МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В.Н. Каразіна

Факультет: **ННІ Каразінський банківський інститут**
Кафедра: **Інформаційних технологій та математичного моделювання**
Спеціальність: **122 Комп'ютерні науки**
Освітня програма: **Комп'ютерні науки та інформаційні технології в бізнесі**

Група: АК-41б **денна форма навчання**

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

на тему:

«ОПТИМІЗАЦІЯ ВИКОРИСТАННЯ БАЗ ДАНИХ ДЛЯ ЗБЕРІГАННЯ ТА ОБРОБКИ ДАНИХ ДЛЯ НАВЧАННЯ НЕЙРОННИХ МЕРЕЖ»

ЗА НАКАЗОМ № 4601-5/335 ВІД 07 ЛЮТОГО 2025 РОКУ

здобувача вищої освіти **ЛИСЮКА ВАСИЛЯ ВАСИЛЬОВИЧА**

Робота допущена до захисту в ЕК
протокол кафедри ІТММ № 13 від 31.05.2025р.

Завідувач кафедри ІТММ
к.п.н., доцент
_____ Н.І. Стяглик

Науковий керівник
Ph.D з «Комп'ютерних наук»
_____ Д.М. Ковальчук

м. Харків 2025 р

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В. Н. Каразіна

Факультет навчально-науковий інститут "Каразінський банківський інститут"
Кафедра інформаційних технологій та математичного моделювання
Рівень вищої освіти перший (бакалаврський)
Спеціальність 122 Комп'ютерні науки
Освітня програма Комп'ютерні науки та інформаційні технології в бізнесі

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ **Н. І. Стяглик**
Підпис ініціали, прізвище

"08" лютого 2025 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ (ПРОЄКТ)

Лисюка Василя Васильовича
(прізвище, ім'я, по батькові студента)

1. Тема роботи «Оптимізація використання баз даних для зберігання та обробки даних для навчання нейронних мереж»

керівник роботи Ph.D з «Комп'ютерних наук» Ковальчук Д.М.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від "08" лютого 2025 року № 4601-5/335

2. Строк подання студентом роботи 15 травня 2025 року

3. Перелік питань, які потрібно розробити:
У розділі 1: розглянути теоретичні основи машинного навчання та обробки даних

У розділі 2: визначити особливості використання БД у системах машинного навчання

У розділі 3: порівняти між собою обрані підходи зберігання та обробки даних

4. План роботи

№ з/п	Назви етапів роботи
1	Вибір здобувачем теми кваліфікаційної бакалаврської роботи
2	Затвердження плану і завдання кваліфікаційної бакалаврської роботи
3	Здача кваліфікаційної бакалаврської роботи керівнику
4	Підпис кваліфікаційної бакалаврської роботи керівника
5	Підпис кваліфікаційної бакалаврської роботи у нормоконтролера
6	Допуск завідувачем кафедри до захисту кваліфікаційної бакалаврської роботи
7	Захист кваліфікаційної бакалаврської роботи

5. Дата видачі завдання 08 лютого 2025 року

Студент _____ В.В. Лисюк
підпис ініціали, прізвище

Керівник роботи _____ Д.М. Ковальчук
підпис ініціали, прізвище

РЕФЕРАТ
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ «ОПТИМІЗАЦІЯ ВИКОРИСТАННЯ БАЗ ДАНИХ ДЛЯ ЗБЕРІГАННЯ ТА ОБРОБКИ ДАНИХ ДЛЯ НАВЧАННЯ НЕЙРОННИХ МЕРЕЖ»

Лисюка Василя Васильовича

Кваліфікаційна бакалаврська робота містить 54 сторінки, 12 таблиць, 3 рисунка, список літератури з 14 найменувань.

Об'єктом дослідження є процеси оптимізації застосування баз даних у контексті забезпечення ефективного зберігання та обробки великих обсягів даних для потреб машинного навчання.

Предметом дослідження є методи та засоби оптимізації структури баз даних, запитів до них, а також організації потоків даних з баз даних до моделей машинного навчання з метою підвищення продуктивності та ефективності навчального процесу нейронних мереж.

Метою роботи є дослідження та оптимізація процесів зберігання і обробки даних у базах даних, що застосовуються для навчання моделей глибокого навчання.

Для досягнення поставленої мети вирішено наступні завдання:

– у першому розділі проаналізовано теоретичні засади машинного навчання та методи обробки даних;

– у другому розділі визначено специфіку використання баз даних у системах машинного навчання;

– у третьому розділі здійснено порівняльний аналіз обраних підходів до зберігання та обробки даних.

Актуальність дослідження полягає у підвищенні ефективності застосування методів машинного навчання шляхом раціонального вибору платформ та архітектур баз даних для обробки відповідних обсягів інформації.

За результатами дослідження проведено порівняльний аналіз трьох платформ зберігання та обробки даних, адаптованих до потреб навчання нейронних мереж.

Практична новизна полягає у визначенні критеріїв ефективного вибору платформи баз даних залежно від типу завдань машинного навчання.

Одержані результати можуть бути використані під час реалізації систем підготовки даних для моделювання та тестування нейронних мереж.

КЛЮЧОВІ СЛОВА: БАЗИ ДАНИХ, МАШИННЕ НАВЧАННЯ, НЕЙРОННІ МЕРЕЖІ, ОБРОБКА ДАНИХ, ОПТИМІЗАЦІЯ.

ABSTRACT
AT QUALIFICATION BACHELOR WORK
«OPTIMIZING THE USE OF DATABASES FOR STORING AND PROCESSING DATA FOR TRAINING NEURAL NETWORKS»

Vasily Lysiuk

The qualification bachelor's thesis contains 54 pages, 12 tables, 3 figures, a list of references with 14 titles.

The object of the study is the optimization processes of database use in the context of ensuring effective storage and processing of large amounts of data for the needs of machine learning.

The subject of the study is methods and means of optimizing the structure of databases, queries to them, as well as the organization of data flows from databases to machine learning models in order to increase the productivity and efficiency of the training process of neural networks.

The purpose of the work is to study and optimize the processes of storing and processing data in databases used for training deep learning models.

To achieve the set goal, the following tasks were solved:

– in the first section, the theoretical foundations of machine learning and data processing methods are analyzed;

– in the second section, the specifics of the use of databases in machine learning systems are determined;

– in the third section, a comparative analysis of the selected approaches to data storage and processing is carried out.

The relevance of the study is to increase the efficiency of the application of machine learning methods through the rational selection of database platforms and architectures for processing the corresponding amounts of information.

Based on the results of the study, a comparative analysis of three data storage and processing platforms adapted to the needs of training neural networks was carried out.

The practical novelty lies in determining the criteria for effective selection of a database platform depending on the type of machine learning tasks.

The results obtained can be used in the implementation of data preparation systems for modeling and testing neural networks.

KEYWORDS: DATABASES, MACHINE LEARNING, NEURAL NETWORKS, DATA PROCESSING, OPTIMIZATION.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧОК, СИМОВЛІВ І ТЕРМІНІВ

БД – база даних

СКБД – система керування базами даних

ETL – extract, transform, load

SQL – structured query language

HDF5 - hierarchical data format

ML – machine learning

DL – deep learning

ВСТУП

У сучасних умовах стрімкого розвитку інформаційних технологій та штучного інтелекту особливої ваги набуває проблема ефективної обробки великих обсягів даних. Одним із ключових аспектів успішного навчання моделей глибокого навчання є якісне зберігання та своєчасна подача даних. Ефективність функціонування систем машинного навчання значною мірою залежить від організації процесів обробки даних, що передують безпосередньому навчанню моделей. Традиційні підходи до зберігання та обробки даних часто не забезпечують необхідного рівня продуктивності, масштабованості та швидкодії, що особливо критично для глибоких нейронних мереж, які потребують великих обсягів структурованої та неструктурованої інформації. У зв’язку з цим актуальним є дослідження та оптимізація підходів до використання баз даних із урахуванням специфіки задач машинного навчання.

Питання організації процесів зберігання та підготовки даних для задач машинного навчання активно досліджується як у наукових працях, так і в індустріальних розробках. Існує низка підходів до оптимізації цих процесів, зокрема використання нереляційних баз даних (NoSQL), кешування, паралельна обробка, а також застосування спеціалізованих форматів зберігання даних, таких як TFRecord або HDF5. Проте універсального підходу до вибору баз даних та організації їх роботи для різних типів задач машинного навчання наразі не існує, що зумовлює необхідність подальших досліджень у цьому напрямі.

Об’єктом дослідження є процеси оптимізації застосування баз даних у контексті забезпечення ефективного зберігання та обробки великих обсягів даних для потреб машинного навчання.

Предметом дослідження є методи та засоби оптимізації структури баз даних, запитів до них, а також організації потоків даних з баз даних до

моделей машинного навчання з метою підвищення продуктивності та ефективності навчального процесу нейронних мереж.

Мета роботи полягає у дослідженні та оптимізації процесів зберігання і обробки даних у базах даних для підвищення ефективності навчання нейронних мереж.

Для досягнення поставленої мети в роботі, необхідно вирішити наступні задачі:

- проаналізувати існуючі підходи до зберігання та обробки даних у системах машинного навчання;

- дослідити особливості використання різних типів баз даних у контексті задач ML та DL;

- провести експериментальне порівняння обраних рішень щодо продуктивності, масштабованості та інтеграційних можливостей;

- сформулювати практичні рекомендації з оптимізації організації даних у базах даних для забезпечення ефективного навчання нейронних мереж.

Робота складається зі вступу, трьох розділів та висновків.

У першому розділі розглянуто теоретичні засади машинного навчання та обробки даних, класифікацію типів навчання, а також основні алгоритми.

У другому розділі досліджено особливості використання баз даних у системах машинного навчання, визначено їхні переваги та недоліки.

