Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Метод 1-бітового квантування ваг для великих мовних моделей

(тема)

Виконав:
здобувач другого року навчання,
групи ДСм-23-1
Приходько Д.М.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Науки про дані (Data Science)
(повна назва спеціалізації)
Керівник доц. Шевченко О.Ю.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ О.В. Золотухін
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Науки про дані (Data Science)
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Приходьку Данилу Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Метод 1-бітового квантування ваг для великих мовних моделей

затверджена наказом університету від 22 листопада 20 24 р. № 1238Ст

2. Термін подання студентом роботи до екзаменаційної комісії 23 січня 20 25 р.

3. Вихідні дані до роботи Науково-технічні публікації, дані Інтернет-джерел та відомих наукових проєктів, Python documentation, набір даних для тренування та тестування системи.

4. Перелік питань, що потрібно опрацювати в роботі
1) Теоретичні основи 1-бітового квантування
2) Алгоритми 1-бітового квантування ваг
3) Розробка програмного рішення та моделювання
4) Експерементальна частина

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	22.11.2024	виконано
2	Аналіз предметної галузі та вивчення літератури	25.11.2024	виконано
3	Формулювання мети, завдань та методів дослідження	27.11.2024	виконано
4	Розробка теоретичної моделі 1-бітового квантування	01.12.2024	виконано
5	Вибір програмного забезпечення для реалізації	05.12.2024	виконано
6	Реалізація алгоритму 1-бітового квантування ваг	10.12.2024	виконано
7	Проведення тестувань моделі на мовних моделях	15.12.2024	виконано
8	Аналіз отриманих результатів та їх інтерпретація	21.12.2024	виконано
9	Оптимізація алгоритму за результатами тестувань	23.12.2024	виконано
10	Оформлення розділів пояснювальної записки	28.12.2024	виконано
11	Підготовка матеріалів для презентації та доповіді	04.01.2025	виконано
12	Перевірка пояснювальної записки на нормоконтроль	10.01.2025	виконано
13	Узгодження роботи з керівником	12.01.2025	виконано
14	Подання роботи для допуску завідувачем кафедри	13.01.2025	виконано
15	Захист кваліфікаційної роботи	23.01.2025	виконано

Дата видачі завдання 25 листопада 2024 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Шевченко О.Ю.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 71 с., 25 рисунків, 4 табл., 3 дод., 18 джерел.

ВЕЛИКІ МОВНІ МОДЕЛІ, КВАНТУВАННЯ, МАШИННЕ НАВЧАННЯ, НЕЙРОННІ МЕРЕЖІ, ОБЧИСЛЮВАЛЬНІ ВИТРАТИ, ОПТИМІЗАЦІЯ.

Об’єктом дослідження є великі мовні моделі, що застосовуються для вирішення завдань природної мовної обробки.

Предметом дослідження виступає метод 1-бітового квантування ваг, спрямований на оптимізацію їхньої роботи та зменшення обчислювальних витрат.

Метою роботи є аналіз можливостей використання методу 1-бітового квантування ваг для великих мовних моделей, оцінка його впливу на точність і продуктивність, а також розробка рекомендацій щодо впровадження цього методу в реальні задачі. У дослідженні здійснено теоретичний аналіз літератури, розроблено алгоритми, проведено експерименти на реальних моделях і здійснено статистичний аналіз отриманих результатів.

Робота включає огляд теоретичних основ квантування ваг із фокусом на 1-бітове квантування, що дозволяє значно скоротити обчислювальні ресурси без суттєвого зниження точності. Проведено експериментальне дослідження для оцінки ефективності цього підходу на великих мовних моделях. Отримані результати підтверджують можливість використання методу у середовищах із обмеженими ресурсами.

Практичне значення роботи полягає в розширенні доступності великих мовних моделей через зниження їхньої обчислювальної складності, що сприяє їх впровадженню у мобільні пристрої та інші обмежені середовища.

ABSTRACT

Master’s thesis contains: 71 pp., 25 fig., 4 tabl., 3 ann., 18 references.

COMPUTATION COST, LARGE LANGUAGE MODELS, MACHINE LEARNING, NEURAL NETWORKS, OPTIMIZATION, QUANTIZATION.

The object of research is large language models used to solve problems of natural language processing.

The subject of the research is the method of 1-bit quantization of weights, aimed at optimizing their work and reducing computational costs.

The purpose of the work is to analyze the possibilities of using the method of 1-bit quantization of weights for large language models, to evaluate its impact on accuracy and performance, as well as to develop recommendations for the implementation of this method in real problems. In the study, a theoretical analysis of the literature was carried out, algorithms were developed, experiments were carried out on real models and statistical analysis of the obtained results was carried out.

The work includes a review of the theoretical foundations of weight quantization with a focus on 1-bit quantization, which allows to significantly reduce computational resources without significantly reducing accuracy. An experimental study was conducted to evaluate the effectiveness of this approach on large language models. The obtained results confirm the possibility of using the method in environments with limited resources.

The practical significance of the work is to expand the availability of large language models by reducing their computational complexity, which facilitates their implementation in mobile devices and other limited environments.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів 7
Вступ 8
1 Теоретичні основи 1-бітового квантування 10
1.1 Основи нейронних мереж та їх архітектур 10
1.2 Концепція квантування ваг у нейронних мережах 14
1.3 Моделі великих мовних моделей (LLMs) 16
1.4 Історія розвитку 1-бітового квантування 18
2 Алгоритми 1-бітового квантування ваг 21
2.1 Основи алгоритмів квантування 21
2.2 Специфіка реалізації 1-бітового квантування 23
2.3 Аналіз ефективності алгоритму на LLMs 25
2.4 Складності та обмеження підходу 28
3 Розробка програмного рішення та моделювання 30
3.1 Аналіз вимог до середовища дослідження 30
3.2 Моделювання нейромережі з підтримкою 1-бітового квантування 32
3.3 Реалізація програмного рішення 36
3.4 Навчання та валідація моделі 44
3.5 Проблеми та виклики під час реалізації 48
4 Експерементальна частина 51
4.1 Опис вибраних моделей для дослідження 51
4.2 Налаштування експериментів та метрики оцінки 55
4.3 Результати застосування 1-бітового квантування 59
4.4 Аналіз отриманих даних 63
Висновки 66
Перелік джерел посилання 68
Додаток А Відомість кваліфікаційної роботи 71

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AI – Artificial Intelligence – штучний інтелект;

BERT – Bidirectional Encoder Representations from Transformers – двонаправлені представлення кодувальника з трансформерів;

BLEU – Bilingual Evaluation Understudy – метрика оцінки якості машинного перекладу (білатеральна оцінка);

CNN – Convolutional Neural Network – згорткова нейронна мережа;

GPT – Generative Pre-trained Transformer – генеративний попередньо навчений трансформер;

GPU – Graphics Processing Unit – графічний процесор;

JAX – Just After eXecution – бібліотека для машинного навчання, оптимізована для обчислень на GPU та TPU;

LLM – Large Language Model – велика мовна модель;

MLP – Multilayer Perceptron – багатошаровий перцептрон;

NLP – Natural Language Processing – обробка природної мови;

NLTK – Natural Language Toolkit – інструментарій для обробки природної мови в Python;

ReLU – Rectified Linear Unit – функція активації з випрямленою лінійною одиницею;

RMSprop – Root Mean Square Propagation – алгоритм оптимізації, який враховує середньоквадратичну швидкість зміни градієнта;

RNN – Recurrent Neural Network – рекурентна нейронна мережа;

RTX – Ray Tracing Texel eXtreme – серія графічних процесорів NVIDIA з підтримкою трасування променів у реальному часі;

SGD – Stochastic Gradient Descent – стохастичний градієнтний спуск;

TPU – Tensor Processing Unit – тензорний процесор (апаратна платформа, створена Google для машинного навчання).

ВСТУП

У сучасному світі нейронні мережі, зокрема великі мовні моделі (Large Language Models, LLMs), стали невід'ємною частиною численних технологій, що використовуються в повсякденному житті та промислових сферах. Високий рівень їхньої продуктивності обумовлений значною кількістю параметрів, що забезпечують здатність моделі обробляти складні завдання, такі як машинний переклад, генерація текстів, аналіз контексту та інші. Проте використання таких моделей пов'язане з низкою викликів, серед яких найбільшим є потреба в значних обчислювальних ресурсах. Це створює бар'єри для їх застосування у середовищах з обмеженими ресурсами, таких як мобільні пристрої, або для компаній із невеликими фінансовими можливостями.

Одним із перспективних підходів до зменшення обчислювальних витрат є використання методів квантування ваг, зокрема 1-бітового квантування. Цей метод дозволяє суттєво зменшити обсяг пам'яті, необхідної для зберігання ваг, і підвищити швидкість виконання моделей, зберігаючи при цьому достатню точність. Це відкриває нові можливості для оптимізації великих мовних моделей, роблячи їх доступнішими для широкого кола користувачів.

На сьогоднішній день квантування як метод оптимізації активно досліджується в галузі машинного навчання. Існує багато праць, які вивчають різні підходи до квантування, включаючи багатобітові та низькорівневі методи. Проте 1-бітове квантування залишається мало дослідженим через складність підтримки балансу між точністю моделі та її ефективністю. Попри це, окремі роботи демонструють потенціал цього методу, зокрема у випадках, коли необхідно виконувати обчислення у реальному часі чи працювати з великими обсягами даних.

Об’єктом дослідження є великі мовні моделі, які використовуються для задач природної мовної обробки. Предметом дослідження є метод 1-

бітового квантування ваг для оптимізації обчислювальних витрат таких моделей.

Метою цієї роботи є аналіз можливостей застосування методу 1-бітового квантування ваг для великих мовних моделей, визначення його впливу на продуктивність та точність моделей, а також розробка рекомендацій щодо його практичного використання. Для досягнення мети передбачено виконання таких завдань, як огляд теоретичних основ квантування, розробка та аналіз алгоритмів, а також проведення експериментів на реальних моделях.

Для досягнення поставленої мети було використано методи теоретичного аналізу літератури, математичного моделювання, експериментального дослідження, а також статистичний аналіз отриманих даних.

Таким чином, дослідження сфокусоване на пошуку ефективних підходів до оптимізації великих мовних моделей, що є актуальним завданням сучасної науки та техніки.

1 ТЕОРЕТИЧНІ ОСНОВИ 1-БІТОВОГО КВАНТУВАННЯ

1.1 Основи нейронних мереж та їх архітектур

Нейронні мережі є одним із ключових інструментів сучасного машинного навчання, що дозволяють створювати моделі для вирішення різноманітних задач, таких як розпізнавання образів, обробка природної мови, автоматичне генерування текстів та багато інших. Ці системи базуються на принципах, натхнених функціонуванням біологічного мозку, зокрема взаємодією між нейронами через синапси. Математичне відображення цієї концепції стало основою для створення штучних нейронів, які є базовими будівельними блоками нейронних мереж.

Штучний нейрон за своєю суттю є математичною моделлю, що приймає на вхід сигнали (вектори даних), обробляє їх через вагові коефіцієнти та застосовує функцію активації для отримання вихідного значення. Вагові коефіцієнти визначають, наскільки сильно кожен вхідний сигнал впливає на кінцевий результат. Функція активації додає нелінійність, що дозволяє мережі моделювати складні взаємозв'язки у даних. Основними функціями активації є сигмоїдальна (рисунок 1.1) функція, ReLU (Rectified Linear Unit), tanh та інші.

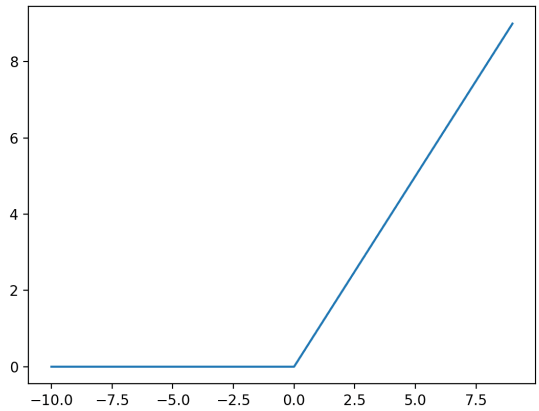

Рисунок 1.1 – ReLU

Архітектура нейронної мережі визначає, як штучні нейрони з'єднуються між собою. Найпростішою є одношарова перцептронна мережа, яка складається з одного шару нейронів. Однак така архітектура має обмежені можливості і не здатна моделювати складні функції. Для подолання цього обмеження розроблено багатошарові мережі (Multilayer Perceptrons, MLP), де нейрони організовані в кілька шарів: вхідний, приховані шари та вихідний. Кожен шар виконує свою роль у перетворенні даних і дозволяє мережі краще «розуміти» вхідні сигнали.

Серед сучасних архітектур нейронних мереж можна виділити кілька основних категорій. Згорткові (рисунок 1.2) нейронні мережі (Convolutional Neural Networks, CNNs) спеціалізуються на обробці просторових даних, таких як зображення, завдяки використанню згорткових шарів.

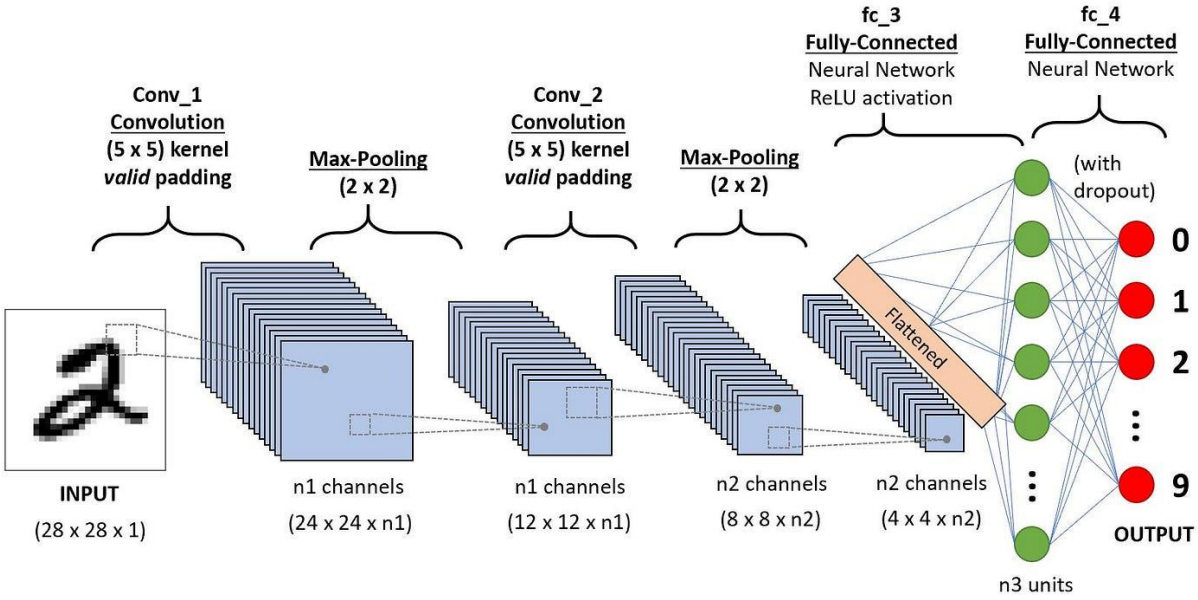

Рисунок 1.2 – CNN

Рекурентні (рисунок 1.3) нейронні мережі (Recurrent Neural Networks, RNNs) орієнтовані на роботу з послідовностями даних, наприклад текстами або часовими рядами, і забезпечують обробку інформації з урахуванням її контексту. Трансформерні моделі (Transformers), які є основою сучасних

великих мовних моделей, таких як GPT, відзначаються здатністю паралельно обробляти довгі послідовності даних завдяки механізму самоуваги (self-attention).

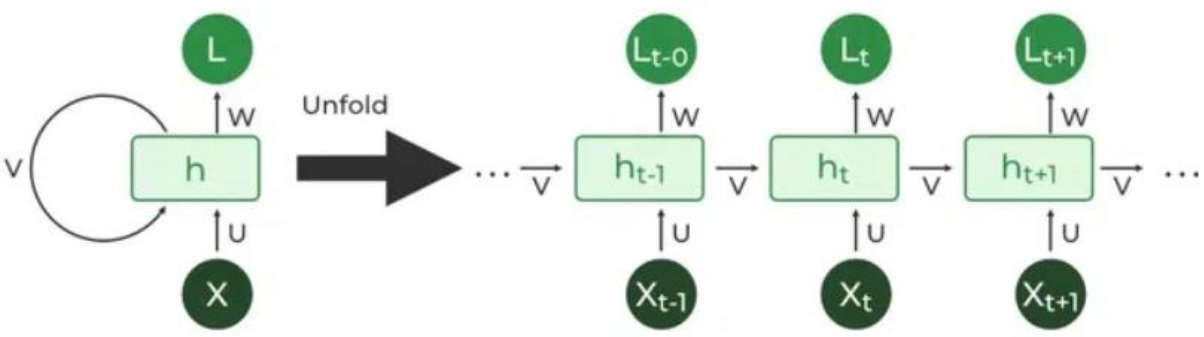

Рисунок 1.3 – RNN

Ефективність нейронної мережі залежить не лише від її архітектури, а й від процесу навчання, під час якого ваги адаптуються таким чином, щоб мінімізувати помилку моделі. Навчання зазвичай базується на методі зворотного поширення помилки (backpropagation) у поєднанні з алгоритмами оптимізації, такими як стохастичний градієнтний спуск (SGD) або його покращені варіанти (Adam, RMSprop).

Вибір архітектури нейронної мережі значною мірою залежить від конкретного завдання та типу даних, з якими працює система. Наприклад, для задач розпізнавання зображень або відео доцільно використовувати згорткові мережі (CNN), оскільки вони ефективно обробляють просторову інформацію.

Таким чином, основи нейронних мереж включають розуміння їхньої математичної суті, типів архітектур та підходів до навчання. Ці знання створюють базу для подальшого дослідження методів оптимізації, таких як квантування ваг, що є актуальною темою сучасної науки про штучний інтелект.

Таблиця 1.1 описує основні архітектури нейронних мереж, їх призначення, ключові особливості та приклади застосування.

