Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему *Квантові згорткові нейронні мережі: особливості реалізації у технічних, природничих і соціально-економічних системах*

Галузь знань: *12 – Інформаційні технології*
Шифр і назва галузі знань

Спеціальність: *122 – Комп'ютерні науки*
Шифр і назва спеціальності

Освітня програма: *Комп'ютерні науки*
Назва освітньої програми

Виконала: *студентка 4 курсу, група КН-19-1* — Курс, група виконавця | Підпис | *І.М. Гринько* — Ініціали, прізвище

Керівник: *д.т.н., проф., завідувач кафедри КН* — Науковий ступінь, посада | Підпис | *О.В.Бармак* — Ініціали, прізвище

Нормоконтроль: *к.т.н., доцент кафедри КН* — Науковий ступінь, посада | Підпис | *Р.О. Багрій* — Ініціали, прізвище

До захисту допускаю:
Зав. кафедри КН, д.т.н., професор — Підпис | *О.В. Бармак* — Ініціали, прізвище
01 06 2023 р.

Хмельницький 2023

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет *інформаційних технологій*
Кафедра *комп'ютерних наук*
Освітній ступінь *бакалавр*
Галузь знань *12 – Інформаційні технології*
Спеціальність *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ

Завідувач кафедри *комп'ютерних наук*

_____
(підпис)

д.т.н., професор О.В. Бармак

«____» *2023* року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

1. Тема кваліфікаційної роботи бакалавра: *«Квантові згорткові нейронні мережі: особливості реалізації у технічних, природничих і соціально-економічних системах»*

2. Завдання видано студентці *Гринько Ірині Миколаївні*
(прізвище, ім'я, по батькові)

3. Керівник роботи *завідувач кафедри КН, Бармак Олександр Володимирович*
(посада, прізвище, ім'я, по батькові)

4. Затверджено наказом університету від «1» 03 2023 р. № 5

5. Дата видачі завдання студентці: «3» 03 2023 р.

6. Зміст пояснювальної записки (перелік задач) та вихідні дані: задачі дослідження: огляд методів та засобів штучного інтелекту та машинного навчання; огляд квантових обчислень для великої кількості даних; огляд технічних, природничих та соціально-економічних інформаційних систем, що потребують нейромережеві підходи та мають великі об'єми даних; запропонувати способи реалізації квантових згорткових нейронних мереж для технічних, природничих і соціально-економічних інформаційних систем; провести реалізацію квантових згорткових нейронних мереж для технічних, природничих і соціально-економічних інформаційних систем.

7. Календарний план виконання кваліфікаційної роботи бакалавра:

Назва етапів (розділів) кваліфікаційної роботи бакалавра	Термін виконання	Примітка
Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи бакалавра з керівником	*грудень 2022*	*виконано*
Ознайомлення з предметною областю, формулювання мети та задач дослідження, визначення об'єкта та предмета дослідження	*січень 2023*	виконано
Робота над розділом 1 – Огляд принципів роботи нейронних мереж та квантових обчислень для великого об'єму даних	*січень 2023*	виконано
Робота над розділом 2 – Способи реалізації квантових згорткових нейронних мереж для технічних, природничих і соціально-економічних інформаційних систем	*березень 2023*	виконано
Робота над розділом 3 – Програмна реалізація з застосуванням квантових згорткових нейронних мереж для технічних, природничих і соціально-економічних інформаційних систем	*квітень 2023*	виконано
Оформлення пояснювальної записки згідно вимог	*травень 2023*	виконано
Підготовка статті до журналу, попередній захист кваліфікаційної роботи бакалавра	*червень 2023*	виконано
Захист кваліфікаційної роботи бакалавра на засіданні Екзаменаційної комісії	*червень 2023*	виконано

навець: *студентка 4 курсу, група КН-19-1* *І.М. Гринько*

Курс, група виконавця — Підпис — Ініціали, прізвище

ник: *д.т.н.,проф., завідувач кафедри КН* *О.В. Бармак*

Науковий ступінь, посада — Підпис — Ініціали, прізвище

Анотація

Тема кваліфікаційної роботи бакалавра: *Квантові згорткові нейронні мережі: особливості реалізації у технічних, природничих і соціально-економічних системах*

Виконавець кваліфікаційної роботи бакалавра: *студентка групи КН-19-1 Гринько Ірина Миколаївна*

Керівник кваліфікаційної роботи бакалавра: *д.т.н., професор, завідувач кафедри КН Бармак Олександр Володимирович*

Кваліфікаційна робота бакалавра містить:

Пояснювальна записка				Кількість додатків
Сторінок	Рисунків	Таблиць	Джерел інформації	
62	26	0	19	4

Мета кваліфікаційної роботи полягає в дослідженні особливостей реалізації квантових згорткових нейронних мереж для технічних, природничих і соціально-економічних інформаційних систем з великим об'ємом даних.

Для досягнення поставленої мети визначені наступні задачі дослідження: огляд методів та засобів штучного інтелекту та машинного навчання; огляд квантових обчислень для великої кількості даних; огляд технічних, природничих та соціально-економічних інформаційних систем, що потребують нейромережеві підходи та мають великі об'єми даних; запропонувати способи реалізації квантових згорткових нейронних мереж для технічних, природничих і соціально-економічних інформаційних систем; провести реалізацію квантових згорткових нейронних мереж для технічних, природничих і соціально-економічних інформаційних систем.

Ключові слова: штучний інтелект, машинне навчання, згорткові нейронні мережі, квантові обчислення, технічні інформаційні системи, природничі інформаційні системи, соціально-економічні інформаційні системи.

Виконавець: *студентка 4 курсу, група КН-19-1* _____ *І.М. Гринько*

Курс, група виконавця | Підпис | Ініціали, прізвище

Зміст

Перелік скорочень

Скорочення, термін, позначення	Пояснення
ШІ	Штучний інтелект
CNN	Convolutional Neural Network
КРБ	Кваліфікаційна робота бакалавра
КН	Комп’ютерні науки
QCNN	Quantum Convolutional Neural Network
НСД	Найбільший спільний дільник

Вступ

Кваліфікаційна робота бакалавра присвячена дослідженню та особливостях реалізації квантових згорткових нейронних мереж у технічних, природничих і соціально-економічних інформаційних системах.

Актуальність теми. Методи та засоби штучного інтелекту стрімко розвиваються та застосовуються як у наукових дослідженнях, так і в бізнесі та соціальному житті. Завдяки ним стрімко розвиваються сучасні інформаційні технології та з’являються все нові способи використання його в щоденному житті.

Машинне навчання, підрозділ штучного інтелекту використовується як механізм надання методам та засобам штучного інтелекту можливості “навчатися”. Проте, не зважаючи на швидкий прогрес як у теоретичний підходах так і практичних реалізаціях машинного навчання, алгоритми, що для цього використовуються все ще потребують вдосконалення.

З появою квантових технологій, почалися з’являтися підходи до створення квантових нейронних мереж. Актуальним та проблематичним є розв’язування квантових завдань для різних предметних областей та типів інформаційних систем.

Мета і завдання кваліфікаційної роботи. Мета кваліфікаційної роботи полягає в дослідженні особливостей реалізації квантових згорткових нейронних мереж для технічних, природничих і соціально-економічних інформаційних систем з великим об’ємом даних.

Для досягнення поставленої мети визначені наступні завдання дослідження:

– огляд методів та засобів штучного інтелекту та машинного навчання;

– огляд квантових обчислень для великої кількості даних;

– огляд технічних, природничих та соціально-економічних інформаційних систем, що потребують нейромережеві підходи та мають великі об’єми даних;

– запропонувати способи реалізації квантових згорткових нейронних мереж для технічних, природничих і соціально-економічних інформаційних систем;

– провести реалізацію квантових згорткових нейронних мереж для технічних, природничих і соціально-економічних інформаційних систем.

Об’єкт дослідження – процес використання квантових згорткових нейронних мереж для технічних, природничих і соціально-економічних інформаційних систем.

Предмет дослідження – моделі, алгоритми та засоби квантових обчислень у згорткових нейронних мережах для використання у технічних, природничих і соціально-економічних інформаційних системах.

Розділ 1 Огляд принципів роботи нейронних мереж та квантових обчислень для великого об’єму даних

1.1 Штучний інтелект та машинне навчання

За останні десятиліття вчені, математики та інформатики дали чимало тлумачень значенню штучний інтелект. Джон Маккарті – американський інформатик, автор терміну «штучний інтелект», винахідник мови Lisp та засновник функціонального програмування пропонує таке визначення [1]: «Це наука та інженерія створення інтелектуальних машин, особливо інтелектуальних комп’ютерних програм. Це пов’язано з подібним завданням використання комп’ютерів для розуміння людського інтелекту, але ШІ не повинен обмежуватися методами, які є біологічно спостережуваними».

На сьогоднішній день даний термін має досить широке значення та багато людей вкладають з нього обширний зміст. В наш час дослідження в області штучного інтелекту та його розвитку відбуваються в різних сферах та напрямках, таких як: надбання знань, обробка образної інформації, управління системами та процесами, планування, прийняття рішень, інтелектуальний аналіз даних, моделювання міркувань, динамічні інтелектуальні системи та інше.

У своєму найпростішому визначенні штучний інтелект [2] – це набір математичних (статистичних) моделей, які натреновані для аналізу і класифікації даних. Системи ШІ працюють шляхом поєднання в собі інформатики та надійних наборів даних для вирішення поставлених проблем та завдань. Дана комбінація дає змогу штучному інтелекту навчатися на основі шаблонів і особливостей проаналізованих даних. Кожний раз, виконуючи цикл обробки даних, система штучного інтелекту перевіряє та вимірює свою результативність та використовує підсумки для розробки додаткового досвіду.

Саме за допомогою машинного навчання штучний інтелект отримує змогу навчатися. Це стається шляхом виявлення закономірностей за допомогою вже існуючих алгоритмів і генерування розуміння на основі вхідний даних. Найбільш

ефективні системи ШІ, такі як розпізнавання мовлення на телефонах та ком’ютерах або автоматичний перекладач Google, були створені завдяки методу під назвою «глибоке навчання» [3]. Даний метод є підмножиною машинного навчання. На комп’ютері це відбувається таким самим чином, як мозок людини обробляє інформацію. Людський мозок має змогу відокремити особливості порівнюючи їх із шаблонами, що були запам’ятовуванні раніше. Глибоке навчання використовує той же самий метод, що являється потужним інструментом в машинному навчанні. Одними з найпопулярніших глибоких нейронних мереж є згорткові нейронні мережі[4] в глибокому навчанні.

Одна з ключових операцій в згортковій нейронній мережі (Convolutional Neural Network, CNN) – це операція згортки (convolution) [5], яка виконується на вхідних даних (наприклад, на зображенні).

Згортка використовує фільтри або ядра (kernels), які представляють собою матриці чисел. Ці ядра скользять по вхідному зображенню з певним кроком і виконують операцію множення між відповідними пікселями зображення та відповідними елементами ядра, а потім додають результати множення. Ця операція може бути представлена у вигляді матричного множення між вектором, який представляє вихідні значення зображення, та ядром.

Результатом згортки є нова матриця, яка називається feature map або картою ознак (feature map or activation map) [6]. Карта ознак містить інформацію про те, які функції або особливості зображення були виявлені ядром під час згортки.

Крім того, згортки можуть бути застосовані не тільки на першому шарі згорткової нейронної мережі, але й на інших шарах, де ядра можуть бути більш складними та представляти собою навчені параметри. На наступному рівні обробки з цих особливостей можна розпізнати повторювані фрагменти, які пізніше можуть скластися в частини об’єкта. В результаті після обробки таким способом ми маємо змогу правильно класифікувати зображення або виокремити на кінцевому етапі потрібний фрагмент на ньому. На рис 1.1 наведено простий

приклад архітектури згорткових нейронних мереж, призначеної для класифікації чисел на зображенні.

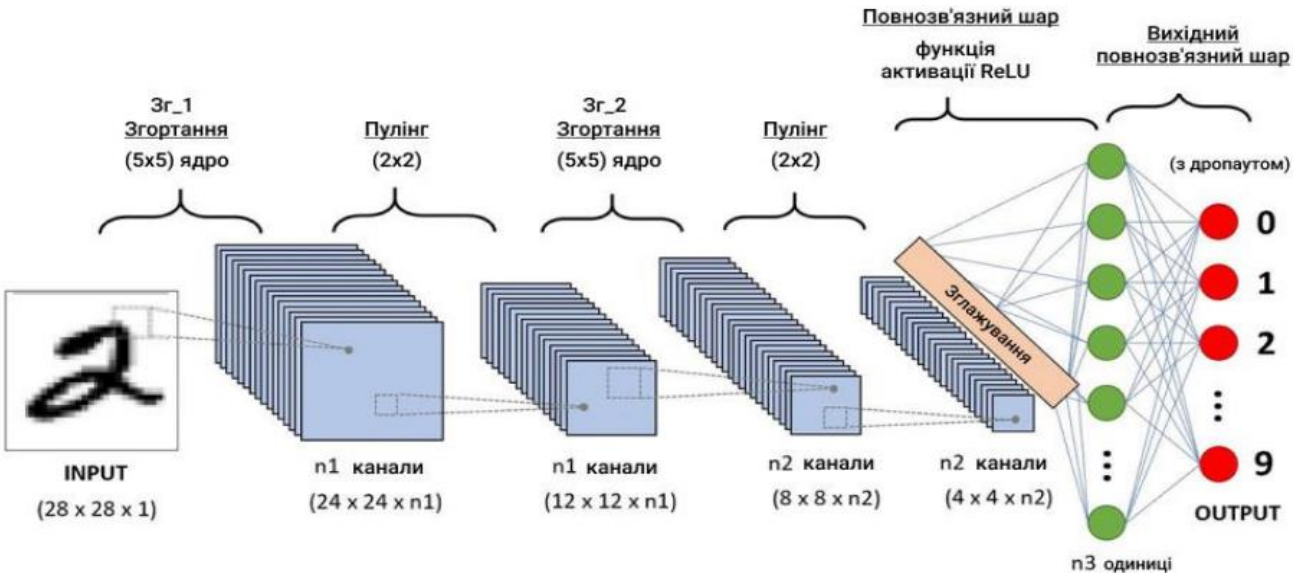


Рисунок 1.1 – Архітектура згорткової нейронної мережі [7]

Згорткова нейронна мережа є ефективним інструментом для обробки великих обсягів даних, особливо зображень. Великий обсяг даних може бути корисним для згорткової нейронної мережі з двох причин:

– Навчання: Навчання згорткової нейронної мережі потребує великого обсягу даних. Це пов'язано з тим, що згорткова нейронна мережа має багато параметрів, які потрібно налаштувати, і це може здійснюватися за допомогою навчального набору даних. Більшість згорткових нейронних мереж навчаються за допомогою наборів даних, що містять мільйони зображень;

– Точність: Великий обсяг даних може допомогти збільшити точність згорткової нейронної мережі. Більшість згорткових нейронних мереж зазвичай працюють краще з більшими наборами даних, оскільки вони можуть відтворювати більш різноманітні зображення та уникнути перенавчання.

Загалом, великий обсяг даних може допомогти згортковій нейронній мережі більш ефективно відтворювати різноманітні особливості та взаємодії між об'єктами на зображеннях, що зробить його більш точним та надійним у виконанні різних завдань обробки зображень. Але з’являється ряд проблем під час

використання великого об’єму даних для навчання. Наприклад, для обробки великого об’єму інформації потрібна більша кількість часу. Крім цього може виникнути проблема спроможності фізичних ресурсів обчислювальної техніки. Для вирішення та уникнення даних проблем використовуються квантові комп’ютери.

Квантові комп'ютери можуть потенційно виконувати певні обчислення швидше, ніж класичні комп'ютери, зокрема у випадку обчислень, пов'язаних з обробкою великих об'ємів даних [8]. Це відкриває нові можливості для використання згорткових нейронних мереж на квантових комп'ютерах. На жаль, в даний час квантові комп'ютери ще не є настільки розвиненими, щоб можна було ефективно використовувати їх для обробки зображень за допомогою згорткових нейронних мереж. Проте, дослідження у цій області продовжуються, і можливо, що у майбутньому квантові комп'ютери будуть використовуватися для збільшення швидкості та точності згорткових нейронних мереж.

Таким чином, хоча квантові комп'ютери потенційно можуть забезпечити певні переваги для згорткових нейронних мереж, на даний момент ця область досліджень залишається в основному теоретичною, і більшість застосувань згорткових нейронних мереж залишається класичними.

1.2 Квантові обчислення для великого об’єму даних

Квантові обчислення [9] – це такий тип обчислень, що базується на двух фундаментальних принципах: суперпозиція та квантова заплутаність. Якщо говорити іншими словами, це інакше обчислення, ніж те, що можна виконати на звичних для нас комп’ютерах чи пристроях.

У класичній обчислювальній техніці використовується одиниця вимірювання інформації «біт», в якій можливі лише два стани: «0» чи «1». У квантовій інформатиці за головний елемент теж взято квантовий об’єкт, який може перебувати у двох базових станах. Він отримав назву «кубіт» («qubit» або квантовий біт) [10]. На машинному рівні він немає потреби в процесорі або

пам’яті, оскільки кубіти є основною і єдиною одиницею даної технології. По своїй суті, квантові кубіти охоплюють набагато більше інформації, їх швидше обробляють і також мають вищу потужність, ніж у традиційних бітів, тому вони є найкращим варіантом для роботи, пов’язаної з обробкою великої кількість даних, моделюванням сценаріїв або масовим розрахунком ймовірностей.

Схема Блоха (рисунок 1.2) – це графічне зображення стану кубіту на сфері Блоха. Сфера Блоха є візуальною репрезентацією кубіту, де кожна точка на сфері відповідає конкретному стану кубіту [11]. Кубіти можуть перебувати у будь-якому стані на сфері Блоха, включаючи дійсні і уявні стани. Схема Блоха забезпечує зручний спосіб відображення та порівняння станів кубіту.

На схемі Блоха кубіт зображується як точка на сфері Блоха. Вертикальна ось, що проходить через центр сфери Блоха, відповідає базисним станам $|0\rangle$ та $|1\rangle$. Горизонтальні осі відображають комплексні стани кубіту.

Наприклад, якщо кубіт перебуває у стані $|0\rangle$, то він зображується на верхній полюсі сфери Блоха. Якщо кубіт перебуває у стані $(\frac{|0\rangle + |1\rangle)}{\sqrt{2}})$, то він зображується на екваторі сфери Блоха. Якщо кубіт перебуває у стані $i|0\rangle$, то він зображується на горизонтальній осі на лівому боці сфери Блоха.

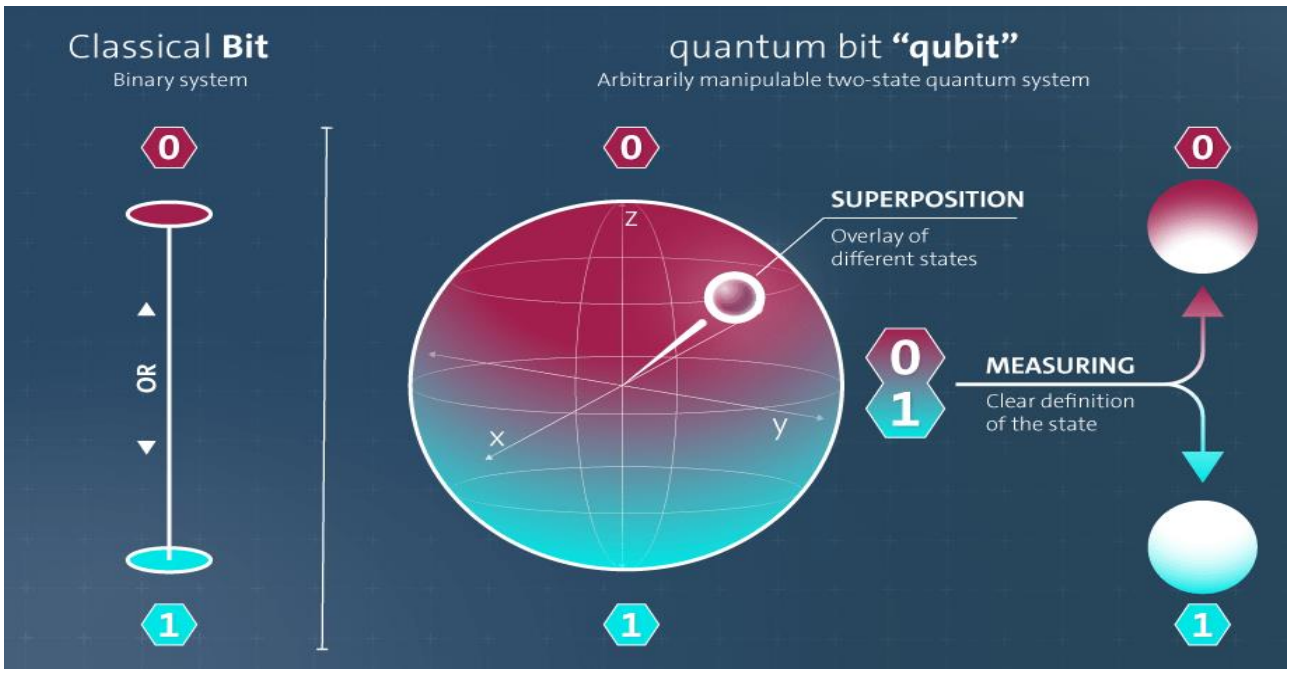


Рисунок 1.2 – Порівняння біта та кубіта та сфера Блоха [12]

Як згадувалось раніше, основними принципами квантових обчислень є квантова заплутаність та квантова суперпозиція. У квантовій фізиці «заплутаність» — це явище, під час якого дві квантові частинки, ними можуть бути фотони або електрони, якщо вони є заплутаними, то перебувають у одному стані і зв’язок між ними зберігається навіть якщо вони далеко одне від одного, в різних частинах Всесвіту [13].

Уявлення про такі заплутані стани може допомогти зрозуміти явище квантової телепортації [14]. Якщо уявити дві частинки, які рухаються в різних напрямках. Одна з них зустрічає третю частинку та з нею заплутується. У той самий час третя частинка, що була на початку без пари, передає свої характеристики без пари. Така передача має назву квантова телепортація. Явище квантової телепортації, як приклад, можна використовувати в передачі оптичним волокнам інформації за допомогою фотонів. В цілому, експерименти, що були проведені такими вченими як Цайлінгер, Аспе та Клаузер, стали основою для розроблення квантових мереж та комп'ютерів. Вони мають принципові відмінності від звичних нам комп'ютерів і дякуючи цьому здатні виконувати значно складніші обчислення.

Квантова суперпозиція [15]— допускає, що квантові частинки мають змогу перебувати в декількох станах одночасно до того часу, поки ми їх не виміряли. Дане явище можна легко зрозуміти та описати за допомогою багатьом відомого експеримента з котом Шредінгера [16]. Зміст цього експерименту полягає в тому, що у коробці з несправжнім котом знаходиться атом радіоактивного елемента і ємність з кислотою, що розіб'ється як тільки радіоактивний елемент розпадеться. Якщо колба буде розбита, то кіт вмре, але ніхто точно не може сказати, чи відбудеться розпад атому, і ніхто, включаючи кота, не має змоги на це вплинути. І тому кіт є живим та мертвим одночасно, що й відображає значення квантової суперпозиції.

Незважаючи на те, що звичайні обчислення грунтуються на абсолютній двійковій системі, квантові обчислення можуть визначати, що значення одночасно має стан «1» і «0» з різними вагами. Інакше кажучи, квантова

технологія має змогу керувати значеннями, які становлять 60% «0» та 40% «1». Це порушує правила, що встановлені в традиційних обчисленнях, що дає можливість розробляти нові алгоритми і тому є змога вирішення проблем, які раніше неможливо було зробити.

Однак в квантових обчисленнях не все так весело, через те, що стабільність та умови, які мають виконуватися для правильної та точної роботи пристрою з цією технологією, є основною перешкодою для квантових обчислень. Через що квантові системи та комп'ютери є досить складними? Теплова ентропія та шум є важливими факторами, які впливають на ефективність та точність квантових обчислень та можуть викликати труднощі [17].

У квантових системах, теплова ентропія виникає з кількох причин. По-перше, квантові біти (кубіти) можуть взаємодіяти з оточуючим середовищем, що призводить до зміни їх стану та зміни фази квантового стану. Ця зміна може спричинити збільшення кількості помилок під час обчислень. По-друге, теплова ентропія виникає з невизначеності зв'язку між квантовими бітами. Це може вплинути на збірку та контроль квантових станів, що також може призвести до помилок у квантових обчисленнях.

Для зменшення впливу теплової ентропії на квантові обчислення, необхідно забезпечити низьку температуру квантової системи та використовувати методи корекції помилок, які можуть бути викликані тепловою ентропією. Крім того, можна застосовувати техніки захисту від помилок, які можуть знизити вплив теплової ентропії на точність квантових обчислень.

Шум є серйозним викликом для квантових обчислень, оскільки він може призвести до помилок у квантових бітах (кубітах) та знизити точність результатів обчислень. Шум може виникати з різних джерел, включаючи невідомі ефекти взаємодії кубітів між собою та з оточуючим середовищем, недосконалості у приладах та електромагнітні впливи.

Для зменшення впливу шуму на квантові обчислення, розробляються різні техніки корекції помилок. Одна з них – квантова корекція помилок (QEC) [18] – використовує квантові коди для виявлення та виправлення помилок, що можуть

виникнути під час обчислень. Інша техніка – динамічне знімання шуму (DDC) – використовується для видалення шуму під час виконання квантових обчислень [19]. Окрім цього, можна зменшити вплив шуму на квантові обчислення за допомогою удосконалення квантових пристроїв та збільшення часу життя кубітів. Наприклад, збільшення часу життя кубітів може бути досягнуте шляхом зменшення взаємодії кубітів з оточуючим середовищем або застосування квантових кодів для зменшення впливу помилок на квантові обчислення.

Для отримання більш чи менш точних результатів розрахунків та контролю квантового світу шуму необхідно позбутися. Використання стохастичної метрики та теорії ймовірності під час описання щільності проходу або розподілу хвильової функціх є вирішенням даної проблеми. Дані процеси входять в спосіб роботи квантового комп'ютера. Взагалі вразливість до шуму має немалий влив на стан квантової системи та є одним із найбільших її ворогів. Розробники квантових комп'ютерів для боротьби з цьою проблемою використовують великий рефрижератор, що працює на дуже низьких температурах для урівняння теплової ентропії, вбудовуючи його в механізм.