У третьому розділі проведено порівняльний аналіз ефективності різних методів зберігання та обробки даних у базах даних для задач глибокого навчання.

Висновки надають результати виконаної роботи, підкреслюють які дії здатні покращити ефективність систем машинного навчання за рахунок їх оптимізації.

РОЗДІЛ 1.
ТЕОРЕТИЧНІ ОСНОВИ МАШИННОГО НАВЧАННЯ ТА ОБРОБКИ ДАНИХ

1.1 Алгоритми навчання

Машинне навчання (Machine Learning, ML) є одним з ключових напрямків у сфері штучного інтелекту (ШІ), який забезпечує системам здатність до навчання з даних без явного програмування кожного кроку. Основна ідея машинного навчання полягає у створенні моделей, здатних автоматично виявляти залежності в даних та робити прогнози або приймати рішення на основі нової інформації [1, 4].

У рамках машинного навчання особливе місце займають нейронні мережі – алгоритмічні структури, що імітують принципи функціонування біологічного мозку. Вони побудовані із великої кількості взаємопов'язанних елементів – штучних нейронів, які об'єднуються у шари. Типова архітектура нейронної мережі включає [1, 2]:

- вхідний шар, який приймає дані;

- приховані шари, які здійснюють обробку за допомогою нелінійних перетворень

- вихідний шар, який формує остаточний результат (наприклад, ймовірності для кожного класу при класифікації).

Кожен штучний нейрон виконує наступні операції:

1. Зважує вхідні сигнали.

2. Обчислює зважену суму.

3. Застосовує функцію активації (наприклад ReLU, сигмоїда, tanh).

4. Передає результат далі по мережі.

Нейронна мережа навчається шляхом функції втрат (loss function) – математичної метрики, що показує, наскільки передбачення моделі відрізняються від реальних значень. Найбільш поширеними функціями втрат

є Mean Squared Error (MSE), Cross-Entropy. MSE – для регресії, Cross-Entropy для класифікації.

Процес навчання нейронної мережі є ітеративним процесом, у якому модель поступово адаптує свої внутрішні параметри (ваги) з метою мінімізації похибки між прогнозованими та фактичними значеннями. Цей процес складається з кількох ключових етапів [2, 3].

Перший етап – пряме поширення сигналу (Forward Pass). На цьому етапі вхідні дані передаються через мережу від вхідного шару до вихідного. Кожен нейрон виконує обчислення зваженої суми сигналів з попереднього шару, після чого застосовується функція активації, яка додає нелінійність у модель. Таким чином, формується вихідний сигнал мережі – тобто її поточне передбачення. Наприклад, при розпізнаванні зображень, мережа може приймати піксельні значення на вході, й повертати ймовірності приналежності зображення до певних класів на виході.

Другий етап – обчислення функції втрат (Loss Calculation). Після того, як мережа сформувала прогноз, оцінюється ступінь його відхилення від правильного (еталонного) результату, за допомогою функції втрат (loss function). Це математичне вираження помилки, яке служить основою для оновлення вагів. Наприклад, при класифікації часто використовується крос-ентропія, а при регресії – середньоквадратична похибка (MSE).

Третій етап – зворотнє поширення похибки (Backpropagation). Це ключовий алгоритм для оновлення вагів у нейронній мережі. Зворотнє поширення ґрунтується на правилі ланцюгової похідної, яке дозволяє визначити, як зміна кожного окремого вагового коефіцієнта вплинула на похибку. Цей процес відбувається у зворотньому порядку - від вихідного шару до вхідного. Для кожного нейрона обчислюється градієнт похибки, та ці градієнти використовуються для оновлення вагів, які з’єднують нейрони.

Четвертий етап – оптимізація параметрів (Parameter Optimization). Оновлення вагів виконується за допомогою оптимізаційоного алгоритму. Найбільш базовим є градієнтний спуск (Gradient Descent) – метод, що змінює

ваги у напрямку, протилежному градієнту функції втрат, з кроком, визначеним параметром навчальної швидкості (learning rate). Існують також більш ефективні модифікації цього методу:

- Adam (Adaptive Moment Descent) – поєднує ідеї моментуму та адаптивного масштабування градієнтів;

- RMSProp – зменшує коливання градієнтів шляхом нормалізації;

- SGD (Stochastic Gradient Descent) - оновлює ваги після кожного прикладу,що пришвидшує навчання при великих наборах даних;

П’ятий етап – повторення (Epochs). Усі наведені етапи виконуються для кожного прикладу у навчальному наборі, повна обробка всіх прикладів називається епохою (epoch). Зазвичай, нейронна мережа навчається протягом десятків або сотень епох, поки не буде досягнута прийнятна точність або зупинка за певним критерієм (наприклад, стабілізація похибки, валідаційна точність).

1.2 Класифікація типів машинного навчання

Машинне навчання є міждисциплінарною галуззю, що охоплює методи створення моделей, здатних виявляти приховані залежності та закономірності у даних. Вибір конкретного підходу до навчання залежить від характеристики вхідної інформації, наявності розмічених вихідних значень, а також від цілей які стоять перед дослідником. Згідно з цими критеріями, виділяють три основні парадигми машинного навчання: навчання з учителем, навчання без учителя, навчання з підкріпленням [3 - 5].

1. Навчання з учителем (Supervised Learning).

Цей тип навчання передбачає використання розмічених даних, де кожному вхідному прикладу відповідає наперед відомий результат (мітка). Основними завданнями алгоритму є встановлення залежності між вхідними параметрами та відповідними мітками, для здійснення точних передбачень на нових, ще не бачених прикладах. Основні задачу цього підходу –

класифікація та регресія. Процес класифікації, це процес віднесення об’єктів до визначених категорій або класів. Регресія – це процес прогнозування числових значень, таких як ціна автомобілів, температура. Як приклад можна узяти створення моделі, яка за допомогою відповідної мітки для кожного зображення, буде визначати наявність щура або кажана на зображенні.

2. Навчання без учителя (Unsupervised Learning).

В даному варіанті навчання відсутня інформація про цільові значення, дані є не розміченими. Алгоритми працюють лише з вхідними ознаками. Вони намагаються виявити внутрішню схожість, структуру даних, або дивні патерни. Застосовується цей підхід широко для обробки даних, аналізу структури вибірки та зменшенню її розмірності.

Задачі, які фігурують у цьому виді навчання: кластеризація та зниження розмірності:

- кластеризація – це групування об’єктів, на основі їхньої схожості;

- зниження розмірності – це трансформація вхідних даних з однаковими ознаками у простір, де зберігається ключова інформація. Наприклад, виявлення без міток окрему групу клієнтів банку, судячи з транзакцій.

3. Навчання з підкріпленням (Reinforcement Learning).

У цьому підході використовується взаємодія інтелектуальних агентів із середовищем, де агент отримує певний результат (винагорода або штраф), після здійснення певних дій. Головною задачею цього підходу є вдосконалення дій, за допомогою максимізації сумарної винагороди, шляхом самонавчання через проби та помилки.

Цей підхід широко використовується у сферах для оптимізації процесів індустрії (управління логістикою, енергоспоживанням), керування роботехнічних систем (навчання ходьбі). Приклад: навчання автопілоту у автомобіля, щоб той міг у динамічних умовах приймати рішення для гальмування, поворотах та ін.

1.3 Глибоке навчання

Глибоке навчання (Deep Learning) являє собою один з найбільш активно досліджуваних, провідних напрямків у межах машинного навчання. Цей напрям орієнтований на побудову та використання глибоких нейронних мереж, які мають у собі багаторівневу структуру. Головною особливістю цих моделей є наявність великої кількості шарів, які приховані. Це дає змогу здійснювати абстрагування ознак та поетапну трансформацію – від простих, до складних, або узагальнених представлень [4 - 6].

Відмінно, від звичайних алгоритмів машинного навчання, що часто потребують попереднє ручне конструювання, глибокі нейронні мережі мають можливість самостійно вилучати інформативні характеристики з неструктурованих, або напівструктурованих даних. Саме ця можливість вплинула на широке впровадження методів глибокого навчання в такі високотехнологічні напрямки [5]:

- системи комп’ютерного зору (Computer Vision) – згорткові нейронні мережі (CNN), котрі демонструють високу ефективність у задачах розпізнаванням об’єктів, медичної діагностики за допомогою візуальних даних;

- обробка природньої мови (Natural Language Processing) – рекурентні архітектури (RNN), нейронні мережі з довгою та короткочасною пам’яттю (Long Short-Term Memory), а також трансформерні моделі, що мають головну роль у реалізації завдань синтаксичного аналізу, генерації тексту, розпізнаванням мовлення;

- генеративне моделювання: сучасні генеративні архітектури (GPT, DALL-E, StyleGAN) дозволяють створювати аудіо матеріали, текстові матеріали, візуальні, що відіграє величезну роль у розвитку медіа індустріях.

Однак, все одно на переваги, успішна реалізація моделей глибокого навчання потребує дуже якісну підготовку даних у великих обсягах. Ризик збільшує висока складність нейронних мереж до пристосування для

навчального набору, тим паче в умовах обмеженості даних, або їх низької репрезентативності. Через це, забезпечення якості даних стає критичним етапом у побудові ефективних моделей глибокого навчання.

1.4 Роль якості даних у навчанні НМ

Під час створення та тренування нейронних мереж, дані виконують головну роль, через те, що саме вони визначають поведінкові характеристики моделі. Кількість, структура, частота даних має значення для побудови не лише стійких, а й продуктивних моделей [6].

Через недостатню увагу до якості навчального набору можуть виникнути наступні проблеми:

- переобучення (overfitting) – випадок, колі модель не здатна узагальнювати, але запам’ятовує шум, випадкові відхилення у навчальній вибірці;

- недонавчання (underfitting) – з’являється тоді, коли через дуже спрощену архітектуру або брак даних не здатна охопити базові патерни.