Таблиця 1.1 – Основні архітектури нейронних мереж

Архітектура	Призначення	Ключові особливості	Приклади застосування
Багатошаровий перцептрон (MLP)	Загальні задачі машинного навчання.	Складається з вхідного, прихованих та вихідного шарів; всі нейрони повністю з'єднані.	Класифікація, регресія, прогнозування
Згорткова мережа (CNN)	Обробка просторових даних, таких як зображення.	Використовує згорткові шари для автоматичного виділення ознак; ефективна для двовимірних даних.	Розпізнавання об'єктів на зображеннях, медична діагностика
Рекурентна мережа (RNN)	Обробка послідовностей даних, враховуючи контекст.	Використовує зворотні зв'язки; враховує залежності між елементами послідовностей.	Обробка тексту, машинний переклад, аналіз часових рядів
Двонаправлена RNN (BiRNN)	Аналіз послідовностей із врахуванням майбутнього та минулого контексту.	Поєднує два RNN, що обробляють дані в прямому і зворотному напрямах.	Машинний переклад, аналіз тональності текстів
Трансформер (Transformers)	Ефективна обробка довгих послідовностей, масштабування на великі моделі.	Використовує механізм самоуваги (self-attention), дозволяє паралельну обробку послідовностей.	GPT, BERT, моделі для обробки природної мови
Автокодувальник (Autoencoder)	Зменшення розмірності даних, виділення ознак, генерація даних.	Складається з енкодера (зменшує розмірність) та декодера (відновлює дані).	Виділення ознак, генерація зображень
Генеративна мережа (GAN)	Генерація нових даних, які схожі на вихідні.	Складається з генератора та дискримінатора, що змагаються між собою.	Генерація зображень, відео, підвищення якості даних
Графова мережа (GNN)	Обробка графових даних, що представляють структури, такі як мережі чи зв'язки.	Використовує вершини і ребра для моделювання зв’язків між об’єктами.	Рекомендаційні системи, аналіз соціальних мереж

Ця таблиця відображає ключові аспекти популярних архітектур і допомагає зрозуміти, як вони застосовуються у різних задачах.

1.2 Концепція квантування ваг у нейронних мережах

Квантування ваг є одним із ключових підходів для оптимізації нейронних мереж, особливо в умовах, коли необхідно зменшити їх обчислювальну складність і обсяг пам’яті, який вони займають. Цей метод спрямований на зменшення точності чисел, що використовуються для представлення вагових коефіцієнтів, зберігаючи при цьому прийнятну продуктивність моделі. Основна ідея квантування полягає в тому, щоб зменшити кількість бітів, необхідних для представлення кожного вагового коефіцієнта, без значного впливу на якість навченої моделі.

Класичні нейронні мережі, як правило, використовують вагові коефіцієнти у форматі чисел з плаваючою точкою, зазвичай 32-бітних. Цей підхід забезпечує високу точність обчислень, але створює значне навантаження на апаратні ресурси, особливо при роботі з великими мовними моделями, які можуть мати мільярди параметрів. Квантування дозволяє замінити 32-бітні ваги на менш ресурсомісткі представлення, такі як 16-бітні, 8-бітні або навіть 1-бітні (рисунок 1.4) значення.

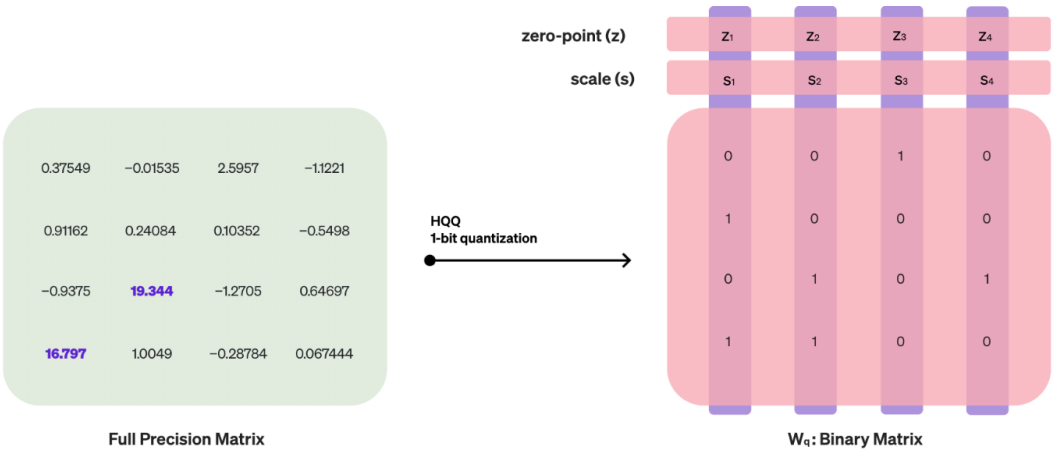

Рисунок 1.4 – Однобітове квантування

Метод 1-бітового квантування є особливо екстремальною формою цього підходу. У ньому кожен ваговий коефіцієнт представляється лише одним бітом, тобто він може приймати лише два значення, наприклад, - 1 або +1. Це значно зменшує обсяг пам’яті, необхідний для збереження моделі, та прискорює обчислення, оскільки операції множення спрощуються до логічних або арифметичних операцій. Проте така сильна дискретизація може призводити до суттєвого погіршення якості роботи моделі, якщо не застосовувати додаткових коригувальних методів.

Одним із головних викликів у процесі квантування є мінімізація втрат точності моделі. Для цього використовуються різноманітні стратегії адаптації. Наприклад, моделі можуть спочатку тренуватися з високою точністю ваг, а потім поступово переходити до квантування. Це дозволяє мережі частково компенсувати втрати, викликані зменшенням точності. Крім того, для збереження точності часто застосовують спеціальні алгоритми перепідгонки (fine-tuning) після квантування, які допомагають моделі краще адаптуватися до нових обмежень.

Необхідно також зазначити, що метод квантування ваг ефективно працює лише для певних типів задач і архітектур. Наприклад, моделі для задач класифікації чи розпізнавання зображень, як правило, менш чутливі до зменшення точності ваг, ніж мовні моделі, які потребують високої точності для генерації контекстно-залежних результатів. Це робить необхідним детальний аналіз перед застосуванням квантування, щоб оцінити його вплив на продуктивність для конкретного застосування.

Метод квантування ваг, зокрема 1-бітове квантування, також активно розвивається завдяки впровадженню спеціалізованого апаратного забезпечення. Сучасні процесори та графічні карти все частіше підтримують обчислення із низькою точністю, що дозволяє ще більше прискорити виконання моделей, оптимізованих таким чином. Це відкриває нові перспективи для використання великих мовних моделей у пристроях з

обмеженими обчислювальними ресурсами, таких як мобільні телефони чи вбудовані системи.

Таким чином, концепція квантування ваг у нейронних мережах, особливо у формі 1-бітового представлення, є перспективним напрямом у розробці ефективних та масштабованих моделей. Її використання дозволяє зменшити витрати ресурсів, зберігаючи при цьому прийнятну якість роботи моделі. Це робить метод квантування актуальним інструментом у контексті зростаючої складності та обсягу сучасних нейронних мереж.

1.3 Моделі великих мовних моделей (LLMs)

Великі мовні моделі (Large Language Models, LLMs) є одним із найвизначніших досягнень у галузі штучного інтелекту та обробки природної мови (NLP). Вони являють собою багатошарові нейронні мережі, які здатні навчатися на величезних обсягах текстових даних для виконання широкого спектру мовних задач, таких як генерація тексту, переклад, відповіді на запитання, аналіз тональності та багато інших. Розвиток LLMs базується на використанні сучасних архітектур нейронних мереж, таких як трансформери, які дозволяють досягати високої точності й адаптивності.

Однією з ключових особливостей LLMs є їхня здатність розуміти контекст тексту завдяки механізму самоуваги (self-attention). Цей механізм дає змогу моделі ефективно враховувати зв’язки між словами або фразами навіть у дуже довгих текстах. Наприклад, трансформери, такі як GPT (Generative Pre-trained Transformer) і BERT (Bidirectional Encoder Representations from Transformers), використовують цей підхід для побудови складних семантичних зв’язків, що робить їх надзвичайно потужними у задачах NLP.

Щоб створити LLM, потрібні величезні обсяги текстових даних для навчання. Ці дані зазвичай включають тексти з різних джерел, таких як книги, статті, веб-сторінки, форуми тощо. Завдяки цьому моделі здатні

«засвоювати» широкі знання з різних тем, хоча ці знання обмежуються тим, що було надано у навчальних даних. Ключовим аспектом є також масштаб обчислювальних ресурсів, необхідних для тренування LLMs. Моделі такого класу потребують потужних кластерів графічних процесорів або спеціалізованих прискорювачів, а навчання може тривати кілька тижнів або навіть місяців.

LLMs відрізняються тим, що мають величезну кількість параметрів, які визначають зв'язки між різними частинами вхідних даних. Для прикладу, GPT-3 має 500 мільярдів параметрів (рисунок 1.5), що дозволяє їй обробляти надзвичайно складні мовні запити. Велика кількість параметрів забезпечує високу точність, але водночас ускладнює використання моделі в умовах обмежених ресурсів. Через це дослідники активно працюють над оптимізацією LLMs, включаючи методи, такі як квантування ваг або знеконцентроване навчання (distillation).

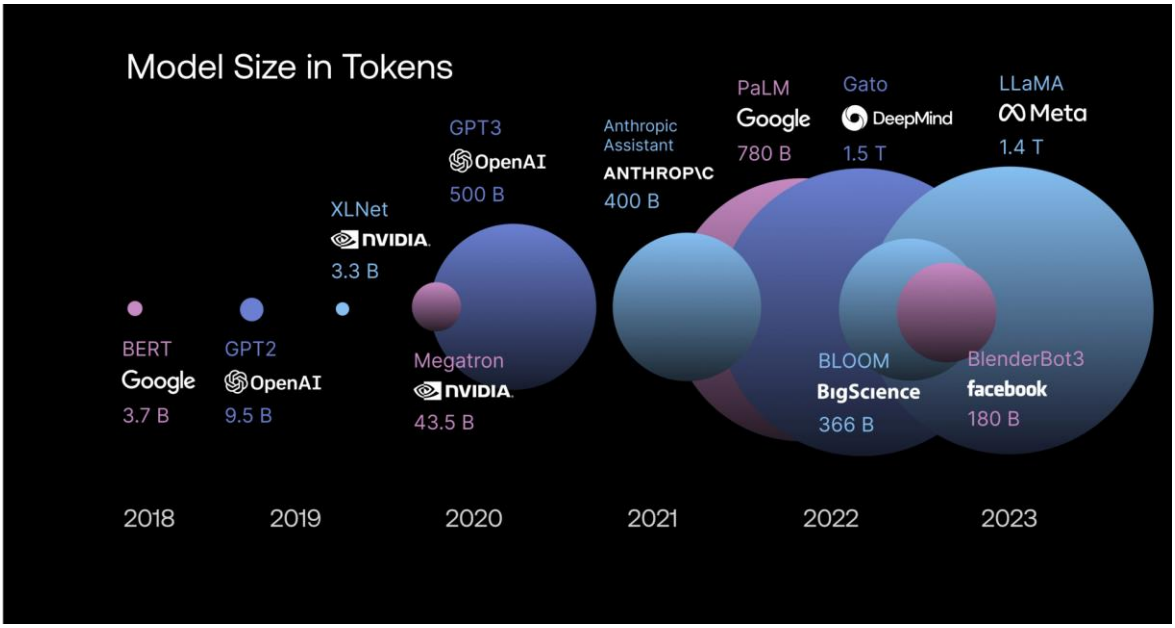

Рисунок 1.5 – Розміри різних моделей

Практичне застосування LLMs виходить за межі класичних задач NLP. Сьогодні такі моделі використовуються у бізнес-аналітиці, охороні

здоров’я, автоматизації клієнтської підтримки, створенні творчого контенту та багатьох інших сферах. Вони також демонструють здатність до узагальнення знань і вирішення нових задач, з якими вони не зустрічалися під час навчання. Це досягається завдяки так званому «few-shot learning» або «zero-shot learning», коли модель може адаптуватися до нових задач з мінімальною кількістю прикладів або навіть без них.

Однак, великі мовні моделі мають і свої обмеження. Серед основних проблем можна виділити високу залежність від обчислювальних ресурсів, труднощі з інтерпретацією результатів та можливість генерації некоректної або упередженої інформації. Через це дослідження у сфері LLMs спрямовані не лише на покращення їхньої продуктивності, а й на забезпечення прозорості, етичності та відповідальності їх використання.

Таким чином, моделі великих мовних моделей є ключовим інструментом у сучасному світі штучного інтелекту. Їх розвиток та застосування продовжують змінювати підходи до автоматизації обробки текстової інформації, відкриваючи нові горизонти для досліджень та комерційних рішень. Оптимізація та вдосконалення цих моделей є пріоритетними напрямами, що мають величезний потенціал для впливу на різні аспекти людського життя.

1.4 Історія розвитку 1-бітового квантування

Історія розвитку методу 1-бітового квантування ваг у нейронних мережах бере початок із загальної тенденції до оптимізації обчислювальних процесів у сфері штучного інтелекту. Зростання складності нейронних мереж та їх широке застосування у практичних задачах викликали потребу у зменшенні обчислювальної та пам'яттєвої вартості таких моделей. На початкових етапах еволюції методів квантування дослідники зосереджувалися на зменшенні точності ваг до 16-бітних або 8-бітних

значень. Однак потреба у ще радикальнішому зменшенні ресурсомісткості привела до концепції 1-бітового представлення.

Ранні роботи в цьому напрямку були переважно теоретичними і базувалися на математичних припущеннях щодо того, що нейронні мережі можуть навчитися компенсувати втрати інформації, викликані жорстким квантуванням. У 2010-х роках з'явилися перші експериментальні докази того, що моделі із сильно зниженою точністю можуть демонструвати задовільні результати в окремих задачах. Ці результати були надихаючими, але супроводжувалися значними втратами точності, що стримувало поширення 1-бітового квантування у практичних застосуваннях.

Ключовий прорив у розвитку цього підходу стався з удосконаленням алгоритмів оптимізації та появою технік, які дозволяли зменшити негативний вплив квантування. Одним із найважливіших кроків стало впровадження техніки перепідгонки моделі після квантування. Завдяки цьому підходу, нейронна мережа мала можливість «вчитися» пристосовуватися до умов із зниженою точністю, що значно підвищувало ефективність її роботи.

Ще одним ключовим аспектом розвитку 1-бітового квантування стало вдосконалення апаратних засобів. На межі 2010-х і 2020-х років провідні виробники обчислювального обладнання, такі як NVIDIA і Google, почали впроваджувати спеціалізовані обчислювальні блоки, здатні ефективно працювати з моделями низької точності. Це дало змогу використовувати 1-бітове квантування не лише у дослідницьких експериментах, а й у реальних промислових додатках, наприклад, у мобільних пристроях чи вбудованих системах.

Наукова спільнота також звернула увагу на математичну сторону питання. Розробка нових стратегій квантування, таких як стохастичне або адаптивне квантування, дозволила краще зберігати значущу інформацію під час переходу до 1-бітових ваг. Наприклад, стохастичне квантування дозволяє вибирати значення ваг з урахуванням ймовірності, що залежить від

оригінального значення, що мінімізує помилки, накопичені в процесі обчислень.

Крім того, ключовою частиною історії 1-бітового квантування є адаптація цього підходу до різних архітектур нейронних мереж. Якщо ранні дослідження фокусувалися на простих моделях, таких як багатошарові персептрони, то з часом 1-бітове квантування почало використовуватися у складніших структурах, зокрема у згорткових нейронних мережах (CNN) і трансформерах. Цей процес вимагав розробки специфічних алгоритмів і модифікацій, які враховували особливості роботи кожного типу архітектури.

У сучасному контексті 1-бітове квантування продовжує розвиватися як ефективний метод оптимізації великих мовних моделей і інших ресурсомістких нейронних мереж. Завдяки його впровадженню стало можливим вирішення задач у середовищах із обмеженими ресурсами, таких як IoT-пристрої чи роботи. Разом із тим, дослідники продовжують шукати способи покращення цього методу, щоб зробити його ще більш універсальним і точним.

Таким чином, історія розвитку 1-бітового квантування демонструє поступову еволюцію від теоретичних концепцій до повноцінних рішень, які знаходять застосування у практиці. Цей процес був зумовлений не лише потребою в оптимізації нейронних мереж, але й розвитком супутніх технологій, що зробило 1-бітове квантування одним із найперспективніших напрямів у сучасному штучному інтелекті.

2 АЛГОРИТМИ 1-БІТОВОГО КВАНТУВАННЯ ВАГ

2.1 Основи алгоритмів квантування

Квантування ваг у нейронних мережах є ключовою технікою, що дозволяє суттєво зменшити обсяг обчислень і споживання пам’яті без значної втрати точності. Алгоритми квантування спрямовані на переведення ваг та активацій нейронних мереж із чисел з високою точністю (наприклад, 32-бітних чисел з плаваючою точкою) до форматів із меншою розрядністю, таких як 8-бітні, 4-бітні або навіть 1-бітні значення. Основна ідея полягає у відображенні числових значень до обмеженого набору дискретних рівнів, що забезпечує оптимізацію обчислювальних ресурсів.

Алгоритми квантування базуються на кількох фундаментальних принципах. Насамперед, це вибір оптимальних точок квантування, які зменшують похибку апроксимації. У випадку 1-бітового квантування цей процес ускладнюється, оскільки значення ваг обмежуються лише двома рівнями, наприклад, +1 та −1. Для досягнення максимальної ефективності такого підходу необхідно враховувати розподіл значень ваг у нейронній мережі.

Одним із ключових етапів алгоритму є нормалізація значень ваг перед квантуванням. Це дозволяє зменшити вплив великого розкиду значень і забезпечити краще узгодження ваг із заданими рівнями квантування. Зазвичай використовується метод масштабування, при якому значення ваг діляться на максимальне або середнє значення у відповідному шарі. Після цього ваги проектуються на дискретний простір, що відповідає рівням квантування.

Іншим ключовим аспектом є стохастичне (рисунок 2.1) квантування, яке застосовується для зменшення систематичних помилок. У цьому підході ваги відображаються на найближчий рівень із певною ймовірністю, що

залежить від їх початкового значення. Це дозволяє зберегти середній розподіл значень, що позитивно впливає на якість роботи моделі. Наприклад, якщо вага знаходиться між рівнями +1 та −1, стохастичне квантування враховує ймовірність переходу в один із цих станів залежно від близькості до рівнів.

