Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

КВАЛІФІКАЦІЙНА РОБОТА

на тему «Порівняльний аналіз сучасних алгоритмів та фреймворків для Quantum Machine Learning»

за спеціальністю 121 «Інженерія програмного забезпечення»

Керівник кваліфікаційної роботи

Гороховський К. С.

(*прізвище та ініціали*)

_____ (*підпис*)

«____» _____ 2026 р.

Виконала студентка Реуцька А. О.

(*прізвище та ініціали*)

_____

«____» _____ 2026 р.

Київ-2026

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем Факультету інформатики

ЗАТВЕРДЖУЮ

Зав. кафедри мультимедійних систем,

Доцент., к. н. Жежерун О. П.

(підпис)

„____”_____2026 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на кваліфікаційну роботу

студентці Реуцькій Арині Олегівні факультету інформатики 4-го курсу

ТЕМА Порівняльний аналіз сучасних алгоритмів та фреймворків для Quantum Machine Learning

Зміст ТЧ до кваліфікаційної роботи:

1. Індивідуальне завдання

2. Календарний план

3. План роботи

Зміст

Анотація

Вступ

Розділ 1. Аналіз стану та наукових підходів у QML

1.1 Взаємозв’язок квантових обчислень та машинного навчання в прикладних задачах

1.2 Сучасні алгоритмічні підходи

1.3 Переваги, обмеження та проблемні аспекти

Розділ 2. Огляд та порівняльний аналіз фреймворків QML

2.1 Критерії порівняльного дослідження

2.2 Характеристика та можливості фреймворків

2.3 Порівняльний аналіз реалізацій експериментів

2.4 Обґрунтування вибору інструментів та середовищ

Розділ 3. Практична реалізація задачі обробки радіосигналів

3.1 Постановка задачі

3.2 Попередня підготовка вхідних даних

3.3 Реалізація моделей обробки сигналів

3.4 Порівняльний аналіз отриманих результатів

Висновки

Список використаних джерел

Додатки

Дата видачі “___” _____ 2026 р.

Керівник _____

ЗМІСТ

АНОТАЦІЯ 5
ВСТУП 7
РОЗДІЛ 1. АНАЛІЗ СТАНУ ТА НАУКОВИХ ПІДХОДІВ QML 9
1.1 Взаємозв'язок квантових обчислень та машинного навчання в прикладних задачах 9
1.2 Сучасні алгоритмічні підходи 12
1.3 Переваги, обмеження та проблемні аспекти 15
РОЗДІЛ 2. ОГЛЯД ТА ПОРІВНЯЛЬНИЙ АНАЛІЗ ФРЕЙМВОРКІВ QML 18
2.1 Критерії порівняльного дослідження 18
2.2 Характеристики та можливості фреймворків 20
2.3 Порівняльний аналіз реалізацій експериментів 23
2.3.1 Qulacs 23
2.3.2 Qibo 27
2.3.3 TorchQuantum 30
2.3.4 TensorFlow Quantum 35
2.3.5 Amazon Braket SDK 39
2.3.6 PennyLane 43
2.4 Обґрунтування вибору інструментів та середовищ 49
РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ЗАДАЧІ ОБРОБКИ РАДІОСИГНАЛІВ 51
3.1 Постановка задачі 51
3.2 Попередня підготовка вхідних даних 52
3.3 Реалізація моделей обробки сигналів 54
3.4 Порівняльний аналіз отриманих результатів 55
ВИСНОВКИ 65
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 67
ДОДАТКИ 73

АНОТАЦІЯ

В даній кваліфікаційній роботі досліджено сучасні алгоритми та фреймворки Quantum Machine Learning у контексті задачі автоматичної класифікації модуляції радіосигналів. Актуальність роботи зумовлена потребою у розробці та порівнянні інструментів машинного навчання, здатних працювати з високорозмірними, шумовими та нелінійними сигналовими даними, а також зростанням інтересу до гібридних квантово-класичних моделей в задачах обробки інформації.

В першому розділі розглянуто теоретичні засади QML, проаналізовано взаємозв'язок квантових обчислень і машинного навчання, описано сучасні алгоритмічні підходи QML, зокрема квантові ядрові методи, варіаційні квантові схеми, квантові нейронні мережі та гібридні моделі. Також визначено основні переваги, обмеження та проблемні аспекти застосування QML, пов'язані з NISQ-пристроями, кодуванням даних, стабільністю навчання та інтерпретованістю результатів.

