Львівський національний університет імені Івана Франка
Фізичний факультет
Кафедра теоретичної фізики

УДК 538.9.01

Магістерська робота на тему:

Контрольоване приготування квантових станів на основі класичних даних

Виконав студент II курсу
спеціальності 104 Фізика та астрономія
групи ФЗФМ-21с Дмитро Явір
Керівники: канд. фіз.-мат. наук
Микола Максименко,
д-р фіз. мат. наук, проф. Христина Гнатенко
Рецензент: доц. Олег Бовгира

м. Львів – 2023р.

Зміст

Розділ 1. Історія квантової механіки та квантових комп'ютерів

Відкриттям шляху до розвитку квантової механіки можна назвати кінець 19 – початок 20 століття, коли німецький фізик Макс Планк запропонував модель теплового випромінювання [1], згідно з якою енергія може передаватися у вигляді дискретної величини. Це стало новою теорією фізики, що описує поведінку мікроскопічних частинок, таких як елементарні частинки та атоми. У 1905 році Альберт Ейнштейн зробив внесок у розвиток квантової механіки, запропонувавши теорію фотоелектричного ефекту, яка стверджувала, що світло можна розглядати як потік квантів енергії, що взаємодіють з електронами в речовині. А в 1935 році разом з Бором і Подольським Ейнштейн опублікував статтю, в якій вони висунули ідею про те, що квантові системи можуть бути з'єднані за допомогою нерозривного каналу, назва цієї ідеї «ЕПР»-парадокс [2]. Цей парадокс викликав дискусію про те, чи може квантова механіка описати всі аспекти реального світу. Ейнштейн також брав участь у створенні фундаментального принципу квантової механіки, так званого принципу невизначеності Гейзенберга.

З розвитком квантової механіки з'явилися такі завдання, які не під силу навіть найпотужнішим класичним комп'ютерам, про це вперше Р. Фейнман розповів на першій конференції з обчислювальної фізики, зазначив, що неможливо ефективно моделювати еволюцію квантової системи на класичному комп'ютері. Пізніше з'явилися ідеї створення квантових комп'ютерів, але на той час не вистачало знань і ресурсів для створення таких машин. Тому основний розвиток квантових комп’ютерів розпочався у 1980-х роках, коли Річард Фейнман запропонував ідею моделювання фізичних процесів за допомогою квантових систем [3]. У 1985 році була створена перша машина з квантовим алгоритмом, яка була використана для моделювання квантового комп’ютера. У 1994 році Пітер Шор запропонував алгоритм розкладання чисел, який можна було б використовувати для розшифровки закодованих повідомлень

[4]. Це був перший приклад того, що квантові комп’ютери можуть бути корисними для обчислень. З тих пір багато компаній і дослідницькі лабораторії не припиняли роботу над вдосконаленням квантових комп'ютерів.

Квантова механіка відкрила нам шлях до створення квантових комп’ютерів, які, у свою чергу, вимагають якісних і правильних алгоритмів для максимального використання їх потужностей. Поява квантових комп'ютерів дала початок розвитку ідеї квантового машинного навчання. У 2000-х роках квантові алгоритми були запропоновані для проблем машинного навчання, наприклад, алгоритм Гровера для проблем кластеризації. Дослідження Сета Ллойда 2013 року [5] широко визнано як перше, в якому запропоновано використовувати квантові явища для вдосконалення методів машинного навчання. Відтоді зростає кількість досліджень, присвячених дослідженню використання квантових обчислень для машинного навчання, що призвело до створення квантових алгоритмів для кластеризації, класифікації, оптимізації та інших завдань машинного навчання.

Звичайно, квантові комп'ютери набагато потужніші за класичні, але на даний момент ми живемо в епоху короткострокових квантових комп'ютерів середнього масштабу. Ці комп’ютери шумні і не мають великої кількості кубітів. Такі комп'ютери не здатні моделювати і виконувати надто великі квантові завдання (є багато нюансів і проблем, про які ми поговоримо пізніше). Однак є сфери, де квантові обчислення можуть виявитися корисними, працюючи з поточними квантовими машинами.

Розділ 2. Квантове машинне навчання

Обмеження в розвитку штучного інтелекту залежить від обчислювальної потужності комп'ютерів. На даний момент можливості сучасного глибинного навчання з використанням нейронних мереж можливі завдяки паралельним кластерам GPU. У квантовому машинному навчанні використовується новий тип обчислювальних даних, який називається квантовими комп'ютерами, що дозволяє розширити типи доступного обладнання для машинного навчання.

Квантові комп'ютери можна використовувати як нейронні мережі, кодуючи параметри в квантову схему та виконуючи певні вимірювання.

Також однією з переваг квантового машинного навчання перед класичним є використання набагато більшої обчислювальної потужності, яку нам вдається використовувати за допомогою квантового комп'ютера, а отже, набагато швидша обробка даних, що дозволить нам прискорити та покращити розробку моделей машинного навчання, нейронних мереж та інших форм штучного навчання.

Квантові комп'ютери можна використовувати як нейронні мережі, кодуючи параметри в квантову схему та виконуючи певні вимірювання.

2.1 Квантові класифікатори

Існують квантові класифікатори, які є навчальними квантовими схемами, які використовуються як моделі в квантовому машинному навчанні. Карта квантових ознак використовується для вбудовування класичної інформації в квантові стани. Квантове вбудовування представляється як відображення класичних точок даних x у квантовий стан |x>. Квантова карта властивостей виконується квантовою схемою $\Phi(x,\theta)$, яка, у свою чергу, з'єднує фізичні параметри для підготовки нашого квантового стану $|x\rangle=\Phi(x,\theta)|0...0\rangle$. Параметр θ є фізичним параметром, який можна використовувати для оптимізації квантової схеми. Існує досить багато типів вбудовування класичних даних у квантові

стани, таких як базове кодування, амплітудне кодування та кутове кодування. Кожне з цих кодувань має свої переваги та недоліки, наприклад, амплітудне кодування обмежує кількість станів, які можуть бути представлені (для двокубітної схеми кількість можливих станів обмежена чотирма). Тому амплітудне кодування не є хорошою ідеєю для використання в нейронних мережах через складність побудови нейронних мереж із більшою кількістю шарів і нейронів. Кутове кодування є найосновнішою формою кодування класичних даних. Це кодування також відоме як кодування продукту Tensor і вимагає n кубітів для представлення n-вимірних даних.

Крім того, кутове кодування вимагає одного оберту на одному кубіті, тому воно дуже дешеве для підготовки та корисне для обробки даних у квантових нейронних мережах. Одна з головних переваг цього кодування полягає в тому, що воно має значну перевагу з точки зору ефективності, оскільки вимагає лише постійної кількості паралельних операцій незалежно від кількості значень даних, які потрібно закодувати. Однак, з точки зору кубіта, це не є оптимальним, оскільки для кожного компонента вхідного вектора потрібен один кубіт [24].

2.2 Квантові карти ознак

По суті, як згадувалося раніше, карти квантових функцій є фундаментальним будівельним блоком алгоритмів квантового машинного навчання. Карти квантових функцій дозволяють маніпулювати класичними даними в квантових алгоритмах.

Двостороннє заплутування є потужним джерелом обробки квантової інформації, а також відіграє важливу роль у картах квантових ознак. Зокрема, двостороннє заплутування можна вміло використовувати для кодування класичних даних у методі, який набагато ефективніше обробляється квантовими алгоритмами, ніж класичне кодування даних. Двостороннє заплутування можна використовувати для підвищення виразності карт квантових ознак. Ми кодуємо наші класичні дані, використовуючи нелокальні кореляції між квантовими

підсистемами.

Можна закодувати дані як стан продукту двох або більше квантових підсистем, де сплутаність між підсистемами кодує відповідні характеристики даних.

На практиці існує декілька методів реалізації двосторонньої заплутаності в квантових картах ознак, таких як карта ознак заплутаності та карта ознак тензорного продукту. Ці методи дозволяють ефективно кодувати класичні дані в квантові стани, які можна використовувати для підвищення продуктивності квантових алгоритмів.

Основною метою квантової інформації є оцінка властивостей невідомих квантових станів. Це можна використовувати для перевірки пристроїв на готовність до певного цільового стану або для виявлення заплутаних невідомих систем. Метод квантової томографії може повністю охарактеризувати будь-який невідомий квантовий стан, але ця процедура вимагає точності в очікуваних значеннях спостережуваних, які експоненціально зростають із кількістю кубітів. Нещодавня стаття Хуанга та інших пропонує класичну тіньову апроксимацію [28] як потенційний обхідний шлях для цих проблем масштабування. Цей протокол є ефективним методом для створення класичного тіньового представлення невідомого квантового стану. Використання класичної тіні дозволяє оцінити різні властивості, такі як точність квантового стану, очікувані значення гамільтоніанів, рівень заплутаності та двоточкові корелятори.

2.3 **Різноманітність квантових алгоритмів**

В останні роки велика увага приділяється варіаційним квантовим алгоритмам. Варіаційні квантові алгоритми — це алгоритми, які використовують квантовий комп’ютер у поєднанні з класичним комп’ютером для виконання конкретного завдання. Класичні методи машинного навчання використовуються для навчання, вибірки та оптимізації станів шумного квантового обладнання. Існують такі алгоритми, як варіаційний квантовий власний розв'язувач (VQE)[6]

або квантовий алгоритм наближеної оптимізації (QAOA)[7]. Ці алгоритми демонструють потенціал для різноманітних короткострокових застосувань, досить надійні проти певних помилок когерентності та можуть мінімізувати ефекти декогерентності шляхом дослідження нетрадиційних послідовностей воріт. Великий інтерес представляють квантові нейронні мережі (QNN), в яких квантові вхідні стани перетворюються на вихідні за допомогою параметризованої квантової схеми (PQC). Вихідні стани потім піддаються ряду вимірювань, які разом називають функцією вартості, і результати вимірювань використовуються для оптимізації схеми.

Існують також інші алгоритми для вирішення різних типів завдань на квантовому комп'ютері.

Існує алгоритм **гібридних квантових автокодерів** (HQA) [8]. Варіант алгоритму квантової оцінки амплітуди (QAE), який називається гібридним квантовим автокодером (HQA). Ця модель складається з двох частин: класичного машинного навчання з використанням нейронних мереж (ANN) і квантового машинного навчання з використанням квантових нейронних мереж (QNN) на основі параметризованих квантових схем (PQC). Конструкція моделі така: це комбінація кодера та декодера, кодер переносить стан із гільбертового простору в підмножину реального векторного простору V розміру v=dim(V) і декодера, який виконує зворотну операцію.

Кодер — це параметризована квантова схема за допомогою вектора. Схема отримує деякий стан $|\psi_{in}>$, потім застосовує унітарний U_1 до вхідного стану з допоміжними кубітами.