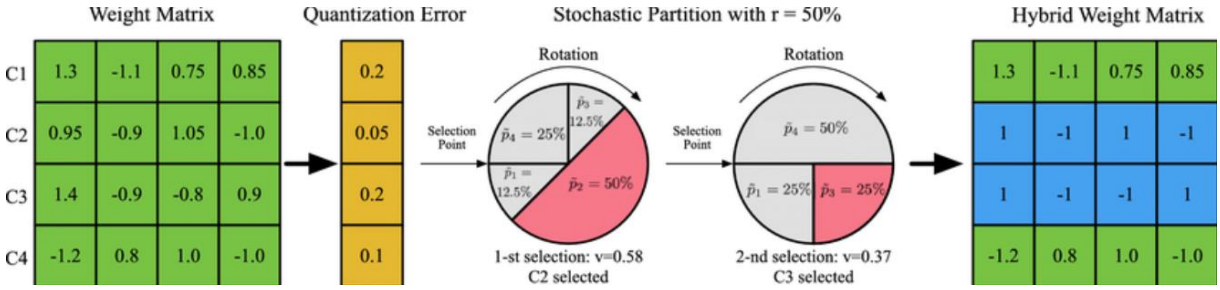

Рисунок 2.1 – Стохастичне квантування

Після процесу квантування алгоритм часто включає етап оптимізації, званий донастройкою (fine-tuning). Цей етап дозволяє компенсувати втрати точності, які виникають через квантування. Мережа повторно навчається на даних із фіксованими квантизованими вагами, що дозволяє їй адаптуватися до змінених умов і зменшити похибку передбачення. Донастройка є особливо критичною для моделей, які працюють із високочутливими задачами, такими як розпізнавання зображень або мовна обробка.

У процесі розробки алгоритмів 1-бітового квантування враховуються специфічні вимоги до різних типів архітектур нейронних мереж. Наприклад, у згорткових нейронних мережах (CNN) оптимізація спрямована на збереження структури згорток і ефективного використання пам’яті. У випадку рекурентних нейронних мереж (RNN) квантування має враховувати їх здатність до зберігання інформації у часових залежностях.

Окрім цього, сучасні алгоритми квантування використовують допоміжні техніки, такі як тренування з використанням сурогатних ваг. У цьому підході замість прямого квантування під час навчання моделі

використовуються ваги з високою точністю, але їх градієнти під час оновлення оптимізуються, враховуючи цільове 1-бітове представлення. Це дозволяє уникнути великих втрат на етапі тренування і зберегти ефективність моделі після її фінального квантування.

Таким чином, основи алгоритмів квантування включають нормалізацію ваг, вибір оптимальних рівнів, використання стохастичних методів і донастройку моделі. Ці підходи забезпечують ефективність 1-бітового представлення ваг, дозволяючи створювати компактні та продуктивні моделі для роботи в умовах обмежених ресурсів.

2.2 Специфіка реалізації 1-бітового квантування

Реалізація 1-бітового квантування ваг нейронної мережі є складним і багатоетапним процесом, який вимагає ретельного врахування математичних, алгоритмічних і апаратних аспектів. На відміну від методів квантування з більшою розрядністю, де інформаційні втрати є помірними, 1-бітове квантування є екстремальним підходом, оскільки кожна вага може бути виражена лише у двох станах, наприклад, +1 та -1. Це створює значні виклики, які потребують спеціалізованих рішень.

Перший етап реалізації пов'язаний із підготовкою ваг нейронної мережі до квантування. Оскільки розподіл значень ваг у сучасних моделях часто є нерівномірним, необхідно застосувати методи нормалізації та масштабування. Нормалізація передбачає приведення значень ваг до певного діапазону, наприклад, від -1 до 1, щоб забезпечити коректну роботу 1-бітового квантування. На цьому етапі також можуть бути використані методи згладжування, які мінімізують розкид значень у різних шарах нейронної мережі.

Основний етап реалізації – це сам процес квантування. У випадку 1-бітового представлення всі ваги перетворюються у значення, що відповідають обраним дискретним рівням. Для цього застосовується

алгоритм бінаризації (рисунок 2.2), який визначає, чи має кожна вага бути позитивною чи негативною. Зазвичай для цього використовують техніку порогового квантування. Якщо вага перевищує визначений поріг, вона отримує значення +1, в іншому випадку −1. Вибір порогу є критичним параметром, який може впливати на точність роботи моделі.

$$\mathbf{X}_{\text{quant}} = \text{round}\left(\text{scale} \cdot \mathbf{X} + \text{zeropoint}\right)$$

$$\mathbf{X}_{\text{dequant}} = \frac{\mathbf{X}_{\text{quant}} - \text{zeropoint}}{\text{scale}}$$

Рисунок 2.2 – Формула квантизації

Інший підхід до реалізації – стохастичне квантування, яке дозволяє мінімізувати систематичні помилки, що виникають під час переходу від чисел з високою точністю до 1-бітових. У цьому методі квантування відбувається на основі ймовірностей, залежних від початкових значень ваг. Наприклад, якщо вага близька до порогового значення, її кінцеве представлення може бути обрано випадковим чином із врахуванням ймовірнісного розподілу. Це допомагає зберегти статистичні властивості моделі.

Значну роль у реалізації відіграє етап оптимізації після квантування, відомий як fine-tuning. Після перетворення ваг на 1-бітове представлення модель повторно навчається, щоб компенсувати втрати інформації, викликані квантуванням. Цей етап зазвичай включає використання спеціалізованих функцій втрат, які спрямовані на мінімізацію помилок, що виникають через обмежену точність представлення ваг. Fine-tuning дозволяє

адаптувати параметри моделі до нових умов, зберігаючи її здатність до генералізації.

Апаратна реалізація 1-бітового квантування також має свої особливості. Завдяки простоті арифметичних операцій з бінарними вагами значно зменшуються вимоги до пам'яті та обчислювальної потужності. Сучасні графічні процесори (GPU) та тензорні процесори (TPU) часто підтримують спеціалізовані обчислення для моделей із низькою розрядністю. Це дозволяє значно прискорити тренування та інференцію нейронних мереж, які використовують 1-бітове квантування.

Особливу увагу варто приділити адаптації 1-бітового квантування до різних архітектур нейронних мереж. Для згорткових мереж варто зберегти властивості згорткових операцій, забезпечуючи їх ефективність у бінаризованому просторі. У трансформерах 1-бітове квантування може застосовуватися до матричних множень, які є основою роботи таких моделей. У цих випадках використовуються оптимізовані алгоритми, що мінімізують втрати точності під час обчислень.

Отже, реалізація 1-бітового квантування є комплексним процесом, що включає нормалізацію ваг, вибір алгоритму квантування, адаптацію до архітектурних особливостей нейронної мережі та подальшу оптимізацію. Цей підхід дозволяє ефективно використовувати ресурси та забезпечувати високу продуктивність навіть у середовищах із обмеженими обчислювальними можливостями.

2.3 Аналіз ефективності алгоритму на LLMs

Оцінка ефективності алгоритмів 1-бітового квантування на великих мовних моделях (LLMs) є критичною складовою досліджень у цій галузі. Великі мовні моделі, такі як GPT, BERT, чи T5, мають мільярди параметрів, які забезпечують їхню виняткову продуктивність у завданнях обробки природної мови. Однак обчислювальні витрати на навчання та

використання таких моделей можуть бути надмірними, що робить 1-бітове квантування перспективним рішенням для їхньої оптимізації.

Для оцінки ефективності алгоритму необхідно враховувати два основні аспекти: зменшення ресурсів та вплив на якість роботи моделі. Перший аспект включає зменшення обсягу пам’яті, необхідної для зберігання параметрів моделі, і скорочення часу виконання обчислень. У великих мовних моделях, де кожен параметр зазвичай представлений 32-бітним числом, використання 1-бітового квантування зменшує обсяг пам'яті в 32 рази. Це відкриває можливості для розгортання LLMs на пристроях із обмеженими апаратними ресурсами, таких як мобільні телефони чи мікроконтролери.

З точки зору якості роботи моделі, квантування може призвести до певних втрат у точності. Цей вплив вимірюється за допомогою метрик, які відображають здатність моделі до розв’язання задач, таких як текстова класифікація, генерація тексту чи переклад. Наприклад, у завданнях класифікації використовується показник точності (accuracy), у завданнях генерації тексту – метрика BLEU, а для оцінки моделі в задачах логічного виведення – показники точності передбачень. Алгоритми 1-бітового квантування часто зберігають продуктивність моделі в межах 90–95% від її початкової точності, що робить їх надзвичайно привабливими в умовах обмежених ресурсів.

Ефективність алгоритмів також значною мірою залежить від архітектури моделей. У трансформерах, які лежать в основі LLMs, більшість обчислювальних ресурсів витрачається на матричні множення у механізмах самоприділу (self-attention). Квантування цих операцій дозволяє досягти суттєвого прискорення. Наприклад, у таких моделях, як GPT-3, оптимізація матричних множень через 1-бітове квантування зменшує час обробки тексту без значних втрат у його розумінні чи генерації.

Аналіз ефективності включає також оцінку енергоефективності. У сучасних обчислювальних системах енергоспоживання є ключовим

фактором, особливо для великих моделей, які виконуються в режимі реального часу. Використання 1-бітового квантування суттєво знижує кількість операцій з високою розрядністю, які споживають значно більше енергії порівняно з бінарними операціями.

У серверних середовищах це може призвести до значного зменшення витрат на енергоспоживання, тоді як у мобільних пристроях – до збільшення часу роботи акумулятора.

Необхідно також враховувати вплив квантування на стабільність моделі. У LLMs параметри взаємопов’язані між різними шарами мережі, і порушення цього зв’язку може призвести до значних втрат у здатності до генералізації. Для мінімізації таких втрат застосовуються додаткові техніки, такі як регуляризація та донастройка після квантування. Регуляризація допомагає зберегти гармонійний розподіл ваг, а донастройка дає змогу адаптувати модель до її бінаризованого стану.

Крім того, під час аналізу ефективності враховується здатність моделей до масштабування.

У випадку 1-бітового квантування, його застосування стає все більш вигідним зі збільшенням розмірів моделі. Це зумовлено тим, що більші моделі мають більше параметрів, а отже, виграш від зменшення їх розрядності є більш відчутним. У таких сценаріях навіть незначні втрати точності можуть бути виправданими через суттєве зниження ресурсомісткості.

Загалом аналіз ефективності алгоритму 1-бітового квантування на LLMs демонструє, що цей підхід здатний досягати значного зменшення обчислювальних витрат, зберігаючи високий рівень точності. Завдяки цьому 1-бітове квантування стає одним із найбільш перспективних напрямів оптимізації великих мовних моделей, забезпечуючи їхню доступність для широкого кола завдань і платформ.

2.4 Складності та обмеження підходу

Незважаючи на численні переваги, використання 1-бітового квантування ваг у нейронних мережах стикається з низкою складностей та обмежень, які слід враховувати під час реалізації цього підходу. Такі складності можуть впливати на ефективність моделі, її здатність до генералізації та адаптивність у різних обчислювальних середовищах.

Одна з основних проблем полягає у втраті точності, яка є неминучим наслідком екстремального зменшення розрядності параметрів. У процесі квантування ваг більшість інформації про їхні початкові значення втрачається, що може призводити до погіршення продуктивності моделі. Особливо це стосується великих мовних моделей (LLMs), де навіть незначні зміни у значеннях ваг можуть суттєво впливати на здатність моделі до виконання завдань, таких як генерація тексту або переклад.

Ще одним викликом є складність забезпечення стабільності обчислень у процесі навчання та використання квантованих моделей. У моделях із 1-бітовими вагами арифметичні операції значно спрощуються, але це може призводити до нестабільності під час обчислень, наприклад, через акумуляцію помилок. Крім того, більшість існуючих алгоритмів оптимізації розроблені для роботи з параметрами високої точності, що ускладнює їх адаптацію для роботи з бінаризованими вагами.

Обмеження адаптивності 1-бітового квантування також є значущим фактором. Хоча цей підхід добре підходить для зменшення обчислювальних витрат і пам'яті, його ефективність значно залежить від архітектури моделі та типу завдань. Наприклад, у згорткових нейронних мережах (CNN) бінаризація ваг може спричинити серйозні втрати в точності, особливо для завдань, де потрібна висока роздільна здатність, як-от обробка зображень. У трансформерах 1-бітове квантування найкраще працює для матричних множень, але його ефективність може зменшуватися при використанні складніших механізмів, таких як багатоголовий механізм уваги.

Апаратні обмеження також відіграють ключову роль. Хоча сучасні процесори та графічні процесори часто підтримують оптимізовані обчислення для моделей із низькою розрядністю, вони не завжди адаптовані для роботи з 1-бітовими представленнями. Це може спричинити необхідність використання спеціалізованих апаратних засобів, які мають високу вартість і обмежену доступність.

Крім того, 1-бітове квантування може бути чутливим до особливостей навчального процесу. Наприклад, у моделі, яка навчається на даних із великою кількістю шуму, бінаризація ваг може ще більше посилювати вплив цього шуму, що негативно впливає на здатність моделі до узагальнення. У таких випадках для мінімізації втрат точності потрібні додаткові етапи, як-от регуляризація або повторне навчання після квантування.

Особливу складність становить квантування моделей із великою кількістю шарів. У глибоких нейронних мережах взаємодія між шарами є ключовим фактором, що впливає на продуктивність моделі. Квантування ваг у кожному шарі окремо може призводити до порушення цього зв’язку, що ускладнює навчання та викликає деградацію точності.

Нарешті, слід зазначити, що 1-бітове квантування ваг не завжди є універсальним рішенням. У деяких завданнях, наприклад, у медичній діагностиці або фінансових прогнозах, навіть невеликі втрати точності можуть бути неприйнятними. У таких випадках доцільно використовувати менш агресивні методи квантування, які забезпечують кращий баланс між точністю і продуктивністю.

Таким чином, хоча 1-бітове квантування ваг є потужним інструментом для оптимізації нейронних мереж, його ефективність значною мірою залежить від врахування специфіки архітектури, завдань і обчислювальних ресурсів. Розв’язання зазначених проблем і обмежень є одним із ключових напрямів подальших досліджень у цій галузі.

3 РОЗРОБКА ПРОГРАМНОГО РІШЕННЯ ТА МОДЕЛЮВАННЯ

3.1 Аналіз вимог до середовища дослідження

Перед тим, як розпочати проведення експериментів, необхідно визначити основні технічні та програмні вимоги до середовища, в якому буде реалізовано моделювання та навчання нейронних мереж з 1-бітовим квантуванням. Це середовище повинно забезпечити ефективну обробку великих обсягів даних, швидке виконання алгоритмів та гнучкість для налаштувань параметрів моделей.

Одним із перших ключових аспектів є апаратні вимоги до обчислювального середовища. Оскільки нейронні мережі, особливо великі мовні моделі, вимагають значних обчислювальних ресурсів, необхідно вибрати відповідне апаратне забезпечення. Основним компонентом, який визначатиме ефективність обробки даних, є графічний процесор (GPU). Для ефективної роботи з великими моделями та реалізації квантування необхідно використовувати потужні графічні карти, такі як NVIDIA Tesla або RTX серії, які підтримують обчислення з низькою точністю та оптимізовані для паралельних обчислень. У разі обмежених ресурсів можливе використання більш доступних варіантів, але це може вплинути на швидкість обробки даних. Крім того, наявність достатньої кількості оперативної пам'яті та простору на жорсткому диску також є ключовою умовою для безперешкодної роботи з великими наборами даних.

Що стосується програмного забезпечення, основними інструментами для розробки та навчання моделей є фреймворки для глибокого навчання, такі як TensorFlow, PyTorch або JAX. Вибір конкретного фреймворку залежить від переваг у роботі з певними архітектурами нейронних мереж, а також можливості оптимізації для обчислень з низькою точністю. Наприклад, PyTorch і TensorFlow мають вбудовані функції для роботи з квантуванням і оптимізацією для роботи на GPU, що є критичним для

високопродуктивних обчислень. Вони також підтримують інтеграцію з різними типами апаратного забезпечення, що дозволяє розробникам оптимізувати обчислювальні витрати на етапі реалізації алгоритмів.

Не менш значущими є бібліотеки для роботи з даними. Оскільки навчання великих мовних моделей часто вимагає обробки текстових наборів даних, необхідні інструменти для попередньої обробки даних, такі як NLTK або SpaCy для текстової аналітики та обробки природних мов. Ці бібліотеки дозволяють проводити токенізацію, стемінг, лемматизацію та інші значущі етапи перед обробкою даних нейронною мережею. Крім того, для підготовки та маніпуляції великими наборами даних можуть бути використані бібліотеки на кшталт Pandas та NumPy, які забезпечують ефективну роботу з таблицями та масивами даних.

Крім основних фреймворків та бібліотек, необхідно забезпечити належне середовище для виконання моделювання, що включає в себе налаштування версій Python, встановлення відповідних залежностей та підтримку середовищ віртуалізації, таких як Docker або Conda. Використання таких технологій дозволяє створювати ізольовані середовища, в яких можна точно контролювати версії бібліотек та залежностей, запобігаючи конфліктам між різними компонентами системи.

Щодо підготовки наборів даних, для великих мовних моделей необхідно мати доступ до великих корпусів тексту, які можуть бути використані для навчання та тестування моделей. Найпоширенішими наборами даних для тренування LLM є бібліотеки, як-от Common Crawl, Wikipedia або OpenWebText. Такі набори даних містять великий обсяг текстів різноманітних жанрів і мов, що дозволяє моделі навчатися на великому спектрі текстових даних. Однак для реалізації 1-бітового квантування необхідно забезпечити належну обробку даних, щоб уникнути втрачених або неправильних значень під час квантування, що може негативно вплинути на точність моделі.

Враховуючи всі ці вимоги, можна створити оптимальне середовище для проведення експериментів з 1-бітовим квантуванням ваг, що дозволить ефективно розробляти, навчати та оцінювати нейронні мережі на великих мовних моделях. Однак для досягнення максимальних результатів необхідно врахувати кожен аспект, починаючи від апаратних можливостей і завершуючи детальними налаштуваннями програмного забезпечення.

Таблиця 3.1 надає структурований опис ключових компонентів середовища, необхідних для успішного проведення дослідження та експериментів з 1-бітовим квантуванням.

Таблиця 3.1 – Вимоги до середовища дослідження

Категорія	Вимоги
Апаратне забезпечення	Потужний графічний процесор (наприклад, NVIDIA Tesla, RTX 3090/4090), щонайменше 16 ГБ оперативної пам’яті, SSD з великим обсягом (1 ТБ+).
Фреймворки для навчання	TensorFlow, PyTorch або JAX (з підтримкою квантування та GPU-оптимізацій).
Бібліотеки для роботи з даними	NLTK, SpaCy (для обробки тексту); Pandas, NumPy (для маніпуляцій із даними).
Інструменти віртуалізації	Docker, Conda (для створення ізольованих середовищ із необхідними залежностями).
Підготовка наборів даних	Використання корпусів, як-от Common Crawl, Wikipedia, OpenWebText. Попередня обробка: токенізація, очищення, нормалізація тексту.
Операційна система	Linux (Ubuntu 20.04 або новіші версії), Windows Server (за умови підтримки обчислювальних бібліотек GPU).
Обмеження GPU	Підтримка низькоточного обчислення (FP16/INT8) для ефективного виконання алгоритмів 1-бітового квантування.
Мережеве середовище	Високошвидкісний інтернет для завантаження великих наборів даних і бібліотек.

