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)

Покращена архітектура трансформера за рахунок квантових обчислень
(тема)

Виконав:
здобувач другого року навчання,
групи СШІм-23-2
Кирило Попович
(власне ім'я, прізвище)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма Системи штучного інтелекту
(повна назва освітньої програми)
Керівник доц. Олександр Шевченко
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Олег ЗОЛОТУХІН
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти другий (магістерський)

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

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

Освітня програма Системи штучного інтелекту
(повна назва)

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Поповичу Кирилу Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Покращена архітектура трансформера за рахунок квантових обчислень

затверджена наказом університету від 21 квітня 20 25 р. № 295Ст

2. Термін подання студентом роботи до екзаменаційної комісії 5 червня 20 25 р.

3. Вихідні дані до роботи Електронні ресурси за обраною тематикою, мова програмування Python, фреймворк PyTorch, хмарне середовище тестування Kaggle або Google Colab.

4. Перелік питань, що потрібно опрацювати в роботі

1) Аналіз предметної галузі та постановка задачі

2) Фізичні величини і базові оператори квантової алгебри

3) Програмні бібліотеки для реалізації квантових обчислень

4) Опис програмної реалізації

5) Проведення експерименту

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	21.04.2025	виконано
2	Аналіз предметної галузі і постановка задачі	30.04.2025	виконано
3	Розробка VQC для квантово-посиленого	07.05.2025	виконано
	трансформера		
4	Програмна реалізація квантово посиленого	14.05.2025	виконано
	трансформера		
5	Проведення експерименту та аналіз результатів	21.05.2025	виконано
6	Підготовка пояснювальної записки	28.05.2025	виконано
7	Підготовка презентації та доповіді	29.05.2025	виконано
8	Перевірка на плагіат	30.05.2025	виконано
9	Нормоконтроль	31.05.2025	виконано
10	Рецензування	01.06.2025	виконано
11	Попередній захист	02.06.2025	виконано
12	Захист перед ЕК	05.06.2025	виконано

Дата видачі завдання 21 квітня 20 25 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Олександр Шевченко
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 59 с., 11 рис., 1 табл., 1 дод., 31 джерело.

ВАРІАЦІЙНІ КВАНТОВІ СХЕМИ, КВАНТОВІ ОБЧИСЛЕННЯ, КВАНТОВО-ПОСИЛЕНА УВАГА, LLM, Q-ТРАНСФОРМЕР.

Об’єктом дослідження є нейронні мережі трансформерної архітектури.

Метою роботи є розробка та дослідження покращеної архітектури трансформера, що використовує квантові обчислювальні елементи для підвищення ефективності, швидкодії та якості обробки даних у задачах штучного інтелекту.

Методами дослідження є аналіз літературних джерел з класичних і квантових трансформерів, математичне моделювання квантових обчислювальних блоків у структурі трансформера, симуляція роботи квантових нейронних мереж із застосуванням фреймворку PyTorch+ PennyLane, експериментальна перевірка моделі на типових задачах та порівняльний аналіз результатів з базовими класичними трансформерами.

У результаті досліджено підходи до побудови гібридної трансформерної архітектури, яка поєднує класичні нейронні мережі з квантовими обчислювальними елементами. Розглянуто концепцію варіаційних квантових схем, реалізовано експериментальну модель на основі фреймворків PyTorch і PennyLane. Проведено навчання моделі на наборі даних IMDb для задачі аналізу сентименту, а також виконано порівняльну оцінку з класичним трансформером за основними метриками точності. Результати підтверджують перспективність інтеграції квантових елементів у глибокі архітектури для покращення узагальнюючих властивостей моделей.

ABSTRACT

Master's thesis contains: 59 pp., 11 fig., 1 tabl., 1 ann., 31 references.

LLM, Q-TRANSFORMER, QUANTUM COMPUTING, QUANTUM-ENHANCED ATTENTION, VARIATIONAL QUANTUM CIRCUITS.

The object of research is the transformer architecture of neural networks.

The purpose of the work is to develop and investigate an improved transformer architecture that uses quantum computing elements to enhance the efficiency, speed, and quality of data processing for artificial intelligence tasks.

Research methods include literature analysis on classical and quantum transformers; mathematical modeling of quantum computing units within the transformer structure; simulation of quantum neural networks using the PyTorch + PennyLane framework; experimental verification of the model on typical tasks; and comparison of the results with those of basic classical transformers.

As a result, approaches to building a hybrid transformer architecture that combines classical neural networks with quantum computational elements have been explored. The concept of variational quantum circuits has been considered, and an experimental model was implemented using the PyTorch and PennyLane frameworks. The model was trained on the IMDb dataset for sentiment analysis tasks, and a comparative evaluation with a classical transformer was carried out using standard accuracy metrics. The results confirm the potential of integrating quantum components into deep architectures to improve the generalization capabilities of models.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AI – Artificial Intelligence – штучний інтелект;

CV – Computer Vision – комп'ютерний зір;

LLM – Large Language Model – модель великої мови;

NLP – Natural Language Processing – обробка природної мови;

PQC – Parameterized Quantum Circuit – параметризована квантова схема;

QLA – Quantum Linear Algebra – квантова лінійна алгебра;

RL – Reinforcement Learning – навчання з підкріпленням;

ViT – Vision Transformer – трансформатор зору;

VQC – Variational Quantum Circuits – варіаційні квантові схеми.

ВСТУП

Сучасні нейромережеві архітектури, зокрема трансформери, сьогодні відіграють ключову роль у розв'язанні широкого кола задач ШІ, від обробки природної мови до комп'ютерного зору та генерації контенту. Проте, незважаючи на значні успіхи, класичні підходи стикаються з фундаментальними обмеженнями щодо масштабування, швидкості навчання та ефективності обчислень. Одним з перспективних напрямків для подолання цих обмежень є інтеграція квантових обчислень у структуру нейронних мереж. Особливий інтерес становить створення гібридних класично-квантових архітектур трансформерів, які можуть потенційно забезпечити значне підвищення швидкодії та точності моделей завдяки використанню квантових обчислювальних блоків.

Актуальність теми дослідження обумовлена інтенсивним розвитком квантових технологій, а також необхідністю подолання меж продуктивності традиційних нейромережевих моделей. Перспективність такого підходу підтверджується зростаючою кількістю публікацій у провідних наукових журналах та активністю міжнародних технологічних компаній, таких як IBM, Google, Xanadu, які активно інвестують у розробку квантових обчислювальних платформ для задач ШІ.

Виконувана робота пов'язана з актуальними науковими напрямками кафедри ШІ ХНУРЕ, зокрема «Обчислювальний інтелект та глибинне навчання», «Інтелектуальні інформаційні технології обробки даних» та «Системи штучного інтелекту з використанням новітніх обчислювальних технологій».

Мета роботи полягає в розробці та експериментальному дослідженні нової гібридної архітектури трансформера з використанням квантових обчислювальних елементів для суттєвого підвищення продуктивності, швидкості навчання та якості вирішення задач штучного інтелекту.

Для досягнення цієї мети необхідно розв’язати такі задачі:

– аналіз існуючих архітектур трансформерів та квантових моделей;

– проектування гібридної архітектури з квантовими елементами;

– реалізація квантових шарів та їх інтеграція у модель;

– експериментальна перевірка та порівняння результатів із класичними моделями.

Об’єктом дослідження є нейронні мережі трансформерної архітектури.

Предметом дослідження є підходи до інтеграції квантових обчислювальних елементів у структуру класичних трансформерів, вплив квантових блоків на ефективність, швидкодію та якість розв'язання типових задач ШІ.

Методами дослідження є аналіз літературних джерел з класичних і квантових трансформерів, математичне моделювання квантових обчислювальних блоків у структурі трансформера, симуляція роботи квантових нейронних мереж із застосуванням фреймворку PyTorch + PennyLane, експериментальна перевірка моделі на типових задачах та порівняльний аналіз результатів з базовими класичними трансформерами.

Наукова новизна роботи полягає у використанні нової гібридної архітектури трансформера, яка вперше поєднує класичні нейронні механізми трансформерів із квантовими обчислювальними блоками та обґрунтованнні ефективності використання квантових технологій для вирішення типових задач глибинного навчання порівняно з класичними архітектурами.

Практична значущість полягає у створенні програмної реалізації гібридної архітектури з використанням бібліотек PyTorch + PennyLane, що може бути використана для подальших досліджень у сфері квантового машинного навчання та інтегрована у практичні системи ШІ. Запропоновані методики можуть бути адаптовані та застосовані у комерційних рішеннях, які потребують високої точності та швидкодії, таких як аналіз природної мови, комп'ютерний зір, медична діагностика та інше.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Трансформер

Трансформери – це глибокі нейромережеві моделі, що стали проривом у сучасному ШІ. Ключова ідея трансформера – механізм самоуваги [1], який дає змогу кожному елементу послідовності враховувати вплив інших елементів. Класичний трансформер складається зі стеку шарів, кожен з яких містить багатоголовий механізм уваги, позиційно-залежні повнозв'язні шари та резідуальні зв'язки з нормалізацією (рисунок 1.1).

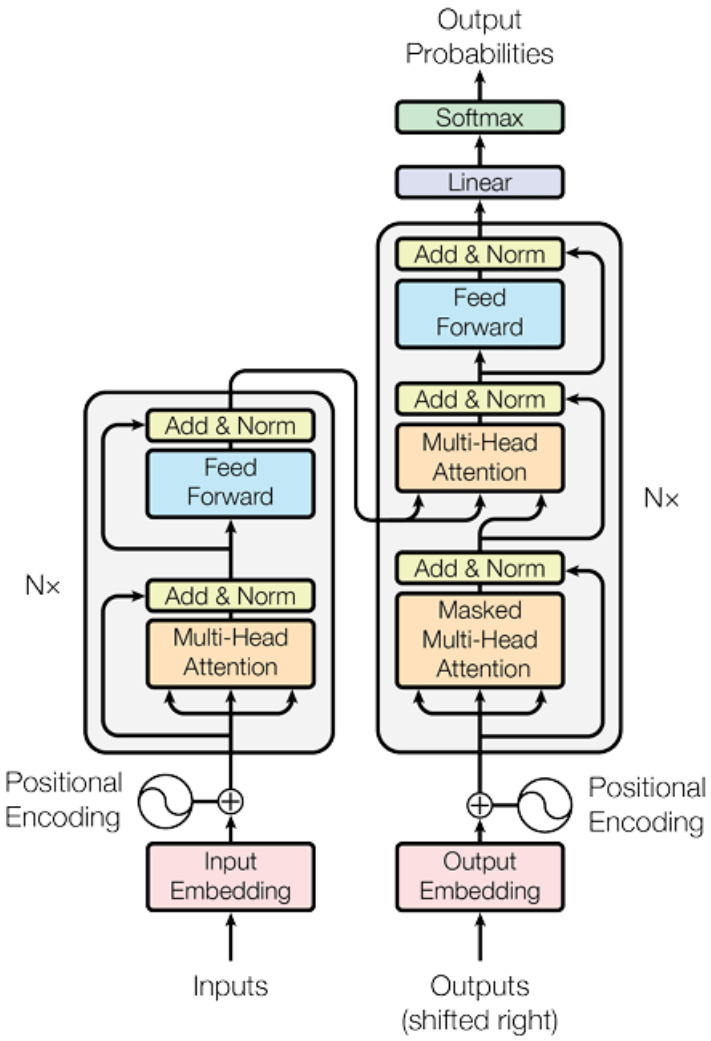


Рисунок 1.1 – Загальна архітектура повного трансформера [1]

Така архітектура ефективно вчиться контекстуалізувати дані: самоувага моделює довгострокові залежності, а резідуальні зв'язки полегшують тренування глибоких моделей.