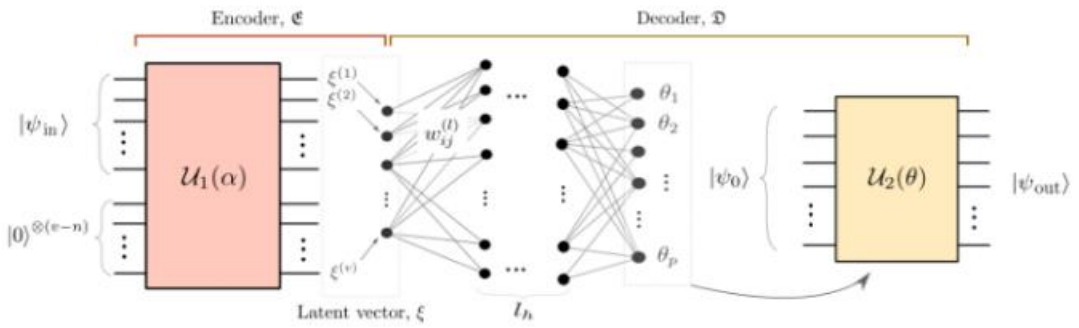

Рисунок 1 - Гібридний квантовий автокодер (HQA) [8]

Інший алгоритм **квантовий мультикласовий класифікатор** (QMCC) [9], який є варіаційною схемою для класифікації набору даних. Схема виконується наступним чином, чотири кубіти знаходяться в стані |0>, потім ми застосовуємо вентиль Адамара до всіх чотирьох кубітів, потім застосовується унітарна операція з унітарною квадратною матрицею, створеною для підготовки стану, таким чином класичні дані кодуються в кубіти .

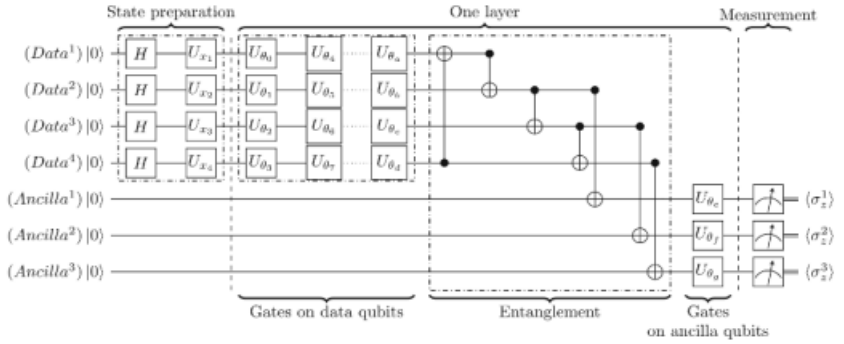

Рисунок 2 - Квантова варіаційна модель схеми [9]

Варіаційна схема розроблена з використанням кількох шарів поворотних вентилів і допоміжних кубітів із параметризованими вагами, які можна регулювати за допомогою методів оптимізації. Нарешті, допоміжні кубіти вимірюються, а отримані кубіти обробляються для отримання мітки класу.

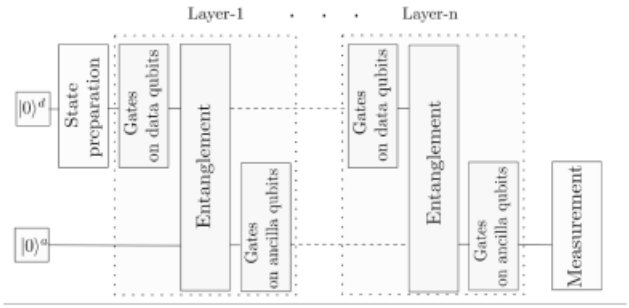

Рисунок 3 - Архітектура квантової варіаційної схеми [9]

Ще один алгоритм під назвою **метод опорних векторів з оцінювачем**

квантового ядра (QSVM-Kernel) [10]. Як відомо, так звана ядерна функція дозволяє неявно працювати в просторі вказівників без обчислення координат у цьому просторі, а просто шляхом обчислення скалярних продуктів відображення пар даних. Цей алгоритм використовує простір функцій як простір функцій для обчислення вхідних даних ядра. Він нелінійно відображає класичні дані x у квантовий стан, застосовуючи квантову схему $\Gamma_{\Phi(x)}$ початкового стану |0>. Ця квантова схема створює 2^N - вимірний простір ознак, який важко оцінити класичним способом. Схема має два повторювані шари.

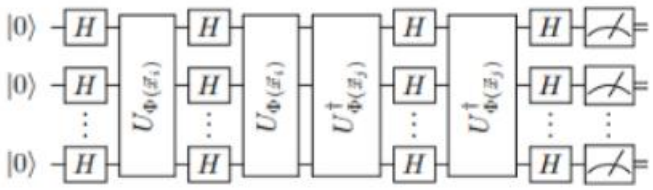

Рисунок 4 - Квантова схема методу опорних векторів [10]

Під час фази навчання записи ядра оцінюються на предмет навчальних даних для встановлення гіперплощини поділу. Пізніше, на етапі тестування, вхідні дані ядра оцінюються як на нових, так і на навчальних даних, щоб класифікувати нові дані на основі роздільної гіперплощини. Квантові комп'ютери оцінюють вхідні дані ядра, тоді як класичні комп'ютери оптимізують гіперплощину розділення та класифікують дані, як це робиться в класичній методі опорних векторів SVM.

Звичайно, ці алгоритми потребують удосконалення, щоб надійно запускати великі проблеми на теперішніх квантових комп'ютерах.

Квантові нейронні мережі (QNN) пропонують простий підхід, але їх реалізація може бути досить складною.

Проблема полягає в тому, щоб зв'язати архітектуру моделі з її здатністю до навчання. На даний момент ми не можемо зрозуміти, наскільки добре тренується наша модель, тому що експоненціальна розмірність гільбертового простору та складність оцінки градієнта роблять вимірювання функції втрат

непридатними для гібридних квантово-класичних алгоритмів, які працюють на кількох кубітах.

2.4 Градієнт

Один з важливих термінів у цій роботі є градієнт функції. Векторне поле, яке вказує в напрямку найвищої швидкості зростання функції, називається градієнтом скалярнозначної диференційованої функції f багатьох змінних. Точніше, градієнт f(p) — це вектор, який у кожному місці p рухається в напрямку найвищої швидкості зростання f у цій точці, величина якого дорівнює цій швидкості збільшення.

Існує дотична площина в p, така що вектор градієнта ортогональний до дотичної площини, якщо градієнт f відмінний від нуля в точці p. Іншими словами, в точці p вектор градієнта виходить безпосередньо з поверхні, визначеної f. Це вказує на те, що найрізкіше збільшення f від точки p буде результатом руху в напрямку вектора градієнта.

З іншого боку, не існує напрямку, у якому f зростає, якщо градієнт f дорівнює 0 у точці p. Ці положення називаються стаціонарними точками і можуть бути сідловими точками f, максимумів або мінімумів. [11]. Градієнт скалярно-значної диференційованої функції f із декількома змінними визначається як векторне поле, позначене як f або grad(f), і визначається за наведеною нижче формулою:

$$\nabla f = (\partial f/\partial x_1, \partial f/\partial x_2, ..., \partial f/\partial x_n),$$

де x_1, x_2, ..., x_n — змінні, за якими беруться часткові похідні. Кожна складова вектора градієнта є частковою похідною від f за відповідною змінною. Крім того, градієнт f можна виразити за допомогою оператора del ∇ наступним чином:

$$\nabla f = \nabla \cdot f = (\partial/\partial x1, \partial/\partial x2, ..., \partial/\partial xn)\cdot f,$$

де крапкова нотація представляє скалярний добуток між оператором del і скалярною функцією f.

2.5. Проблема безплідних плато

Безплідні плато – це області в просторі варіаційних параметрів, де дисперсія градієнта функції втрат експоненціально зникає в кількості кубітів [18]. На даний момент вчені активно працюють над проблемою Безплідних плато і шукають нові методи подолання цієї проблеми.

Проблема безплідних плато є однією з ключових проблем квантового машинного навчання. Безплідні плато — це явище в квантових обчисленнях, де величини градієнтів експоненціально пригнічуються зі зростанням розміру проблеми. Простими словами, ми не можемо знайти глобальні мінімуми, оскільки функція втрат бореться з ВР. Коли розмір квантових нейронних мереж збільшується, додавання кубітів не покращує точність моделі. Це може ускладнити оптимізацію квантових схем, особливо для великої кількості кубітів. Заплутаність — фундаментальна концепція в квантовій механіці, яка описує кореляції між квантовими системами. Багато мір заплутаності спочатку визначаються для чистих станів дводольного гільбертового простору, а потім поширюються на змішані стани ентропії заплутаності. Нещодавні дослідження показали, що заплутаність може бути використана для пом’якшення ефектів безплідних плато в квантових ланцюгах [19][20]. Використовуючи заплутаність як керівний принцип оптимізації, дослідники запропонували та продемонстрували низку методів пом’якшення безплідних плато, включаючи початкову підготовку до стану, оптимізацію на основі вимірювань та оптимізацію на основі заплутаності. Ці методи використовують переваги кореляції між кубітами, щоб керувати процесом оптимізації та уникнути застрягання на безплідних плато.

Зміна архітектури нейронної мережі може вплинути на виникнення безплідних плато. Наприклад, останні дослідження показали, що використання класичних глибоких нейронних мереж може допомогти уникнути безплідних плато у варіаційних квантових алгоритмах [21]. Це тому, що класичні нейронні

мережі мають більш структуровану архітектуру, ніж квантові нейронні мережі, що може полегшити оптимізацію мережі за допомогою методів на основі градієнта. Зміна архітектури нейронної мережі може вплинути на появу безплідних плато[22][23]. Використання класичних глибоких нейронних мереж і функцій активації, які менш схильні до насичення, може допомогти уникнути безплідних плато у варіаційних квантових алгоритмах.

Підводячи підсумок, безплідні плато є проблемою для квантових обчислень, яка може ускладнити оптимізацію квантових схем, особливо для великої кількості кубітів. Було показано, що сплутаність, яка описує кореляції між квантовими системами, є корисним інструментом для пом'якшення ефектів безплідності.

2.6 Заплутаність у квантовому ланцюзі

Ще одна і насправді найважливіша перевага перед квантовим машинним навчанням — це заплутаність у квантових схемах. Квантове машинне навчання є потужним інструментом для роботи з багатьма складними завданнями, тобто вчені стикаються з кількома проблемами при створенні моделей. Існуючі моделі квантового машинного навчання використовують глибокі квантові схеми, які страждають від великого накопичення помилок вентиля та декогеренції. Як я вже згадував раніше, квантові обчислення викликають шум, і результат обчислень може містити багато помилок, тому існують алгоритми, які допомагають зменшити кількість шуму.

Отже, кілька слів про один із найважливіших механізмів отримання квантової переваги – заплутаність. Це поняття є одним з основних у квантовій механіці. Заплутаність - це стан системи, коли дві пари частинок (атомів, протонів) знаходяться в змішаному стані і неможливо описати їх стани окремо один від одного. Якщо ми змінюємо стан однієї з частинок, то заплутана з нею частинка також змінює свій стан. Entanglement дозволяє паралельно виконувати обчислення на кількох кубітах. В даний час існує досить багато досліджень і

статей на тему заплутаності, а в 2022 році фізик Джон Клаузер і його колега Ален Аспект розробили тест, який довів заплутаність фотонів [12]. Якщо говорити про використання заплутаності в квантовому машинному навчанні, то очевидно, що заплутаність прискорює виконання складних завдань і обробку інформації. Наприклад, у квантовій кластеризації заплутаність допомагає збільшити швидкість і точність кластеризації.