1.3 Технічні, природничі та соціально-економічних інформаційні системи, що потребують нейромережеві підходи та мають великі об'єми даних

Квантові обчислення є новою парадигмою, яка має можливість запропонувати неабияку обчислювальну перевагу над звичайними класичними обчисленнями. Очікується, що саме вона допоможе вирішити купу важких і обчислювально нерозв'язних питань у масі сфер досліджень, таких як, наприклад, наука про дані, розробка ліків, фінанси, квантова хімія, чиста енергетика, безпечні комунікації, промислові хімічні розробки та інші. Останні роки є немалий прогрес як у створенні квантового апаратного забезпечення, так і в квантовому програмному алгоритмі, який наблизив квантові комп'ютери до реальності. Очікується, що дана форма обчислень обслуговуватиме сектори, в яких дані

настільки обширні та важкі, що традиційні обчислення відстають за потужністю та швидкістю.

Одним із таких сфер діяльності є сектор охорони здоров’я. Квантові обчислення відкривають можливість розв'язувати обчислювальні задачі, які не можуть бути розв'язані за допомогою існуючих звичних нам методів обчислень. Зараз вважається, що перша наука, яка отримає значний імпульс від досягнень в галузі квантових технологій, буде квантова хімія.

Немалий внесок в дану галузь зробий Річард Фейнман. Він був видатним американським фізиком, який був визнаний світовою науковою спільнотою за свій внесок у розуміння квантової механіки та її застосування у різних областях науки. Одним з таких напрямків є квантова хімія [20].

Однією з головних проблем квантової хімії є обчислення взаємодії між молекулами, що можуть бути надзвичайно складними [21]. Фейнман запропонував використання квантових комп'ютерів для розв'язання цієї проблеми. Квантові комп'ютери використовують квантові біти замість класичних бітів, що дає змогу ефективніше розв'язувати складні задачі, пов'язані з молекулярною взаємодією. У своїй книзі «The Feynman Lectures on Physics» [22], Фейнман відкрив багато принципів квантової механіки та допоміг зрозуміти фізичний зміст цих принципів. Він також досліджував проблему енергетичного спектра молекул, що дозволило розв'язати складну проблему, пов'язану з електронними рівнями в молекулах.

Сьогодні квантова хімія та квантові обчислення залишаються одними з найважливіших напрямків наукових досліджень. Вони дозволяють науковцям отримувати нові знання про структуру різних матеріалів, хімічні реакції та інші процеси на молекулярному рівні, що може привести до розробки нових матеріалів, ліків та інших продуктів хімії.

Іншою цікавою сферою розвитку та використання квантових обчислень є фінансовий сектор. В фінансовому секторі, незважаючи на те, що саме він найменш потребує такого виду технологій, якщо зосередитись на фінансових процесах, то можна сказати, що досить скоро традиційні обчислення застаріють.

Квантові обчислення можуть мати потенційний вплив на прогнозування валютного ринку. Завдяки своїй здатності до обробки великих обсягів даних та здійснення швидких обчислень, квантові комп'ютери можуть допомогти аналітикам та трейдерам отримувати більш точні та надійні прогнози валютного ринку.

Однією з областей, де квантові обчислення можуть допомогти, є аналіз великих обсягів даних про економіку та фінансовий ринок. Квантові комп'ютери можуть здійснювати аналіз таких даних з використанням складних математичних моделей, що дозволяє отримувати більш точні та надійні прогнози щодо динаміки валютних курсів. Крім того, квантові обчислення можуть використовуватись для розробки нових алгоритмів трейдингу, які забезпечують більш точну та ефективну торгівлю на фінансових ринках. Зокрема, квантові комп'ютери можуть використовуватись для розробки алгоритмів штучного інтелекту, які можуть виявляти складні зв'язки між різними фінансовими інструментами та прогнозувати їх рух.

Квантові обчислення можуть мати великий вплив на криптовалюти та блокчейн-технології, які є основою для більшості криптовалют. Однією з основних загроз криптографії з боку квантових обчислень є можливість використання алгоритмів Шора, які можуть ефективно розкладати складні числа на прості множники [23]. Це може мати наслідком, що квантові комп'ютери зможуть легко зламати багато з криптографічних алгоритмів, які використовуються в сучасних криптовалютних протоколах.

Однак, існують криптовалюти, які розроблені з урахуванням можливих загроз від квантових обчислень. Наприклад, криптовалюта Quantum Resistant Ledger (QRL) була спеціально розроблена з метою забезпечення криптографічної безпеки від квантових обчислень [24].

Однак, слід зауважити, що QCNNs є відносно новою технологією, яка потребує значних обчислювальних ресурсів та експертизи у квантових обчисленнях та машинному навчанні. Окрім того, досі не було проведено багато досліджень щодо застосування квантових згорткових нейронних у фінансах.

Тому, хоча QCNNs мають потенціал для застосування в фінансах, їх ефективність та придатність для вирішення практичних фінансових завдань потребує подальшого дослідження та випробування.

Таож квантові нейронні мережі можуть бути використані в криптографії для розв'язання різних завдань, таких як генерація випадкових чисел, підписи на основі геш-функцій та прогнозування ключів. Вони також можуть бути використані для виявлення атак на криптосистеми та захисту від них.

Одним з потенційних застосувань квантових нейронних мереж у криптографії є генерація випадкових чисел. Квантова генерація випадкових чисел використовує властивості квантових об'єктів, таких як квантові біти (qubits), для генерації випадкових чисел, які неможливо передбачити. Квантові нейронні мережі можуть використовуватися для покращення ефективності цього процесу.

Крім того, квантові нейронні мережі можуть бути використані для підписів на основі хеш-функцій. Вони можуть бути використані для створення квантових підписів, які є надійними на основі властивостей квантових об'єктів.

Квантові нейронні мережі також можуть бути використані для прогнозування ключів в криптографічних системах. Вони можуть аналізувати структуру криптографічних ключів та виявляти залежності між ними, що може допомогти у прогнозуванні нових ключів.

Зауважимо, що квантові нейронні мережі також мають певні обмеження та вимоги щодо обробки даних та кількості ресурсів, необхідних для їх роботи. Дослідники продовжують досліджувати можливості та обмеження квантових нейронних мереж в криптографії, а також намагаються знайти способи оптимізації та поліпшення їхньої ефективності та точності. Додатково, квантові нейронні мережі можуть бути використані для виявлення атак на криптосистеми та захисту від них. Наприклад, вони можуть бути використані для виявлення підробки підписів та інших криптографічних атак. Квантові нейронні мережі можуть бути натреновані на виявлення підробки та інших аномалій в поведінці користувачів та систем, що може допомогти відбирати підозрілі дії та захищати криптосистеми від атак.

Узагальнюючи, квантові нейронні мережі мають потенціал для використання в різних аспектах криптографії, від генерації випадкових чисел та підписів до виявлення атак та захисту від них. Однак, наразі це є темою активних досліджень, і більше досліджень необхідно для визначення міри ефективності та можливостей квантових нейронних мереж в криптографії.

1.4 Мета та завдання дослідження

Мета кваліфікаційної роботи полягає в дослідженні особливостей реалізації квантових згорткових нейронних мереж для технічних, природничих і соціально-економічних інформаційних систем.

Для досягнення поставленої мети визначені наступні завдання дослідження:

– огляд методів та засобів штучного інтелекту та машинного навчання;

– огляд квантових обчислень для великої кількості даних;

– огляд технічних, природничих та соціально-економічних інформаційних систем, що потребують нейромережеві підходи та мають великі об’єми даних;

– запропонувати способи реалізації квантових згорткових нейронних мереж для технічних, природничих і соціально-економічних інформаційних систем;

– провести реалізацію квантових згорткових нейронних мереж для технічних, природничих і соціально-економічних інформаційних систем.

Розділ 2 Способи реалізації квантових згорткових нейронних мереж для технічних, природничих і соціально-економічних інформаційних систем

2.1. Методи, принципи та алгоритми переходу від біта до квантового біта

Процес переходу від біта до кубіта та назад є одним з ключових аспектів квантових обчислень. У більшості випадків, біт можна розглядати як дві можливі стани: 0 або 1. Кубіт, з іншого боку, може перебувати в будь-якому стані, що є суперпозицією 0 та 1. Перехід від біта до кубіта здійснюється за допомогою процесу, який називається квантуванням. Воно полягає в тому, щоб закодувати інформацію біта в стан кубіта. Для здійснення операцій над квантовими станами, використовуються гейти в квантових обчисленнях. Декілька типових гейтів, що використовуються для перетворення біта в кубіт включають:

- Гейти Кернела Паулі (Pauli-X, Pauli-Y, Pauli-Z gates);
- Гейт контрольованої НОТ [25] (CNOT або Controlled NOT);
- Гейт Гадамарда або Адамара [26] (Hadamard gate).

Гейти Кернела Поля – це гейти, які використовуються в квантових обчисленнях для перетворення біта в кубіт. Ці гейти включають гейти Поля X, Поля Y та Поля Z, і вони є фундаментальними гейтами в квантових обчисленнях.

Матриця переходу гейту X (також відомий як NOT-гейт) має вигляд:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \tag{2.1}$$

дана матриця заміняє стани $|0\rangle$ та $|1\rangle$ на зворотні стани один відносно одного.

Матриця переходу гейту Y має вигляд:

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \tag{2.2}$$

ця матриця вводить фазовий зсув на стани $|0\rangle$ та $|1\rangle$ і одночасно замінює їх на зворотні стани один відносно одного.

Матриця переходу гейту Z має вигляд:

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \tag{2.3}$$

ця матриця замінює стани |1⟩ на стани з протилежним знаком фази, залишаючи стани |0⟩ без змін.

За допомогою гейтів X, Y та Z можна перетворити біт в кубіт. Наприклад, якщо ми беремо біт зі значенням 0, то ми можемо застосувати гейт X, щоб отримати стан |1⟩, а потім застосувати гейт Y, щоб отримати стан i|0⟩. Таким чином, ми отримали кубіт зі значенням |0⟩ та фазою pi/2.

Загалом, використання гейтів Кернела Полі X, Y та Z для перетворення біта в кубіт залежить від початкового значення біта та потрібного стану кубіту. Для отримання конкретного стану кубіту може бути необхідно використовувати декілька гейтів послідовно або комбінувати різні гейти для отримання бажаного результату.

Гейт CNOT – це квантовий гейт, який виконує операцію NOT на другому кубіті, контролюючи значення першого кубіту. Він є фундаментальним гейтом, який використовується в більш складних квантових операціях. Гейт має два входи: контрольний кубіт (control qubit) і цільовий кубіт (target qubit). Якщо контрольний кубіт має значення 1, то гейт виконує операцію NOT на цільовому кубіті, змінюючи його значення на протилежне. Якщо контрольний кубіт має значення 0, то цільовий кубіт не змінює свого значення. Графічно його можна зобразити наступною схемою:

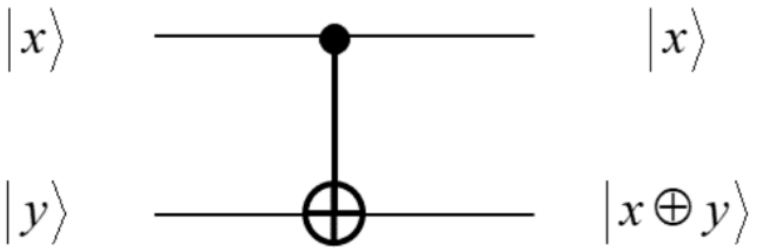


Рисунок 2.1 – Схема гейта CNOT

Контрольний кубіт зображений у вигляді точки, а цільовий кубіт – у вигляді $\oplus$. З цієї схеми видно, що якщо контрольний кубіт має стан $|0\rangle$, то цільовий кубіт не змінює свого значення, а якщо контрольний кубіт має стан $|1\rangle$, то відбувається зміна значення цільового кубіту. Матриця переходу гейту CNOT має вигляд:

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \tag{2.4}$$

Дану матрицю також можна зобразити за допомогою формули:

$$CNOT = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X, \tag{2.5}$$

де I – одинична матриця, а X – гейт Поля (Pauli X).

Операція гейту CNOT є найпростішою операцією з управленим запуском, тому її часто використовують у багатьох квантових алгоритмах.

Гейт Адамара використовується для перетворення одного біта у кубіт за допомогою умовної операції. Процес переходу від біта до кубіта за допомогою схеми Адамара відбувається наступним чином. Спочатку и маємо біт зі значенням 0 або 1, який можна представити у вигляді вектора-стовпця з двох елементів:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \tag{2.6}$$

Далі застосовуємо до цього вектора схему Адамара, що визначається наступною формулою:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \tag{2.2}$$

Тоді, якщо ми застосуємо цю матрицю до векторів $|0\rangle$ та $|1\rangle$, ми отримаємо наступні кубіти:

$$H|0\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}1\\ 0\end{pmatrix} = |\varphi\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\ 1\end{pmatrix} = |+\rangle\,,$$
$$H|1\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}0\\ 1\end{pmatrix} = |\beta\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\ -1\end{pmatrix} = |-\rangle \quad (2.7)$$

Таким чином, ми отримали кубіти $|+\rangle$ та $|-\rangle$, які є двома базисними стани кубіту. Вони представлені наступним чином:

$$|+\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\ 1\end{pmatrix},$$
$$|-\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\ -1\end{pmatrix}, \quad (2.8)$$

Перехід від кубіта до біта також здійснюється за допомогою гейта Адамара. Якщо ми застосуємо гейт Адамара до кубіта в стані $|+\rangle$ та $|-\rangle$, отримаємо біт в стані $|0\rangle$ та $|1\rangle$ відповідно:

$$H|+\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1\\ 1\end{pmatrix} = \begin{pmatrix}1\\ 0\end{pmatrix} = |0\rangle\,,$$
$$H|-\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1\\ -1\end{pmatrix} = \begin{pmatrix}0\\ 1\end{pmatrix} = |1\rangle, \quad (2.9)$$

Перетворення кубіту в біт є невідворотним процесом, оскільки в результаті вимірювання стану кубіту його стан змінюється і втрачається квантова суперпозиція.

Щоб повернутися від кубіта до біта, необхідно виконати процес вимірювання. Коли вимірюється кубіт, він переходить в один з базисних станів $|0\rangle$ або $|1\rangle$. Зазвичай вимірюванням кубіта звужується квантовий стан системи до

класичного бітового значення 0 або 1. При вимірюванні, ймовірність отримати результат |0⟩ або |1⟩ визначається квадратом амплітуди кожного стану.

Формально, при вимірюванні кубіту в квантовому стані |ψ⟩, ймовірність вимірювання |0⟩ дорівнює $|\langle 0|\psi\rangle|^2$, а ймовірність вимірювання |1⟩ дорівнює $|\langle 1|\psi\rangle|^2$.

Отже, якщо ми виміряємо кубіт у стані |0⟩, то він повернеться до значення 0, а якщо виміряємо його у стані |1⟩, то він повернеться до значення 1. Цей процес називається «колапсом хвильової функції» і є одним з основних принципів квантової механіки.

2.2 Особливості використання квантових згорткових нейронних мереж в квантовій хімії

Хімія та квантові згорткові нейронні мережі мають досить багато застосувань в науці та промисловості, зокрема в області дизайну нових матеріалів, фармацевтики, каталізу та інших галузях. Квантові згорткові нейронні мережі є розвитком класичних згорткових нейронних мереж та можуть бути використані для обробки даних в квантових обчислювальних системах.

Квантова хімія – це галузь хімії, що використовує принципи квантової механіки для опису структури, взаємодії та динаміки молекул. Оскільки квантові ефекти є важливими в хімії, вона дозволяє краще розуміти і передбачати властивості молекул та їхні реакції.

Квантові згорткові нейронні мережі (QCNNs) є новим напрямком досліджень у галузі квантової хімії. QCNNs є розвитком класичних згорткових нейронних мереж, використовуючи квантові біти для обробки даних. Квантові згорткові нейронні мережі можуть бути використані для розв'язання задач квантової хімії, таких як передбачення властивостей молекул, розуміння реакційних механізмів та розробка нових матеріалів. Однією з основних переваг QCNNs є їх здатність до моделювання взаємодії між молекулами та реакційних процесів з більшою точністю, ніж класичні методи. Крім того, QCNNs можуть

бути використані для вирішення задач в області матеріалознавства, таких як передбачення властивостей матеріалів та ефективність каталізаторів.

Незважаючи на те, що QCNNs є досить новим інструментом у галузі квантової хімії, вони вже знайшли застосування в різних дослідженнях. Наприклад, дослідники використовували QCNNs для визначення енергії зв'язку та властивостей молекул, а також для розв'язання задач в області матеріалознавства.

У галузі квантової хімії та квантових згорткових нейронних мереж (QCNNs) існує декілька фреймворків та бібліотек, які дозволяють виконувати квантові обчислення та моделювання QCNNs. Ось кілька з них:

– Pennylane – бібліотека для розвитку квантових обчислень та машинного навчання на квантових комп'ютерах. Вона має можливість створення та тренування QCNNs для задач квантової хімії та матеріалознавства. Бібліотека дозволяє вирішувати різноманітні завдання, такі як квантова хімія, оптимізація та машинне навчання, використовуючи квантові пристрої та класичні алгоритми.

– Qiskit – відкритий фреймворк для програмування квантових обчислювальних систем. Він містить бібліотеку Aqua, яка надає засоби для вирішення задач квантової хімії та матеріалознавства, включаючи знаходження енергійних спектрів, моделювання хімічних реакцій та побудову квантових машинних навчань.

– TensorFlow Quantum – фреймворк, який дозволяє розробляти та тренувати QCNNs з використанням TensorFlow. Він надає засоби для моделювання квантових систем, розв'язання задач квантової хімії та матеріалознавства, а також використання квантових обчислювальних алгоритмів для рішення класичних задач машинного навчання.

– Strawberry Fields – відкритий фреймворк для програмування та моделювання квантових систем з використанням квантових обчислень на базі фотонів. Він може бути використаний для розв'язання задач квантової хімії та матеріалознавства. Дозволяє використовувати фізичні платформи квантових обчислень, такі як квантові оптичні обчислювальні машини (QOMs) для створення та обчислення складних квантових обчислювальних графів. Strawberry

Fields надає інтерфейс для взаємодії з платформами квантових обчислень, дозволяючи користувачам створювати та запускати квантові програми, а також збирати результати обчислень. Особливість Strawberry Fields полягає в тому, що вона зосереджена на фотонних квантових обчисленнях.

Квантові алгоритми, такі як оцінка квантової фази (QPE) і варіаційний квантовий розв’язувач власних сигналів (VQE), широко вивчаються в квантовій хімії як потенційні шляхи для вирішення проблем, які нерозв’язні для звичайних комп’ютерів. Однак наразі у немає квантових комп’ютерів чи симуляторів, здатних реалізувати масштабні версії цих алгоритмів. Це ускладнює належне дослідження їх точності та ефективності для розмірів задач, де потенційно може статися фактична перевага квантових алгоритмів. Незважаючи на ці труднощі, все ще можливо виконати оцінку ресурсів, щоб оцінити, що нам потрібно для реалізації таких квантових алгоритмів.

Алгоритм VQE (Variational Quantum Eigensolver) – це квантовий алгоритм для розрахунку енергії основного стану молекули з використанням квантових комп'ютерів. Давайте розглянемо детальніше алгоритм VQE на прикладі молекули водню H_2.

Алгоритм VQE використовує вихідну інформацію про молекулу, що досліджується, тобто визначені параметри гамільтоніана. Гамільтоніан в квантовій хімії описує енергетичний стан системи молекул та її динаміку, тому ми можемо використовувати гамільтоніан для визначення енергії молекули.

Для розв'язання квантової задачі на квантовому комп'ютері необхідно використовувати кубіти, які можна використовувати як рівні 0 та 1. В алгоритмі VQE, ми використовуємо додатковий параметр θ, який буде використовуватися для визначення кутів на квантовому гейті.

Алгоритм VQE складається з наступних етапів:

1. Підготовка вихідного стану: Створення початкового стану з використанням квантових гейтів на кубітах. Зазвичай, як початковий стан використовують одиничний вектор $|0\rangle$ на кожному кубіті. Далі до цих станів можна застосовувати квантові гейти для отримання більш складного стану.

Початковий вихідний стан на кубітах може бути вибраний відповідно до властивостей системи. Наприклад, у випадку молекули водню, яка має два атоми водню, можна використовувати два кубіти, кожен з яких відповідає одному атому водню. Для цього можна використовувати схему Адамара. Наприклад, якщо ми маємо два кубіти, то схема Адамара буде виглядати наступним чином:

$$\mathrm{H} = H_1 \otimes H_2, \tag{2.10}$$

де H_1 та H_2 – гейти Адамара, що застосовуються до першого та другого кубітів відповідно. Отже, застосувавши схему Адамара до початкового стану $|0\rangle$, отримаємо стан рівномірної суперпозиції, який можна використовувати як початковий стан для методу VQE.

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |01\rangle + |10\rangle + |11\rangle), \tag{2.11}$$

2. Побудова гамільтоніана: Визначення гамільтоніана системи. Гамільтоніан – це оператор, що описує енергетичний стан молекули. У випадку молекули водню, гамільтоніан може бути записаний в такій формі:

$$\mathrm{H} = \alpha Z_0 + \beta X_0 X_1 + \gamma Z_1, \tag{2.12}$$

де X та Z – оператори Паулі, а коефіцієнти α, β та γ – константи, які залежать від геометрії молекули та властивостей її складових атомів.

3. Третім кроком буде використовуючи згорткову мережу на класичному комп'ютері, обчислити енергію молекули, що відповідає квантовому стану $|\Psi(\theta)\rangle$. Енергію можна обчислити шляхом вимірювання очікуваної величини гамільтоніана H у квантовому стані $|\Psi(\theta)\rangle$:

$$E(\theta) = \langle\Psi(\theta)|H|\Psi(\theta)\rangle, \tag{2.13}$$

де E(θ) – енергія, залежна від параметрів θ вихідного стану ψ(θ), H – гамільтоніан системи.

4. Варіаційна оптимізація: Використання класичного оптимізатора для знаходження оптимальних значень параметрів θ, які мінімізують енергію системи. Це може бути здійснено шляхом використання різних методів оптимізації, таких як градієнтний спуск або метод Нелдера-Міда.

5. Повторення кроків 3 та 4 до тих пір, поки не буде досягнуто достатньої точності вимірювання енергії. Це може бути досягнуто шляхом встановлення критерію зупинки, такого як максимальна кількість ітерацій або задана точність енергії.

6. Отримання результату: Після знаходження оптимальних значень параметрів θ, можна використовувати ці значення для побудови фінального вихідного стану ψ(θ) та обчислення остаточної енергії системи E(θ), яка буде наближеною енергією основного стану молекули водню.

Отже, алгоритм VQE використовує квантовий апарат для підготовки квантового стану |Ψ(θ)⟩, класичний комп'ютер для обчислення енергії молекули, що відповідає квантовому стану, і метод оптимізації для пошуку параметрів θ, які мінімізують енергію. Алгоритм VQE є одним з найбільш перспективних алгоритмів квантової хімії для вирішення складних проблем хімії та матеріалознавства.

2.3 Особливості використання квантових згорткових нейронних мереж для економічних систем та фінансової аналітики

Застосування квантових згортковх нейронних мереж у фінансах може допомогти вирішити такі завдання, як прогнозування цін на активи, ризик-менеджмент та аналіз даних з фінансових ринків. Однією з можливих застосувань QCNN є розв'язання задачі оптимізації портфеля, де QCNN можуть

використовуватися для виявлення залежностей між різними активами та їх впливу на ризик та доходність портфеля.

Існують бібліотеки та платформи, що надають розробникам та дослідникам можливість експериментувати з квантовими згортковими нейронними мережами та іншими квантовими моделями у фінансах. Однак, оскільки ці технології є відносно новими, необхідно бути обережними при використанні їх у реальних фінансових застосуваннях та проводити додаткові дослідження для оцінки їхньої точності.

Для реалізації квантових згорткових нейронних мереж в фінансах можуть бути використані різні бібліотеки та інструменти. Ось деякі з них:

– Paddle Quantum – це бібліотека програмного забезпечення, яка розроблена компанією PaddlePaddle для квантових обчислень і досліджень у квантових алгоритмах. Вона надає набір інструментів та функцій для розробки, тренування та виконання квантових обчислювальних моделей. Paddle Quantum дозволяє користувачам конструювати квантові обчислювальні моделі за допомогою кількох легко зрозумілих та ефективних інтерфейсів програмування, включаючи такі, як Quantum Circuit, Quantum Neural Network та Variational Quantum Eigensolver. Бібліотека підтримує різні квантові пристрої, такі як квантові біти та квантові кубіти, і має вбудовану підтримку для декількох популярних квантових платформ, включаючи IBM Q, Alibaba Quantum та Baidu Quantum. Paddle Quantum також має вбудовані функції квантового моделювання та симуляції, що дозволяє користувачам перевіряти та тестувати свої квантові моделі на віртуальних пристроях до реалізації на квантових пристроях.

– GS Quant – це бібліотека для мови програмування Python, розроблена компанією Goldman Sachs. Основним призначенням GS Quant є спрощення процесу розробки та тестування квантових моделей у фінансовому секторі. Бібліотека дозволяє розробникам отримувати доступ до фінансових даних, оброблювати їх та виконувати складні аналізи для прийняття рішень на основі квантових моделей. Вона також містить інструменти для виконання операцій з

обробкою даних та машинним навчанням, таких як побудова регресійних моделей та прогнозування величин за допомогою навчання з учителем та без учителя.

– TF Quant Finance – це бібліотека для мови програмування Python, розроблена компанією Google з метою надання інструментів для розробки квантових моделей у фінансовому секторі. TF Quant Finance містить різноманітні інструменти для обробки та аналізу фінансових даних, включаючи роботу з історичними даними, побудову кривих дохідності, створення портфеля та оптимізацію ризиків. Крім того, бібліотека містить інструменти для розробки квантових моделей, таких як квантові нейронні мережі та квантові генетичні алгоритми. Ці інструменти дозволяють розробникам створювати складні квантові моделі для аналізу фінансових даних та прийняття рішень.

Поточні фінансові проблеми можна в основному вирішити за допомогою трьох областей квантових алгоритмів: квантової симуляції, квантової оптимізації та квантового машинного навчання. Багато фінансових проблем по суті є комбінаторними задачами оптимізації, а відповідні алгоритми зазвичай мають високу складність часу та складно виконуватись. Завдяки потужності квантових обчислень, ці складні проблеми очікуються бути вирішеними квантовими алгоритмами в майбутньому.

