НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і

спеціалізованих комп'ютерних систем

«На правах рукопису»
УДК 004.048

До захисту допущено:
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
«___»_____2024 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Системне програмування та спеціалізовані комп'ютерні системи»

зі спеціальності 123 «Комп'ютерна інженерія»

на тему: «Засоби оптимізації використання обчислювальних ресурсів у операціях машинного навчання»

Виконав:
студент ІІ курсу, групи КВ-21 мп
Марченко Олександр Борисович _____

Науковий керівник:
доктор технічних наук, професор
Романкевич Віталій Олексійович _____

Рецензент:
Посада, науковий ступінь, вчене звання,
Прізвище, ім'я, по батькові _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2024 року

ЗМІСТ

Глосарій

MLOps - операції машинного навчання.

Великі обсяги даних - обсяг даних, який не вміщується на фізичному диску, підключеному до одного навчального сервера.

Blob - двійковий великий об'єкт.

teraFLOPS - в обчисленнях, операції з плаваючою комою в секунду - міра продуктивності комп'ютера, корисна в галузях наукових обчислень, які вимагають обчислень з плаваючою комою. Один терафлопс дорівнює 1.000.000.000.000.000 (одному трильйону) FLOPS.

Waveform Audio File - стандарт формату аудіофайлів, розроблений компаніями IBM і Microsoft для зберігання аудіопотоку на персональних комп'ютерах.

Трансформація - результат маніпуляцій з даними, які можуть змінити їх, виділити певні ознаки або додати додаткові мітки, виконавши ручне або автоматизоване анотування.

Завдання обробки - крок конвеєра обробки даних.

Метадані - інформація, що описує конкретний файл, його властивості та іншу системну інформацію у структурованому представленні.

Партиція - виділена незалежна частина сховища, створена з метою розбиття даних на фрагменти.

Попередня обробка - ланцюжок перетворень, які застосовуються до даних перед їх запитом.

Обробка на льоту - ланцюжок перетворень, які застосовуються до даних під час запиту.

Датасет - набір даних.

Вступ

MLOps (також відомі як операції машинного навчання) - відносно нова галузь досліджень, яка постійно розвивається і досі має багато невирішених проблем. Основними цілями цієї галузі інженерної науки є зниження вартості інфраструктури машинного навчання, підвищення надійності процесів та прискорення отримання результатів. Для досягнення цих цілей ця дисципліна була створена як симбіоз DevOps та прикладної науки.

У цій дисертації розглядається комплекс засобів для оптимізації різних аспектів операцій машинного навчання, а саме:

- оптимізація використання ресурсів при збереженні великих даних;
- організація масштабованого середовища для попередньої обробки данних;
- пришвидшення доступу до даних для тренування моделей машинного навчання;
- оптимізація використання ресурсів продуктовим оточенням під час тренування моделей;

В сучасному інформаційному віці, де обробка та аналіз даних стали ключовими компонентами багатьох сфер життя, проблема збереження та доступу до великих обсягів даних стала актуальною як ніколи раніше. Машинне навчання, зокрема, потребує значних обсягів даних для навчання та вдосконалення моделей, і від цієї проблеми залежить успішність багатьох застосувань цієї технології.

Ця дисертація торкається проблеми збереження великих обсягів даних для операцій машинного навчання та розробці ефективних методів та стратегій для забезпечення їхньої доступності та швидкого використання в процесі навчання моделей. Дослідження, проведені в рамках цієї роботи, мають на меті вирішити цю проблему та внести

вагому концептуальний та практичний внесок у розвиток сучасних систем збереження та управління даними для машинного навчання.

Здійснюючи аналіз інноваційних технологій та методів, а також використовуючи результати власних досліджень, ми розглядаємо можливості оптимізації зберігання даних та розробки нових підходів для забезпечення ефективного використання ресурсів під час машинного навчання. Основні завдання дисертації включають в себе аналіз існуючих систем збереження даних, розробку нових засобів, а також експериментальне дослідження їхньої ефективності на реальних наборах даних.

Варто зазначити, що крім збереження даних перед нами також постає проблема їх обробки. Обсяги інформації зростають з кожним днем, обробка та аналіз даних стають важливими завданнями, що стоять перед науковими дослідниками, інженерами та аналітиками. Великі обсяги даних, їх різноманітність та складність створюють унікальні виклики та можливості для вдосконалення методів та інфраструктури обробки інформації. Особливо важливим стає завдання попередньої обробки даних, що передує їх аналізу та використанню в різних додатках, включаючи машинне навчання, наукові дослідження, фінансовий аналіз і багато інших.

В цій роботі ми також торкнемося створення та дослідження масштабного середовища для попередньої обробки даних, яке відповідає потребам у сфері обробки та аналізу великих обсягів інформації. Метою цього дослідження є розробка ефективних засобів, інструментів та стратегій для обробки та підготовки даних перед їх подальшим використанням.

- Розробка інфраструктури, яка дозволяє створювати та управляти масштабованими середовищами для попередньої обробки даних,

здатними підтримувати велику кількість даних та обчислювальних ресурсів.

- Розробка та оптимізація засобів та інструментів для швидкого та ефективного виконання операцій попередньої обробки, включаючи чистку, фільтрацію, агрегацію та трансформацію даних.
- Розробка механізмів забезпечення надійності, резервного копіювання та версіонування даних.
- Розробка механізмів, що дозволять швидко та легко отримувати доступ до оброблених даних для подальшого використання в машинному навчанні, аналізі даних та інших додатках.

Ця магістерська дисертація має на меті сприяти поглибленню розуміння принципів та технологій, що стоять за створенням масштабованих середовищ для попередньої обробки даних, а також сприяти подальшому розвитку сучасних методів обробки та аналізу даних, роблячи їх більш доступними та продуктивними для застосувань у машинному навчанні та інших сферах, де великі обсяги даних є важливим ресурсом. Віримо, що результати цієї дисертації сприятимуть подальшому розвитку технологій та забезпечать зручний та ефективний доступ до великих обсягів даних та сприятимуть покращенню якості та ефективності обробки даних у різних галузях і додатках, що використовують великі обсяги інформації. Цим ми полегшимо роботу для науковців, інженерів та інших фахівців у галузі машинного навчання та аналітики даних.

1. Проблеми використання ресурсів у операціях машинного навчання та їх наслідки

Розвиток сучасних інформаційних технологій та зростання обсягів доступної інформації спричинили значний прогрес у використанні машинного навчання в різних сферах діяльності. Операції машинного навчання стали важливим елементом роботи багатьох організацій та суб'єктів господарювання, а їхня роль у розв'язанні завдань та досягненні цілей стала найбільш актуальною.

У цьому розділі будемо детально розглядати проблеми, пов'язані з використанням ресурсів у операціях машинного навчання та їхні наслідки для різних галузей. Почнемо з визначення основних понять та компонентів операцій машинного навчання, щоб зрозуміти їхню структуру та роль у сучасному інформаційному суспільстві. Далі перейдемо до аналізу проблем, які виникають у процесі використання ресурсів у цих операціях, та розглянемо наслідки відсутності оптимізації.

Цей спрямований на розкриття ключових аспектів, пов'язаних із застосуванням машинного навчання у сучасному світі. Ми також визначимо області, де машинне навчання відіграє вирішальну роль і враховує активний розвиток штучного інтелекту та збільшення обсягів інформації в останні роки. Розділ покладе фундамент для подальшого дослідження та аналізу оптимізації використання ресурсів у операціях машинного навчання, який буде розглянутий в наступних розділах.

1.1. Основні поняття операцій машинного навчання

Однією з найбільш захоплюючих та перспективних галузей сучасної інформаційної технології є машинне навчання. Воно відкриває перед собою безмежні можливості для автоматизації процесів,

прогнозування, створення інтелектуальних систем та багато інших сфер використання. Але перед тим, як поглиблюватися в деталі оптимізації використання ресурсів у цих операціях, нам необхідно ретельно розібратися у базових поняттях та компонентах операцій машинного навчання.

Ми розпочнемо з визначення ключових термінів та понять, які становлять основу операцій машинного навчання. Ми розглянемо структуру та послідовність цих операцій, їхню роль та значення у великому спектрі сучасних застосувань. Під час огляду цих понять, ми розглянемо їхнє важливе значення в контексті автоматизації конвеєру тренування моделей машинного навчання, збереження даних, попередньої обробки, тренування, оцінки якості моделей та подальшого розгортання у продуктовому оточенні.

1.1.1. Складові операцій машинного навчання

Розглянемо складові операцій машинного навчання та їх автоматизацію, включаючи збір та збереження даних, попередню обробку, тренування моделей, оцінку якості моделей та подальше розгортання у продуктовому оточенні.

1. **Збір та збереження даних**

Збір та збереження даних є ключовими кроками для операцій машинного навчання. Автоматизація цього процесу включає в себе наступні аспекти:

- **Збір даних**

Автоматизована система може встановити з'єднання з джерелами даних, такими як бази даних, датчики, API веб-сервісів та інші, та регулярно завантажувати дані.

- **Каталогізація та індексація**

Зібрані дані повинні бути каталогізовані та індексовані для легкого пошуку та доступу. Автоматична індексація дозволяє швидко знаходити необхідні дані для тренування та аналізу.

- **Забезпечення безпеки**

Автоматизовані системи повинні забезпечувати захист даних, включаючи шифрування та автентифікацію, щоб забезпечити конфіденційність та цілісність даних.

2. **Попередня обробка даних**

Попередня обробка даних перед тренуванням моделей включає в себе ряд автоматизованих завдань:

- **Очищення даних**

Система може автоматично виявляти та видаляти аномалії, відсутні значення та шум у вхідних даних.

- **Нормалізація**

Автоматизована нормалізація даних дозволяє перетворити дані так, щоб вони знаходилися в одному масштабі, що полегшує тренування моделей.

- **Трансформація**

Деякі операції з обробки даних, такі як векторизація тексту чи кодування категорій, можуть бути автоматизовані для прискорення підготовки даних.

3. **Тренування моделей машинного навчання**

Автоматизація тренування моделей машинного навчання допомагає зберегти час та зусилля:

- **Розподіл ресурсів**

Система може автоматично розподіляти завдання тренування між різними ресурсами, такими як CPU та GPU, для максимізації продуктивності.

- **Оптимізація гіперпараметрів**

Автоматизована оптимізація гіперпараметрів дозволяє визначити оптимальні значення для параметрів моделі, що підвищує її ефективність.

- **Моніторинг процесу**

Система може надавати реальний час моніторинг процесу тренування та автоматично реагувати на можливі помилки чи перерви.

4. **Оцінка якості моделей**

Автоматизована оцінка якості моделей робить процес більш ефективним та точним:

- **Автоматичні метрики**

Система може автоматично обчислювати різні метрики якості моделей, такі як точність, витрати тощо.

- **Візуалізація**

Графіки та візуалізації допомагають зрозуміти, як модель працює на практиці, і допомагають виявити проблеми.

5. **Розгортання у продуктовому оточенні**

Розгортання моделей у продуктовому оточенні може бути автоматизованим процесом:

- **Інтеграція з продуктом**

Система може автоматично інтегрувати моделі у веб-сервіси, мобільні додатки та інші продукти.

- **Моніторинг та керування**

Автоматизована система забезпечує моніторинг та керування моделями у реальному часі, виявляючи та виправляючи проблеми.

- **Шкальованість**

Система може автоматично масштабувати ресурси для обробки великих обсягів запитів.

Автоматизована система для операцій машинного навчання допомагає зберегти час та зусилля дослідників та інженерів, забезпечуючи ефективне використання обчислювальних ресурсів та покращену якість моделей, що використовуються в продуктах та послугах.

1.1.2. Сфера застосування

Розглянемо різноманітні сфери застосування операцій машинного навчання, зокрема їх комерційну та наукову складову, враховуючи активний розвиток штучного інтелекту (ШІ) та зростаючі обсяги інформації за останні роки.

1. **Комерційна застосування операцій машинного навчання**

- **Реклама та маркетинг**

Персоналізована реклама: Операції машинного навчання використовуються для аналізу поведінки користувачів та створення індивідуальних рекламних пропозицій, що збільшує конверсію та ефективність рекламних кампаній.

Аналіз соціальних медіа: Машинне навчання допомагає відслідковувати та аналізувати відгуки користувачів у соціальних медіа, щоб компанії могли відповідати на запити та реагувати на тренди в реальному часі.

Прогнозування попиту: В сфері роздрібної та онлайн-торгівлі, машинне навчання використовується для прогнозування попиту на товари та оптимізації управління запасами.

- **Фінанси та фінансова аналітика**

Аналіз ризиків: Машинне навчання допомагає фінансовим установам виявляти ризики та аналізувати портфелі для прийняття обгрунтованих інвестиційних рішень.

Аналіз ринку: Великі обсяги фінансових даних можуть бути проаналізовані за допомогою алгоритмів машинного навчання для прогнозування ринкових трендів.

Виявлення шахраївства: Машинне навчання допомагає виявляти аномалії та несанкціоновану діяльність у фінансових транзакціях.

- **Електронна торгівля**

Рекомендаційні системи: Машинне навчання використовується для підбору товарів та послуг для користувачів на основі їхніх попередніх покупок та переглядів.

Оптимізація цін: Алгоритми машинного навчання допомагають визначати оптимальні ціни для товарів в режимі реального часу, з урахуванням попиту та конкурентної динаміки.

Обробка замовлень та логістика: Машинне навчання допомагає оптимізувати процеси обробки замовлень та доставки товарів.

- **Клієнтський сервіс**

Чат-боти та автоматична підтримка: Операції машинного навчання використовуються для створення чат-ботів та систем автоматичної підтримки, які взаємодіють з клієнтами та вирішують їхні питання.

Аналіз відгуків клієнтів: Машинне навчання допомагає відстежувати та аналізувати відгуки клієнтів, щоб вдосконалювати якість послуг та товарів.

2. **Наукова застосування операцій машинного навчання**

- **Дослідження в галузі штучного інтелекту**

Розвиток нових алгоритмів: Вчені активно працюють над розробкою нових методів навчання та архітектур нейронних мереж для досягнення кращої продуктивності та точності.

Зображення та обробка природних мов: Машинне навчання використовується для аналізу та розпізнавання зображень, а також

обробки природних мов, що має застосування в галузях, таких як комп'ютерний зор, обробка тексту та автоматичний переклад.

- **Медичні дослідження**

Діагностика та лікування: Машинне навчання допомагає виявляти хвороби на ранніх стадіях, створювати індивідуальні рекомендації щодо лікування та прогнозувати результати.

Аналіз геномів: Вивчення геномів за допомогою машинного навчання сприяє розумінню генетичних причин хвороб та розробці персоналізованих методів лікування.

- **Наука про дані**

Аналіз великих даних: Машинне навчання використовується для аналізу та виділення корисної інформації з великих обсягів даних, що дозволяє виявляти нові закономірності та тренди.

- **Астрофізика та космос**

Аналіз сигналів з космосу: Машинне навчання допомагає виявляти та аналізувати сигнали з космосу, такі як радіохвилі з далеких галактик.