Існує декілька архітектурних варіантів трансформера: тільки енкодер, тільки декодер, повний енкодер-декодер.

Тільки енкодер приймає вхідний текст і перетворює його в компактне представлення (вектор ознак). Підходить для класифікації, аналізу тональності, екстракту набору ознак з тексту, пошуку семантично близьких текстів. Прикладами таких трансформерів є BERT, RoBERTa, DistilBERT та інші.

Тільки декодер працює авто-регресивно (генерує вихід покроково, послідовно). Підходить для генерації тексту (як продовження), чат-ботів, автозавершення. Прикладами таких трансформерів є GPT, LLAMA, Mistral та інші.

Повний енкодер-декодер будує набір ознак з вхідного тексту та генерує відповідь. Підходить для задач, де вхід і вихід дуже сильно зв'язані, наприклад машинний переклад, складання підписів до зображень, узагальнення тексту, діалогів типу питання-відповідь. Прикладами таких трансформерів є T5, BART, MarianMT, Pegasus та інші.

З моменту анонсування у 2017 р. трансформери досягли видатних успіхів у різних галузях. Зокрема, в обробці природної мови [2] трансформери стали базовим інструментом: такі моделі як BERT і GPT-4 (175 млрд параметрів) [3] продемонстрували неперевершене розуміння і генерування тексту.

У комп'ютерному зорі архітектура ViT показала, що можна відмовитися від згорткових мереж і досягти конкурентних результатів, кинути виклик домінуванню CNN у задачах класифікації та обробки зображень [4]. ViT став еталоном для досліджень у CV [5].

Трансформери знаходять застосування й в інших сферах. Зокрема, біоінформатика активно переймає цю архітектуру: з'явилися потужні

моделі для ДНК та білків (такі, як Nucleotide Transformer, DNABERT тощо), що моделюють послідовності геному подібно до того, як мовні моделі працюють з текстом [6].

Так само моделі на базі трансформерів застосовуються у хімії (для моделювання послідовностей молекул), в обробці сигналів, а також експериментуються у зміцненні RL [7].

Можна зробити висновок, що завдяки гнучкості й потужності механізму самоуваги трансформери стали універсальною архітектурою у сучасному ШІ, здатною працювати як з мовними, так і з візуальними чи послідовними біологічними даними.

1.2 Обмеження класичних трансформерів

Попри успіхи, класичні трансформери стикаються з суттєвими обмеженнями під час реалізації на класичних обчислювальних машинах:

– квадратична складність роботи і використаної пам’яті на довгих вхідних послідовностях;

– повільне навчання;

– масштабованість та інфраструктурні вимоги.

Найбільший недолік трансформера – це масштабування механізму уваги. Обчислювальна складність і використання пам’яті зростають квадратично зі збільшенням довжини послідовності n, адже потрібно обчислити повну матрицю уваги розміром $n \times n$ для кожного шару [1]. Така складність $O(n^2)$ швидко стає непідйомною на довгих текстах або послідовностях – наприклад, для контексту у 10 000 токенів розмір матриці уваги сягає 100 млн елементів. Це обмежує максимальну довжину вхідної послідовності (більшість сучасних мовних моделей обмежені кількома тисячами токенів) і вимагає дуже великого об’єму пам’яті на GPU для зберігання проміжних результатів. Моделі на основі Transformer

застосовують цю повну увагу і тому обмежені відносно короткими вхідними секвенціями (BERT ~512 токенів, GPT-3 ~2048 токенів) [9,10].

Кількість обчислень у трансформері залежить не лише від довжини вхідної послідовності, а й від числа параметрів моделі. Сучасні трансформери налічують мільярди параметрів (ембедінги, матриці ваг всіх блоків уваги Query, Key, Value, матриці ваг шарів FeedForward, нормалізації, вихідного шару або декодеру), тому їхнє навчання потребує величезних обчислювальних ресурсів і часу. Наприклад, навчання моделі GPT-3 з 175 млрд параметрів оціночно спожило близько *1 300 МВт·год* електроенергії [10]. Це еквівалент річного енергоспоживання ~130 середніх домогосподарств у США [10]. Такий тренінг тривав би місяці навіть на великих кластерах GPU. Хоча трансформери добре паралелізуються по різних позиціях послідовності, глибина моделей (сотні шарів) і складність кожного кроку призводять до значних затримок без потужного апаратного забезпечення.

Щоб тренувати сучасні трансформери, потрібна спеціалізована інфраструктура – кластери з сотень GPU/TPU, високошвидкісні мережі й продумане розподілене навчання. Не кожна організація може собі це дозволити. Розподіл моделі по кількох пристроях теж ускладнює процес – зростають витрати на синхронізацію, пам’ять повинна вмістити модель, а пропускна здатність пам’яті стає вузьким місцем. Це спонукає до пошуку нових архітектурних та алгоритмічних рішень, щоб подолати перелічені вузькі місця.

1.3 Сучасні підходи до вирішення проблем ефективності трансформерів

Щоб вирішити зазначені проблеми, розробляється низка підходів для підвищення ефективності трансформерів. Серед них можна виділити дві великі групи: 1) модифікації архітектури трансформера в межах класичних

обчислень, що оптимізують алгоритм уваги та інші компоненти; 2) використання квантових обчислень для прискорення або покращення роботи трансформера.

1.3.1 Модифікація архітектури трансформера

В рамках першої групи дослідженнями було породжено сімейство «X-former» моделей. До них належать, наприклад, Reformer, Linformer, Performer, Longformer, BigBird, Sparse Transformer та інші [7]. Такі моделі спрощують або наближують класичну реалізацію самоуваги, щоб зменшити вимоги до обчислень і пам'яті.

Модель Reformer [11] пропонує використовувати локально-чутливе хешування для наближеного пошуку найближчих сусідів у самоувазі. Це дозволило зменшити складність обчислення уваги з $O(n^2)$ до субквадратичної $O(n \times \log(n))$. Додатково Reformer використовує оборотні шари, щоб знизити використання пам'яті під час навчання – активації зберігаються лише один раз замість n разів для n шарів.

Модель Performer [12] пропонує метод рандомізованих ознак для наближення softmax-уваги. Алгоритм FAVOR+ обчислює увагу через випадкові проєкції в просторі більшої розмірності, що дозволяє оцінити результуючу матрицю уваги з малим похибкою. Використовуючи цей підхід, алгоритм досягає лінійної складності і в часі, і в пам'яті. Він забезпечує неупереджену оцінку повної уваги з теоретичними гарантіями збіжності, а експерименти показали конкурентну якість на різноманітних задачах при значно кращій масштабованості.

Модель Longformer [13] пропонує розріджену схему уваги для довгих документів. Замість повної матриці уваги використовується комбінація локального вікна (кожен токен бачить сусідів у межах фіксованого вікна) та глобальних токенів (декілька позицій можуть бачити всю послідовність). Це дає лінійну складність $O(n)$ за довжиною послідовності, дозволяючи моделі

обробляти тисячі токенів. Longformer успішно застосовується для задач з довгим контекстом і показує кращу якість, ніж базові BERT-подібні моделі на довгих вхідних даних. Модель BigBird [14] реалізує розріджену увагу шляхом поєднання трьох видів зв'язків: кожен токен має увагу в локальному вікні, до кількох глобальних токенів та до випадкової підмножини інших токенів. Така схема дозволяє зменшити квадратичну залежність до лінійної. На практиці BigBird обробляє послідовності ~8 разів довші на тому самому обладнанні, порівняно зі стандартним Transformer [14], і покращує результати на задачах читання довгих текстів та у біомедичних даних.

Модель FlashAttention [15] пропонує точний алгоритм обчислення самоуваги, оптимізований для ієрархії пам'яті GPU. Ключова ідея – розбити обчислення матриці QK^T на малі блоки, які цілком розміщуються у швидкій on-chip пам'яті GPU. Блоки ключів і значень завантажуються з глобальної пам'яті по черзі, і проміжні результати агрегуються на чипі без запису повної матриці в повільну пам'ять. Таким чином FlashAttention знижує витрати по пам'яті уваги з $O(n^2)$ до $O(n)$.

Узагальненням робіт у цьому напрямі стало теоретичне дослідження [16], в якому формально доводиться, що прискорити самоувагу без втрати точності надто складно. Автори встановили умовну нижню межу часу для точного self-attention: алгоритм, що обробляє усі вхідні випадки без значної похибки, вимагає не менше квадратичного часу. Ця робота пояснює, чому більшість ефективних трансформерів вводять певні наближення – розрідження, агрегування або стохастику – аби обійти гірший випадок, і чому повністю точна і водночас субквадратична увага малоймовірна.

1.3.2 Використання квантових обчислень у трансформерах

Використання квантових обчислень є більш радикальним шляхом подолання меж класичних трансформерів. У цій галузі виокремлюють два основні підходи [17]:

– гібридні трансформери на основі параметричних квантових схем (PQC);

– квантово-лінійна алгебра для трансформерів (QLA).

У випадку PQC у класичну модель інтегруються варіаційні квантові схеми (VQC). VQC – це невеликі квантові підмоделі з параметризованими квантовими воротами. По суті, частина обчислень трансформера (або окремі шари) замінюється на квантову схему, яка виконується на квантовому процесорі або його симуляторі.