Як було сказано раніше, заплутаність є надзвичайно важливим інструментом у квантовій механіці і, звичайно, має свої закони та правила. Щоб кількісно визначити зв'язок між двома підсистемами, вимірюється ентропія зв'язку. Ентропія заплутаності фон Неймана [13][14][15], також відома як ентропія заплутаності, є одним із найпростіших методів вимірювання заплутаності. Ентропія заплутаності зникає лише тоді, коли немає квантової заплутаності між двома частинами (системи), і тоді стан є станом продукту. Щоб обчислити ентропію заплутаності підсистеми A (з об'ємом VA) $|\psi>$, слід відстежити додаткову підсистему B (з об'ємом $V - V_A$, де V — загальний об'єм), щоб отримати змішану матрицю щільності $\rho_A = Tr_{HB} |\psi > <\psi|$. І вихідна ентропія S_A підсистеми A дорівнює

$$S_A = -Tr (\rho_A \ln \rho_A).$$

n-та ентропія Реньї [16][17] визначається як

$$S^{(n)}_A = - \ln [Tr (\rho^n_A)].$$

2.7 Закон площі та об'єму

Закон заплутаності площі стосується поведінки масштабування ентропії заплутаності квантової системи, де ентропія заплутаності масштабується з площею поверхні межі між двома підсистемами. Цей тип заплутаності зазвичай спостерігається в основних станах розривних квантових систем, де ентропія

заплутаності між двома підсистемами пропорційна площі поверхні межі між двома підсистемами. Закон заплутаності площ також спостерігається в одновимірних гратчастих системах. Закон заплутаності площі пов’язаний з квантовою геометрією системи, яка охоплює як кривизну Беррі, так і квантову метрику. Квантова геометрія відіграє ключову роль у багатозонних взаємодіючих електронних системах. Сплутаність за законом площі відрізняється від заплутаності за законом об’єму, де ентропія заплутаності масштабується з обсягом підсистем. Закон об’ємної заплутаності зазвичай спостерігається в критичних або безщілинних квантових системах і типових власних станах квантових багатотільних гамільтоніанів.

2.8 Тензорні мережі

Методи тензорної мережі стають важливими в сучасній квантовій фізиці та інших областях. Вони можуть ефективно апроксимувати певні типи квантових станів, а графічна мова, яка використовується для їх опису, дозволяє легко виражати та міркувати про квантові схеми.

Тензорні мережі — це універсальна математична структура, яка використовується в багатьох галузях, таких як фізика, машинне навчання чи інформатика. Простіше кажучи, тензорна мережа — це певний спосіб представити ваші багатовимірні дані або стани шляхом розбиття цих даних на більш дрібні частини, якими легше керувати та обробити. Основна ідея полягає в тому, щоб розкласти високовимірний об’єкт, такий як квантова хвильова функція або набір даних, на мережу тензорів меншої розмірності. Тензор представляє локальну функцію системи, а мережа цих тензорів кодує глобальні кореляції між функціями.

В даний час існує досить велика кількість тензорних мереж. Найпопулярніші тензорні мережеві моделі включають Матричні стани продукту (MPS) [25], Стани тензорного продукту (TPS), Multi-scale Entanglement Renormalization Ansatz (MERA) [26] і Проекційні стани заплутаної пари (PEPS)

[27] .

Розділ 3. Наша робота

3.1 Проблеми.

У цій роботі загалом досліджується, як різна заплутаність матиме різну можливість навчання наших моделей. Наприклад, певні типи заплутаності можуть покращити здатність моделі класифікувати дані з високою точністю, тоді як інші можуть бути не настільки ефективними для цього завдання. Також те, як навчання моделі залежить від типу введення даних.

Для початку ми почнемо з обчислення заплутаності квантового стану. Як я вже говорив раніше, найпрактичнішим способом обчислення заплутаності квантового стану є обчислення другої ентропії Реньї або ентропії заплутаності.

Заплутаність квантового стану може бути обчислена за допомогою другої ентропії Реньї.

Друга ентропія Реньї є мірою заплутаності квантового стану, яка дає безперервний спектр, параметризований змінною [29]. Друга ентропія Реньї може бути безпосередньо виміряна шляхом інтерференції двох ідентичних і незалежних копій квантового стану на 50%–50% розділювачі променя. Цей метод дозволяє отримати ентропію заплутаності Реньї різних квантових багаточастинкових систем з високою ефективністю та точністю [30][31]. Ентропію заплутаності Реньї можна переписати як

$$S^{(n)}_A = - \ln [Tr (\rho^n_A)].$$

Спостережувана величина збирається з усіх N паралельних процесів і підсумовується, щоб отримати ентропію заплутаності Реньї. Щоб знайти значення сплутаності на кубіт, можна створити матрицю щільності для системи та використати часткові сліди матриці щільності.

Для цієї роботи я використовував міжплатформенну бібліотеку Python з відкритим кодом під назвою Pennylane [32], яка підтримує широкий спектр

завдань у квантових обчисленнях, квантовому машинному навчанні та квантовій хімії. Це дозволяє використовувати існуючі бібліотеки машинного навчання, такі як PyTorch, JAX або TensorFlow, для розробки та запуску квантових алгоритмів на різних квантових комп'ютерах або симуляторах схем. PennyLane має стандартні симулятори пристроїв і підключається до зовнішнього програмного та апаратного забезпечення для запуску квантових схем.

3.2 Трансферне навчання

Також у цій роботі присутній метод трансферного навчання. Трансферне навчання [33][34] — це техніка машинного навчання, яка дозволяє використовувати попередньо навчену модель, розроблену для конкретного завдання, для схожих проблем шляхом простого перенавчання нової моделі для поточних завдань. Мета трансферного навчання полягає в тому, щоб покращити навчання в новому завданні, використовуючи знання з пов'язаного завдання, яке вже було засвоєно. ImageNet, AlexNet і Inception є типовими прикладами моделей, заснованих на Transfer learning.

Традиційні моделі машинного навчання вимагають навчання практично з нуля, що вимагає багато ресурсів і багато попередньо навчених даних для досягнення достатньої точності моделі. Трансферне навчання набагато ефективніше витрачає ресурси та потребує невеликого набору даних, на відміну від традиційної моделі.

Крім того, модель традиційного ML навчається для певної мети і не залежить від попередніх знань, натомість трансферне навчання, яке частково використовує знання з попередньої моделі, таким чином має набагато ширший підхід до навчання та швидше досягає високої продуктивності, оскільки вони мають так званий досвід попереднього навчання.

Було продемонстровано, що в багатьох випадках ефективніше починати з глибокої мережі, яка вже пройшла навчання, а потім лише додавати остаточні рівні для заданої роботи та набору даних, що цікавить [35][36].

Існують різні стратегії трансферного навчання, які можна застосовувати залежно від самого завдання та наявності даних, наприклад: індуктивне трансферне навчання, трансдуктивне трансферне навчання, неконтрольоване трансферне навчання.

При роботі з гібридною системою ми можемо поєднувати різні фізичні властивості мереж:

Таблиця 1 — методи трансферного навчання.

Мережа A	Мережа B	Схема трансферного навчання
Класичне	Класичне	CC - Стандартний класичний метод
Класичне	Квантове	CQ – Гібридна модель
Квантове	Класичне	QC - Гібридна модель
Квантове	Квантове	QQ – Квантова модель

Найпривабливішим навчанням в епоху NISQ є CQ, це навчання дозволяє класичну попередню обробку великих вхідних вибірок за допомогою будь-якої глибокої нейронної мережі. За допомогою варіаційної квантової схеми можна маніпулювати кількома, але досить інформативними функціями. Ця стратегія дуже практична, оскільки вона поєднує силу квантових обчислень із випробуваними класичними методами машинного навчання.

Трансферне навчання широко використовується в комп’ютерному зорі та обробці природної мови.

Кілька слів про Трансферне навчання в контексті Глибокого навчання або Deep learning. Обробка природної мови та розпізнавання зображень вважаються одними з найцікавіших напрямків досліджень. Нейронні мережі трансферного навчання, які складають основу в контексті глибокого навчання, називають глибоким трансферним навчанням. Системи глибокого навчання — це багаторівневі архітектури, які можуть вивчати різні типи функцій на різних рівнях. Функції вищого рівня збираються на зовнішніх рівнях мережі та стають точнішими, коли ви просуваєтеся глибше в мережу.

Трансферне навчання — це постійна тема, яка цікавить науковців машинного навчання, і досі є відкриті проблеми, які потрібно вирішити. Однак це потужна техніка, яка може покращити навчання в нових завданнях і заощадити час і ресурси під час навчання моделі.

3.2.1 Готові моделі, попередньо навчені як екстрактори функцій

Щоб отримати кінцевий результат, ці рівні врешті-решт пов’язуються з кінцевим рівнем, який часто є повністю пов’язаним рівнем у випадку навчання під наглядом.

Це дає змогу використовувати добре відомі попередньо навчені мережі для додаткових завдань, не покладаючись на їхній останній рівень як екстрактор фіксованих функцій.

Хитрість тут полягає в тому, щоб використовувати зважені шари навченої моделі для вилучення ознак, але не змінювати ваги моделі під час навчання зі свіжими даними для нової роботи.

Попередньо навчені моделі можна використовувати для надання широкого представлення візуального світу, оскільки вони були навчені на досить великих і різноманітних наборах даних.

3.2.2 Точне налаштування готових попередньо навчених моделей

Цей метод більш цікавий, оскільки він замінює останній шар, а також вибірково перенавчає деякі попередні шари, а не покладається лише на характеристики, отримані з попередньо навчених моделей.

Глибокі нейронні мережі мають численні настроюванні гіперпараметри та багатошарову архітектуру. Функція перших шарів полягає в записі загальних характеристик, тоді як останні шари більше зосереджені на конкретній роботі. Представлення функцій вищого порядку базової моделі слід налаштувати, щоб зробити їх більш корисними для даної мети. Зберігаючи певні шари моделі

замороженими під час навчання, ми можемо перенавчати інші.

3.2.3 Заморожування проти тонкого налаштування

Перенавчання (або «точне налаштування») ваг верхніх шарів попередньо навченої моделі разом із навчанням класифікатора, є розумною стратегією ще більшого покращення продуктивності моделі.

Це змусить модель, яка вивчила загальні карти ознак із початкового завдання, оновити ваги. Модель зможе використовувати попередні знання в цільовій області та повторно вивчати деякі елементи шляхом тонкого налаштування.

Крім того, замість того, щоб налаштовувати всю модель, потрібно налаштувати декілька верхніх шарів. нижні шари підбирають прості, універсальні властивості, які застосовуються практично до всіх видів даних. Тож здається доцільним заморозити ці шари та перепрофілювати фундаментальну інформацію, отриману під час попереднього навчання. Функції стають більш адаптованими до набору даних, на якому була навчена модель, коли ми піднімаємося вище. Замість заміни загального навчання, тонке налаштування прагне адаптувати ці спеціалізовані функції до нового набору даних.

3.3 Гібридні моделі

У цій роботі також використовуються гібридні моделі, приклади яких я навів трохи вище. Останніми роками ці моделі набули великого розголосу. Гібридні моделі складаються з класичних і квантових елементів, що дозволяє доповнити попередньо навчену класичну нейронну мережу варіаційною квантовою схемою. Така модель дозволяє проводити попередню обробку високовимірних даних за допомогою класичної нейронної мережі та вбудовувати набір функцій у квантовий процесор, що робить цю процедуру досить оптимальною, особливо в епоху квантових комп’ютерів середнього

масштабу.

Не усвідомлюючи цього, більшість із нас, мабуть, певною мірою використовували алгоритми HML. Можливо, ми змішали методи з різних дисциплін або поєднали методи, які вже використовувалися. Перш ніж надсилати наші дані для підходів ML, ми час від часу змінюємо дані за допомогою таких методів, як аналіз головних компонент або базовий лінійний кореляційний аналіз. Деякі експерти автоматизують оптимізацію параметрів поточних методів машинного навчання за допомогою еволюційних алгоритмів. Основою алгоритмів HML є архітектура ML, яка дещо відрізняється від традиційного робочого процесу. Здається, ми сприймаємо алгоритми машинного навчання як належне, тому що часто використовуємо їх просто з полиці, не замислюючись про особливості того, як усе поєднується.