Прикладом реалізації та цікавою сферою дослідження в економічному та фінансовому секотрі є поняття арбітражу. Арбітраж описує той факт, що один і той самий актив може мати різні ціни на різних ринках і може бути торгований між кількома ринками для отримання позитивного доходу. Тобто, за наявності набору активів та транзакційних витрат, можливо створити цикл між різними ринками, який може забезпечити позитивний дохід.

Ця проблема може бути сформульована мовою теорії графів: заданий зважений орієнтований граф з вершинами, що позначають валюту на ринку, вага ребра від вершини, що позначає валюту до вершини, що позначає обмінний курс, який перетворює валюту до валюти. Оптимізаційна проблема полягає в знаходженні циклу максимального прибутку на заданому орієнтованому графі з

вершинами. В цьому фінансовому модулі користувачі можуть визначати кількість вершин, що містяться в циклі арбітражу відповідно до своїх потреб.

Для того, щоб перетворити задачу оптимізації арбітражної можливості на проблему, яку можна застосувати для параметризованих квантових ланцюгів, потрібно закодувати проблему оптимізації арбітражу в гамільтоніан. Ми реалізуємо кодування, спочатку складаючи задачу цілочисельного програмування. Припустимо, що в графі G є $|V| = n$ вершин, то для кожної вершини $i \in V$, визначаються n бінарні змінні $x_{i,k}$ де $k \in [0, K-1]$ такі, що:

$$x_{i,k} = \begin{cases} 1 \\ 0, \end{cases} \tag{2.14}$$

Потрібно зауважити, що коефіцієнти у гамільтоніані є великими числами, що можуть вплинути на точність підрахунку в квантовому комп'ютері.

Оскільки граф G має n вершин, ми маємо n^2 змінних в загальному, значення яких позначаються рядом бітів $x = x_0 \dots x_{n-1,K-1}$. Покищо припустимо, що рядок бітів x представляє арбітражний цикл. Тоді для кожного ребра $(i, j, w_{ij}) \in E$, ми матимемо $x_{i,k} = x_{j,k+1} = 1$, тобто $x_{i,k} \cdot x_{j,k+1} = 1$, якщо тільки арбітражний цикл відвідує вершину i в час k і вершину j в час $k+1$. В іншому випадку буде $x_{i,k} \cdot x_{j,k+1} = 0$. Отже, логарифм прибутку циклу є:

$$P(x) = -\sum_{i,j \in V} \log(c_{ij}) \sum_{k=0}^{K-1} x_{i,k} x_{j,k+1}, \tag{2.15}$$

Для того, щоб x представляв дійсний арбітражний цикл, потрібно виконувати наступне обмеження:

$$\sum_{i=0}^{n-1} x_{i,k} \text{ та } \sum_{k=0}^{K-1} \sum_{(i,j \notin E)} x_{i,k} x_{j,k+1} \tag{2.16}$$

де перше рівняння гарантує відвідування лише однієї вершини в кожен час. Друге – обмежує виявлення неіснуючого ребра в знайденому арбітражному циклі. Ці два рівняння забезпечують те, що параметризовані квантові схеми знаходять x як простий цикл. Тоді функцію вартості при зазначеному обмеженні можна сформулювати нижче:

$$C_x = -P(x) + A\sum_{k=0}^{K-1}\left(1-\sum_{i=0}^{n-1} x_{i.k}\right)^2 + A\sum_{k=0}^{K-1}\sum_{(i,j)\notin E} x_{i,k}x_{j,k+1}\,, \qquad (2.17)$$

де V – кількість вершин графа, E – множина ребер графа та K – кількість вершин найбільш корисного циклу. Зверніть увагу, що оскільки ми хочемо максимізувати $P(x)$, забезпечуючи x, що представляє дійсний арбітражний цикл, ми краще встановимо A великою, щонайменше більшою за найбільшу вагу ребер.

Ми тепер потрібно перетворити функцію вартості C_x в гамільтоніан, щоб реалізувати кодування задачі оптимізації можливостей арбітражу. Кожна змінна $x_{i,k}$ має два можливі значення, 0 та 1, що відповідають квантовим станам $|0\rangle$та $|1\rangle$. Зверніть увагу, що кожна змінна відповідає кубіту, тому для вирішення задачі оптимізації можливостей арбітражу потрібно n^2 кубитів. Оператор Паулі Z має два власні стани, $|0\rangle$ та $|1\rangle$. Власні значення дорівнюють 1 та -1 відповідно. Тому ми розглядаємо кодування функції вартості в гамільтоніан, використовуючи матрицю Паулі Z.

Тепер розглянемо відображення:

$$x_{i,k} \to \frac{I-Z_{i,k}}{2}, \qquad (2.18)$$

де $Z_{i,k} = I \otimes I \otimes \ldots \otimes Z \otimes \ldots \otimes I$ Z виконується на кубіті на позиції (i,k). Під час цього відображення значення $x_{i,k}$можна проілюструвати по-іншому. Якщо кубіт (i,k) перебуває в стані $|0\rangle$, тоді $x_{i,k}|1\rangle = \frac{I-Z_{i,k}}{2}|1\rangle = 1|1\rangle$, що означає, що

вершина i відвідується в момент часу k. Крім того, для кубіту (i,k), який перебуває в стані $|0\rangle$,$x_{i,k}|0\rangle = \frac{I-Z_{i,k}}{2}|0\rangle = 0|0\rangle$.

Таким чином, використовуючи вищезазначене відображення, ми можемо перетворити функцію вартості C_xв гамільтоніан H_c для системи n^2 кубітів і реалізувати квантову оптимізацію можливостей арбітражу. Тоді ґрунтовий стан H_c є оптимальним рішенням задачі оптимізації можливостей арбітражу.

2.4 Особливості використання квантових згорткових нейронних мереж в криптоаналізі та шифруванні

Квантові комп'ютери, які можуть стати доступними в майбутньому, вплинуть на багато наукових галузей, зокрема на криптографію, оскільки багато асиметричних примітивів є ненадійними проти противника з квантовими можливостями. Криптографи вже передбачають цю загрозу, запропонувавши та вивчаючи кілька потенційно квантово-безпечних альтернатив для цих примітивів. Головною метою криптографії є забезпечення безпеки комунікації. Хоча шифрування забезпечує таємність повідомлення, його цілісність гарантується за допомогою автентифікації. У криптографії з симетричним ключем передбачається, що учасники мають спільний приватний ключ, невідомий будь-якому зловмиснику. Ситуація є симетричною між ними тому цей сценарій також називається симетричною криптографією. Алгоритм Саймона може бути використаний для демонстрації небезпеки загальновживаних симетричних криптографічних примітивів з ключем.

Алгоритм Саймона вирішує наступну задачу: дана $f: \{0,1\}^n \to \{0,1\}^2$ з умовою, що для деякого $s \in \{0,1\}^n$ виконується $|f(x) = f(y)| \Leftrightarrow |x \oplus y \in \{0^n, s\}|$. Можна показати, що будь-який класичний алгоритм потребує $\Omega(2^{\frac{n}{2}})$ класичних запитів до f, щоб знайти s, тоді як алгоритм Саймона успішно знаходить s, використовуючи лише $O(n)$ квантових запитів до f. Квантова частина алгоритму Саймона полягає в виконанні наступної схеми:

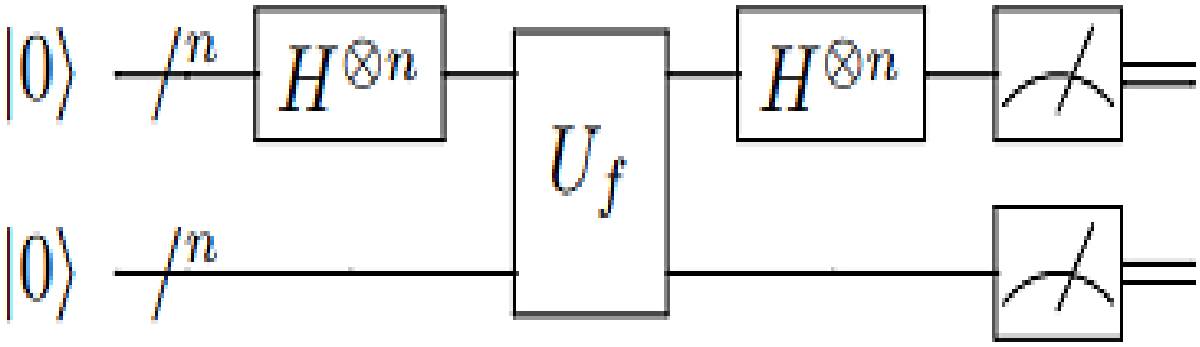


Рисунок 2.2 – Частина алгоритму Саймона

Квантова частина алгоритму Саймона полягає в розв'язанні задачі про знаходження періоду функції f(x), де x – бітовий рядок довжини n, а f – періодична функція, що працює з бітовими рядками довжини n.

Основний етап квантової частини алгоритму Саймона полягає в тому, щоб використовувати кубіти для побудови періодичної функції f(x). Для цього використовуються так звані «чорнильні плями», які діють як перетворення для вхідного кубіту $|x\rangle$ та вихідного кубіту $|y\rangle$ за наступною формулою:

$$U_f|x\rangle\,|y\rangle = |x\rangle\,|y \oplus f(x)\rangle, \qquad (2.19)$$

де $\oplus$ позначає побітове додавання за модулем 2, тобто XOR. Іншими словами, чорнильна пляма додає до вихідного кубіту $|y\rangle$ значення f(x), де f – періодична функція, залежна від вхідного кубіту $|y\rangle$.

Перша операція призводить до стану $\sum_{x\in\{0,1\}^n} |x\rangle|0\rangle$, який після запиту f стає $\sum_{x\in\{0,1\}^n} |x\rangle|f(x)\rangle$. Вимірюючи другий регістр отримуємо випадковий результат $f(x)$, і перший регістр обвалюється до $\frac{1}{\sqrt{2}}(|x\rangle + |x \oplus s\rangle)$ через обіцяну структуру f. Застосування ще однієї операції призводить до стану $\sum_{y\in\{0,1\}^n} (-1)^{(y\cdot x)}\,(1 + (-1)^{(y\cdot s)})|y\rangle$. Зверніть увагу, що для $y \in \{0,1\}^n$ з $y \cdot s = y_1 s_1 \oplus \cdots \oplus y_n s_n = 1$, маємо деструктивну інтерференцію, і амплітуди тих рядків y зникають. Отже, вимірювання реєстру призведе до (випадкової) n-бітової строки y з властивістю $y \cdot s = 0$. Виконання цієї квантової процедури

$O(n)$ разів призведе до повного рангу системи лінійних рівнянь, яку можна ефективно вирішити класично, щоб отримати рядок s.

Ще одним прикладом алгоритму, шо може бути загрозою для сучасної криптографії є алгоритм Шора. Це квантовий алгоритм, який дозволяє розкладати складні числа на прості множники. Для застосування алгоритму Шора до криптографічних ключів RSA, які базуються на складних числах, можна використовувати його для розкладання публічного ключа на прості множники. Ідея цього алгоритму досить проста. На вхід подаються два регістри кубітів: перший відповідає вхідним значенням функції, другий – вихідним. У першому регістрі створюється суперпозиція всіх можливих вхідних значень, другий регістр ініціалізується фіксованим станом, після чого на виході ми отримуємо квантову суперпозицію всіх можливих входів та відповідних їм виходів. Потім робимо вимірювання вихідного регістру, в результаті чого отримуємо деяке випадкове значення функції, а в іншому регістрі – суперпозицію всіх аргументів функції, що відповідають отриманому значенню. Далі застосовуємо квантове перетворення Фур'є над першим регістром і в вимірювальному вимірюванні отримуємо величину, пропорційну оберненому періоду функції. Повторюючи цю операцію кілька разів і використовуючи класичний алгоритм Євкліда для пошуку найбільшого спільного дільника (НСД), ми отримуємо сам період.

Даний алгорим можна розділити умовно на дві частин: класичне розкладання на множники функції та квантове обчислення періоду даної функції.

Для початку потрібно визначити три константи:

– M – число, що використовується для розкладання на множники;

– N – розмір регістра пам’яті. Бітовий розмір даної пам’яті $n = \log_2 N$, що в два рази більше M;

– t – рандомий параметр, такий що: $1 < t < M$ і $\text{НСД}(t, M) = 1$.

Класична частина алгоритму має наступні кроки:

1) розрахувати $K = \text{НСД}(t, N)$;
2) якщо, $K \neq 1$ то K нетривіальний фактор N і алгоритм на цьому закінчується;

3) в іншому випадку потрібно скористатися підпрограмою пошуку квантового періоду (рис 2.2), щоб знайти r, що позначає період наступної функції;
4) якщо r виявилося парне число, то перейти до пункту 1;
5) якщо виконується умова $a^{\frac{r}{2}} = -1 mod N$, то перейти до пункту 1;
6) в іншому випадку обидва НСД $\left(a^{\frac{r}{2}} + 1, N\right)$ та НСД $\left(a^{\frac{r}{2}} - 1, N\right)$ є нетривіальними факторами.

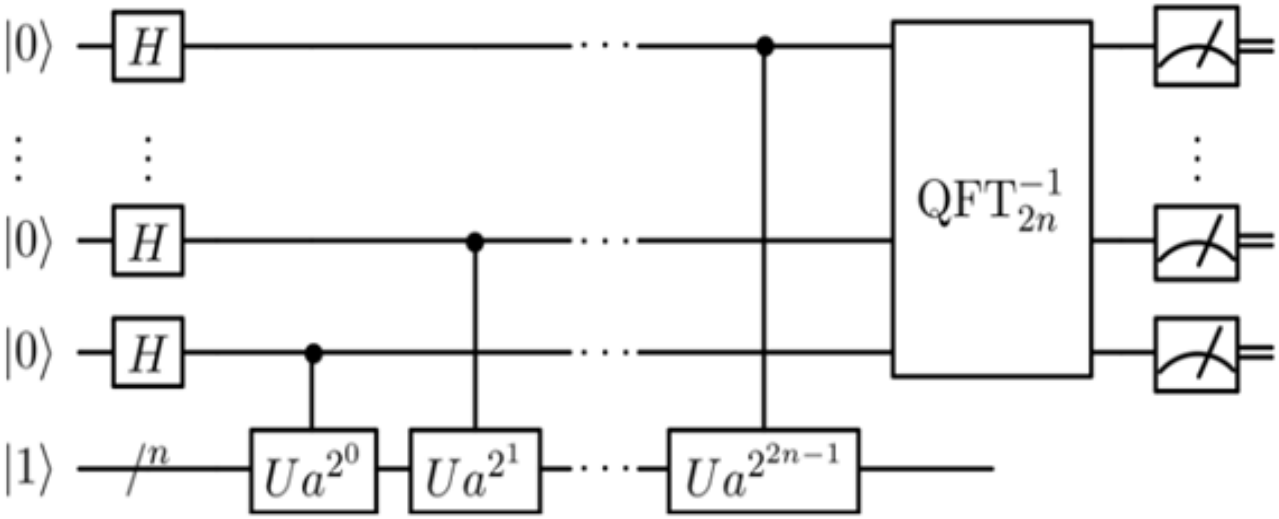


Рисунок 2.3 – Квантова підпрограма в алгоритмі Шора

Квантова підготовка включає створення суперпозиції квантових станів, а квантова фазова оцінка дозволяє знайти періодичність функції, що визначає факторизацію числа.

Основна формула, яка використовується в квантовій підготовці, це формула Гадамарда. Вона використовується для перетворення базисних квантових станів (|0⟩ та |1⟩) у рівновагу суперпозицій:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \tag{2.20}$$

$$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle), \tag{2.21}$$

Ці формули дозволяють створити рівновагу суперпозицію з n кубітів, використовуючи послідовність Гадамарда на кожному кубіті.

У квантовій фазовій оцінці використовується квантовий алгоритм зворотнього дискретного перетворення Фур'є (QFT). Формула QFT виглядає наступним чином:

$$QFTH|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} e^{\frac{2\pi i x y}{N}} |\psi_y\rangle, \tag{2.22}$$

де |ψ⟩ – квантовий стан, $N = 2^{\text{т}}$ – кількість можливих станів, x та y – цілі числа в діапазоні від 0 до $N-1$, а $e^{\frac{2\pi i x y}{N}}$ – комплексне число, яке дозволяє зв'язати квантові стани $|\psi_x\rangle$та $|\psi_y\rangle$.

Іншим алгоритмом, який являє загрозу для криптографії є квантовий алгоритм дискретного логарифму. Він є алгоритмом, який може використовуватися для взлому криптографічних протоколів, які використовують дискретний логарифм для захисту від атак. Дискретний логарифм – це математична задача, яка полягає у знаходженні значення x в рівнянні $a^x = b \bmod p$, де a, b та p – цілі числа, які задовольняють умови p – просте число, а a не є кратним p, тобто НСД$(a, p) = 1$.

Квантовий алгоритм дискретного логарифму базується на алгоритмі періодичного пошуку. Для того, щоб застосувати цей алгоритм до задачі дискретного логарифму, потрібно використовувати спеціально побудовані квантові оператори, які діють на квантові біти.

Основним кроком квантового алгоритму дискретного логарифму є побудова квантового оператора перетворення Фур'є за модулем N, де N є модулем відносно якого шукаємо дискретний логарифм. Оператор перетворення Фур'є за модулем N позначається як $U_{a,N}$ і визначається формулою:

$$(U_{a,N}|\psi_x\rangle)_y = \frac{1}{\sqrt{N}}\left|\psi_y\right\rangle_{xy} e^{2\pi i\frac{axy}{N}}, \quad (2.23)$$

де $\left|\psi_y\right\rangle_{xy}$ – квантовий стан, що відповідає значенню x у двійковому коді, y – десяткове число в діапазоні від 0 до $N-1$, а a – ціле число, яке є основою генератора групи, в якій шукаємо дискретний логарифм.

Після побудови оператора $U_{a,N}$, застосовується алгоритм періодичного пошуку, щоб знайти період функції $f(x) = a^x \, mod \, N$. Для цього використовується квантовий оператор Q, який позначається також як $U_{a,N}$, і визначається формулою:

$$(U_{f,N}|\psi_x\rangle)_y = (-1)^{f(y)}|\psi_x\rangle_y, \quad (2.24)$$

де $f(y) = a^{\,y} mod \, N$.

Проте, на сьогоднішній день, квантові комп'ютери, які мають достатньо кубітів для вирішення складних криптографічних проблем, все ще знаходяться на ранній стадії розвитку. Більшість експертів вважають, що наступних 10-20 років, класична криптографія буде захищена від квантових атак, але потрібно враховувати можливість злому криптографії, що базується на публічних ключах, в тому числі.

2.3 Висновки до розділу 2

Отже, квантові згорткові нейронні мережі мають потенціал для використання в різних сферах. Проте, їхній успіх залежить від наявності достатньої кількості квантових бітів та ефективних методів навчання та оптимізації моделей. Подальші дослідження можуть допомогти виявити найбільш ефективні способи застосування квантових згорткових нейронних мереж в різних областях.

Для успішного використання квантових згорткових нейронних мереж в хімії необхідно враховувати складність та ресурсоємність їх розробки та використання. Важливо також враховувати етичні та регуляторні аспекти використання таких технологій у хімії та вживати заходів для запобігання можливих ризиків.

Подальші дослідження та розробки у галузі квантових згорткових нейронних мереж можуть допомогти вирішувати складні проблеми у хімії та розвивати нові матеріали та технології, які можуть бути корисні для людства.

Усілякі автоматизовані та автоматичні системи торгівлі та інвестування повинні бути розроблені з урахуванням ризиків та відповідно до найкращих практик, щоб запобігти можливості фінансових шахрайств та збитків. Подальші дослідження та розробки можуть допомогти покращити ефективність та точність квантових згорткових нейронних мереж в арбітражі та інших областях фінансової сфери.

Хоча квантові згорткові нейронні мережі мають потенціал для використання в задачах криптоаналізу, поки що немає достатніх доказів їхньої ефективності в цій області. Подальші дослідження можуть допомогти визначити, які задачі в криптографії можуть бути успішно вирішені за допомогою квантових згорткових нейронних мереж та які їх обмеження і недоліки в цьому напрямку.

Розділ 3 Програмна реалізація з застосуванням квантових згорткових нейронних мереж для технічних, природничих і соціально-економічних інформаційних систем

3.1 Програмна реалізація з застосуванням квантових нейронних мереж для обчислення енергії зв'язку між протоном і нейтроном у ядрі дейтерія

На поточному етапі розвитку квантових обчислень доступні квантові процесори ще не мають впроваджених кодів корекції помилок і, отже, все ще мають шум, з яким потрібно боротися. Цей шум може негативно впливати на результати і обмежувати типи проблем, які ми можемо вирішувати на поточному етапі, оскільки через шум не можна очікувати високого рівня впевненості та точності в отриманих результатах.

Однак, існують проблеми, які вимагають вирішення лише декількох кубітів, наприклад, обчислення зв'язкової енергії між протоном та нейтроном у ядрі дейтерону. Ця проблема є прикладом ситуації, де ми можемо досягти хороших результатів незважаючи на зазначені раніше виклики.

Для вирішення даної задачі для початку був створений метод create_deuteron_hamiltonian (див. Додаток А). Даний метод створює гамильтоніан для системи деутерон у квантовій обчислювальній моделі. Гамільтоніан визначає енергійний спектр та взаємодії системи.

Функція отримує наступні параметри:

– ціле число, яке визначає розмірність системи "деутерон".

– дійсне число, що представляє значення сталої Планка помножене на частоту деутерону.

– дійсне число, яке визначає потенціалну енергію взаємодії між протоном і нейтроном у ядрі.

У коді створюється порожній список hamiltonian_terms, який буде містити внески до гамильтоніана. Потім виконуюься дві вкладки циклів for для обчислення коефіцієнтів кінетичної та потенціальної енергії.

Після завершення циклів створюється об'єкт hamiltonian типу FermionicOp, який представляє гамильтоніан у ферміонній базисній формі.

Наступною кроком використовується мапер JordanWignerMapper, який відповідає за перетворення ферміонного гамильтоніана на кубітний гамильтоніан. Отриманий кубітний гамильтоніан зберігається у об'єкті qubit_hamiltonian. Якщо qubit_hamiltonian не є екземпляром SparsePauliOp, то він перетворюється на його примітивний вигляд. На останньому кроці функція повертає qubit_hamiltonian, який є гамильтоніаном системи деутерон у кубітній формі.

Загальною метою цього коду є створення гамильтоніана для системи деутерон у квантовій обчислювальній моделі, що є важливим етапом для вивчення та аналізу властивостей цієї системи.

Після цього створюється список для зберігання гамільтоніанів H_1, H_2 та H_3 оскільки ми будемо використовувати ці гамільтоніани пізніше для обчислення основного стану. Для цього було використане списокове складання з функцією create_deuteron_hamiltonian, яку ми визначили раніше та були отримані гамільтоніани (рисунок 3.1).

```
Гамільтоніан Дейтерія: H_1
SparsePauliOp(['I', 'Z'],
              coeffs=[-0.21829055+0.j,  0.21829055+0.j])


Гамільтоніан Дейтерія: H_2
SparsePauliOp(['II', 'IZ', 'XX', 'YY', 'ZI'],
              coeffs=[ 5.90670945+0.j,  0.21829055+0.j, -2.14330352+0.j, -2.14330352+0.j,
 -6.125     +0.j])


Гамільтоніан Дейтерія: H_3
SparsePauliOp(['III', 'IIZ', 'IXX', 'IYY', 'IZI', 'XXI', 'YYI', 'ZII'],
              coeffs=[15.53170945+0.j,  0.21829055+0.j, -2.14330352+0.j, -2.14330352+0.j,
 -6.125     +0.j, -3.91311896+0.j, -3.91311896+0.j, -9.625     +0.j])
```

Рисунок 3.1 – Отримані гамільтоніани

Для побудови ланцюгів, нам потрібно визначити два параметри, theta та eta, що можна зробити за допомогою Parameter() з бібліотеки Qiskit. Параметри задаються у форматі рядків, де θ та η відповідають відповідним математичним символам у форматі LaTex. Ці параметри будуть використовуватися для конструювання квантових експериментів та квантових алгоритмів.

Використовуючи визначені вище параметри отримуємо наступні схеми (рисунок 3.2 – 3.4).

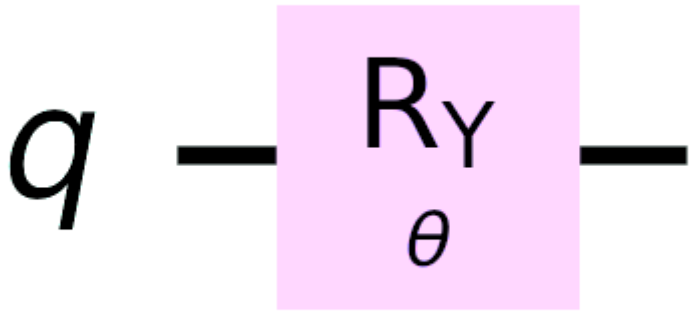


Рисунок 3.2 – Схема з одним кубітом

На цьому кубіті застосовується ворот RY (ворот обертання навколо вісі Y) з параметром theta, викликаючи метод .ry(theta, 0) на wavefunction.

Після цього викликається метод draw, що візуалізує квантовий ланцюг у вигляді схеми за допомогою бібліотеки Matplotlib. В результаті отримуємо графічне представлення квантового ланцюга, де ворот RY з параметром theta застосовується до першого кубіту.

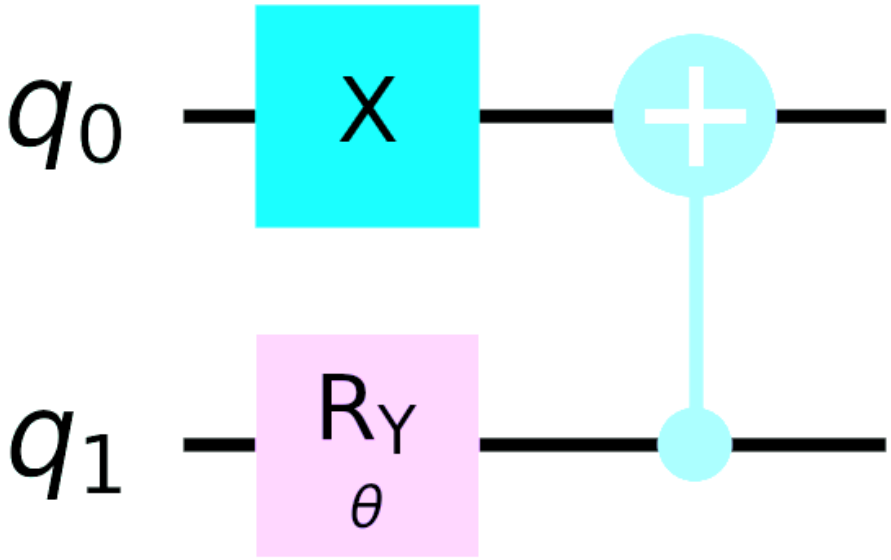


Рисунок 3.3 – Схема з двома кубітами

Для двух кубітів створюється квантовий ланцюг з двома кубітами за допомогою QuantumCircuit(2).