Щоб все ж таки уникнути таких випадків, треба висунути вимоги:

- форматованість і структурованість (подача даних з виді числових матриць, векторів для ефективної обробки даних);

- очищення даних (усунення дублікатів, шумів за рахунок їх виявлення);

- нормалізація та стандартизація (для стабільної та швидкої роботи алгоритмів оптимізації необхідно переводити числові ознаки до одного масштабу);

- балансування вибірки (це актуально для задач з класифікації, де нерівномірне представлення класів може зумовити упередженість моделі);

- аугментація даних (метод штучного розширення вибірки за рахунок генерації модифікованих варіантів наявних прикладів через зміну масштабу або яскравості зображень).

Коли йде навчання моделей з великою кількістю параметрів, значення мають не тільки якість, обсяг даних, а й зберігання, доступ та обробка даних. Через це у масштабних системах штучного інтелекту вкрай важливо впроваджувати спеціалізовану інфраструктуру з використанням Data Lakes, та Data Warehouses, та інших механізмів потокової обробки, які забезпечують стабільність та масштабованість процесів машинного навчання на всіх його етапах.

1.5 Типи баз даних

У сучасних інформаційних системах є високий перелік типів баз даних, кожен з яких направлений на розв'язання певних категорій задач, які обумовлені структурою, характером, обсягами оброблювання даних.

У контексті зберігання та обробленні великих обсягів даних, які використовуються для навчання штучних нейронних мереж, дуже велике значення має вибір типу бази даних, так як через нього залежить масштабованість, ефективність доступу, швидкодія та гнучкість інформаційної системи. У цій сфері є основні категорії баз даних: реляційні, нереляційні (NoSQL), об'єктивно-орієнтовані, графові бази даних [3 - 7].

1. Реляційні бази даних (Relational Databases).

Ця модель баз даних є однією з найстабільніших, найдавніших методів зберігання інформації, яка й на даних момент доволі широко застосовується в інформаційних системах різного призначення. В основі цієї моделі лежить представлення даних у формі таблиць, структура яких чітко визначається схемою – із зазначенням полів, первинних і зовнішніх ключів, типів даних, і також обмеженням цілісності. SQL (Structured Query Language) – це мова взаємодії з системами такого типу, яка є стандартом для створення, вибірки, оновлення даних. MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server – це одні з найпоширеніших реалізацій реляційних СКБД. Переваги реляційної моделі:

- тільки формалізована структура даних;

- висока ефективність при виконанні складних аналітичних запитів;

- підтримка транзакцій з гарантією цілісності та узгодженості.

Не зважаючи на це, використання реляційних баз показує певні обмеження, наприклад в контексті обробки неструктурованої або частково структурованої інформації, та у ситуаціях, які потребують горизонтального масштабування – типових для сучасних ML-архітектур.

2. NoSQL-бази даних.

Альтернатива базовим реляційним системам є NoSQL-бази даних, які були розроблені з урахуванням нових викликів, пов’язаних із зростанням обсягів даних, їх різнорідною природою та необхідністю масштабованої обробки в розподілених середовищах. Назва NoSQL (Not Only SQL) вказує на гнучкість моделі. Вона не передбачає використання SQL, не обмежується лише табличним представленням.

Основні типи NoSQL-середовищ включають:

- документно-орієнтовані БД (MongoDB, CouchDB): інформація зберігається у вигляді гнучких структурованих документів (JSON/BSON), які можуть мати різну структуру;

- ключ-значення сховища (Redis, Riak): високопродуктивні системи для асоціативного зберігання даних, зручні для кешування;

- колонкові бази (Apache Cassandra, HBase): оптимізовані для обробки великих аналітичних навантажень;

- графові бази.

NoSQL-бази даних активно використовуються в машинному навчанні для зберігання великих обсягів тренувальних даних, логів, потоків подій, мультимедійного контенту, що не піддаються ефективній обробці в рамках базової табличної моделі. Завдяки своїй масштабованості, вони легко інтегруються з хмарними ML-сервісами.

3. Об’єктно-орієнтовані бази даних.

Ці системи управління базами даних (ООБД) поєднують у собі концепцію програмування у парадигмі об’єктної орієнтації з механізмами зберігання інформації. Саме в таких системах збереженню підлягають об’єкти, які інкапсулюють як дані, так і пов’язані з ними методи.

Об’єкти належать до певних класів, підтримують механізми наслідування, інкапсуляції та поліморфізму – це дозволяє зберігати складні ієрархічні структури. Застосування об’єктно-орієнтованих баз даних правильне в тих сферах, де є тісний зв’язок між логікою програми та структурою даних, зокрема в інженерних CAD/CAE-системах, моделюванні фізичних явищ та автоматизованому проектуванні. Не зважаючи на свою виразну модель даних ООБД не здобули широкого застосування в сфері машинного навчання, через їх масштабованість та обмежені підтримку паралельної обробки.

4. Графові бази даних

Графові бази реалізують представлення інформації у вигляді графових структур, де об’єкти моделюються у виді вершин, а взаємозв’язки між ними – у вигляді ребер. Ця модель гарно підходить до тих випадків, коли аналіз зв’язків відіграє ключове значення.

Найвідоміше рішення для графових баз даних – Neo4j, яка дозволяє ефективно моделювати та обробляти соціальні мережі, онтології, системи рекомендацій та мережі знань. Cypher - мова, яка застосовується для здійснення запитів, та оптимізована для роботи з графовими структурами.

В свою чергу, ML-графові бази даних застосовуються для:

- побудови графів знань, що служать джерелами для моделей типу Graph Neural Networks(GNN);

- моделювання складних предметних галузей – починаючи з біоінформатики та закінчуючи медичними системами підтримки прийняття рішень;

- зберігання структур взаємодії користувачів, продуктів, документів;

1.6 Вимоги до БД у контексті ML/DL

У сучасних інтелектуальних системах, що базуються на підходах машинного (ML) і глибокого (DL) навчання, питання ефективної організації зберігання доступу до даних є головним для продуктивності обчислювальних процесів.

Обсяги навчальної інформації постійно зростають та охоплюють широкий спектр форматів – неструктуровані текстові та зображувальні дані, структуровані таблиці, часові ряди. Відтак, системи зберігання які функціонують у таких архітектурах, мають задовольняти низьку дуже важливих вимог, насамперед – масштабованість, паралельний доступ до інформації та високу швидкість обробки запитів.

1. Масштабованість (Scalability). Під цим терміном розуміється здатність системи керування даними підтримувати ефективне функціонування зі зростанням обсягів даних та кількості запитів. Для ML/DL-платформ це означає необхідність обробки фантастично великих розмірів вхідної інформації (терабайти або петабайти інформації), а ця інформація знаходиться у різних джерелах – соціальних мережах, цифрових медіа, системах моніторингу, телеметричних сенсорах, тощо.

Особливої важливості набуває горизонтальне масштабування – можливість збільшення обчислювальних або зберігальних ресурсів, без змін архітектури застосунку. Даному масштабуванню сприяють механізми кластеризації, реплікації та шардінг. Це все дозволяє рівномірно розподіляти навантаження між вузлами.

Amazon S3 з Athena, Azure Synapse, Google BigQuery – хмарні обчислювальні сервіси, які демонструють приклади ефективного масштабування в автоматичному режимі, що особливо актуально при тренуванні ресурсномістких моделей глибокого навчання.

2. Швидкість доступу до інформації (Low Latency / High Throughput).

Під час побудови моделей штучного інтелекту одним із надважливих є забезпечення мінімальних затримок у читанні та передачі даних, а також підтримка високої пропускної здатності каналів доступу. Це обумовлюється необхідністю постійного завантаження батчів (пакетів) даних у навчальні системи, які виконують численні ітерації у рамках одного тренувального циклу.

Для досягнення високої продуктивності використовується:

- високошвидкісні файлові системи (наприклад, Lustre, Alluxio);

- бази даних з оптимізацією під SSD-накопичувачі;

- системи кешування (Redis, Memcached);

- сховища оперативної пам’яті, що забезпечують миттєвий доступ до даних без звернення до диску.

Найвища ефективність досягається у разі інтеграції СКБД безпосередньо з ML-фреймворками. Як приклад формати типу TFRecord (у TensorFlow) або DataLoader (у PyTorch) дозволяють подавати дані про моделі без надмірних витрат на конвертацію, що істотно скорочує час навчання.

3. Паралельна та розподілена обробка (Parallel and Distributed Processing).

Постійно зростаючі обсяги даних і збільшення складності моделей будують потребу у паралельному опрацюванні інформації. Розпаралелювання дає змогу розділяти навантаження між кількома процесами або обчислювальними вузлами, що дуже пришвидшує переробку даних та сам процес навчання.

Існують три основні підходи для реалізації паралелізму:

- Data parallelism (одночасне оброблення різних частин датасету копіями моделі на кількох GPU або вузлах);

- Model parallelism (розподілення великої нейронної мережі на фрагменти, які обробляються незалежно);

- Pipeline parallelism (послідовна передача даних між етапами обробки з використанням окремих обчислювальних елементів).

Підтримка потокової обробки (stream processing) з використанням технологій Apache Kafka, Apache Flink дають можливість у реальному часі здійснювати навчання, що є дуже важлива для онлайн-адаптивних систем. Щоб оптимізувати доступ до великих масивів інформації активно застосовують спеціалізовані формати з підтримкою вибіркового зчитування та паралелізації. Apache Parquet, Avro, HDF5 – ці формати дозволяють зменшити навантаження на дискову підсистему та ефективно використовувати ресурси CPU/GPU.