Моделювання космічних явищ: Вчені створюють моделі для дослідження космічних явищ, таких як вибухи супернових та рух планет.

- **Екологічні дослідження**

Моніторинг змін в екосистемах: Машинне навчання допомагає визначати та аналізувати зміни в природних екосистемах, що є критичним для збереження біорізноманіття та здоров'я планети.

Прогнозування кліматичних змін: Вчені використовують дані та моделі машинного навчання для прогнозування кліматичних змін та вивчення впливу людської діяльності на навколишнє середовище.

Зростання обсягів інформації та активний розвиток штучного інтелекту роблять операції машинного навчання незамінним інструментом

в різних галузях, де аналіз та обробка даних грають ключову роль. Вони стають важливим чинником у досягненні успіху як в комерційних, так і в наукових застосуваннях.

1.1.3. Прикладне значення, роль у економічному розвитку і суспільстві

Розглянемо прикладне значення операцій машинного навчання, їхню роль у економічному розвитку та вплив на суспільство.

1. **Прикладне значення операцій машинного навчання**

- **Покращення якості продуктів і послуг**

Операції машинного навчання дозволяють компаніям створювати продукти та послуги вищої якості. Наприклад, у роздрібній торгівлі рекомендаційні системи допомагають користувачам знаходити товари, які їхній смак, зменшуючи кількість невдалих покупок. У медицині, машинне навчання допомагає виявляти хвороби на ранніх стадіях та розробляти ефективні методи лікування.

- **Підвищення ефективності операцій**

Машинне навчання дозволяє автоматизувати багато рутинних операцій, що призводить до підвищення продуктивності та зменшення витрат часу та ресурсів. Наприклад, виробництво може бути оптимізоване завдяки автоматизованій якості контролю, або торгові платформи можуть використовувати роботів для виконання операцій на фондовому ринку.

- **Зменшення ризиків та підвищення безпеки**

Машинне навчання дозволяє виявляти аномалії та передбачати ризики. У фінансовій сфері, алгоритми машинного навчання виявляють підозрілі транзакції, що допомагає уникнути шахрайства. У кібербезпеці,

машинне навчання використовується для виявлення вразливостей та запобігання атакам.

2. **Роль у економічному розвитку**

- **Створення нових ринків і робочих місць**

За допомогою операцій машинного навчання створюються нові ринки та індустрії. Наприклад, розвиток робототехніки і автономних систем створює попит на фахівців у сфері робототехніки та програмування. Також рекомендаційні системи створюють нові ринки для персоналізованих послуг та товарів.

- **Збільшення продуктивності**

Застосування операцій машинного навчання призводить до збільшення продуктивності у багатьох галузях. Виробництво, транспорт, фінанси та інші галузі отримують користь від автоматизації процесів та оптимізації виробництва.

- **Розвиток стартапів та інноваційних проектів**

Машинне навчання створює унікальні можливості для стартапів та інноваційних проектів. Молоді компанії можуть створювати інноваційні продукти та послуги на основі алгоритмів машинного навчання, що сприяє росту економіки.

3. **Вплив на суспільство**

- **Покращення якості життя**

Машинне навчання призводить до розвитку медицини, автономних транспортних засобів, робототехніки та інших технологій, що покращують якість життя людей. Відкриття нових методів діагностики та лікування хвороб рятує життя, а роботи-помічники полегшують повсякденне життя.

- **Соціальні та економічні виклики**

Зростання застосування машинного навчання також породжує нові етичні та соціальні питання, такі як конфіденційність даних, безпека ШІ, та рівень доступу до новітніх технологій. Суспільство вимагає ефективних регуляторних механізмів та етичних стандартів для забезпечення справедливого та безпечного використання машинного навчання.

Загальний висновок полягає в тому, що операції машинного навчання мають велике прикладне значення для поліпшення якості продуктів і послуг, сприяють економічному розвитку через створення нових ринків і підвищення продуктивності, а також мають значний вплив на суспільство, розв'язуючи сучасні виклики і породжуючи нові.

1.2. Проблеми використання ресурсів у операціях машинного навчання

Розділ 1.2. розглядає ключові проблеми, пов'язані з використанням ресурсів у машинному навчанні. Машинне навчання, будучи однією з найбільш динамічно розвиваючихся областей інформаційних технологій, стикається з безліччю викликів, пов'язаних із забезпеченням і оптимізацією ресурсів. Ці виклики включають в себе обчислювальні складнощі, використання пам'яті, управління даними, енергоспоживання, а також вартість ресурсів.

Огляд Основних Проблем

Машинне навчання, з його широким спектром застосувань від обробки природної мови до комп'ютерного зору, залежить від великих обсягів даних та складних алгоритмів. Це ставить підвищені вимоги до обчислювальних ресурсів, що може включати спеціалізоване апаратне

забезпечення, таке як GPU та TPU. Проблеми управління, зберігання та обробки великих наборів даних, а також забезпечення їх безпеки та конфіденційності, відіграють важливу роль у процесах машинного навчання.

Виклики ефективності та економічності

Ефективність використання ресурсів стає особливо важливою, оскільки недостатньо оптимізовані системи можуть призводити до непотрібного витрачання часу, енергії та фінансових коштів. Витрати на обладнання, електроенергію та технічне обслуговування є значними, особливо у великих масштабних проектах. Тому, вивчення та розробка методів для оптимізації використання ресурсів є критично важливим для подальшого розвитку та масштабування технологій машинного навчання.

У цьому розділі ми детально розглянемо кожну з цих проблем, аналізуючи їх причини, наслідки та можливі шляхи вирішення. Від розуміння технічних складнощів до оцінки економічного впливу, цей розділ має на меті надати глибоке розуміння викликів, пов'язаних з ресурсами у машинному навчанні, та вказати шляхи їх ефективного управління.

1.2.1. Інфраструктурні складнощі

Інфраструктурні складнощі у сфері машинного навчання є фундаментальною проблемою, що впливає на всі аспекти цих операцій. В цьому розділі розглянемо інфраструктурні проблеми, з якими стикаються при використанні машинного навчання, та детально розглянемо додаткові важкості та аспекти, пов'язані з інфраструктурою

1. **Прогнозування необхідної обчислювальної потужності**

Проблема: Один із основних викликів у машинному навчанні полягає в прогнозуванні потреби в обчислювальній потужності для конкретного завдання. Недостатньо обчислювальних ресурсів може призвести до затримок у тренуванні моделей, тоді як зайві ресурси призводять до непотрібних витрат.

Існуючі рішення: Один з підходів до розв'язання цієї проблеми запозичений із класичного програмування високонавантажених систем - автоматизоване масштабування обчислювальної потужності. Використання хмарних обчислювальних платформ, які дозволяють динамічно змінювати обсяги ресурсів в залежності від потреби, може бути корисним рішенням. Додатково, аналітика даних та моніторинг використання ресурсів може допомогти в управлінні ресурсами більш ефективно.

Проблеми існуючих рішень: Все ще залишається відкритим питання оцінки необхідної кількості ресурсів для тренування моделі. Адже не завжди можно точно сказати скільки пам’яті треба, адже це залежить від вибірки даних, архітектури моделі, розміру набору данних для ітерації та гіперпараметрів. Час необхідний на тренування моделі також не так просто предикувати, оскільки ми не можемо точно сказати на якій ітерації ми хочемо зупинитись, а час однієї ітерації залежить від вищезазначених зміних. Попри те даних підхід може бути корисним для повторюваних операцій попередньої обробки даних та задля автоматичного розгортання обчислювальних машин у кластері для тренування моделей.

2. **Комунікаційні затримки (затримка отримання даних)**

Проблема: У складних системах машинного навчання, де великі обсяги даних обробляються розподілено, комунікаційні затримки можуть

суттєво уповільнити операції. Зокрема, затримки при отриманні даних зі сховища даних або передачі даних між вузлами кластера можуть призвести до збільшення часу виконання.

Існуючі рішення: Для подолання цього виклику використовуються різні техніки оптимізації. Однією з них є кешування даних на локальних вузлах, що дозволяє зменшити кількість запитів до віддаленого сховища. Крім того, використання високошвидкісних мереж та оптимізація передачі даних можуть допомогти зменшити затримки.

Проблеми існуючих рішень: Швидкість отримання даних також залежить від типу та організації сховища, саме це ми будемо оптимізовувати далі у цій роботі, адже кешування не завжди є ефективним способом оскільки ми маніпулюємо з великими масивами даних і перш ніж елемент повториться ми повинні повністю проітеруватись через увесь масив даних. Локальне кешування петабайтів інформації є неможливим, або дуже коштовним з існуючим інфраструктурними можливостями.

3. Масштабуванні інфраструктури при зростанні кількості даних

Проблема: Зі зростанням обсягів даних, інфраструктура для їх обробки повинна масштабуватися. Це може бути складною задачею, оскільки збільшення кількості даних може вимагати значних змін у системі.

Існуючі рішення: Однією з ключових стратегій є використання розподіленої обробки даних та розподіленої обчислювальної потужності. Використання технологій, таких як Apache Hadoop або Apache Spark, дозволяє обробляти великі обсяги даних паралельно на багатьох вузлах. Також важливо розробляти ефективні алгоритми обробки даних, які можуть працювати з великими обсягами інформації.

4. **Доступ до обчислювальних ресурсів**

Проблема: Деякі операції машинного навчання можуть вимагати спеціалізованого обладнання, такого як графічні процесори (GPU) для швидкого тренування моделей. Крім того, паралелізація та узгодження кластерів машин можуть бути важливими для забезпечення ефективного використання ресурсів.

Існуючі рішення: Важливо мати можливість легко розширювати інфраструктуру для включення спеціалізованого обладнання, якщо це необхідно. Використання оркестраторів контейнерів, таких як Kubernetes, може спростити управління обчислювальними ресурсами. Для ефективного використання GPU та узгодження роботи машин в кластері можна використовувати технології, такі як NVIDIA CUDA або Apache Mesos.

5. **Забезпечення відмовостійкості**

Проблема: Втрати обчислювальних ресурсів або даних можуть призвести до серйозних перебоїв у роботі системи. Забезпечення відмовостійкості є важливим аспектом інфраструктури машинного навчання.

Існуючі рішення: Для забезпечення відмовостійкості можна використовувати різні підходи, такі як резервне копіювання даних, розподілення ресурсів між різними центрами обробки даних, автоматичне відновлення та моніторинг стану системи.

6. **Управління конфігураціями та завданнями**

Проблема: У складних системах машинного навчання необхідно ефективно управляти конфігураціями та завданнями. Це може включати в себе налаштування параметрів моделей, розподілення завдань між вузлами та моніторинг стану системи.

Існуючі рішення: Використання інструментів управління конфігураціями, таких як Ansible або Puppet, дозволяє автоматизувати налаштування системи. Для управління завданнями та моніторингу можна використовувати інструменти, такі як Apache Airflow або Prometheus.

Проблеми існуючих рішень: Дані рішення чудово оркеструють та спостерігають за кроками попередньої обробки та MLOps конвеєру загалом, проте для середовище розробки повинно бути орієнтоване на вчених і поліпшувати простоту експериментування, в той час як дані системи орієнтовані на автоматизацію програмними шляхами та потребують додаткової експертизи для налаштування, що веде до великих операційних витрат, адже кожен науковець повинен глибино розуміти інструментарій та принципи інженерії програмного забезпечення.

7. **Безпека та Захист Даних**

Проблема: Однією з найважливіших аспектів інфраструктури машинного навчання є забезпечення безпеки та захисту даних. Зберігання та обробка великих обсягів даних вимагає заходів для запобігання несанкціонованому доступу та втраті інформації.

Рішення: Для забезпечення безпеки можна використовувати різні заходи, включаючи шифрування даних, багаторівневий захист, автентифікацію та авторизацію. Регулярний аудит та моніторинг системи також є важливими для виявлення потенційних загроз та інцидентів безпеки.

Інфраструктурні складнощі в операціях машинного навчання включають в себе ряд важкостей, таких як прогнозування потреби в обчислювальній потужності, комунікаційні затримки, масштабування інфраструктури, доступ до спеціалізованого обладнання, відмовостійкість, управління конфігураціями та завданнями, а також

безпека та захист даних. Розуміння і вирішення цих проблем є критичним для успішних операцій машинного навчання. У наступних розділах ми розглянемо способи вирішення цих викликів та оптимізації інфраструктури для машинного навчання.

1.2.2. Робота з даними

Робота з даними в машинному навчанні стикається з рядом складних викликів, особливо коли мова йде про неструктуровані та напів-структуровані дані. Ці виклики можуть значно впливати на якість та ефективність моделей машинного навчання.

При роботі з великими даними виникають різні виклики:

1. **Збір даних**

Збір даних у машинному навчанні, особливо неструктурованих та напів-структурованих, часто супроводжується складнощами, пов'язаними з неповнотою, неконсистентністю та неточністю інформації. Наприклад, збір текстових даних з соціальних мереж може містити невідфільтрований шум, сленг, або випадкові помилки, що ускладнює подальшу обробку та аналіз. Наслідки таких недоліків можуть бути значними: від спотворення результатів аналізу до неправильного висновків та галюцюнування моделей машинного навчання. Вирішення цієї проблеми вимагає розробки та застосування складних алгоритмів попередньої обробки, які можуть ідентифікувати та коригувати неконсистентності та помилки.

Також постає проблема виявлення та інтеграції розрізнених та неповних даних. Наприклад, зібрані з соціальних мереж текстові дані можуть бути фрагментованими та містити неповну інформацію. Як наслідок неповність даних може призвести до неправильних висновків та потенційне створити упереджені моделі через нерепрезентативність

даних. До прикладу аналіз соціальних медіа без урахування всіх джерел може дати упереджену картину суспільних настроїв.

2. **Попередня обробка даних**

Часто даними є великі тексти, зображення, які потребують значних ресурсів для обробки. Підготовка великих обсягів неструктурованих даних може бути трудомісткою. Ресурси необхідні на обробку великих обсягів таких даних, як-от очищення тексту від непотрібних символів або шуму в зображеннях можуть суттєво вплинути на кінцеву вартість тренування моделі. Як наслідок неефективна обробка даних призводить до збільшення часу на підготовку даних, що може затримати початок тренування моделей. Це призводить до затримок в проектах та збільшенню витрат на обчислювальні потужності.

3. **Паралелізація обробки**

Паралелізація обробки даних в машинному навчанні є ключовою для ефективного використання обчислювальних ресурсів, особливо при роботі з великими обсягами даних. Тим не менш, цей процес може бути ускладнений через потребу синхронізації даних між різними обчислювальними вузлами, що може призвести до комунікаційних затримок та нерівномірного навантаження. Наприклад, у великомасштабних системах, таких як рекомендаційні системи або обробка природної мови, затримки в обробці даних можуть серйозно вплинути на продуктивність та швидкість відгуку системи. Рішення полягає у розробці ефективних механізмів розподілу завдань та оптимізації обчислювальних процесів. Узгодження обробки даних між декількома процесорами чи вузлами в розподіленій системі, що може бути складним при аналізі великих датасетів. Налагодження ефективного паралелізму вимагає складної архітектури. Неефективне використання обчислювальних ресурсів призводить до погіршення продуктивності та

зростання витрат. Навіть більше неправильно налаштована паралельна обробка або неправильно підібрані обчислювальні машини можуть уповільнити процес обробки замість прискорення.