Спочатку на першому кубіті застосовується ворот X (ворот Полінга X) за допомогою методу .x(0), що встановлює початковий стан першого кубіту в стан |1⟩. Потім на другому кубіті застосовується ворот RY (ворот обертання навколо вісі Y) з параметром theta. Далі застосовується ворот CNOT (ворот КНОТ) з другого кубіту на перший, реалізуючи контрольовану операцію на кубітах.

Нарешті, код викликає метод draw що візуалізує квантовий ланцюг у вигляді схеми за допомогою бібліотеки Matplotlib. В результаті отримуємо графічне представлення квантового ланцюга, де вороти X, RY та CNOT застосовуються до відповідних кубітів.

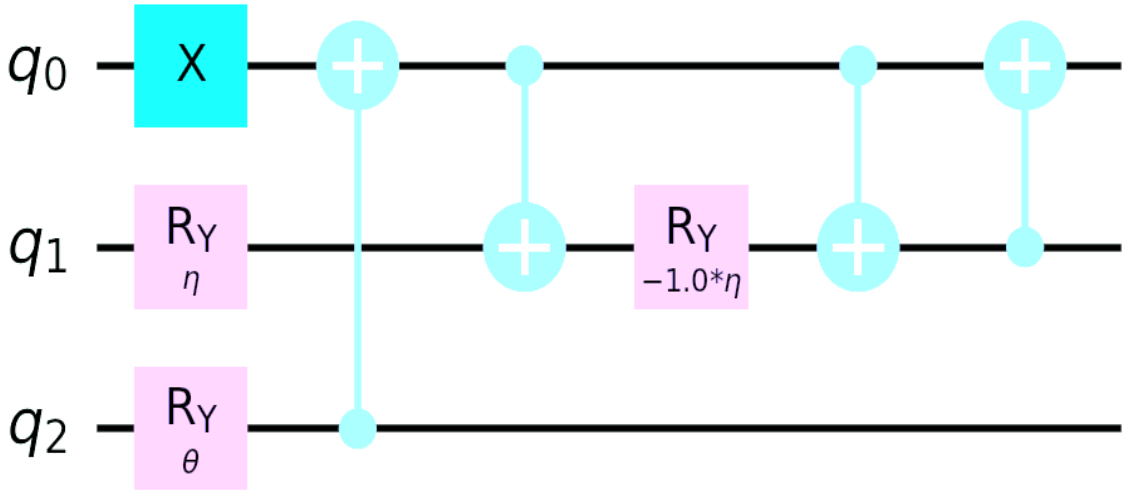


Рисунок 3.4 – Схема для трьох кубітів

Квантова схема з трьох кубітів і виконує наступні операції:

– Перевертаємо перший кубіт за допомогою гейту x.

– На другому кубіті застосовуємо ротацію RY з параметром eta.

– На третьому кубіті застосовуємо ротацію RY з параметром theta.

– Виконуємо контрольовану операцію CX (CNOT) між кубітами 2 і 0.

– Виконуємо контрольовану операцію CX (CNOT) між кубітами 0 і 1.

– Знову застосовуємо ротацію RY на другому кубіті з оберненим параметром eta.

– Знову виконуємо контрольовану операцію CNOT між кубітами 0 і 1.

– Знову виконуємо контрольовану операцію CNOT між кубітами 1 і 0.

Також був написаний код для обчислення енергії зв'язку для гамільтоніанів H_1, H_2 та H_3 з використанням Estimator для кожного гамільтоніана із вказаним анзацом. Зіткнувши один генератор випадкових чисел, за допомогою VQE і SLSQP optimizer, використовуються вказані анзаци і гамільтоніани для обчислення мінімальної власної енергії і біндуючої енергії для H_1, H_2 та H_3. Результати друкується в консоль (рисунокк 3.5)

```
Results using Estimator for H_1, H_2 and H_3 with the ansatz

Binding energy for H_1: -0.4365811096105766 MeV
Binding energy for H_2: -1.7491595316575461 MeV
Binding energy for H_3: -2.045670898257444 MeV
```

Рисунок 3.5 – Обчислення енергії звязку

Графіки результатів, отриманих під час виконання алгоритму VQE зображені на рисунку 3.6.

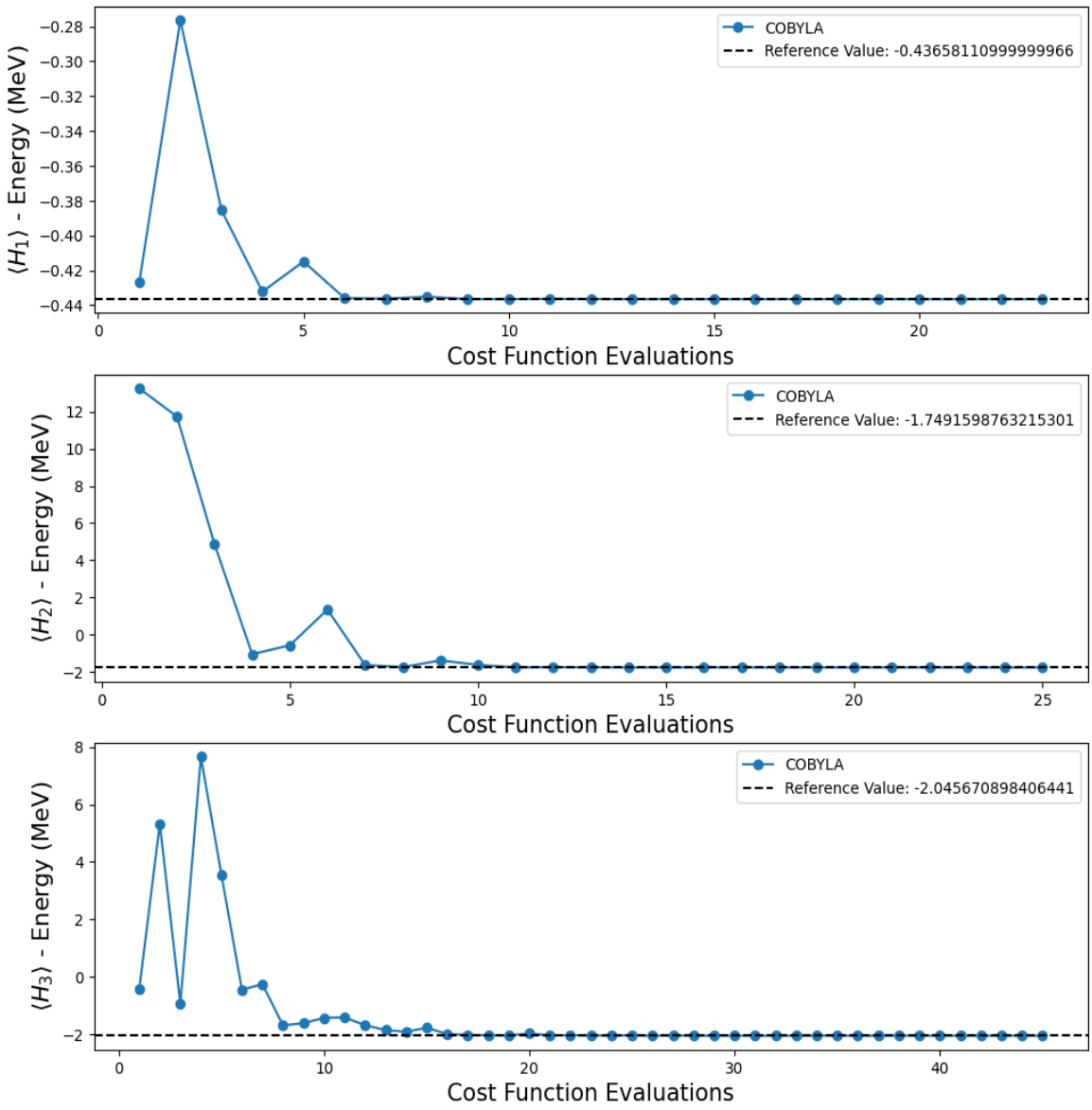


Рисунок 3.6 – Графіки енергії

Також було обчислене математичне сподівання спостережуваної величини на основі квантової схеми, що представляє хвильову функцію, та списку параметрів (рисунок 3.7).

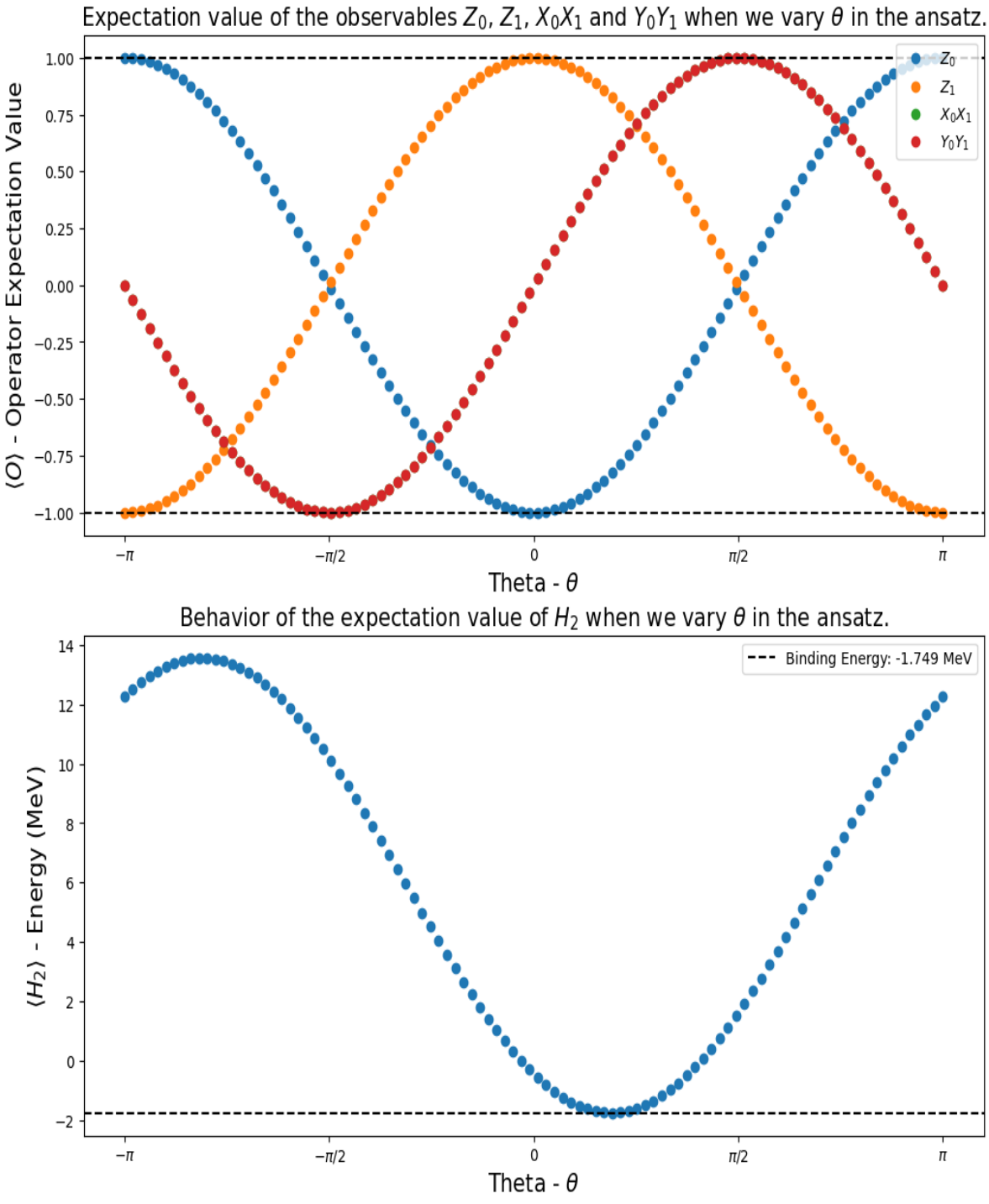


Рисунок 3.7 – Математичне сподівання

Для реалізації даного процесу була створена функція calculate_observables_exp_values (див. Додаток А). Ця функція перебирає кожну

спостережувану величину зі списку observables і кожен кут зі списку angles. Для кожної комбінації спостережуваної величини і кута вона прив'язує значення параметра theta до квантового кола, використовуючи метод bind_parameters. Потім вона використовує Estimator та функцію estimate_observables для обчислення математичного сподівання спостережуваної величини на квантовому стані qc.

Отримані значення математичних сподівань додаються до списку exp_values, який потім додається до загального списку list_exp_values. На виході функція повертає список математичних сподівань для кожної спостережуваної величини.

3.2 Програмна реалізація з застосуванням квантових згорткових нейронних мереж для економічних систем та фінансової аналітики

Криптовалюти, як відомо, нестабільні, і їх складно передбачити, однак передбачення їхньої вартості є великим фінансовим стимулом. Метою реалізаціїї даної програми є порівняння квантових і класичних методів машинного навчання для прогнозування часових рядів криптовалюти. Це може бути корисним для трейдерів і інвесторів, які хочуть приймати обґрунтовані рішення щодо купівлі, продажу або утримання криптовалют.

Використані дані – це ціна криптовалюти Ethereum за 2019–2023 роки, включаючи відкриття, максимум, мінімум, закриття, скориговане закриття та обсяг за кожен день [27].

Вхідні дані для аналізу отримані з сайту finance.yahoo.com та мають вигляд зображений на рисунку 3.8.

Рисунок 3.8 – Вхідні дані криптовалюти на сайті

Аналітичні дані про криптовалюту формуються у вигляді табличних даних, які зберігаються у файлі у форматі CSV. Ці дані описують відношення криптовалюти до долара США та дозволяють проводити аналіз цих відношень. (рисунок 3.9).

A	B	C	D	E	F	G
Date	Open	High	Low	Close	Adj Close	Volume
5/15/2018	731.143005	739.052002	700.994995	708.870972	708.870972	2523069952
5/16/2018	708.086975	710.200012	682.541016	707.049988	707.049988	2476130048
5/17/2018	708.718018	718.833008	668.833984	672.656982	672.656982	2350619904
5/18/2018	672.10199	695.031006	663.809021	694.367004	694.367004	2305740032
5/19/2018	695.072021	715.578003	686.791016	696.530029	696.530029	2021549952
5/20/2018	697.922974	723.752991	692.669006	715.369019	715.369019	2156910080
5/21/2018	717.192993	719.278015	692.494019	699.221985	699.221985	2005170048
5/22/2018	700.177979	700.976013	644.026001	647.741028	647.741028	2230469888
5/23/2018	646.669983	651.635986	572.952026	583.585022	583.585022	2995429888
5/24/2018	584.536011	610.817993	557.205994	601.755005	601.755005	2791099904
5/25/2018	602.140015	617.185974	575.624023	586.734009	586.734009	2110919936
5/26/2018	587.426025	606.174988	583.512024	587.280029	587.280029	1694300032
5/27/2018	588.52002	590.328003	562.866028	572.66803	572.66803	1788790016
5/28/2018	573.044983	576.049011	512.552002	516.036011	516.036011	2356900096
5/29/2018	516.14801	572.263977	516.14801	565.388	565.388	2330820096
5/30/2018	566.830017	583.135986	545.43103	559.590027	559.590027	2053970048
5/31/2018	558.497009	585.538025	557.065979	577.64502	577.64502	1985040000
6/1/2018	578.671997	589.093018	567.664978	580.04303	580.04303	1945890048

Рисунок 3.9 – Дані для аналізу

Програмне забезпечення та послуги, що використовуються, включають:

– PennyLane: для гібридного CNN і варіаційного алгоритму.
– Tkinter: для створення графічного інтерфейсу.
– Torch: для CNN і Hybrid CNN.
– Scikit-Learn: для попередньої обробки даних.
– Matplotlib: для побудови даних.
– Pandas і Numpy: для представлення даних.

Графічний інтерфейс складається з декількох елементів, які включають текстове поле, кнопки, полоси прокрутки та ін. Ці елементи допомагають взаємодіяти з програмою або додатком, забезпечуючи зручність та ефективність використання. Він зображений за рисунком 3.10.

Детальний опис графічного інтерфейсу:

– Кнопка «Запустити навчання»: Ця кнопка призначена для початку процесу навчання моделі. При натисканні на неї, програма запускає навчання з використанням відповідних наборів даних і розпочинає обчислення.

– Текстове поле для виводу процесу: Це поле відображає інформацію про поточний стан процесу, прогрес навчання, повідомлення про помилки або іншу інформацію, яка пов'язана з процесом. Воно оновлюється під час виконання процесу та надає зручний спосіб спостереження за станом системи.

– Три кнопки для отримання графіків: Дані кнопки надають можливість користувачу отримати графіки, пов'язані з результатами навчання або процесу. При натисканні на кожну кнопку, відповідний графік буде відображатися в окремому вікні.

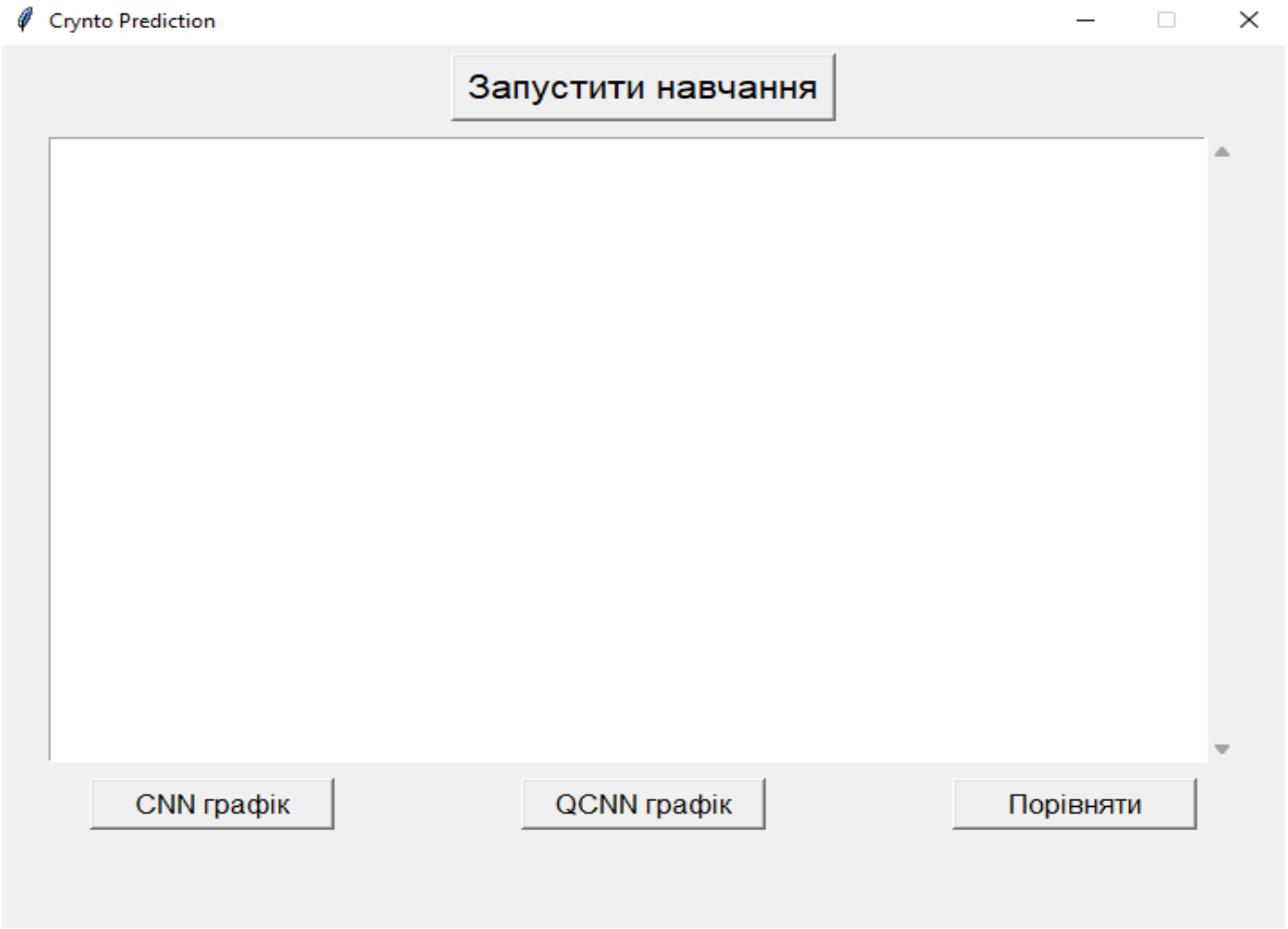


Рисунок 3.10 – Графічний інтерфейс

Цей графічний інтерфейс дозволяє користувачу зручно керувати процесом навчання, спостерігати за його прогресом та отримувати графічні візуалізації, які допомагають зрозуміти результати та характеристики процесу.

При натисканні на кнопку «Запустити навчання» користувач має змогу підтведрити початок операції навчання (рисунок 3.11).

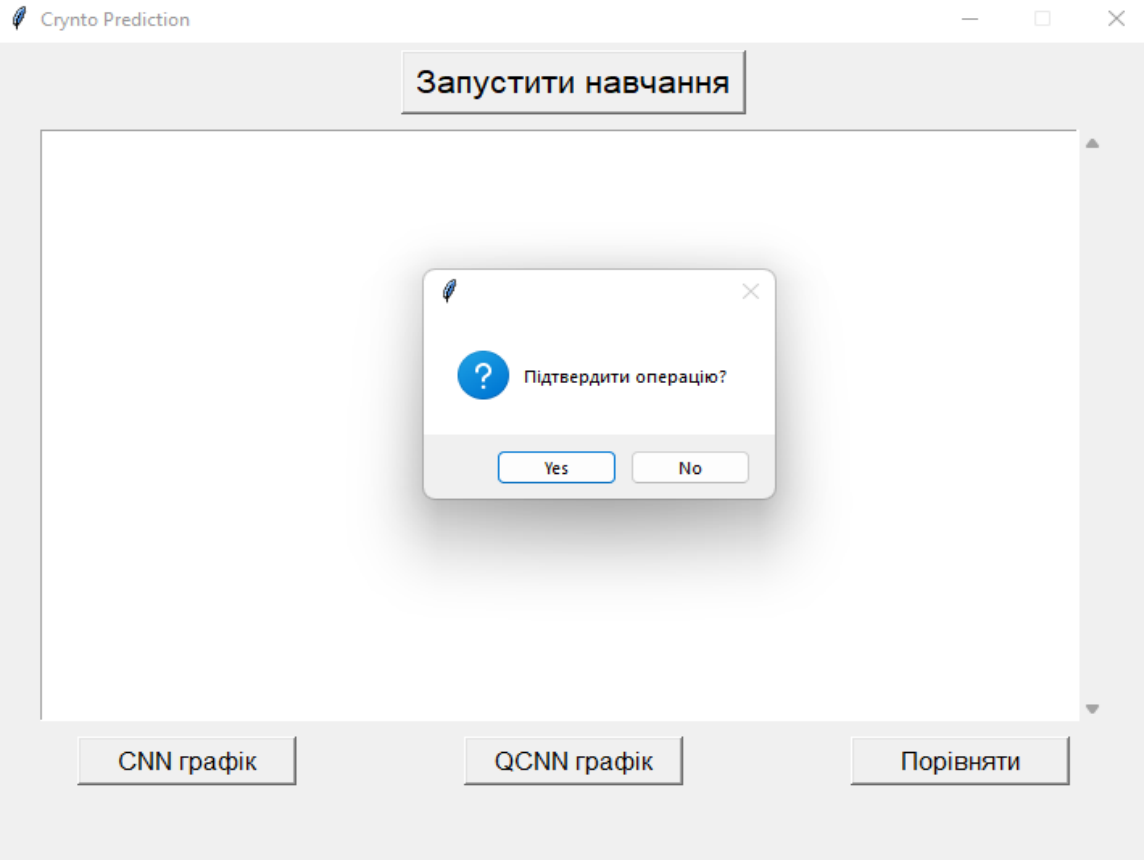


Рисунок 3.11 – Підтвердження операції

Якщо операція була підтверджена, запускається метод runProcess(), який виконує процес навчання і тестування моделей для обробки часових рядів криптовалютних даних. Основний алгоритм складається з таких кроків:

– Встановлюються різні параметри для процесу навчання, такі як шлях до файлу з даними, колонка індексу, колонка цільової змінної, шлях до файлу для запису результатів, кількість прикладів для тренування, кількість прикладів для тестування, початквий індекс, швидкість навчання та інші.

– Створюються дві моделі, які базуються на класі ConvCryptoTimeSeriesModel. Обидва моделі мають однакові параметри, за винятком встановлення quantum на False для першої моделі і True для другої. Це дозволяє порівняти результати навчання моделей з та без квантового підходу.

Для кожної моделі виконуються такі кроки:

– Зчитуються дані з файлу з вхідними даними.

– Виконується попередня обробка даних, включаючи виконання інверсії цільової змінної.

– Якщо модель є екземпляром класу ConvCryptoTimeSeriesModel, ініціалізуються шари моделі.

– Виконується навчання та тестування моделі

– Виконується зворотне перетворення цільової змінної.

– Результати навчання записуються в файл (рисунок 3.12).

