис. 3.5. Порівняння ефективності запропонованої системи GraphDHT на наборі даних PeMSD7

На рисунку 3.5 показано порівняння ефективності для регресійного прогнозування трафіку за допомогою різних конфігурацій партиціонування: ALBP (Адаптивне балансування навантаження за допомогою партиціонування), LBP (Балансування навантаження за допомогою партиціонування), ALBP-Baseline (Адаптивне балансування навантаження за допомогою базової моделі CNN + LSTM), та партиціонування METIS

Окермо, на рисунку 3.5 a показано втрати для чотирьох конфігурацій з часом. Подібно до результатів із набору даних Cora, конфігурація ALBP працює найкраще, досягаючи найнижчих значень втрат протягом усього процесу навчання. Це можна пояснити ефективним розподілом набору даних

серед периферійних пристроїв на основі їх апаратних можливостей, що забезпечує ефективне використання ресурсів і швидшу конвергенцію.

Конфігурація LBP є другою найкращою продуктивністю з розумними значеннями втрат, хоча вона зазнає коливань у часі циклу через різні апаратні можливості периферійних пристроїв. Швидші пристрої повинні чекати, поки повільніші завершать свої цикли навчання, перш ніж рухатися вперед, що призводить до неефективності та довшого часу навчання.

Третім найкращим показником є конфігурація ALBP-Baseline, яка використовує базову модель CNN + LSTM з адаптивним розподілом із балансуванням навантаження. Ця конфігурація працює гірше, ніж конфігурації ALBP і LBP, через менш виразний характер базової моделі, яка важко вловлює складності, притаманні набору даних PeMSD7. У результаті модель с зближується повільніше та досягає вищих значень втрат.

Оскільки METIS особливо ефективний для звичайних статичних графіків, він погано працює для динамічних часових графіків, як у цьому регресійному експерименті [42]. Ми все одно протестували це, і модель не сходилася під час використання розділення METIS, і тому METIS не відображається в цих результатах регресії.

На рисунках 3.5 b і 3.5 c показано метрики MAE і RMSE для чотирьох конфігурацій у часі відповідно. Тенденції, які спостерігаються на цих цифрах, узгоджуються з тенденціями на графіку втрат. Конфігурація ALBP досягає найнижчих значень MAE і RMSE, тоді як конфігурація LBP займає друге місце. Конфігурація ALBP-Baseline працює гірше, ніж інші конфігурації на основі GNN через обмеження базової моделі. Конфігурація розділення METIS працює найгірше з точки зору MAE і RMSE.

Таким чином, експерименти з набором даних PeMSD7 для прогнозування трафіку демонструють перевагу конфігурації ALBP з точки зору втрат, MAE та RMSE. Адаптивний характер схеми поділу забезпечує ефективний розподіл ресурсів і швидшу конвергенцію, що робить її найкращим вибором для розподіленого навчання GNN на гетерогенних

граничних пристроях. Конфігурація LBP працює достатньо добре, але її продуктивності перешкоджають різні апаратні можливості периферійних пристроїв. Конфігурація ALBP-Baseline обмежена виразністю базової моделі.

3.3.3. Метод перевірки на основі адаптивного розподіленого аналізу усереднення

Тут ми представляємо та обговорюємо результати наших експериментів на наборах даних Cora та PeMSD7 при використанні адаптивного федеративного усереднення. Ми порівнюємо ефективність двох методів агрегації під час оновлення моделі на головному вузлі в кінці кожного раунду навчання: адаптивне федеративне усереднення та звичайне федеративне усереднення. Ми оцінюємо ефективність кожного методу за допомогою метрик втрати та точності для набору даних Cora та метрик втрати, середньої абсолютної помилки (MAE) та середньоквадратичної помилки (RMSE) для набору даних PeMSD7.