4. Масштабування та зберігання великих даних

Масштабування та зберігання даних є однією з найважливіших проблем у машинному навчанні, особливо у контексті зростання обсягів неструктурованих або напів-структурованих даних. Ця проблема стає ще більш вираженою зі збільшенням кількості даних, що генеруються і використовуються для навчання та оцінювання моделей.

З розвитком технологій і збільшенням обсягу доступних даних, організації та дослідники зіштовхуються з викликом їх ефективного зберігання та доступу. Великі обсяги даних потребують значних ресурсів для зберігання, а також високої швидкості доступу для їх обробки. Традиційні системи зберігання можуть бути неефективними або високовартісними при масштабуванні, особливо для даних, що вимагають швидкого доступу.

Часто виникають проблеми для забезпечення доступного та швидкого сховища для зберігання великих обсягів даних, наприклад, відеоданих для системи розпізнавання образів, або аудіо записів для систем синтезу голосу. При неправильно обраних підходах до зберігання даних забезпечення ключових аспектів функціональності може потребувати стрімкого збільшення витрат на зберігання даних і призводити до потенційних затримок в доступі до даних, що є критичним при тренуванні моделей машинного навчання. Управління великими обсягами даних може бути складним, вимагаючи спеціалізованого програмного забезпечення та апаратного обладнання.

5. **Вибірка даних для експериментів**

Викликом постає фільтрація даних великих датасетів для тренування моделей. Такі операції не повинні створювати додаткових затримок, при цьому набори даних не завжди є консистентними, що унеможливлює фільтрацію за певними властивостями, що не притаманні конкретному набору даних.

Крім того необхідно забезпечити рівномірне представлення різних класів у даних, наприклад, забезпечення балансу між різними категоріями зображень для системи класифікації. Без механізмів, що вирішують призводить до поганих прогнозів на певних типах даних. Упередженість у вибірках даних є значним викликом, який може призвести до створення упереджених моделей машинного навчання. Це особливо актуально у задачах, де необхідно забезпечити представництво різних груп. Наприклад, упередженість у медичних датасетах може призвести до розробки діагностичних інструментів, які неефективні для певних популяцій. Вирішення цієї проблеми вимагає уважного підходу до збору та аналізу даних, забезпечуючи різноманітність та представництво різних груп. Іншим аспектом цієї проблеми є тенденція до вибору даних, які не відображають реальну різноманітність. Як результат використання даних лише з одного географічного регіону для глобальної проблеми веде до створення моделей, що не є ефективними або об'єктивними для всіх користувачів. На жаль збір різноманітних даних з різних джерел та культур, що вимагає значних зусиль.

6. **Якість даних**

Проблема якості даних, включаючи їх "брудність" та неточність, суттєво впливає на кінцеву якість моделей машинного навчання. Дані, згенеровані штучним інтелектом або зібрані з нестандартних джерел, можуть містити помилки або неточності, що спотворюють результати

навчання. Крім того, процес анотування даних, як ручний, так і автоматизований, може бути пов'язаний з помилками та суб'єктивністю, що веде до подальших складнощів у навчанні моделей. Розвиток інструментів для перевірки та забезпечення якості даних, а також методів для ефективного анотування є ключовими для вирішення цих проблем.

7. **Законодавчі обмеження**

Законодавчі обмеження, особливо у сфері захисту даних, такі як GDPR в Європі, впливають на збір та обробку даних в машинному навчанні. Це створює додаткові виклики у вирішенні питань конфіденційності, зберігання та передачі даних, особливо в міжнародних проектах. Вирішення цих викликів вимагає глибокого розуміння відповідних законів та розробки відповідних політик та процедур для їх дотримання.

Розвиток передових технологій та методів є ключовим для вирішення викликів, пов'язаних з великими обсягами, різноманітністю та складністю даних у машинному навчанні. Використання хмарних технологій, паралельних обчислень, а також розробка алгоритмів для ефективної обробки різноманітних даних є важливими кроками для оптимізації використання даних та забезпечення точності моделей машинного навчання. Вирішення цих викликів часто передбачає використання розподілених систем зберігання даних, що дозволяє масштабувати інфраструктуру під зростаючі потреби в зберіганні даних. Також важливою є оптимізація систем управління даними для забезпечення швидкого доступу та високої продуктивності їх обробки. Застосування методів машинного навчання для автоматизації процесів управління даними та розробка більш ефективних алгоритмів індексації

та пошуку можуть також сприяти оптимізації обробки великих обсягів даних.

1.2.3. Використання ресурсів під час тренування моделей

Тренування моделей машинного навчання є складним і ресурсоємним процесом, який включає використання значних обчислювальних ресурсів, пам'яті, часу та іншої інфраструктури. Ефективне управління цими ресурсами є критично важливим для забезпечення ефективності та масштабованості машинного навчання, особливо в умовах зростаючих обсягів даних та складності моделей.

1. **Використання обчислювальних ресурсів**

Тренування моделей машинного навчання, особливо глибокого навчання, вимагає значної обчислювальної потужності. Використання GPU (графічних процесорів) та TPU (тензорних процесорних одиниць) стало стандартом у цій сфері, оскільки вони пропонують значно кращу продуктивність для паралельних обчислень, необхідних для тренування нейронних мереж. Проте, висока вартість та енергоспоживання цих технологій вимагають уважного планування та оптимізації з метою мінімізації витрат та впливу на довкілля. Також висока вартість та обмежена доступність GPU можуть стати перешкодою, особливо для невеликих організацій або дослідницьких груп.

2. **Використання пам'яті та зберігання**

Тренування складних моделей машинного навчання також вимагає значних обсягів пам'яті. Великі набори даних та складні архітектури моделей можуть вимагати терабайтів пам'яті для ефективного тренування. Крім того, потреба в швидкому доступі до даних під час тренування вимагає використання високошвидкісних систем зберігання, таких як SSD, що також підвищує вартість. Тренування сучасних моделей вимагає значних обсягів оперативної та дискової пам'яті. Великі датасети та

складні архітектури моделей вимагають ефективних систем зберігання, що може підвищити вартість інфраструктури. Проблеми зі швидкістю доступу до даних також можуть знизити ефективність тренування.

3. Часові витрати

Час тренування моделей машинного навчання може варіюватися від кількох годин до декількох тижнів, в залежності від складності моделі та обсягу даних. В деяких випадках процес тренування може займати тижні або навіть місяці, що вимагає великих обчислювальних потужностей, постійного моніторингу, ефективного розподілу часу та ресурсів, особливо в дослідницьких та розробницьких відділах, де час є обмеженим ресурсом.

4. Оптимізація ресурсів

Для оптимізації використання ресурсів під час тренування моделей машинного навчання важливо застосовувати техніки, такі як ефективне планування завдань, використання хмарних обчислень для масштабування ресурсів вгору або вниз залежно від потреби, та застосування технік оптимізації моделей, щоб зменшити їх обчислювальні вимоги без втрати продуктивності. Також ефективне використання ресурсів можна досягти за допомогою оптимізації алгоритмів та шляхом розподілу завдань між різними обчислювальними вузлами. Оптимізація коду та вибір ефективних фреймворків також можуть сприяти більш ефективному використанню обчислювальних ресурсів. Попри це все ще невирішеною є проблема оцінки необхідної кількості ресурсів для тренування моделі, адже кожен експеремнт не є таким детермінованим як запит користувача у звичайному веб-додатку.

5. Проблеми простою ресурсів

Затримки у отриманні даних та неповна утилізація ресурсів можуть призводити до простою системи, втрати часу та енергії. Це може бути викликано недостатньою пропускною здатністю мережі, обмеженнями системи зберігання або неоптимальними алгоритмами тренування. Рішення полягає у балансуванні між обчислювальними потужностями і доступністю даних, а також у використанні технологій для забезпечення більш рівномірного розподілу навантаження. Крім того під час тренування моделей доволі часто обчислювальні машини мають великий CPU ресурс, що майже не використовується, адже всі обчислення проводяться на GPU. Цієї проблеми ми також торкнемося далі у цій роботі.

Управління ресурсами під час тренування моделей машинного навчання є складним завданням, яке вимагає балансу між доступними обчислювальними потужностями, вартістю, часом та ефективністю. Оптимізація використання цих ресурсів не тільки підвищує ефективність тренування, але й знижує загальні витрати, сприяючи більш швидкому та ефективному розвитку моделей машинного навчання.

1.3. Засоби оцінювання та моніторингу

Засоби оцінювання та моніторингу є критично важливими для ефективного та відповідального використання ресурсів у машинному навчанні. Вони виконують важливу роль у виявленні та вирішенні проблем, пов'язаних з продуктивністю, використанням ресурсів, ефективністю та надійністю систем машинного навчання. Також правильні засоби спостереження дозволяють виявити потенційні вузькі місця, оцінити якість тренування моделей, виявити помилки та недоліки у алгоритмах, попереджувати проблеми пов’язані з якістю даних, допомагають у виявленні аномалій, а також з оцінкою вуглецевого сліду.

Інструменти та техніки моніторингу

Візуалізація даних і метрик: Використання інструментів візуалізації, таких як TensorBoard або Matplotlib, для відображення ключових метрик тренування, включаючи втрати та точність моделі. Це допомагає виявити проблеми, такі як перенавчання або недонавчання.

Логування та звітність: Збір та аналіз логів може надати інформацію про продуктивність та ефективність системи. Автоматизовані звіти допомагають відстежувати прогрес та виявляти аномалії у процесі тренування.

Методи оцінки якості даних

Аналіз якості даних: Використання статистичних методів та інструментів аналітики для оцінки повноти, точності, консистентності та актуальності даних. Це включає перевірку на наявність відсутніх значень, виявлення викидів та аналіз розподілів даних. Також використовують попередньо натреновані моделі машинного навчання для виявлення специфічних вад якості, наприклад для тренування моделей синтезу голосу вхідні дані перевіряють на якість транскрипції, наявність перекриваючих голосів , наявність фонового шуму, тощо.

Перевірка даних: Використання автоматизованих інструментів для перевірки даних на предмет помилок або неправильних форматів. Важливим є також аудит джерел даних для забезпечення їх надійності та відповідності.

Засоби спостереження за використанням ресурсів

Моніторинг ресурсів: Використання інструментів моніторингу, таких як Prometheus, Grafana, для відстеження використання CPU, GPU,

пам'яті та мережі. Це дозволяє ідентифікувати потенційні вузькі місця та оптимізувати використання ресурсів.

Аналіз логів: Збір та аналіз логів для виявлення ненормального використання ресурсів. Логи можуть вказувати на неефективні процеси або помилки у програмному забезпеченні.

Методи оцінки ефективності використання ресурсів

Бенчмаркінг: Використання стандартизованих тестів для оцінки продуктивності алгоритмів та обчислювальних систем. Це допомагає порівнювати різні підходи та технології з точки зору їх ефективності.

Аналіз вартості-ефективності: Оцінка вартості використання обчислювальних ресурсів порівняно з отриманими результатами. Це включає аналіз вартості обладнання, електроенергії, та інших витрат, пов'язаних із тренуванням та використанням моделей.

Аналіз енергоспоживання: Оцінка загального споживання енергії обчислювальними системами та інфраструктурою, включаючи тренування та використання моделей. Це дозволяє визначити вуглецевий слід та ідентифікувати можливості для його зниження. Застосування підходів та технологій, орієнтованих на зниження енергоспоживання та вуглецевих викидів є необхідним у сучасному світі, адже питання екології стоїть як ніколи гостро. Це включає в себе оптимізацію алгоритмів, вибір більш ефективного обладнання та використання відновлюваних джерел енергії.

Засоби виявлення помилок та аномалій

Системи виявлення аномалій: Використання алгоритмів машинного навчання для виявлення аномальної поведінки або результатів у даних. Це може включати ненормальні показники продуктивності, відхилення у якості даних або несподівані результати моделей.

Аналіз помилок: Ретельний аналіз помилок та відхилень, що виявлені під час тренування або використання моделей, для визначення їх причин та можливих рішень.

Оцінювання моделей та алгоритмів

Валідація та тестування моделей: Використання наборів даних для валідації та тестування для оцінки ефективності та точності моделей. Це включає в себе аналіз помилок, конфузійних матриць та інших ключових метрик.

Аналіз вузьких місць: Ідентифікація та аналіз вузьких місць у тренуванні та використанні моделей, що може включати неефективні алгоритми, недостатність даних або неправильне конфігурування параметрів.

Проблеми та виклики

Неповні або невірні дані: Необхідність виявлення та корекції неповних, невірних або спотворених даних, що можуть впливати на точність моделей.

Складність інтеграції інструментів: Інтеграція різних інструментів та систем моніторингу може бути складною, особливо у великих та складних системах.

Оптимізація та поліпшення

Автоматизація моніторингу та оцінювання: Розвиток автоматизованих систем для постійного моніторингу та оцінювання, що може забезпечити більш ефективне виявлення та усунення проблем.

Вдосконалення алгоритмів і процесів: Постійне оновлення та вдосконалення алгоритмів і процесів тренування з метою оптимізації використання ресурсів та підвищення ефективності моделей.

Оцінювання та моніторинг у машинному навчанні є необхідними для забезпечення якості, ефективності та продуктивності моделей. Використання сучасних інструментів та технік, постійне оновлення процесів та вдосконалення алгоритмів є ключовими для досягнення цих цілей. Засоби оцінювання та моніторингу в машинному навчанні є невід'ємною частиною забезпечення ефективного, економічного та екологічно відповідального використання ресурсів. Вони допомагають не тільки у виявленні та вирішенні поточних проблем, але й у плануванні майбутнього розвитку та оптимізації систем машинного навчання.

1.4. Існуючі рішення для оптимізації використання обчислювальних ресурсів

У сучасному світі, де машинне навчання (ML) набуває все більшої ролі в різних сферах від бізнес-аналітики до розробки програмного забезпечення, оптимізація використання обчислювальних ресурсів стає ключовим фактором для ефективності та успішності ML проектів. Розділ 1.4. зосереджується на аналізі різних стратегій та інструментів, які можуть бути використані для підвищення ефективності обчислень у сфері машинного навчання.

Основні виклики

- **Масштабування даних та обчислень**: З ростом обсягів даних та складності моделей машинного навчання, постає виклик ефективного масштабування обчислювальних ресурсів.
- **Оптимізація використання ресурсів**: Важливо оптимізувати використання обчислювальної потужності, пам'яті та інших ресурсів, щоб забезпечити максимальну продуктивність при мінімальних витратах.

- **Автоматизація та покращення робочих процесів**: Інтеграція автоматизованих інструментів для покращення робочих процесів машинного навчання та забезпечення швидкої та ефективної ітерації.