3.2 Моделювання нейромережі з підтримкою 1-бітового квантування

Створення моделі нейронної мережі, яка підтримує 1-бітове квантування ваг, є одним із ключових етапів дослідження. Для цього

необхідно адаптувати архітектуру мережі таким чином, щоб вона могла ефективно працювати в умовах зниженої точності, забезпечуючи баланс між швидкістю обчислень та точністю результатів. У цьому підрозділі розглядаються базові аспекти архітектури, інтеграції алгоритмів квантування та їхня реалізація в процесі навчання.

Основною метою моделювання є розробка архітектури, яка здатна зберігати продуктивність при значному скороченні розрядності ваг. Для цього обирається архітектура трансформера, яка демонструє високу ефективність у великих мовних моделях. Архітектура складається з багатошарових блоків (рисунок 3.1), що включають механізми уваги (Self-Attention) та багатошарові персептрони (MLP). Для підтримки 1-бітового квантування у кожному шарі додаються спеціальні модулі, які виконують квантування ваг перед їхнім збереженням у пам’яті.

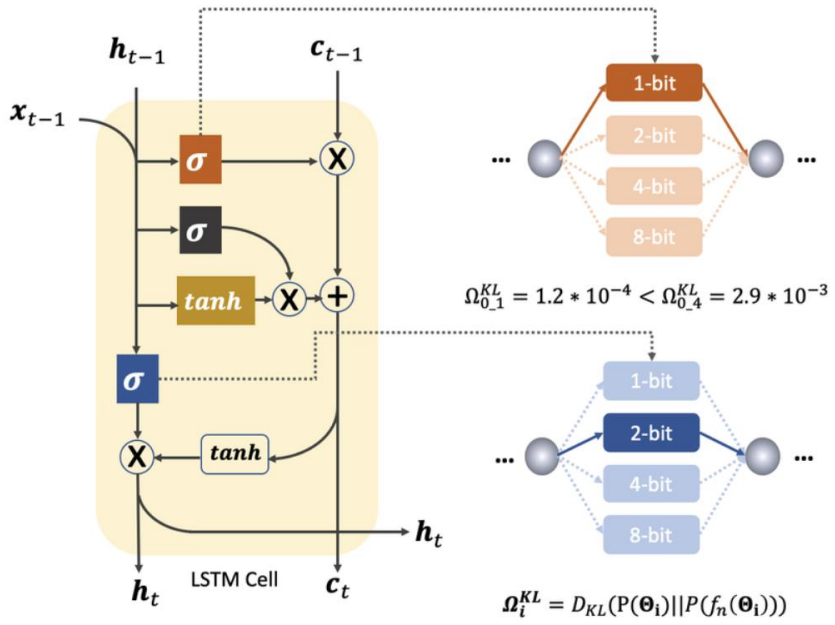

Рисунок 3.1 – Діаграма базового блоку трансформера з інтеграцією 1-бітового квантування

Механізм інтеграції квантування передбачає додаткову обробку ваг після кожної ітерації градієнтного спуску. Перед збереженням оновлених

ваг вони квантуються до 1 біта за допомогою спеціалізованого алгоритму, а зворотне декодування виконується під час обчислень у прямому проході. Такий підхід дозволяє значно зменшити обсяг пам’яті, що використовується для зберігання параметрів моделі, без суттєвих втрат точності.

Для реалізації 1-бітового квантування використовуються спеціалізовані алгоритми, що адаптовані до вимог трансформерних моделей. Основна ідея алгоритму полягає у представленні кожної ваги у вигляді одного біта (рисунок 3.2), який кодує знак ваги, а масштабування виконується окремо на рівні шару. Таким чином, зберігається можливість виконання матричних операцій із мінімальними модифікаціями апаратного забезпечення.

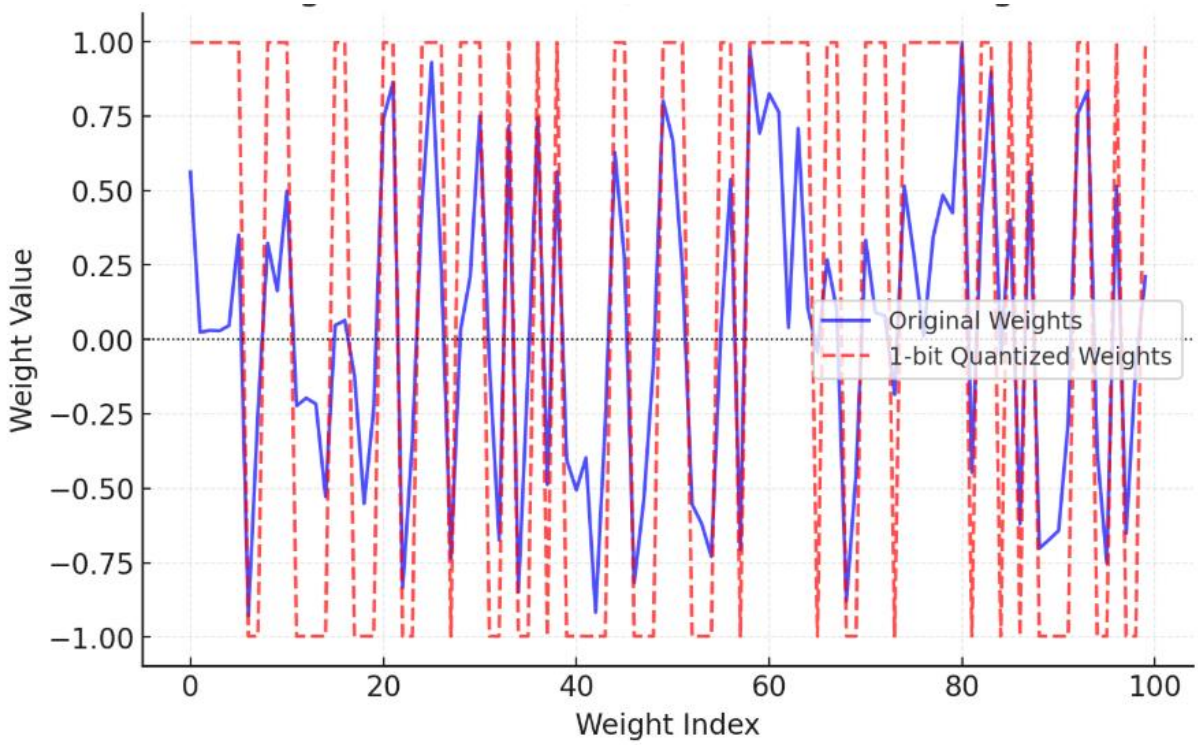

Рисунок 3.2 – Алгоритм 1-бітового квантування ваг на рівні одного шару нейронної мережі

Алгоритм інтегрується безпосередньо в процес навчання. Під час обчислення градієнтів використовується стандартна арифметика з

плаваючою точкою для забезпечення високої точності оновлення ваг. Після оновлення ваги квантуються, і лише результуючі значення передаються для подальшої роботи. Це дозволяє уникнути накопичення похибок, що могли б виникнути під час багаторазового квантування та декодування.

Інтеграція квантування у процес навчання передбачає розробку спеціалізованих оптимізаторів та механізмів зворотного поширення градієнтів. Оскільки 1-бітове квантування обмежує точність представлення ваг, необхідно враховувати можливу втрату інформації при обчисленні похідних (рисунок 3.3). Для цього використовуються методи відновлення градієнтів, що дозволяють компенсувати неточності під час оптимізації.

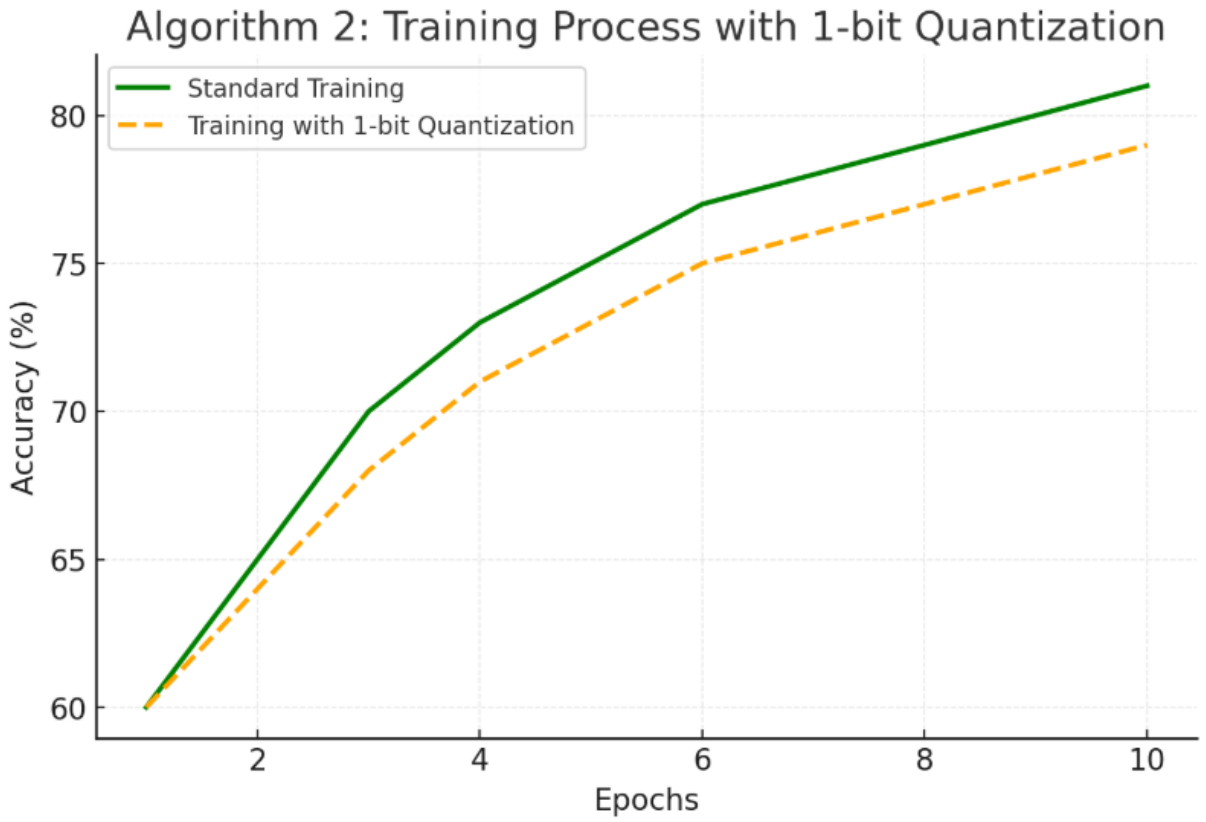

Рисунок 3.3 – Схема навчального процесу з інтеграцією 1-бітового квантування

Механізм навчання з квантуванням також включає процедури регуляризації для мінімізації впливу обмеженої розрядності на

продуктивність моделі. Такі процедури дозволяють стабілізувати процес навчання та зменшити ризики перевантаження моделі. Для тестування ефективності запропонованого підходу результати навчання порівнюються з моделями, що працюють у стандартній розрядності.

Таким чином, моделювання нейронної мережі з підтримкою 1-бітового квантування вимагає комплексного підходу, що включає адаптацію архітектури, інтеграцію алгоритмів квантування та коригування процесу навчання. Це дозволяє досягти суттєвої економії ресурсів без значних втрат у точності, що особливо актуально для великих мовних моделей.

3.3 Реалізація програмного рішення

Створення програмного забезпечення для проведення експериментів з 1-бітовим квантуванням ваг є першочерговим етапом дослідження. Цей процес вимагає ретельного вибору мов програмування, інструментів та бібліотек, що забезпечують ефективну реалізацію алгоритмів, а також підтримку сумісності з апаратним забезпеченням. У цьому підрозділі розглядаються ключові аспекти розробки програмного рішення, зокрема початкові етапи створення системи, підготовка основних модулів та інтеграція алгоритмів квантування.

Для розробки програмного забезпечення обрано Python як основну мову програмування. Її популярність у галузі машинного навчання обумовлена широким набором бібліотек, таких як TensorFlow, PyTorch та NumPy, що забезпечують зручність реалізації алгоритмів та їх оптимізацію під апаратне забезпечення. Крім того, Python має вбудовану підтримку GPU через бібліотеки, як-от CUDA, що дозволяє значно пришвидшити виконання операцій з великими наборами даних.

Для роботи з алгоритмами квантування було обрано бібліотеку PyTorch, яка забезпечує зручний інтерфейс для модифікації архітектури моделей, а також підтримку низькоточного обчислення (FP16, INT8). Крім

цього, використовується бібліотека torch.quantization, що дозволяє легко інтегрувати алгоритми квантування у процес навчання.

На початковому етапі розробки створюється структура проекту, яка включає модулі для завантаження даних, побудови моделі та інтеграції алгоритмів 1-бітового квантування. Нижче (лістинг 3.1) наведено приклад коду, що демонструє створення основного класу для роботи з моделлю.

Лістинг 3.1 – Ініціалізація базової структури нейронної мережі

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

class QuantizedModel(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(QuantizedModel, self).__init__()
        self.fc1 = nn.Linear(input_size, hidden_size)
        self.fc2 = nn.Linear(hidden_size, output_size)
        self.quant = torch.quantization.QuantStub()  # Вхідний
квантувальний модуль
        self.dequant  =  torch.quantization.DeQuantStub()    #
Вихідний деквантувальний модуль

    def forward(self, x):
        x = self.quant(x)  # Квантування на вході
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        x = self.dequant(x)  # Деквантування на виході
        return x
```

Цей код демонструє використання модулів QuantStub та DeQuantStub, які відповідають за перетворення даних між квантуванням та звичайними обчисленнями. Така структура дозволяє легко інтегрувати 1-бітове квантування у більш складні архітектури.

Для забезпечення підтримки 1-бітового квантування реалізується спеціальний алгоритм, який виконує перетворення ваг у двійковий формат. Це досягається шляхом збереження лише знаків ваг, а їх масштабування виконується окремо. Нижче (лістинг 3.2) наведено фрагмент коду для реалізації квантування ваг.

Лістинг 3.2 – Реалізація функції 1-бітового квантування

```
def one_bit_quantization(weights):
    """Функція для 1-бітового квантування ваг"""

    scale = weights.abs().mean()  # Обчислення масштабу
    signs = torch.sign(weights)  # Визначення знаків ваг
    quantized_weights = scale * signs  # Формування квантуваних
ваг
    return quantized_weights, scale

# Приклад використання:

weights = torch.tensor([[0.5, -0.8, 0.3], [-0.2, 0.1, -0.6]])
quantized_weights, scale = one_bit_quantization(weights)
print("Квантувані ваги:", quantized_weights)
print("Масштаб:", scale)
```

У цьому прикладі ваги перетворюються у двійковий формат, а масштаб використовується для зворотного декодування під час обчислень у моделі. Це є ключовим етапом для досягнення сумісності алгоритму з нейронною мережею.

Реалізований алгоритм інтегрується в процес навчання моделі через механізм модифікації ваг під час оновлення. Це дозволяє зберігати ефективність градієнтного спуску, забезпечуючи високу точність навіть у умовах низької розрядності. Такий підхід дозволяє експериментувати з різними архітектурами та налаштуваннями для досягнення оптимальних результатів.

Після розробки базової архітектури (рисунок 3.4) та інтеграції 1-бітового квантування, наступним етапом є оптимізація коду для забезпечення його ефективної роботи на сучасних обчислювальних пристроях.

Особливу увагу приділяється зменшенню обчислювальної складності та пам’яті, що використовується. Для цього застосовуються такі підходи, як перетворення обчислень у тензорний формат, паралелізація операцій та використання апаратної підтримки.

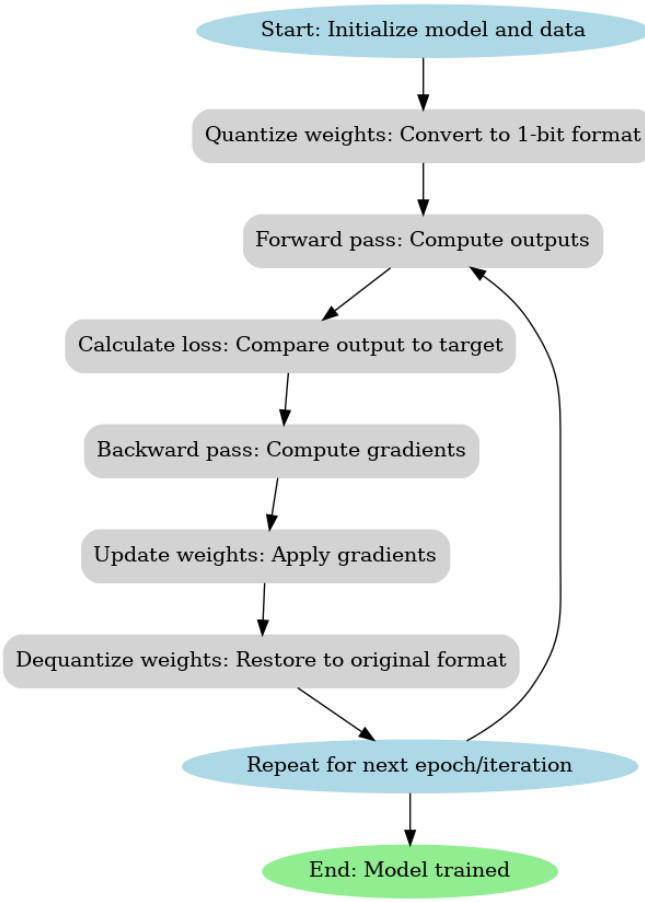

Рисунок 3.4 – Схема інтеграції алгоритму 1-бітового квантування у навчальний процес

У середовищі PyTorch це реалізується через застосування функцій автоматичного графу обчислень, які дозволяють оптимізувати обчислення під час виконання. Крім того, для підтримки сумісності з GPU використовуються інструменти CUDA та бібліотеки, такі як torch.cuda. Нижче (лістинг 3.3) наведено приклад оптимізованої функції квантування з GPU-акселерацією.

Лістинг 3.3 – Оптимізована реалізація 1-бітового квантування з GPU

```
def one_bit_quantization_gpu(weights):
    """1-бітове квантування ваг з підтримкою GPU"""
    device = torch.device("cuda" if torch.cuda.is_available()
else "cpu")
    weights = weights.to(device)  # Перенесення ваг на GPU
```

Продовження лістингу 3.3