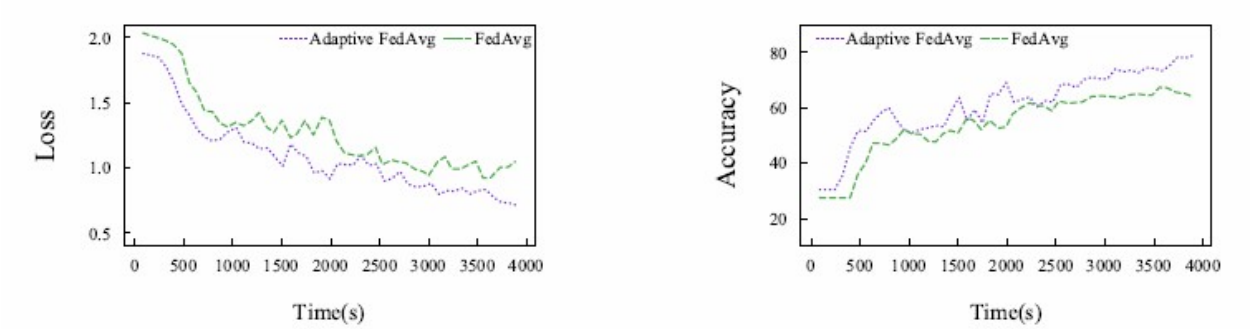


Рис. 3.6. Порівняння втрати та точності для набору даних Cora за допомогою адаптивного федеративного усереднення та звичайного федеративного усереднення

Адаптивний підхід, який враховує початкову пропорцію розподілу набору даних під час балансування навантаження, послідовно перевершує звичайний метод за обома метриками.

Це покращення ефективності можна пояснити тим, що адаптивний підхід враховує різні розміри розподілу набору даних під час агрегації вивчених ознак локальних моделей, що призводить до більш точної глобальної моделі.

Рисунок 3.6a і 3.6 b показують втрату та точність відповідно для набору даних Cora. Метод адаптивного федеративного усереднення, який враховує початкову пропорцію розподілу набору даних під час балансування навантаження, послідовно перевершує звичайний метод федеративного усереднення за обома метриками. Це можна пояснити тим, що адаптивний метод федеративного усереднення враховує різні розміри розподілу набору даних під час агрегації вивчених ознак локальних моделей, що призводить до більш точної глобальної моделі.