Параметрами таких схем є кути обертання квантових воріт. Ці параметри навчаються класичними методами оптимізації (градієнтними алгоритмами) на основі вимірювань квантових станів. Цей гібридний підхід розрахований на шумні квантові комп'ютери проміжного масштабу (NISQ-пристрої) – тобто реальні квантові процесори з десятками–сотнями кубітів, що поки не є ідеальними, але вже доступні [18]. Квантові шари можуть генерувати більш ефективні ознаки та зменшувати необхідну кількість параметрів мережі, не погіршуючи якість, а також обчислювати коефіцієнти уваги паралельно (зразу для всіх пар токенів), використовуючи суперпозицію – це відкриває шлях до потенційного прискорення обчислення матриці уваги.

Розвиток QLA – це більш теоретичний підхід [18]. Його мета – переписати алгоритми трансформера з використанням квантових алгоритмів лінійної алгебри, які мають кращу асимптотичну складність. У квантовій обчислювальній теорії відомо багато алгоритмів, що дають експоненційний або полілогарифмічний виграш для задач лінійної алгебри: наприклад, швидкі алгоритми для розв'язання систем лінійних рівнянь, обчислення власних значень (QLSA, HHL-алгоритм), для перетворення Фур'є, виконання матричних експонент тощо.

QLA-підхід застосовує ці квантові примітиви до основних операцій трансформера: множення матриць при обчисленні уваги і у feed-forward шарах, застосування нелінійних функцій активації. Зокрема, пропонується

кодувати всю матрицю ваг уваги у квантовий стан і реалізувати обчислення $Q * K^T$ та Softmax через квантові підпрограми – це теоретично може зменшити складність обчислення уваги з $O(n^2)$ до $O(n \times \log(n))$.

Однак QLA-методи наразі майже не реалізовані експериментально, адже вимагають масштабних квантових комп'ютерів (десятки тисяч логічних кубітів з корекцією помилок), яких поки не існує.

1.4 Постановка задачі

Виходячи з аналізу предметної галузі, можна зробити висновок, що актуальною та такою, що піддається практичній реалізації, є задача подолання обмежень класичних трансформерів шляхом впровадження параметричних квантових схем (PQC). Такі гібридні моделі надалі будемо називати квантово-посиленими трансформерами.

Основною гіпотезою дослідження є припущення, що використання квантових блоків у структурі трансформерів може покращити обчислювальну ефективність або якість моделі у певних класах задач, особливо тих, де класичні методи стикаються з обмеженнями масштабованості чи швидкодії.

Для досягнення цієї мети передбачається виконати такі кроки:

– провести огляд сучасних підходів до реалізації окремих блоків трансформера за допомогою варіаційних квантових схем (VQC), зокрема FeedForward;

– провести огляд та порівняння сучасних програмних бібліотек для емуляції квантових обчислень (PennyLane, Qiskit, т.д.), з вибором оптимальної для реалізації експериментальної частини;

– обрати прикладну задачу (наприклад, класифікація текстів), для якої буде досліджуватися ефективність квантово-посиленого трансформера;

– реалізувати гібридну (посилену) модель трансформера з квантовими блоками для розв’язання обраної задачі, провести її навчання та тестування на відповідних наборах даних;

– провести порівняльний аналіз отриманих результатів із класичною реалізацією трансформера за ключовими метриками (точність, швидкодія, споживання ресурсів);

– проаналізувати потенціал масштабування квантових підходів, а також їхню стійкість та залежність від параметрів (кількість кубітів, глибина схеми тощо).

Результати дослідження дозволять оцінити переваги та обмеження сучасних квантово-посилених трансформерних архітектур у вирішенні практичних задач.

2 РОЗРОБКА VQC ДЛЯ КВАНТОВО-ПОСИЛЕНОГО ТРАНСФОРМЕРА

В основі квантово-посиленого трансформера лежать параметричні квантові схеми (PQC). PQC – це загальна квантова схема, що містить операції (гейти), параметризовані певними параметрами (наприклад, кутами повороту кубітів). PQC може бути застосована у різних контекстах (як частина квантового алгоритму, симуляція фізичних процесів тощо). Параметри можуть задаватися користувачем вручну, випадковим чином, чи налаштовуватись будь-яким алгоритмом.

Варіаційна квантова схема (VQC) – це окремий випадок PQC. Вони застосовуються у варіаційних алгоритмах, тобто параметри цієї схеми оптимізуються класичним оптимізатором відповідно до певної цільової функції (класична зворотя петля). Параметри схеми цілеспрямовано оновлюються для оптимізації цільової функції. Завданням VQC є мінімізація/максимізація деякої функції втрат.

Загальна схема VQC зображена на рисунку 2.1.

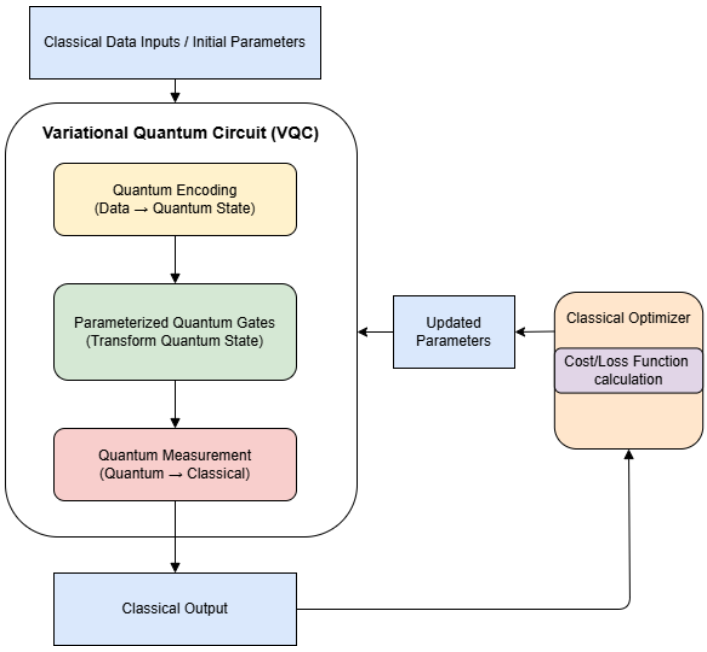


Рисунок 2.1 – Загальна схема VQC (рисунок виконано самостійно)

VQC використовують так:

– спочатку дані кодують у квантовий стан;

– параметризовані квантові ворота здійснюють трансформацію цього стану;

– після цього здійснюється вимірювання квантового стану, і результат перетворюється на класичний вихід (наприклад, імовірності класів у класифікації).

Параметри квантових схем так само як і нейромереж навчаються методом зворотного поширення помилки, тому функції, які реалізують VQC повиння бути гладкими і диференційовними для легкого обчислення градієнту.

2.1 Фізичні величини і базові оператори квантової алгебри

В даному підрозділі стисло викладений матеріал, який використовується для формалізації елементів квантово-посиленого трансформера. Детально теорія квантової механіки описана в [19].

2.1.1 Кубіт

Базова одиниця квантової інформації – це кубіт:

$$|0\rangle = \begin{bmatrix}1\\0\end{bmatrix},\ |1\rangle = \begin{bmatrix}0\\1\end{bmatrix}. \tag{2.1}$$

У квантовій механіці, стан кубіта ψ або будь-якого квантового об'єкта можна описати як вектор (або суперпозицію векторів) у квантовому Гільбертовому просторі:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{2.2}$$

де α та β є комплексними числами, що є амплітудами для станів (базових векторів) $|0\rangle$ та $|1\rangle$ відповідно.

Вони задають ймовірності вимірювання кубіта в одному з двох базових станів.

Сума ймовірностей вимірювання кубіта в стані $|0\rangle$ або в стані $|1\rangle$ має бути рівною 1.

$$|\alpha|^2 + |\beta|^2 = 1. \quad (2.3)$$

Гільбертовий простір – це математичний векторний простір, що використовується для опису квантових станів у квантовій механіці. Він є узагальненням класичного евклідового простору на випадок комплексних чисел і має важливі властивості, що дозволяють описувати операції з квантовими системами.

Гільбертовий простір має дві основні характеристики:

– скалярний добуток: кожен вектор у просторі можна порівнювати з іншими за допомогою операції скалярного добутку. Це дозволяє визначити, чи є два вектори ортогональними (тобто перпендикулярними);

– нормованість: існує норма вектора, що дозволяє визначити його довжину або величину.

Ортогональність векторів у Гільбертовому просторі використовується для визначення незалежних квантових станів.

Щоб представити кубіт як вектор у тривимірному просторі використовується сфера Блоха.

Це сфера радіусом 1 з центром у початку координат. Тоді кубіт $|\psi\rangle$ – це одиничний вектор з початком у 0 та кінцем на поверхні сфери. Цей вектор характеризується двома кутами повороту, як показано на рисунку 2.2.

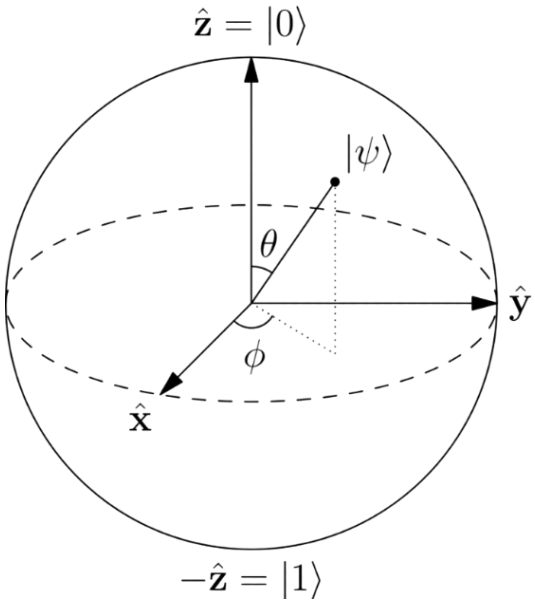


Рисунок 2.2 – Подання кубіта на сфері Блоха [20]

2.1.2 Оператори з квантовими системами

У квантовій механіці кожна фізична величина описується оператором. Наприклад, спін кубіта вздовж осі Z описується оператором квантові ворота Паулі Z, енергія квантової системи – оператором Гамільтона H, імпульс – оператором Імпульсу P.

Фізичні величини не представлені просто числами – вони представлені операторами.