У цьому розділі ми розглянемо наступні ключові аспекти:

1. **Апаратні рішення**: Аналіз апаратних платформ та технологій, які сприяють оптимізації обчислень у машинному навчанні.
2. **Мережеві рішення**: Розгляд мережевих технологій та інфраструктур, які забезпечують ефективне використання ресурсів.
3. **Програмні рішення та хмарні сервіси**: Оцінка різних програмних інструментів, платформ та хмарних сервісів, що підтримують оптимізацію процесів машинного навчання.

Мета цього розділу - надати всебічний погляд на доступні технології та методи, що дозволяють максимізувати ефективність машинного навчання. Важливість розділу полягає у визначенні та розумінні найкращих практик та інструментів, які допоможуть дослідникам та розробникам у досягненні високої продуктивності в їхніх ML проектах.

1.4.1. Апаратні рішення

Існують різні апаратні рішення від провідних гравців ринку, зокрема рішення, які пропонують провайдери хмарних технологій, такі як AWS, Azure та GCP, а також на новітні апаратні рішення від NVidia та інших провідних виробників чіпів. Ці рішення є ключовими для оптимізації використання обчислювальних ресурсів у машинному навчанні та штучному інтелекті.

Апаратні рішення від провайдерів хмарних технологій

1. **Amazon Web Services (AWS)**

EC2 Instances: AWS пропонує різні типи EC2 інстансів, оптимізованих для обчислень, включаючи інстанси з GPU, оптимізовані для тренування моделей машинного навчання:

- **Amazon EC2 Trn1 instances**: Спеціалізовані для тренування глибоких нейронних мереж, пропонують найкращу цінову ефективність.
- **Amazon EC2 DL1 instances**: Працюють на Gaudi accelerators від Habana Labs, забезпечують високу продуктивність для тренування моделей.
- **Amazon EC2 P4 та P3 instances**: Підтримують швидке тренування моделей з високою продуктивністю завдяки NVIDIA GPU.
- **Amazon EC2 G5 та G4 instances**: Відповідають за машинне навчання з нижчими витратами, ідеальні для менших проектів, або для інфернсу моделей.

AWS Inferentia: Спеціалізоване апаратне рішення для оптимізації процесів інференсу, здатне знижувати витрати та покращувати продуктивність.

2. **Microsoft Azure**

Microsoft Azure пропонує різноманітні варіанти GPU та CPU, оптимізовані для машинного навчання, включаючи серії N та H, які забезпечують високу продуктивність для тренування та розгортання моделей.

Azure Machine Learning: Платформа, яка інтегрує велику кількість апаратних ресурсів, включаючи GPU та FPGA, для надання гнучкості у виборі обладнання для конкретних потреб машинного навчання.

Azure N-Series: Віртуальні машини, які включають в себе потужні GPU, оптимізовані для обчислень та графічних задач.

3. **Google Cloud Platform (GCP)**

GCP використовує власні TPU, які є спеціалізованими для тренування та виконання моделей машинного навчання, забезпечуючи високу швидкість обробки при меншому енергоспоживанні порівняно з традиційними GPU.

Compute Engine: GCP пропонує віртуальні машини з високопродуктивними GPU, що підходять для складних завдань машинного навчання.

Cloud TPU: Тензорні процесорні одиниці від Google, спеціально розроблені для прискорення процесів тренування та інференсу в машинному навчанні.

Новітні апаратні рішення від NVidia та інших виробників чіпів

NVidia продовжує бути лідером у виробництві GPU для машинного навчання, з їх останніми інноваціями, такими як A100 Tensor Core GPU, які пропонують безпрецедентну продуктивність. Інші виробники, такі як Intel з їх Habana Gaudi accelerators, також вносять важливий внесок у розвиток апаратних рішень для штучного інтелекту.

1. **NVidia**

Tesla V100 та T4 GPU: Спеціалізовані графічні процесори, розроблені для тренування та інференсу в машинному навчанні, з високою пропускною спроможністю та оптимізацією для глибокого навчання.

NVidia DGX Systems: Ці системи включають в себе набір інтегрованих GPU для високопродуктивних обчислень у машинному навчанні та штучному інтелекті.

2. **Інші виробники чіпів**

Intel Xeon Processors: Використовуються у багатьох серверних рішеннях, ці процесори підходять для різноманітних задач машинного навчання.

AMD Radeon Instinct: GPU, оптимізовані для високопродуктивних обчислень та машинного навчання, які пропонують конкурентоспроможні альтернативи рішенням NVidia.

Апаратні рішення, які пропонують провайдери хмарних технологій та виробники чіпів для машинного навчання постійно розвиваються, пропонуючи різноманітні опції для різних потреб і бюджетів. Вони відіграють критичну роль у вирішенні викликів, пов'язаних з операціями штучного інтелекту. Вибір між CPU, GPU, TPU та іншими спеціалізованими пристроями залежить від конкретних вимог проекту, та може значно підвищити продуктивність, оптимізувати використання ресурсів та знизити загальні витрати.

1.4.2. Рішення на мережевому рівні

Проведемо аналіз рішень на мережевому рівні, які спрямовані на оптимізацію використання обчислювальних ресурсів у машинному навчанні. В умовах зростаючих обсягів даних та потреби в швидкому обміні інформацією, ефективне управління мережевими ресурсами стає важливим аспектом для підвищення продуктивності та ефективності операцій машинного навчання. Ці рішення включають інноваційні технології та продукти, які сприяють оптимізації мережевого трафіку, підвищують безпеку, забезпечують ефективність обчислень та знижують затримки в обробці даних. Існує багато мережевих рішень, доступних на ринку, які допомагають вирішувати поставлені проблеми. Розглянемо декілька з них.

Широкосмугові мережі

Забезпечення високошвидкісних мережевих з'єднань є важливим для забезпечення ефективного обміну даними між серверами та обчислювальними вузлами. Це особливо актуально в розподілених системах та хмарних обчисленнях.

Компанії, такі як Cisco та Juniper Networks, пропонують рішення для побудови високопродуктивних мережевих інфраструктур з широкосмуговим доступом. Ці рішення допомагають мінімізувати затримки у передачі даних, особливо критичні для розподілених обчислень та хмарних технологій.

Оптимізація мережевого трафіку

Використання технологій для оптимізації мережевого трафіку, таких як протоколи компресії даних, може зменшити затримки та поліпшити пропускну спроможність мережі.

Продукти від компаній, таких як F5 Networks та Citrix, пропонують рішення для оптимізації мережевого трафіку та балансування навантаження. Ці технології забезпечують раціональне використання мережевих ресурсів, запобігаючи перевантаженню мережі та підвищуючи загальну ефективність системи.

Обчислення на краю мережі (Edge Computing)

Розміщення обчислювальних ресурсів ближче до джерела даних (наприклад, IoT пристроїв) може знизити затримки та зменшити навантаження на центральні мережеві системи.

Компанії як Akamai та Cloudflare розробляють рішення Edge Computing, які дозволяють розміщувати обчислювальні ресурси ближче до кінцевих користувачів. Ці рішення знижують затримки, покращують швидкість обробки даних та розвантажують центральні обчислювальні системи.

Мережевий балансувальник навантаження

Використання мережевих балансувальників навантаження дозволяє рівномірно розподіляти запити між серверами, оптимізуючи використання ресурсів та запобігаючи перевантаженню окремих вузлів.

Технології від компаній, таких як VMware та Barracuda Networks, пропонують передові рішення для балансування навантаження в мережі. Ці системи допомагають рівномірно розподіляти трафік між серверами, підвищуючи надійність та доступність ресурсів.

Засоби мережевої безпеки

Шифрування даних забезпечує безпеку даних під час їх передачі через мережу є важливим аспектом, особливо при роботі з конфіденційною інформацією, в той час, як розробка стратегій для захисту від DDoS-атак забезпечує стабільність та доступність мережевих ресурсів.

Рішення від компаній, таких як Palo Alto Networks та Fortinet, надають високий рівень захисту мережевих ресурсів. Ці продукти забезпечують захист від DDoS-атак, шифрування даних та інші функції безпеки для забезпечення безпечного обміну даними.

Оптимізація обчислень у хмарі

Інтеграція з хмарними обчисленнями дозволяє масштабувати обчислювальні ресурси відповідно до потреб, забезпечуючи гнучкість та ефективність у використанні обчислювальних потужностей. Автоматизація процесів розгортання та управління мережевими ресурсами може значно покращити продуктивність та зменшити ручні витрати роботи.

Хмарні провайдери, як Amazon Web Services, Microsoft Azure та Google Cloud Platform, пропонують гнучкі хмарні рішення для динамічного масштабування обчислювальних ресурсів. Ці рішення

дозволяють швидко масштабувати ресурси вгору або вниз залежно від потреб, що забезпечує оптимальне використання обчислювальних потужностей і ефективність витрат.

Рішення на мережевому рівні відіграють вирішальну роль у підвищенні ефективності використання обчислювальних ресурсів у машинному навчанні. Від оптимізації мережевої інфраструктури та балансування навантаження до інтеграції з хмарними обчисленнями та засобів мережевої безпеки, ці рішення забезпечують необхідну підтримку для складних обчислювальних процесів машинного навчання та ефективної роботи сучасних технологій, знижуючи затримки, підвищуючи продуктивність та забезпечуючи безпеку даних.

1.4.3. Програмні рішення та хмарні сервіси

Проведемо детальний огляд програмних рішень та хмарних сервісів, спрямованих на підвищення ефективності операцій машинного навчання. Включаючи фреймворки, інструменти управління даними, автоматизацію, моніторинг та хмарні платформи, ці рішення охоплюють весь спектр потреб у машинному навчанні та включають різноманітні інструменти та платформи, що забезпечують ефективність, швидкість та масштабованість обчислень, а також допомагають керувати великими наборами даних і складними алгоритмами машинного навчання. Ми не будемо детально описувати всі можливі альтернативи, але роздивимось провідних гравців на ринку.

Фреймворки машинного навчання

Високорівневі фреймворки спрощують розробку та тренування моделей машинного навчання. Ці фреймворки забезпечують інструменти для оптимізації алгоритмів, автоматизації процесів тренування та ефективного використання обчислювальних ресурсів.

TensorFlow та **Keras**. TensorFlow – це відкритий фреймворк від Google для машинного навчання оптимальний для тренування та розгортання моделей, з підтримкою масштабованих обчислень, а Keras – це високорівневий API, який може використовуватися з TensorFlow. Keras слугує інтерфейсом для TensorFlow, який забезпечує спрощене створення моделей. У поєднанні вони забезпечують широкі можливості для моделювання, візуалізації та деплойменту моделей машинного навчання.

PyTorch - це це популярний відкритий фреймворк від Facebook, відмінний вибір для досліджень та розробки завдяки гнучкості та динамічній природі. До переваг перш за все відносять легкість використання, високу гнучкість та підтримку динамічних нейронних мереж.

Інструменти для управління даними

Системи як Hadoop та Spark дозволяють ефективно обробляти великі обсяги даних, забезпечуючи швидкий доступ та обробку. Ці інструменти допомагають вирішити проблеми обробки великих датасетів, забезпечуючи необхідну швидкість та масштабованість для складних алгоритмів машинного навчання.

Apache Spark є ефективним рішення для обробки великих датасетів та виконання складних обчислень. Spark ефективно виконує розподілені обчислення великих обсягів даних, що є важливим для тренування та впровадження моделей машинного навчання. До переваг можна віднести масштабованість, велику екосистему, швидкість обробки даних, підтримка різних джерел даних та інтеграція з машинним навчанням. Також Spark чудово інтегрований з хмарними рішеннями.

Apache Hudi - фреймворк, що дозволяє ефективно управляти великими датасетами на Hadoop. Він забезпечує швидкісні операції на

великих обсягах даних, що сприяє кращому управлінню даними для машинного навчання.

ETL (Extract, Transform, Load) - це набір сервісів, що дозволяють ефективно екстрагувати дані з різних джерел, трансформувати їх відповідно до потреби та завантажувати у цільові системи. ETL сервіси спрощують інтеграцію, обробку та аналіз великих обсягів даних, що є критично важливим для тренування та використання моделей машинного навчання.

Pandas, NumPy - бібліотеки Python для обробки та аналізу даних, що надають потужні інструменти для маніпулювання даними. На даний момент є стандартом індустрії через інтуїтивний синтаксис та широкі можливості.

Автоматизація та оркестрація

Рішення як Jenkins та Kubernetes дозволяють автоматизувати розгортання, масштабування та управління контейнеризованими застосунками.

Kubernetes, Docker - системи для контейнеризації та оркестрації додатків, що забезпечують гнучке управління ресурсами та легке розгортання. Вони відкривають можливості гнучкого масштабування, управління ресурсами та ізоляція середовищ. Переваги використання таких систем полягають у полегшення процесу розгортання, масштабування та управління додатками машинного навчання.

Jenkins - автоматизація процесів CI/CD.

Моніторинг та оптимізація

Використання інструментів моніторингу, як Prometheus та Grafana, дозволяє відстежувати продуктивність та стан системи, швидко виявляючи та усуваючи проблеми.

Prometheus, Grafana - Інструменти для моніторингу стану системи та візуалізації даних, що допомагають виявити та усунути проблеми в обчислювальних процесах.

Оптимізація програмного коду

Практики ефективного кодування, а саме Оптимізація алгоритмів та використання ефективних структур даних допомагають знизити витрати на обчислювальні ресурси. Ефективний програмний код забезпечує більш швидке виконання програм і менше використання пам'яті, що є критично важливим для обчислювально вимогливих задач машинного навчання.

Хмарні сервіси

Використання хмарних платформ, як AWS, Azure, GCP для машинного навчання, дозволяє гнучко масштабувати ресурси вгору або вниз залежно від потреб. Хмарні рішення надають необмежені обчислювальні ресурси та високу доступність, що дозволяє обробляти великі обсяги даних та комплексні моделі машинного навчання. Розглянемо кілька провідних рішень від AWS:

AWS Elastic MapReduce - платформа для обробки великих датасетів, що використовує Hadoop, Spark та інші популярні фреймворки. EMR забезпечує ефективне масштабування для обробки великих даних, важливе для задач аналітики та машинного навчання.

AWS AuroraDB - високопродуктивна реляційна база даних, сумісна з MySQL та PostgreSQL. AuroraDB пропонує масштабованість, високу продуктивність та надійність для управління даними, які використовуються в машинному навчанні.

AWS Glue - Сервіс ETL, який дозволяє легко підготувати та переміщати дані між різними джерелами та цільовими системами. AWS Glue спрощує обробку та інтеграцію даних, що є ключовим для ефективного використання даних у машинному навчанні.

AutoML

AutoML (Automated Machine Learning) - це технологія, що автоматизує процеси створення, тренування та оптимізації моделей машинного навчання. Ці рішення призначені для того, щоб зробити машинне навчання доступнішим, швидшим та ефективнішим. Ось детальний огляд декількох провідних AutoML рішень на ринку.