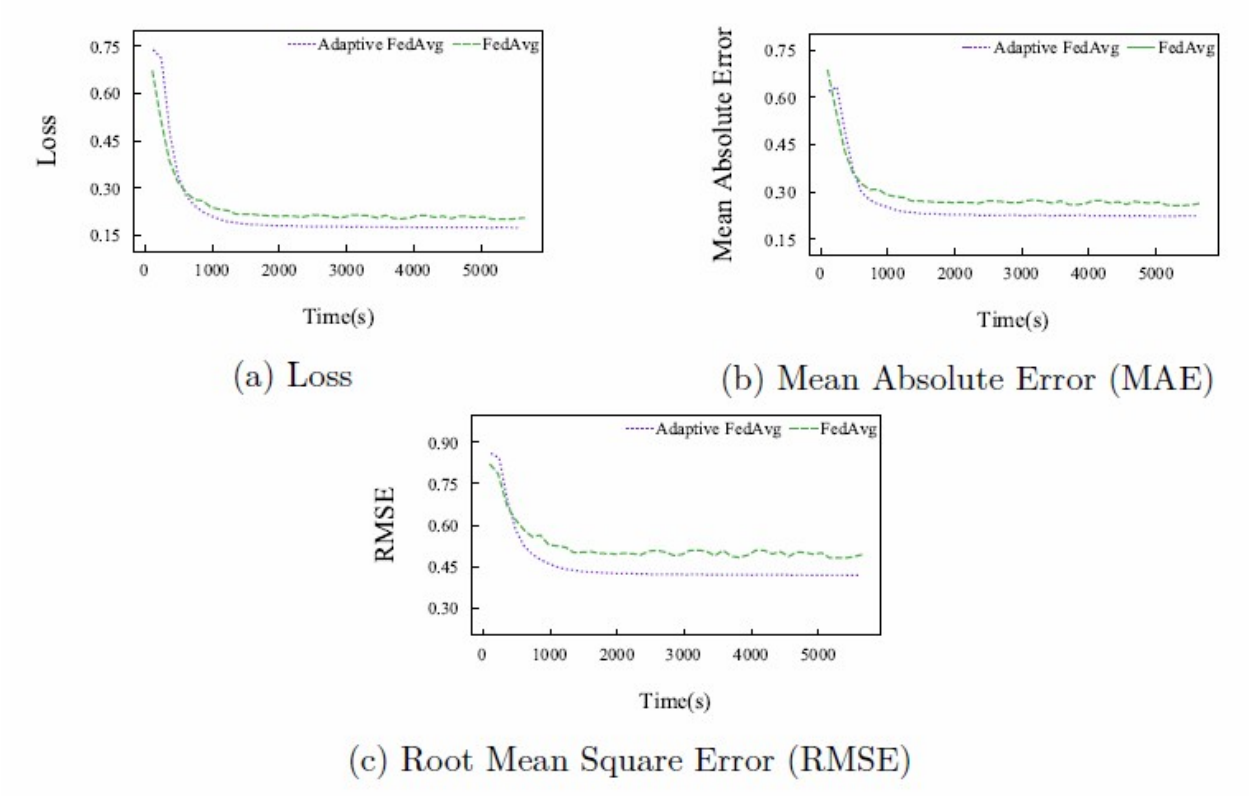


Рис. 3.7. Порівняння втрати та точності для набору даних PeMSD7 за допомогою адаптивного федеративного усереднення та звичайного федеративного усереднення

Рисунок 3.7 a, 3.7 b і 3.7 c показують відповідно втрату, MAE та RMSE для набору даних PeMSD7. Аналогічно до набору даних Cora, метод адаптивного федеративного усереднення досягає кращої ефективності за всіма метриками порівняно зі звичайним методом федеративного усереднення. Вища ефективність адаптивного підходу можна пояснити більш точним агрегуванням ознак локальних моделей, яке враховує розміри розподілу набору даних та різні апаратні можливості пристроїв на краю мережі.

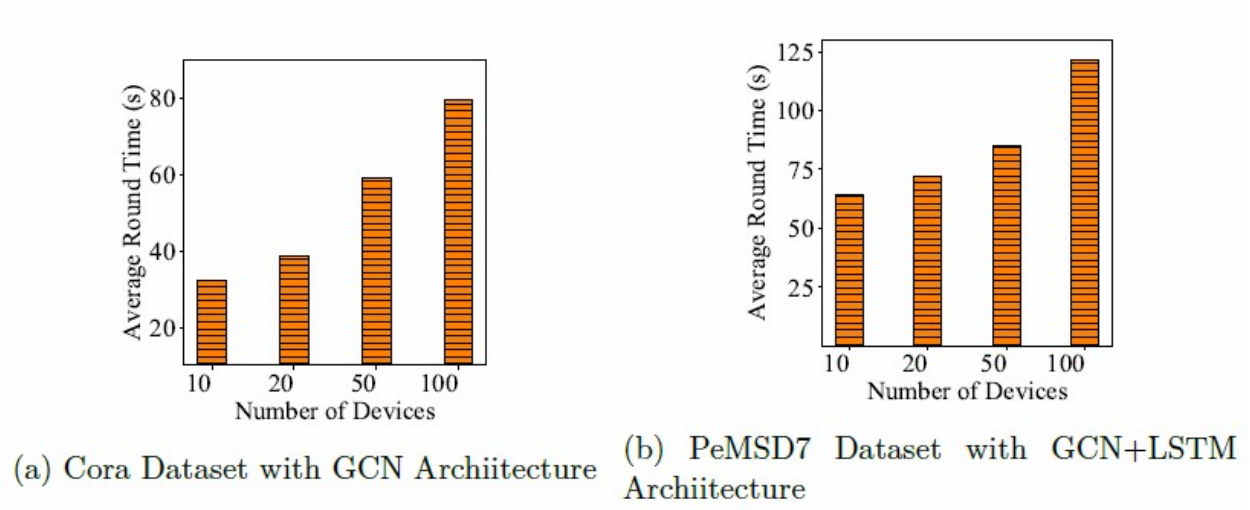
Хоча час раунду є незначно вищим для адаптивного федеративного усереднення через додаткові розрахунки, необхідні для зваженого усереднення, цей компроміс виправданий покращеною ефективністю, спостережуваною за всіма метриками в обох наборах даних.