З метою вдосконалення один одного, HML є розвитком робочого потоку ML, який плавно поєднує різні алгоритми, процеси або процедури з пов'язаних або не пов'язаних галузей дослідження чи застосування. Жодне рішення машинного навчання (ML) не може вирішити всі проблеми, так само як жодна капелюх не підходить усім. Хоча деякі методи можуть обробляти шумові дані, вони можуть бути не в змозі обробляти вхідний простір із великою кількістю вимірів. Інші можуть досить добре обробляти багатовимірний простір введення, але мають проблеми з розрідженими даними. Ці обставини забезпечують міцну основу для використання HML для доповнення підходів-кандидатів і використання одного для компенсації недоліків інших.

Існує незліченна кількість способів гібридизації класичних методів машинного навчання, і це можна зробити для кожного з них, щоб створити нові гібридні моделі різними способами. Три з них — архітектурна інтеграція, маніпулювання даними та оптимізація параметрів моделі.

3.3.1 Гібридне машинне навчання на основі архітектурної інтеграції

Щоб створити більш надійний незалежний алгоритм, цей тип гібридного

машинного навчання (HML) плавно поєднує дизайн двох або більше звичайних алгоритмів, повністю або частково. Прикладом, який часто використовується, є адаптивна мережа на основі системи нечіткого виводу . (ANFIS). ANFIS використовується деякий час і зазвичай розглядається як окрема звичайна техніка ML. Фактично, він поєднує концепції штучної нейронної мережі (ANN) і нечіткої логіки. Архітектура ANFIS складається з п'яти рівнів. Перші три походять від нечіткої логіки, а останні два від (ANN). Аніфоуз та ін. надали додаткову технічну інформацію про ANFIS, зокрема про функції кожного рівня [37].

Наївне дерево Байєса, або NBTree, є ще однією ілюстрацією техніки гібридного машинного навчання, заснованої на архітектурній інтеграції. Він поєднує в собі дерево рішень і наївні алгоритми Байєса. Подібно до стандартного дерева рішень, вузли дерева рішень мають одновимірні розбиття, але його листя містять наївні байєсовські класифікатори. (Кохаві 1996). Недоліки першого були суттєво усунені другим. Наївний метод Байєса має надзвичайну ефективність, але, згідно з дослідженням, він погано масштабується з даними більшої розмірності. Дерева рішень, з іншого боку, можуть легко збільшуватися в тій самій ситуації, але схильні до переобладнання. Для використання взаємодоповнюючих властивостей цих двох технологій необхідним став їх гібрид.

3.3.2 Маніпулювання даними на основі гібридного машинного навчання

Цей вид гібридного навчання плавно поєднує класичні підходи машинного навчання з процесами або процедурами маніпулювання даними з метою покращення результатів перших.

Останнім часом широкого поширення набули змішані методи, наприклад процес маніпулювання даними. Вони засновані на відборі функцій, який має на меті доповнити вбудований процес вибору моделі канонічних методів ML.

Зрозуміло, що алгоритм ML може вибрати найкращу модель на основі оптимального набору вхідних характеристик. Дослідження показали, що для підвищення продуктивності та точності традиційних алгоритмів машинного навчання допомагає виконувати цю процедуру за допомогою зовнішнього алгоритму як етапу попередньої обробки (Anifowose та ін. 2014; Sasikala 2016).

3.3.3 Гібридне машинне навчання на основі оптимізації параметрів моделі

Добре відомо, що кожен класичний підхід ML вибирає найкращі параметри налаштування за допомогою певної стратегії оптимізації або пошуку, такої як градієнтний спуск або пошук по сітці. Цей вид гібридного навчання використовує різноманітні передові методи, які базуються на еволюційних алгоритмах, щоб доповнити або, у деяких випадках, замінити вбудований підхід до оптимізації параметрів.

Розділ 4. Час експерименту

Тож почнемо із простої квантової схеми

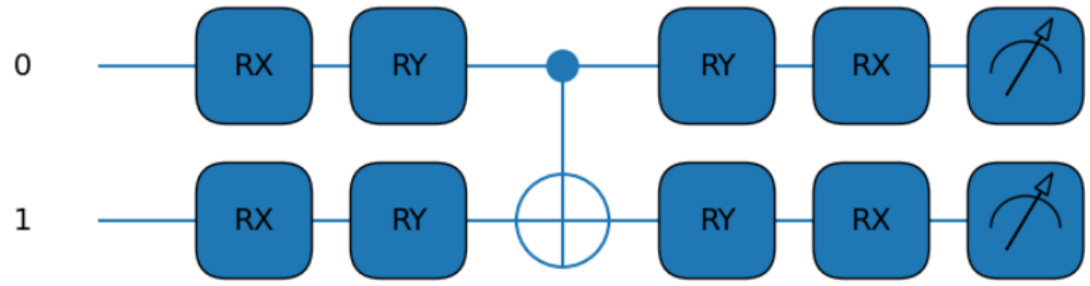

Рисунок 5 — Квантова схема для моделі з пошуку ентропії заплутаності

Тут ми маємо шар воріт RX, потім шар воріт RY, шар воріт CNOT і знову шар воріт RY і рівень воріт RX. Вентилі RX і RY першого рівня мають чотири змінні параметри θ, φ, Ψ і α. Ці параметри регулюються під час навчання, тому в основному ці два шари є нашими вбудованими шарами. Інший набір параметрів, який ви можете побачити на рисунку 5 після воріт CNOT, є випадковим значенням, яке змінюється після кожної епохи. А щоб виміряти схему, ми використовуємо вбудовану функцію Pennylane density_matrix(wires), де wires є номер кубітів, тому ми отримуємо матрицю щільності залежно від того, який кубіт ми встановили.

Функція витрат є

$$C = |S - T|^2$$

яка використовується для циклу навчання моделі, S — друга ентропія Реньі, а T — наша цільова ентропія, яку ми встановлюємо самі. Оптимізатор — це оптимізатор Адама, який також вбудований у бібліотеку Pytorch.

Використовуючи іншу швидкість навчання, інший набір даних для

навчання та тестування, збільшуючи глибину квантової схеми (додаючи більше вентилів RY, RX і CNOT) і додаючи більше кубітів до квантової схеми, мною були отримані наступні резутьтати

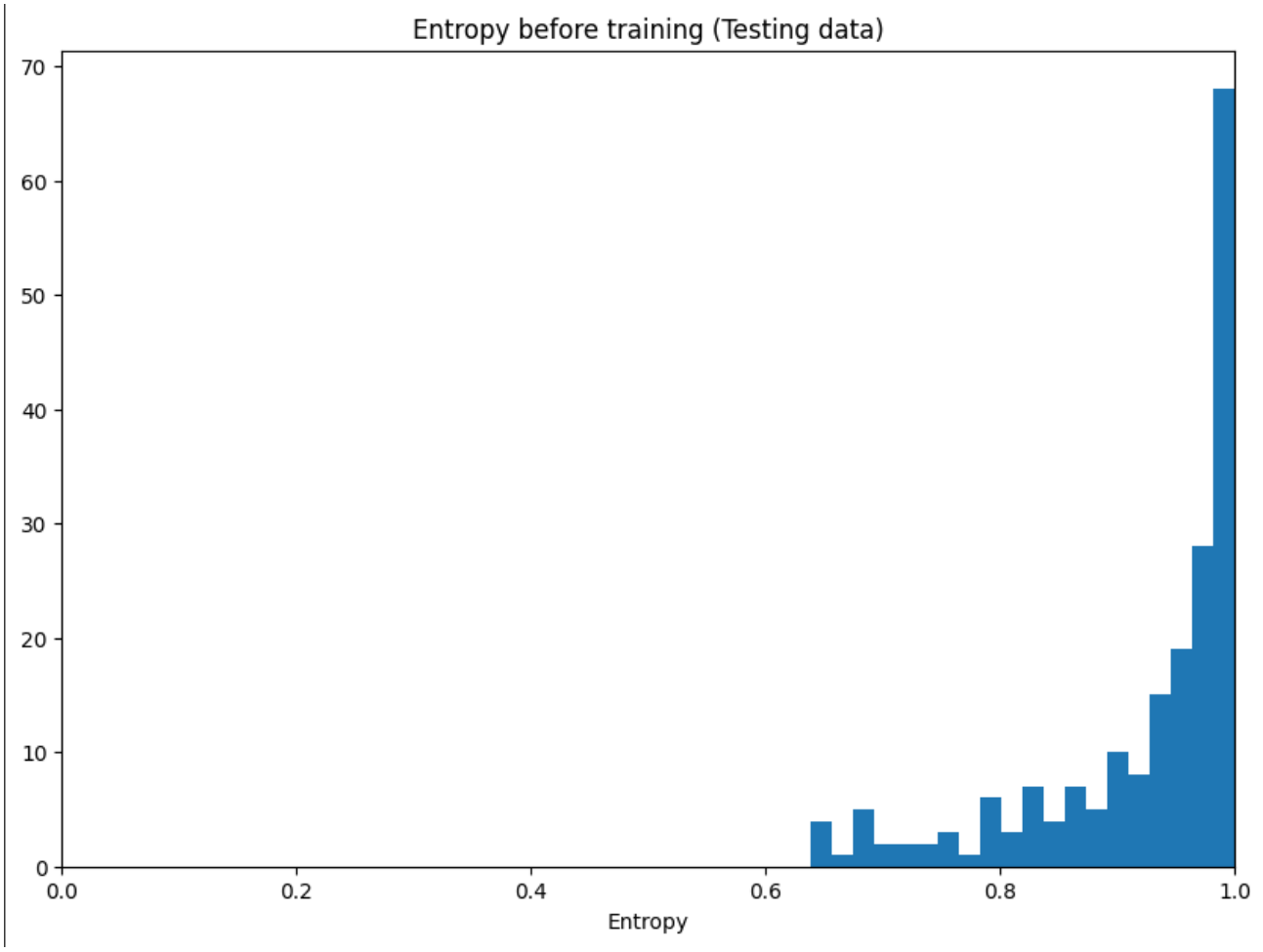

Рисунок 6 — Розподіл ентропії перед тренуванням моделі на тестових даних.

На цій гістограмі ми отримуємо розподіл ентропії для навчальної моделі перед тренуванням моделі, цей розподіл є на тестовому наборі даних. З гістограми видно, що розподіл ентропії між 0.7 і 1.0.