Google AutoML надає набір інструментів та інтерфейсів, які дозволяють користувачам з обмеженим досвідом у машинному навчанні створювати високоефективні моделі. Має широкий спектр застосування від розпізнавання образів до аналізу емоцій у тексті. Серед особливостей:

- Підтримка різних видів даних: текст, зображення, відео.
- Інтеграція з іншими сервісами Google Cloud.

Microsoft Azure AutoML - частина платформи Azure Machine Learning, пропонує автоматизацію процесу створення моделей машинного навчання. Відрізняється широким спектр алгоритмів і метрик оцінки, а також вбудованими засобами для візуалізації та аналізу моделей. Чудово підходить для автоматизованого відбіру функцій та оптимізація гіперпараметрів.

AWS AutoML - частина сервісів Amazon SageMaker, спрощує процес створення моделей машинного навчання. Також надає широкий спектр застосувань, від прогнозування до оптимізації. Особливості:

- Інтуїтивно зрозумілий інтерфейс для автоматизації процесу створення моделей.
- Інтеграція з іншими AWS сервісами.

H2O AutoML - відкрите програмне забезпечення, що надає широкі можливості для автоматизації машинного навчання. Підходить для різноманітних задач, від класифікації до регресії. Особливості:

- Автоматичний відбір моделей, крос-валідація.
- Легко інтегрується з іншими платформами та мовами програмування.

DataRobot - комерційна платформа AutoML, яка забезпечує швидке створення та розгортання моделей. Використовується в фінансах, охороні здоров'я, ритейлі. Особливості:

- Великий вибір предобраних моделей.
- Глибокий аналіз моделей та автоматизація весь процесу машинного навчання.

AutoML рішення забезпечують значні переваги у швидкості та ефективності розробки моделей машинного навчання, роблячи машинне навчання доступнішим для широкого спектру користувачів. Від великих корпорацій до незалежних розробників, ці інструменти відкривають нові можливості для інновацій та досліджень у сфері машинного навчання.

Автоматизовані платформи машинного навчання

AWS SageMaker

Amazon SageMaker - це повністю керована служба, яка дозволяє науковцям і розробникам швидко та легко створювати, тренувати та розгортати моделі машинного навчання. Особливості:

- Інтегроване середовище Jupyter Notebook.
- Автоматизоване масштабування для тренування та розгортання.
- Вбудовані алгоритми та підтримка приведення власних алгоритмів.

Переваги: Простота використання, масштабованість, висока інтеграція з іншими сервісами AWS.

Azure Machine Learning

Сервіс від Microsoft, що надає інструменти та середовища для розробки, тренування та розгортання моделей машинного навчання. Особливості:

- Візуальний інтерфейс для будування моделей.
- Підтримка AutoML для автоматичного відбору та оптимізації моделей.
- Інтеграція з іншими продуктами Microsoft та Azure Cloud Services.

Переваги: Легкість використання, гнучкість, потужні можливості AutoML.

Google AI Platform

Платформа від Google, що надає інтегровані сервіси для створення, аналізу та розгортання моделей машинного навчання. Особливості:

- Підтримка TensorFlow, PyTorch та інших популярних фреймворків.
- Інструменти для автоматизації та оптимізації моделей.
- Інтеграція з Google Cloud Storage та іншими Google Cloud Services.

Переваги: Сильна інтеграція з екосистемою Google, підтримка передових технологій та алгоритмів.

ClearML

Відкрите програмне забезпечення для управління та автоматизації процесів машинного навчання. Особливості:

- Автоматизація робочих процесів, трекінг експериментів.
- Версіонування даних та моделей, моніторинг продуктивності.
- Підтримка багатьох фреймворків та мов програмування.

Переваги: Гнучкість та відкритість, підтримка спільної роботи та співпраці.

Порівняння

Інтеграція з екосистемами:

- SageMaker: Сильна інтеграція з AWS.
- Azure ML: Інтеграція з Microsoft Azure.
- Google AI: Інтеграція з Google Cloud.
- ClearML: Незалежність від платформ, підтримка різних середовищ.

Легкість використання: SageMaker і Azure ML надають більш інтуїтивні візуальні інтерфейси, в той час як Google AI та ClearML зосереджені на гнучкості та кастомізації.

AutoML: Azure ML та Google AI мають сильні AutoML функції. SageMaker і ClearML також підтримують автоматизацію, але з меншим фокусом на AutoML.

Відкритість та гнучкість: ClearML як відкрите ПЗ виграє за гнучкістю та відкритістю. Інші платформи більше прив'язані до своїх хмарних сервісів.

Кожна з цих платформ має свої унікальні переваги та особливості, залежно від потреб користувача. AWS SageMaker, Azure Machine Learning та Google AI Platform забезпечують сильну інтеграцію з відповідними хмарними екосистемами, в той час як ClearML надає більшу гнучкість та відкритість, що може бути важливим для певних проектів та організацій.

Висновок

Сукупність цих програмних рішень та хмарних сервісів створює потужний фундамент для оптимізації операцій машинного навчання. Вони забезпечують необхідні інструменти для ефективної обробки даних, гнучкого тренування та розгортання моделей, а також управління ресурсами і масштабуванням, відкриваючи широкі можливості для інновацій та розвитку в області машинного навчання. Використання цих

рішень та хмарних сервісів дозволяє значно підвищити ефективність та продуктивність операцій машинного навчання.

1.5. Негативні наслідки відсутності оптимізації

Оптимізація в машинному навчанні відіграє критичну роль у забезпеченні ефективності, точності та економічної вигоди проектів. Відсутність оптимізації може призвести до значних негативних наслідків, які обмежують потенціал та вартість ML ініціатив.

1. **Зниження ефективності обчислень**

Надмірне використання ресурсів

Неоптимізовані моделі можуть потребувати непропорційно більшої кількості обчислювального часу та потужності. Наслідки: Збільшення операційних витрат та витрат на інфраструктуру.

Недостатнє використання ресурсів

Так само як надмірне використання ресурсів - неповна утилізація ресурсів створює простір для оптимізації, оскільки простой тих чи інших обчислювальних ресурсів змушує витрачати кошти без отримання користі.

Повільна швидкість обробки

Відсутність оптимізації алгоритмів може уповільнити обробку даних, затримуючи отримання результатів. Наслідки: Затримки у прийнятті рішень та зниження продуктивності проекту.

2. **Економічні наслідки**

Збільшення оперативних витрат

Неефективне використання ресурсів веде до зайвих витрат на хмарні сервіси, сервери та інше обладнання. Наслідки: Підвищення

загальних витрат проекту, що може бути особливо критично для стартапів та малих підприємств.

Надмірне використання енергії

Неефективність в обчисленнях збільшує споживання електроенергії. Наслідки: Збільшення екологічного відбитку та витрат на енергію.

3. **Проблеми з якістю та надійністю**

Погіршення точності та надійності Моделей

Неоптимізовані моделі можуть виробляти неточні або ненадійні результати. Наслідки: Неправильні прогнози та аналітика можуть призвести до невдалого прийняття рішень.

Труднощі з масштабуванням та гнучкістю

Системи без оптимізації часто виявляються негнучкими для масштабування або адаптації до нових вимог. Наслідки: Обмеження потенціалу зростання та адаптації до змінюванихумов ринку.

4. **Технологічні та операційні виклики**

Складнощі в інтеграції та управлінні

Неоптимізовані системи можуть бути складнішими для інтеграції з іншими технологіями та управління. Неоптимізовані рішення можуть бути менш сумісними з новітніми технологіями або стандартами. Наслідки: Збільшення часу та зусиль на технічну підтримку та управління системами.

Відсутність гнучкості у виборі технологій

Обмежена можливість адаптувати технології під специфічні потреби проекту. Наслідки: Обмеження у виборі оптимальних інструментів та технологій для конкретних задач.

Відсутність оптимізації в машинному навчанні призводить до ряду серйозних негативних наслідків, що впливають на ефективність, вартість, якість та гнучкість проектів. Підкреслюючи важливість оптимізаційних стратегій, цей розділ підкреслює необхідність розуміння та впровадження ефективних рішень для управління та оптимізації процесів машинного навчання.

1.6. Висновки до розділу

Ретельно підсумуємо ключові висновки та уроки, отримані з аналізу викликів і проблем, пов'язаних з використанням ресурсів у операціях машинного навчання. Мета цього розділу - забезпечити узагальнене бачення нашого дослідження, вказуючи на стратегічне значення оптимізації для ефективного використання технологій машинного навчання.

Ми зосередимось на викликах у використанні ресурсів у машинному навчанні, з акцентом на невирішених проблемах, які потребують подальших досліджень та інноваційних підходів.

Ключові висновки

- **Значення оптимізації обчислювальних ресурсів** - оптимізація обчислювальних ресурсів є фундаментальною для забезпечення ефективності та продуктивності машинного навчання. Недооцінка цього аспекту може призвести до збільшення витрат, зниження продуктивності та погіршення якості моделей.
- **Вплив неоптимізованих ресурсів** - неоптимізоване використання ресурсів може призвести до значних затримок у процесах машинного навчання, збільшуючи час обробки даних та уповільнюючи впровадження моделей.

- **Інноваційні рішення та практики** - впровадження інноваційних технологій та методів, включаючи апаратні рішення, мережеві оптимізації та програмні інструменти, демонструє значний потенціал для підвищення ефективності ML операцій.
- **Стратегічне планування та управління** - стратегічне планування і компетентне управління обчислювальними ресурсами є критично важливими для досягнення високих результатів у машинному навчанні.

Невирішені проблеми

- **Масштабування для великих даних** - незважаючи на значний прогрес, проблема ефективного масштабування обробки великих обсягів даних залишається невирішеною, особливо у контексті складних датасетів.
- **Оптимізація енергоспоживання** - проблема оптимізації енергоспоживання в обчислювальних системах машинного навчання залишається відкритою, з огляду на зростаючі екологічні та економічні турботи.
- **Адаптація до змінних даних** - адаптація моделей та систем до швидко змінних датасетів є складною задачею, що вимагає більш гнучких та адаптивних підходів у обробці даних.
- **Інтеграція з різними технологічними стеками** - інтеграція оптимізаційних рішень з різноманітними технологічними стеками та платформами є складною, особливо у контексті швидкого розвитку технологій.

Рекомендації для подальших досліджень

Розвиток алгоритмів для великих Даних

Рекомендується розвиток нових алгоритмів та методологій для ефективної обробки великих обсягів даних. Це включає розробку технік для розподілених обчислень та паралельної обробки даних, що дозволять більш ефективно використовувати обчислювальні ресурси.

Дослідження енергоефективних технологій

Потребується подальше дослідження в області енергоефективних технологій, що дозволять знизити споживання енергії обчислювальними системами, особливо в контексті великих дата-центрів та хмарних обчислень.

Розробка адаптивних моделей

Значну увагу слід приділити розробці адаптивних моделей, які можуть швидко реагувати на зміни в даних та забезпечувати стабільність та точність у динамічних умовах.

Універсальність та інтеграція рішень

Важливим напрямком досліджень є створення універсальних рішень, які можуть бути інтегровані з різними технологічними стеками та платформами, забезпечуючи більшу гнучкість та широкий спектр застосування.

Підкреслимо, що незважаючи на значний прогрес у сфері оптимізації використання ресурсів у машинному навчанні, існують численні невирішені виклики та проблеми, які потребують подальших досліджень та інноваційних рішень. Вирішення цих проблем вимагає глибокого розуміння як технічних аспектів, так і складних динамік даних та їх обробки. Ключем до успіху в цій області є поєднання теоретичних

знань з практичним застосуванням, що дозволить розробити ефективні та економічно вигідні рішення для майбутнього машинного навчання.

2. Засоби оптимізації

2.1. Стратегія оптимізації управління великими даними в операціях машинного навчання

У сучасному світі даних, де обсяги інформації ростуть з кожним днем, ефективне управління великими даними стає критичним аспектом успіху в машинному навчанні (ML). Сфокусуємось на стратегіях оптимізації управління великими даними, щоб максимально підвищити ефективність та точність ML операцій.

Розглянемо засоби для оптимізації управління великими даними в операціях машинного навчання. Стратегія базується на визначеній структурі зберігання даних, запровадженні засобів версіонування файлів та інструментів трансформацій та управлінні метаданими. Основною метою запропонованого рішення є забезпечення прозорості використання даних та відтворюваності експериментів машинного навчання уникаючи створення додаткових затримок, необхідності дублювання даних та виконання повторних операцій.

Важливість оптимізації

- **Обробка та аналітика великих даних**: Ефективна обробка та аналітика великих даних є ключовою для генерації точних та корисних інсайтів з даних.
- **Виклики з масштабованістю та зберіганням**: Масштабування систем під зростаючі обсяги даних та оптимізація сховищ даних є серйозними викликами, які потребують вдумливого підходу.

Для пошуку ефективного рішення проблеми необхідно провести наступні дії:

- **Аналіз поточних викликів**: Розгляд конкретних проблем, з якими стикаються організації при роботі з великими даними у контексті ML.
- **Розробка стратегій оптимізації**: Обговорення підходів та методів, які можуть бути застосовані для підвищення ефективності обробки та аналізу великих даних.
- **Інтеграція з машинним навчанням**: Вивчення того, як стратегії управління великими даними можуть бути інтегровані з ML процесами для покращення загальної продуктивності.
- **Проведення експерименту**: Проаналізувати розроблену стратегію на відповідність вимогами, зробити заміри використання ресурсів різними сопособами та підсумувати отримані результати в порівнянні з існуючими рішеннями.

Цей розділ має на меті надати читачам глибоке розуміння важливості оптимізації управління великими даними в машинному навчанні. З акцентом на практичних стратегіях та реальних прикладах, надане дослідження покликане допомогти фахівцям у виборі та впровадженні найефективніших підходів для своїх проектів.

Постановка задачі

Існують різні способи та рішення для роботи з великими даними. Загальною потребою є можливість навчати моделі машинного навчання та вирішувати аналітичні задачі, будувати графіки та звіти. Однак навіть при початковому забезпеченні достатньої інфраструктури для зберігання і пошуку таких даних виникає ряд проблем і потреб, які змушують такі системи розвиватися.

Першою проблемою є можливість відтворення результатів пошуку даних у минулому, оскільки дані змінюються з часом, а версифікація

петабайтів інформації є нетривіальним завданням, оскільки це може суттєво вплинути на продуктивність та вартість інфраструктури.

Другою проблемою, що виникає, є необхідність попередньої обробки даних, що є значним витоком ресурсів на великих обсягах.

Третя проблема, що випливає з необхідності попередньої обробки: результати певних перетворень даних стають вхідними даними для інших перетворень, що створює ще одну потребу у створенні підваріантів даних з конкретних перетворень (наприклад, при зміні параметрів конкретного завдання на обробку).

Нарешті, через певний проміжок часу система може потребувати видалення деяких даних (здебільшого через зміну законодавчих вимог або на вимогу власника даних), або ж може виникнути необхідність доповнити набори даних новими об'єктами. Крім того, коли ми отримуємо дані, нам потрібно ефективно і швидко зрозуміти, які дані нам потрібні, а які ми можемо використовувати (знову ж таки, виходячи з вимог законодавства та дозволів володільця даних). І найголовніше - всі ці проблеми вимагають раціональних рішень, які дозволяють вирішувати їх за допомогою ресурсоефективних і продуктивних механізмів.