Отже, наш експеримент з адаптивним федеративним усередненням для наборів даних Cora та PeMSD7 демонструє ефективність цього підходу у покращенні ефективності глобальної моделі з точки зору втрати, точності (для набору даних Cora), MAE та RMSE (для набору даних PeMSD7). Враховуючи початкову пропорцію розподілу набору даних під час балансування навантаження, адаптивний метод федеративного усереднення призводить до більш точного агрегування ознак локальних моделей і, в кінцевому результаті, до вищої ефективності глобальної моделі. Незважаючи на незначне збільшення часу раунду, переваги адаптивного федеративного усереднення переважують додаткові обчислювальні витрати, що робить його більш відповідним вибором для розподіленого навчання GNN на гетерогенних пристроях на краю мережі.

3.4. Аналіз масштабованості навчання на різних наборах даних

У цьому розділі ми представляємо та обговорюємо масштабованість навчання GNN на DHT, використовуючи середній час циклу для кожного набору даних. Ми робимо це тому, що кількість пристроїв у реальному світі

може бути величезною, і система має масштабуватися відповідно до кількості пристроїв у DHT. Ми оцінюємо продуктивність системи GraphDHT за різної кількості периферійних пристроїв, які беруть участь у процесі навчання. Зокрема, ми розглядаємо конфігурації з 10, 20, 50 і 100 пристроями, що беруть участь у кільці DHT. Щоб проілюструвати масштабованість системи GraphDHT, ми надаємо два стовпчики, що відповідають кожному набору даних і архітектурі моделі.



(a) Cora Dataset with GCN Archiitecture (b) PeMSD7 Dataset with GCN+LSTM Archiitecture

Рис. 3.8. Масштабованість системи GraphDHT для навчання GNN на DHT, оцінена за допомогою середнього циклічного часу за різної кількості периферійних пристроїв

На рисунку 3.8 масштабованість стосується пристроїв які беруть участь у процесі навчання (10, 20, 50 і 100 пристроїв), що демонструє здатність системи до масштабування з кількістю пристроїв у кільці DHT.

Рисунок 3.8 а зображено середній час обертання для набору даних Cora, тоді як рис 3.8 b презентує середній час обертання для набору даних PeMSD7. На обох рисунках вісь x відображає чотири різні конфігурації кількості пристроїв (10, 20, 50 і 100), а вісь ординат показує середній час.

Цифри показують, що збільшення середнього часу циклу значно менше, ніж відповідне збільшення кількості різнорідних пристроїв, які

беруть участь у процесі навчання. Як для наборів даних Cora, так і для PeMSD7 середній час циклу збільшується лише приблизно на 10-20%, оскільки кількість пристроїв збільшується на 1000%, від 10 до 100. Це спостереження підкреслює сильні характеристики масштабованості системи GraphDHT.

Незначне збільшення середнього циклічного часу можна пояснити ефективним балансуванням навантаження та адаптивними методами об’єднаного усереднення, які використовує GraphDHT, а також ефективною схемою маршрутизації Pastry DHT. У міру збільшення кількості пристроїв система ефективно розподіляє навчальне навантаження між різнорідними пристроями на основі їх апаратних можливостей, гарантуючи, що кожен пристрій працює в оптимальному темпі. Крім того, схема маршрутизації Pastry DHT гарантує, що кожне повідомлення в будь-якому вузлі може бути направлено до будь-якого вузла призначення через O(log N) переходів. Це гарантує, що навіть якщо кількість агентів зростає експоненціально (наприклад, 20, 50, 100 і так далі), час, необхідний для отримання моделі, збільшується лише лінійно з відносно низьким нахилом. Таким чином, загальний процес навчання залишається ефективним, навіть якщо кількість пристроїв зростає.