Інші вимоги до баз даних для задач машинного та глибокого навчання наведено в табл. 1.1.

Таблиця 1.1

Вимоги до баз даних для задач машинного та глибоко навчання

Критерій	Опис	Технології/Приклади
Інтеграція з ML/DL платформами	Можливість прямого зчитування даних з БД у навчальні фреймворки з мінімальними накладними втратами	TFRecord(TensorFlow), DataLoader(PyTorch), Petastorm
Формати оптимального зберігання	Підтримка форматів, які дозволяють ефективно зчитувати великі масиви даних частинами або потоками	Apache Parquet, Avro, HDF5
Гнучкість у роботі з типами даних	Підтримка різних типів структурованих/неструктурованих даних(текст, відео, зображення)	NoSQL БД(MongoDB, Cassandra) Хмарні об’єктні сховища (S3,GCS)
Доступність та надійність	Відмовостійкість, безперервний доступ до даних у розподіленому середовищі	Реплікація, резервне копіювання, geo-розподілені кластери

РОЗДІЛ 2.
ОСОБЛИВОСТІ ВИКОРИСТАННЯ БД У СИСТЕМАХ МАШИННОГО НАВЧАННЯ

2.1 Сучасні методи організації зберігання даних у машинному навчанні

У процесі розробки моделей машинного навчання, важливо не лише мати якісні дані, але й надважливо забезпечити їх ефективне зберігання і доступ. Через зростання обсягів даних, підвищення вимог до продуктивності, з'являється потреба у використанні надійних та сучасних технологій для зберігання, версіонування, передачі та доступу даних до навчальних вибірок. У цьому розділі ми розглянемо інструменти та технології, які забезпечують надійну інфраструктуру даних у ML-проєктах [8, 9].

1. Реляційні бази даних. Застосування PostgreSQL.

Реляційні системи керування базами даних (РСКБД) були й залишаються популярним вибором для роботи із структурованими даними. PostgreSQL, як одна з провідних РСКБД підтримує складні запити, індексацію, розширення через сторонні модулі, транзакції. Використання PostgreSQL у машинному навчанні:

- зберігання оброблених або агрегованих табличних даних;

- створення підготовчих вибірок за допомогою SQL-запитів із JSON, GROUP BY, WINDOW FUNCTIONS;

- інтеграції з бібліотеками Python через psycopg2, SQLAlchemy або безпосередньо з pandas (read_sql_query()).

Окрім цього всього, PostgreSQL підтримує формат JSONB для зберігання напівструктурованих даних, що корисно при роботі з логами, метаданими або параметрами моделей.

2. Документоорієнтовані сховища. MongoDB у ML-застосуваннях.

Щоб зберігати неструктуровані або гнучко структуровані дані буде доцільним використовувати документноорієнтовані СКБД, зокрема

MongoDB. Ця база даних оперує даними у форматі BSON (тобто бінарний JSON), що дозволяє ефективно маніпулювати вкладеними структурами, об’єктами та списками. Для чого використовується MongoDB у ML-середовищах:

- для зберігання сировинних даних із динамічною структурою (наприклад відповіді API, текстові документи, журнали подій);
- організації Data Lake-структури з використанням MongoDB Atlas;
- підготовки до стрімінгової обробки даних перед передачею у фреймворки типу TensorFlow, PyTorch.

Щоб уникнути складних попередніх трансформацій, гнучка система прискорює прототипування моделей та інтеграцію із реальними джерелами даних.

3. Спеціалізовані формати. HDF5 для зберігання об’ємних даних.

Формат HDF5 (Hierarchical Data Format version 5) являється одним із найбільш ефективних засобів для зберігання наукових та інженерних даних у вигляді багатовимірних масивів. Його структура дає можливість організувати дані у вигляді ієрархічних груп, аналогічно до файлової системи, що зручно для зберігання великих датасетів, пов’язаних із комп’ютерним зором, обробкою звуку або часових рядків.

HDF5 у ML-практиці використовується для зберігання батчів зображень, аудіо або сенсорних даних. Він сумісний з бібліотеками h5py, PyTables, TesnorFlow, Keras. А ще HDF5 дозволяє реалізувати часткове зчитування, що важливо при тренуванні моделей на обмежених ресурсах пам’яті.

Якщо у проєкті обсяг вхідних даних перевищує обсяг оперативної пам’яті, але необхідно підтримувати високу швидкість доступу - використовують HDF5.

4. Контроль версій даних. DVC як система управління ML-дослідженнями.

Дивлячись на повторюваний і експериментальний характер роботи з ML-моделями, особливого значення набуває питання версіонування даних. Для цього активно використовують DVC (Data Version Control), який за функціональністю нагадує Git. Але DVC спеціалізується на об’ємних даних та артефактах моделювання. Можливості DVC:

- підтримка віддалених сховищ (локаційних, S3, GCS), що дозволяє будувати відтворювані пайплайни машинного навчання;

- зберігання, версіонування та відстеження змін у датасетах;

- реплікація структури експерементів (включаючи конфігураційні файли, скрипти, ваги моделей);

Інтеграція DVC з CI/CD-інфраструктурою сприяє автоматизації процесів валідації моделей, що значно підвищує якість дослідження та прозорість.

5. Підготовлені набори даних. TensorFlow Datasets(TFDS).

Щоб пришвидшити процес побудови та тестування моделей глибокого навчання, компанія GOOGLE розробила TensorFLow Datasets - бібліотеку, де є готові датасети, які оптимізовані для роботи у фреймворку TensorFLow. TFDS надає API для автоматичного завантаження, нормалізації, кешування, та батчування популярних наборів (наприклад MNIST, CIFAR-10, ImageNet).

Переваги, які має TFDS:

- уніфікований формат (tf.data.Dataset) для сумісності з пайплайнами TensorFlow;

- автоматичне кешування та препроцесинг (ресайз, нормалізація);

- підтримка розподіленого завантаження та обробки, що знижує навантаження на основний CPU.

2.2 Використання Data Lake та Data Warehouse у завданнях машинного навчання

Сучасні інформаційні системи, які оперують великими обсягами даних, виникає потреба у високоефективних рішеннях для їхнього зберігання та обробки, які здатні забезпечити масштабованість, адаптивність і оперативний доступ до інформації. У сфері машинного навчання ці вимоги є надважливими, через те, що моделі працюють з великими наборами даних, що надходять із різних джерел та характеризуються неоднорідною структурою. Для задовільнення таких потреб використовують часто два підходи до організації сховищ даних – Data Lake та Data Warehouse. Кожен з цих підходів має свої переваги, обмеження, та сфери застосування, особливо в контексті побудови систем машинного навчання [7, 8].

Data Lake – підхід до зберігання даних, націлений на накопичення необробленої або напівструктурованої інформації у її первинному вигляді. Єдиний централізований репозиторій – основний принцип, який дозволяє зберігати будь-які види даних – структуровані, непівструктуровані та неструктуровані – без попереднього проектування схеми даних [8].

Характеристики Data Lake:

- можливість зберігання сирих даних (raw data) із подальшим очищенням, трансформацією та аналітикою;

- підтримку широкого спектру форматів файлів (наприклад CSV, JSON, XML, Parquet, Avro);

- здатність до масштабування завдяки використанню розподілених файлових систем (наприклад HDF, Amazon S3);

- інтеграцію з іструментами для обробки великих даних і машинного навчання, такими як Apache Spark, TensorFLow, PyTorch.

Data Lake у машинному навчанні забезпечує централізоване сховище різноманітних типів даних, включаючи історичні дані, журнали подій, телеметричні показники, зображення, дані сенсорів що створює фундамент

для побудови багатошарових моделей. Як приклад: при прогнозуванні енергоспоживання можуть одночасно використовуватися метеорологічні дані, історія споживання, інформація з IoT-пристроїв – це все легко агретується у Data Lake без обмежень структури. Основні переваги Data Lake у машинному навчанні:

- зниження витрат на підготовку даних завдяки відсутності проблеми необхідності попередньої побудови схеми;

- можливість багаторазового використання одних і тих самих даних для побудови різних моделей без дублювання;

- готовність до обробки даних у режимі реального часу завдяки інтеграції з потоковими платформами (Kafka, Flint);

Концепція Data Warehouse. Data Warehouse (сховище даних) представляє собою високоструктуровану платформу, яка призначена для зберігання очищених, інтегрованих та агрегованих даних, які підготовлені для швидкої аналітики та звітності. Саме цей підхід більш орієнтований на бізнес-аналітику, але ефективно застосовується у завданнях машинного навчання, особливо на етапах попереднього аналізу та формування ознак. Особливості, які має Data Warehouse – використання системи даних, визначеної на етапі запису (schema-on-write), що гарантує цілісність і консистентність даних; застосування процесів ETL/ELT (Extract, Transform, Load) для забезпечення високої якості підготовлених даних; оптимізація для виконання аналітичних запитів, включаючи OLAP-операції (Online Analytical Processing); висока продуктивність при виконанні SQL-запитів до великих обсягів табличних даних.

У задачах машинного навчання Data Warehouse використовується для:

- ведення журналів експериментів, збереження результатів валідації моделей;

- зберігання фінальних наборів ознак, отриманих внаслідок попереднього аналізу даних;

- проведення аналізу трендів і реалізації процедур періодичної переоцінки моделей (model retraining) на основі оновлених агрегованих даних.

Як приклад сучасних реалізацій Data Warehouse можна назвати Snowflake, Google BigQuery, Amazon Redshift, Azure Synapse. Всі вони підтримують реалізацію з ML-інструментами, включаючи запуск моделей безпосередньо на SQL-рівні або через інтеграцію з середовищами як Jupyter Notebooks.

У таблиці 2.1 наведено порівняння головних характеристик Data Lake та Data Warehouse із урахуванням специфіки задач машинного навчання.