– В результаті виводяться файли з графіками для кожної моделі, а також файл зі спільним графіком для багатьох моделей.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
LVDq	CNN	0	3.454	0	100	50	1000	5	1000	0.005	0.003263	0.016192	5	[128, 64]	FALSE
KC5q	CNN_Qua	0	340.742	1.735	100	50	1000	5	1000	0.005	0.001673	0.013895	5	[128, 64]	TRUE
0Foz	CNN	0	4.003	0.024	100	50	1000	5	1000	0.005	0.00067	0.016696	5	[128, 64]	FALSE
QAP7	CNN	0	3.527	0.018	100	50	1000	5	1000	0.005	0.000711	0.017167	5	[128, 64]	FALSE
RiVi	CNN	0	3.376	0.022	100	50	1000	5	1000	0.005	0.000259	0.012496	5	[128, 64]	FALSE
xDkS	CNN	0	4.334	0.02	100	50	1000	5	1000	0.005	0.004481	0.022475	5	[128, 64]	FALSE
EkO8	CNN	0	3.637	0.015	100	50	1000	5	1000	0.005	0.014885	0.016886	5	[128, 64]	FALSE

Рисунок 3.12 – Файл з результатами навчання

Була створена базова модель model, яка використовується для завантаження й попередньої обробки даних, запису й побудови результатів, а спеціальні моделі машинного навчання conv_model.py успадковують цю модель і реалізують функції для навчання й тестування даних (див. Додаток Б).

Основним методом файлу model є метод preprocess, який виконує попередню обробку даних для подальшого використання їх у моделі прогнозування. Основні етапи обробки даних включають:

– перетворення даних у послідовності – дані розбиваються на послідовності фіксованої довжини з використанням параметра lookback, який визначає скільки попередніх значень використовувати для прогнозування наступного значення.

– розділення на x та y – дані розбиваються на матрицю з вхідними параметрами x та матрицю з виходним значенням y.

– трансформація даних – застосовуються дві функції масштабування даних: MinMaxScaler та StandardScaler.

– розділення на навчальні та тестові дані – дані розділяються на навчальні та тестові за вказаними параметрами num_train та num_test.

– перетворення формату даних – дані конвертуються у формат Tensor та відповідні формати для використання в моделі прогнозування.

Для реалізаціїї згорткової нейронної мережі та квантової згорткової нейроної мережі був створений файл conv_model. Ініціалізація шарів нейронної мережі відбувається за допомогою методу initialize_layers. У першій частині методу задаються параметри для квантового та згорткового шарів. Кількість вхідних каналів та вихідний розмір для згорткового шару встановлюються на основі значень lookback та num_qubits. Параметри ваг у квантовому шарі встановлюються у словнику weight_shapes. Далі створюються шари для нейронної мережі.

Якщо встановлено прапорець quantum, тоді використовується квантовий шар TorchLayer з параметрами qnode та weight_shapes. Якщо прапорець не встановлено, тоді використовуються лінійні шари замість квантового шару.

У кінці методу ініціалізується оптимізатор Adam для навчання мережі з використанням параметрів, що підлягають оптимізації, та заданим коефіцієнтом швидкості навчання lr.

Навчання нейронної мережі відбувається за допомогою метода train (див. Додаток А). У ньому відбувається ітерація через задану кількість iterations. На кожній ітерації вибирається випадкова партія даних з тренувального набору за допомогою генерування випадкових індексів та відповідні партії даних x_batch і y_batch з тренувальних даних і після цього виконується передача даних через мережу, отримуючи вихід за допомогою методу forward(). Після цього обнуляються градієнти оптимізатора.

Потім обчислюється значення функції втрат loss за допомогою заданої функції втрат, яка залежить від отриманих виходів і партії цільових значень. Після

цього виконується зворотне поширення помилки за допомогою методу backward, який обчислює градієнти ваг.

Наступною дією оновлюються параметри мережі з використанням оптимізатора за допомогою методу step. Кожну соту ітерацію виводиться інформація про номер ітерації та значення функції втрат на текстове поле що бкористувач мав змогу відслідковувати процес навчання (рисунок 3.13).

Після закінчення навчання зберігаються значення функції втрат train_loss та часу навчання train_time та починається тестування.

Визначені параметри для навчання:

– кількість прикладів для тренування – 100;
– кількість прикладів для тестування – 50;
– кількість попередніх спостережень – 6;
– швидкість навчання – 0.005.

Метод test (див. Додаток Б) виконує тестування моделі шляхом прямого передавання тестових даних через модель та обчислення втрати. Результати, такі як прогнозовані значення, значення втрати та час тестування, записуються та виводяться у відповідні журнали або консоль.

Даний метод можна описати коротко наступними кроками, які відбуваються:

– записується повідомлення про початок тестування;
– встановлюється початковий час тестування;
– виконується пряме передавання тестових даних через модель, отримуються прогнозовані значення;
– прогнозовані значення та тестові значення перетворюються на NumPy масиви;
– обчислюється значення втрати за допомогою визначеного критерію;
– записується та виводиться значення втрати;
– записуються та виводяться значення тестових даних та прогнозованих значень.

Метод test допомагає оцінити ефективність моделі шляхом тестування її на незалежних тестових даних та обчислення втрати, що дає нам змогу оцінити точність моделі.

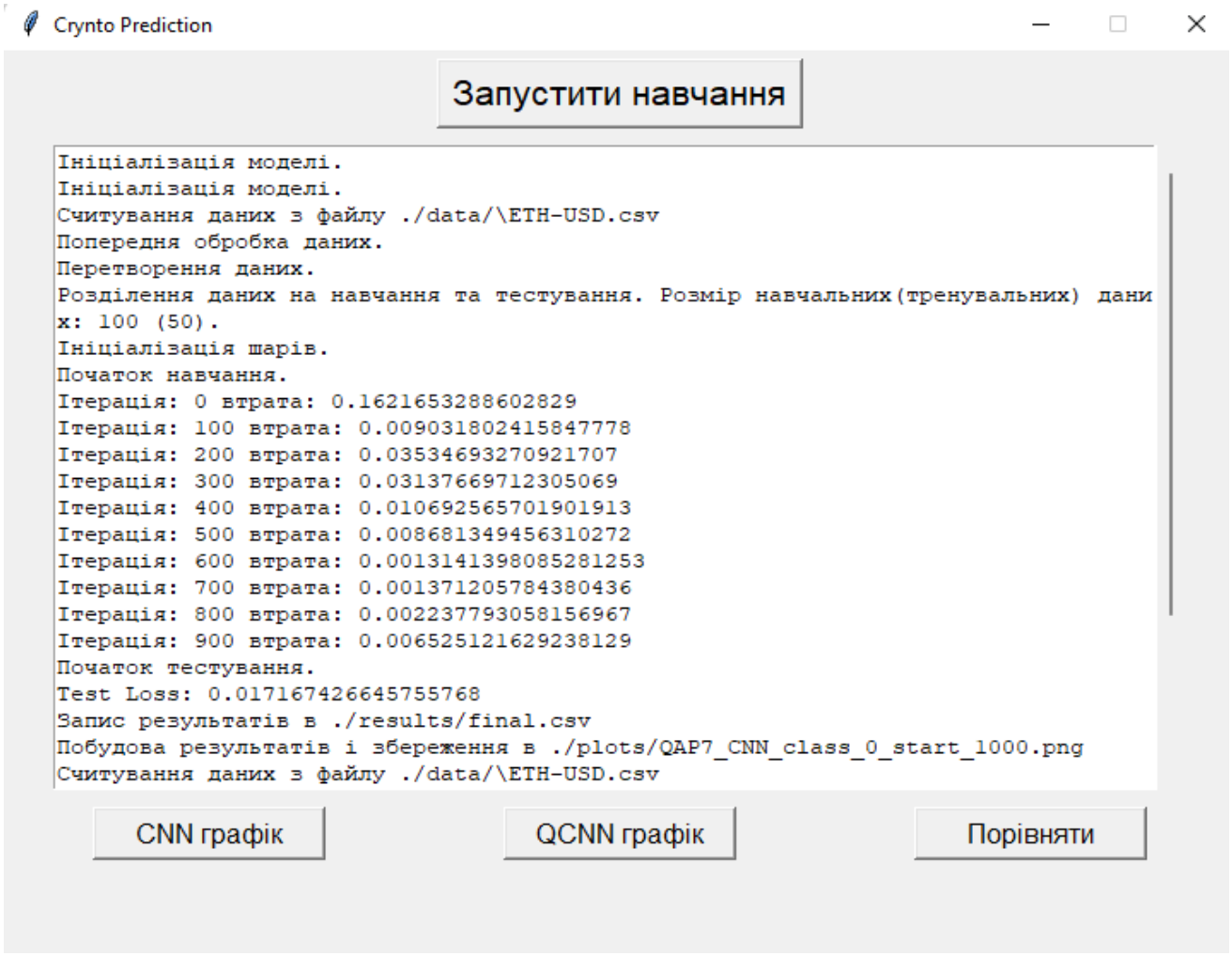


Рисунок 3.13 – Текстове поле з ітераціями

Після завершення навчання створюються три графіка, які порівнюють вхідні дані та результати навчання нейронної мережі. Переглянути отримані дані можна при натисканні бажаних кнопок знизу графічного інтерфейсу (рисунок 3.14 – 3.16).

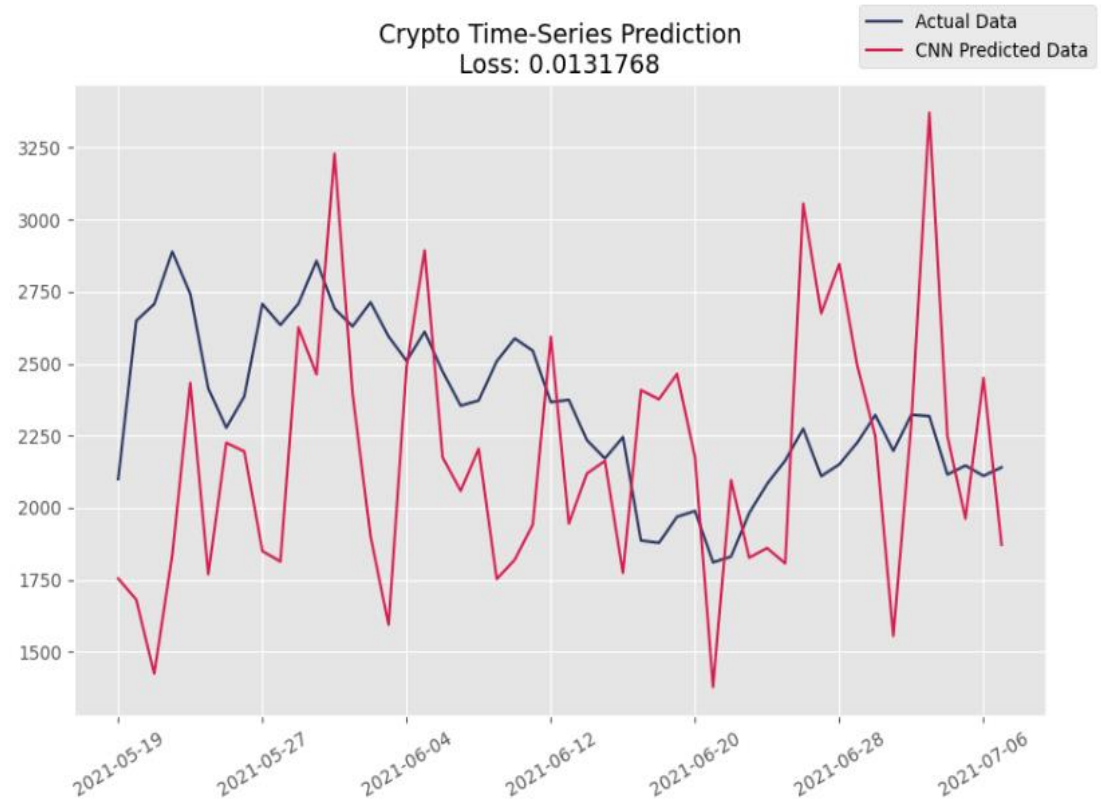


Рисунок 3.14 – Вхідні дані та звичайна згорткова мережа

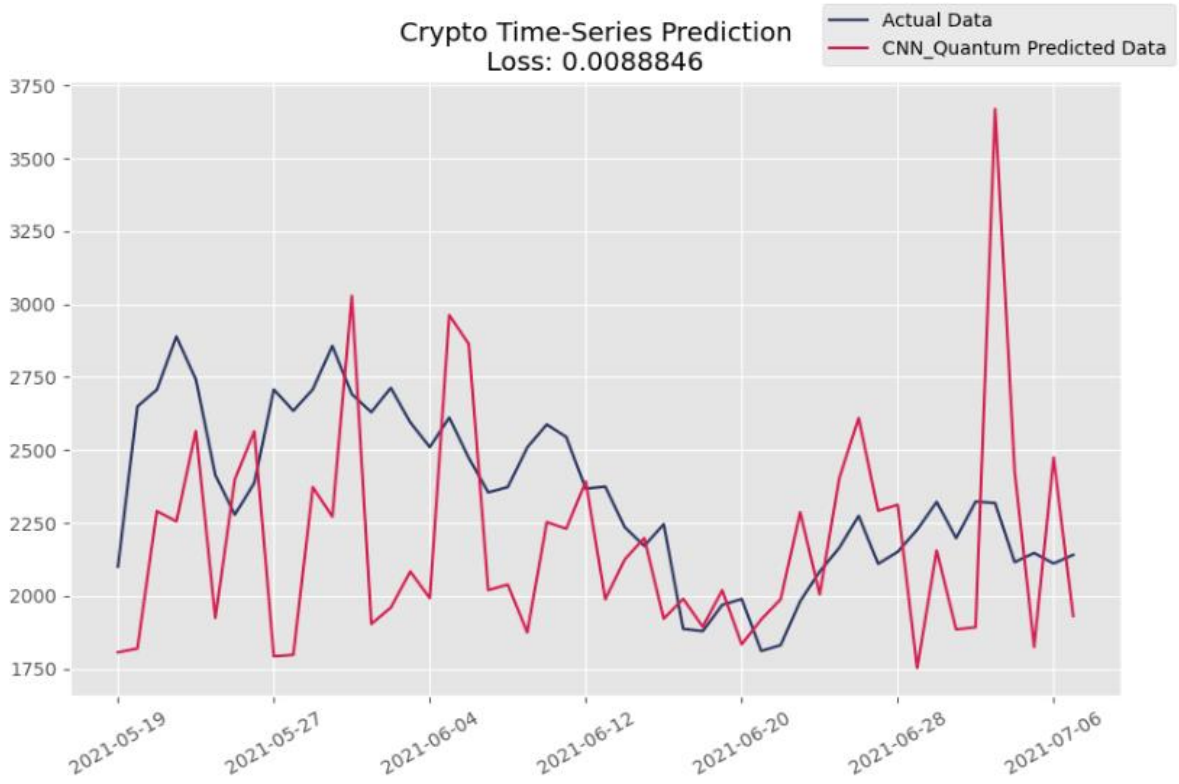


Рисунок 3.15 – Вхідні дані та квантова згорткова мережа

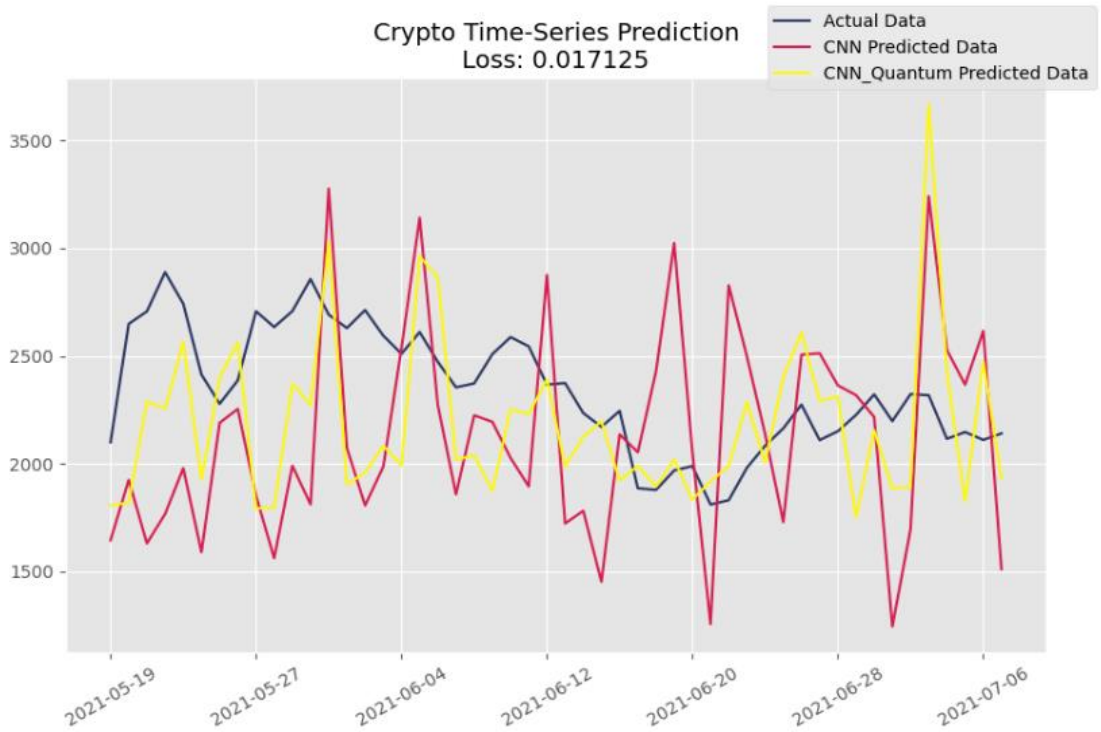


Рисунок 3.16 – Порівняння всіх результатів

Якщо графіка не було створено користувач отримає інформацію про помилку(рисунок 3.17).

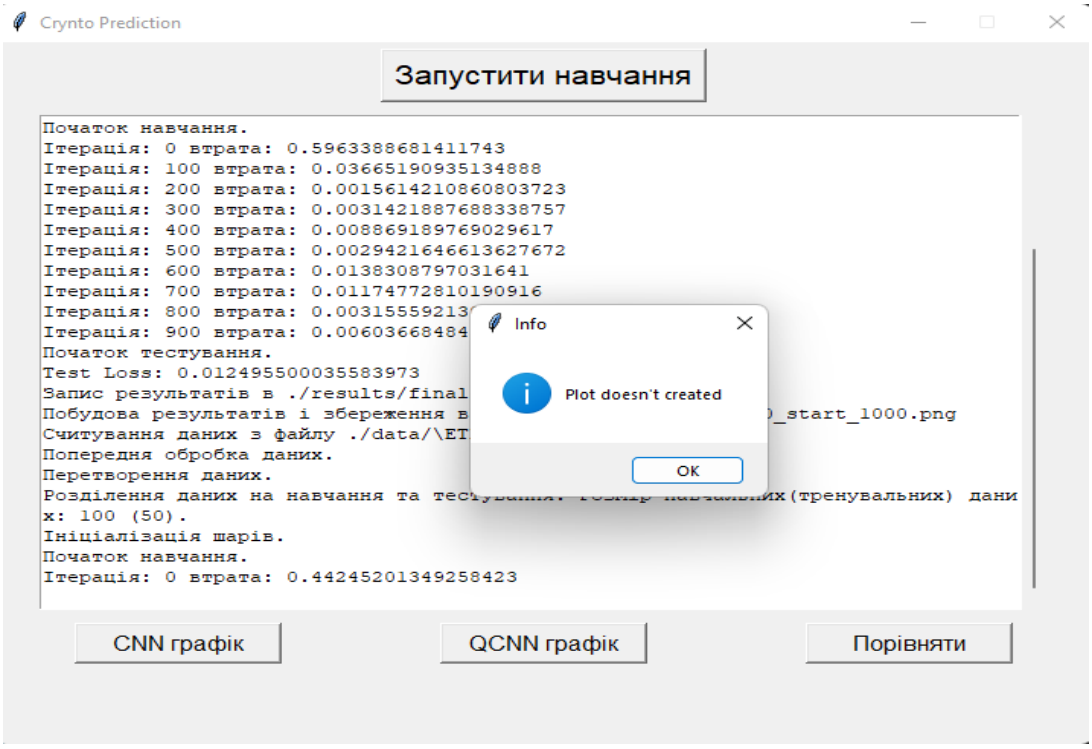


Рисунок 3.17 – Повідомлення про помилку

Отримані графіки зберігаються в окремій директорії в png форматі. Реалізація даного процесу відбувається в методі plot (див. Додаток Б). В методі спочатку формується повний шлях до файлу, куди буде збережений графік.

Далі створюється графічна фігура та одна підграфіка за допомогою plt.subplots(). На графіку будуть відображені фактичні дані та передбачені дані.

Для кожного набору даних використовується різний колір. Легенда графіку містить підписи для фактичних даних, передбачених даних та інших передбачених даних. Осі графіка налаштовуються для відображення дат на осі x. Заголовок графіка містить інформацію про втрату test_loss. Нарешті, графік зберігається у файл за допомогою plt.savefig().

3.3 Програмна реалізація з застосуванням квантових згорткових нейронних мереж в криптоаналізі та шифруванні

У цьому розділі буде представлена програмна реалізація, яка використовує квантові згорткові нейронні мережі (QCNN) для криптоаналізу та шифрування. Вони є потужним інструментом, який комбінує переваги квантових обчислень та здатності згорткових нейронних мереж до розпізнавання зразків.

Реалізація програмного продукту відбувалася за допомогою мови програмування Python. Був використаний модуль Qiskit від IBM Quantum Experience для розробки квантової схеми, яка працює на кубітах замість традиційних бітів.

Розроблена програмна реалізація може бути використана для криптоаналізу різних криптографічних систем та для передбачення їх руйнування шляхом зламу.

Програмне забезпечення та послуги, що використовуються, включають:

- Qiskit: для розробки квантової схеми;
- Tkinter: для створення графічного інтерфейсу;
- Numpy: для обчислювальних операцій;

Зображення графічного інтерфейсу за рисунком 3.18.

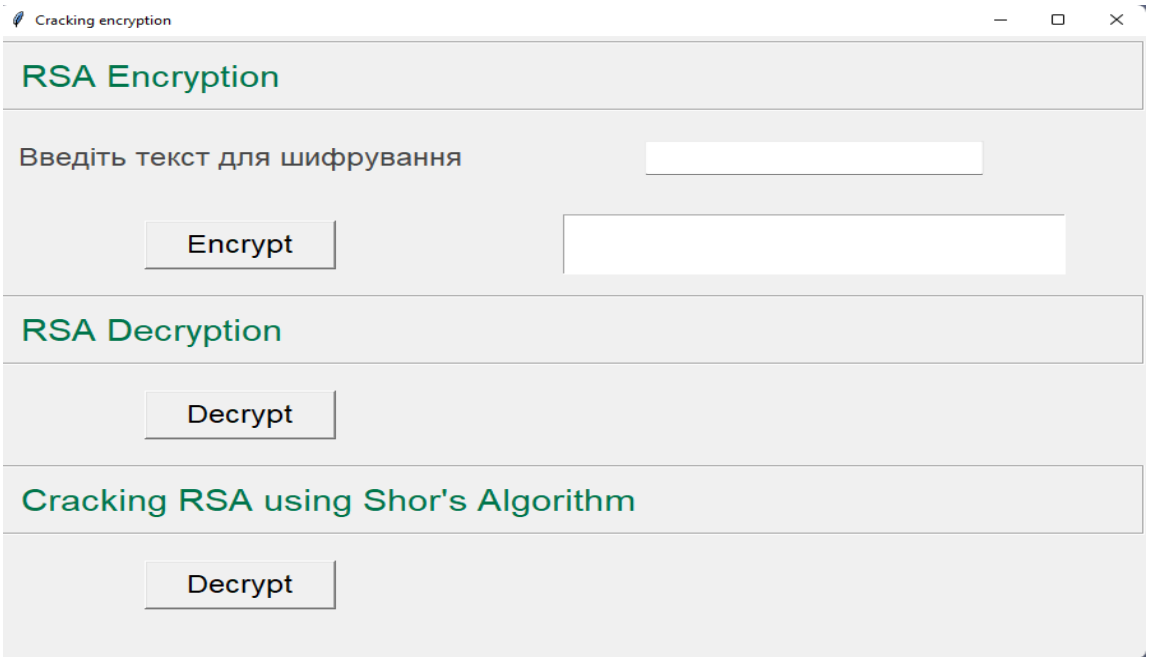


Рисунок 3.18 – Графічний інтерфейс

Графічний інтерфейс містить наступні елементи.

– Поле для вводу тексту: Це текстове поле, в якому користувач може ввести текст, який потрібно зашифрувати. В цьому полі користувач може вводити будь-який текст з бажаними символами, який він бажає зашифрувати.

– Кнопка «Encrypt»: Ця кнопка ініціює шифрування введеного тексту за допомогою алгоритму RSA. Після натискання цієї кнопки, зашифрований текст буде відображений в полі для виводу зашифрованого тексту.

– Поле для виводу зашифрованого тексту: Це текстове поле, в якому буде відображатися зашифрований текст у вигляді числового набору після натискання кнопки «Encrypt».

– Кнопка «Decrypt» в RSA Decryption секції: Ця кнопка ініціює дешифрування зашифрованого тексту за допомогою алгоритму RSA. Після натискання цієї кнопки, розшифрований текст буде відображений в полі для виводу зашифрованого тексту, яке знаходиться біля кпонки.

– Кнопка «Decrypt» в Cracking RSA using Shor's Algorithm секції: Ця кнопка ініціює дешифрування зашифрованого тексту за допомогою алгоритму Шора. Після натискання цієї кнопки, розшифрований текст буде відображений в полі для виводу зашифрованого тексту.

Даний графічний інтерфейс надає зручний спосіб для користувача вводити бажаний ним текст, шифрувати його за допомогою алгоритму RSA, а також дешифрувати зашифрований текст для отримання початкового значення.

Для шифрування введенного тексту використовується RSA алгоритм. Суть алгоритму полягає в генерації двох ключів: публічного та приватного. Публічний ключ використовується для шифрування даних, тоді як приватний ключ використовується для розшифрування.

Головна властивість RSA полягає у тому, що шифрування за допомогою публічного ключа може бути легко виконано, але розшифрування без знання приватного ключа є практично неможливим. Цей алгоритм забезпечує безпеку передачі даних шляхом зашифровування та розшифровування з використанням математичних операцій з числами. Результат роботи шифрування зображено на рисунку 3.19.

За генерування пари ключів відповідає метод generate_keypair (див. Додаток Б), який створює два ключі – публічний та приватний, для використання в криптографічному алгоритмі RSA.

Також створюється об'єкт qasm_sim, який представляє симулятор квантових обчислень у бібліотеці Qiskit. Qiskit є відкритою бібліотекою для квантових обчислень, яка надає інструменти для створення, маніпулювання та виконання квантових обчислень на різних пристроях, включаючи симулятори та реальні квантові комп'ютери.