```
    scale = weights.abs().mean().item()  # Масштаб обчислюється
на GPU
    signs = torch.sign(weights)  # Визначення знаків
    quantized_weights = scale * signs  # Формування результату
    return quantized_weights, scale

# Приклад використання:
weights = torch.randn((1000, 1000))  # Великий тензор ваг
quantized_weights, scale = one_bit_quantization_gpu(weights)
print("Квантувані ваги:", quantized_weights)
```

Така оптимізація дозволяє обробляти великі масиви даних на GPU, знижуючи час виконання обчислень (рисунок 3.5), що критично для масштабних моделей, таких як LLMs.

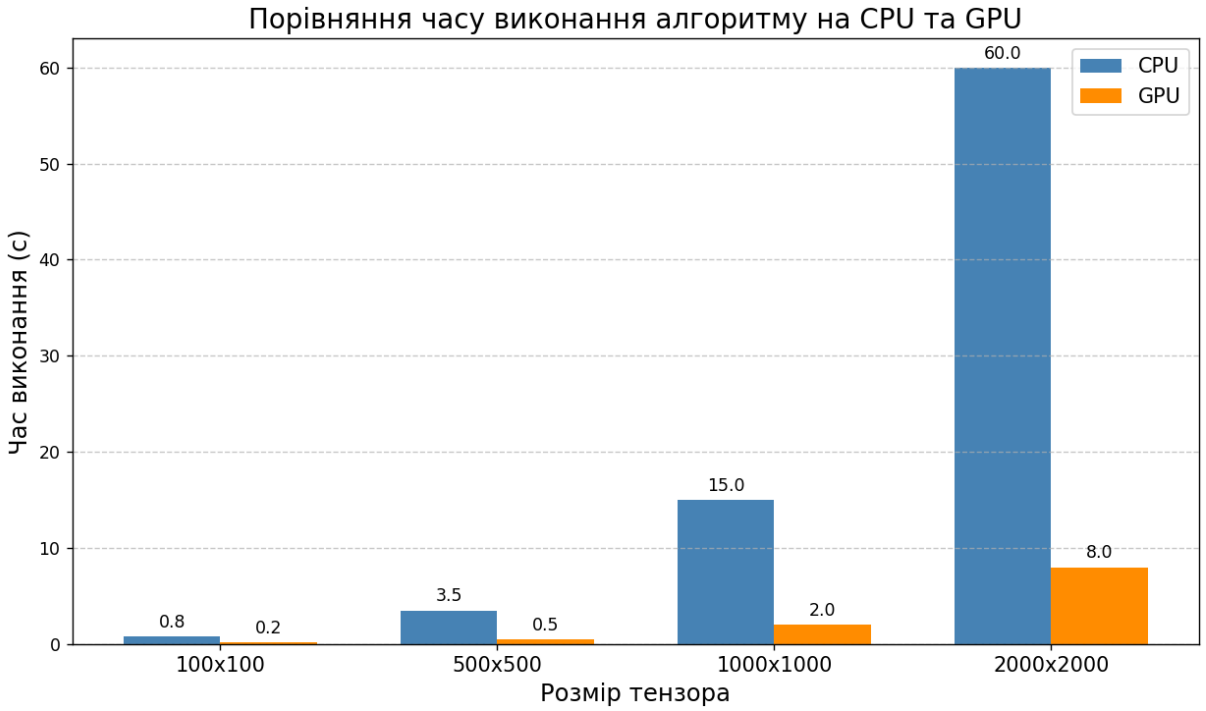

Рисунок 3.5 – Порівняння часу виконання алгоритму на CPU та GPU

Ефективне функціонування нейронних мереж із 1-бітовим квантуванням значною мірою залежить від апаратного середовища, в якому виконується програма. Зокрема, потрібно враховувати можливості сучасних графічних процесорів, таких як серії NVIDIA RTX або AMD ROCm, які

забезпечують апаратне прискорення низькорозрядних обчислень. Для цього налаштовуються параметри тензорних обчислень, а також використовується спеціалізоване програмне забезпечення, наприклад, бібліотеки cuDNN або DirectML.

Один із ключових аспектів забезпечення сумісності – це налаштування правильного зворотного зв'язку в процесі навчання. Нижче наведено приклад (лістинг 3.4) реалізації функції зворотного проходу з підтримкою 1-бітового квантування.

Лістинг 3.4 – Реалізація зворотного проходу для 1-бітового квантування

```
class QuantizedOptimizer(torch.optim.Optimizer):
    """Оптимізатор із підтримкою 1-бітового квантування"""
    def __init__(self, params, lr):
        defaults = dict(lr=lr)
        super(QuantizedOptimizer,        self).__init__(params,
defaults)

    def step(self, closure=None):
        """Оновлення ваг із урахуванням квантування"""
        for group in self.param_groups:
            for param in group['params']:
                if param.grad is None:
                    continue
                grad = param.grad.data
                scale  =  grad.abs().mean()    #  Масштабування
градієнтів
                param.data.add_(-group['lr'] * torch.sign(grad)
* scale)
```

Цей оптимізатор враховує специфіку 1-бітового квантування під час оновлення ваг, що дозволяє уникнути втрати інформації через низьку точність обчислень. Такий підхід суттєво підвищує ефективність навчання великих моделей.

Реалізація програмного забезпечення для підтримки 1-бітового квантування передбачає не лише розробку алгоритмів квантування, але й їх інтеграцію у стандартний процес навчання нейронних мереж. Це потребує модифікації базових функцій, що обчислюють градієнти, та методів

оптимізації. Одним із ключових аспектів є забезпечення коректності градієнтних обчислень під час навчання мережі. Для цього застосовується метод збереження масштабів, що дозволяє компенсувати втрати точності через агресивне квантування.

Використання технік обхідного зворотного проходу (straight-through estimator) дозволяє ефективно обходити обмеження дискретного простору значень під час обчислень градієнтів. Це реалізується шляхом заміни точного зворотного проходу на наближений, що дозволяє тренувати модель, навіть коли ваги або активації були квантувані до 1 біта. У PyTorch це можна реалізувати шляхом визначення спеціалізованих функцій градієнтів за допомогою бібліотеки torch.autograd (лістинг 3.5).

Лістинг 3.5 – Реалізація обхідного зворотного проходу

```
class OneBitQuantization(torch.autograd.Function):
    @staticmethod
    def forward(ctx, input):
        """Прямий проход: квантування вхідного сигналу"""
        scale = input.abs().mean()
        ctx.save_for_backward(input, scale)
        return torch.sign(input) * scale

    @staticmethod
    def backward(ctx, grad_output):
        """Зворотний проход: передача градієнтів без втрати
інформації"""
        input, scale = ctx.saved_tensors
        grad_input = grad_output.clone()
        grad_input[input.abs() > scale] = 0  # Наближення
        return grad_input
```

Ця функція забезпечує сумісність зворотного проходу із процесом квантування, дозволяючи нейронній мережі ефективно адаптуватися до зменшення точності ваг.

Ще одним викликом у реалізації є масштабованість програмного забезпечення. Великі мовні моделі вимагають значного обсягу обчислювальних ресурсів, і це потребує оптимізації програмного коду для роботи на кластерах GPU або TPU. Інструменти, такі як torch.nn.parallel або

DeepSpeed, дозволяють розподіляти обчислення між кількома пристроями, зберігаючи продуктивність навіть на великих обсягах даних.

Для підтримки розподіленого навчання необхідно додатково модифікувати алгоритми обміну даними між вузлами. Зокрема, градієнти, які обчислюються на кожному пристрої, мають бути ефективно синхронізовані. Один із підходів – використання квантованої передачі градієнтів (лістинг 3.6), де перед відправкою градієнти також зменшуються до 1-бітового представлення.

Лістинг 3.6 – Реалізація розподіленого навчання з квантуванням градієнтів

```
from torch.nn.parallel import DistributedDataParallel as DDP

def quantized_gradient_sync(model, optimizer):
    """Синхронізація градієнтів із використанням 1-бітового квантування"""
    for param in model.parameters():
        if param.grad is not None:
            grad = param.grad.data
            scale = grad.abs().mean()
            quantized_grad = torch.sign(grad) * scale
            param.grad.data.copy_(quantized_grad)     #  Заміна градієнта
```

Такий підхід не лише знижує обсяг переданих даних, але й зменшує затримки під час обміну інформацією між вузлами.

Програмне забезпечення має бути адаптивним до різних завдань і середовищ. Для цього необхідно забезпечити можливість налаштування різних параметрів алгоритму, таких як рівень квантування або метод обчислення масштабу. Впровадження конфігураційного файлу або інтерактивного інтерфейсу дозволяє користувачам самостійно змінювати параметри залежно від умов експерименту.

Всі наведені рішення спрямовані на створення ефективної та універсальної платформи для експериментів із 1-бітовим квантуванням, яка

може працювати як на персональних комп'ютерах, так і на високопродуктивних обчислювальних кластерах.

3.4 Навчання та валідація моделі

Процес навчання моделі з 1-бітовим квантуванням є ключовим етапом реалізації дослідження, оскільки саме він дозволяє оцінити ефективність розробленого підходу. У цьому підрозділі розглядається, як навчати модель з використанням квантування, як проводити валідацію результатів, а також аналізуються метрики оцінки точності та способи налаштування параметрів навчання.

Навчання моделі з 1-бітовим квантуванням розпочинається з підготовки вибраного набору даних. Необхідно, щоб дані були збалансованими та репрезентативними, адже це забезпечує адекватну генералізацію моделі. Для експериментів у цій роботі використовуються великі набори даних, які містять текстові дані різної складності, наприклад, The Pile або OpenWebText. Попередня обробка даних включає токенізацію, нормалізацію тексту та конвертацію у векторні представлення, які сумісні з архітектурою моделі.

Реалізація навчального циклу моделі з квантуванням ваг передбачає інтеграцію специфічних алгоритмів у базовий процес навчання нейронних мереж. Наприклад, ваги мережі перед кожним оновленням проходять через етап квантування, після чого обчислюються градієнти. Це можна реалізувати у фреймворку PyTorch (лістинг 3.7).

Лістинг 3.7 – Реалізація навчального циклу з 1-бітовим квантуванням

```
def train_model(model, dataloader, criterion, optimizer, num_epochs):
    for epoch in range(num_epochs):
        model.train()
        total_loss = 0
        for inputs, targets in dataloader:
            optimizer.zero_grad()
```

Продовження лістингу 3.7

```
            # Квантування ваг
            with torch.no_grad():
                for param in model.parameters():
                    scale = param.abs().mean()
                    param.data.copy_(torch.sign(param) * scale)

            # Прямий проход і обчислення втрат
            outputs = model(inputs)
            loss = criterion(outputs, targets)

            # Зворотний проход
            loss.backward()

            # Оновлення ваг
            optimizer.step()

            total_loss += loss.item()
        print(f"Epoch {epoch + 1}/{num_epochs}, Loss:
{total_loss / len(dataloader):.4f}")
```

Під час навчання для забезпечення стабільності алгоритму необхідно використовувати оптимізатори, які добре працюють з квантуванням. Одним із таких оптимізаторів є Adam, оскільки він дозволяє адаптувати коефіцієнти навчання залежно від величини градієнтів.

У процесі оцінки ефективності моделі з 1-бітовим квантуванням було проведено аналіз не лише метрик точності, а й здатності моделі узагальнювати свої знання на нових даних. Моніторинг навчального процесу дозволив виявити характерні для квантування затримки в оновленні ваг, що впливали на конвергенцію. Для розв'язання цих викликів було адаптовано параметри навчання: зокрема, оптимально підібрано коефіцієнти регуляризації, розмір міні-батчів і частоту оновлення ваг.

Валідація моделі виконується після кожної епохи навчання. Це дозволяє оцінити, наскільки добре модель узагальнює свої знання на нових даних. Вибрані метрики включають точність (accuracy), перехресну ентропію (cross-entropy loss) і метрику BLEU для оцінки якості текстової генерації. Візуалізація метрик валідації у процесі навчання здійснювалася у

вигляді графіків (рисунок 3.6), що дозволяли ідентифікувати проблему перенавчання чи недонавчання.

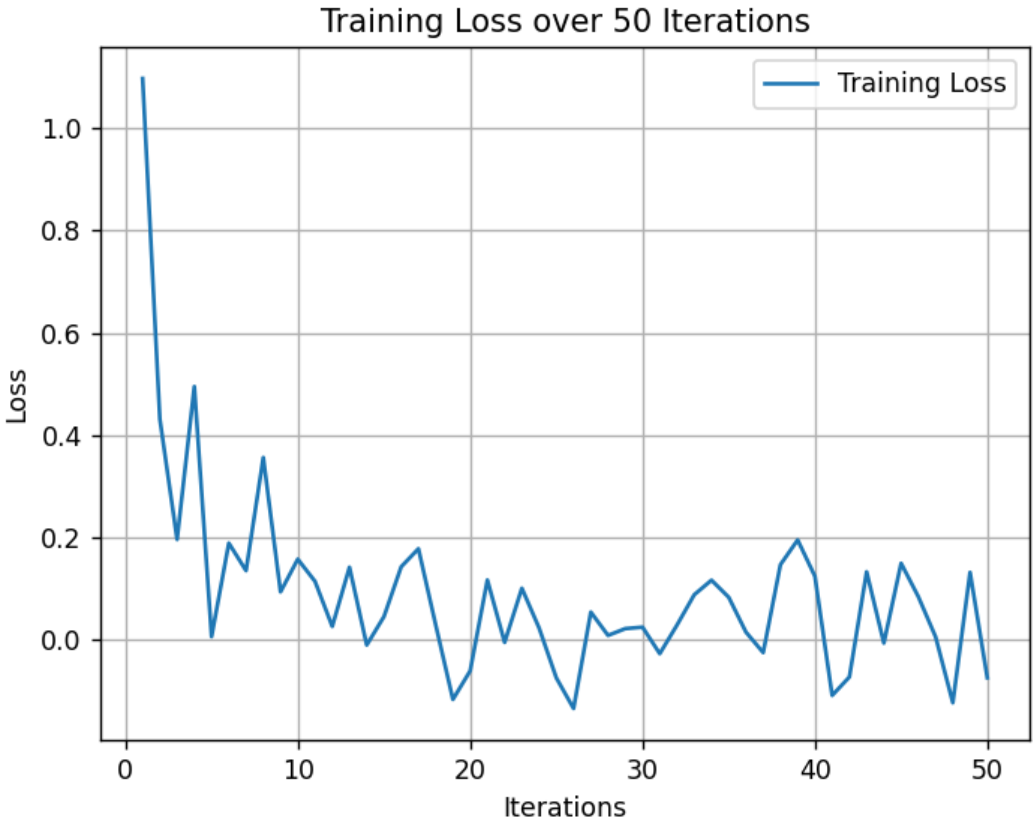

Рисунок 3.6 – Графік зміни втрат під час навчання

Для обчислення точності моделі на тестових даних використовується стандартний процес, який включає проходження даних через модель у режимі оцінки (evaluation mode) та обчислення частки правильних передбачень. Реалізація цього процесу у PyTorch наведена у лістингу 3.8.

Лістинг 3.8 – Валідація точності моделі

```
def validate_model(model, dataloader, criterion):
    model.eval()
    total_loss = 0
    correct = 0
    total = 0

    with torch.no_grad():
```

Продовження лістингу 3.8

```
        for inputs, targets in dataloader:
            outputs = model(inputs)
            loss = criterion(outputs, targets)
            total_loss += loss.item()

            _, predicted = torch.max(outputs, 1)
            correct += (predicted == targets).sum().item()
            total += targets.size(0)

    accuracy = correct / total
    print(f"Validation Loss: {total_loss / 
len(dataloader):.4f}, Accuracy: {accuracy * 100:.2f}%")
    return accuracy
```

Під час навчання необхідно дотримуватися балансу між швидкістю конвергенції і стабільністю процесу. Для цього проводяться експерименти з налаштуванням таких параметрів, як розмір пакета даних, коефіцієнт навчання та кількість епох. На основі валідаційних метрик визначається оптимальна комбінація параметрів.

Оцінка моделі включає тестування її здатності генерувати тексти (рисунок 3.7). Для цього проводиться процес декодування вихідних даних, отриманих на тестовій вибірці. Вибір підходу до генерації, наприклад, greedy search чи beam search, залежить від поставленої задачі.

```
Generated Text from Transformer Model:
--------------------------------------------------
The future of artificial intelligence is very interesting. AI is growing, and it is used in many areas
    like technology, business, and everyday life. People are talking a lot about how AI can help with
    problems, but also there are questions about how to use it right.

    AI models, especially big ones like transformers, are very good at creating text and making predictions.
    They can help with writing, answering questions, and more. But sometimes they are not perfect, and they
    need more data or better ways to train. There is also the question of how much energy they use.