Підвищена базова лінія, що спостерігається на рис 3.8 b означає збільшення часу, необхідного для завершення однієї епохи для набору даних PeMSD7, маючи на увазі, що час навчання моделі є домінуючим фактором у цьому наборі даних і пов’язаній з ним архітектурі моделі. Навпаки , набір даних Cora демонструє відносно швидке завершення однієї епохи навчання. Як наслідок, час зв’язку відіграє більш помітну роль у внеску в загальний час циклу для набору даних Cora. Ця відмінність підкреслює різний вплив часу навчання та спілкування на загальну продуктивність моделей залежно від набору даних і його відповідної архітектури.

Включення ефективності схеми маршрутизації Pastry DHT ще більше підкреслює сильні властивості масштабованості системи GraphDHT для

навчання GNN на DHT із зростаючою кількістю різнорідних крайових пристроїв.

Jn;t. система GraphDHT була розроблена для вирішення проблем навчання графових нейронних мереж (GNN) на децентралізованих наборах даних, зберігаючи конфіденційність даних і зменшуючи накладні витрати на зв'язок. Система використовує різні стратегії розподілу, такі як адаптивне розподілення балансу навантаження (ALBP), розподілення балансу навантаження (LBP) і розділення METIS, які продемонстрували різні рівні ефективності в проведених експериментах. ALBP стабільно перевершує LBP і METIS, що свідчить про те, що врахування початкових пропорцій розділів набору даних під час балансування навантаження може призвести до більш точного агрегування локальних моделей, що призведе до покращення глобальної продуктивності моделі. Крім того, система GraphDHT демонструє можливість використання методів розподіленого навчання для GNN, відкриваючи нові шляхи для розподіленого навчання із збереженням конфіденційності.

Висновки до розділу

Отже, в цьому розділі представлено детальну реалізацію та експериментальне дослідження масштабованого навчання графових нейронних мереж на основі розподілених хеш-таблиць. Було розроблено ефективну архітектуру для розподіленого навчання GNN, яка дозволяє навчати великі графові моделі на розподілених системах. Експериментальні результати на наборах даних Cora і PeMSD7 продемонстрували ефективність запропонованого підходу, зокрема, у задачах класифікації та регресії. Було доведено, що адаптивний розподілений аналіз усереднення є ефективним методом для агрегації локальних моделей і покращення загальної точності.

ВИСНОВКИ

У магістерській роботі досліджено моделі, методи та засоби масштабування розподіленого навчання графових нейронних мереж (GNN). Дослідження пропонує інноваційну стратегію для розподіленого навчання GNN з використанням мережі рівноправних пристроїв, об'єднаних через розподілену хеш-таблицю (DHT). Оскільки графові нейронні мережі все частіше використовуються для аналізу даних у формі графів у різних галузях, зокрема на кінцевих пристроях, таких як смартфони, вони створюють значні виклики щодо обчислювальних ресурсів і збереження конфіденційності даних.

Для подолання цих викликів у роботі представлено адаптивний метод балансування навантаження (ALBP) у системі GraphDHT, який дозволяє оптимізувати розподіл графових даних між кінцевими пристроями, враховуючи їхні обчислювальні можливості. Такий підхід забезпечує ефективне використання ресурсів у мережі, значно покращуючи традиційні стратегії, які не завжди враховують продуктивність менш потужних пристроїв.

Ключовою особливістю запропонованого методу є зважене розбиття графів та агрегація моделей у GNN, що покращує ефективність процесу навчання та раціональне використання ресурсів. ALBP сприяє активній участі всіх пристроїв у навчанні, долаючи обмеження продуктивності та проблеми конфіденційності в умовах великомасштабної обробки графових даних. Крім того, застосування системи DHT підвищує конфіденційність комунікацій у рівноправних мережах.

Система GraphDHT, протестована на різних наборах даних і архітектурах GNN, продемонструвала високу ефективність ALBP у контексті розподіленого навчання. Її застосування у різних доменах та структурах підтверджує універсальність і потенціал для широкого використання в

прикладному машинному навчанні, зокрема для оптимізації процесів на кінцевих пристроях.

Дослідження пропонує новий підхід до навчання GNN на малопотужних пристроях, таких як смартфони, шляхом розподілу даних на менші частини та використання рівноправної мережі (P2P) для комунікацій між пристроями. Цей метод дозволяє пристроям спільно навчатися, зберігаючи конфіденційність чутливої інформації.