Метод qiskit.Aer.get_backend('qasm_simulator') (див. Додаток В) отримує симулятор qasm_sim із бекенда Aer, який є одним із симуляторів, що надається в Qiskit. Симулятор qasm_sim дозволяє виконувати квантові обчислення у вигляді QASM (Quantum Assembly) коду, який є мовою опису квантових операцій та вимірювань.

Отже, після виконання даного коду, змінна qasm_sim буде містити об'єкт симулятора квантових обчислень, який можна використовувати для симуляції квантових алгоритмів та вимірювання їх результатів.

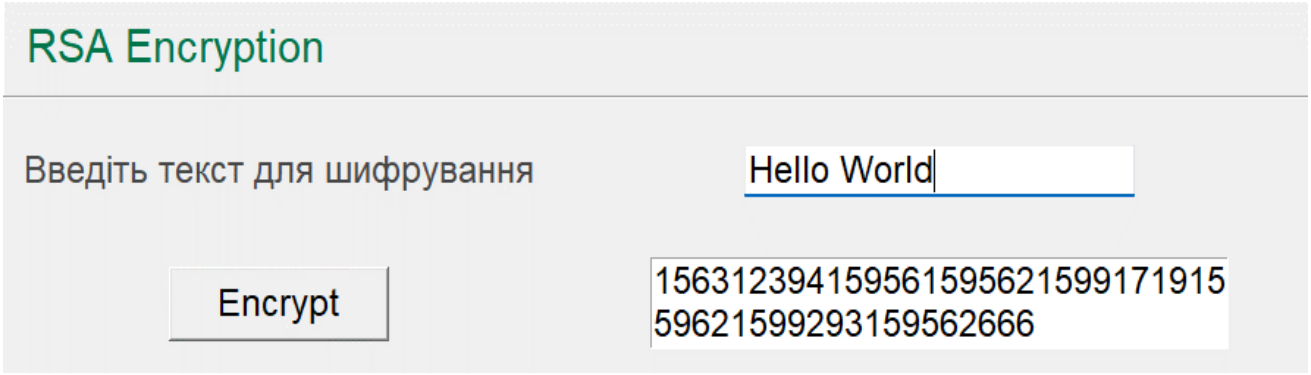


Рисунок 3.19 – Зашифрований текст

Даний процес реалізований за допомогою функції encrypt (див. Додаток В), яка використовує алгоритм RSA для шифрування повідомлення та приймає в себе два значення: msg_plaintext та package. В перший параметр метода передається введений користувачем текст. У параметрах package передаються два значення: e (експонента шифрування) та n (модуль). Ці значення визначають публічний ключ для шифрування.

У тілі функції, повідомлення msg_plaintext розбивається на окремі символи. Для кожного символу з повідомлення виконується обчислення за формулою, яка виконує піднесення значення ASCII-коду символу до степеня e та обчислення остачі при діленні на n. Результатом цих обчислень є шифровані значення символів повідомлення.

Отримані шифровані значення символів зберігаються в змінній msg_ciphertext у вигляді списку. Для зручності, цей список перетворюється в рядок за допомогою функції join, де кожне значення в рядку перетворюється на рядкове представлення за допомогою lambda x: str(x).

На виході функція повертає два значення: зашифрований рядок та сам список зашифрованих значень.

Користувач має змогу вводити будь-які дані від чисел до символів, але при цьому буде вибивати помилка, якщо він ці дані не введе (рисунок 3.20).

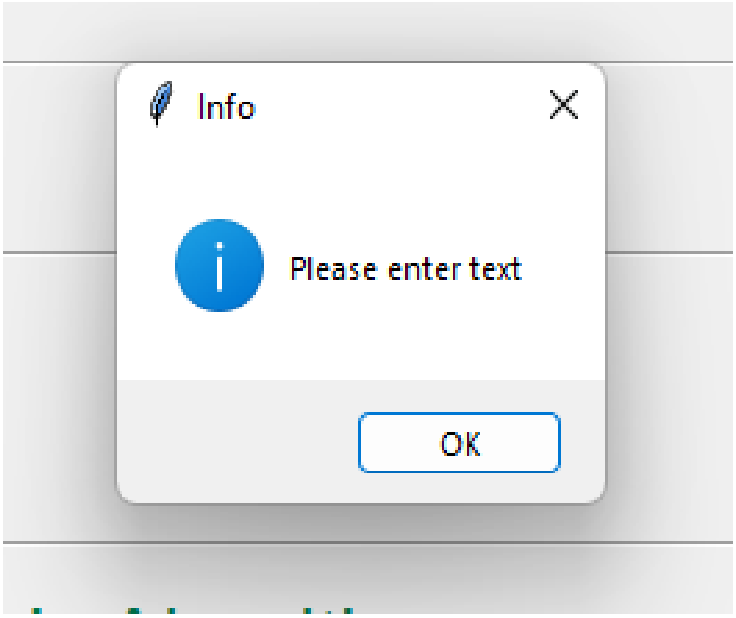


Рисунок 3.20 – Помилка при введенні некоректних даних

Після цього користувач може розшифрувати введений раніше текст за допомогою RSA розшифровки та квантового алгоритму Шора. Даний процес зображений на рисунку 3.21.

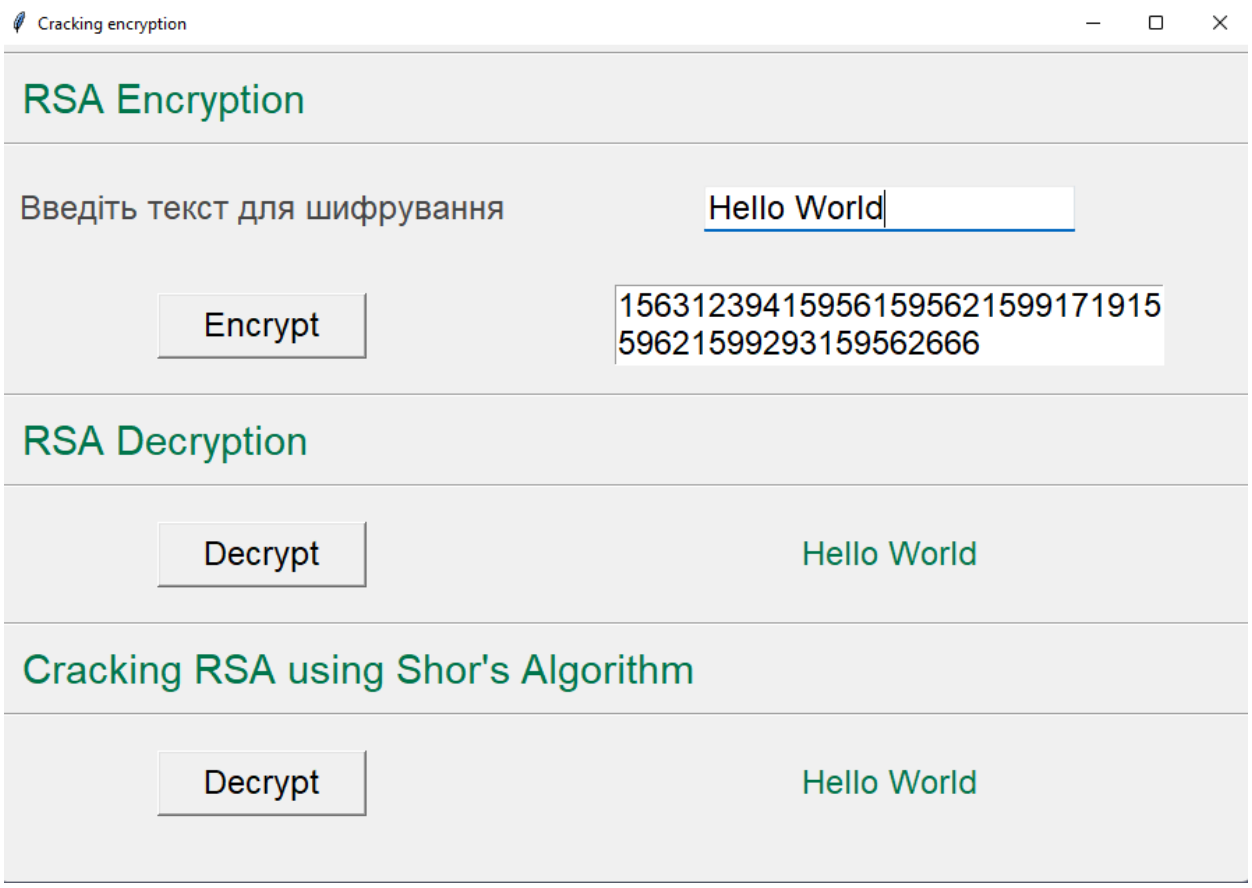


Рисунок 3.21 – Результат роботи програми

Процес RSA розшифровки реалізовується функцією decrypt (див. Додаток В), яка використовує алгоритм RSA для розшифрування зашифрованого повідомлення з використанням пакета ключів. В перший параметр метода передається зашифроване повідомлення. У параметрах package передаються два значення: d (експонента розшифрування) та n (модуль). Ці значення визначають приватний ключ для розшифрування.

У тілі функції, зашифроване повідомлення розбивається на окремі значення. Для кожного шифрованого значення c виконується обчислення за формулою, що виконує піднесення шифрованого значення до степеня d та обчислення остачі при діленні на n. Результатом цих обчислень є відновлені значення символів повідомлення. Відновлені значення символів перетворюються на відповідні символи ASCII-коду за допомогою функції chr(). Ці символи зберігаються в змінній у вигляді списку. На виході функція повертає розшифрований рядок, який отримується з'єднанням всіх символів зі списку msg_plaintext в один рядок.

Основна частина реалізації алгоритму Шора складається з двох методів (додаток А):

– shors_breaker, що використовує алгоритм Шора для факторизації числа N та пошуку його простих множників;

– period, використовується для знаходження періоду функції у модулярній арифметиці за допомогою квантових обчислень на квантовому комп'ютері.

В методі shors_breaker першому рядок коду перетворює вхідний параметр (N в майбутньому) в ціле число за допомогою функції int(). Це забезпечує, що він буде мати правильний числовий тип для подальших обчислень.

У тілі функції виконується цикл while True, що означає безкінечне повторення. У кожній ітерації циклу обирається випадкове значення в діапазоні від 0 до N-1 за допомогою функції randint().

Потім обчислюється найбільший спільний дільник між вхідним параметром та випадковим значенням за допомогою функції gcd(). Якщо НСД не дорівнює 1

або вхідне значення дорівнює 1, то повертаються значення g, N/g. Це означає, що N було розкладено на множники.

Якщо умова if не виконалась, то виконується розрахунок періоду для випадкого числа та N за допомогою методу period(). Далі виконуються декілька перевірок у if-elseif блоках. Якщо період не є парним, цикл продовжується з новою ітерацією. У наступній умові перевіряється, чи pow(a, r//2, N) дорівнює -1. Якщо так, цикл також продовжується з новою ітерацією.

Якщо ж жодна з перевірок не спрацювала, виконуються дві останні перевірки. За допомогою функції gcd() обчислюються значення p та q, які є можливими простими множниками N. Якщо p дорівнює N або q дорівнює N, цикл продовжується з новою ітерацією. Якщо жодна з умов не виконалась, то повертаються значення p та q. Це означає, що алгоритм Шора знайшов прості множники N і успішно здійснив факторизацію.

Для обчислення періоду r числа a за модулем N спочатку перевіряється розмір N. Якщо N більше або дорівнює 2 в степені кількості доступних кубітів (16 у даному випадку), то виводиться повідомлення про те, що число N занадто велике для використання на квантових комп'ютерах IBMQX.

Далі створюються квантові регістри та квантовий ланцюжок. Обирається випадкове початкове значення x0. Значення x_binary обчислюється з числа N у двійковій формі, де кожен біт вказує, чи містить число N певну степінь двійки.

У циклі виконується побітове перетворення x_binary на стан кубітів. Якщо біт в x_binary встановлений, виконується операція, що встановлює кубіт у стан |1⟩. Потім починається цикл, де виконуються операції на квантових логічних воротах. Після цього проводиться вимірювання стану кубітів та отримання результатів виконання квантової програми.

Найбільш ймовірний результат s зберігається та кінцеве значення повертається з функції.

3.4 Висновки до розділу 3

У даному розділі було представлено програмну реалізацію застосування квантових згорткових нейронних мереж для технічних, природничих і соціально-економічних інформаційних систем.

Для фінансоіої сфери було описано клас моделі, що містить у собі реалізацію квантової та класичної нейронної мережі, функції для тренування та прогнозування, а також методи для обробки даних і побудови графіків результатів прогнозування.

Застосування квантових згорткових нейронних мереж дозволяє отримати більш точні прогнози в порівнянні з класичними моделями, зокрема, в областях, де присутні складні залежності між даними. Програмна реалізація показала свою ефективність в прогнозуванні цін криптовалют, а також може бути застосована в інших сферах, де важливо точне прогнозування на основі великої кількості даних.

Застосування квантових згорткових нейронних мереж у криптографії відкриває нові можливості для розробки стійких криптосистем, які забезпечують високий рівень захисту від атак з використанням квантових обчислювальних алгоритмів. Програмна реалізація показала ефективність в дешифруванні інформації, де квантові згорткові нейронні мережі використовуються для розшифрування даних.

У подальших дослідженнях можна розширити обсяг використання квантових згорткових нейронних мереж у криптографії та інших галузях, де вимоги до безпеки та точності є критичними. Також можна подальшувати дослідження в напрямку покращення алгоритмів квантової обчислювальної мережі та їхнього застосування в практичних задачах.

Таким чином, програмна реалізація з застосуванням квантових нейронних мереж для криптування відкриває нові перспективи у сфері криптографії, дозволяючи розробляти стійкі криптосистеми з високим рівнем безпеки і ефективності.

В загальному результати дослідження показали, що квантові нейронні мережі мають потенціал для поліпшення ефективності і точності аналізу даних у різних інформаційних системах. Вони можуть забезпечити швидку обробку великого обсягу даних, а також здатні до виявлення складних залежностей та патернів у вхідних даних.

Висновки

Дана кваліфікаційна робота мала на меті дослідити особливості реалізації квантових згорткових нейронних мереж для технічних, природничих і соціально-економічних інформаційних систем з великим об'ємом даних. Для досягнення цієї мети були визначені наступні завдання дослідження.

– Проведено огляд методів та засобів штучного інтелекту та машинного навчання. Були вивчені існуючі підходи до розвитку нейронних мереж та їх використання для обробки великого об'єму даних.

– Виконано огляд квантових обчислень для великої кількості даних. Були розглянуті принципи квантових обчислень і їх потенціал для ефективної обробки великих обсягів інформації.

– Проаналізовано технічні, природничі та соціально-економічні інформаційні системи, які вимагають підходів на основі нейронних мереж та мають значні обсяги даних. Виявлено основні проблеми, з якими такі системи стикаються при використанні традиційних підходів та потенціал використання квантових згорткових нейронних мереж.

– Запропоновано способи реалізації квантових нейронних мереж для технічних, природничих і соціально-економічних інформаційних систем. Розроблено нові архітектури та алгоритми, які використовують квантові принципи для поліпшення ефективності обробки даних та зменшення обчислювального навантаження.

– Виконано реалізацію квантових нейронних мереж для технічних, природничих і соціально-економічних інформаційних систем. У рамках дослідження були розроблені імплементації квантових згорткових нейронних мереж, що враховують особливості кожної з досліджуваних інформаційних систем. Були проведені експерименти та аналіз результатів для оцінки ефективності цих мереж в різних сферах застосування.

Отримані результати дозволяють зробити наступні висновки.

– Квантові згорткові нейронні мережі мають потенціал для використання в технічних, природничих і соціально-економічних інформаційних системах з великим об'ємом даних. Вони дозволяють здійснювати швидку та ефективну обробку великих обсягів інформації, що є критичним для багатьох сучасних застосувань.

– Огляд методів та засобів штучного інтелекту та машинного навчання виявив, що квантові згорткові нейронні мережі представляють новий підхід, який може покращити точність та швидкість обробки даних порівняно з традиційними нейромережами.

– Аналіз технічних, природничих та соціально-економічних інформаційних систем показав, що багато з них мають великі обсяги даних та потребують нейромережевих підходів для ефективної обробки цих даних. Використання квантових згорткових нейронних мереж може сприяти вирішенню цих проблем та поліпшенню результатів.

Перелік посилань

1. Джон Маккарті [Електронний ресурс] – Режим доступу: https://www-formal.stanford.edu/jmc/whatisai.pdf.
2. Штучний інтелект [Електронний ресурс] – Режим доступу: http://www-formal.stanford.edu/jmc/whatisai/whatisai.html.
3. Глибоке навчання [Електронний ресурс] – Режим доступу: https://web.archive.org/web/20160314152112/http://research.microsoft.com/pubs/209355/DeepLearning-NowPublishing-Vol7-SIG-039.pdf.
4. Convolutional Neural Network [Електронний ресурс] – Режим доступу: https://www.sciencedirect.com/science/article/abs/pii/S0378475420301580?via%3Dihub.
5. Convolutional Neural Network операція згортки [Електронний ресурс] – Режим доступу: https://www.deeplearningbook.org/.
6. Feature Map [Електронний ресурс] – Режим доступу: https://drive.google.com/file/d/0B65v6Wo67Tk5ODRzZmhSR29VeDg/view?resourcekey=0-WZ9_n1lL8vmwKdDTohTbSQ.
7. Зображення згорткової нейронної мережі [Електронний ресурс] – Режим доступу: https://www.mdpi.com/2079-9292/11/24/4100.
8. Квантові комп’ютери [Електронний ресурс] – Режим доступу: https://www.hpcwire.com/2019/01/10/ibm-quantum-update-q-system-one-launch-new-collaborators-and-qc-center-plans/.
9. Квантові обчислення [Електронний ресурс] – Режим доступу: https://citeseer.ist.psu.edu/viewdoc/download;jsessionid=0D01FC6F66C81F349B409BAE3F515CAC?doi=10.1.1.51.5477&rep=rep1&type=pdf.
10. Кубіт [Електронний ресурс] – Режим доступу: https://worldcat.org/title/1076208411.
11. Сфера Блоха [Електронний ресурс] – Режим доступу: https://quantum-journal.org/papers/q-2018-08-06-79/.

12. Зображення порівняння біта та кубіта [Електронний ресурс] – Режим доступу: https://devopedia.org/qubit.
13. Understanding Quantum Entanglement [Електронний ресурс] – Режим доступу: https://www.sciencedirect.com/topics/physics-and-astronomy/quantum-entanglement.
14. Квантова телепортація [Електронний ресурс] – Режим доступу: https://journals.aps.org/prl/pdf/10.1103/PhysRevLett.70.1895.
15. Квантова суперпозиція [Електронний ресурс] – Режим доступу: https://www.vedu.ru/bigencdic/60737/.
16. Експеримент Шредінгера [Електронний ресурс] – Режим доступу: http://pdf.lib.vntu.edu.ua/books/2015/Yuhnov_2002_392.pdf.
17. How Quantum Entanglement Creates Entropy [Електронний ресурс] – Режим доступу: https://www.youtube.com/watch?v=vgYQglmYU-8.
18. Квантова корекція помилок [Електронний ресурс] – Режим доступу: https://iopscience.iop.org/article/10.1088/2058-9565/aa63a4.
19. Quantum error correction [Електронний ресурс] – Режим доступу: https://www.sciencedirect.com/science/article/pii/S2667325820300145?via%3Dihub.
20. Річард Фейман [Електронний ресурс] – Режим доступу: https://books.google.com.ua/books?id=M_ONmDLmGO4C&pg=PA218&lpg=PA218&dq=&redir_esc=y#v=onepage&q&f=false.
21. Квантова хімія [Електронний ресурс] – Режим доступу: https://web.archive.org/web/20160419152451/http://esu.com.ua/search_articles.php?id=11534.
22. The Feynman Lectures on Physics [Електронний ресурс] – Режим доступу: https://www.academia.edu/28997045/The_Feynman_Lectures_on_Physics_VOL1.
23. Shore algorithm [Електронний ресурс] – Режим доступу: https://dl.acm.org/doi/10.1109/SFCS.1994.365700.
24. Quantum Resistant Ledger (QRL) [Електронний ресурс] – Режим доступу: https://www.binance.com/en/price/quantum-resistant-ledger/.

25. Гейт контрольованої НОТ [Електронний ресурс] – Режим доступу: https://tf.nist.gov/general/pdf/140.pdf.
26. Гейт Гадамарда [Електронний ресурс] – Режим доступу: https://ieeexplore.ieee.org/document/1675334.
27. Криптовалюта Ethereum USD [Електронний ресурс] – Режим доступу: https://finance.yahoo.com/quote/ETH-USD/history/

ДОДАТКИ

Додаток А

Перша програма

```
from qiskit.algorithms.minimum_eigensolvers import VQE
from qiskit.algorithms.observables_evaluator import estimate_observables
from qiskit.algorithms.optimizers import COBYLA,                                SLSQP
from qiskit.circuit import QuantumCircuit,                                  Parameter
from qiskit.circuit.library import TwoLocal
from qiskit.quantum_info import Pauli,                                  SparsePauliOp
from qiskit.utils import algorithm_globals
from qiskit_nature.second_q.operators import FermionicOp
from qiskit_nature.second_q.mappers import JordanWignerMapper
import numpy                                    as                              np
import matplotlib.pyplot                              as                        plt
from IPython.display import display,                                  clear_output
from qiskit.primitives import Estimator

def _deltaFunc(n: int,                                              m: int) -> int:
    return int(n == m)

def createDeuteronHamiltonian(
    num_particles: int,     hbar_omega: float = 7.0,     V_0: float = -5.68658111
) ->                                                          SparsePauliOp:

    hamTerms = {}
    for m in range(num_particles):
        for n in range(num_particles):

            label = "+_{}                                -_{}".format(str(n), str(m))
            kinetic_coef = (hbar_omega / 2) * (
                (2 * n + 3 / 2) * _deltaFunc(n,                                  m)
                - np.sqrt(n * (n + (1 / 2))) * _deltaFunc(n,                  m + 1)
                - np.sqrt((n + 1) * (n + (3 / 2)) * _deltaFunc(n,          m - 1))
            )
            hamTerms[label] = kinetic_coef

            potentialCoef = (
                V_0 * _deltaFunc(n, 0) * _deltaFunc(n,                           m)
            )
            hamTerms[label] += potentialCoef

    ham = FermionicOp(hamTerms,                    num_spin_orbitals=num_particles)
```

```
mapper = JordanWignerMapper()
    qubit_ham = mapper.map(ham)
    if not isinstance(qubit_ham,                                    SparsePauliOp):
        qubit_ham = qubit_ham.primitive

    return qubit_ham

deuteronHamiltonians = [createDeuteronHamiltonian(i) for i in range(1, 4)]

for i,                                  ham in enumerate(deuteronHamiltonians):
    print("Тамільтоніан              Дейтерія:              H_{}".format(i + 1))
    print(ham)
    print("\n")

theta = Parameter(r"$\theta$")
eta = Parameter(r"$\eta$")

wavefunc = QuantumCircuit(1)
wavefunc.ry(theta, 0)
wavefunc.draw(output='mpl',  style={'backgroundcolor': '#d5fff3', 'figwidth': 5})

wavefunc2 = QuantumCircuit(2)
wavefunc2.x(0)
wavefunc2.ry(theta, 1)
wavefunc2.cx(1, 0)
wavefunc2.draw(output='mpl', style={'backgroundcolor': '#d5fff3', 'figwidth': 7})

wavefunc3 = QuantumCircuit(3)
wavefunc3.x(0)
wavefunc3.ry(eta, 1)
wavefunc3.ry(theta, 2)
wavefunc3.cx(2, 0)
wavefunc3.cx(0, 1)
wavefunc3.ry(-eta, 1)
wavefunc3.cx(0, 1)
wavefunc3.cx(1, 0)
wavefunc3.draw(output='mpl',
style={'backgroundcolor': '#d5fff3', 'figwidth': 10})
ansatz = [wavefunc,                     wavefunc2,                     wavefunc3]
refValues = []
print("Точні  енергії  зв'язку,  розраховані  через  numpy.linalg.eigh  \n")
for i,                              hamiltonian in enumerate(deuteronHamiltonians):
    evalues,                    estates = np.linalg.eigh(hamiltonian.to_matrix())
```

```
refValues.append(evalues[0])
    print("Точна енергія зв'язку, розрахована через H_{}: {}".format(i + 1,
estates[0]))

print(
    "Результати з використанням оцінювача для H_1, H_2 і H_3 з анзацом \n"
)
for i in range(3):
    seed = 42
    algorithm_globals.random_seed = seed
    vqe = VQE(Estimator(),          ansatz=ansatz[i],          optimizer=SLSQP())
    vqeResult = vqe.compute_minimum_eigenvalue(deuteronHamiltonians[i])
    bindingEnergy = vqeResult.optimal_value
    print("Binding energy for H_{}: {} MeV".format(i + 1, bindingEnergy))

def callback(eval_count,          parameters,          mean,          std):
    display("Оцінка: {}, Енергія: {}, Стандарт: {}".format(eval_count, mean, std))
    clear_output(wait=True)
    counts.append(eval_count)
    values.append(mean)
    params.append(parameters)
    deviation.append(std)

plots = []

for i in range(3):

    counts = []
    values = []
    params = []
    deviation = []
    seed = 42
    algorithm_globals.random_seed = seed
    vqe = VQE(Estimator(),          ansatz=ansatz[i],          optimizer=COBYLA(),
callback=callback)
    vqe_result = vqe.compute_minimum_eigenvalue(deuteronHamiltonians[i])
    plots.append([counts,                                                values])

fig,                    ax = plt.subplots(nrows=3,                    ncols=1)
fig.set_size_inches((12, 12))
for i,                                              plot in enumerate(plots):
    ax[i].plot(plot[0],              plot[1], "o-",              label="COBYLA")
    ax[i].axhline(
```

```
        y=refValues[i],
        color="r",
        linestyle="-",
        label=f"Значення:                                {refValues[i]}",
    )
    ax[i].legend()
    ax[i].set_xlabel("Оцінки        функції        витрат",        fontsize=15)
    ax[i].set_ylabel(r"$\langle  H_{}  \rangle$  -  Енергія  (MeV)".format(i + 1),
fontsize=15)
plt.show()

def calcObservablesValues(
    quantumCirc:       QuantumCircuit,       observables: list,       angles: list
) -> list:

    expValuesList = []
    for obs in observables:
        expValues = []
        for angle in angles:
            qc = quantumCirc.bind_parameters({theta:                        angle})
            result = estimate_observables(
                Estimator(),
                quantum_state=qc,
                observables=[obs],
            )

            expValues.append(result[0][0])
        expValuesList.append(expValues)

    return expValuesList

angles = list(np.linspace(-np.pi,                                  np.pi, 100))
observables = [
    Pauli("IZ"),
    Pauli("ZI"),
    Pauli("XX"),
    Pauli("YY"),
    deuteronHamiltonians[1],
]
obsValues = calcObservablesValues(wavefunc2,         observables,         angles)

fig,                    ax = plt.subplots(nrows=2,                    ncols=1)
fig.set_size_inches((12, 12))
```

```
ax[0].plot(angles,                obsValues[0], "o",                label=r"$Z_0$")
ax[0].plot(angles,                obsValues[1], "o",                label=r"$Z_1$")
ax[0].plot(angles,              obsValues[2], "o",              label=r"$X_0X_1$")
ax[0].plot(angles,              obsValues[3], "o",              label=r"$Y_0Y_1$")
ax[0].axhline(
    y=1,
    color="g",
    linestyle="solid",
)
ax[0].axhline(y=-1,                    color="r",                    linestyle="-")
ax[0].legend()
ax[0].set_xlabel(r"Тета            -            $\theta$",            fontsize=15)
ax[0].set_ylabel(r"$\langle  O  \rangle  $  -  Очікуване  оператором  значення",
fontsize=15)
ax[0].set_xticks(
    [-np.pi, -np.pi / 2, 0,                    np.pi / 2,                    np.pi],
    labels=[r"$-\pi$",       r"$-\pi/2$", "0",       r"$\pi/2$",       r"$\pi$"],
)
ax[0].set_title(
    r"Математичне очікування спостережуваних $Z_0$, $Z_1$, $X_0X_1$ і $Y_0Y_1$,
коли          ми          змінюємо          $\theta$          в          анзаці.",
    fontsize=15,
)
ax[1].plot(angles,                                                obsValues[4], "o")
ax[1].axhline(
    y=refValues[1],
    color="y",
    linestyle="-",
    label="Очікувана    енетрія:    {}    MeV".format(np.round(refValues[1], 3)),
)
ax[1].legend()
ax[1].set_xlabel(r"Тета            -            $\theta$",            fontsize=15)
ax[1].set_ylabel(r"$\langle    H_2    \rangle    $    -    Енергія",    fontsize=15)
ax[1].set_xticks(
    [-np.pi, -np.pi / 2, 0,                    np.pi / 2,                    np.pi],
    labels=[r"$-\pi$",       r"$-\pi/2$", "0",       r"$\pi/2$",       r"$\pi$"],
)
ax[1].set_title(
    r"Поведінка очікуваного значення $H_2$, коли ми змінюємо $\theta$ в анзаці.",
fontsize=15
)

plt.show()
```