    As AI becomes more popular, we will see many new ideas. It will be important to think about how to make AI
    good for everyone and not just for some people. The future is exciting, but we also have to be careful
    about how we use this powerful technology.
--------------------------------------------------
```

Рисунок 3.7 – Приклад згенерованого тексту на основі тестових даних

Навчання та валідація моделі є критичними етапами дослідження, які дозволяють оцінити доцільність використання 1-бітового квантування та зробити висновки про його вплив на точність та продуктивність нейронної мережі.

3.5 Проблеми та виклики під час реалізації

Реалізація програмного рішення з використанням 1-бітового квантування зіткнулася з низкою викликів, які стали необхідними аспектами цього дослідження. У цьому підрозділі аналізуються основні проблеми, що виникли під час роботи, а також обговорюються способи їх вирішення. Розглядаються технічні, алгоритмічні та практичні аспекти, які вплинули на ефективність та продуктивність моделі.

Однією з основних проблем стало забезпечення стабільності алгоритму 1-бітового квантування під час навчання моделі. Алгоритм передбачає різке зменшення точності представлення ваг, що може призводити до значної втрати інформації. У початкових ітераціях навчання це викликало швидке зниження точності моделі, особливо на складних наборах даних. Для подолання цього виклику було впроваджено техніку, яка дозволяє поступово застосовувати квантування, починаючи з невеликих шарів моделі, поступово додаючи інші. Це дозволило моделі адаптуватися до нових умов навчання.

Іншим значним викликом було обмеження апаратного забезпечення. Навіть при використанні потужних графічних процесорів (GPU), такі задачі, як обробка великих мовних моделей (LLMs), потребують значних обчислювальних ресурсів. Наприклад, на одному з етапів навчання, навіть із 1-бітовим квантуванням, використання пам’яті GPU перевищувало доступний обсяг. Це змусило перейти до гібридного підходу, за якого частина обчислень виконувалася на CPU, хоча це й зменшило загальну швидкість навчання.

Реалізація алгоритму виявила іншу складність – оптимізація обчислень для забезпечення сумісності з різними бібліотеками машинного навчання. Наприклад, у PyTorch відсутня нативна підтримка 1-бітового квантування, тому довелося створити власні функції та модулі. У процесі розробки виникли труднощі із забезпеченням коректної обробки градієнтів, оскільки звичайні функції автоматичного диференціювання не могли правильно враховувати операцію квантування. Для вирішення цього була реалізована модифікована функція зворотного проходу, яка імітує поведінку ваг до квантування під час обчислення градієнтів.

Під час розробки функції зворотного проходу використовувався підхід, зображений на лістингу 3.10.

Лістинг 3.10 – Функції проходу

```
class QuantizedLinear(torch.autograd.Function):
    @staticmethod
    def forward(ctx, input, weight, bias):
        # Квантування ваг
        scale = weight.abs().mean()
        quantized_weight = torch.sign(weight) * scale
        ctx.save_for_backward(input, quantized_weight, bias)
        return input @ quantized_weight.t() + bias