Квантові оператори (іноді їх ще називають гейтами) здійснюються як лінійні операції над векторами квантового простору.

Базовими операторами над кубітами є ворота Паулі (X,Y,Z), параметризовані поворотні ворота ($R_x(\theta)$, $R_y(\theta)$, $R_z(\theta)$), $CNOT$, NOT, cR_x, cR_y, cR_z, загальні ворота ротації $Rot(\alpha, \beta, \gamma)$ та ін.

Кожен оператор представляється матрицею. Застосування оператора до вектора квантового простору записується як у формулі (2.4):

$$Operator(parameters) \mid \psi\rangle, \tag{2.4}$$

де $Operator(parameters)$ – це довільний квантовий оператор з відповідним йому набором параметрів.

Оператори Ворота Паулі (2.5):

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, Z = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad (2.5)$$

Ворота Паулі *X*, наприклад, діють на кубіт як класична операція NOT, тобто вони міняють місцями базові стани $|0\rangle$ і $|1\rangle$ (2.6):

$$X|0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle. \quad (2.6)$$

Поворотні ворота здійснюють обертання вектора стану кубіта навколо однієї з основних осей (*X*, *Y*, *Z*) сфери Блоха (сфери радіусом 1 з початком координат в центрі сфери), що еквівалентно унітарному перетворенню у відповідному двовимірному гільбертовому просторі.

Приклад поворотних воріт (поворот навколо осі *y*)(2.7):

$$R_y(\theta) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix}. \quad (2.7)$$

cNOT контрольовані ворота – створюють заплутаність (entangling) двох кубітів контрольного та цільового. Якщо контрольний кубіт (позначається на схемах як ●) має стан $|1\rangle$, тоді на цільовому кубіті (позначається на схемах як X) застосовуються ворота Паулі-*X* (тобто, стан цільового кубіта змінюється на протилежний).

Загальні ворота ротації $Rot(\alpha, \beta, \gamma)$ здійснює довільне однокубітне унітарне перетворення. Його використанням можна орієнтувати вектор

кубіта так, що він буде повернутий до будь-якої цільової точки на одиничній сфері Блоха, тобто будь-якого можливого стану кубіта (2.8):

$$Rot(\alpha, \beta, \gamma) = R_z(\alpha) R_y(\beta) R_z(\gamma). \quad (2.8)$$

Через таку властивість *Rot* часто використовується як найпростіший елемент квантової схеми, який завдяки підбору параметрів α, β та γ в процесі навчання дозволяє підібрати довільну функцію перетворення.

Тензорний добуток – оператор, що використовується для реалізації багатокубітових систем.

Якщо є дві матриці $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ та $B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$, то їх тензорний добуток обчислюється як (2.9):

$$A \oplus B = \begin{bmatrix} \mathbf{a_{11}} b_{11} & \mathbf{a_{11}} b_{12} & \mathbf{a_{12}} b_{11} & \mathbf{a_{12}} b_{12} \\ \mathbf{a_{11}} b_{21} & \mathbf{a_{11}} b_{22} & \mathbf{a_{12}} b_{21} & \mathbf{a_{12}} b_{22} \\ \mathbf{a_{21}} b_{11} & \mathbf{a_{21}} b_{12} & \mathbf{a_{22}} b_{11} & \mathbf{a_{22}} b_{12} \\ \mathbf{a_{21}} b_{21} & \mathbf{a_{21}} b_{22} & \mathbf{a_{22}} b_{21} & \mathbf{a_{22}} b_{22} \end{bmatrix}. \quad (2.9)$$

Наприклад (2.10):

$$|01\rangle = |0\rangle \oplus |1\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \times 0 \\ 1 \times 1 \\ 0 \times 0 \\ 0 \times 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}. \quad (2.10)$$

Тензорний добуток створює новий простір, розмірність якого є добутком розмірностей початкових просторів. Наприклад, якщо є два простори *V* і *W* з розмірами dim*(V)* = (*n*,*m*) і dim(*W)* = (*k*,*l*), тоді (2.11):

$$\mathrm{Dim}(V \oplus W) = (n \times k, m \times l). \quad (2.11)$$

Таким чином, тензорний добуток розширює простір станів, створюючи багатовимірний простір для опису складних багатокомпонентних систем.

Найчастіше кожне число типу float з входу нейрона перед подачею на квантову схему кодується у кут обертання одного кубіта (тобто на одне число входу – один кубіт). Це початковий стан кубіта перед початком роботи квантової схеми. Таке кодування лише відносно співставляє значення одного числа до кута повороту.

Але якщо на вхід квантової схеми необхідно подавати високовимірний класичний вектор у сотні довільних чисел і потрібно передати з високою точністю всю інформацію про кожен елемент цього вектора у квантовий стан, то кутове кодування неможливе (фізично) або неефективне. Таке трапляється, якщо необхідно закодувати аналоговий сигнал або виконати перетворення Фур'є. В цьому випадку замість перетворення 1 число => 1 кубіт (через кут) застосовують кодування через багатокубітну схему, де кожен елемент вектора стає амплітудою одного з елементів багатокубітної системи.

Наприклад, для передачі вектора $\overline{x} = [x_0, x_1, x_2, x_3]$ можна використати таке кодування (2.12):

$$|x\rangle = x_0|00\rangle + x_1|01\rangle + x_2|10\rangle + x_3|11\rangle, \qquad (2.12)$$

де елементи вектора $\overline{x}$ стають амплітудами двокубітних базисів (базисних станів) $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$.

Тобто для кодування 4 чисел знадобилося тензорний добуток 2-х кубітів. А потім серія квантових операторів поступово виставляє амплітуди кожного багатокубітного базису $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ відповідно до компонент вектора $|x\rangle$. Таким чином за допомогою тензорного добутку можна без втрати точності закодувати дані великих розмірів, а саме 2^n чисел у n кубітах (бо в n кубітів буде 2^n пар комбінацій їх станів 0 та 1).

З точки зору реалізації схема виставлення тензорного добутку в певний стан – це затратна глибока квантова схема з багатьма операціями, часто з багатокубітними гейтами типу CNOT. Тому амплітудне кодування часто використовується як «ідеальний» або «теоретичний» спосіб обчислення теоретичої оцінки складності і у практичних моделях квантово посилених трансформерів використовується дуже рідко.

2.1.3 Очікуване значення оператора у квантовій механіці

Теоретично, кубіт має нескінченну кількість можливих станів (за рахунок безперервних параметрів – кутів θ та φ на сфері Блоха, рисунок 2.2), що відповідає числу з плаваючою точкою великої точності. Але насправді це перетворення тільки одностороннє з числа до стану кубіта. Таке перетворення використовується для виставлення початкового стану кубіта перед початком роботи квантової схеми.

Зворотне перетворення (вимірювання стану кубіта) дає лише 0 або 1. Вимірювання – це дія, під час якої знищується вся квантова інформацію про суперпозицію (параметри α та β у формулах (2.2) та (2.3)) і повертається конкретний класичний результат: 0 або 1, який відповідає лише 1 біту інформаццїї.

Тому для зворотного перетворення інформації з виходу квантової схеми на вхід нейрона наступного шару часто використовують інші вимірювання, наприклад, очікуване значення операторів над кубітами.

Очікуване значення оператора, що відповідає вимірюваній фізичній величині, у квантовій механіці – це усереднене значення результатів вимірювання певної фізичної величини у заданому квантовому стані. Воно обчислюється як (2.13):

$$\langle O \rangle = \langle \psi \mid O | \psi \rangle, \tag{2.13}$$

де $|\psi\rangle$ – квантовий стан системи;

O – оператор, що відповідає вимірюваній фізичній величині;

$\langle\psi|$ – спряжений стан (трансформований) до $|\psi\rangle$.

Очікуване значення вказує, який результат в середньому можна отримати, якщо багато разів повторювати вимірювання певної величини у тому самому квантовому стані.

Обчислимо, наприклад, очікуване значення оператора Паулі Z $\langle\psi|\text{Паулі}Z|\psi\rangle$, для деякого кубіта з вектором $\psi = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$. Для цього перетворимо ψ у матричний вигляд (2.14):

$$\begin{gathered}\Psi = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}} \times (|0\rangle + |1\rangle) = \frac{1}{\sqrt{2}}\left(\begin{bmatrix}1\\0\end{bmatrix} + \begin{bmatrix}0\\1\end{bmatrix}\right) = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix} \\ Z|\psi\rangle = \begin{bmatrix}1 & 0\\0 & -1\end{bmatrix} \times \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix} \\ \langle\psi = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}^T = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\end{bmatrix} \\ \langle Z\rangle = \tfrac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\end{bmatrix} \times \tfrac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix} = \tfrac{1}{2}(1-1) = 0.\end{gathered} \tag{2.14}$$

Взагалі маємо $\langle 0 \mid Z|0\rangle = +1$, для $\langle 1 \mid Z|1\rangle = -1$. Тобто діапазон оператора $\langle Z\rangle$ – це [-1,+1].

В квантових нейронних мережах саме $\langle Z\rangle$ зазвичай використовується для перетворення стану кожного кубіта після роботи квантового блоку до числа в діапазоні [–1,+1], яке потім подається на вхід наступного шару трансформера.

2.2 Математична модель FeedForward блоку

Різні схеми реалізації шарів нейронної мережі відрізняються глибиною та складністю операцій (гейтів) між кубітами.

Глибина квантової схеми визначається як довжина найдовшого шляху від входу до виходу в схемі, де шлях складається з послідовно виконуваних гейтів. Це означає, що глибина – це максимальна кількість гейтів, які діють послідовно на одному кубіті, враховуючи, що деякі гейти можуть виконуватися паралельно на різних кубітах.

Зменшення глибини схеми – це ключове завдання при розробці ефективних квантових алгоритмів (особливо для поточних квантових пристроїв з обмеженими можливостями). Глибина схеми впливає на тривалість виконання обчислення на квантовому комп'ютері. Також менша глибина зменшує ймовірність накопичення помилок.