Додаток Б

Друга програма

```
import threading
from tkinter import Button,    Frame,    Image,    Label,    Tk,    Toplevel,    font
from tkinter.messagebox import askyesno,                                    showinfo
from tkinter.scrolledtext import ScrolledText
from tkinter.tix import IMAGETEXT
from Container import TextHandler
from conv_model import ConvCryptoTimeSeriesModel
import threading
import logging
import tkinter                                  as                                  tk
from PIL import ImageTk,                                                    Image

files = []
class ProtFiles:
    def __init__(self,                    modeltype,                    filepath):
        self.modeltype = modeltype
        self.filepath = filepath

class App(tk.Frame):
    def __init__(self,                                parent, *args, **kwargs):
        tk.Frame.__init__(self,                                parent, *args, **kwargs)
        self.root = parent

        self.root.title("Crynto                                                Prediction")
        self.root.geometry("720x550")
        self.root.resizable(False, False)

        self.root.columnconfigure(0,                                                weight=1)
        self.root.columnconfigure(1,                                                weight=1)
        self.root.columnconfigure(2,                                                weight=1)

        myFont = font.Font(family='Arial',                                 size=16)
        encButton = Button(self.root,    text="Запустити    навчання",    width=17,
command=self.runTask)
        encButton['font'] = myFont
        encButton.grid(row=0,    column=0,    columnspan=3,    padx=1,    pady=5)

        st = ScrolledText(state='disabled')
        st.configure(font='TkFixedFont')
```

```
st.grid(row=1,        column=0,        columnspan=3,        padx=1,        pady=5)

        bntFont = font.Font(family='Arial',                                size=12)
        self.cnnButton = Button(self.root,     text="CNN     графік",     width=14,
font=bntFont,                                              command=self.openCnnPlot)
        self.cnnButton.grid(row=3,        column=0,        padx=1,        pady=5)

        qcnnButton = Button(self.root, text="QCNN графік", width=14, font=bntFont,
command=self.openQcnnPlot)
        qcnnButton.grid(row=3,         column=1,         padx=1,         pady=5)

        compareButton = Button(self.root,       text="Порівняти",       width=14,
font=bntFont,                                          command=self.openMultiPlot)
        compareButton.grid(row=3,         column=2,         padx=1,         pady=5)

        #                              Create                              textLogger
        text_handler = TextHandler(st)

         #          Add          the          handler          to          logger
        logger = logging.getLogger()
        logger.addHandler(text_handler)

    def openCnnPlot(self):
        file = self.findFirst("CNN")
        print(files)
        if(file == None):
            showinfo(title="Info",      message="Plot      doesn't      created")
            return
        window = Toplevel()
        window.title("CNN                                                  Plot")
        window.geometry("900x700")

        frame = Frame(window,               width=600,               height=400)
        frame.pack()
        frame.place(anchor='center',             relx=0.5,             rely=0.5)

        #       Create       an       object       of       tkinter       ImageTk
        img = Image.open(file)
        img = img.resize((900, 700),                                Image.ANTIALIAS)
        img = ImageTk.PhotoImage(img)

        #   Create   a   Label   Widget   to   display   the   text   or   Image
        label = Label(frame,                                          image = img)
```

```
    label.pack()
    window.mainloop()


def openQcnnPlot(self):
    file = self.findFirst("CNN_Quantum")
    if(file == None):
        showinfo(title="Info",      message="Plot      doesn't      created")
        return
    window = Toplevel()
    window.title("CNN                                                Plot")
    window.geometry("900x700")

    frame = Frame(window,               width=600,               height=400)
    frame.pack()
    frame.place(anchor='center',             relx=0.5,             rely=0.5)

    #       Create       an       object       of       tkinter       ImageTk
    img = Image.open(file)
    img = img.resize((900, 700),                            Image.ANTIALIAS)
    img = ImageTk.PhotoImage(img)

    #   Create   a   Label   Widget   to   display   the   text   or   Image
    label = Label(frame,                                        image = img)
    label.pack()
    window.mainloop()

def openMultiPlot(self):
    file = self.findFirst("Multi")
    if(file == None):
        showinfo(title="Info",      message="Plot      doesn't      created")
        return
    window = Toplevel()
    window.title("CNN                                                Plot")
    window.geometry("900x700")

    frame = Frame(window,               width=600,               height=400)
    frame.pack()
    frame.place(anchor='center',             relx=0.5,             rely=0.5)

    #       Create       an       object       of       tkinter       ImageTk
    img = Image.open(file)
    img = img.resize((900, 700),                            Image.ANTIALIAS)
```

```
        img = ImageTk.PhotoImage(img)

        # Create a Label Widget to display the text or Image
        label = Label(frame, image = img)
        label.pack()
        window.mainloop()

    def findFirst(self, type):
        for i in files:
            if i.modeltype == type:
                return i.filepath

    def runTask(self):
        result = askyesno(message="Підтвердити операцію?")
        if result:
            files.clear()
            t1.start()

def runProccess():
        file = "\ETH-USD.csv"
        indexcolumn =  "Date"
        ycolumn = 0
        finalFile = "final.csv"
        trainNumber = 125
        testNumber = 250
        sizeBatch = 5
        startInd = 1000
        lookback = 10
        lr = .005
        ytest = None
        ypreds = {}

        cnnModel = ConvCryptoTimeSeriesModel(
            num_train = trainNumber,
            num_test = testNumber,
            iterations = 1000,
            lr = lr,
            batch_size = sizeBatch,
            start_index = startInd,
            lookback = lookback,
            conv = [128, 64],
            quantum=False
        )
```

```
        quantumModel = ConvCryptoTimeSeriesModel(
            num_train = trainNumber,
            num_test = testNumber,
            iterations = 1000,
            lr = lr,
            batch_size = sizeBatch,
            start_index = startInd,
            lookback = lookback,
            conv = [128, 64],
            quantum=True
        )
        models = [cnnModel,                                          quantumModel]

        for i in range(len(models)):
            model = models[i]
            model.readFile(file,indexcolumn)
            model.preprocess(y_col = ycolumn)
            if type(model) == ConvCryptoTimeSeriesModel:
model.initialize_layers()
            model.train()
            model.test()
            y_test,                          y_pred_i = model.inversed_transform_y()
            model.writeToFile(writefile=finalFile)
            if i                 > 0:                 ypreds[model.type] = y_pred_i
            plotfile = "{}_class_{}_start_{}.png".format(model.type,  model.y_col,
startInd)
            file = model.definePlot(plotfile=plotfile)
            files.append(ProtFiles(model.type,file))
            print(files)


        if len(models)                                                       > 1:
            file = cnnModel.plot(plotfile="Multiple_class_{}_index_{}.png".format
(cnnModel.y_col,                    startInd),                    y_preds=ypreds)
            files.append(ProtFiles("Multi",file))


t1 = threading.Thread(target=runProccess,                                 args=[])


def main():

    root = tk.Tk()
```

```
App(root)

    root.mainloop()
    t1.join()

main()


from asyncore import write
import csv
import tkinter                                  as                                  tk
from tkinter import Button,                Text,                Tk,                ttk
import tkinter.font                               as                               font

import pandas                                  as                                  pd
import numpy                                  as                                  np
import matplotlib.pyplot                                as                                plt

import torch.nn                                  as                                  nn
from torch import Tensor,                                                  reshape
from torch.autograd import Variable
from sklearn.preprocessing import StandardScaler,                        MinMaxScaler

import logging
logging.basicConfig()
logger = logging.getLogger(__name__)
logger.setLevel(logging.INFO)

global number_of_qubits
number_of_qubits = 6

import string
import random

def generate_Id(size=4,
chars=string.ascii_uppercase + string.ascii_lowercase + string.digits):
    return ''.join(random.choice(chars) for _ in range(size))


class CryptoTimeSeriesModel(nn.Module):
    def __init__(self,
        trainNumber,  testNumber,  iters,  time,  sizeBatch,startindex,  lookback,
        isQuantum=None,                cnn=None,                layersNumber = None
```

```
    ) -> None:
        logger.info("Ініціалізація                                моделі.")
        super(CryptoTimeSeriesModel, self).__init__()
        self.trainNumber = trainNumber
        self.testNumber = testNumber
        self.iters = iters
        self.time = time
        self.sizeBatch = sizeBatch
        self.criterion = nn.MSELoss()
        self.startindex = startindex
        self.lookback= lookback
        self.isQuantum = isQuantum
        self.cnn = cnn
        self.layersNumber = layersNumber
        self.weights = None
        self.bias = None
        self.id = generate_Id()

    def readFile(self,                  datafile,                  index_col):
        full_datafile = "./data/" + datafile
        logger.info("Считування   даних   з   файлу   {}".format(full_datafile))
        self.df = pd.read_csv(full_datafile,              index_col = index_col,
parse_dates=True)
        self.dates = pd.read_csv(full_datafile)["Date"][self.startindex + self.tr
ainNumber: self.startindex + self.trainNumber + self.testNumber]


    def preprocess(self,                                          y_col):
        logger.info("Попередня                обробка                даних.")
        self.y_col = y_col

        raw_data = self.df.to_numpy()
        data = []
        for index in range(len(raw_data) - self.lookback):
            data.append(raw_data[index:                  index + self.lookback])

        X = np.array(data)[:,                      :-1,                      :]
        y = np.array(data)[:, -1,                                            :]
        y = np.array([elt[y_col] for elt in y]).reshape(y.shape[0], 1)
        logger.debug("X   shape   {},   y   shape   {}".format(X.shape,   y.shape))

        logger.info("Перетворення                                        даних.")
        self.mm = MinMaxScaler()
```

```
        self.ss = StandardScaler()
        X = np.array([self.ss.fit_transform(x) for x in X])
        y = self.mm.fit_transform(y)

        logger.info("Розділення даних на навчання та тестування. Розмір навчальних(тренувальних) даних: {} ({}).".format(self.trainNumber, self.testNumber))
        self.X_train_1 = X[self.startindex:self.startindex+self.trainNumber, :]
        self.X_test_1 = X[self.startindex+self.trainNumber:self.startindex +self.trainNumber + self.testNumber, :]
        self.y_train_1 = y[self.startindex:self.startindex+self.trainNumber, :]
        self.y_test_1 = y[self.startindex+self.trainNumber:self.startindex+self.trainNumber + self.testNumber, :]

        X_train_2  = Variable(Tensor(self.X_train_1))
        X_test_2 = Variable(Tensor(self.X_test_1))
        y_train_2 = Variable(Tensor(self.y_train_1))
        y_test_2 = Variable(Tensor(self.y_test_1))

        self.X_train = reshape(X_train_2, (X_train_2.shape[0],1, X_train_2.shape[2] * X_train_2.shape[1]))
        self.X_test = reshape(X_test_2, (X_test_2.shape[0], 1, X_test_2.shape[2] * X_test_2.shape[1]))
        self.y_train = y_train_2
        self.y_test = y_test_2

        self.y_train_1 = reshape(Variable(Tensor(self.y_train)), (self.y_train.shape[0],1))
        self.y_test_1 = reshape(Variable(Tensor(self.y_test)), (self.y_test.shape[0],1))

    def writeToFile(self, writefile) -> None:
        full_writefile = "./results/" + writefile
        logger.info("Запис результатів в {}".format(full_writefile))
        with open(full_writefile, 'a', newline='') as csv_file:
            writer = csv.writer(csv_file, delimiter=',')
            row = [
                self.id, self.type, self.y_col,
                round(self.train_time,3), round(self.test_time,3), self.trainNumber, self.testNumber,
                self.startindex,
                self.sizeBatch, self.iters, self.time, self.train_loss, self.test_loss,
```

```
                self.layersNumber, self.cnn, self.isQuantum, self.weights, self.b
ias
            ]
            row = row[0:-2]
            if self.type == "isQuantumSequenceCircuit":            row = row[0:-2]
            writer.writerow(row)

    def inversed_transform_y(self):
        self.invtransformed_y_test = self.mm.inverse_transform(
            reshape(Variable(Tensor(self.y_data_test)),  (self.y_data_test.shape[
0], 1)))
        self.invtransformed_y_predict = self.mm.inverse_transform(
            reshape(Variable(Tensor(self.y_data_predict)),  (self.y_data_predict.
shape[0], 1)))
        return self.invtransformed_y_test, self.invtransformed_y_predict

    def definePlot(self,              plotfile,              y_preds = {}) -> str:
        full_plotfile= "./plots/" + self.id + "_" + plotfile
        logger.info("Побудова       результатів       і       збереження       в
{}".format(full_plotfile))
        plt.style.use('ggplot')
        figure,              (ax) = plt.subplots(1,1,              figsize=(10,6))
        display_dates = np.array(self.dates)
        ax.plot(display_dates, self.invtransformed_y_test, label='Фактичні дані',
color='#293462')
        ax.plot(display_dates, self.invtransformed_y_predict,           label='{}
Прогнозовані               дані'.format(self.type),              color='#D61C4E')
        for y_pred_type in y_preds.keys():
            ax.plot(display_dates,  y_preds[y_pred_type],  label='{}  Прогнозовані
дані'.format(y_pred_type),                                          color='#FFF80A')
        ax.legend(loc ='upper         right',         bbox_to_anchor=(1.06, 1.14))
        ax.set_xticks(display_dates[::8])
        ax.set_xticklabels(display_dates[::8],                      rotation=30)
        ax.set_title('Прогнозування                              криптовалюти\nВтрата:
{}'.format(round(float(self.test_loss),7)),                              fontsize=14)
        plt.savefig(full_plotfile)
        return full_plotfile
import time
from matplotlib.pyplot import xkcd
import numpy                                as                                  np
import pennylane                              as                                qml

import torch
```

```
import torch.nn                                    as                                    nn
from pennylane.templates import AngleEmbedding,          StronglyEntanglingLayers

import logging
logging.basicConfig()
logger = logging.getLogger(__name__)
logger.setLevel(logging.INFO)

from model import CryptoTimeSeriesModel,                              number_of_qubits
from utils import layer

dev = qml.device("default.qubit",                              wires=number_of_qubits)

class ConvCryptoTimeSeriesModel(CryptoTimeSeriesModel):
    def __init__(self,
        num_train,  num_test,  iterations,  lr,  batch_size,start_index,  lookback,
conv,                                                                        quantum
    ) -> None:
        super(ConvCryptoTimeSeriesModel, self).__init__(num_train,       num_test,
iterations,  lr,  batch_size,start_index,  lookback,  conv=conv,  quantum=quantum)
        if self.quantum: self.type =  "CNN_Quantum"
        else: self.type = "CNN"

    def initialize_layers(self,                                                    ):
        logger.info("Ініціалізація                                    шарів.")

        self.input_channels, self.output_size = 6*(self.lookback- 1), 1
        self.num_layers = 5
        weight_shapes = {"weights":    (self.num_layers,    number_of_qubits, 3)}

        self.relu = nn.ReLU(inplace=True)
        self.conv1 = nn.Conv1d(
            in_channels=1,  out_channels=self.conv[0],  kernel_size=6,  stride=6,
padding=5,                                                              bias=True
        )
        self.pool1 = nn.MaxPool1d(2)
        self.conv2 = nn.Conv1d(
            in_channels=self.conv[0],  out_channels=self.conv[1],  kernel_size=3,
stride=3,                          padding=5,                           bias=True
        )
        self.pool2 = nn.MaxPool1d(3)
        self.fc1 = nn.Linear(self.conv[1], 6)
```

```
        if self.quantum:
            self.qlayer = qml.qnn.TorchLayer(qnode = self.circuit,
weight_shapes = weight_shapes)

        else:
            self.fc2 = nn.Linear(6, 6)
        self.fc3 = nn.Linear(6, 1)
        self.optimizer = torch.optim.Adam(self.parameters(), self.lr)

    def forward(self,x):
        logger.debug("Input              Shape:              {}".format(x.shape))
        self.input_size = x.size(0)
        x = self.relu(self.conv1(x))
        logger.debug("Shape:                                {}".format(x.shape))
        x = self.pool1(x)
        logger.debug("Shape:                                {}".format(x.shape))
        x = self.relu(self.conv2(x))
        logger.debug("Shape:                                {}".format(x.shape))
        x = self.pool2(x)
        logger.debug("Shape:                                {}".format(x.shape))
        x = x.view(self.input_size, -1)
        logger.debug("Shape:                                {}".format(x.shape))
        x = self.fc1(x)
        logger.debug("Shape:                                {}".format(x.shape))
        if self.quantum:
            x = self.qlayer(x)
            logger.debug("Shape:                            {}".format(x.shape))
        else:
            x = self.fc2(x)
        x = self.fc3(x)
        logger.debug("Shape:                                {}".format(x.shape))
        return x

    def train(self) -> None:
        logger.info("Початок                                    навчання.")
        start = time.time()
        for iteration in range(self.iterations):
            batch_index = np.random.randint(0, self.y_train.shape[0],
(self.batch_size))
            X_batch = self.X_train[batch_index]
            y_batch = self.y_train[batch_index]
            outputs = self.forward(X_batch)
            self.optimizer.zero_grad()
```

```
            logger.debug("X_batch.shape {} y_batch.shape {}".format(X_batch.shape,
y_batch.shape))
            loss = self.criterion(outputs,                                  y_batch)
            loss.backward()
            self.optimizer.step()
            if iteration % 100 == 0:     logger.info("Ітерація:     {}     втрата:
{}".format(iteration,                                  loss.detach().numpy()))
        self.train_loss = loss.detach().numpy()
        self.train_time = time.time() - start
        logger.debug("Втрата:                                  {}".format(loss))

    def test(self) -> None:
        logger.info("Початок                                  тестування.")
        start = time.time()
        self.y_pred = self.forward(self.X_test)
        self.y_data_predict = self.y_pred.data.numpy()
        self.y_data_test = self.y_test.data.numpy()
        self.test_loss = self.criterion(self.y_pred, self.y_test).detach().numpy(
)
        logger.info("Test            Loss:            {}".format(self.test_loss))
        self.test_time = time.time() - start

        logger.debug("Тест
{}\Прогноз{}".format(self.y_data_test, self.y_data_predict))
        logger.debug("Тест
{}\nПрогноз{}".format(self.y_data_test.shape, self.y_data_predict.shape))

    @qml.qnode(dev)
    def circuit(inputs,                                                 weights):
        logger.debug("Call     circuit     {}     {}".format(inputs,     weights))
        AngleEmbedding(inputs,                      wires=range(number_of_qubits))
        StronglyEntanglingLayers(weights,           wires=range(number_of_qubits))
        return [qml.expval(qml.PauliZ(wires=i)) for i in range(number_of_qubits)]

    @qml.qnode(dev)
    def circuit2(inputs,                                                weights):
        AngleEmbedding(inputs,                      wires=range(number_of_qubits))
        for weight in weights:
            layer(weight)

        return [qml.expval(qml.PauliZ(wires=i)) for i in range(number_of_qubits)]
```

```
import pennylane as qml

from model import number_of_qubits

def square_loss(labels, predictions):
    loss = 0
    for l, p in zip(labels, predictions):
        loss += (l - p) ** 2
    loss = loss / len(labels)
    return loss

def layer(weights):
    for wire in range(number_of_qubits - 1):
        qml.RX(weights[wire, 0], wires=[wire])
        qml.RY(weights[wire, 1], wires=[wire])
        qml.RZ(weights[wire, 2], wires=[wire])
        qml.CNOT(wires=[wire, wire + 1])

def full_layer(weights):
    for wire in range(number_of_qubits):
        qml.RX(weights[wire, 0], wires=[wire])
        qml.RY(weights[wire, 1], wires=[wire])
        qml.RZ(weights[wire, 2], wires=[wire])
    for wire in range(number_of_qubits-1):
        for wire2 in range(wire + 1, number_of_qubits):
            qml.CNOT(wires=[wire, wire2])
```