Таблиця 2.1.

Порівняння Data Lake та Data Warehouse у ML

Критерій	Data Lake	Data Warehouse
Тип даних	Будь-які (сирі, структуровані)	Структуровані, очищені
Підхід до схеми	Schema-on-write	Schema-on-write
Гнучкість	Висока	Обмежена схемою
Придатність до ML	Оптимально для сирих даних	Оптимально для підготовлених даних
Швидкість аналітики	Залежить від формату	Висока
Інтеграція з ML-фреймворками	Пряма (Spark, TensorFlow)	Переважно через SQL або API

Сучасні ML-рішення також доволі часто об’єднують за рахунок використання гібридних архітектур – Data Lakehouse.

Data Lakehouse виникла як вихід від потреби з’єднати гнучкість Data Lake і структурованість Data Warehouse. Саме в такій архітектурі дані зберігаються у Data Lake, а поверх них створюється структура, що забезпечує властивості, які притаманні сховищам даних [8, 9].

Apache Iceberg, Delta Lake, Apache Hudi – приклади реалізації Data Lakehouse.

Ці рішення дають можливість зберігати дані у відкритих форматах (Parquet, ORC) із підтримкою ACID-транзакцій, що критично важливо для відтворюваності ML-експериментів. Додатково можливість використання механізму rime-traveling (збереження історії змін) дає змогу повертатися до попередніх версій даних без дублювання інформації.

Типи даних, специфіки завдань – це ті критерії, через які стоїть вибір між Data Lake та Data Warehouse. Data Lake – висока гнучкість, масштабованість при роботі з великими обсягами неоднорідних даних. Data Warehouse – структурованість, стабільність та ефективність аналітичних запитів. А інтеграція обох відходів у Data Lakehouse надає універсальну архітектуру, яка здатна охопити повний цикл у роботі з ML-проєктами.

2.3 Метрики для систем машинного навчання

Для сучасних системи машинного навчання які оперують величезними обсягами даних, ефективність баз даних є критично важливим чинником, який впливає на швидкість підготовки навчальних вибірок, виконання аналітичних запитів та оновлення моделей. Щоб забезпечити високу продуктивність, необхідно враховувати низьку ключових метрик – затримку (lanetcy), пропускну здатність (throughput), використання оперативної пам’яті (memory usage), та використання дискового простору (disk usage) [10].

Затримка визначає час, що проходить від моменту надсилання запиту до отримання відповіді від баз даних. У ML, особливо під час обробки даних або під час inference у реальному часі, низька затримка є дуже важливою для забезпечення своєчасної реакції системи.

Класифікації затримки:

- затримка читання (read latency): час, який необхідний для отримання даних з бази;

- затримка запису (write latency): час, який потрібний для збереження або оновлення даних;

- затримка коміту (commit latency): час, який витрачається на фіксацію транзакції.

Щоб зменшити затримку, використовуються методи оптимізації індексів, кешування даних (наприклад використовуючи Redis або Memcached), а також денормалізація структури даних. Моніторинг затримки здійснюється за допомогою інструментів pg_stat_statement у PostgreSQL або EXPLAIN ANALYZE.

Пропускна здатність (Throughput) відображає кількість операцій, які база даних може обробити за одиницю часу, і вимірюється вона в операціях за секунду (ops/sec). У ML-системах висока пропускна здатність забезпечує ефективну обробку великих обсягів даних, що особливо важливо при генерації навчальних вибірок, або при обробці потокових даних з платформ типу Kafka.

Оптимізація пропускної здатності досягається шляхом:

- партиціювання даних: розподіл на частини для паралельної обробки;

- реплікації: створення копій для забезпечення доступності та розподілу навантаження;

- асинхронної обробки: виконання операцій без блокування основного потоку;

- використання in-memory рішень: зберігання критичних даних у оперативній пам’яті для швидкого доступу.

Правильне та ефективне використання оперативної пам’яті (Memory usage) сприяє забезпеченню продуктивності бази даних. Недостатня її кількість може призвести до swap-файлів, що значно знижує швидкодію системи. У ML-сценаріях важливо, щоб: дані, індекси які часто використовуються зберігалися в пам’яті для швидкого доступу; результати запитів кешувалися для зменшення навантаження на систему; пам’ять використовувалася ефективно при багатопоточній обробці великих обсягів даних.

Моніторинг використання пам’яті здійснюється за допомогою інструментів, таких як htop, vmstat, top, а також вбудованих засобів моніторингу СУБД(наприклад shared_buffers у PostgreSQL або статистика використання пам’яті в MongoDB).

Обсяг дискового простору (Disk usage), зайнятого даними, індексами та службовими файлами є дуже важливим показником продуктивності баз даних. У великих ML-проєктах, де обсяги даних можуть доходити то терабайтів, ефективне використання дискового простору дозволяє зменшити витрати на зберігання та покращення швидкодії системи.

Фактори, які впливають на використання дискового простору:

- формат зберігання даних: колонкові формати (наприклад Parquet, ORC) забезпечують ліпше стиснення та швидкий доступ до даних;

- стиснення даних: використання алгоритмів стиснення (LZ4, ZSTD) зменшує обсяг зайнятого простору;

- кількість індексів: надмірна кількість індексів може збільшити використання дискового простору;

- фрагментація таблиць: регулярна дефрагментація допомагає зменшити зайнятий простір.

Комплексна оцінка продуктивності баз даних у ML-системах являє собою аналіз усіх вищезазначених метрик у сукупності. Наприклад – зменшення затримки може вимагати додаткового використання оперативної пам’яті або збільшення використання дискового простору через кешування. Оптимізація пропускної здатності може потребувати реплікації, що також впливає на дискові ресурси. Щоб ML-системи мали все ж таки високу продуктивність, необхідно порівнювати альтернативні архутектури баз даних (SQL та NoSQL, OLTP та OLAP, row-store та column-store), необхідно приймати обґрунтовані рішення щодо оптимізації ресурсів, та масштабувати інфраструктуру ML-систем відповідно до зростаючих потреб проєкту.

Через це, проєктування високопродуктивних ML-орієнтованих баз даних вимагає правильного та збалансованого підходу, з урахуванням усіх ключових метрик продуктивності.

2.4 Порівняльний аналіз популярних систем зберігання та обробки даних у ML-задачах

Розробка інфраструктури машинного навчання потребує детального вибору системи управління базами даних та платформи зберігання даних. Дивлячись на обсяги даних, різнорідність джерел, потребі у масштабуванні, швидкодії та інтеграції з інструментами ML, це завдання стає критично важливим. Порівняння найбільш популярних рішень дозволяє підібрати оптимальну систему з урахуванням специфіки та напрямку застосування.

На сучасному етапі найбільш використовуваними в наукових і промислових ML-проєктах є: PostgreSQL, MongoDB, Apache Cassandra, ClickHouse, Amazon S3 (разом із форматами Parquet/ORC), Google BigQuery, Apache Hadoop (HDFS), а також Redis як проміжне кеш-сховище. Розглянемо їхні ключові особливості, переваги та недоліки [10, 11].

1. PostgreSQL – гарна реляційна СКБД з відкритим вихідним кодом, що вирізняється суворим дотриманням стандартів SQL і високою гнучкістю завдяки можливості підключення модулів (pgvector, TimescaleDB). PostgreSQL добре інтегрується з екосистемою Python, включно з бібліотеками SQLAlchemy та psycopg2, що дуже важливо для ML-проєктів.

Переваги PostgreSQL:

- підтримка структурованих даних і суворої типізації;
- можливість створення складної логіки через stored procedures, тригери, CTE;
- доступність векторних індексів (pgvector) для задач подібності.

Недоліки PostgreSQL:

- менш ефективна для обробки великих обсягів неструктурованих даних;

- обмежені можливості горизонтального масштабування (потребує додаткових рішень, наприклад Citus).

Використовується PostgreSQL у невеликих та середніх ML-системах зі структурованими даними та побудові навчальних вибірок.

2. MongoDB – це NoSQL система, зо зберігає дані у форматі BSON-документів. Цей підхід сприяє високій гнучкості під час роботи з даними із змінною або слабо структурованою схемою, що доволі часто трапляється при обробці логів, сенсорних даних чи метаданих.

Переваги MongoDB:

- висока швидкість операцій читання та запису;

- гнучкість схеми зберігання;

- можливість масштабувати через шардинг.

Недоліки MongoDB:

- обмежені можливості для складних агрегацій у порівнянні з SQL;

- підвищене споживання пам’яті на великих обсягах даних.

Застосовується MongoDB для зберігання неструктурованих даних, метаданих, проміжних результатів ML-конвеєрів.

3. Apache Cassandra – розподілена колоночна NoSQL-СУБД, яка створена для обробки великих потоків даних у реальному часі. Відсутність єдиного вузла керування дає змогу легко масштабувати систему горизонтально без зупинок.

Переваги Apache Cassandra:

- масштабованість систем без погіршення продуктивності при збільшенні кількості вузлів;

- здатність обробляти записи з високою частотою оновлень і вставок;

- гнучка система реплікацій з можливістю налаштування фактору реплікації для окремих дата-центрів;

- висока відмовостійкість завдяки анатомічному переміщенню даних при збої вузлів.

Недоліки Apache Cassandra:

- відсутність підтримки складних транзакцій(ACID-підхід обмежений);

- низька ефективність для складних ad hoc-запитів і аналітики;

- необхідність глибоких знань для налаштування та оптимізації кластерної архітектури.

Apache Cassandra застосовується у сфері обробки телеметричних даних від IoT-пристроїв, логування подій, зберігання даних для онлайн ML-сервісів з високим навантаженням, систем рекомендацій у реальному часі, а також системи відстеження користувацької активності на масштабних веб-ресурсах.