У дослідженнях, проаналізованих в даній роботі, застосовують декілька варіантів реалізації FeedForward шару класичної нейронної мережі:

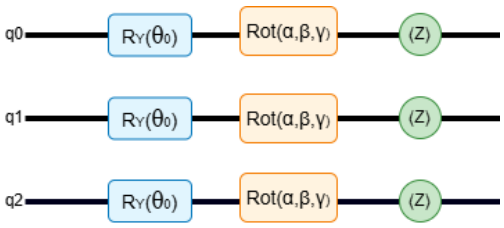
– амплітудне кодування+різні квантові гейти (*cY*, *NOT* та ін) [21]. Вхідні дані кодуються безпосередньо у амплітуди квантових станів, дозволяючи ефективно кодувати більше інформації з меншою кількістю кубітів. Недолік цієї схеми – складність реалізації (через необхідність точно контролювати квантовий стан) та обмеженність на розмірність і нормування вхідних даних;

– параметричні поворотні ворота+загальні ворота ротації [22] (рисунок 2.2 а)). Це класична проста для реалізації схема, що швидко обчислюється на квантових емуляторах. Схема досить гнучка, дозволяє охопити широкий спектр трансформацій квантових станів, що важливо для якісного навчання. Це перевірена часом схема, яка довела ефективність у багатьох дослідженнях і публікаціях.

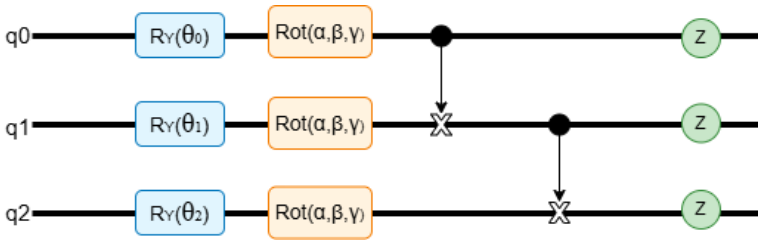
– параметричні поворотні ворота+шари сильної заплутаності [23] (див. рисунок 2.2 б)). Етап Rot+CNOT на схемі може поворюватися багато разів, кількість повторювань задається гіперпараметром блоку. На рисунку 2.2 б) зображено лише 1 повторювання етапу Rot+CNOT. параметричні поворотні ворота+шари сильної заплутаності – це потужна варіаційна схема з сильним

переплутуванням (entanglement). Її недолік – висока складність обчислень. Але як показують дослідження [23] вона може покращити результати у задачах класифікації або складних даних.

Схематично варіанти 1 та 3 зображено на рисунках 2.3 а) та б) відповідно. На обох рисунках вхідні значення спочатку кодуються як кути повороту навколо осі Y відповідно до сфери Блоха, потім починає діяти схема квантової трансформації, а результат кожного кубіта після роботи схеми декодується через очікуване значення оператора Паулі Z.



а) амплітудне кодування+різні квантові гейти



б) параметричні поворотні ворота+шари сильної заплутаності

Рисунок 2.3 – Варіанти реалізації квантового блоку FeedForward трансформера: а) (рисунок виконано самостійно)

Далі в роботі для реалізації FeedForward блоку буде використано варіант параметричні поворотні ворота + шари сильної заплутаності. Це середній за складністю варіант, який добре зарекомендував себе в останніх дослідницьких роботах.

3 ПРОГРАМНІ БІБЛІОТЕКИ ДЛЯ РЕАЛІЗАЦІЇ КВАНТОВИХ ОБЧИСЛЕНЬ

Квантові обчислення швидко розвиваються як нова парадигма обчислювальної техніки, обіцяючи розв'язання задач надзвичайно високої складності, недоступних для класичних комп'ютерів. Для дослідження можливостей квантових обчислень важливо мати програмні інструменти, що дозволяють емуляцію їх на наявних класичних комп'ютерах, адже сучасні квантові процесори все ще мають обмежену кількість кубітів і високий рівень шумів. Для дослідження цих можливостей важливо мати програмні інструменти, що дозволяють емуляцію квантових обчислень на наявних класичних комп'ютерах, адже сучасні квантові процесори все ще мають обмежену кількість кубітів і високий рівень шумів. Це найбільш популярні зараз бібліотеки та фреймворки: Qiskit, Microsoft QDK та PennyLane.

Qiskit [24] – це відкритий програмний фреймворк для квантових обчислень, створений компанією IBM у 2017 році. Нині він є одним із найпопулярніших SDK квантового програмування, широко вживаним в промисловості

Він написаний мовою Python з використанням високопродуктивних компонентів на C++ для критичних частин, що забезпечує ефективність виконання.

Архітектурно Qiskit поділений на кілька модулів: Terra – ядро для створення квантових схем, реалізації базових операцій та компіляції (транспіляції) схем під конкретне апаратне забезпечення; Aer – модуль високопродуктивних симуляторів квантових схем (стан-векторів, унітарної еволюції, а також симуляції шумів і декогеренції), Ignis – набір інструментів для характеризації шумів та пом'якшення їх впливу, Aqua – бібліотеки застосувань, зокрема Qiskit Machine Learning, Nature,

Optimization тощо, що надають реалізації алгоритмів для хімії, оптимізації, AI і інших доменів

Qiskit дозволяє як створювати і відлагоджувати квантові алгоритми на симуляторах, так і виконувати їх на справжніх квантових процесорах IBM. Однією з ключових переваг Qiskit є тісна інтеграція з хмарною платформою IBM Quantum Experience, через яку користувач може відправити створену схему на реальні квантові пристрої IBM (5-, 7-кубітні тощо) і отримати результати виконання

Qiskit вирізняється зрілістю і всеосяжністю екосистеми. Він надає повний стек – від рівня базових квантових логічних елементів до алгоритмів прикладного рівня – і дозволяє виконувати експерименти на реальному квантовому обладнанні IBM з мінімальними зусиллями

Симулятори Qiskit Aer написані на C++ і оптимізовані для максимального використання ресурсів CPU і GPU, завдяки чому продуктивність емуляції є однією з найвищих серед подібних пакетів

Qiskit також має багаті можливості для роботи з шумовими моделями та квантовими помилками (через розширення Aer та модуль Ignis), що дозволяє досліджувати роботу алгоритмів у режимі NISQ (шумних квантових пристроїв).

Основні обмеження Qiskit пов’язані з тим, що він не був першопочатково орієнтований на інтеграцію з платформами глибокого навчання. Хоча Qiskit містить модуль Machine Learning та підтримує реалізацію варіаційних алгоритмів (VQE та QAOA), процес оптимізації параметрів здійснюється через класичний цикл Python-коду (з використанням оптимізаторів SciPy), а не безпосередньо через фреймворки на кшталт PyTorch чи TensorFlow. На відміну від PennyLane відсутня нативна підтримка автоматичного диференціювання квантових схем; для інтеграції з популярними ML-бібліотеками потрібні додаткові обгортки або використання зовнішніх пакетів. Таким чином, для задач саме квантового deep learning Qiskit може поступатися спеціалізованим інструментам. Також

варто відзначити, що хоч документація Qiskit і багата, но бібліотека може бути складною для користувачів без базових знань квантової механіки – через необхідність розуміти концепції квантових схем, кубітів, гейтів і т.д.

Microsoft Quantum Development Kit та Q# [25] – це набір інструментів для квантового програмування від Microsoft, центральним компонентом якого є мова програмування Q# яка була представлена у 2017 році як спеціалізована доменно-орієнтована мова для квантових алгоритмів. На відміну від Qiskit та PennyLane, що є по суті Python-бібліотеками, Q# – самостійна мова з власним синтаксисом (чимось схожим на C#/Python) і компілятором. Вона призначена для того, щоби задавати квантові обчислення на високому рівні абстракції, поєднуючи їх з класичними обчисленнями у єдиній програмі. Код на Q# зазвичай запускається або через спеціальні блоки Jupyter-ноутбуку, або викликається із «хост-програми» на Python.

Мова Q# спроєктована з урахуванням особливостей квантового програмування. В ній введені власні типи даних (кубіт, результат вимірювання, масиви кубітів тощо) і ключові слова для квантових операцій. Q# має розвинену систему базових квантових єлементів: всі типові однокубітні та двокубітні гейти, операції обертання, готові підпрограми для реалізації оракулів, перетворення Фур'є та інші. Одна з сильних сторін Q# – підтримка класичного програмного керування всередині квантового коду: мова дозволяє виконувати вимірювання кубіта під час алгоритму і в залежності від отриманого класичного біта умовно виконувати ті чи інші квантові операції (if-else), або повторювати квантові кроки в циклах *while* до виконання певної умови. Такі конструкції потрібні для алгоритмів з корекцією помилок чи адаптивних протоколів. У Q# це робиться засобами самої мови, що ближче до реального майбутнього виконання на квантовому комп'ютері. Ще одна риса Q# – строга статична типізація і захист від квантово-некоректних операцій. Q# автоматично виділяє і звільняє кубіти в

пам’яті, що полегшує управління ресурсами, наприклад, у довгих алгоритмах з багатьма тимчасовими кубітами.

Оскільки повноцінні квантові комп’ютери великого масштабу поки недоступні, програми на Q# зазвичай виконуються на класичних симуляторах. Microsoft QDK включає кілька симуляторів квантових схем: основний Full State Simulator здатен моделювати еволюцію стану ~30 кубітів на звичайному комп’ютері, а на потужних машинах в Azure може дійти і до 32–40 кубіт за рахунок великих обсягів пам’яті. Також є Toffoli Simulator – оптимізований симулятор для схем, що складаються лише з нелінійних операцій X, CNOT (аналоги класичних XOR), який може моделювати тисячі кубітів, якщо схема представляє собою фактично класичний обчислювальний процес. Крім того, QDK має Resource Estimator – утиліту, що аналізує код Q# і вираховує, скільки логічних/фізичних кубітів та гейтів знадобиться для виконання алгоритму, що особливо корисно для алгоритмів, призначених для майбутніх fault-tolerant комп’ютерів.

Для доступу до реальних пристроїв Q# інтегрується з платформою Azure Quantum – хмарним сервісом Microsoft для оркестрування квантових завдань. Через Azure Quantum можна запускати Q#-програми на наявних комерційних квантових процесорах. Таким чином, екосистема Microsoft підтримує повний цикл: написання алгоритму в Q#, його тестування на локальному симуляторі, отримання оцінки ресурсів, а далі – відправка на виконання у хмару, де вибирається реальний квантовий бекенд.

Q# вирізняється тим, що закладає фундамент на майбутнє – на епоху універсальних квантових комп’ютерів з корекцією помилок. Ця мова спроектована таким чином, щоб було зручно описувати складні квантові алгоритми з класичним керуванням, які знадобляться при масштабуванні. Q# вже зараз дозволяє відпрацьовувати підходи, які знадобляться для обчислень на реальних кубітах.

У Q# основний бар'єр це необхідність вивчення окремої мови програмування.