Додаток В

Третій додаток

```
import tkinter as tk
from tkinter import Button, StringVar, Text, ttk
import tkinter.font as font
from tkinter.messagebox import showinfo

from Program.Decryption import Decryption

class App(tk.Tk):

    def __init__(self):
        super().__init__()

        self.geometry("900x600")
        self.configure(bg="#f0f0f0")
        self.title("Cracking encryption")

        # configure the grid
        self.columnconfigure(0, weight=1)
        self.columnconfigure(1, weight=6)

        self.inputed_text = StringVar()
        self.dectypted_text = StringVar()
        self.shor_result_text = StringVar()
        self.rsa_result_text = StringVar()
        self.enc_onj = list
        self.dec = Decryption()
        self.Text = Text()

        self.create_widgets()

    def create_widgets(self):
        encryptLabel = ttk.Label(self, text="RSA Encryption", font=("Arial", 22),
borderwidth=5, relief="groove", foreground="#00754B", padding=15, justify="left",
width="500")
        encryptLabel.grid(row=0, column=0, columnspan=2, padx=1, pady=5)


        # Створення етикетки "Введіть текст для шифрування"
        encryptText = ttk.Label(self, text="Введіть текст для шифрування",
```

```
font=("Arial", 18), foreground="#474747", justify="left")
        encryptText.grid(row=1, column=0,ipady=20, padx=5, pady=5)

        # Створення поля вводу
        input_entry = ttk.Entry(self, font=("Arial", 18),
textvariable=self.inputed_text)
        input_entry.grid(row=1, column=1, padx=5, pady=5)

        # Кнопка шифрування
        myFont = font.Font(family='Arial', size=18)
        encButton = Button(self, text="Encrypt", width=10,
command=self.encrypt_command)
        encButton['font'] = myFont
        encButton.grid(row=2, column=0, padx=5, pady=20)


        # Результат шифрування
        self.Text["font"] = ("Arial,",18)
        self.Text["height"] = 2
        self.Text["width"] = 30
        self.Text.grid(row=2, column=1, padx=5, pady=5)


        # Створення етикетки "RSA розшифрування"
        rsaDecypt = ttk.Label(self, text="RSA Decryption", font=("Arial", 22),
borderwidth=4, relief="groove",foreground="#00754B", padding=15, justify="left",
width="500")
        rsaDecypt.grid(row=3, column=0, columnspan=2, padx=1, pady=5)

        # Кнопка дешифрування rca
        rsaButton = Button(self, text="Decrypt", width=10,
command=self.decrypt_rsa_command)
        rsaButton['font'] = myFont
        rsaButton.grid(row=4, column=0, padx=5, pady=20)

        # Результат дешифрування
        rsaResultLabel = ttk.Label(self, font=("Arial", 18),
foreground="#00754B", textvariable=self.rsa_result_text)
        rsaResultLabel.grid(row=4, column=1, padx=5, pady=5)


        # Створення етикетки "Cracking RSA using Shor's Algorithm"
        rsaDecypt = ttk.Label(self, text="Cracking RSA using Shor's Algorithm",
```

```
font=("Arial", 22),borderwidth=5, relief="groove", foreground="#00754B",
padding=15, justify="left", width="500")
        rsaDecypt.grid(row=5, column=0, columnspan=2, padx=1, pady=5)

        # Кнопка шифрування шор
        shorButton = Button(self, text="Decrypt", width=10,
command=self.decrypt_shor_command)
        shorButton['font'] = myFont
        shorButton.grid(row=6, column=0 ,padx=5, pady=20)

        # Результат дешифрування шор
        shorResultLabel = ttk.Label(self, font=("Arial", 18),
foreground="#00754B", textvariable=self.shor_result_text)
        shorResultLabel.grid(row=6, column=1, padx=5, pady=5)

    def encrypt_command(self):
        t = self.inputed_text.get()
        if(t == ""):
            showinfo(title="Info", message="Please enter text")
        else:
            encrypted_msg, encryption_obj = self.dec.encrypt_text(t)
            self.dectypted_text.set(encrypted_msg)
            self.Text.insert("1.0", encrypted_msg)
            self.enc_onj = encryption_obj

    def decrypt_rsa_command(self):
        ect_text = self.dec.decrypt(False, self.enc_onj)
        self.rsa_result_text.set(ect_text)

    def decrypt_shor_command(self):
        ect_text = self.dec.decrypt(True, self.enc_onj)
        self.shor_result_text.set(ect_text)

import time
from matplotlib.pyplot import xkcd
import numpy as np
import pennylane as qml

import torch
import torch.nn as nn
from pennylane.templates import AngleEmbedding, StronglyEntanglingLayers

import logging
```

```
logging.basicConfig()
logger = logging.getLogger(__name__)
logger.setLevel(logging.INFO)

from model import CryptoTimeSeriesModel, number_of_qubits
from utils import layer

dev = qml.device("default.qubit", wires=number_of_qubits)

class ConvCryptoTimeSeriesModel(CryptoTimeSeriesModel):
    def __init__(self,
        num_train, num_test, iterations, lr, batch_size,start_index, lookback,
conv, quantum
    ) -> None:
        super(ConvCryptoTimeSeriesModel, self).__init__(num_train, num_test,
iterations, lr, batch_size,start_index, lookback, conv=conv, quantum=quantum)
        if self.quantum: self.type =  "CNN_Quantum"
        else: self.type = "CNN"

    def initialize_layers(self, ):
        logger.info("Ініціалізація шарів.")

        self.input_channels, self.output_size = 6*(self.lookback- 1), 1
        self.num_layers = 5
        weight_shapes = {"weights": (self.num_layers, number_of_qubits, 3)}

        self.relu = nn.ReLU(inplace=True)
        self.conv1 = nn.Conv1d(
            in_channels=1, out_channels=self.conv[0], kernel_size=6, stride=6,
padding=5, bias=True
        )
        self.pool1 = nn.MaxPool1d(2)
        self.conv2 = nn.Conv1d(
            in_channels=self.conv[0], out_channels=self.conv[1], kernel_size=3,
stride=3, padding=5, bias=True
        )
        self.pool2 = nn.MaxPool1d(3)
        self.fc1 = nn.Linear(self.conv[1], 6)
        if self.quantum:
            self.qlayer = qml.qnn.TorchLayer(qnode = self.circuit,
weight_shapes = weight_shapes)

        else:
```

```
self.fc2 = nn.Linear(6, 6)
        self.fc3 = nn.Linear(6, 1)
        self.optimizer = torch.optim.Adam(self.parameters(), self.lr)

    def forward(self,x):
        logger.debug("Input Shape: {}".format(x.shape))
        self.input_size = x.size(0)
        x = self.relu(self.conv1(x))
        logger.debug("Shape: {}".format(x.shape))
        x = self.pool1(x)
        logger.debug("Shape: {}".format(x.shape))
        x = self.relu(self.conv2(x))
        logger.debug("Shape: {}".format(x.shape))
        x = self.pool2(x)
        logger.debug("Shape: {}".format(x.shape))
        x = x.view(self.input_size, -1)
        logger.debug("Shape: {}".format(x.shape))
        x = self.fc1(x)
        logger.debug("Shape: {}".format(x.shape))
        if self.quantum:
            x = self.qlayer(x)
            logger.debug("Shape: {}".format(x.shape))
        else:
            x = self.fc2(x)
        x = self.fc3(x)
        logger.debug("Shape: {}".format(x.shape))
        return x

    def train(self) -> None:
        logger.info("Початок навчання.")
        start = time.time()
        for iteration in range(self.iterations):
            batch_index = np.random.randint(0, self.y_train.shape[0],
(self.batch_size))
            X_batch = self.X_train[batch_index]
            y_batch = self.y_train[batch_index]
            outputs = self.forward(X_batch)
            self.optimizer.zero_grad()
            logger.debug("X_batch.shape {} y_batch.shape
{}".format(X_batch.shape, y_batch.shape))
            loss = self.criterion(outputs, y_batch)
            loss.backward()
            self.optimizer.step()
```

```
if iteration % 100 == 0: logger.info("Ітерація: {} втрата: {}".format(iteration,
loss.detach().numpy()))
        self.train_loss = loss.detach().numpy()
        self.train_time = time.time() - start
        logger.debug("Втрата: {}".format(loss))

    def test(self) -> None:
        logger.info("Початок тестування.")
        start = time.time()
        self.y_pred = self.forward(self.X_test)
        self.y_data_predict = self.y_pred.data.numpy()
        self.y_data_test = self.y_test.data.numpy()
        self.test_loss = self.criterion(self.y_pred, self.y_test).detach().numpy(
)
        logger.info("Test Loss: {}".format(self.test_loss))
        self.test_time = time.time() - start

        logger.debug("Тест
{}\Прогноз{}".format(self.y_data_test, self.y_data_predict))
        logger.debug("Тест
{}\nПрогноз{}".format(self.y_data_test.shape, self.y_data_predict.shape))

    @qml.qnode(dev)
    def circuit(inputs, weights):
        logger.debug("Call circuit {} {}".format(inputs, weights))
        AngleEmbedding(inputs, wires=range(number_of_qubits))
        StronglyEntanglingLayers(weights, wires=range(number_of_qubits))
        return [qml.expval(qml.PauliZ(wires=i)) for i in range(number_of_qubits)]

    @qml.qnode(dev)
    def circuit2(inputs, weights):
        AngleEmbedding(inputs, wires=range(number_of_qubits))
        for weight in weights:
            layer(weight)

        return [qml.expval(qml.PauliZ(wires=i)) for i in range(number_of_qubits)]

from Ui.App import App

if __name__ == "__main__":
    app = App()
    app.mainloop()
from __future__ import unicode_literals
```

```
from math import sqrt
import random
from random import randint as rand

def gcd(a, b):
    if b == 0:
        return a
    else:
        return gcd(b, a % b)

def mod_inverse(a, m):
    for x in range(1, m):
        if (a * x) % m == 1:
            return x
    return -1

def isprime(n):
    if n < 2:
        return False
    elif n == 2:
        return True
    else:
        for i in range(1, int(sqrt(n)) + 1):
            if n % i == 0:
                return False
    return True

def generate_keypair(keysize):
    p = rand(1, 1000)
    q = rand(1, 1000)
    nMin = 1 << (keysize - 1)
    nMax = (1 << keysize) - 1
    primes = [2]
    start = 1 << (keysize // 2 - 1)
    stop = 1 << (keysize // 2 + 1)
    if start >= stop:
        return []
    for i in range(3, stop + 1, 2):
        for p in primes:
            if i % p == 0:
                break
        else:
            primes.append(i)
```

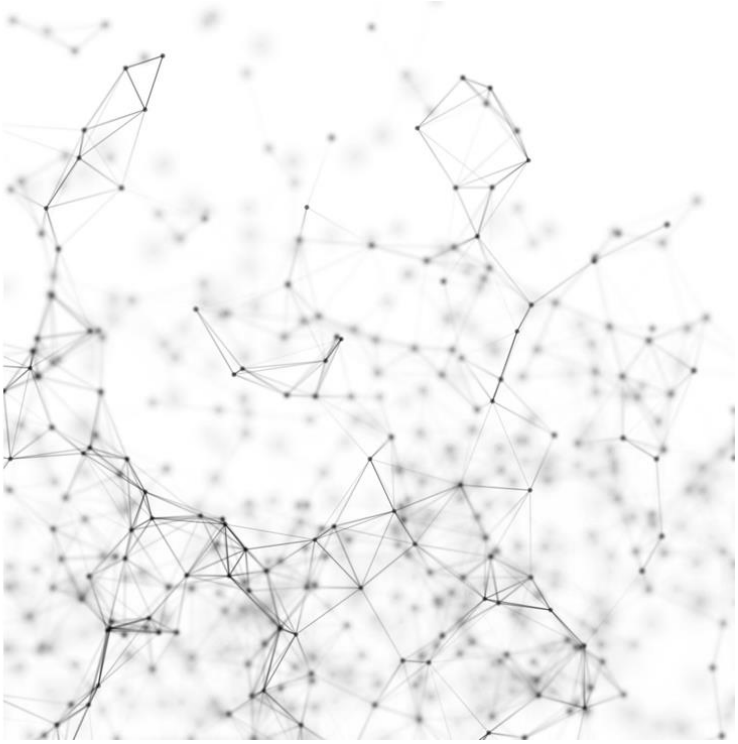
```
    while (primes and primes[0] < start):
        del primes[0]
    while primes:
        p = random.choice(primes)
        primes.remove(p)
        q_values = [q for q in primes if nMin <= p * q <= nMax]
        if q_values:
            q = random.choice(q_values)
            break
    print(p, q)
    n = p * q
    phi = (p - 1) * (q - 1)
    e = random.randrange(1, phi)
    g = gcd(e, phi)
    while True:
        e = random.randrange(1, phi)
        g = gcd(e, phi)
        d = mod_inverse(e, phi)
        if g == 1 and e != d:
            break

    return ((e, n), (d, n))

def encrypt(msg_plaintext, package):
    e, n = package
    msg_ciphertext = [pow(ord(c), e, n) for c in msg_plaintext]
    return ''.join(map(lambda x: str(x), msg_ciphertext)), msg_ciphertext

def decrypt(msg_ciphertext, package):
    d, n = package
    msg_plaintext = [chr(pow(c, d, n)) for c in msg_ciphertext]
    return (''.join(msg_plaintext))
```

Додаток Г
Презентаційний матеріал

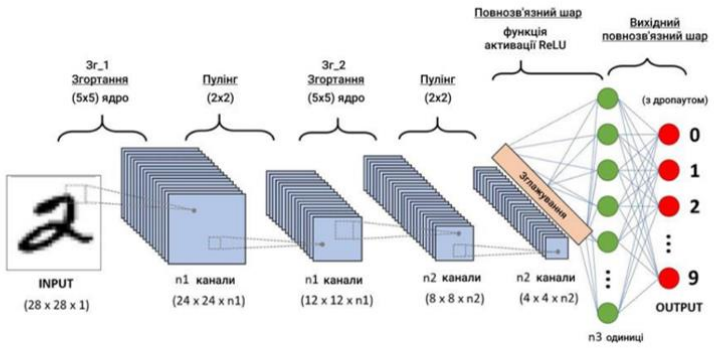
Квантові згорткові нейронні мережі: особливості реалізації у технічних, природничих і соціально-економічних системах

Виконав: студентка групи КН-19-1
Гринько І. М.

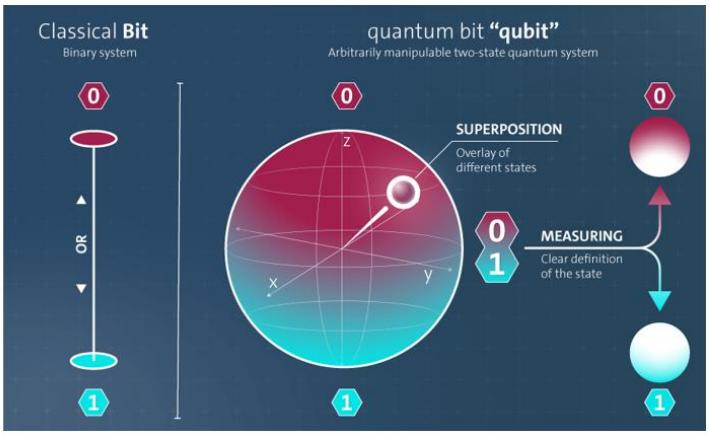
Керівник: д.т.н., проф., завідувач кафедри КН Бармак О.В.



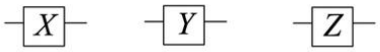
Архітектура згорткової нейронної мережі



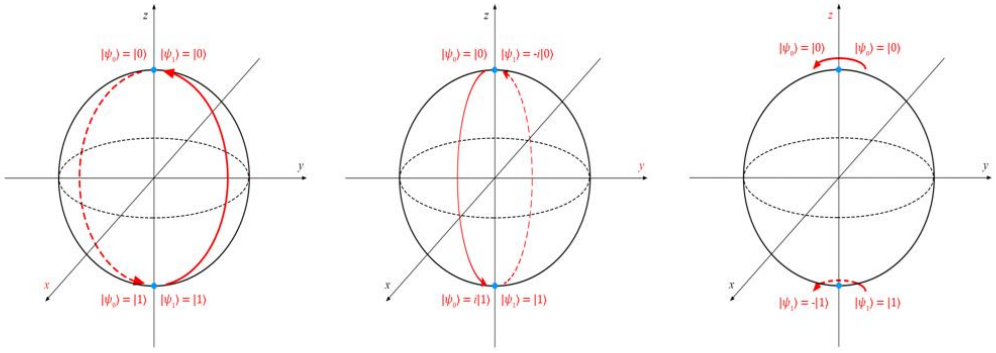
Сфера Блоха



Гейти Паулі



$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$



Гейт Адамара

$$H(|0\rangle) = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle =: |+\rangle$$

$$H(|1\rangle) = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle =: |-\rangle$$

$$H\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{2}\Big(|0\rangle + |1\rangle\Big) + \frac{1}{2}\Big(|0\rangle - |1\rangle\Big) = |0\rangle$$

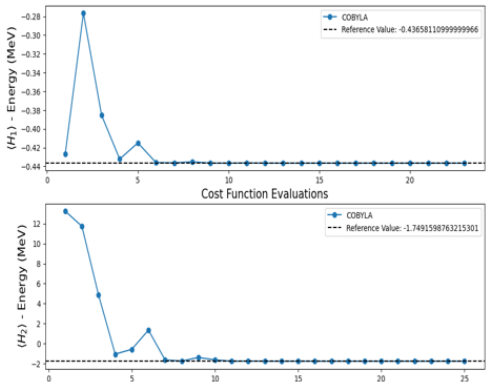
$$H\left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{2}\Big(|0\rangle + |1\rangle\Big) - \frac{1}{2}\Big(|0\rangle - |1\rangle\Big) = |1\rangle$$

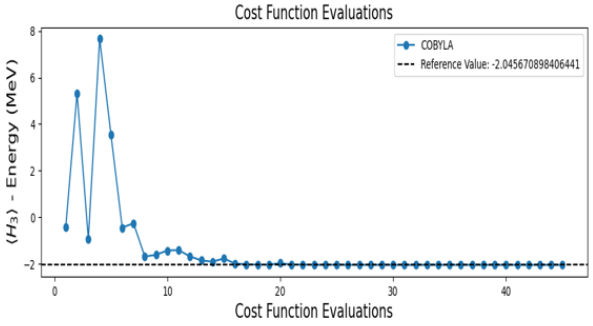
Перехід від кубіта до біта

$$H|+\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1\\ 1\end{pmatrix} = \begin{pmatrix}1\\ 0\end{pmatrix} = |0\rangle\,,$$

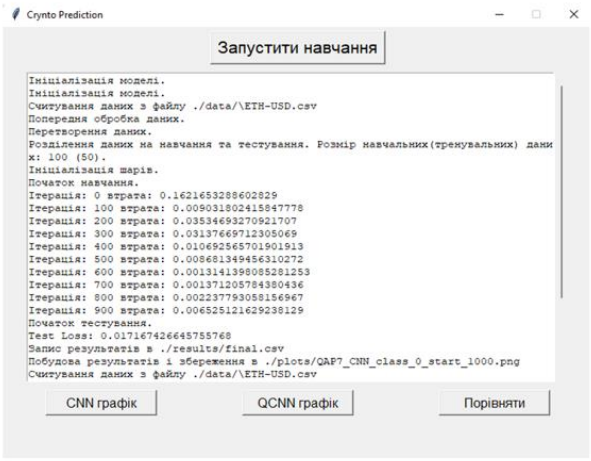
$$H|-\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1\\ -1\end{pmatrix} = \begin{pmatrix}0\\ 1\end{pmatrix} = |1\rangle,$$

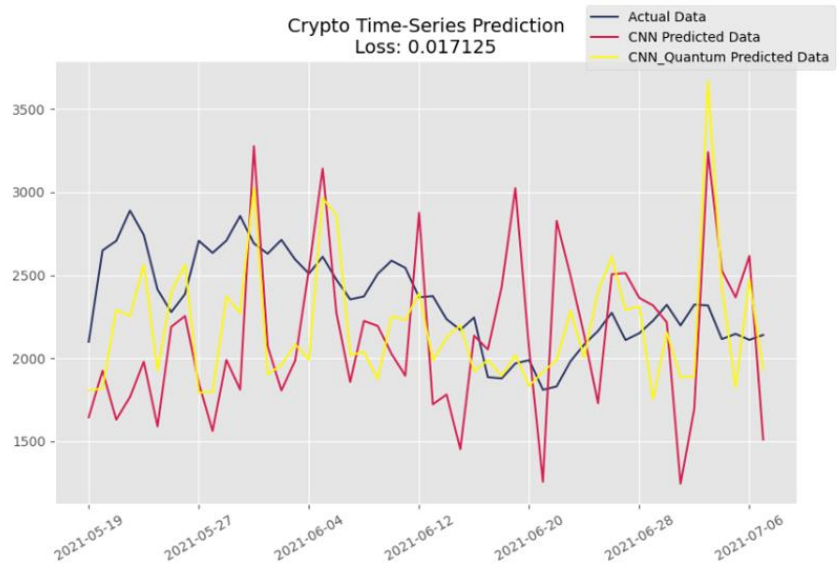
Результати роботи першої програми



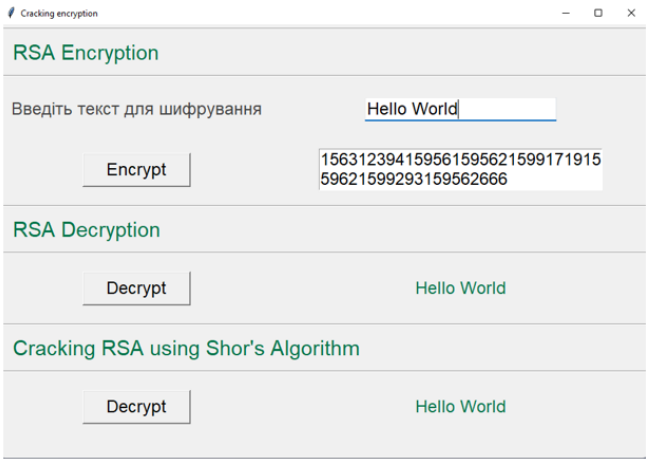


Результати роботи другої програми





Результат роботи третьої програми



Tue May 30 11:59:16 EEST 2023, Петровський Сергій Степанович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 11%**

	Документ		Сумарний збіг по Базі Даних	
ID: 114293 Назва: КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА Додано в БД: 2023-05-30 Автора: І.М. Гринько Керівники: О.В.Бармак Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	76091	1156	1767 (2%)	27 (2%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

UNICHECK

Ім'я користувача:
Кафедра КН

ID перевірки:
1015318500

Дата перевірки:
30.05.2023 12:53:58 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
30.05.2023 12:58:33 EEST

ID користувача:
100005671

Назва документа: **КН-19-1 Гринько**

Кількість сторінок: **67** Кількість слів: **12250** Кількість символів: **90344** Розмір файлу: **1.59 MB** ID файлу: **1014989043**

4.33%
Схожість

Найбільша схожість: **2.06%** з джерелом з Бібліотеки (ID файлу: **1014939129**)

3% Джерела з Інтернету	324	Сторінка 69
2.87% Джерела з Бібліотеки	120	Сторінка 70

0.02% Цитат

Цитати	1	Сторінка 71

Не знайдено жодних посилань

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	10

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ КАФЕДРИ КОМП'ЮТЕРНИХ НАУК
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:
Назва: *Квантові згорткові нейронні мережі: особливості реалізації у технічних, природничих і соціально-економічних системах*

Автор: *студентка групи КН-19-1 Гринько Ірина Миколаївна*
Спеціальність: *122 – Комп'ютерні науки*
Освітня програма: *освітньо-професійна*
Науковий керівник: *д.т.н., проф. Бармак О.В.*
Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	***відповідає***
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі Гринько І.М., не є плагіатом, оскільки: запозичення розміщені в розділі огляду існуючих підходів, не описують безпосередньо авторську роботу і не стосуються її результатів; усі запозичення фрагментарні; до запозичень входять фрагменти поширених конструкцій мови та загальновідомі терміни та скорочення.

Обсяг запозичень, визначений системами виявлення збігів/ ідентичності/схожості, складає:

- за системою Anti-Plagiarism: 1%;

- за системою Unicheck: 4,33%.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ідентичності/ схожості, складає 1% і 4,33% відповідно, що, з урахуванням наведених обґрунтувань, відповідає характеру дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ *Олександр БАРМАК*

Гарант ОП _____ *Олександр МАЗУРЕЦЬ*

Завідувач кафедри КН _____ *Олександр БАРМАК*

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

ВІДГУК НАУКОВОГО КЕРІВНИКА

на кваліфікаційну роботу бакалавра

студента *гр. КН-19-1 Гринько Ірини Миколаївни*

за темою *Квантові згорткові нейронні мережі: особливості реалізації у технічних, природничих і соціально-економічних системах*

1. Актуальність теми

Актуальним завданням, яке потребує аналізу і досліджується у даній роботі, є застосування квантових згорткових нейронних мереж у технічних, природничих і соціально-економічних системах. Квантові згорткові нейронні мережі є новим підходом до обробки інформації, який базується на принципах квантової механіки та штучного інтелекту. Розробка такого метода є актуальною задачею комп'ютерних наук.

2. Відповідність роботи предметній області Стандарту спеціальності 122 Комп'ютерні науки

За стандартом, а саме описом предметної області, об'єктами вивчення та діяльності є математичні, інформаційні, імітаційні моделі реальних явищ, об'єктів, систем і процесів та методи і технології отримання, зберігання, обробки, передачі та використання інформації. Метою роботи саме є розробка квантових згорткових нейронних мереж у технічних, природничих і соціально-економічних системах. При вирішенні поставленої задачі використано математичні моделі, методи та алгоритми розв'язання теоретичних і прикладних задач, що виникають при розробці інформаційних технологій. Тому результати виконання кваліфікаційної роботи бакалавра відповідають стандарту бакалавра спеціальності 122 – Комп'ютерні науки.

3. Професійні та особистісні якості бакалавра

При роботі над кваліфікаційною роботою бакалавра Гринько Ірина Миколаївна проявила себе кваліфікованим фахівцем та дисциплінованим студентом, вчасно виконуючи поставлені етапи дослідження. Як в процесі написання пояснювальної записки, так і при розробці прикладного програмного забезпечення проявив достатні для одержання успішного результату компетентності та результати навчання. Опанувала професійні скіли за напрямком «Комп'ютерні науки» та достатньо значний софт скіл .

4. Ступінь самостійності під час виконання кваліфікаційної роботи

Одержані в роботі результати є наслідком особистої діяльності студента, який самостійно виконував всі поставлені задачі.

5. Ступінь оволодіння методами дослідження

При реалізації кваліфікаційної роботи показав достатній рівень компетентностей та володіння необхідними інструментами та обладнанням, методами, методиками та технологіями предметної області комп'ютерних наук.

6. Повнота та якість розкриття теми роботи

Тема роботи в повній мірі обґрунтована й розкрита, проведено аналіз актуальності та відомих досліджень в межах обраної теми, поставлені завдання, які у роботі виконані, та розроблено програмне забезпечення для валідації та верифікації запропонованого метода.

7. Логічність, послідовність, аргументованість, літературна грамотність викладення матеріалу

Структура роботи та послідовність викладення логічні та відповідають поставленій меті. Викладення матеріалу послідовне, аргументоване, літературно грамотне.

8. Можливість практичного застосування кваліфікаційної роботи бакалавра, окремих її частин

Розроблений у роботі метод та його програмна реалізація може бути використана в кібербезпеці для покращення захисту мереж від кібератак в технічній сфері, в соціально-економічній сфері для прогнозування та аналізу валютного ринку та ринку критовалют та в природничій сфері для аналізу стану молекул та атомів

9. Висновок про можливість допуску кваліфікаційної роботи бакалавра до захисту, на яку оцінку заслуговує робота

Враховуючи високий рівень виконання та забезпечення усіх необхідних вимог, робота може бути допущена до захисту. Рекомендована оцінка «_____».

Керівник _____ *д.т.н., проф. зав. каф. КН Олександр БАРМАК*

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

РЕЦЕНЗІЯ

на кваліфікаційну роботу бакалавра

студента *гр. КН-19-1 Гринько Ірини Миколаївни*

за темою *Квантові згорткові нейронні мережі: особливості реалізації у технічних, природничих і соціально-економічних системах*

1. Актуальність обраної теми

Актуальним завданням, яке потребує аналізу і досліджується у даній роботі, є застосування квантових згорткових нейронних мереж у технічних, природничих і соціально-економічних системах. Квантові згорткові нейронні мережі є новим підходом до обробки інформації, який базується на принципах квантової механіки та штучного інтелекту. Розробка такого метода є актуальною задачею комп'ютерних наук.

2. Повнота розкриття мети та завдань роботи

Тема роботи в повній мірі обґрунтована й розкрита, проведено аналіз актуальності та відомих досліджень в межах обраної теми, поставлені завдання, які у роботі виконані, та розроблено програмне забезпечення для валідації та верифікації запропонованого метода.

3. Зміст кожного розділу роботи

В технічних системах досліджено можливість застосування квантових згорткових нейронних мереж для розв'язання складних задач, таких як спроби ввзлому криптографічних ключів та криптографічного шифрування. У природничих системах проведено дослідження використання квантових згорткових нейронних мереж для моделювання та прогнозування складних природних процесів. У соціально-економічних системах досліджено можливості використання квантових згорткових нейронних мереж для аналізу соціальних мереж, прогнозування фінансових ринків та криптовалют.

4. Оцінка розробленої інформаційної системи, її практична цінність

Розроблений у роботі метод та його програмна реалізація може бути використана в кібербезпеці для покращення захисту мереж від кібератак в технічній сфері, в соціально-економічній сфері для прогнозування та аналізу валютного ринку та ринку критовалют та в природничій сфері для аналізу стану молекул та атомів.

5. Якість оформлення кваліфікаційної роботи бакалавра

Структура роботи та послідовність викладення логічні та відповідають поставленій меті. Викладення матеріалу послідовне, аргументоване, літературно грамотне.

6. Недоліки кваліфікаційної роботи бакалавра

Недоліків в даній кваліфікаційній роботі немає

7. Загальний висновок (допускається чи не допускається до захисту), та оцінка на яку заслуговує кваліфікаційна робота.

Враховуючи рівень виконання та забезпечення усіх необхідних вимог, робота може бути допущена до захисту. Рекомендована оцінка «відмінно».

Рецензент _____ доц., к.ф.-м.н. Наталія Дрешпак