4. ClickHouse – високопродуктивна колонкова система управління базами даних, оптимізована для обробки великих масивів даних у режими OLAD (Online Analytical Processing). Сама головна відмінність ClickHouse – це зберігання даних у колонковому форматі, що надає перевагу при виконанні запитів, орієнтованих на вибірки з великої кількості записів, але обмеженої кількості колонок. Архітектура ClickHouse представляє гнучке партиціювання та індексування даних, через що з’являться можливість швидко знаходити потрібні підмножини даних навіть у масштабних таблицях.

Переваги ClickHouse:

- висока швидкість виконання запитів завдяки колонковій структурі зберігання;

- ефективні алгоритми стиснення та партиціювання даних;

- підтримка горизонтального масштабування через шардинг та реплікацію;

- розвинений механізм матеріалізованих представлень для пришвидшення аналітики.

Недоліки ClickHouse:

- система в основному орієнтована на читання, що обмежує використання для транзакційних навантажень(OLTP);

- складність налаштування системи для досягнення оптимальної продуктивності на розподілених конфігураціях;

- відсутність підтримки складних транзакцій і обмежена робота з оновленнями.

Використовують ClickHouse для аналізу історичних даних у великих дата-логах, зберігання та обробки ознак (feature store) для підготовки даних у ML, обробка журналів подій, побудова систем бізнес-аналітики та інтерактивних звітів із великих датасетів.

5. Amazon Simple Storage Servise(S3) являється фундаментом для побудови Data Lake – централізованого сховища, яке дозволяє зберігати структуровані та неструктуровані дані будь-якого типу в їхньому оригінальному форматі. Використанні колонкових форматів файлів, таких як Parquet або ORC, дають можливість зменшити обсяг даних та пришвидшити аналітичні запити за рахунок ефективного стиснення та можливість вибіркового читання лише потрібних колонок. Інтеграція Amazon S3 з інструментами аналізу даних (Apache Spark, Dask, TensorFlow, Snowflake) робить з Amazon S3 невід'ємну складову сучасних пайплайнів машинного навчання [10, 11].

Переваги Amazon S3:

- майже необмежена масштабованість і надійність завдяки архітектурі хмарного сховища;

- економічна ефективність за рахунок моделі pay-as-you-go та багаторівневого зберігання (S3 Standart, S3 Glacier);

- можливість прямої інтеграції з численними сервісами AWS та сторонніми аналітичними платформами;

- підтримка стандартів безпеки та шифрування даних на рівні сховища.

Недоліки Amazon S3:

- вагома затримка доступу при роботі з даними у сирому вигляді;

- обмежена підтримка транзакцій і оновлень даних без використання додаткових сервісів (наприклад QWS Lake Formation);

- потреба у зовнішніх інструментах для управління метаданими та каталогами даних;

Створення централізованих дата-озер для зберігання первинних та оброблених даних, архівація великих датасетів, створення історичних репоризаріїв даних для подальшого аналізу, підготовка даних для навчання моделей глибокого навчання – це ті сфери, де використовується Amazon S3.

6. Google BigQuery - високопродуктивна хмарна аналітична платформа, яка має колонкову систему зберігання та паралельні обчислення. Через це Google BigQuery забезпечує можливість швидкої обробки петабайтних обсягів даних. Будучи серверлес-сервісом, Google BigQuery позбуває користувача завдання управління інфраструктурою, автоматично масштабуючи ресурси, залежно від їх навантаження. Google BigQuery для взаємодії з даними має підтримку SQL, це значно полегшує адаптацію користувачів, а також інтеграцію з екосистемою Google Cloud Platform включно з Vertex AI, Dataflow та іншими сервісами машинного навчання.

Переваги Google BigQuery:

- автоматичне масштабування обчислювальних потужностей під навантаження;

- висока швидкодія під час виконання складних аналітичних запитів на великі дані;

- інтеграція з ML-платформами для розгортання моделей на рівні бази даних;

- спрощене управління даними за рахунок Data Catalog та іншими інструментами метаданих.

Недоліки Google BigQuery:

- модель оплати, що залежить від обсягу оброблених даних – це може призвести до непередбачуваних витрат;

-обмежена підтримка постійних оновлень даних (імплементація UPSET операцій потребує додаткових процедур);

- потреба у глибоких знаннях принципів оптимізації SQL-запитів для зниження витрат.

Використовують Google BigQuery для побудови централізованих сховищ для аналітики, реалізації складних звітів і дашбордів у реальному часі, обробки даних та їхнього аналізу для навчання моделей машинного навчання на великих вибірках, інтеграції з ML-алгоритмами у запити (BigQuery ML).

7. Redis - високошвидісне кешування та обробка даних у пам’яті. Redis являє собою інтелектуальне сховище даних у пам’яті, яке поєднує функції бази даних, кеша та брокера повідомлень. Маючи низьку затримку доступу, Redis здатен обробляти сотні тисяч операцій за секунду – це робить його незамінною деталлю для підвищення швидкодії ML-інфраструктур. Redis підтримує різні структури даних – рядки, множини, списки, хеші, впорядковані множини, а також функціональність pub/sub для організації обміну повідомленнями. Додаткові модулі, типу RedisAI дають змогу зберігати ML-моделі безпосередньо у Redis та виконувати інференс у пам’яті.

Переваги Redis:

- дуже мала затримка доступу до даних завдяки in-memory архітектурі;

- підтримка TTL (time-to-live) для автоматичного видалення тимчасових даних;

- інтеграція із системами розподіленої обробки через pub/sub і streams;

- можливість масштабування через Redis Cluster.

Недоліки Redis:

- обмежений обсяг до доступної пам’яті (залежить від конфігурації серверів);

- потреба у додаткових механізмах для забезпечення постійності даних (наприклад snapshotting, journaling);

- не підходить як єдине сховище для великих обсягів даних, що потребують зображення на диску.

Сфери застосування Redis – кешування результатів ML-інференсу для прискорення відповідей, реалізація систем короткострокового зберігання проміжних даних, управління чергами завдань, організація швидких lookup-таблиць для рекомендаційних систем.

Таблиця 2.2.

Порівняльна таблиця популярних рішень між собою

Платформа	Тип СУБД	Сховище	Затримка	Швидкість	Підтримка ML	Масштабування
PostgreSQL	Реляційна	Row	Середня	Середня	Висока	Вертикальне
MongoDB	NoSQL	Документ	Низька	Висока	Середня	Горизонтальне
Cassandra	NoSQL	Колонки	Низька	Висока	Низька	Горизонтальне
ClickHouse	Аналітична	Колонки	Низька	Висока	Середня	Горизонтальне
Amazon S3+Parquet	Об’єктне	Колонки	Висока	Низька	Висока	Необмежене
Big Query	Хмарна аналітика	Колонки	Низька	Висока	Висока	Автоматичне
Redis	In-memory	Key-value	Дуже низька	Висока	Висока (RedisAI)	Обмежене

Вибір сховища даних у ML-завданнях залежить від типу, обсягу даних, швидкодія та характеру навантаження (онлайн або офлайн).

Для реального часу краще використовувати Redis або Cassandra, вони мають низьку затримку та можливість горизонтального масштабування.

Для офлайн-аналітики: ClickHouse, BigQuery, Amazon S3 + Parquet/OR – це високопродуктивні платформи для читання великих обсягів і історичних даних.

PostgreSQL та MongoDB – це універсальні рішення для середніх навантажень, та вони добре інтегруються з ML-інструментами.

Гібридна аналітика – PostgreSQL + Redis + S3 забезпечує високу адаптивність, оптимізуючи витрати, продуктивність і гнучкість ML-інфраструктур.

РОЗДІЛ 3.
ПОРІВНЯННЯ ПІДХОДІВ ЗБЕРІГАННЯ ТА ОБРОБКИ ДАНИХ

3.1 Розробка моделі порівняння підходів зберігання та обробки даних

У рамках реалізації практичної частини дослідження було поставлено завдання головною ціллю якого – це здійснити порівняльну оцінку ефективності процесу підготовки даних до машинного навчання при використанні різних СКБД та форматів зберігання. Увага буде концентруватися на впливі бази даних на продуктивність та ресурсоємкість підготовчого етапу моделювання.

Проблема бінарної класифікації структурованих (табличних) даних – це приклад задачі машинного навчання яку було обрано. Це доволі типова проблема у аналітиці – зокрема, у сферах фінансових послуг, маркетингу або електронної торгівлі. Таблична форма даних надає зручності її інтеграції з класичними реляційними СКБД, так і з гнучкими NoSQL-системами.

Як джерело даних, мною було використано відкритий набір “Bank Marketing Data Set”, який був оприлюднений на платформі UCI Machine Learning Repository. Цей датасет містить інформацію, яка була зібрана під час маркетингових кампаній одного з португальських банків, і включає у себе характеристики клієнтів: соціально-демографічні показники, фінансові параметри, дані про попередні контакти та цільову змінну – що сигналізує про успішність залучення клієнта до депозитної програми [12 - 14].

Структура вхідної вибірки. Розглянутий мною набір містить понад 45000 спостережень (тобто рядків) та 17 змінних (стовпців), серед яких:

- категоріальні змінні: професійна діяльність (job), рівень освіти (education), сімейний статус (material);

- числові змінні: вік респондента (age), залишок на рахунку (balance), кількість контактів під час кампанії (campaign);

- цільовий клас (y), представлений у вигляді бінарної мітки з можливим значенням yes або no.

Вибір моделі для задачі класифікації. Щоб вирішити поставлену задачу мною було використано модель штучної нейронної мережі MLP (Multi-Layer Perceptron), яка в свою чергу демонструє високу ефективність при роботі із табличними даними які включають у себе як числові, так і категоріальні ознаки. Модель реалізовано з використанням фреймворку TensorFlow у поєднанні з Keras – високорівневим інтерфейсом для побудови нейронних мереж.