Вищезазначене зумовлює потребу у створенні механізмів версіонування, які б відповідали наступним функціональним вимогам:

- Отримання моментального знімка даних з певного моменту часу.
- Доступ до результатів перетворень, створених різними версіями одного кроку обробки даних.
- Видалення частини даних за запитом без порушення цілісності та узгодженості даних.
- Відтворення ланцюжка кроків, які були виконані для отримання результатів трансформації.
- Працювати з усіма типами даних: неструктурованими, структурованими та напівструктурованими.

А також наступні нефункціональні вимоги:

- Уникнення повторних операцій обробки одних і тих самих наборів даних для забезпечення оптимального використання обчислювальних ресурсів.
- Відсутність дублювання даних для запобігання неефективному використанню сховища.
- Не створювати додаткових затримок при отриманні даних.

Способи версіонування

Резервне копіювання

Для версійності даних найочевиднішим рішенням є збереження знімків даних (створення копій) у різні моменти часу. Цей підхід схожий на резервне копіювання бази даних, але має багато недоліків, оскільки ми можемо отримати дані лише зі збережених копій, а частота створення копій залежить від нас, тому проміжні зміни можуть бути нам недоступні. Крім того, лінійне зберігання копій даних збільшує розмір сховища, що можна обійти шляхом створення інкрементних копій, але для ефективної роботи з такими даними всі копії мають бути оптимізовані для читання, а відтворення знімка даних з великої кількості інкрементних копій сповільнює роботу. Частково цю проблему можна вирішити, комбінуючи два підходи, а саме періодичні повні резервні копії та проміжні інкрементні резервні копії. В цьому випадку ми можемо гарантувати певну швидкість незалежно від кількості періодів резервного копіювання, оскільки для створення знімка, в найгіршому випадку, нам завжди потрібно об'єднати лише N інкрементів з останньої повної резервної копії.

Для прискорення цього підходу можна розглянути можливість диференційованого копіювання, в цьому випадку кількість копій, які потрібно об'єднати, завжди буде дорівнювати двом, але загальний розмір

сховища для зберігання копій також буде значно більшим. Цей підхід також можна оптимізувати, періодично створюючи повні копії, а між повними копіями створювати диференціальні копії.

Підсумовуючи, можна сказати, що цей метод є хорошим рішенням у випадках, коли абсолютна детермінованість не є основною потребою, а обмеження на збереження даних не поширюються на резервні копії. Цей підхід найкраще підходить для версіонування необроблених даних або всього сховища для подальшого відновлення з резервних копій.

Створення копій при зміні файлів

Більшість сховищ Blob надають можливість версіонувати окремі файли. Це дозволяє отримати повний знімок даних за довільний проміжок часу, причому нові версії створюються лише тоді, коли файли змінюються, що є найкращим рішенням з точки зору використання сховища.

Цей метод демонструє свої переваги у випадках, коли ми переважно працюємо з останньою версією даних, а доступ до минулих версій може бути обмежений відновленням повного стану сховища на довільний момент часу. На відміну від попереднього методу, тут ми не обмежені періодами створення резервних копій і необхідністю виконувати злиття для читання певної версії даних.

Версіонування конвеєру обробки

Альтернативним підходом є версіонування окремих елементів конвеєра обробки, їхніх послідовностей і параметрів. Завдяки системам контролю версій, таким як GIT або SVN, ми можемо версіонувати наші інструменти обробки даних, але цей підхід вимагає обробки даних "на льоту" і може бути дуже дорогим для великих наборів даних. Таким чином, користувач може вибрати, які версії інструментів використовувати

за запитом, після чого трансформації відтворюються з визначених версій, і користувач може розпочати потокову обробку даних.

Такий підхід позбавляє нас можливості фільтрувати на основі витягнутих властивостей під час обробки, що унеможливлює часткову обробку на льоту для деяких випадків використання. Крім того, версіонування конвеєра обробки не покриває необхідність версіонування вихідних даних, що унеможливлює використання цього підходу без поєднання з іншими.

Підсумовуючи, можна сказати, що цей метод є необхідним компонентом, за допомогою якого можна відтворити ланцюжок кроків, які були зроблені для досягнення певної трансформації. Однак він не може бути використаний самостійно для задоволення наших вимог, оскільки вимагає занадто багато ресурсів для повторення кроків обробки даних для кожного запиту і не дозволяє версіонувати сирі дані, а лише трансформації.

Версіонування структурованих даних

Запропонований нами підхід до версіонування структурованих (табличних або документ-орієнтованих) даних полягає у створенні сховища, де кожен елемент (рядок таблиці або документ) представляє одну версію і має посилання на попередню версію, дату її створення та властивість, яка визначає, чи був файл видалений.

До цього сховища можна додавати лише нові елементи і заборонено вносити зміни до існуючих елементів на основі Append-only (за винятком властивості, яка визначає, чи був файл видалений).

Коли ми трансформуємо дані на основі попередніх даних, ми створюємо посилання у новостворених елементах на оригінальні елементи, з яких вони були створені.

Перевагою цього підходу є прозорість і простота, оскільки всі дані пов'язані з тим, на основі чого вони були створені. Слабкою стороною цього підходу є те, що він базується на структурованих даних, що значно обмежує сферу його самостійного використання і не відповідає нашим потребам у роботі з неструктурованими даними. Тим не менш, цей недолік не заважає нам використовувати цей метод у поєднанні з іншими, адже ми можемо створити структуроване сховище метаданих на основі цього методу і використовувати посилання на файли як атрибути.

Використання хеш-сум

Запропонований нами підхід до версійності файлів полягає в тому, щоб зберігати хеш-суми як імена файлів і зберігати всі дані в єдиному сховищі даних. При цьому дані повинні бути розділені на різні розділи відповідно до потреб домену. Це дозволить здійснювати автоматичну дедуплікацію, оскільки в одному наборі даних не може бути двох файлів з однаковою хеш-сумою.

Якщо різні версії різних файлів дають однаковий результат перетворення, то замість збереження двох однакових файлів ми збережемо один файл, на який можна посилатися довільну кількість разів. Це може бути особливо корисно, якщо ми обробляємо дані з різними версіями одного і того ж перетворення, оскільки досить часто незначні зміни в конкретному перетворенні можуть не мати суттєвого впливу на більшу частину даних.

Цей метод є оптимальним механізмом, який відповідає нашим нефункціональним вимогам. Його сила полягає в атомарності кожної версії як окремого файлу. Сам репозиторій абстрагований від версійності, оскільки інформація про версії знаходиться на рівні метаданих. Крім того, така структура зберігання дозволяє нам працювати з усіма версіями даних одночасно, без необхідності інтегрувати роботу з метаданими в конвеєр

обробки, адже ми можемо просто передати повний пакет даних, який вже містить всі версії. Таким чином, ми можемо об'єднати два попередні методи з цим методом і, при правильній структурі сховища метаданих і правильних розділах, також задовольнити всі наші функціональні вимоги.

Запропоноване рішення

Ми розробимо остаточне архітектурне рішення на основі комбінації запропонованих методів. Рішення буде включати в себе:

- Зберігання файлів з версіонуванням (використання хешу суми в якості імен файлів)
- Версіонування конвеєра обробки
- Створення сховища метаданих

Сховище файлів

Ми будемо використовувати сховище Blob і створювати розділи відповідно до потреб домену та оригінальних наборів даних, а також впровадимо багаторівневу (шарову) модель зберігання трансформацій.

Кожен оригінальний набір даних зберігається в окремому сховищі для забезпечення контролю доступу.

Ми розділили оригінальні набори всередині сховища на розділи відповідно до потреб предметної області для зручності навігації та підвищення ефективності маніпулювання даними, оскільки в цьому випадку ми можемо маніпулювати меншими підмножинами.

Модель багаторівневого сховища трансформації має на меті розділити сховище на партії, де кожна партія в межах одного розділу представляє один крок конвеєра трансформації. Це дозволяє легко розгалужувати конвеєр обробки на різні потоки, створюючи нові партії. Водночас, різні версії однієї і тієї ж трансформації зберігають дані в

межах одного пакету, що дозволяє уникнути дублювання версій однієї і тієї ж трансформації.

Версіонування конвеєра обробки

Конвеєр трансформації версифікується за допомогою системи контролю версій Git. Кожне окреме завдання обробки також має таку саму версію. Конвеєр відповідає лише за організацію окремих кроків та зберігання параметрів перетворення і не повинен містити специфічної для набору даних логіки (за винятком визначення порядку кроків, управління ресурсами та механізмів повторних спроб). Це дозволяє легко відтворити перетворення з будь-якого моменту часу. Ідентифікатори версій (commit sha) конвеєра і завдання обробки, яке створило конкретний файл, зберігатимуться в метаданих, що дозволить вам легко отримати кодовий знімок трансформації конкретного файлу.

Кожен елемент конвеєра обробки маніпулює файлами незалежно від метаданих і приймає пакет, що містить всі версії вхідних файлів певної підмножини. Такий підхід дозволяє виконувати трансформації не тільки над останньою версією даних, але й над усіма існуючими версіями та всіма існуючими гілками попередніх трансформацій.

Сховище метаданих

Сховище метаданих організовано в табличному форматі. Кожен набір даних має окрему таблицю, в якій зберігаються посилання на вихідні дані.

Для кожної трансформації створюється окрема таблиця, де кожен рядок містить посилання на файл, з якого було створено трансформацію (або таблицю з'єднань, якщо було використано кілька файлів), ідентифікатор версії трансформації та ідентифікатор версії конвеєра обробки даних. Після виконання певної трансформації до таблиці трансформацій додаються рядки зі словника вхідних даних трансформації

та її результатів. Такий підхід дозволяє відстежити всі трансформації конкретного файлу, а також використовувати рекурсивні запити для маніпуляцій з усіма версіями певних файлів.

Висновки

Розроблене архітектурне рішення дозволяє користувачеві ефективно отримувати будь-які дані зі сховища на основі метаданих, задовольняючи всі функціональні вимоги, визначені як проблемні.

Компоненти системи залишаються відносно незалежними, оскільки сховище метаданих відповідає за управління версіями, файлове сховище - лише за зберігання файлів, а конвеєр обробки працює виключно з розділами файлового сховища.

Отриманий інтерфейс може дозволити отримати довільну версію файлу. Це включає в себе отримання версії файлу з довільного моменту часу, яка була перетворена певним ланцюжком перетворень, без необхідності обробки на льоту, що виключає можливість повторного використання ресурсів для одних і тих же операцій. Рішення дозволяє дотримуватися різних законодавчих обмежень, пов'язаних зі зберіганням і використанням даних, дозволяючи видаляти файли і проводити аудит використання даних, в тому числі для швидкого визначення необхідності перенавчання тієї чи іншої моделі машинного навчання.

2.2. Спосіб оптимізації швидкості отримання напівструктурованих даних в операціях машинного навчання

У сфері машинного навчання, ефективне використання даних визначає успіх або провал алгоритмічних моделей. Особливу увагу привертають напівструктуровані дані, які, завдяки своїй гетерогенній

природі, представляють значні виклики для збору, обробки та аналізу. Розглянемо методології та техніки оптимізації, які сприяють підвищенню швидкості отримання та обробки напівструктурованих даних, враховуючи їх критичну роль у процесах машинного навчання.

Обговоримо рішення для оптимізації швидкодії доступу до даних в операціях машинного навчання. Рішення базується на організації напівструктурованих даних таким чином, щоб їх можна було ефективно розділити для потокової передачі в тренувальне середовище, щоб уникнути простоїв ресурсів під час навчання невеликих моделей і в той же час зберегти контроль над вмістом пакетів для експериментів.

Значення Оптимізації

Ефективне збирання та обробка напівструктурованих даних є не лише технічним завданням, але й фундаментальною необхідністю для забезпечення високої якості досліджень у галузі машинного навчання. Оптимізація цих процесів сприяє не тільки економії часу та ресурсів, але й забезпечує більш точне та глибоке розуміння вихідних даних.

Основні Аспекти Оптимізації

- **Технічні виклики** - детальний аналіз технічних перешкод у процесі збору та обробки напівструктурованих даних, включаючи розгляд форматів даних, інструментів обробки та алгоритмічних підходів.
- **Методологія оптимізації** - огляд можливих методологічних підходів та практик для оптимізації збору та обробки даних, акцентуючи на інноваційних алгоритмах та автоматизованих системах.
- **Інтеграція з процесами машинного навчання** - аналіз способів інтеграції оптимізованих процесів збору та обробки даних у загальні процеси машинного навчання, з метою підвищення ефективності та точності моделей ML.

Ми прагнемо надати науковцям, інженерам, аналітикам та іншим професіоналам, які працюють у сфері машинного навчання, комплексне розуміння стратегій оптимізації процесів збору та обробки напівструктурованих даних. Кінцева мета полягає у підвищенні ефективності використання даних та покращенні якості ML моделей, розвитку галузевих стандартів та підтримці інноваційного розвитку в цій області.

Постановка задачі

Досить часто вчені придумують невеликі моделі і хочуть тренувати їх на великих обсягах даних. Маленька модель не означає погана. Невелика модель потребує менше часу на навчання та виведення висновків і не обов'язково погіршує точність. Однією з невирішених проблем MLOps є навчання швидких моделей на великих обсягах даних. Проблема проявляється тоді, коли час на отримання пакету даних перевищує час на навчання моделі на цьому пакеті. Це призводить до простою дорогого обладнання (GPU). Такі обсяги даних зазвичай виникають, коли ми працюємо з неструктурованим або напівструктурованим контентом (аудіо, відео, зображення тощо).

Ми прагнемо порівняти існуючі підходи до отримання даних для навчальних середовищ і запропонувати ефективні рішення, які можуть задовольнити потреби у продуктивності навіть найшвидших моделей, що використовують розумні інфраструктурні ресурси.

Засоби отримання даних

Пряма вибірка з BLOB сховища.

Найочевидніший варіант, коли ми хочемо зберігати великі обсяги даних і мати до них доступ, - це вибірка даних безпосередньо зі сховища Blob.

Якщо ми працюємо з неструктурованими даними, ми повинні мати принаймні посилання на джерела даних. У більшості випадків для навчання ML ми виконуємо попередню обробку для створення напівструктурованих наборів даних. Коли ми зберігаємо напівструктуровані дані, ми повинні підтримувати зв'язки між метаданими та об'єктами.

Щоб отримати дані безпосередньо зі сховища Blob, ми повинні попередньо завантажити всі метадані в навчальне середовище, а потім ми можемо використовувати посилання для подальшого потокового завантаження даних зі сховища Blob. Проблема полягає в тому, що завантаження великої кількості маленьких двійкових об'єктів займає катастрофічно багато часу порівняно із завантаженням великих об'єктів. На жаль, у більшості випадків ми не можемо навчати моделі на великих об'єктах, у нас не вистачає пам'яті для створення збалансованих партій великих об'єктів.