Після тренування функція втрат виглядає так:

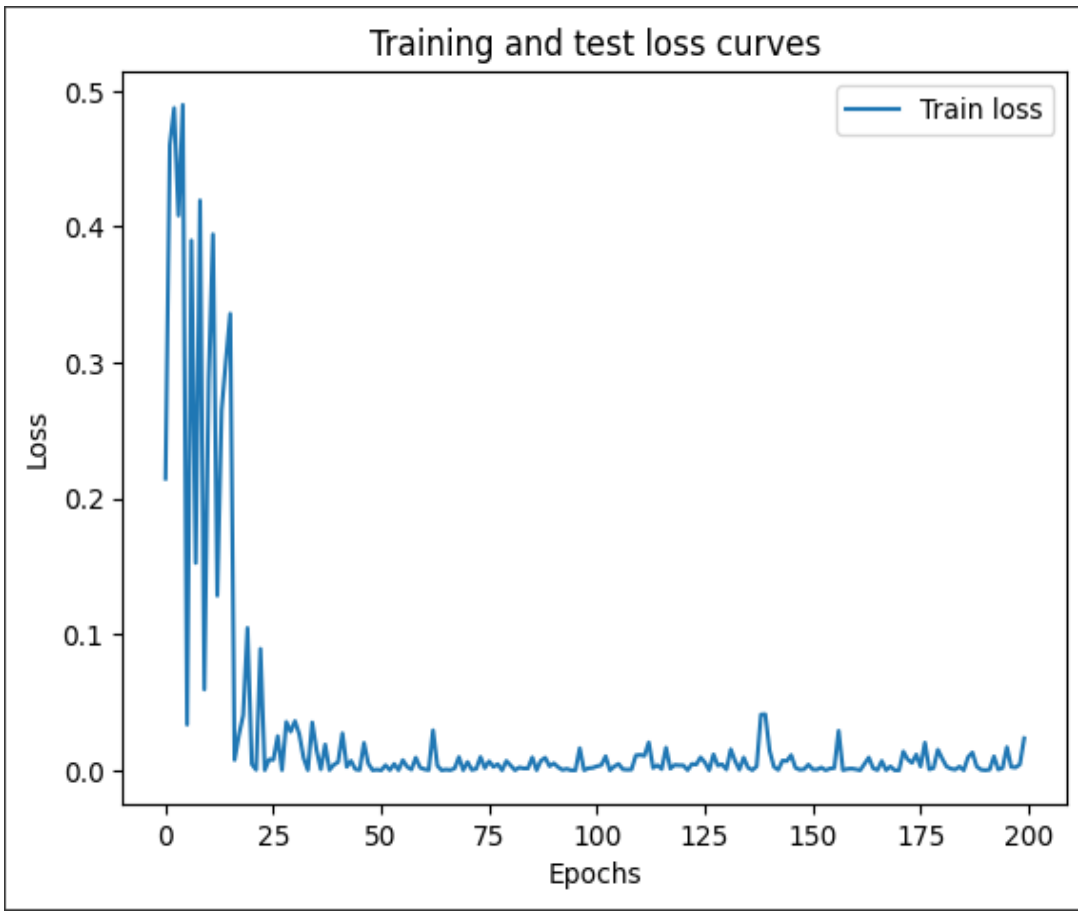

Рисунок 7 — Функція втрат для моделі з пошуку ентропії заплутаності.

Нище показано розподіл ентропії при встановленій цільовій ентропії 0.3:

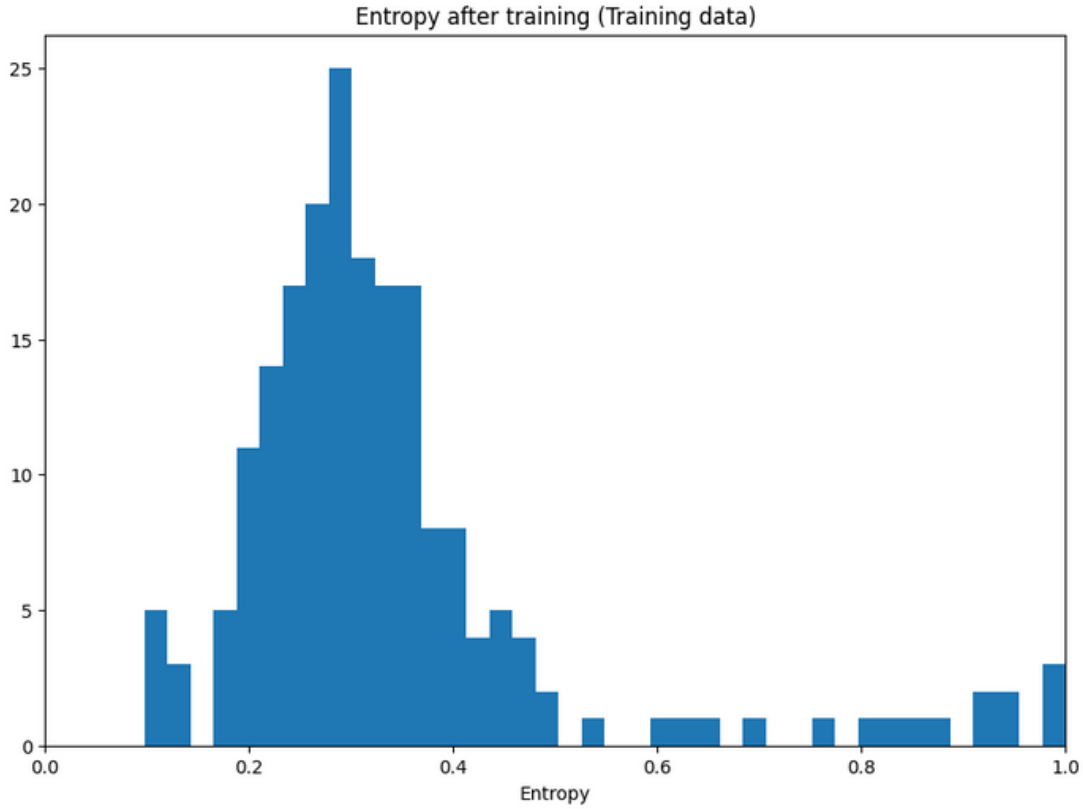

Рисунок 8 — Розподіл ентропії після тренуванням моделі на тренованих даних.

Як видно на малюнку, розподіл змінився, тепер він між 0,1 та 0.5. Отже, наша модель дечому навчилася. Давайте перевіримо нашу модель на тестовому наборі даних.

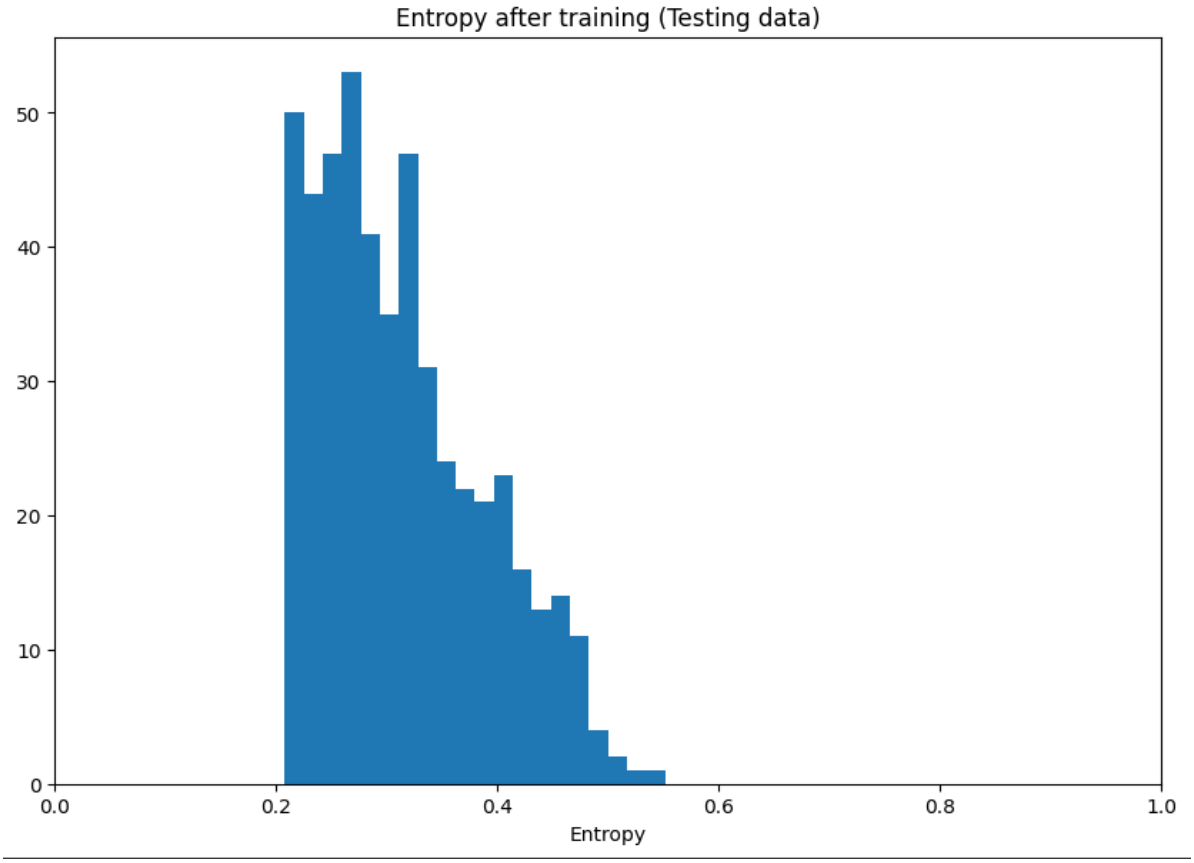

Рисунок 9 — Розподіл ентропії після тренуванням моделі на тестових даних (крок навчання 0.1).

Тут ми можемо побачити розподіл ентропії після навчання на тестовому наборі даних. І результат навіть кращий, ніж із навчальним набором даних. Отже, як ви бачите, наша нейронна мережа дала нам вбудовування для нашої схеми на вказану ентропію. Використовуючи ці параметри, можна навчити більш складну модель, контролюючи середнє значення ентропії в схемі.

На рисунку 10 було встановлено менший крок навчання (0.01) що дало змогу зменшити ентропію (код моделі присутній у додатку А).

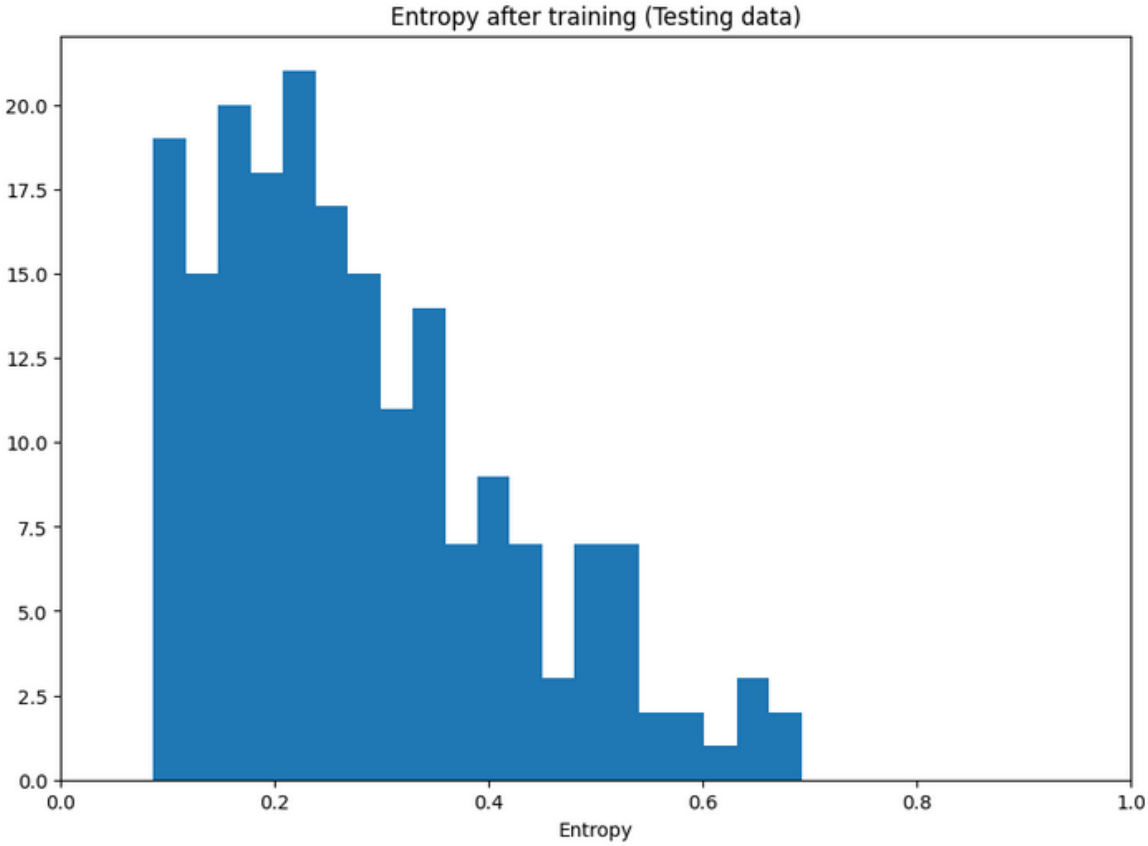

Рисунок 10 — Розподіл ентропії після тренуванням моделі на тестових даних (крок навчання 0.01).