Основні параметри моделі:

- вхідний шар: кількість нейронів відповідає числу ознак після перетворення (категоріальні змінні були закодовані за допомогою one-hot-encoding);

- приховані шари: два щільно з’єднані шари по 64 нейрони кожен з функцією активації ReLu;

- вихідний шар: один нейрон з активацією sigmoid, що відповідає за бінарний поділ класів;

- функція втрат: binary_crossentropy.

Оптимізаційний алгоритм: Adam.

Мета дослідження та варіанти її реалізації.

1. Застосування PostgreSQL як сховища, з подальшим формуванням навчальної вибірки SQL-запитів.

2. Використання MongoDB з динамічним вилученням даних та перетворенням структури на етапі отримання.

3. Опрацювання даних у форматі Parquet, збережених у локаційній файловій системі з подальшим завантаженням засобами бібліотек pandas та pyarrow.

Так саме є характеристики, за якими буде проведено оцінювання:

- тривалість процесу підготовки даних до навчання моделі;

- рівень використання оперативної пам’яті під час підготовки;

- загальний час навчання нейронної мережі;

- якість класифікації за метриками точності (accuracy), повноти (recall) та гармонічного середнього (F1-score).

3.2 Вибір середовищ зберігання даних та структура їх організації

Під час підготовки даних для задач машинного навчання важливою деталлю є не лише вміст та якість самих даних, але й оптимальність рішень щодо їхнього зберігання, доступу та попередньої обробки. У цьому контексті, у рамках експерементального дослідження мою було здійснено оцінювання ефективності трьох поширених підходів до організації зберігання даних за допомогою реляційної СКБД PostgreSQL, документоорієнтованої NoSQL MongoDB, і також використання формату колонкового зберігання файлів Parquet у файловій системі [12, 14].

Розглянемо, чому були обрані ці технологічні платформи. Кожна з обраних платформ реалізує різні концептуальні моделі зберігання структурованої інформації, що в свою чергу дозволяє отримати широкий спектр аналітичних висновків для їхнього порівняння.

PostgreSQL є представником реляційних СКБД, які в свою чергу підтримують транзакційність, сувору схему даних та потужну мову запитів SQL. PostgreSQL демонструє високу надійність при роботі із табличними даними, дає можливість використовувати індексацію, агрегаційні операції та має гнучкі можливості розширення, включно з інтеграцією Python-середовища за допомогою бібліотеки psycopg2 або ORM-інструменту SQLAlchemy.

MongoDB як документоорієнтована база даних надає можливість зберігати записи у вигляді об’єктів BSON, що забезпечує гнучкість при роботі зі слабоструктурованими або динамічно змінними схемами. Можливість обробки даних та підтримка агрегатних операцій роблять її зручною для реалізації ETL-процесів.

Parquet – це формат зберігання, який оптимізований для колонкового доступу, який особливо ефективний при аналітичному навантаженні. Використання Parquet дозволяє суттєво зменшити обсяг даних, що зчитуються, завдяки вибірковому доступу до колонок. Формат широко підтримується у сучасних бібліотеках для аналітики в Python (pandas, pyarrow, fastparquet) та дозволяє масштабовану обробку великих обсягів даних.

Організація структури даних у кожному середовищі. З метою об’єктивного порівняння дані з набору Bank Marketing було трансформовано в ідентичну логістичну структуру для усіх названих систем зберігання. У PostgreSQL дані було імпортовано до уніфікованої таблиці зі строго визначеною схемою. Приклад наведено на рис. 3.1.

```
CREATE TABLE marketing_data (
  id SERIAL PRIMARY KEY,
  age INTEGER,
  job TEXT,
  marital TEXT,
  education TEXT,
  balance NUMERIC,
  contact TEXT,
  y BOOLEAN
);
```

Рис. 3.1. Організація PostgreSQL

У MongoDB кожен запис представлений у вигляді окремого документа зі збереженням всіх ключових полів. Приклад наведено на рис. 3.2.

Рис. 3.2. Організація MongoDB

У Parquet дані було збережено у вигляді табличного файлу, де кожен рядок відповідає одному запису, а колонки відповідають змінним. Завантаження і фільтрація виконувалися з використанням функції pandas.read_parquet() з підтримкою параметра filters для вибіркового зчитування. Приклад наведено на рис. 3.3.

```
import pandas as pd

columns_to_load = ['age', 'job', 'balance', 'y']
df = pd.read_parquet("bank_data.parquet", columns=columns_to_load)

df_filtered = pd.read_parquet(
    "bank_data.parquet",
    filters=[('y', '=', True)]
)
```

Рис. 3.3. Організація Parquet

Засоби інтеграції з Python. Для забезпечення автоматизованої взаємодії між джерелами даних та обчислювальним середовище були застосовані відповідні інструменти:

- psycopg2 або SQLAlchemy для PostgerSQL;
- pymongo для MongoDB;
- pyarrow чи fastparquet у випадку з Parquet-файлами.

Усі ці бібліотеки інтегрувалися в єдиний оброблювальний конвеєр, що включав у себе етапи підключення до джерела, вилучення даних, їх трансформацію, формування batch-вибірок та передачу їх на вхід моделі машинного навчання.

3.3 Реалізація пайплайну обробки та навчання (ETL – ML model)

Процес підготовки даних для навчання моделей машинного навчання потребує побудови надійного, адаптивного та масштабованого пайплайну, котрий буде поєднувати всі ключові етапи обробки – починаючи з вилучення даних із сховищ, закінчуючи подачею їх до моделі. У межах мого дослідження було реалізовано повноцінний ETL-ML ланцюг, котрий охоплює усі стадії – витяг (Extract), трансформація (Transform), завантаження (Load), а також етап навчання моделі (Train). Таких підхід забезпечив цілісну автоматизовану структуру.

Архітектура пайплайну. Конструювання системи відбувалася відштовхуючись від модельної архітектури, яка дозволяє варіативно змінювати джерело даних без модифікації логіки обробки чи навчання. Це стало передумовою для здійснення порівняльного аналізу між трьома поширеними засобами зберігання – PostgreSQL, MongoDB, Parquet. Кожне рішення було інтегровано як належний модуль, що відповідає за етап Extract.

Етапи обробки даних:

1. Витяг даних (Extract).

a) для PostgreSQL була використана бібліотека psycopg2 з виконанням SQL-запитів до цільових таблиць;

b) для MongoDB – клієнтська бібліотека pymongo, яка в свою чергу дозволяла здійснювати запити з гнучкою умовною логікою;

c) для Parquet-файлів використовувалася функція pandas.read_parquet() з можливістю селективного зчитування колонок та фільтрації рядків, що забезпечує ефективність I/O-операцій.

2. Трансформація (Transform). Усі дані проходили стандартну процедуру попередньої обробки, яка містила у собі:

a) кодування категоріальних змінних(one-hot encoding);

b) масштабування числових ознак (SrandartScaler);

c) усунення пропущених значень.

Усі ці операції були реалізовані за допомогою бібліотек pandas та scikit-learn.

3. Завантаження (Load). Після трансформації дані перетворювалися у структури numpy.ndarray або pandas.DataFrame для подальшого формування навчальних та валідаційних батчів. Мною було реалізовано попереднє кешування та побудова міні-пакетів відповідно до заданого розміру партій.

4. Навчання моделі (Train). Архітектура моделі передбачила:

a) вхідний шар;

b) один прихований шар з активацією ReLU;

c) вихідний шар з sigmoid для вирішення задачі бінарної класифікації.

Саме навчання здійснювалося за допомогою функції втрат binary_crossentropy, оптимізатора Adam, реалізованих у фреймворку TensorFlow/Keras.

3.4 Порівняльний аналіз продуктивності

Здійснимо порівняльну оцінку ефективності трьох підходів до зберігання та обробки даних, що застосовувалися в рамках реалізованого ETL-ML пайплайну. Підходи, оцінка яких проводилась – PostgreSQL, MongoDB, Parquet. Порівняння здійснювалося за головними технічними параметрами, що впливають на загальну продуктивність системи:

- час, витрачений на підготовки даних до навчання (табл 3.1);
- споживання оперативної пам’яті (табл 3.2);
- загальний час навчання моделі (табл 3.3);
- якість моделі за метриками (точність, повнота, гармонічне середнє) (табл 3.3).

Для забезпечення об’єктивності тестування, всі експерименти були проведені на єдиному датасеті обсягом 45000 записів. Він містив – числові, категоріальні та булеві змінні. Під час кожного етапі виконувався один і той самий алгоритм навчання.

Таблиця 3.1.

Час підготовки даних (ETL).

Джерело даних	Час витягу,с	Час трансформації,с	Сумарно(ETL),с
PostgreSQL	3.25	4.10	7.35
MongoDB	2.81	4.12	6.93
Parquet	1.14	3.85	4.99

Таблиця 3.2.

Споживання оперативної пам’яті(RAM)

Джерело даних	Споживання RAM, МБ
PostgreSQL	420
MongoDB	465
Parquet	312

Таблиця 3.3.

Час навчання моделі та якість.

Джерело даних	Час навчання,с	Точність	Повнота	Гармонічне середне
PostgreSQL	11.82	0.891	0.859	0.876
MongoDB	11.67	0.887	0.854	0.872
Parquet	10.04	0.893	0.864	0.879

Для узагальнення побудована підсумкова таблиця 3.4, яка демонструє ефективність кожного формату зберігання за п’ятьма основними критеріями. Для кожної метрики прораховано оцінку. Оцінка 1 – найкращий результат, оцінка 2 – середній, оцінка 3 – найгірший.

Таблиця 3.4.