Зберігання фрагментів даних у tar-архівах

Це оптимізована версія попереднього підходу. Рішення базується на завантаженні метаданих у навчальне середовище, а потім трансляції заархівованих фрагментів медіа зі сховища Blob. Недоліком є те, що нам доведеться витратити час на завантаження декількох архівів на основі посилань у метаданих. Це також призводить до простоїв, але принаймні

ми можемо створити кеш локально, щоб не завантажувати архіви повторно.

Використання мережевої файлової системи

Альтернативний підхід - створити Lustre (мережеву файлову систему) і підключити її до навчального середовища. Такий підхід дозволить нам отримати доступ до файлів зі швидкістю, порівнянною з локальним доступом до файлів. Однак вартість отримання даних з такої системи в іншій зоні доступності надзвичайно висока, що вимагає значних витрат на обслуговування. Як наслідок, необхідно реплікувати їх у різні зони доступності та постійно синхронізувати для забезпечення відмовостійкості.

Використання проміжного сховища

Ми пропонуємо наступний шлях: Використання проміжних середовищ на базі багатопроцесорних машин (без GPU) для створення навчального сховища. Вихідні дані зберігаються у сховищі BLOB-об'єктів. Ми перетворимо об'єкти з вихідного сховища в трубчасте представлення у вигляді parquet файлів, стиснутих у формат gzip. Тобто для кожного медіафайлу буде окремий рядок у таблиці, в якому зберігатиметься байтовий масив медіафайлу + його метадані. Оскільки метадані можуть змінюватися залежно від набору даних, для кожного напівструктурованого сховища вихідних даних ми створимо нове навчальне сховище. Кожен parquet файл матиме обмежену кількість рядків залежно від розміру навчальних партій. За допомогою архівації gzip ми зможемо ще більше зменшити обсяг трафіку та час завантаження файлів, а завдяки тому, що в байтових масивах різних медіафайлів буде велика кількість повторень, архівація буде досить ефективною. Оскільки графічний процесор не задіяний, запуск машин для перетворення об'єктів з вихідного сховища в навчальне буде досить дешевим та екологічно

чистим. Далі ми використаємо підхід вибіркового мультиплексування, щоб збалансувати дані всередині пакетів під час навчання. Мультиплексор видає об'єкти шляхом вибірки з потоків даних різних навчальних сховищ з урахуванням їхньої ваги. Коли окремі потоки вичерпано, продовжується вибірка з решти DataPipes відповідно до їхньої відносної ваги.

Висновки

Ми розглянули проблему отримання файлів для тренування моделей машинного навчання і запропонували засоби, що забезпечують найбільшу гнучкість експериментування, при цьому не призводячи до значних витрат на відміну від альтернативних рішень.

Використання проміжного сховища та мережевої файлової системи задовольнить більшість експериментальних вимог, але мережева файлова система спричиняє багато витрат на обслуговування та призводить до недостатнього використання інфраструктури. Підходи, що залишилися, працюватимуть у деяких випадках, але якщо ми збільшимо пропускну здатність моделі або кількість паралельних експериментів, або захочемо мати більший контроль над вмістом пакетів, ми можемо легко запровадити затримки.

Засіб отримання даних з використанням проміжного сховища демонструє значні переваги, порівняно з іншими методами. Цей підхід, базуючись на стисненні даних у формат gzip та організації їх у parquet файлах, сприяє значному прискоренню процесу завантаження та обробки даних. Ефективність стиснення, забезпечена повторенням байтових масивів, знижує обсяг трафіку та час завантаження, що є критичним для оптимізації обробки великих датасетів. Використання такого методу може суттєво підвищити загальну продуктивність та ефективності операцій машинного навчання “швидких” моделей.

2.3. Висновки до розділу

Ми обговорили ключові стратегії оптимізації управління великими даними і швидкості отримання напівструктурованих даних у машинному навчанні. Серед розглянутих існуючих та традиційних підходів на жаль жоден не позбавлений недоліків, таких як обмежена масштабованість, висока вартість та недостатня швидкість обробки даних. Запропоновані інноваційні рішення зосереджені на підвищенні ефективності та зниженні витрат, зокрема через використання розширених алгоритмів та інтегрованих систем. Запропоновані засоби мають значний потенціал для вирішення існуючих викликів і поліпшення загальної продуктивності операцій машинного навчання.

Спроектовані рішення виявляються кращими за існуючі методи через кілька ключових аспектів. По-перше, вони забезпечують покращену масштабованість, що дозволяє ефективніше обробляти великі обсяги даних. По-друге, запропоновані інтегровані підходи сприяють зниженню витрат на обробку даних. Також наші методи пропонують покращену швидкість обробки, що є важливим для забезпечення оперативності машинного навчання. Підсумовуючи, запропоновані рішення вирішують ряд невирішених проблем існуючих підходів, тим самим підвищуючи ефективність загальних операцій машинного навчання.

3. Результати застосування запропонованих засобів оптимізації

3.1. Дослідження оптимізації швидкодії отримання напівструктурованих даних в операціях машинного навчання

Опис вхідних даних для експериментування

Ми розглянемо вужчу область моделей машинного навчання для встановлення вимог, але подібний підхід може бути застосований до всіх

моделей. Ми обрали моделі розпізнавання мови, оскільки вони вимагають великих обсягів медіа-даних, а час пакетного навчання може бути відносно швидким.

В якості навчальної інфраструктури ми обрали сервер з 8 чіпами NVIDIA Tesla V100. Ми обрали ці чіпи, тому що вони розроблені на основі архітектури Volta, яка забезпечує достатню продуктивність, розумну енергоефективність і поки що цей чіп є найкращим варіантом у порівнянні з іншими чіпами за ціною одного терафлопса.

Для навчання будемо використовувати сегментовані аудіофайли і обмежимо тривалість до 20 секунд на сегмент. Така тривалість вибірки буде містити достатньо інформації і дозволить нам використовувати різноманітний контент в одній партії. Розмір навчальної вибірки - 1 мільйон годин. Типовий розмір хвильового аудіофайлу з частотою дискретизації 16 кГц становить 10 МБ/хв. Розмір сховища без стиснення для такого набору становить приблизно 1.000.000 * 60 * 10 = 600.000.000 МБ ~ 600 ТБ.

Виходячи з навчальних тестів моделей розпізнавання мови, ми можемо встановити пропускну здатність, що дорівнює 2 ГБ даних на секунду, як порогову пропускну здатність. Це трохи більше, ніж можуть обробити досліджені моделі. Переведемо пропускну здатність моделі з ГБ/с у файли/сек:

$$\textit{Model throughput, files per seconds} = \textit{Model throughput, MB per second} / \textit{Avg. file size, MB} = 2000 / 3{,}3 \sim 600 \text{ files per second}$$

Також для кращої читабельності розрахунків ми використовуємо деяке наближення, що 1 ТБ = 1 ТіБ = 1000 Гб і 1 Гб = 1000 Мб.

Отже, наші припущення:

- пропускна здатність моделі 2 ГБ/с ~ 600 файлів на секунду

- розмір набору даних - 600 ТБ
- для навчання моделі потрібне аудіо з частотою 16 кГц
- гранична швидкість завантаження даних зі сховища Blob становить 5.500 запитів на секунду
- дані зчитуються випадковим чином з рівною ймовірністю зчитування кожного файлу, всі файли мають однаковий розмір.

Порівняння швидкості отримання даних

Рішення	Швидкість
Пряма вибірка з BLOB сховища	15 MB/s на потік
Зберігання фрагментів даних у tar-архівах	80 MB/s на потік
Використання мережевої файлової системи	125 - 1000 MB/s/TiB (залежить від обраної персистентной файлової системи)
Використання проміжного сховища	80 MB/s на потік

Як бачите, одна мережева файлова система є відносно швидкою навіть без розпаралелювання, проте її максимальна швидкість обмежує нас від наявності декількох споживачів даних. Отже, максимальна кількість споживачів:

$$\textit{Max. throughput of storage} = \textit{Max. speed per unit of storage} * \textit{Count of units} = 600 \text{ GB/s}$$

Max. number of consumers = Max. throughput of storage / Model throughput = 600 / 2 = 300

Отже, наше обмеження для мережевої файлової системи - 300 одночасних споживачів даних на одну мережеву файлову систему.

Обидва рішення, де дані розбиті на частини, забезпечують хорошу швидкість і дозволяють нам розпаралелити вибірку даних на декілька потоків. Однак у випадку використання проміжного сховища нам не потрібно завантажувати непотрібні дані, оскільки ми зберігаємо в проміжному сховищі лише запитані та відфільтровані дані. Отже, якщо порівняти швидкість отримання корисних даних за допомогою цих 2 підходів, то рішення зі зберіганням фрагментів даних у tarballs сповільнить нас і збільшить загальний обсяг переданого трафіку. Давайте припустимо, що ми зберігаємо дані з 3 різними частотами дискретизації: 48 кГц, 24 кГц, 16 кГц (поширений випадок, коли дані використовуються для різних експериментів), і лише 25% даних відповідають нашому запиту на фільтрацію, і їх можна відфільтрувати за метриками, що зберігаються в метаданих. Також припустимо, що ступінь стиснення gzip-архівів дорівнює ступені стиснення gzip-файлів формату parquet.

*Size of 24kHz audio = Size of 16kHz audio * 1,5*

*Size of 48kHz audio = Size of 16kHz audio * 3*

*Total size of raw dataset = Size of 48kHz files + Size of 24kHz files + Size of 16kHz files = Size of 16kHz files * 5,5* = 600 * 5,5 = 3300 TB

*Coef. of useful data = Percent of data that meets requirements * Size of 16kHz / Total size of raw dataset* = 25% * 600 / 3300 ~ 4,5%

Отже, використання проміжного сховища зменшить обсяг трафіку в 22 рази!

Несправедливо вважати, що ми можемо створювати tarballs тільки з усіма даними, які у нас є. Давайте хоча б створимо окремі tarballs для різних частот дискретизації, тоді коефіцієнт корисності буде більшим. Давайте хоча б створимо окремі тарболи для різних частот дискретизації аудіо, тоді коефіцієнт корисних даних буде дорівнювати відсотку даних, що відповідають вимогам. Також припустимо, що коефіцієнт стиснення дорівнює 10.

Враховуючи подібну швидкість отримання даних, для досягнення вимог моделі нам може знадобитися приблизно стільки ж:

*Min. Parallel threads count = CEILING(Model throughput / (Coef. of useful data * Max. storage speed * Compression ratio))*

Для підходу зі зберіганням фрагментів даних у тарболах мінімальна кількість паралельних потоків = CEILING(2000 / (25% * 80 * 10)) = 10.

Якщо ми використовуємо проміжну файлову систему, мінімальна кількість паралельних потоків = CEILING(2000 / (80 * 10)) = 3.

Також припустимо, що кожен чанк містить 1000 аудіофайлів.

*Avg. chunk size = Count of files in chunk * (Avg. file size / Compression ratio)* = 1000 * (3,3 / 10) = 330 MB = 0,33 GB

Requests per second for single thread = Max. storage speed / Avg. chunk = 80 / 330 ~ 0,25

Для наших вхідних даних у найгіршому випадку кількість паралельних запитів дорівнюватиме кількості паралельних потоків (кількість запитів за секунду для одного потоку < 1). Виходячи з цього:

*Max. number of consumers = Throttling limit / (Parallel threads count * CEILING(Requests per second for single thread))*

Для підходу зі зберіганням блоків даних у тарболах максимальна кількість споживачів = 5500 / 10 ~ 550.

Якщо ми використовуємо проміжну файлову систему, максимальна кількість споживачів = 5500 / 3 = 1833.

Пряма вибірка зі сховища Blob є найповільнішою, і навіть після розпаралелювання запитів і якщо ми витягуємо тільки корисні дані, то Min. Кількість паралельних потоків = 2000 / 15 ~ 130.

Requests per second for single thread = Max. storage speed / Avg. file size = 15 / 3,3 = 4,5

Max. number of consumers = 5500 / (130 * 4,5) = 10

Виходячи з наведених вище розрахунків, пряма вибірка з blob-сховища вкрай обмежує можливість паралельного виконання декількох експериментів порівняно з іншими підходами і вимагає набагато більше ресурсів для вибірки даних.

Вартість підтримки і обслуговування

Припустимо, що вартість зберігання 1 ГБ даних у сховищі Blob дорівнює 1 одиниці ціни. Також ми повинні розглянути мультирегіональний доступ, щоб забезпечити відмовостійкість і уникнути регіональних обмежень. Виходячи з реальних кейсів використання, оптимально працювати в 4 регіонах (обидва узбережжя США, Західна та Центральна Європа).

Пряма вибірка з бінарного сховища великих об'єктів є найдешевшим варіантом - нам не потрібно дублювати дані (окрім реплікації) і не потрібно ніяких додаткових механізмів для їх перетворення.

*Total storage price = Replicas count * Price per GB * Total size of raw dataset * Compression ratio* = 4 * 1 * 3.300.000 * 1 = 13.200.000 units of price

Для будь-якого іншого рішення ці витрати неминучі. Тому давайте проігноруємо їх у порівнянні і припустимо, що пряма вибірка з Blob не потребує витрат на обслуговування, і вилучимо її з порівняння.

Коли ми використовуємо мережеву файлову систему, ми повинні пам'ятати, що отримання даних з різних зон доступності є надзвичайно дорогим. Перед доступом до даних ми повинні попередньо завантажити їх до мережевої файлової системи. Якщо ми хочемо використовувати кілька навчальних середовищ у різних регіонах, ми повинні реплікувати мережеві файлові системи та синхронізувати їх. Ціна за гігабайт пам'яті та витрати на передачу даних також значно вищі порівняно з іншими рішеннями.

На даний момент вартість зберігання даних у мережевій файловій системі приблизно в 30 разів вища, ніж у Blob-сховищі. Кожен регіон зазвичай має 3 зони доступності. Це означає, що ми маємо 12 реплік сховища. З попередніх розрахунків ми пам'ятаємо, що приблизний загальний розмір набору даних = 3300 ТБ = 3.300.000 ГБ

Total storage price = 12 * 30 * 3.300.000 * 1 = 1.188.000.000 units of price

У випадку проміжного сховища або зберігання фрагментів даних у tarballs нам доведеться створити механізм перетворення наборів даних з

вихідного сховища в інше, що потребуватиме певного обслуговування. Використання проміжного сховища дасть нам значну перевагу в обсязі трафіку. Він нижчий, ніж у інших рішеннях, тому що нам не потрібно завантажувати непотрібні дані, оскільки ми зберігаємо в проміжному сховищі лише запитані та відфільтровані дані. Загальний розмір сховища майже такий самий, як і при зберіганні фрагментів даних у tarballs. Передача даних безкоштовна в межах одного регіону, тому ми можемо мати лише 1 репліку на регіон. Таким чином, загальна ціна зберігання порівняно з мережевою файловою системою дорівнює:

*Total storage price = Replicas count * Price per GB * (Total size of raw dataset / Compression ratio)* = 4 * 1 * (3.300.000 / 10) = 1.320.000 units of price

Отже, результат показує, що наступні підходи в 900 разів дешевші! Однак слід пам'ятати, що у випадку проміжного сховища ми можемо мати більше однієї репліки в одному регіоні, оскільки ми попередньо фільтруємо дані, і для різних експериментів ми можемо використовувати різну логіку фільтрації, що може призвести до створення різних підмножин. Припустимо, що кожна з таких підмножин відфільтрує 75% даних. 4 репліки проміжного сховища коштуватимуть нам стільки ж, скільки 1 репліка сховища дьогтю. Щоб уникнути постійного збільшення кількості навчальних наборів даних, створених за допомогою цього підходу, ми можемо ввести період зберігання для таких наборів даних, щоб вони автоматично видалялися через певний проміжок часу.