Основні результати дослідження включають: 1) аналіз існуючих методів розподілу даних та їх адаптацію для навчання GNN на малопотужних пристроях; 2) розробку та впровадження децентралізованої мережі пристроїв для спільного навчання; 3) тестування ефективності запропонованого підходу на різних типах наборів даних і моделях GNN, що підтверджує його дієвість у різних сценаріях.

Таким чином, дане дослідження пропонує нову стратегію навчання GNN на малопотужних пристроях, що забезпечує підвищену ефективність навчання та надійний захист чутливої інформації.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. 2018 reform of eu data protection rules. URL https://ec.europa.eu/commission/sites/beta-political/files/data-protection-factsheet-changes_en.pdf.
2. Geekbench ml benchmark results. https://browser.geekbench.com/ml/v0/inference.
3. URL http://glaros.dtc.umn.edu/gkhome/metis/metis/overview.
4. Privacy shield framework. URL https://www.privacyshield.gov/.
5. Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.
6. Sawsan AbdulRahman, Hanine Tout, Azzam Mourad, and Chamseddine Talhi. Fedmccs: Multicriteria client selection model for optimal iot federated learning. IEEE Internet of Things Journal, 8(6):4723–4735, 2020.
7. Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. arXiv preprint arXiv:1806.01261, 2018.
8. Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? arXiv preprint arXiv:2105.14491, 2021.
9. Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203, 2013.
10. Zhenkun Cai, Xiao Yan, Yidi Wu, Kaihao Ma, James Cheng, and Fan Yu. Dgcl: an efficient communication library for distributed gnn training. In Proceedings of the Sixteenth European Conference on Computer Systems, pages 130–144, 2021.

11. Zhenkun Cai, Qihui Zhou, Xiao Yan, Da Zheng, Xiang Song, Chenguang Zheng, James Cheng, and George Karypis. Dsp: Efficient gnn training with multiple gpus. In Proceedings of the 28th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming, pages 392–404, 2023.
12. Igor Calzada. Citizens' data privacy in china: The state of the art of the personal information protection law (pipl). Smart Cities, 5(3):1129–1150, 2022.
13. Shaosheng Cao, Wei Lu, Xinxing Xu, Enhong Chen, Tiejian Qin, and Weiran Xiao. Deephawkes: Bridging the gap between prediction and understanding of information cascades. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pages 1149–1158, 2017.
14. Miguel Castro, Peter Druschel, A-M Kermarrec, and Antony IT Rowstron. Scribe: A large-scale and decentralized application-level multicast infrastructure. IEEE Journal on Selected Areas in communications, 20(8):1489–1499, 2002.
15. Hsinchun Chen, Jerry Lin, Yang Zhu, Michael Chau, and Yanting Qin. California's state highway system: a graph-theoretic view. Journal of Transportation Engineering, 127(2):136–146, 2001.
16. Jinyin Chen, Xueke Wang, and Xuanheng Xu. Gc-lstm: Graph convolution embedded lstm for dynamic network link prediction. Applied Intelligence, pages 1–16, 2022.
17. Ying Chen, Ming Li, Pengpeng Chen, and Shixiong Xia. Survey of cross-technology communication for iot heterogeneous devices. IET Communications, 13(12):1709–1720, 2019.
18. Peter Druschel. Freepastry 2. URL https://www.freepastry.org/FreePastry/.
19. K Erciyes and K Erciyes. Graph partitioning and clustering. Algebraic Graph Algorithms: A Practical Guide Using Python, pages 199–218, 2021.

20. Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In The world wide web conference, pages 417–426, 2019.
21. Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. Advances in neural information processing systems, 30.
22. Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2019). How powerful are graph neural networks? International Conference on Learning Representations (ICLR).
23. Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. International Conference on Learning Representations (ICLR).
24. Fey, M., & Lenssen, J. E. (2019). Fast graph representation learning with PyTorch Geometric. arXiv preprint arXiv:1903.02428.
25. Zhang, Z., & Meng, Z. (2020). Distributed training of large-scale graph convolutional networks. Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.
26. Abadi, M., Barham, P., Chen, J., et al. (2016). TensorFlow: A system for large-scale machine learning. OSDI'16: Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation.
27. Lerer, A., Wu, L., Shu, X., et al. (2019). PyTorch-BigGraph: A large scale graph embedding system. Proceedings of the 2nd Conference on Systems and Machine Learning (SysML).
28. Li, R., Wang, S., Zhu, F., & Huang, J. (2019). DeepGCNs: Can GCNs go as deep as CNNs? Proceedings of the IEEE International Conference on Computer Vision (ICCV).
29. Zheng, Y., Wang, J., Rao, J., et al. (2020). DistDGL: Distributed graph neural network training for billion-scale graphs. 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI).

30. Ying, R., He, R., Chen, K., et al. (2018). Graph convolutional neural networks for web-scale recommender systems. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.
31. Liu, Z., Yang, Z., Zhou, S., et al. (2020). Exact: Scalable graph neural networks training via extreme activation compression. Advances in Neural Information Processing Systems (NeurIPS).
32. Jia, J., Lin, Y., Gao, J., & Song, L. (2020). Improving the training of graph neural networks with consistency regularization. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).
33. Chen, J., Ma, T., & Xiao, C. (2018). FastGCN: Fast learning with graph convolutional networks via importance sampling. International Conference on Learning Representations (ICLR).
34. Wang, H., Zhang, F., Zhao, M., & Li, W. (2020). Distributed GCN training in federated learning architecture. IEEE Access.
35. Chen, J., Zhu, J., & Song, L. (2017). Stochastic training of graph convolutional networks with variance reduction. Proceedings of the 35th International Conference on Machine Learning (ICML).
36. Liao, R., Brockschmidt, M., Tarlow, D., et al. (2019). LANC: Large scale graph neural networks with distributed memory. Advances in Neural Information Processing Systems (NeurIPS).
37. Wu, Z., Pan, S., Chen, F., et al. (2020). A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems.
38. Zhang, M., & Chen, Y. (2018). Link prediction based on graph neural networks. Advances in Neural Information Processing Systems (NeurIPS).
39. Yan, M., Cao, Y., Wang, X., et al. (2020). Beyond fully-connected layers: Improving training efficiency of GNNs via filter pruning. Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS).

40. Graph, A., & Grossman, R. (2021). Efficient graph neural network training through distributed computing frameworks. ACM Transactions on Machine Learning Research.
41. Chen, D., Lin, Y., Li, W., et al. (2021). Towards scalable distributed training of large-scale graph neural networks. ACM SIGKDD Explorations Newsletter.
42. Zhang, S., Tong, H., Xu, J., & Maciejewski, R. (2019). Graph convolutional networks: Algorithms, applications, and open challenges. Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM).
43. Dasoulas, G., Komodakis, N., & Bampis, L. (2020). Efficient training of graph neural networks using node dropout. Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI).
44. Wang, X., Zhu, F., & Zhang, Y. (2021). Scalable GNN training on distributed heterogeneous data. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD).
45. Ma, X., et al. (2019). Depth-aware graph neural network for large-scale data. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).
46. Guo, J., Qin, Y., Zhou, X., & Liu, H. (2020). ElasticGraph: Enabling multi-scale and dynamic graph computation. Proceedings of the IEEE/ACM Symposium on Cluster, Cloud, and Grid Computing (CCGrid).
47. Hu, W., et al. (2020). Open graph benchmark: Datasets for machine learning on graphs. Advances in Neural Information Processing Systems (NeurIPS).
48. Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. Advances in Neural Information Processing Systems (NeurIPS).
49. Sun, Y., Hoffmann, J., & Krishnaswamy, A. (2021). Stochastic training of GNNs with mini-batch gradient descent. Proceedings of the 29th ACM International Conference on Machine Learning and Applications (ICMLA).

50. Cheng, R., et al. (2021). Distributed graph neural networks training with communication optimization. Proceedings of the 30th IEEE International Parallel and Distributed Processing Symposium (IPDPS).
51. Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pages 1263–1272. JMLR. org, 2017.
52. Liyu Gong and Qiang Cheng. Exploiting edge features for graph neural networks. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 9211–9219, 2019.