Оцінка форматів зберігання

Критерій	PostgreSQL	MongoDB	Parquet
Час ETL	3	2	1
Час навчання	2	3	1
Точність	3	2	1
Повнота	2	3	1
Гармонічне середнє	2	3	1
Споживання RAM	2	3	1
Сумарний бал	14	16	6

3.4 Оптимізація продуктивності: підходи та результати

Під час розробки високопродуктивного середовища для зберігання та підготовки даних до машинного навчання, дуже важливу роль відіграє не лише тип обраного сховища, алей й способи організації доступу до інформації. Дивлячись на результати, які були вже отримані в результаті попередніх досліджень, є потенціал для підвищення ефективності за рахунок реалізації додаткових оптимізаційних технік. Зараз здійсню порівняльну оцінку чотирьох основних підходів до оптимізації: кешування, пакетного завантаження (batch), індексації та патриціонування.

1. Кешування: прискорення повторного доступу до даних.

Кешування представляє собою один із найбільших механізмів для зниження навантаження на системи зберігання при повторному використанні даних. У рамках експерименту було реалізовано попереднє збереження трансформованих наборів у форматі NumPy у оперативній пам’яті для подальшого використання при крос-валідації.

Таблиця 3.5.

Вплив кешування на повторне навчання моделі

Метод	Без кешу(с.)	З кешем (с.)	Скорочення часу (%).
Повторне навчання	18.92	13.61	-28.1%

Результати показують, що кешування трансформованих масивів зменшує потребу у повторному зчитуванні з диска, тим самим скорочуючи загальний час повторного навчання.

2. Пакетне завантаження: контроль за споживанням ресурсів.

Технологія batch-завантаження дає можливість зменшити пікове споживання оперативної пам’яті, шляхом обробки даних фрагментами. У дослідженні було реалізовано завантаження з Parquet-файлів з розміром батча 1024 записи.

Таблиця 3.6.

Вплив batch-завантаження на ресурси та час навчання

Параметр	До оптимізації	Після batch-завантаження	Покращення (%)
Пікове споживання RAM, МБ	832	366	-56%
Час навчання (с.)	10.04	8.81	-12.2%

Цей від оптимізації дозволив суттєво зменшити оперативну пам’ять, яка необхідна для обробки одного епоху, та зменшити тривалість навчання.

3. Індексація: пришвидшення витягу з БД.

Індексація в реляційних системах та документоорієнтованих БД забезпечує значне прискорення виконання фільтраційних запитів. У

PostgreSQL було реалізовано B-tree-індекси для user_id та timestamp, а у MongoDB – аналогічні індекси для полів з високою селективністю.

Таблиця 3.7.

Ефективність індексації в базах даних

Система	Без індексації(с).	З індексацією (с).	Прискорення (%)
PostgreSQL	3.25	1.91	-41.2%
MongoDB	2.81	1.76	-37.4%

Ці результати свідчать про те, що доцільно використовувати індекси для підвищення швидкодії при витязі з великих сховищ.

4. Патриціонування: масштабованість при аналітичній обробці.

Патриціонування являє під собою логічне розбиття великих таблиць або колекцій, що забезпечує покращення продуктивності при агрегаціях та паралельній обробці. Для PostgreSQL було використано range_partitioning та timestamp, для MongoDB – горизонтальне розподілення даних (шардінг) за user_id.

Таблиця 3.8.

Ефект патриціонування на аналітичні запити.

Сховище	Без партицій (с).	З партиціями (с).	Прискорення (%)
PostgreSQL	2.61	1.75	-32.9%
MongoDB	2.44	1.68	-31.1%

Таким чином партиціонування підтвердило свою ефективність у задачах часової агрегації та при симуляції багатопотокових запитів.

5. Сумарна оцінка ефекту оптимізації.

На останньому етапі було підведено вплив усіх використаних технік на загальний час виконання повного пайплайну (ETL+навчання).

Середнє скорочення часу на 35% показує нам високу ефективність комплексного впровадження механізмів оптимізації, незалежно від типу сховища.

Таблиця 3.9.

Загальний вплив оптимізації на продуктивність

Джерела даних	Початковий час (с).	Оптимізований час (с).	Прискорення (%)
PostgreSQL	19.17	12.42	-35.2%
MongoDB	18.60	12.08	-35%
Parquet	15.03	9.72	-35.3%

У таблицях 3.8 та 3.9 немає Parquet через те, що Parquet – це файловий колонковий формат, а не система управління базами даних. Через це він не підтримує класичну індексацію, які властиві PostgreSQL та MongoDB. Parquet реалізує міні-індексацію на рівні колонок та row groups, але це більше стосується оптимізації читання при скануванні, а ніж доступу за ключами. Патриціонування у Parquet реалізується на рівні файлової системи або каталогу, а не через DDL-конструкції або механізми СУБД. Основними джерелами оптимізації Parquet є: колонкове зберігання; міні-індексація та статистика на рівні блоків; стиснення та кодування; row groups та паралелізація.

ВИСНОВОК

У межах виконаної кваліфікаційної бакалаврської роботи було досліджено особливості організації зберігання, обробки та підготовки даних для задач навчання нейронних мереж. Актуальність обраної теми зумовлена стрімким зростанням обсягів інформації, що обробляється в процесі машинного навчання, а також необхідністю підвищення ефективності ETL-процесів у прикладних інтелектуальних системах.

У першому розділі роботи розглянуто теоретичні основи функціонування нейронних мереж, принципи їх навчання, а також фактори, що впливають на якість класифікації. Було встановлено, що продуктивність моделей глибокого навчання значною мірою залежить не лише від архітектури нейронної мережі, але й від якості, структури та формату вхідних даних.

У другому розділі проведено аналітичний огляд підходів до зберігання даних та їхньої придатності до використання у системах машинного навчання. Особливу увагу приділено швидкості вибірки, можливості паралельної обробки, відповідності сучасним вимогам до обробки великих даних.

Третій розділ містить результати експериментального дослідження, у межах якого було побудовано експериментальне середовище для вимірювання часу виконання основних етапів підготовки та навчання моделі. Під час тестування оцінювались такі показники, як:

- тривалість витягу та трансформації даних;
- час тренування моделі;
- використання оперативної пам’яті;
- якісні метрики: повнота (recall) та гармонічне середнє (F1-score).

На основі проведених вимірювань встановлено, що формат зберігання Parquet продемонстрував стабільно вищу продуктивність порівняно з іншими

форматами. Це пояснюється колонковим принципом зберігання, ефективним стисненням даних та можливістю селективного читання, що суттєво знижує навантаження на систему при великому обсязі даних.

Окремо було досліджено застосування методів оптимізації – кешування, індексації, партиціювання та пакетного завантаження. Комбіноване використання цих методів дозволило зменшити загальний час обробки даних та навчання моделі більш ніж на 30% без зниження точності класифікації.

У результаті проведеного дослідження підтверджено: вибір формату зберігання даних, типу бази даних і застосування відповідних методів оптимізації мають вирішальне значення для ефективного функціонування систем машинного навчання.

На основі виконаної роботи сформульовано такі рекомендації щодо оптимізації організації даних у БД для ефективного навчання нейронних мереж:

- використовувати пакетне завантаження (batch loading) даних при передачі до моделей з обмеженим обсягом оперативної пам’яті;

- застосовувати індексацію на полях, що активно використовуються у фільтрації та сортуванні;

- впроваджувати кешування на етапах підготовки датасетів або після їх перетворення у формат, зручний для навчання;

- здійснювати партиціювання таблиць або колекцій при обробці часових або користувацьких даних;

- оцінювати ефективність використання БД не лише за стандартними CRUD-операціями, а й з урахуванням загальної продуктивності в процесі тренування моделей.

Таким чином, результати дослідження можуть бути використані для вдосконалення систем з глибоким навчанням, зокрема в частині інтеграції з базами даних та оптимізації процесів підготовки даних.

ПЕРЕЛІК ПОСИЛАНЬ

1.Гудфеллоу І., Бенджіо Й., Курвілл А. Глибоке навчання. – К.: Видавництво «Діалектика», 2018. – 800 с. (переклад Deep Learning by Goodfellow et al.)

2.Хейкен С., Маклін О. Нейронні мережі і навчальні машини. – К.: Видавництво «Вільямс», 2006. – 1100 с.

3.Chodorow K. MongoDB: The Definitive Guide. – 3rd ed. – O'Reilly Media, 2019. – 500 p.

4.Petrov D. PostgreSQL 13: The Complete Guide for Developers. – Apress, 2021. – 412 p.

5.Apache Software Foundation. Apache Parquet Documentation [Електронний ресурс] - https://parquet.apache.org/documentation/latest/ .

6. Apache Arrow Documentation [Електронний ресурс] - https://arrow.apache.org/docs/ .

7.Chen T., Li M., Li Y. et al. MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems 2015.

8.Abadi M., Barham P., Chen J. et al. TensorFlow: Large-scale machine learning on heterogeneous systems 2016.

9.Dean J., Ghemawat S. MapReduce: Simplified Data Processing on Large Clusters // Communications of the ACM. – 2008. – Vol. 51(1). – P. 107–113.

10.Lakehouse: A New Generation of Open Platforms That Unify Data Warehousing and Advanced Analytics // Databricks, 2020 [Електронний ресурс] - https://databricks.com/p/webinars/lakehouse-new-generation .

11.Van der Vaart A. W., Wellner J. A. Weak Convergence and Empirical Processes. – Springer, 2000.

12.Redmon J., Farhadi A. YOLOv3: An Incremental Improvement 2018. (Як приклад реальної ML-моделі з великим обсягом даних)

13.Dunning T., Friedman E. Time Series Databases: New Ways to Store and Access Data. – O'Reilly Media, 2014.

14.Sadalage P., Fowler M. NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. – Addison-Wesley, 2012.