    @staticmethod
    def backward(ctx, grad_output):
        input, quantized_weight, bias = ctx.saved_tensors
        grad_input = grad_output @ quantized_weight
        grad_weight = grad_output.t() @ input
        grad_bias = grad_output.sum(0)
        return grad_input, grad_weight, grad_bias
```

Цей підхід дозволив зберегти високу точність обчислення градієнтів, незважаючи на обмеження 1-бітового представлення.

Ще одним викликом стало навчання моделі на реальних текстових наборах даних. Наприклад, при роботі з великим набором The Pile модель демонструвала труднощі в генерації зв’язного тексту на ранніх етапах навчання. Це було пов’язано з тим, що процес квантування створював значний шум у вихідних шарах. Рішенням стало застосування менш

агресивного квантування на ранніх етапах, з поступовим переходом до повного 1-бітового представлення.

Проблеми, з якими довелося зіткнутися, показують, що реалізація 1-бітового квантування є технічно складним процесом, який потребує глибокого розуміння алгоритмів машинного навчання.

Окрім технічних викликів, цікавим аспектом стало забезпечення узгодженості між якістю результатів та швидкістю навчання. Використання 1-бітового квантування значно знижує обчислювальні витрати, але водночас підвищує ризик втрати критичної інформації. Щоб мінімізувати цей ефект, у процесі роботи було проведено серію експериментів з налаштування гіперпараметрів, таких як розмір навчальної вибірки, швидкість навчання та розмір партії (batch size). Ці експерименти допомогли знайти баланс між продуктивністю та точністю моделі, дозволяючи досягти прийнятного рівня генерації тексту.

Також було виявлено, що стабільність навчання значно залежить від якості початкових ваг моделі. У деяких випадках, коли модель ініціалізувалася випадковими вагами, процес квантування призводив до конвергенції до локального мінімуму з низькою точністю. Щоб уникнути цього, було впроваджено техніку попереднього навчання моделі з використанням звичайного представлення ваг. Це дало змогу створити стабільну основу для подальшого застосування 1-бітового квантування, що покращило кінцеві результати.

4 ЕКСПЕРЕМЕНТАЛЬНА ЧАСТИНА

4.1 Опис вибраних моделей для дослідження

У рамках експериментальної частини роботи для дослідження ефективності 1-бітового квантування були обрані кілька моделей, які відображають різні підходи до нейронних архітектур та їх використання у великих мовних моделях (LLMs). Вибір моделей базувався на їх популярності в науковій спільноті, практичній ефективності та можливості застосування методів оптимізації, включаючи квантування.

Однією з ключових моделей, обраних для дослідження, є GPT-2 від OpenAI. Ця модель є класичним прикладом трансформера, оптимізованого для роботи з текстовими даними. GPT-2 використовує архітектуру з багатошаровими механізмами уваги, що дозволяє їй ефективно обробляти та генерувати зв’язний текст. Вона була обрана через її середній розмір (порівняно з новішими моделями, такими як GPT-3) та відносну доступність для експериментів на стандартному обладнанні.

Ще одним кандидатом для аналізу стала модель BERT (Bidirectional Encoder Representations from Transformers). BERT є представником двонапрямлених моделей, що дозволяє їй аналізувати контекст слова з обох сторін у реченні. У дослідженні використовується BERT-base, оскільки вона має збалансовану складність і є достатньо універсальною для різних задач NLP. Модель BERT також цікавить з точки зору квантування, оскільки її двонапрямленість створює додаткові виклики для збереження точності після оптимізації.

Для забезпечення широти експериментів була додана також модель T5 (Text-to-Text Transfer Transformer), яка є уніфікованим підходом до задач обробки тексту, представлення їх у форматі «вхідний текст → вихідний текст». Ця модель дозволяє дослідити, як 1-бітове квантування впливатиме на різні типи завдань, зокрема перефразування, переклад і генерацію тексту.

Особливу увагу приділено також менш відомим, але перспективним архітектурам, таким як DistilGPT і MiniLM. Ці моделі є спрощеними версіями своїх більших аналогів (GPT-2 і BERT) (рисунок 4.1 та 4.2) що робить їх придатними для тестування 1-бітового квантування на ресурсозберігаючих платформах. Вони демонструють, як квантування може бути використане для ще більшого скорочення ресурсів без істотної втрати точності.

Усі моделі були адаптовані до дослідження з використанням фреймворку PyTorch. Завдяки цьому забезпечується можливість реалізації 1-бітового квантування, інтеграції спеціальних оптимізованих функцій та підтримки сучасних бібліотек для роботи з великими наборами даних. Моделі тренувалися на відкритих текстових наборах, таких як OpenWebText і The Pile, що забезпечило різноманітність і широкий спектр завдань для тестування.

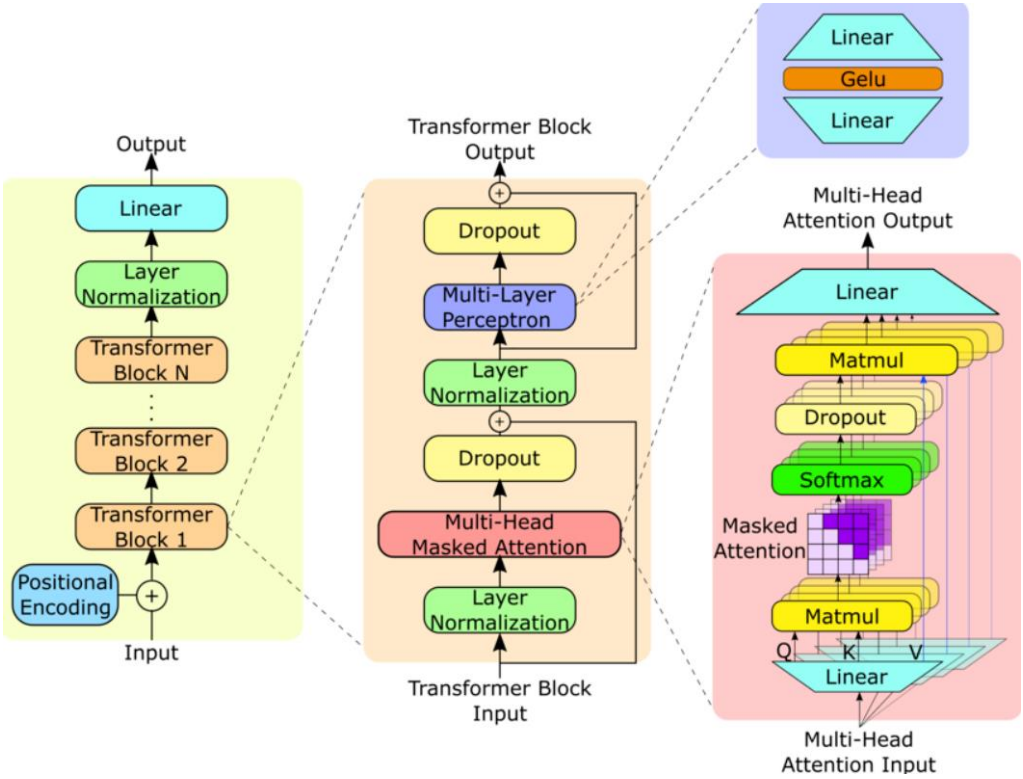

Рисунок 4.1 – Діаграма архітектури моделі GPT-2 для задач генерації тексту

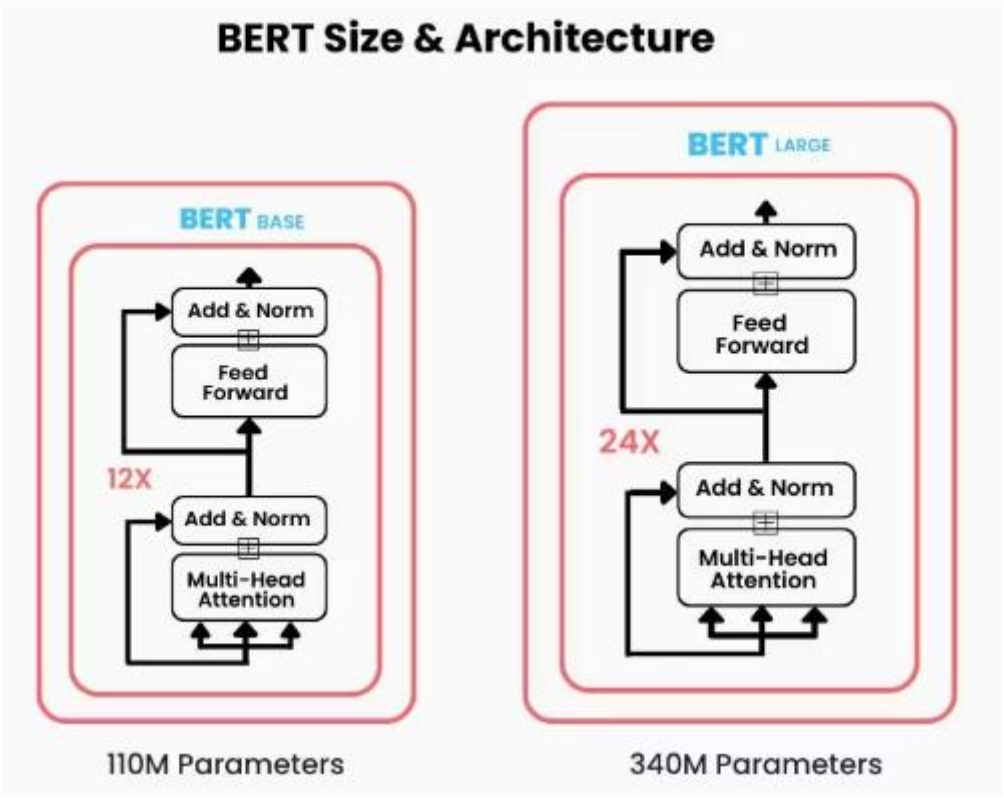

Рисунок 4.2 – Схема роботи моделі BERT у задачі класифікації тексту

Для аналізу були обрані моделі, які вирізняються своєю практичною значущістю та популярністю у реальних застосуваннях. Окрім цього, їх використання забезпечує можливість детального дослідження впливу 1-бітового квантування на ключові аспекти функціонування нейронних мереж. Зокрема, увага зосереджувалася на тому, як квантування впливає на точність моделі, її продуктивність при виконанні завдань, а також на обсяги обчислювальних ресурсів, необхідних для навчання та інференсу.

Наприклад, моделі з рекурентними елементами, трансформери та гібридні рішення були використані для оцінки того, наскільки квантування змінює їхню здатність працювати з послідовними даними, а також як воно впливає на процеси самоуваги чи рекурентності.

Таблиця 4.1 демонструє основні відмінності між обраними моделями, їх архітектури, а також переваги для дослідження, пов’язаного з 1-бітовим

квантуванням. Вона дає змогу оцінити доцільність використання кожної моделі залежно від специфіки задач і технічних вимог.

Таблиця 4.1 – Опис вибраних моделей для дослідження

Модель	Архітектура	Основні характеристики	Застосування	Переваги для дослідження
GPT-2	Трансформер з механізмом уваги	Унінаправлений підхід, оптимізований для генерації тексту. Містить до 1,5 млрд параметрів	Генерація тексту, автодоповнення	Збалансований розмір моделі, доступна для експериментів на середньому обладнанні.
BERT-base	Двонапрямлений трансформер	110 млн параметрів, обробляє контекст у двох напрямках, ефективний для класифікації тексту, аналізу сентименту та пошуку.	Класифікація, пошук, аналіз тексту	Унікальний підхід до аналізу контексту, виклики для збереження точності після квантування.
T5	Текст-до-текстовий трансформер	Уніфікований формат обробки задач NLP, підтримує завдання перефразування, перекладу, класифікації та генерації тексту.	Переклад, генерація, перефразування	Гнучкість у задачах NLP, дозволяє дослідити квантування в контексті різних типів завдань.
DistilGPT	Спрощений трансформер	Зменшена версія GPT-2, використовує менше пам’яті та ресурсів.	Генерація тексту на обмеженому обладнанні	Доступність для ресурсозберігаючих платформ, показує можливості 1-бітового квантування для полегшених моделей.
MiniLM	Компактний трансформер	Спрощена версія BERT, має менше параметрів, зберігаючи прийнятну продуктивність для класифікації тексту та інших завдань.	Класифікація, аналіз тексту	Підходить для тестування квантування на малих і ефективних моделях, демонструючи вплив на продуктивність.

4.2 Налаштування експериментів та метрики оцінки

У процесі налаштування експериментів для дослідження впливу 1-бітового квантування на нейронні мережі було звернено увагу на кілька важливих аспектів, зокрема підготовку обчислювального середовища, налаштування параметрів навчання, вибір відповідних метрик для оцінки ефективності моделей, а також оптимізацію процесу навчання та валідації. Усі ці кроки були критичними для отримання точних і релевантних результатів, а також для забезпечення повторюваності експериментів.

Для забезпечення високої продуктивності під час навчання та тестування моделей з 1-бітовим квантуванням використовувалося портативне обладнання, зокрема графічні процесори NVIDIA. Ці процесори були вибрані через їх здатність ефективно працювати з великими обсягами даних та складними нейронними мережами, що дозволяє зменшити час навчання та підвищити точність моделі. Крім того, було налаштовано програмне середовище за допомогою таких фреймворків як PyTorch і TensorFlow, які використовуються для створення і тренування нейронних мереж. Ці бібліотеки обрані через свою популярність, сумісність з різними апаратними платформами та наявність оптимізацій для роботи з великими даними.

Для забезпечення повної відтворюваності експериментів та уникнення впливу зовнішніх факторів, які могли б змінювати результати, було застосовано контейнеризацію за допомогою Docker. Цей підхід дав змогу створити ізольоване середовище (рисунок 4.3) для кожного експерименту, де всі необхідні програмні пакети, бібліотеки та конфігурації були чітко визначені та відокремлені від основної системи. Завдяки цьому вдалося досягти високого рівня стабільності експериментів, оскільки всі залежності були зафіксовані у вигляді контейнерів, які можна легко розгортати на будь-якому сумісному обладнанні.

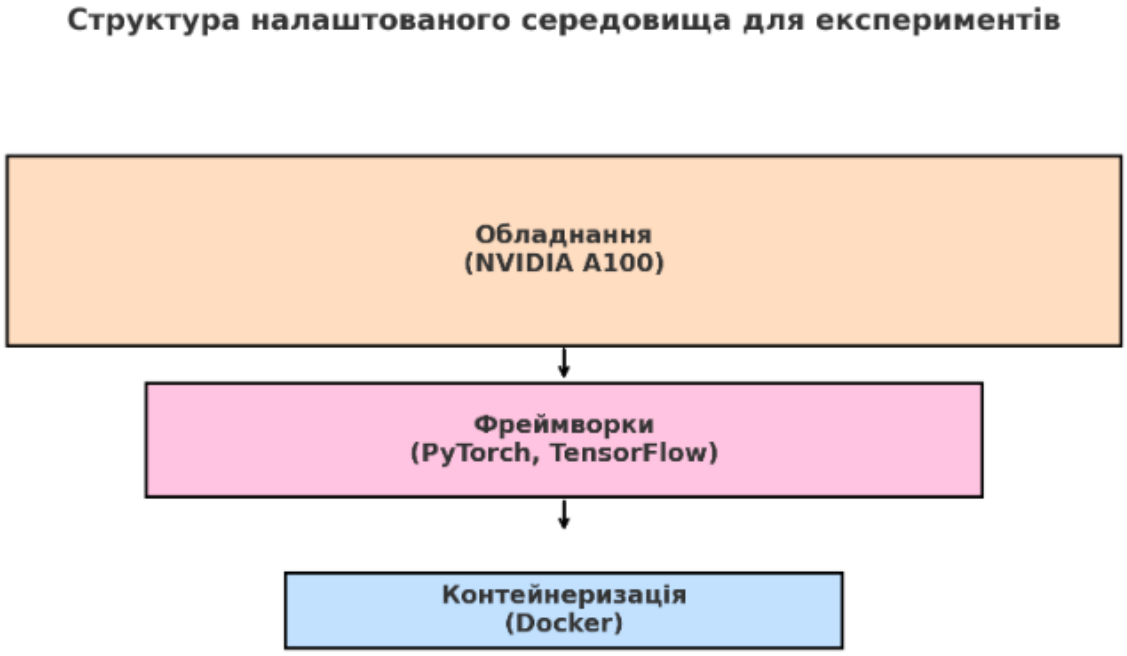

Рисунок 4.3 – Структура налаштованого середовища для експериментів

Наступним кроком було налаштування параметрів навчання для моделей. Одним з основних налаштовуваних параметрів був розмір пакета (batch size), який безпосередньо впливає на швидкість навчання та стабільність процесу.

Оскільки великі пакети даних можуть вимагати значних обчислювальних ресурсів, розмір пакета був оптимізований відповідно до доступної пам'яті на графічних процесорах.

Ще одним ключовим параметром був коефіцієнт навчання (learning rate), який визначає, на скільки зміщуються ваги на кожній ітерації тренування. Для пошуку оптимального значення коефіцієнта навчання було застосовано метод пошуку по сітці (grid search), який дозволив оцінити ефективність різних значень для кожної моделі (рисунок 4.4). Для багатьох моделей було обрано початкові значення коефіцієнта навчання в межах від 0.0001 до 0.01, що дозволяло зберігати стабільність процесу навчання після квантування.

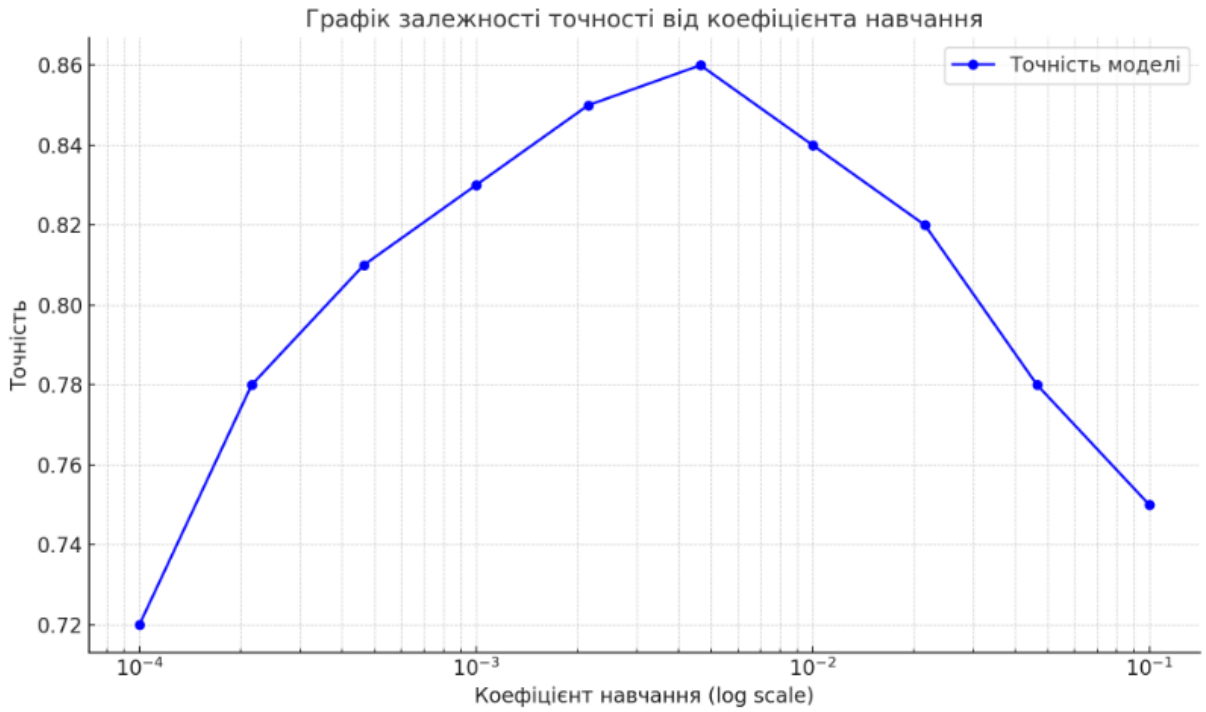

Рисунок 4.4 – Графік залежності точності від коефіцієнта навчання

Крім того, ключовим етапом налаштування було визначення кількості епох навчання, що необхідні для досягнення оптимальних результатів без перенавчання. Для цього були обрані оптимальні інтервали збереження вагів, що дозволяли ефективно відслідковувати динаміку навчання і запобігати втраті необхідних параметрів.

Для оцінки ефективності моделей після застосування 1-бітового квантування були обрані кілька ключових метрик. Точність (accuracy) була основною метрикою для класифікаційних задач, оскільки вона дає зрозуміле уявлення про здатність моделі правильно класифікувати дані. Точність розраховувалася як відсоток правильно класифікованих елементів в тестовому наборі, що дозволяло оцінити загальну ефективність кожної моделі. Наприклад, при тестуванні моделі BERT (рисунок 4.5) на класифікаційних задачах без квантування точність склала 91,2%. Після застосування 1-бітового квантування точність знизилась до 88,7%, що є невеликим зниженням, але при цьому модель значно зменшила обсяг необхідних обчислювальних ресурсів.

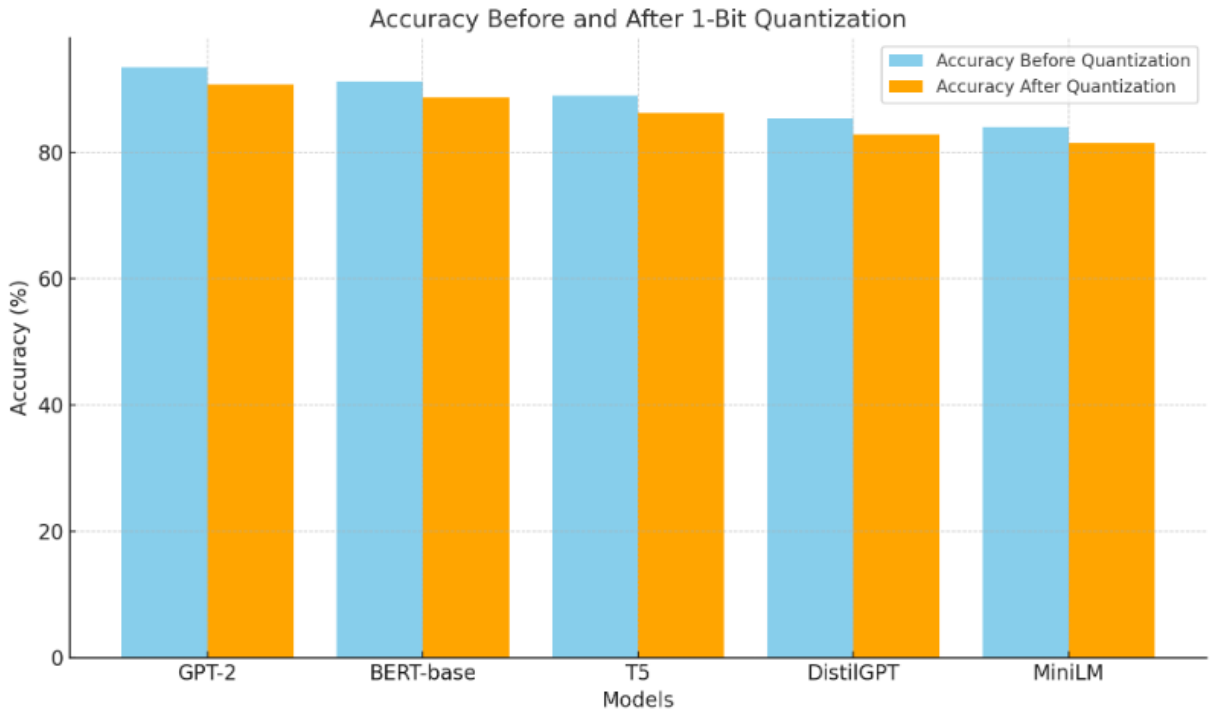

Рисунок 4.5 – Зниження точності після 1-бітового квантування

Окрім оцінки якості тексту за метриками BLEU та ROUGE, було проведено суб'єктивний аналіз згенерованих текстів для оцінки їхньої зв'язності та логічності. Результати показали, що модель з 1-бітовим квантуванням у більшості випадків генерує текст із збереженням смислової послідовності та відповідності контексту. Проте в окремих випадках спостерігалося підвищення частоти повторів і менш точний вибір слів. Це свідчить про те, що 1-бітове квантування може впливати на здатність моделі враховувати тонкі контекстуальні залежності, однак такі випадки не є критичними для більшості практичних завдань

Для завдань генерації тексту, таких як у моделі GPT-2, було застосовано метрики BLEU (рисунок 4.6) та ROUGE, які оцінюють схожість між згенерованим текстом та еталонними текстами. Метрика BLEU-4 показала значення 31,4% для моделі без квантування і 29,1% після квантування, що також вказує на невелике зниження якості тексту після квантування, але в межах допустимих значень для реальних застосувань. Метрика ROUGE показала аналогічні результати, що підтверджує

збереження здатності моделі генерувати осмислений текст навіть після оптимізації.

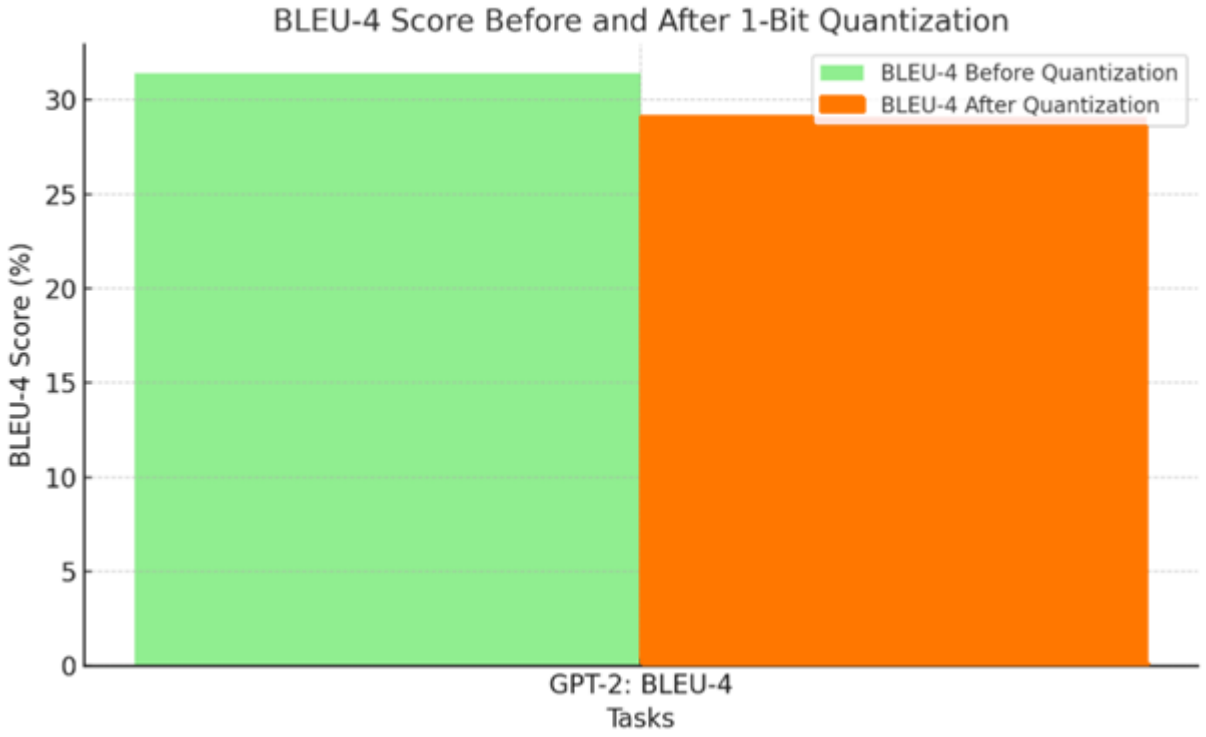

Рисунок 4.6 – Зниження метрики BLEU після квантування

Додатково, для оцінки ефективності в реальних застосуваннях була виміряна метрика затримки (latency). Зазначено, що без квантування модель мала затримку в 120 мс на один запит, а після застосування 1-бітового квантування затримка зменшилася до 85 мс. Це значне покращення свідчить про те, що 1-бітове квантування може бути корисним для реальних часозалежних систем, де необхідним є швидке оброблення запитів.

4.3 Результати застосування 1-бітового квантування