PennyLane [26] – це відкритий фреймворк, розроблений канадською компанією Xanadu, який має унікальний фокус на квантовому машинному навчанні та гібридних класично-квантових обчисленнях. На відміну від «класичних» інструментів, орієнтованих суто на реалізацію квантових алгоритмів, PennyLane задумано як міст між квантовими обчисленнями і сучасними засобами машинного навчання. Бібліотека побудована за принципом диференційованого програмування: квантова схема розглядається як функція, яку можна включити в модель машинного навчання і обчислювати її градієнти для налаштування параметрів. PennyLane написана на Python і є hardware-agnostic: вона підтримує виконання квантових програм на різних бекендах через систему плагінів. Зокрема, PennyLane може використовувати симулятори й реальні пристрої таких платформ, як IBM Qiskit, Rigetti Forest, Google Cirq, а також власну платформу Xanadu Strawberry Fields для фотонних квантових обчислень.

Головна особливість PennyLane – підтримка варіаційних квантових алгоритмів і пов'язаних з ними задач оптимізації. Бібліотека забезпечує автоматичне обчислення градієнтів квантових схем за їх параметрами, використовуючи метод параметричного зсуву або інші техніки, що еквівалентно реалізації backpropagation для квантових обчислень. Користувач може визначити квантову підпрограму, наприклад, схему з деякими параметрами кутів поворотів і включити її у склад ширшої моделі машинного навчання. PennyLane пропонує інтерфейси до популярних ML-фреймворків: NumPy, PyTorch, TensorFlow, JAX. Це означає, що вихід квантової частини можна трактувати як тензор у класичній нейромережі, а параметри квантових гейтів – як змінні, що оптимізуються градієнтними методами класичного оптимізатора. Таким чином, PennyLane дозволяє будувати гібридні моделі (наприклад, класифікатор, де частина ознак генерується квантовою схемою, а подальша обробка здійснюється

класичним нейронним шаром) і навчати всю систему end-to-end методами глибокого навчання.

PennyLane сьогодні є одним із основних інструментів для досліджень у сфері квантового машинного навчання. Його вибирають у випадках, коли задача потребує оптимізації параметрів квантової схеми – будь то пошук основного стану молекули, навчання квантового класифікатора чи реалізація варіаційного квантового розв'язувача задач оптимізації. PennyLane значно спрощує таку роботу, автоматизуючи обчислення похідних і узгоджуючи типи даних з класичними бібліотеками ML

Попри універсальність, PennyLane не завжди є оптимальним вибором. Якщо стоїть задача реалізувати непараметричний квантовий алгоритм без потреби в навчанні параметрів, то можливості цієї бібліотеки не потрібні. У таких випадках пряме використання Qiskit може бути кращім, ніж обгортання в PennyLane.

Вважаючи усю цю інформацію, для розробки елементів квантово посиленого трансформера з необхідністю навчання найбільш зручною та простою в інтеграції є бібліотека PennyLane, яка і буде використана в даній роботі.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

4.1 Опис прикладної задачі

Для демонстрації можливостей квантово посиленого трансформера необхідно обрати одну з класичних задач, для реалізації якої трансформери вже добре себе зарекомендували. Одною з таких задач є аналіз сентименту тексту. Ця задача добре вивчена, для неї є багато доступних для тестування онлайн наборів даних (таблиця 4.1). Для розв'язання такої задачі підходить найпростіша модель трансформера, яка складається лише з енкодера, наприклад BERT.

Таблиця 4.1 – Порівняння загальнодоступних наборів даних для сентимент аналізу

Назва датасету	Обсяг	Формат / Характеристики
IMDB Movie Reviews Dataset	50,000 рецензій	тексти оглядів фільмів(довгі абзаци); мова англійська; класифікація (позитив/негатив)
Sentiment140 Dataset	1,600,000 твітів	короткі повідомлення з Twitter (до 140 символів); мова англійська; класифікація (позитив/негатив/нейтральний); має шум у мітках
Amazon Reviews for Sentiment Analysis	>200,000 оглядів	відгуки про товари; мова англійська; класифікація (1–2 = негатив, 4–5 = позитив)
Yelp Reviews Polarity	560,000 рецензій	тексти користувачів про заклади (ресторани, кафе тощо); мова англійська; класифікація (двокласова версія без 3-х зіркових «нейтральних» відгуків)

Продовження таблиці 4.1

Stanford Sentiment Treebank	~11,855 коротких речень	короткі фрази з кінооглядів; мова англійська; анотовані вручну за чіткою семантичною структурою (дерева синтаксису); класифікація позитив/негатив без нейтральних прикладів

Найбільш простим і зручним варіантом для демонстрації роботи трансформера є IMDB Movie Reviews Dataset, оскільки він дуже популярний і легко доступний, достатнього обсягу для демонстраційних прикладів та є багато готових прикладів коду для роботи з ним.

4.2 Завантаження даних для тренування та перевірки

Стандартний набір IMDb Movie Reviews (v1) завантажується з локального архіву або за відсутності через інтернет через urllib.request.urlretrieve.

Для тренування беруться лише позитивні і негативні відгуки. В об’єкті train_data міститься:

- train_data.data – список текстів відгуків;
- train_data.target – відповідні мітки (0 або 1).

На кожну ітерацію навчання випадково вибирається 2000 відгуків. Далі, із цієї підвибірки відбувається розділення на тренувальну та валідаційну частини (80% і 20% відповідно) за допомогою train_test_split.

Для токенізації тексту використовується стандартний навчений токенізатор BERT. Для кожної частини (train_texts, val_texts) тексти кодуються у числові тензори. Кожен текстовий приклад при кодуванні перетворюється на послідовність із максимум 128 токенів. Таке число як компроміс між захопленням достатнього контексту та економією ресурсів.

Для аналізу тональності зазвичай 128 токенів вистачає, щоб передати основний зміст відгуку.

Результат токенізації – словники з тензорами:

- input_ids – закодовані номери токенів;
- attention_mask – маска видимості (1 – токен враховується, 0 – паддінг, якщо відгук був коротший за 128 токенів).

Далі всі тензори агрегуються у стандартний PyTorch TensorDataset. Дані подаються у модель батчами по 16 прикладів через DataLoader. Всі 16 прикладів одного батча подаються на вхід моделі за одну ітерацію оптимізації. Для них обчислюється середнє функції втрат, на її основі обчислюється градієнт та оновлюються ваги моделі. Розмір батча задається змінною batch_size.

Саме токени вхідного тексту кожного прикладу одночасно подаються на вхідний шар трансформера.

4.3 Обгортка-трансформер

Обгортка-трансформер реалізована у вигляді окремого PyTorch-модуля (QuantumBERT), який інкапсулює стандартну модель BERT, додає квантовий шар та фінальний шар класифікації.

В якості ядра береться готова модель BERT з бібліотеки HuggingFace Transformers через AutoModel.from_pretrained.

BERT – це трансформерна модель, яка на вході отримує токенізований текст і на виході повертає тензор прихованих станів для кожного токена в реченні. Оскільки в роботі вирішуються задача класифікації, то з виходу BERT береться агрегований з усього тексту вектор ознак, представлений [CLS]-токеном.

Далі цей агрегований токен подається на вхід dropout-шару, який під час навчання з імовірністю 0.3 відключає частину нейронів, щоб запобігти перенавчанню. В проєкті квантовий шар працює з вектором фіксованої

довжини 8 кубітів (змінна n_qubits), тому з pooler_output вибираються лише перші 8 компонентів (обрізка даних).

Вектор з 8 компонентів подається у кастомний модуль QuantumLayer, який виконує нелінійне перетворення ознак у квантовому просторі завдяки використанню симульованої квантової схеми, описаної у п.2.2. QuantumLayer повертає новий вектор тієї ж розмірності.

На виході результат обробляється звичайним повнозв'язним шаром, що дає логіти для двох класів.

Вся модель цілком диференційована.

4.4 Реалізація квантової схеми

Реалізація квантової схеми наведена на рисунку 4.1. Вхідні параметри схеми: inputs – класичний вектор ознак розмірності n_qubits=8 та weights – набір параметрів квантових воріт, які навчаються під час оптимізації (формат (q_depth, n_qubits, 3)). Число 3 задає кількість ітерацій етапу Rot+CNOT у StronglyEntanglingLayers.

```
@qml.qnode(qml_dev, interface='torch', diff_method="adjoint")
def quantum_circuit(inputs, weights):
    qml.AngleEmbedding(inputs, wires=range(n_qubits), rotation='Y')
    qml.StronglyEntanglingLayers(weights, wires=range(n_qubits))
    return [qml.expval(qml.PauliZ(i)) for i in range(n_qubits)]
```

Рисунок 4.1 – Реалізація квантової схеми

Функція AngleEmbedding завдяки параметру rotation=’Y’ реалізує параметричний гейт R_y. Таким чином класичний вектор переводиться у квантовий стан.

StronglyEntanglingLayers реалізує набір параметризованих воріт. Кількість шарів у StronglyEntanglingLayers дорівнює q_depth=3. Кожен шар містить параметризовані обертання $Rot(\alpha, \beta, \gamma)$ та двокубітні ворота заплутування CNOT. Параметри шарів входять у список trainable parameters PyTorch, тому навчаються під час тренування моделі.

Далі для кожного кубіта вимірюється очікуване значення оператором Паулі Z. Результатом є класичний вектор розмірності n_qubits=8.

Сумарно глибина схеми дорівнює 5.

4.5 Тренування моделі

Після підготовки даних і створення DataLoader-ів, для кожної підвибірки ініціалізується модель і обирається оптимізатор AdamW і функція втрат CrossEntropyLoss. Навчання проводиться у кілька епох. За одну епоху трансформер проходить по всіх елементах навчальної вибірки по одному разу. Для кожного батча рахується функція втрат, обчислюється градієнт та виконується оновлення ваг. Розмір батча дорівнює 16 прикладам.

CrossEntropyLoss – це функція втрат, яка вимірює відстань між передбаченим розподілом ймовірностей виходів моделі та справжніми значеннями виходів з тренувальної вибірки. Вона найчастіше використовується для задач класифікації (4.1).

$$CrossEntropyLoss = -\log(\mathrm{P}_{\mathrm{true}}). \quad (4.1)$$

де $\mathrm{P}_{\mathrm{true}}$ – ймовірність, яку модель призначила правильному класу.

Втрата усереднюється по всіх прикладах одного батчу. Чим менше значення CrossEntropyLoss, тим кращі передбачення.