Нище представлена 4 - кубітна квантова схема.

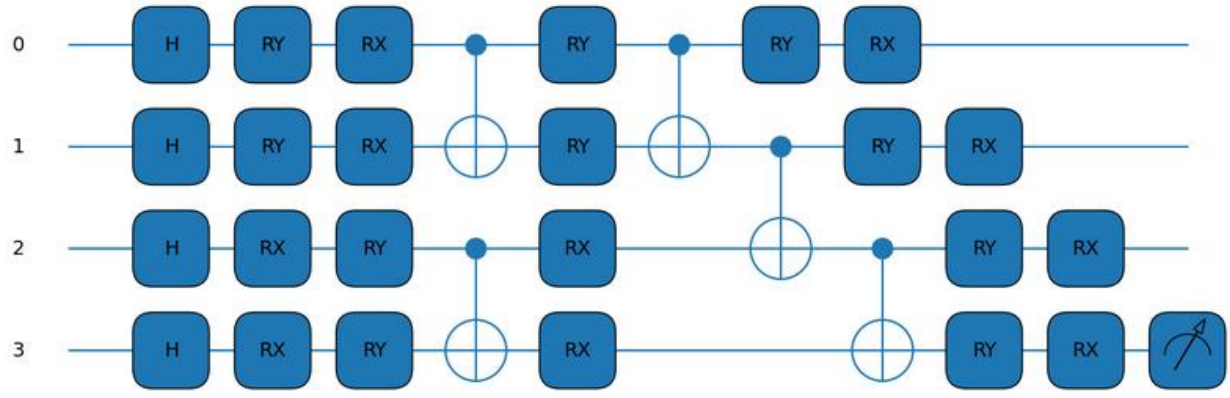

Рисунок 11 — Квантова схема для моделі з пошуку ентропії заплутаності (4-кубітна)

У цієї схеми з'явився додатковий шар воріт Адамара, а змінні значення, які тренуються під час навчання, тепер є одновимірними тензорами, вони присутні

в шарі RY та RX. Так виглядає розподіл ентропії перед тренуванням моделі на тестових даних для кроку навчання 0.1 (рисунок 12).

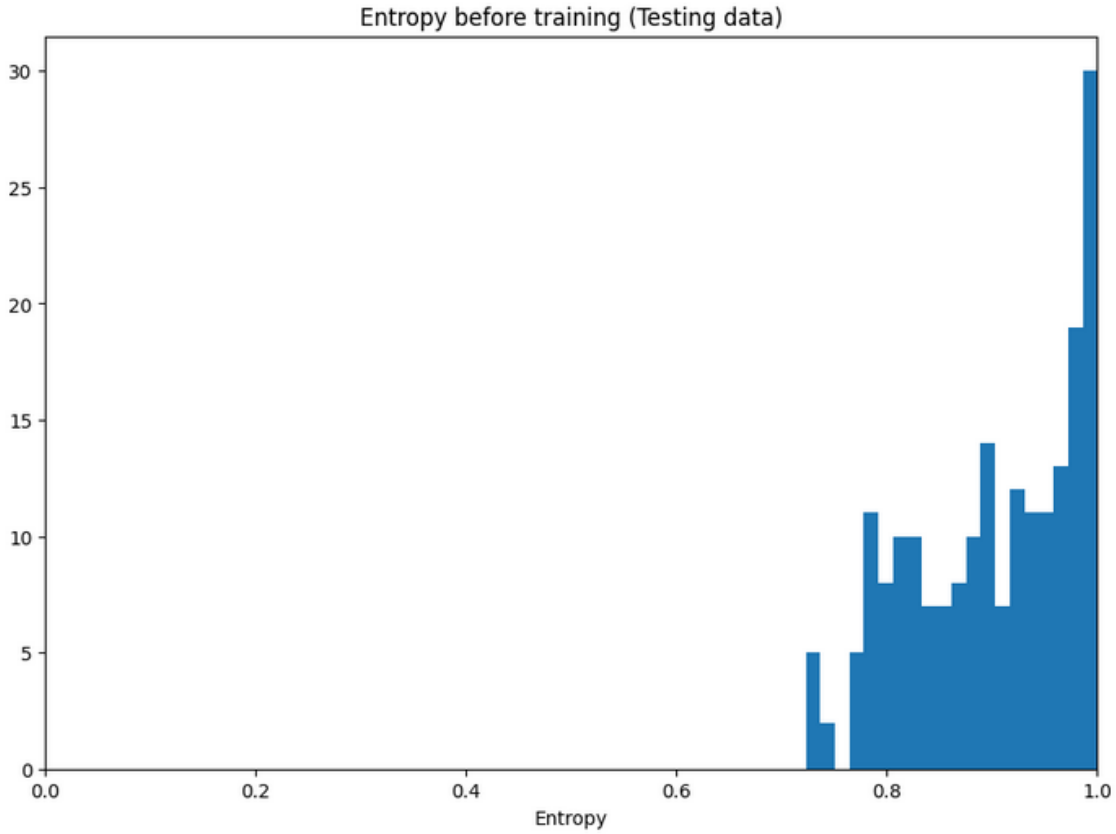

Рисунок 12 — Розподіл ентропії перед тренуванням моделі на тестових даних (крок навчання 0.1, 4 - кубіти).

На рисунку 13 та 14 показано розподіл ентропії після тренуванням моделі на тренованих та тестових даних відповідно. Як видно з гістограм модель успішно навчена. На рисунку 15 зображена функція втрат для 4 кубітної квантової схеми з кроком навчання 0.1.

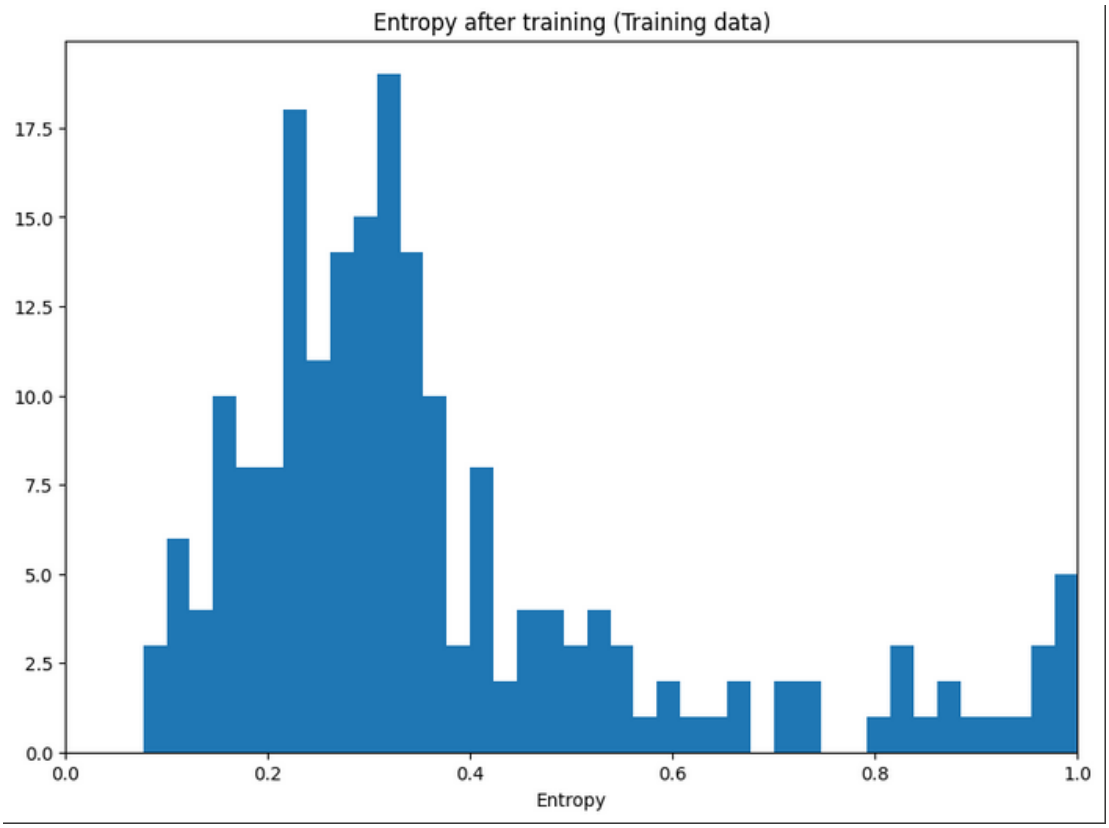

Рисунок 13 — Розподіл ентропії після тренуванням моделі на тренованих даних (4 - кубіти).

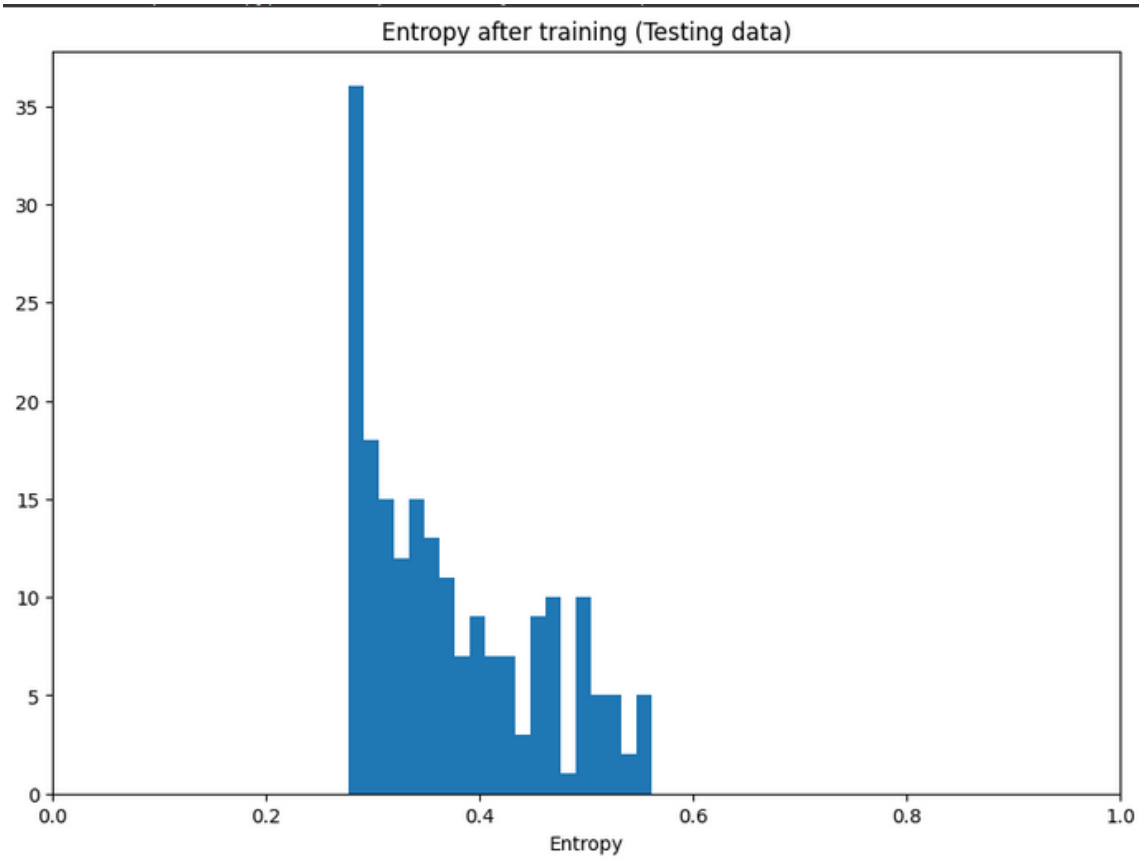

Рисунок 14 — Розподіл ентропії після тренуванням моделі на тестових даних (крок навчання 0.1, 4 кубіти).