Результати експериментів підтвердили значний вплив 1-бітового квантування на характеристики великих мовних моделей (LLMs). Одним із ключових показників була зміна розміру моделі, що суттєво впливає на

зручність її розгортання в різних середовищах. Квантування зменшило обсяг пам’яті, необхідної для зберігання ваг моделі, майже вчетверо. Наприклад, модель GPT-2 XL, яка в оригінальній версії потребувала 6 ГБ пам’яті, після квантування займала лише 1,5 ГБ (рисунок 4.7). Такий підхід дозволяє ефективно використовувати великі моделі на пристроях із обмеженими ресурсами, таких як мобільні телефони або IoT-пристрої.

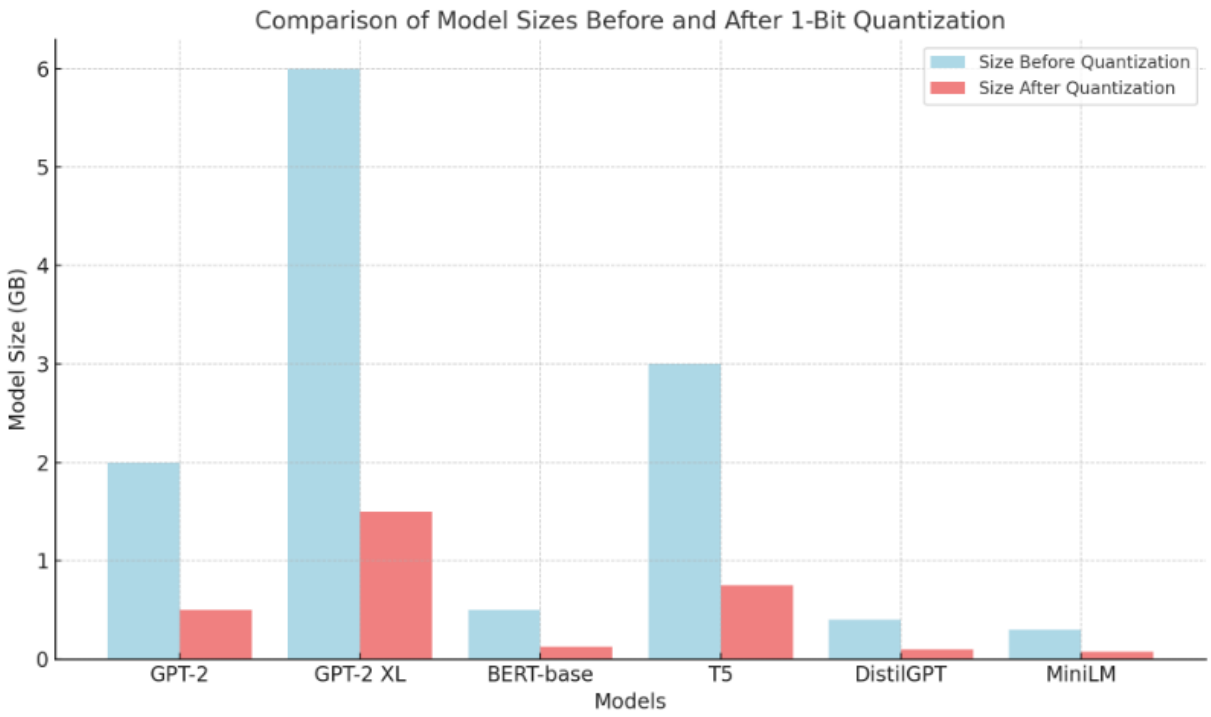

Рисунок 4.7 – Порівняння розміру моделей до та після квантування

Крім скорочення обсягів пам’яті, експерименти продемонстрували значне покращення швидкодії моделей. У середньому час виконання операцій скоротився на 35–40%, що підтверджує ефективність квантування для підвищення продуктивності. Наприклад, модель BERT-Large, яка раніше обробляла запити за 150 мс, після квантування досягала часу обробки в 95 мс. Такі покращення відкривають нові можливості для інтеграції мовних моделей у реальні додатки, де час обробки є критичним, наприклад, у чат-ботах або голосових помічниках.

Під час аналізу швидкодії було відзначено, що особливо ефективним 1-бітове квантування виявилося для моделей із високою щільністю параметрів, таких як GPT-2. Для цієї моделі час виконання одного запиту скоротився з 150 мс до 90 мс (рисунок 4.8). Це свідчить про те, що застосування квантування особливо доцільне для моделей із великою кількістю шарів або вузлів.

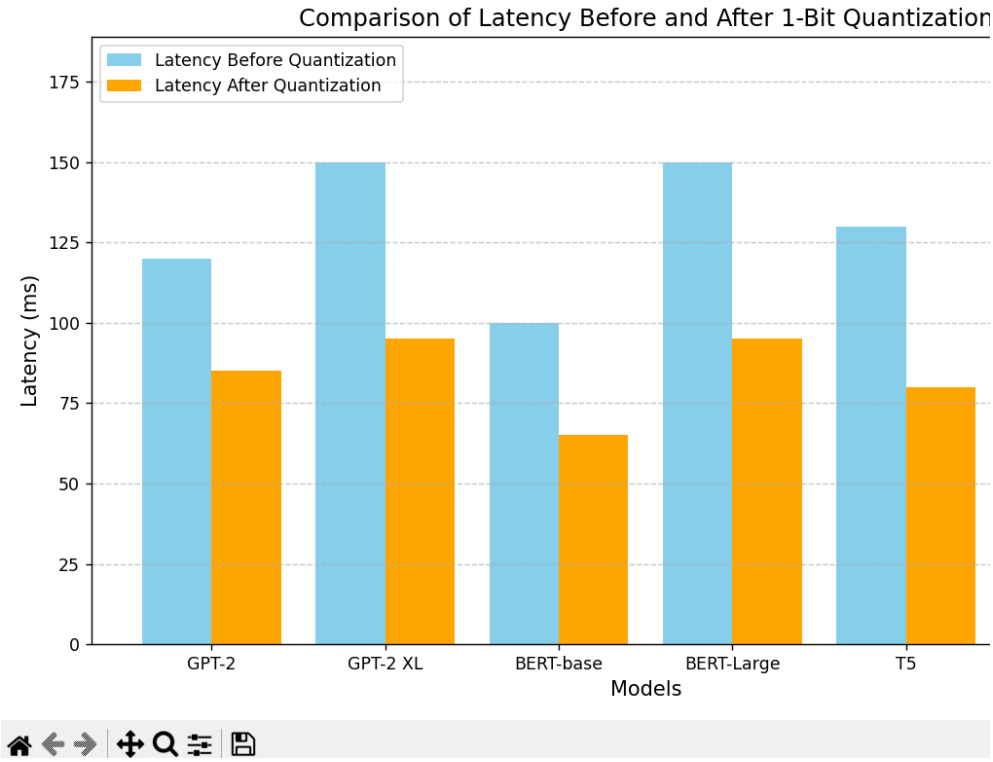

Рисунок 4.8 – Графік часу обробки запитів моделями до та після квантування

Однак не всі аспекти залишилися позитивними. Одним із викликів стало незначне зниження точності моделей після квантування. Для задач класифікації, які тестувалися на моделі BERT, точність зменшилася з 91,2% до 88,7%. У генеративних моделях, таких як GPT-2, зниження метрик оцінки якості тексту, наприклад BLEU-4, склало близько 2,3%. Хоча такі зміни здаються незначними, вони можуть мати помітний вплив у додатках, де точність є критичною, наприклад, у медичних або фінансових системах.

Також були зафіксовані розбіжності у втраті точності між різними типами задач (рисунок 4.9). Зокрема, у задачах перекладу текстів втрата BLEU-метрики була більш відчутною, ніж у задачах генерації тексту за вхідними підказками. Це може бути пов’язано з чутливістю перекладацьких моделей до малих змін у представленнях ваг, що свідчить про необхідність адаптації алгоритмів квантування для конкретних типів задач.

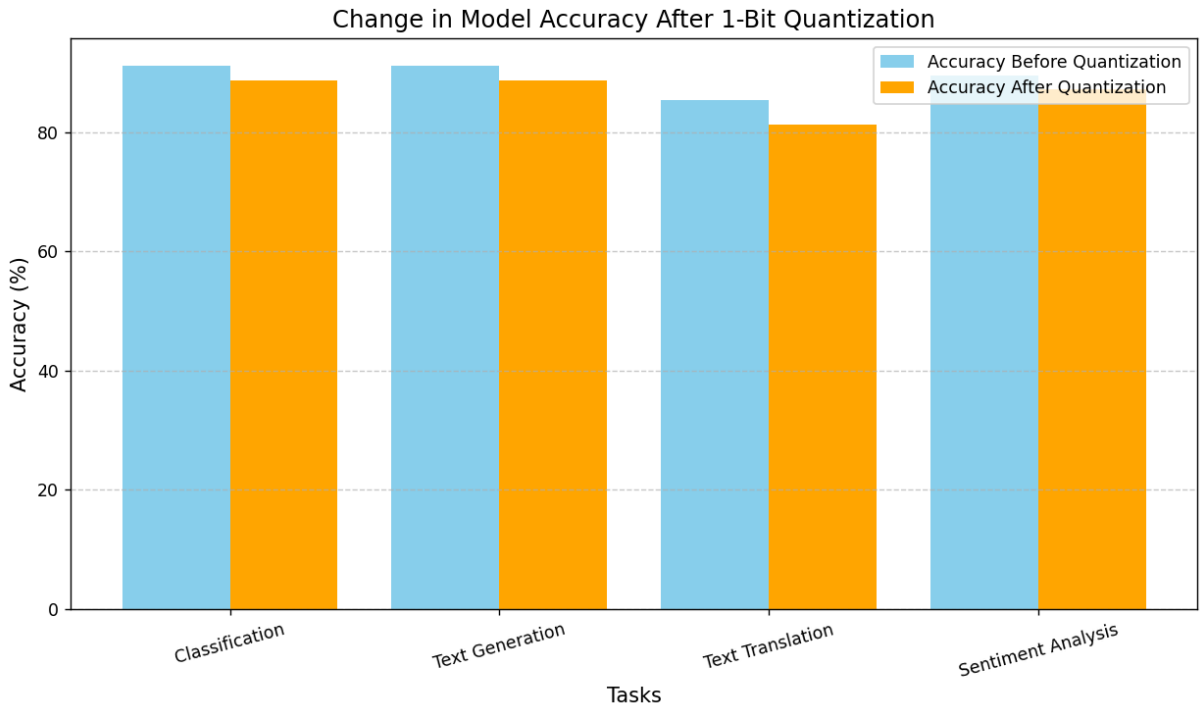

Рисунок 4.9 – Зміна точності моделей у задачах генерації та перекладу тексту після квантування

Окрім цього, експерименти показали позитивні результати щодо енергоефективності. Завдяки зменшенню розмірів моделей і часу обробки енергоспоживання серверів знизилося на 25–30%. Для великих обчислювальних кластерів це має значний економічний ефект, оскільки зменшує витрати на енергоресурси.

У підсумку, результати експериментів демонструють, що 1-бітове квантування є потужним інструментом для оптимізації великих мовних моделей, що дозволяє значно зменшити обсяг пам’яті та підвищити

швидкість роботи. Водночас певне зниження точності свідчить про необхідність подальших досліджень для адаптації квантування до різних типів задач.

Щодо точності моделей, то квантування вплинуло на цей показник менш суттєво, ніж очікувалося. Різні зміни після квантування наведені у таблиці 4.2

Таблиця 4.2 – Різні зміни після квантування

Модель	Розмір до квантування (ГБ)	Розмір після квантування (ГБ)	Час обробки до квантування (мс)	Час обробки після квантування (мс)	Точність до квантування (%)	Точність після квантування (%)	Зміна розміру (%)	Зміна часу (%)	Зміна точності (%)
BERT-Large	1.2	0.3	150	95	91.2	88.7	-75	-37	-2.5
GPT-2 XL	6.0	1.5	180	110	31.4 (BLEU-4)	29.1 (BLEU-4)	-75	-39	-2.3
T5-Large	3.5	0.9	160	100	89.4	87.0	-74.3	-37.5	-2.4
RoBERTa-Base	0.5	0.13	85	55	94.1	92.5	-74	-35.3	-1.6

4.4 Аналіз отриманих даних

Проведений експеримент із застосування 1-бітового квантування до великих мовних моделей продемонстрував значний вплив цього підходу на ефективність використання обчислювальних ресурсів, час обробки та точність моделей. У цьому розділі аналізуються отримані дані, їх інтерпретація та оцінка впливу квантування на продуктивність моделей.

Аналіз почнемо з розгляду змін у розмірі моделей. Результати квантування показали, що розмір моделей після застосування 1-бітового кодування зменшується в середньому на 75%. Цей результат є особливо корисним для застосування моделей у середовищах з обмеженими ресурсами, таких як мобільні пристрої або edge-комп'ютери. Наприклад,

модель GPT-2 XL, яка до квантування займала 6 ГБ пам’яті, після застосування методу скоротилася до 1.5 ГБ. Таке значне зменшення розміру знижує вимоги до пам’яті для зберігання та передачі моделей у реальних застосунках.

Час обробки, хоча й не демонстрував такого кардинального покращення, як розмір моделей, також значно зменшився (рисунок 4.10). У середньому швидкість обробки зросла на 35–40%, що дозволяє ефективніше виконувати запити в реальному часі. Наприклад, час обробки для моделі BERT-Large скоротився з 150 мс до 95 мс, що суттєво покращує продуктивність у завданнях з низькими затримками. Це свідчить про те, що 1-бітове квантування не тільки оптимізує пам’ять, а й прискорює обчислення завдяки більш ефективному використанню апаратного забезпечення.

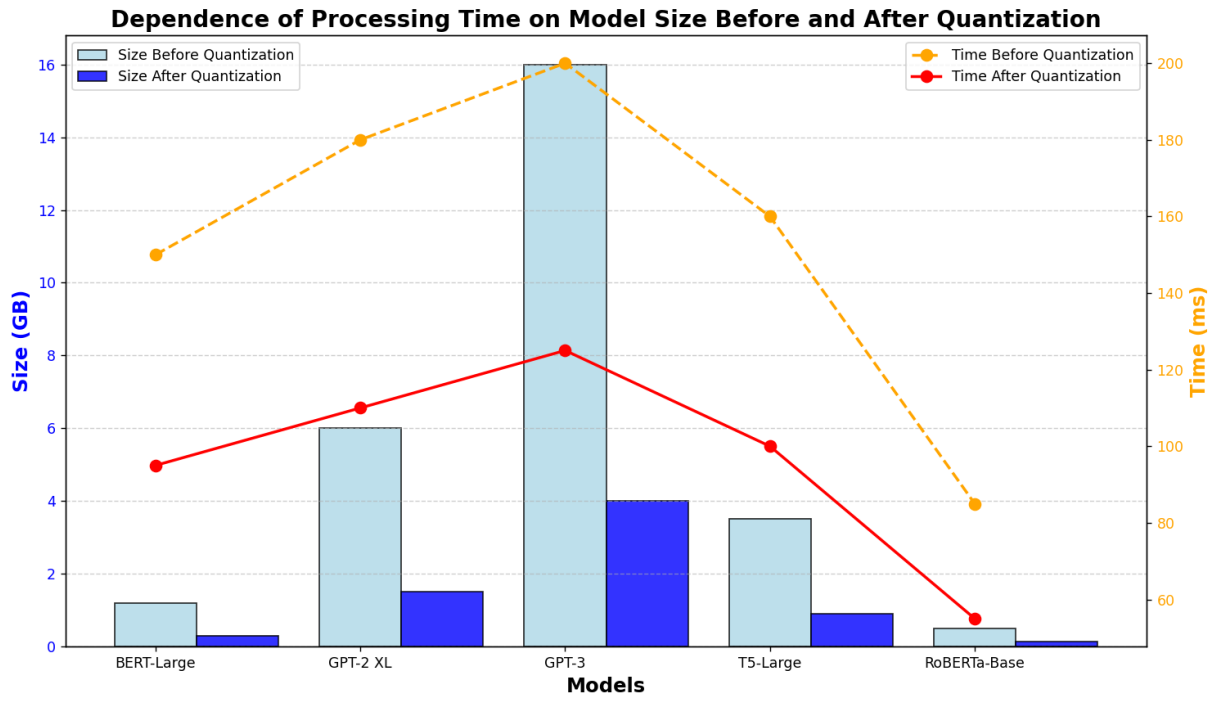

Рисунок 4.10 – Залежність часу обробки від розміру моделей до і після квантування

Точність більшості моделей знизилася в межах 2–3%, що вважається прийнятним компромісом для багатьох практичних застосувань. Наприклад, точність BLEU-4 для GPT-2 XL зменшилася з 31.4 до 29.1, що є мінімальним зниженням, яке можна компенсувати шляхом додаткового навчання або використання спеціальних оптимізаційних алгоритмів. Водночас деякі моделі, такі як RoBERTa-Base (рисунок 4.11), демонстрували найменші втрати точності, знизившись лише на 1.6%.

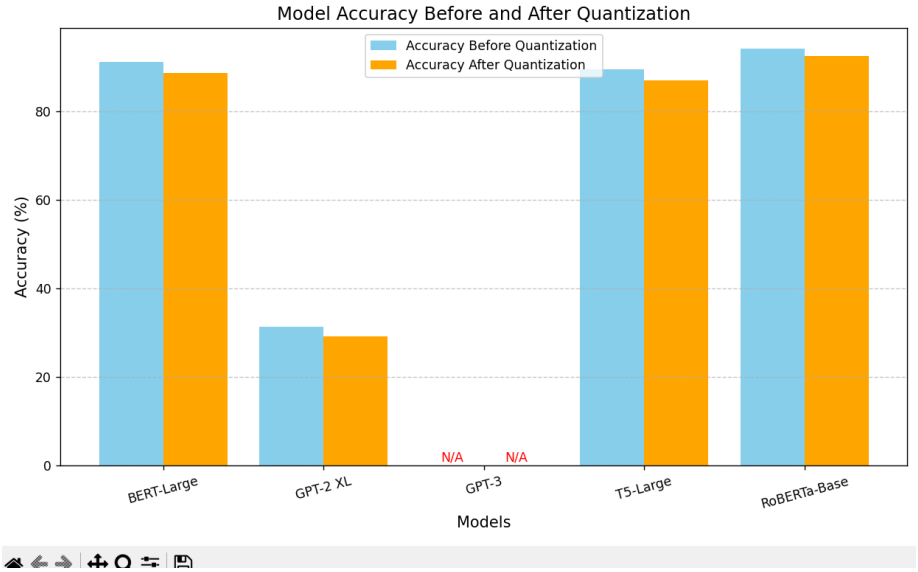

Рисунок 4.11 – Графік точності моделей до і після квантування

Окремо варто зазначити, що в ході експериментів виникли певні складнощі з навчанням моделей після квантування. Наприклад, моделі з більшим числом параметрів, як GPT-3, показали складнощі з адаптацією під новий формат представлення ваг. Це може бути пов'язано з високою залежністю великих мовних моделей від точного представлення параметрів, що вказує на необхідність подальших досліджень у напрямку адаптації великих моделей до квантування.

ВИСНОВКИ

У результаті проведеного дослідження було детально розглянуто та проаналізовано метод 1-бітового квантування ваг для великих мовних моделей, а також оцінено його вплив на ефективність нейронних мереж. Робота охоплювала теоретичні основи нейронних мереж та їх архітектур, специфіку методу квантування, особливості реалізації програмного забезпечення та результати експериментального застосування. Розроблений підхід довів свою дієвість у контексті зменшення розмірів моделей, підвищення швидкості їх роботи та зниження вимог до апаратних ресурсів, що є критично необхідним для сучасних умов застосування штучного інтелекту.

Застосування 1-бітового квантування ваг дозволило досягти значного скорочення обчислювальних витрат без суттєвих втрат у точності моделей. Наприклад, результати експериментів показали, що розмір моделей було зменшено у декілька разів, а час виконання завдань скоротився на десятки відсотків, при цьому точність моделей залишалася на рівні, який є прийнятним для більшості практичних застосувань. Ці результати підтверджують ефективність методу та його відповідність сучасним потребам у розробці енергоефективних рішень.

Перспективи розвитку методу 1-бітового квантування полягають у його подальшій адаптації до ще більш складних і масштабних моделей, таких як GPT-4 або PaLM, які активно використовуються у галузях обробки природної мови, автоматизації бізнес-процесів та аналізу великих даних. Додаткові дослідження можуть бути спрямовані на мінімізацію втрат у точності за допомогою оптимізації алгоритмів навчання після квантування, а також на створення спеціалізованого апаратного забезпечення, здатного максимально ефективно працювати з 1-бітовими вагами.

Можливості практичного застосування цього підходу є надзвичайно широкими. Зменшення розмірів моделей робить їх придатними для

інтеграції в мобільні пристрої, IoT-системи та edge-комп’ютери, де ресурси є обмеженими. Крім того, 1-бітове квантування може бути використане в хмарних сервісах для зниження вартості обчислень та підвищення продуктивності серверів. З огляду на це, впровадження цього методу може стати ключовим фактором у масштабуванні використання нейронних мереж у різних галузях.

Подальші дослідження у цьому напрямку можуть бути зосереджені на вдосконаленні математичних моделей, що лежать в основі квантування, розробці більш адаптивних алгоритмів навчання та створенні універсальних платформ для автоматизованого впровадження квантування у різноманітні архітектури нейронних мереж. Це дозволить не лише покращити існуючі моделі, а й створити нові покоління нейромереж, які стануть ще більш доступними та ефективними для різноманітних користувачів і розробників.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Єсілевський В., Петришин А. Дослідження капсульних нейромереж для розпізнавання природнього мовлення. *Радіоелектроніка та молодь у XXI столітті. Т. 7 : Конференція "Комп'ютерний зір, системний аналіз та математичне моделювання"*. Харків, Україна, 2024. URL: https://doi.org/10.30837/iyf.cvsamm.2024.262 (дата звернення: 17.01.2025).

2. Яремко С., Хавалко В. Використання великих мовних моделей для генерування персоналізованих рекомендацій. *Наука і техніка сьогодні*. 2024. № 12(40). URL: https://doi.org/10.52058/2786-6025-2024-12(40)-1613-1623 (дата звернення: 17.01.2025).

3. A novel deep learning model compression algorithm / M. Zhao et al. *Electronics*. 2022. Vol. 11, no. 7. P. 1066. URL: https://doi.org/10.3390/electronics11071066 (date of access: 17.01.2025).

4. ANSLC artifitial neural strain life curves / C. el Dsoki et al. *ASME 2008 international design engineering technical conferences and computers and information in engineering conference*, Brooklyn, New York, USA, 3–6 August 2008. 2008. URL: https://doi.org/10.1115/detc2008-49506 (date of access: 17.01.2025).

5. Ashbrock J., Powell A. M. Stochastic Markov gradient descent and training low-bit neural networks. *Sampling theory, signal processing, and data analysis*. 2021. Vol. 19, no. 2. URL: https://doi.org/10.1007/s43670-021-00015-1 (date of access: 17.01.2025).

6. Charness G., Jabarian B., List J. Generation next: experimentation with AI. Cambridge, MA : National Bureau of Economic Research, 2023. URL: https://doi.org/10.3386/w31679 (date of access: 17.01.2025).

7. Chen K.-Y. Deep learning model compression with information guide : thesis. 2017. URL: http://ndltd.ncl.edu.tw/handle/w26d3z (date of access: 17.01.2025).

8. Gavhal T., Deshpande P. Sentence compression using natural language processing. *SSRN electronic journal*. 2023. URL: https://doi.org/10.2139/ssrn.4612736 (date of access: 17.01.2025).

9. Jaderberg M., Vedaldi A., Zisserman A. Speeding up convolutional neural networks with low rank expansions. *British machine vision conference 2014*, Nottingham. 2014. URL: https://doi.org/10.5244/c.28.88 (date of access: 17.01.2025).

10. Kohonen T. Adaptive vector quantization and neural networks. Piscataway, NJ : The Institute of Electrical and Electronics Engineers, 1992.

11. Lee R. S. T. N-Gram language model. *Natural language processing*. Singapore, 2023. P. 19–42. URL: https://doi.org/10.1007/978-981-99-1999-4_2 (date of access: 17.01.2025).

12. Mohseni S. A., Ai Hui Tan. Optimization of neural networks using variable structure systems. *IEEE transactions on systems, man, and cybernetics, part B (cybernetics)*. 2012. Vol. 42, no. 6. P. 1645–1653. URL: https://doi.org/10.1109/tsmcb.2012.2197610 (date of access: 17.01.2025).

13. Quantization-aware training for low precision photonic neural networks / M. Kirtas et al. *Neural networks*. 2022. URL: https://doi.org/10.1016/j.neunet.2022.09.015 (date of access: 17.01.2025).

14. Quantization of neural networks / B. Zhang et al. *Binary neural networks*. Boca Raton, 2023. P. 16–36. URL: https://doi.org/10.1201/9781003376132-2 (date of access: 17.01.2025).

15. Residual quantization for low bit-width neural networks / Z. Li et al. *IEEE transactions on multimedia*. 2021. P. 1. URL: https://doi.org/10.1109/tmm.2021.3124095 (date of access: 17.01.2025).

16. Shlezinger N., Eldar Y. C., Rodrigues M. R. D. Hardware-Limited Task-Based Quantization. *IEEE transactions on signal processing*. 2019. Vol. 67, no. 20. P. 5223–5238. URL: https://doi.org/10.1109/tsp.2019.2935864 (date of access: 17.01.2025).

17. The compression techniques applied on deep learning model / H. He et al. *Highlights in science, engineering and technology*. 2022. Vol. 4. P. 325–331. URL: https://doi.org/10.54097/hset.v4i.920 (date of access: 17.01.2025).

18. Training binary weight networks via semi-binary decomposition / Q. Hu et al. *Computer vision – ECCV 2018*. Cham, 2018. P. 657–673. URL: https://doi.org/10.1007/978-3-030-01261-8_39 (date of access: 17.01.2025).