AdamW – це сучасний алгоритм оптимізації, який використовується для оновлення ваг мережі під час навчання. Це різновид Adam-

оптимізатора, спеціально адаптований для правильної роботи з регуляризацією ваг. Він допомагає уникати перенавчання, поступово зменшуючи абсолютні значення ваг. На кожній ітерації навчання AdamW:

– обчислює середнє значення градієнтів (моментум); моментум допомагає рухатися по інерції у потрібному напрямку, згладжує оновлення, пришвидшує збіжність;

– обчислює середньоквадратичне значення градієнтів (адаптивність); адаптивність дозволяє кожній вазі мати свою швидкість навчання: якщо вага часто стрибає (градієнти великі) – зменшуємо крок; якщо градієнти маленькі – крок більший;

– оновлює ваги із урахуванням цих оцінок та weight decay. Weight decay трохи зменшує ваги на кожному кроці, що покращує узагальнюючу здатність моделі.

Формула оновлення градієнтів:

$$w_{new} = w_{old} - \eta \cdot grad - \eta \cdot \lambda \cdot w_{old}. \quad (4.1)$$

де η – швидкість навчання (в проєкті 5e-5), λ – коефіцієнт weight decay (в проєкті 0).

Повний код проєкту наведено у додатку А.

5 ПРОВЕДЕННЯ ЕКСПЕРИМЕНТУ

Для оцінки якості розробленої моделі було проведено 5 експериментів. Кожен експеримент – це незалежне навчання та тестування на випадково обраному наборі у 2000 відгуків з 50000 (повний обсяг датасету IMDb). Випадково обраний набір розділяється на тренувальну та валідаційну частини (80% і 20% відповідно). Під час кожного експерименту проводиться 5 епох навчання за алгоритмом, описаним в п.4.5.

Для проведення експерименту використовувалося хмарне інтерактивне середовище Kaggle Notebook з активованим графічним процесором NVIDIA Tesla T4.

Під час тренування за допомогою бібліотеки sklearn.metrics обраховується стандартний набір метрик. На кожній епосі обчислюється середня функція втрат (Train Loss) по всіх батчах тренувальної вибірки. Train Loss помітно зменшується від епохи до епохи від максимум ~0.67 (випадкові ваги квантової схеми та нових шарів трансформера та попередньо навчені ваги та ембедінги претренованого BERT) до мінімум ~0.29 (навчена модель). Це свідчить про нормальну збіжність навчання. Середнє функції втрат $\overline{TrainLoss} = 0.33$, стандартне відхилення функції втрат під час навчання $\sigma = 0.05$.

Середнє обчислюється як (5.1):

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \tag{5.1}$$

де $\bar{x}$ – середнє значення (mean);

n – кількість елементів;

x_i – і-й елемент вибірки.

Відхилення обчислюється як (5.2):

$$s = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2}, \quad (5.2)$$

де s – стандартне відхилення (standard deviation);

x_i – і-й елемент вибірки;

$\bar{x}$ — середнє значення.

Детальні результати навчання по епохах у кожному 3 п’яти експериментів наведено на рисунку 5.1.

```
=== Repeat 1/5 ===
Epoch 1/5, Train Loss: 0.6736
Epoch 2/5, Train Loss: 0.5581
Epoch 3/5, Train Loss: 0.4627
Epoch 4/5, Train Loss: 0.3858
Epoch 5/5, Train Loss: 0.3487
Val Loss: 0.4411, Accuracy: 0.8200, Precision: 0.8009, Recall: 0.8564, F1-score: 0.8278

=== Repeat 2/5 ===
Epoch 1/5, Train Loss: 0.6327
Epoch 2/5, Train Loss: 0.4336
Epoch 3/5, Train Loss: 0.3550
Epoch 4/5, Train Loss: 0.2926
Epoch 5/5, Train Loss: 0.2701
Val Loss: 0.4546, Accuracy: 0.8200, Precision: 0.7804, Recall: 0.8698, F1-score: 0.8227

=== Repeat 3/5 ===
Epoch 1/5, Train Loss: 0.6690
Epoch 2/5, Train Loss: 0.5572
Epoch 3/5, Train Loss: 0.4787
Epoch 4/5, Train Loss: 0.4449
Epoch 5/5, Train Loss: 0.4188
Val Loss: 0.5157, Accuracy: 0.7975, Precision: 0.7292, Recall: 0.9162, F1-score: 0.8121

=== Repeat 4/5 ===
Epoch 1/5, Train Loss: 0.6648
Epoch 2/5, Train Loss: 0.4857
Epoch 3/5, Train Loss: 0.4600
Epoch 4/5, Train Loss: 0.3430
Epoch 5/5, Train Loss: 0.2856
Val Loss: 0.4660, Accuracy: 0.8125, Precision: 0.7700, Recall: 0.8632, F1-score: 0.8139

=== Repeat 5/5 ===
Epoch 1/5, Train Loss: 0.6367
Epoch 2/5, Train Loss: 0.4979
Epoch 3/5, Train Loss: 0.4322
Epoch 4/5, Train Loss: 0.3812
Epoch 5/5, Train Loss: 0.3432
Val Loss: 0.6230, Accuracy: 0.7025, Precision: 0.6413, Recall: 0.9953, F1-score: 0.7800
```

Рисунок 5.1 – Статистика навчання по експериментах

Після кожного повного навчання оцінюється середнє функції втрат на валідаційній вибірці ValLoss, точність Accuracy, точність позитивних передбачень Precision, повнота Recall і F1-score – гармонійне середнє між Precision і Recall.

Accuracy показує, яка частка всіх передбачень моделі виявилася правильними (5.3).

$$\text{Accuracy} = \frac{\text{Кількість правильних передбачень}}{\text{Всього прикладів}}. \quad (5.3)$$

Precision показує скільки передбачень насправді були позитивними серед усіх прикладів, які модель позначила як позитивні (5.4).

$$\text{Precision} = \frac{TP}{TP+FP}, \quad (5.4)$$

де *TP (True Positive)* – правильно знайдені позитивні приклади;

FP (False Positive) – помилково знайдені позитивні приклади.

Recall серед усіх справжніх позитивних прикладів обраховує, скільки модель знайшла як позитивні (5.5).

$$\text{Recall} = \frac{TP}{TP+FN}, \quad (5.5)$$

де *FN (False Negative)* – позитивні, яких модель не знайшла;

F1-score – збалансована метрика, що враховує Precision і Recall разом. Висока тільки тоді, коли обидва показники високі (5.6).

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \quad (5.6)$$

Наприкінці тестування по кожній метриці обраховується середнє та відхилення. Їх значення наведено на рисунку 5.2.

```
==== Statistics over runs ====
Train Loss: mean = 0.3333, std = 0.0528
Val Loss: mean = 0.5001, std = 0.0664
Accuracy: mean = 0.7905, std = 0.0448
Precision: mean = 0.7444, std = 0.0566
Recall: mean = 0.9002, std = 0.0520
F1-score: mean = 0.8113, std = 0.0166
```

Рисунок 5.2 – Узагальнення результатів валідації

Точність класифікації ~80% – це хороший рівень для базових моделей на IMDb з малими вибірками. Низьке стандартне відхилення метрик свідчить про стабільність моделі незалежно від вибірки. Модель не переадаптується (не падає на нових валідаційних даних), бо розрив між TrainLoss і ValLoss помірний.

Було також проведено порівняння результатів квантово-посиленого трансформера з класичним такої самої архітектури і параметрів, де FeedForward блок має стандартну реалізацію.

Для цього код трансформера було змінено. Код класичного BERT з додаванням шарів Dropout та класифікації наведено на рисунку 5.3

```python
class ClassicalBERT(nn.Module):
    def __init__(self, bert_model="bert-base-uncased"):
        super().__init__()
        self.bert = AutoModel.from_pretrained(bert_model)
        self.dropout = nn.Dropout(0.3)
        self.fc = nn.Linear(n_qubits, 2)

    def forward(self, input_ids, attention_mask):
        bert_output = self.bert(input_ids, attention_mask=attention_mask).pooler_output
        bert_output = self.dropout(bert_output[:, :n_qubits])
        return self.fc(bert_output)
```

Рисунок 5.3 – Класичний BERT з шарами Dropout та класифікацією

Результат порівняння за параметром TrainLoss наведено на рисунку 5.4.

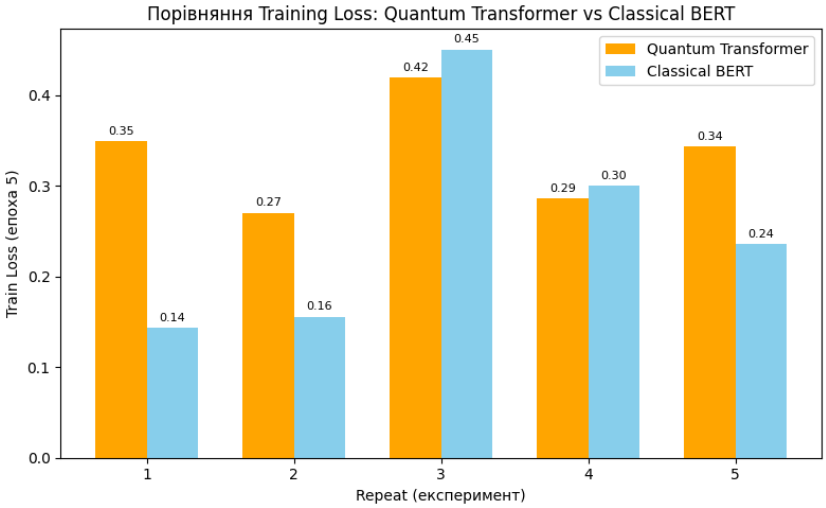


Рисунок 5.4 – Порівняння параметру Train Loss

Результат порівняння за параметрами Accuracy, Precision, Recall і F1-score наведено на рисунку 5.5.

Рисунок 5.5 – Порівняння параметрів Accuracy, Precision, Recall і F1-score

В результаті порівняння можна зробити висновок, що загальна якість обох моделей досить висока, що свідчить про ефективність обох підходів для задачі сентимент-аналізу. Класична модель BERT має трохи кращу загальну точність (accuracy) та суттєво вищу точність позитивних передбачень (precision). Це свідчить про її стабільність у визначенні позитивних прикладів. Квантова модель виграє за показником recall (повноти), тобто вона краще знаходить позитивні приклади, менше пропускаючи їх. F1-score, що є збалансованою метрикою між precision та recall, майже не відрізняється. Це означає, що порівнювані урізані (через обрізку розмірності вхідних даних) моделі однаково ефективні у задачі сентимент-аналізу.