Гнучкість експериментів з різними наборами даних

Безумовними лідерами тут є пряма вибірка зі сховища Blob та мережевої файлової системи. Ми маємо повну гнучкість у виборі будь-яких даних у будь-якому порядку, який нам потрібен.

Зберігання фрагментів даних у tar-архівах накладає жорсткі обмеження, оскільки ми маємо запитувати весь архів за посиланнями з метаданих, але використовувати лише його частину. Інакше ми не зможемо контролювати вміст пакетів. Це обмеження призводить до збільшення часу навчання або змушує нас адаптувати вміст пакетів до вмісту tarballs. Припустимо щось близьке до найгіршого випадку: ми хочемо взяти 10% даних, які відповідають нашим вимогам, з кожного тарболу, а потім переключитися на інший. Потім ми не хочемо повертатися до цього пакету, поки не опрацюємо 10% з кожного пакету.

*Coef. of useful data = Percent of data taken from batch * Percent of data that meets requirements* = 10% * 25% ~ 2,5%

*Min. Parallel threads count = CEILING(Model throughput / (Coef. of useful data * Max. storage speed * Compression ratio))* = 2000 / (2,5% * 80 * 10) = 100

*Max. number of consumers = Throttling limit / (Parallel threads count * CEILING(Requests per second for single thread))* = 5500 / (100 * 1) = 55

Отже, очевидно, що якщо ми хочемо керувати пакетним вмістом у такий спосіб, продуктивність впаде в 10 разів.

Проміжне сховище з табличними даними дасть нам менше контролю над порядком елементів в одному наборі даних, але дані можна перемішати перед створенням такого сховища. Таким чином, маючи контроль над партіями та навчальним сховищем, ми зможемо виконати вимоги експерименту.

Час налаштування навчальних наборів даних

Перш ніж ми зможемо розпочати навчання у випадку з мережевою файловою системою, ми повинні створити її та попередньо завантажити до неї файли. Підходи з проміжним сховищем і tarballs є швидшими, оскільки нам потрібно лише запустити завдання перетворення, і це перетворення можна розпаралелити. Час, необхідний для виконання цих операцій, приблизно однаковий, тому тут немає явного переможця.

Пряма вибірка не потребує додаткових налаштувань.

Висновки

Враховуючи, що ми навчаємо одну і ту ж модель, використовуючи різні підходи до отримання даних, загальний час навчання буде залежати від затримок, що створюються сховищами та обсягу трафіку даних. Якщо ми маємо великі затримки при доступі до даних, то графічні процесори будуть простоювати в очікуванні наступної партії даних.

Використання проміжного сховища та мережевої файлової системи задовольнить більшість експериментальних вимог, але мережева файлова система спричиняє багато витрат на обслуговування та призводить до недостатнього використання інфраструктури. Підходи, що залишилися, працюватимуть у деяких випадках, але якщо ми збільшимо пропускну здатність моделі або кількість паралельних експериментів, або захочемо мати більший контроль над вмістом пакетів, ми можемо легко запровадити затримки.

Виходячи з результатів порівняння, до винайдення нових ефективних графічних процесорів, запропоноване рішення є оптимальним способом навчання ML-моделей з великими обсягами даних.

3.2. Дослідження оптимізації управління великими даними в операціях машинного навчання

Опис вхідних даних для експериментування

Ми розглянемо можливі способи версіонування даних за наступних умов:

- початковий розмір сховища становить 1 петабайт;
- щотижня змінюється 100 терабайт даних, і так відбувається протягом року (52 тижні).
- Для простоти будемо вважати, що загальний розмір даних не змінюється.

Отже, ми маємо наступні вхідні дані:

$$d = Weekly\ changes\ size = 0.1\ PB$$
$$s = Dataset\ size = 1\ PB$$
$$w = Weeks\ count = 52$$

Проведемо дослідження способів резервного копіювання

Мета полягає в тому, щоб мати можливість отримувати знімки наборів даних за довільний тиждень.

Для повних копій:

$$Total\ size = s + s * w = 1 + 1 * 52 = 53\ PB$$

Кількість копій, задіяних у пошуку даних, завжди буде дорівнювати 1, оскільки ми маємо зображення за будь-який тиждень.

Для інкрементальних копій:

$$Total\ size = s + d * w = 1 + 0.1 * 52 = 6.2\ PB$$

Кількість копій, задіяних у пошуку даних, буде залежати від часу, а саме: вона дорівнює номеру тижня, знімок якого ми хочемо отримати, і лінійно зростає з часом. У нашому випадку він знаходиться в діапазоні від 1 до 52.

Для підходу з інкрементною контрольною точкою ми будемо створювати повну копію кожні 4 тижні, а інкрементні копії - щотижня.

$$Total\ size = s + (s\ *\ (w/4)\ ) + (d\ *\ w\ *\ (3/4)) = 17.9\ PB$$

Кількість копій, що беруть участь у пошуку даних, буде в діапазоні від 1 до 4, оскільки для кожної повної копії ми зберігаємо не більше 3 інкрементних копій.

Для диференціальних копій:

$$Total\ size = s + \sum_{i=1}^{w} d\ *\ i = 1 + \sum_{i=1}^{52} 0.1\ *\ i = 138.8\ PB$$

Кількість копій, задіяних у пошуку даних, завжди буде дорівнювати 2, тому що ми маємо диференціальну копію для кожного тижня.

Для підходу з диференційованою контрольною точкою ми створюватимемо повну копію кожні 4 тижні, а диференціальні копії - щотижня.

$$Total\ size = s + (s\ *\ (w/4)\ ) + ((\sum_{i=1}^{3} d\ *\ i)\ *\ (w/4)) = 21.8\ PB$$

Кількість копій, що беруть участь у відновленні даних, також завжди буде дорівнювати 2, оскільки ми маємо диференціальну копію для кожного тижня, як і в попередньому підході.

Як бачимо, існують різні підходи до класичного резервного копіювання, всі вони досить ефективні з точки зору відновлення даних, але всі вони мають певні недоліки:

- Існує ймовірність ненавмисного збереження дубльованих даних.
- Неможливість отримати знімок даних за довільний момент часу, оскільки ми обмежені графіком резервного копіювання
- Якщо нам заборонено зберігати якісь дані, нам потрібно видалити їх з усіх копій
- Якщо до конвеєра обробки даних додаються кроки, то попередні знімки даних не будуть містити необхідних перетворень.

Проведемо дослідження способу збереження копій файлів при зміні

Загальний розмір сховища:

$$Total\ size = s + d * w = 1 + 0.1 * 52 = 6.2\ PB$$

Всі версії даних зберігаються в сховищі, оптимізованому для читання, що означає, що швидкість доступу до останньої версії даних буде еквівалентна швидкості доступу до першої версії.

Недоліком такого підходу є функціональні обмеження:

- Існує ймовірність ненавмисного зберігання дублікатів даних.
- Якщо до конвеєра обробки даних додаються кроки, нам потрібно визначити, які дані мають які версії, і виконати необхідні перетворення над усіма даними. Якщо цього не зробити, лише остання версія даних міститиме всі необхідні перетворення.
- У випадку, якщо нам потрібно зробити кілька версій одного і того ж перетворення (наприклад, з різними параметрами), нам потрібно мати окремий механізм версій для створення дерев версій.

Проведемо дослідження версіонування файлів за допомогою хеш-сум

Загальний обсяг зберігання буде меншим або дорівнюватиме початковому обсягу та обсягу реальних змін:

$$Total\ size <= s + d * w = 1 + 0.1 * 52 = 6.2\ PB$$

Цей метод вимагає зберігання метаданих з посиланнями на версії файлів. Коли ми додаємо нову трансформацію до конвеєра обробки, ми можемо легко отримати всі файли, включаючи всі їхні версії, відповідно до цих метаданих.

Для прикладу, розглянемо сценарій, в якому ми оновили лише версію певної бібліотеки для трансформації. В результаті було змінено лише 10% файлів. Водночас, зберігаючи різні версії в метаданих, ми можемо легко отримати як результати трансформації з різними версіями бібліотеки, так і знімки даних на довільний момент часу.

Висновки

В результаті нашого дослідження різних методів зберігання та версіонування файлів ми підтвердили теоретичну ефективність способу з використанням хеш-сум, виявлено значні переваги обраного підходу у порівнянні з іншими методами. Використання хеш-сум дозволяє ефективно зменшити загальний обсяг зберігання в порівнянні з традиційними способами, запобігаючи ненавмисному дублюванню даних, забезпечуючи при цьому легкий доступ до різних версій файлів.

Ця стратегія забезпечує високу гнучкість у версіонуванні та легкість управління різними версіями файлів, що є ключовим для оптимізації процесів машинного навчання. Застосування цього підходу значно покращує ефективність маніпулювання даними, що є важливим фактором у забезпеченні швидкості та точності машинного навчання.

Базуючись на результатах дослідження ми можемо стверджувати, що запропонована стратегія задовольняє нашим вимогам і може бути застосована для оптимізації машинного навчання, оскільки вона підвищує ефективність обробки даних, зменшує витрати на зберігання та полегшує управління версіями даних.

3.3. Висновки до розділу

Розроблені засоби забезпечують підвищену ефективність управління даними в операціях машинного навчання. При наших вхідних даних їх поєднання дозволяє збільшити можливу кількість паралельних експериментів без необхідності реплікувати сховище **до 200 разів**, зменшити кількість мережевого трафіку **на один порядок** та зменшити видатки на обслуговування сховища даних **на один-три порядки** в порівнянні з існуючим рішеннями, забезпечуючи гнучкість у виборі вхідних наборів даних та відтворюванні експериментів. Отримані рішення не вводять додаткових затримок, а навпаки пришвидшують можливість отримання великих обсягів даних, що дозволяє тренувати навіть швидкі моделі нівелюючи марні витрати коштовних інфраструктурних та часових ресурсів. В залежності від обраних серверів для тренування моделі з пропускною здатністю 2 GB/s запропоновані способи оптимізації можуть **пришвидшити час тренування у 1.5-3 рази**, а як наслідок пропорційно зменшити вуглецевий слід, вартість тренувальної інфраструктури.

Створена архітектура забезпечує оптимізований доступ до даних завдяки структурованим метаданим, ефективно відповідаючи на всі встановлені функціональні вимоги. Розроблена система складається з модулів, кожен з яких відповідає за свою функцію: управління версіями, зберігання файлів, обробку даних, створення тренувальних наборів та завантаження даних у тренувальне середовище забезпечуючи їх незалежну взаємодію. Реалізовані інтерфейси дозволяють отримувати

будь-які версії файлів, включаючи ті, що пройшли перетворення та забезпечують можливість відповідати специфічним регіональним та законодавчим обмеженням.

Важливим аспектом розробленого рішення є його масштабованість на довільну кількість паралельних експериментів та технологічна агностичність, що є ключовим фактором для інтегрування у виробничі процеси різних секторів економіки, що використовують операції машинного навчання та аналітики великих даних. Проведена робота може допомогти різноманітним підприємствам, незалежно від масштабу департаментів дослідження та розробки для ефективного управління великими даними та оптимізації процесів машинного навчання.

Висновки

Ми дослідили проблеми та рішення в оптимізації обчислювальних ресурсів для машинного навчання. Пророблена робота охоплює повний цикл операцій машинного навчання, роблячи основний фокус на роботі з даними. Ми розглянули головні інфраструктурні та програмні виклики, функціональні обмеження існуючих рішень, механізми обробки даних, використання ресурсів під час навчання моделей, інструменти оцінки та моніторингу, існуючі рішення для оптимізації та негативні наслідки відсутності оптимізації.

Дослідження заглиблюється в різні стратегії управління великими даними в машинному навчанні, способи оптимізації швидкості пошуку напівструктурованих даних. На основі виявлених проблем ми ідентифікували вузькі місця, що потребують оптимізації та розробили інструменти для забезпечення відповідального використання ресурсів у операціях машинного навчання. Порівняння запропонованих рішень з існуючими чітко демонструє ефективність запроваджених механізмів і наочні результати пришвидшення доступу до даних, зменшення розміру

необхідного сховища та кількості мережевого трафіку, а також збільшення кількості можливих паралельних експериментів та гнучкості експериментування та маніпулювання з даними. Внаслідок, чого при використані запропонованих оптимізацій ми отримуємо можливість зменшити загальний час тренування моделей, витрати ресурсів та вуглецевий слід, а отже пришвидшили технологічний прогрес та економічний розвиток задовольняючи різноманітні законодавчі вимоги щодо збереження та видалення даних.

Дисертація має на меті зробити значний внесок у розвиток систем зберігання та управління даними для машинного навчання, вирішуючи ключові проблеми в цій галузі.

Список літератури

1. Database Internals: A Deep-Dive Into How Distributed Data Systems Work / Алекс Петров (18 Жовтня, 2019)
2. Practical DataOps: Delivering Agile Data Science at Scale / Харвіндер Атвал (9 Грудня, 2019)
3. What is the difference between incremental, differential, and full backup? / Acronis (28 Вересеня, 2023)
4. BigData/Streaming: Amazon S3 Data Lake | Storing & Analyzing the Streaming Data on the go (in near real-time) | A Serverless Approach / Бурхануддін Бхопалвала – Towards Data Science (4 Лютого, 2020)
5. Reduce costs and complexity of ML preprocessing with Amazon S3 Object Lambda / Нейт Бахмайєр – dataintegration.info (10 лютого, 2022)
6. Optimizing costs in Amazon Elastic Inference with TensorFlow / Корі Прюс та Шрінівас Ханабе – AWS Machine Learning Blog (10 Липня, 2019)

7. Performance Measurements of Supercomputing and Cloud Storage Solutions / Майкл Джонс, Джеремі Кепнер, Вільям Арканд, Девід Бестор, Білл Бержерон, Віджай Гадепаллі, Майкл Хоул, Метью Хаббелл, Пітер Міхаліас, Ендрю Праут, Альберт Ройтер, Сіддхарт Самсі, Пол Монтіціолло - Лабораторія Лінкольна MTI, Лексінгтон, штат Массачусетс, США (2017)
8. A method for optimizing the performance of semi-structured data fetching in machine learning operations / Романкевич В.О, Марченко О.Б., (Листопад, 2023)
9. Strategy to improve the efficiency of big data management in machine learning operations / Романкевич В.О, Марченко О.Б., (Грудень, 2023)