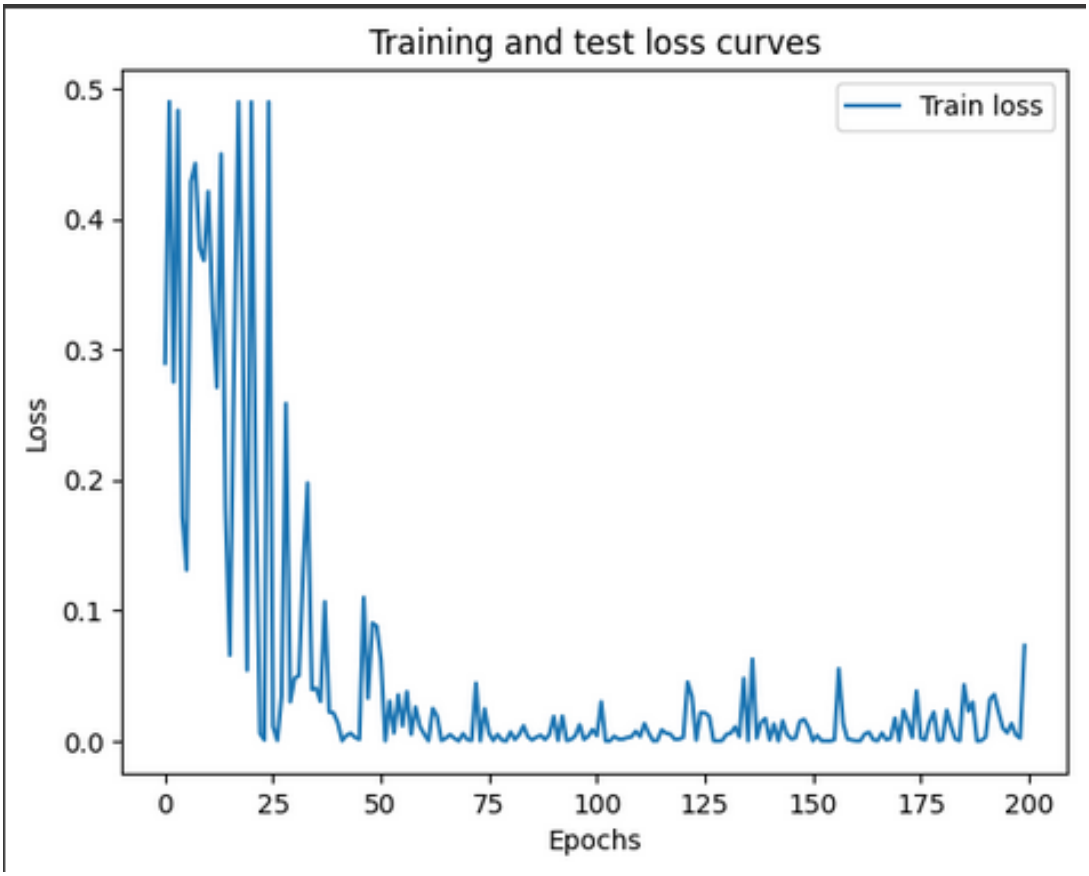

Рисунок 15 — Функція втрат для моделі з пошуку ентропії заплутаності (крок навчання 0.1, 4 кубіти).

Таким чином використовуючи цю модель можна керувати заплутаністю квантової схеми покращення певної потрібної вам моделі. Нище представлений алгоритм керування заплутаності для трансферного машинного навчання.

Перехід від класичного до квантового

Тут було використано підручник з квантового трансферного навчання від Pennylane[38]. У цьому посібнику трансферне навчання було застосовано до класифікатора зображень на основі гібридної класично-квантової мережі.

Цей експеримент зосереджений на схемі навчання передачі CQ, про яку я згадував раніше. Конкретний приклад виглядає так:

1. Для попередньо навченої мережі використовується ResNet18, представлений Microsoft [39].

2. Потім видаляємо останній шар ResNet18 і отримайте блок попередньої обробки, який відображає будь-яке вхідне зображення високої роздільної здатності в 512 абстрактних об’єктів.

3. Такі особливості класифікуються 4-кубітною варіаційною квантовою схемою, затиснутою між двома класичними шарами.

4. Гібридна модель навчена на наборі даних, що містить зображення бджіл і мурах.

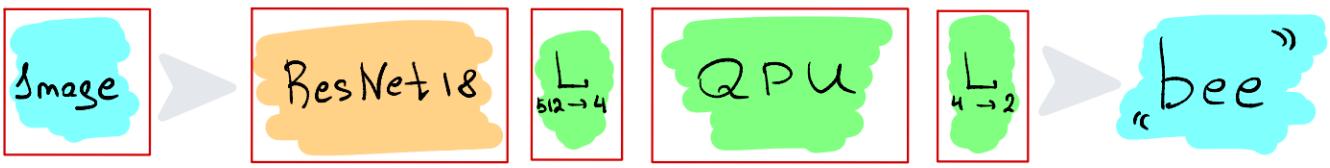

Рисунок 16 – Графічне зображення обробки даних.

Це алгоритм навчання з класичного переходу на квантовий. Квантова схема, яка

використовується в алгоритмі, виглядає наступним чином:

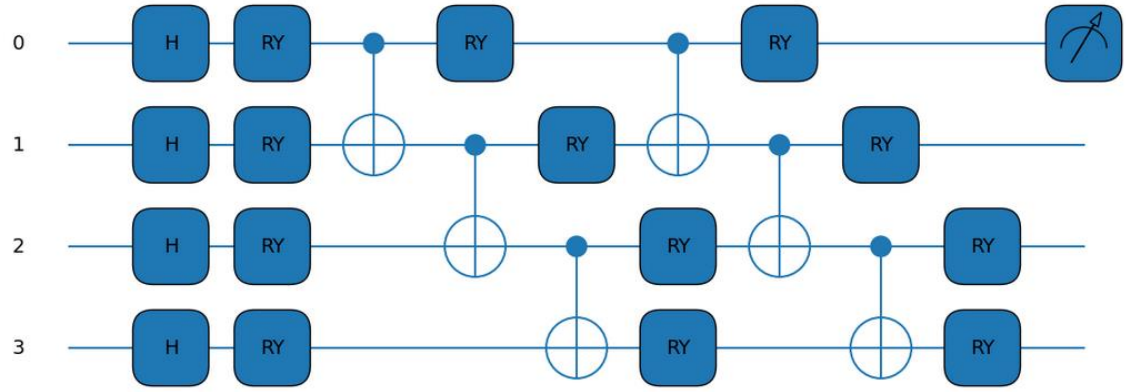

Рисунок 12— Алгоритм трансферного навчання квантової схеми

Щоб реалізувати мій алгоритм в алгоритмі трансферного навчання, потрібно створити квантову схему, яка використовується в алгоритмі трансферного навчання та додати вбудування яке буде контролювати ентропію. Щоб створити вбудовування, потрібно додати додаткові шари, щоб модель мала параметри для навчання. Тому було додано два додаткових шари з вентилями RX і RY, а також додаткові вентелі CNOT.

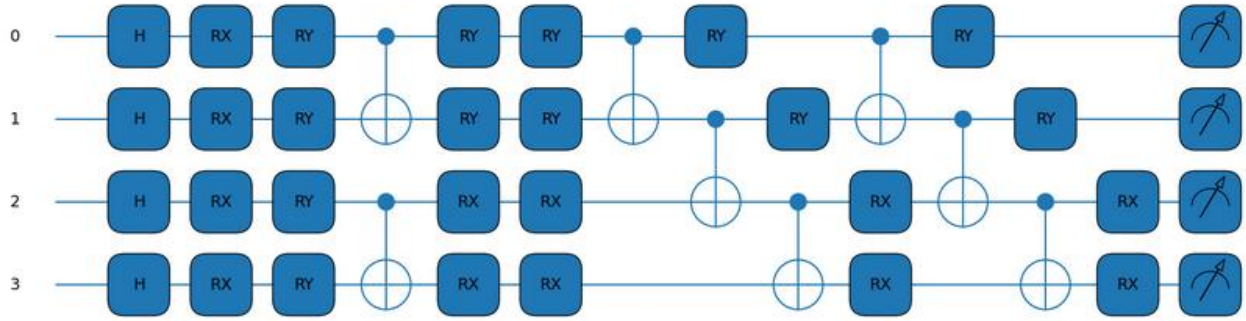

Рисунок 13 - Алгоритм трансферного навчання квантової схеми з додатковими шарами.

Параметри, які буде видавати модель, повинні бути реалізовані після класичного лінійного шару (nn.Linear). Після цього алгоритм повинен мати такий

вигляд:

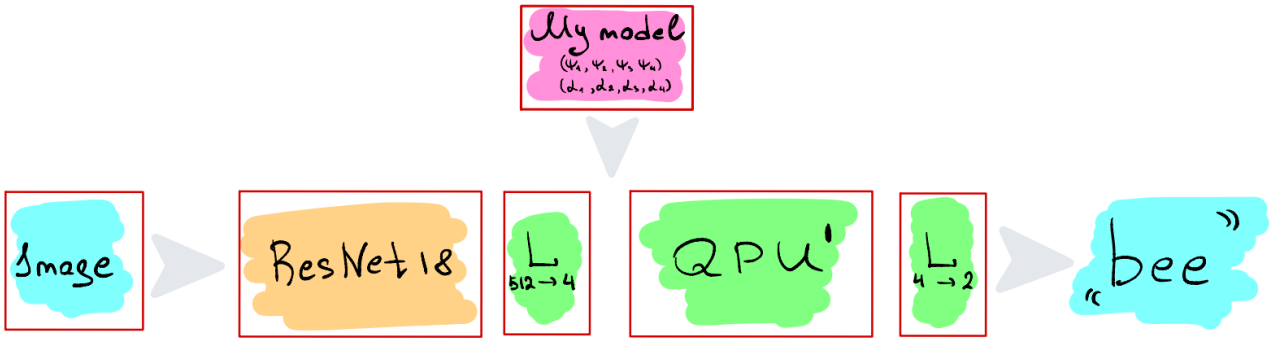

Рисунок 17 - графічне представлення обробки даних з моделлю «обчислення заплутаності».