Також проведене порівняння узагальнених результатів для різних підходів до класифікації текстів на повному (окрім QuantumFeedForward) датасеті IMDb (50 000 рецензій). Вказано середню точність класифікації та F1-score (двокласова класифікація «позитивний/негативний»), згідно з даними досліджень. Результати наведені в таблиці 5.1

Таблиця 5.1 – Порівняння узагальнених результатів для різних підходів до класифікації текстів на повному датасеті IMDb

Модель	Точність	F1-score
Логістична регресія (BoW/TF-IDF) [27]	85–89%	0.85–0.89
SVM (лінійний, TF-IDF)[28]	87%	0.87
Random Forest (дерева)[29]	84%	0.84
CNN (нейронна мережа)[27]	88–90%	0.88–0.90
LSTM (рекурентна мережа)[30]	86%	0.86
BiLSTM (двонапрямна LSTM)[30]	87–88%	0.87–0.88
BERT (трансформер)[31]	91–93%	0.91–0.92
RoBERTa (трансформер)[31]	92%	0.92
Розроблена гібридна модель з квантовим FF блоком	0.79%	~0.81

ВИСНОВКИ

У кваліфікаційній роботі було розглянуто перспективний напрямок інтеграції квантових обчислень у нейронні мережі трансформерного типу з метою покращення їх обчислювальної ефективності та якості вирішення задач штучного інтелекту.

У ході роботи було проведено ґрунтовний аналіз сучасних трансформерних моделей та виявлено їхні ключові недоліки, такі як квадратична складність механізму самоуваги, великі вимоги до пам'яті та ресурсів для навчання. Запропоновано гібридну архітектуру трансформера з використанням варіаційних квантових схем для заміни класичних блоків FeedForward. Такий підхід дозволив зменшити кількість необхідних параметрів моделі та потенційно збільшити її продуктивність.

Виконано детальний огляд сучасних програмних інструментів для квантового машинного навчання і обґрунтовано вибір бібліотеки PennyLane, яка має перевагу у простій інтеграції з популярними фреймворками глибокого навчання (в даному випадку PyTorch), та підтримує ефективне автоматичне диференціювання квантових схем.

Реалізовано гібридну модель QuantumBERT на основі класичного трансформера BERT з інтегрованим квантовим шаром, який включає AngleEmbedding і StronglyEntanglingLayers. Експериментально перевірено модель на стандартній задачі сентимент-аналізу IMDb Movie Reviews. Проведено серію експериментальних досліджень із порівнянням роботи гібридної квантово-посиленої моделі з аналогічною класичною моделлю. Експерименти продемонстрували, що гібридна модель показує близькі за ефективністю результати, зокрема трохи меншу точність позитивних передбачень Precision, але кращу повноту Recall. Це підтверджує потенціал квантових шарів для підвищення специфічних метрик якості моделей штучного інтелекту.

Статистична стабільність та порівняно низьке стандартне відхилення показників якості моделі свідчать про її стійкість до варіативності даних і хорошу узагальнюючу здатність.

Таким чином, припущення про переваги інтеграції квантових блоків у структуру класичних трансформерів частково підтвердилася. Подальший розвиток цього напрямку може бути спрямований на масштабування кількості кубітів, вдосконалення архітектури квантових шарів та їх тестування на більш складних та об'ємних наборах даних.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser Ł., Polosukhin I. Attention is all you need. Proceedings of the 31st *International Conference on Neural Information Processing Systems (NIPS'17).* Red Hook, NY, USA: Curran Associates Inc., 2017. P. 6000–6010.

2. Arabic image captioning using pre-training of deep bidirectional transformers / J. Emami et al. *Proceedings of the 15th international conference on natural language generation*, Waterville, Maine, USA

3. Peer review of GPT-4 technical report and systems card / J. Gallifant et al. *PLOS digital health*. 2024. Vol. 3, no. 1. P. e0000417.

4. Dosovitskiy A. та ін. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint.* arXiv:2010.11929. 2020.

5. Vision transformers for image classification: a comparative survey / Y. Wang et al. *Technologies*. 2025. Vol. 13, no. 1. P. 32.

6. Build Reproducible and Scalable Computational Biology Systems. URL: https://outerbounds.com/blog/reproducible-scalable-computational-biology-systems (дата звернення: 23.05.2025).

7. Efficient Transformers: A Survey / Y. Tay et al. *ACM Computing Surveys*. 2022. URL: https://doi.org/10.1145/3530811 (дата звернення: 23.05.2025).

8. Zaheer M., Guruganesh G., Dubey A., Ainslie J., Alberti C., Ontanon S., Pham P., Ravula A., Wang Q., Yang L., Ahmed A. Big bird: transformers for longer sequences. *Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS '20).* Red Hook, NY, USA: Curran Associates Inc., 2020. 1450, P. 17283–17297.

9. Beltagy I., Peters M. E., Cohan A. Longformer: The Long-Document Transformer. *arXiv: Computation and Language*. 2020. URL: https://arxiv.org/abs/2004.05150v2 (дата звернення: 23.05.2025).

10. World Economic Forum. AI and energy: Will AI help reduce emissions or increase power demand? Here's what to know. URL: https://www.weforum.org/stories/2024/07/generative-ai-energy-emissions (дата звернення: 23.05.2025).

11. Kitaev N., Kaiser Ł., Levskaya A. Reformer: The Efficient Transformer. *International Conference on Learning Representations*. 2020.

12. Choromanski K., Likhosherstov V., Dohan D. та ін. Rethinking Attention with Performers. *International Conference on Learning Representations.* 2021.

13. Beltagy I., Peters M. E., Cohan A. Longformer: The Long-Document Transformer. *arXiv: Computation and Language*. 2020. URL: https://arxiv.org/abs/2004.05150v2 (дата звернення: 23.05.2025).

14. Zaheer M. та ін. Big Bird: Transformers for Longer Sequences. *Neural Information Processing Systems*. 2020. Vol. 33. P. 17283–17297. URL: https://proceedings.neurips.cc/paper/2020/file/c8512d142a2d849725f31a9a7a361ab9-Paper.pdf (дата звернення: 01.01.2020).

15. Dao T., Fu D. Y., Ermon S., Rudra A., Ré C. FLASHATTENTION: fast and memory-efficient exact attention with IO-awareness. *Proceedings of the 36th International Conference on Neural Information Processing Systems (NIPS '22).* Red Hook, NY, USA: Curran Associates Inc., 2022. 1189. P. 16344–16359.

16. Duman Keles F., Wijewardena P. M., Hegde C. On The Computational Complexity of Self-Attention. *Proceedings of The 34th International Conference on Algorithmic Learning Theory,* ALT 2023, Singapore. PMLR. Vol. 201. P. 597–619.

17. Quantum Artificial Intelligence: A Brief Survey / M. Klusch et al. *KI - Künstliche Intelligenz*. 2024. URL: https://doi.org/10.1007/s13218-024-00871-8 (дата звернення: 25.05.2025).

18. Zhang H., Zhao Q. A Survey of Quantum Transformers: *Technical Approaches, Challenges and Outlooks.* 2025. DOI: https://doi.org/10.48550/arXiv.2504.03192 (дата звернення: 25.05.2025).

19. Griffiths D. J. Introduction to Quantum Mechanics. 3rd ed. *Cambridge University Press*, 2018. 508 с.

20. Сфера Блоха. *Вікіпедія*. URL: https://uk.wikipedia.org/wiki/Сфера_Блоха (дата звернення: 25.05.2025).

21. Time series quantum classifiers with amplitude embedding / M. P. Cuéllar et al. *Quantum Machine Intelligence*. 2023. Vol. 5, no. 2. URL: https://doi.org/10.1007/s42484-023-00133-0 (дата звернення: 25.05.2025).

22. Hybrid Quantum Cycle Generative Adversarial Network for Small Molecule Generation / M. Anoshin et al. *IEEE Transactions on Quantum Engineering*. 2024. P. 1–15. URL: https://doi.org/10.1109/tqe.2024.3414264 (date of access: 25.05.2025).

23. Suryotrisongko H., Musashi Y. Evaluating hybrid quantum-classical deep learning for cybersecurity botnet DGA detection. *Procedia Computer Science*. 2022. Vol. 197. P. 223–229. URL: https://doi.org/10.1016/j.procs.2021.12.135 (date of access: 25.05.2025).

24. Johnson P. Mastering Quantum Programming with Qiskit: A Practical Guide. HiTeX Press, 2024. 503 P.

25. Hooyberghs J. Introducing Microsoft Quantum Computing for Developers: Using the Quantum Development Kit and Q#. Apress, 2021. 495 P.

26. Hundt R. Quantum Computing for Programmers. Cambridge University Press, 2022. 350 P.

27. Derbentsev V. et al. A comparative study of deep learning models for sentiment analysis of social media texts. *Proceedings of the Selected and Revised Papers of 10th International Conference on Monitoring, Modeling & Management of Emergent Economy*. Kryvyi Rih, Ukraine, November 17–18, 2022. P.168–188.

28. Tayal N., Singh M., Raj M. A BERT-Based Technique on IMDb For False Movie Review Detection. *Electrical Systems*. 2024. Vol. 20, № 10s. P. 5322–5333.

29. Shuvo R. A. Sentiment Analysis of IMDB Dataset using RF, KNN, and MNB. URL: https://medium.com/@rajvir.ahmed.shuvo/sentiment-analysis-of-imdb-dataset-using-rf-knn-and-mnb-5dd398e556a3 (дата звернення: 23.05.2025).

30. Sentiment Analysis on IMDB Movie Reviews using Machine Learning and Deep Learning Algorithms / K. Amulya et al. *2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT)*, Tirunelveli, India, 20–22 January 2022. 2022. URL: https://doi.org/10.1109/icssit53264.2022.9716550 (date of access: 25.05.2025).

31. Comparative Analysis of Sentiment Classification on IMDB 50k Movie Reviews: A Study Using CNN, LSTM, CNN-LSTM, and BERT Models / M. T. Islam et al. *2024 IEEE International Conference on Power, Electrical, Electronics and Industrial Applications (PEEIACON)*, Rajshahi, Bangladesh, 12–13 September 2024. 2024. P. 512–517. URL: https://doi.org/10.1109/peeiacon63629.2024.10800035 (date of access: 25.05.2025).