Таким чином, заморожуючи лінійний шар Linear(512 -> 4), модель з обчислення заплутаності (My model) тренується для різних значень ентропії. Ці дані зберігаються для подальшого використання у моделі.

Висновок

У цій дипломній роботі була створена модель для квантового машинного навчання, яка успішно досягає заданої ентропії заплутаності. Це досягнення створює нові можливості для застосування систем машинного навчання та квантових обчислень. Результати дослідження демонструють, що модель успішно обробляє заплутані квантові дані, що свідчить про її потенційне застосування в ряді областей, де потрібне найкраще використання квантових ресурсів. Ці висновки можна застосувати для подальшого створення моделей квантового машинного навчання та вдосконалення технологій квантового обчислення.

Загалом наше дослідження демонструє важливість розгляду ролі ентропії заплутаності в моделях квантового машинного навчання. Він підкреслює потенціал для вбудовування певного ступеня заплутаності для покращення результатів навчання та забезпечує основу для оптимізації продуктивності алгоритмів квантового машинного навчання.

Додаток А.

```
from pennylane.math.utils import requires_grad
import matplotlib.pyplot as plt
import pennylane as qml
import torch
from torch.autograd import Variable
from pennylane import numpy as np

# Create a random seed
np.random.seed(67)
torch.manual_seed(42)

# Establish our weights

phi = Variable(torch.tensor(1.), requires_grad=True)
theta = Variable(torch. tensor (0.05), requires_grad=True)
psi = Variable(torch.tensor(np.pi), requires_grad=True)
alpha = Variable(torch.tensor(1.), requires_grad=True)

# Create targer value and epoch
target=torch.tensor(0.3)
epoch = 200
# Create device

dev = qml.device('default.qubit', wires=2)

# Create decorator

@qml.qnode(dev, interface= 'torch')

# Create quantum circuit, (phi, theta, psi, alpha) our weight
# (params) our random parameters
def circuit(phi, theta, psi, alpha, params):
```

```
    qml.RX(theta, wires=0)
    qml.RX(psi, wires=1)
    qml.RY(phi, wires=0)
    qml.RY(alpha, wires=1)
    qml.CNOT(wires=[0, 1])
    qml.RY(torch.tensor(params[0]), wires=0)
    qml.RY(torch.tensor(params[1]), wires=1)
    qml.RX(torch.tensor(params[2]), wires=0)
    qml.RX(torch.tensor(params[3]), wires=1)
    return qml.density_matrix([1])

# Calculate second Renyi entanglement entropy using our weights and random parameters

def calculate_entropy(phi, theta, psi, alpha, params):
  density_matrix=(circuit (phi, theta, psi, alpha, params))
  entropy = -torch.log2(torch.trace(density_matrix**2))

 #entropy = -torch.trace(density_matrix * torch.log2(density_matrix))
 #print(entropy)
  return entropy

# Calculate cost function using entanglement entropy and target value of entanglement entro
py
def cost(entropy, target):
  return torch.abs(entropy-target)**2



loss_count=[]
test_loss_count=[]
all_entropy=[]
entropy_bf_train=[]
train_entropy=[]
# Create optimizer
opt = torch.optim.Adam([phi, theta, psi, alpha], lr = 0.01)
```

```
train_matrix = np.random.rand(epoch, 4)
test_matrix = np.random.rand(200, 4)
#test_matrix *= 5

# Create loss calculating loop

# Train the model on the training set

for i in range(len(test_matrix)):
    entropy = calculate_entropy(phi, theta, psi, alpha, test_matrix[i])

    entropy_bf_train.append(entropy.detach().numpy())

fig, ax = plt.subplots(figsize =(10, 7))
ax.hist(entropy_bf_train, bins = 20)
ax.set_title("Entropy before training (Testing data)")
plt.xlabel("Entropy")
ax.set_xlim(0, 1)
plt.show

for i in range(len(test_matrix)):
    opt.zero_grad()
    entropy = calculate_entropy(phi, theta, psi, alpha, train_matrix[i])
    train_entropy.append(entropy.detach().numpy())
    loss = cost(entropy, target)
    loss.backward()
    opt.step()
    if i % 20 == 0:
        print(f"Epoch= {i} | Loss= {loss}")
    loss_count.append(loss.detach().numpy())



fig, ax = plt.subplots(figsize =(10, 7))
ax.hist(train_entropy, bins = 40)
```

```
ax.set_title("Entropy after training (Training data)")
plt.xlabel("Entropy")
ax.set_xlim(0, 1)
plt.show



# Evaluate the model on the testing set
total_loss = 0
for i in range(len(test_matrix)):
    entropy = calculate_entropy(phi, theta, psi, alpha, test_matrix[i])
    test_loss = cost(entropy, target)
    total_loss += test_loss
    test_loss_count.append(test_loss.detach().numpy())
    all_entropy.append(entropy.detach().numpy())

avg_loss = total_loss / len(test_matrix)
print(f"Average loss on testing set: {avg_loss}")
```

Посилання

[1] Luis J. Boya (Zaragoza University) - The Thermal Radiation Formula of Planck (1900)

[2] Einstein, A., B. Podolsky, and N. Rosen, 1935, “Can quantum-mechanical description of physical reality be considered complete?”, *Physical Review*, 47: 777–780

[3] Feynman, R. P. (1982). Simulating Physics with Computers. International Journal of Theoretical Physics, 21(6/7), 467-488.

[4] Shor, P. W. (1997). Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. SIAM Journal on Computing, 26(5), 1484-1509.

[5] Lloyd, S., Mohseni, M., & Rebentrost, P. (2013). Quantum principal component analysis. Nature Physics, 10(9), 631-633.

[6] Peruzzo, A., McClean, J., Shadbolt, P., Yung, M.-H., Zhou, X.-Q., Love, P. J., Aspuru-Guzik, A., & O'Brien, J. L. (2014). A variational eigenvalue solver on a photonic quantum processor. Nature Communications, 5, 4213.

[7] Farhi, E., Goldstone, J., & Gutmann, S. (2014). A Quantum Approximate Optimization Algorithm. arXiv preprint arXiv:1411.4028.

[8] M. Srikumar, C. D. Hill, L. C. L. Hollenberg, Clustering and enhanced classification using a hybrid quantum autoencoder (2021). arXiv: 2107.11988.

[9] A. Chalumuri, R. Kune, B. S. Manoj, A hybrid classical-quantum approach for multi-class classification, Quantum Information Processing 20 (3) (mar 2021). doi:10.1007/s11128-021-03029-9.

[10] S. L. Wu, S. Sun, W. Guan, C. Zhou, J. Chan, C. L. Cheng, T. Pham, Y. Qian, A. Z. Wang, R. Zhang, M. Livny, J. Glick, P. K. Barkoutsos, S. Woerner, I. Tavernelli, F. Carminati, A. D. Meglio, A. C. Y. Li, J. Lykken, P. Spentzouris, S. Y.-C. Chen, S. Yoo, T.-C. Wei, Application of quantum machine learning using the quantum kernel algorithm on high energy physics analysis at the lhc (2021). arXiv:2104.05059.

[11] https://en.wikipedia.org/wiki/Gradient

[12] https://www.nobelprize.org/prizes/physics/2022/summary/

[13] von Neumann, J. (1955). Mathematical foundations of quantum mechanics. Princeton University Press.

[14] Nielsen, M. A., & Chuang, I. L. (2010). Quantum computation and quantum information. Cambridge University Press.

[15] Horodecki, R., Horodecki, P., Horodecki, M., & Horodecki, K. (2009). Quantum entanglement. Reviews of modern physics, 81(2), 865.

[16] Rényi, A. (1960). On measures of entropy and information. In Proceedings of the 4th Berkeley Symposium on Mathematical Statistics and Probability (Vol. 1, No. 1960, pp. 547-561).

[17] Nielsen, M. A., & Chuang, I. L. (2010). Quantum computation and quantum information. Cambridge University Press.

[18] McClean, J. R., Romero, J., Babbush, R., & Aspuru-Guzik, A. (2018). The barren plateau phenomenon: the difficulty of training deep neural networks.

[19]https://www.researchgate.net/publication/359054180_The_Existence_of_Barren_Plateaus_in_Detection_of_Quantum_Entanglement

[20] Liao, Q., Wang, L., Wang, X., & Wang, H. (2021). Quantum-inspired neural network for mitigation of the barren plateau phenomenon. Physical Review Research, 3(3), 033090.

[21] Chatterjee, S., & Bhattacharyya, A. (2021). Avoiding barren plateaus with classical deep neural networks. Physical Review A, 103(5), 052404

[22] "Barren Plateau Mitigation by Early Stopping of Quantum Neural Network Training" by James Stokes, et al. (2021)

[23] "Optimizing Neural Networks with Kronecker Product Based Covariance Approximations" by Roger B. Grosse, et al. (2015)

[24] Lloyd, S., Schuld, M., Ijaz, A., Izaac, J., & Killoran, N. (2018). Quantum embeddings for machine learning.

[25] F. Verstraete, V. Murg, and J. Cirac, “Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin

systems," Advances in Physics, vol. 57, no. 2, pp. 143-224, 2008.

[26] G. Vidal, "Entanglement renormalization," Phys. Rev. Lett. 99, 220405 (2007).

[27] Cirac, I., Perez-Garcia, D., Schuch, N., & Verstraete, F. (2011). Matrix product states and projected entangled pair states: Concepts, symmetries, and theorems.

[28] Huang, Hsin-Yuan, Richard Kueng, and John Preskill, "Predicting many properties of a quantum system from very few measurements", Nature Physics 16.10 (2020): 1050-1057.

[29] Wang, Y.-X., Mu, L.-Z., Vedral, V., & Fan, H. (2003). Entanglement Rényi α-entropy. Physical Review A, 67(2), 022304.

[30] Islam, R., Ma, R., Preiss, P. M., Tai, M. E., Lukin, A., Rispoli, M., & Greiner, M. (2015). Measuring entanglement entropy in a quantum many-body system. Nature, 528(7580), 77-83.

[31] Brydges, T., Elben, A., Jurcevic, P., Vermersch, B., Maier, C., Lanyon, B. P., Zoller, P., Blatt, R., & Roos, C. F. (2019). Probing entanglement entropy via randomized measurements. Nature communications, 10(1), 1-10.

[32] https://pennylane.ai/qml/what-is-quantum-computing.html

[33] https://ftp.cs.wisc.edu/machine-learning/shavlik-group/torrey.handbook09.pdf

[34] https://www.v7labs.com/blog/transfer-learning-guide

[35] Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? In Advances in neural information processing systems (pp. 3320-3328).

[36] Raina, R., Battle, A., Lee, H., Packer, B., & Ng, A. Y. (2007). Self-taught learning: transfer learning from unlabeled data. Proceedings of the 24th international conference on Machine learning, 759-766.

[37] Jang, J.-S. R., Sun, C.-T., & Mizutani, E. (1997). Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence. Prentice Hall.

[38] https://pennylane.ai/qml/demos/tutorial_quantum_transfer_learning.html

[39] Kaiming He, Xiangyu Zhang, Shaoqing ren and Jian Sun. *Deep residual learning for image recognition*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 770-778 (2016)