У другому розділі виконано огляд та порівняльний аналіз фреймворків QML. Було сформовано критерії оцінювання, що охоплюють архітектурні особливості фреймворків, динаміку навчання, якість класифікації, обчислювальну вартість та стабільність середовища. Проаналізовано можливості PennyLane, TorchQuantum, TensorFlow Quantum, Qulacs, Qibo та Amazon Braket, а також обґрунтовано вибір інструментів для подальшої практичної реалізації.

В третьому розділі реалізовано практичну задачу багатокласової класифікації модуляцій BPSK, QPSK, 8PSK та QAM16 на основі датасетів RadioML 2016.10A та RadioML 2016.10B. Було виконано попередню підготовку сигналових даних, сформовано ознаковий простір, реалізовано моделі TorchQuantumFineTuned та QiboFineTuned, проведено внутрішнє й зовнішнє

тестування, а також здійснено порівняльний аналіз отриманих результатів. За підсумками експериментів встановлено, що обидві моделі є працездатними для задачі класифікації I/Q-сигналів, при цьому QiboFineTuned продемонструвала дещо вищу якість на внутрішньому тесті та меншу обчислювальну вартість, а TorchQuantumFineTuned показала близькі результати на зовнішньому наборі даних.

Ключові слова: Quantum Machine Learning, QML, радіосигнали, RadioML, автоматична класифікація, Qibo, TorchQuantum, квантово-класичні моделі.

ВСТУП

Автоматична класифікація модуляції радіосигналів тримає свою важливість для бездротових систем зв’язку через своє поширене використання у спектральному моніторингу, когнітивному радіо, засобах радіоелектронного спостереження та адаптивних системах передавання даних. Таке значення зумовлене насамперед в необхідності розпізнавання типу модуляції сигналу в умовах змінного радіочастотного середовища, наявності шумів, обмеженої апріорної інформації та високої динамічності каналів зв’язку. Тому є актуальним застосування методів машинного навчання, здатних виявляти закономірності у складних сигналових даних та формувати рішення на основі їх амплітудних, фазових, частотних та статистичних характеристик [10], [30], [31]. Наразі розвиток ML суттєво розширив свої можливості, оскільки класичні моделі, зокрема згорткові нейронні мережі, рекурентні архітектури, графові нейронні мережі тощо, демонструють високу ефективність у задачах автоматичного розпізнавання модуляції. Наприклад, частою темою для досліджень є моделі з attention-компонентами, блоками “стиснення та збудження” (squeeze-and-excitation) та динамічними графовими структурами, що підтверджує актуальність подальшого вдосконалення архітектур обробки сигналів [10], [30], [31]. Водночас у задачах з високою розмірністю, шумом, обмеженими навчальними вибірками та потребою у компактних моделях виникає необхідність дослідження альтернативних підходів до формування ознакового простору. Окремий науковий інтерес становить Quantum Machine Learning, тобто напрям, у якому принципи квантових обчислень застосовуються для побудови моделей машинного навчання. QML використовує кодування класичних даних у квантові стани, параметризовані квантові схеми, квантові ядра, гібридні квантово-класичні архітектури та процедури вимірювання для отримання класичного результату. Це напрям, що поєднує математичний апарат квантової механіки з алгоритмами класифікації,

регресії, кластеризації, оптимізації, зниження розмірності та виявлення аномалій [1], [2], [4], [5], [6].

Об’єктом дослідження є процес застосування алгоритмів квантового машинного навчання до задачі автоматичної класифікації модуляції радіосигналів, а предметом є QML-фреймворки, способи підготовки сигналових даних, квантово-класичні моделі, метрики оцінювання якості та характеристики реалізацій в задачі розпізнавання модуляцій.

Проблематика дослідження полягає в тому, що практичне застосування QML у задачах автоматичної класифікації модуляції радіосигналів потребує не лише побудови окремих моделей, а й визначення придатності відповідних програмних фреймворків до роботи з реальними або наближеними до реальних сигналовими даними. Qiskit Machine Learning, TensorFlow Quantum, Qibo, QiboML, Qulacs, TorchQuantum та інші інструменти мають різні архітектурні принципи, різний рівень інтеграції з класичними ML-бібліотеками, різні можливості симуляції квантових схем, підтримку noise-aware обчислень, доступ до апаратних backend-ів та різні підходи до побудови гібридних моделей [19], [20], [21], [22], [23]. Відповідно метою роботи є порівняльний аналіз сучасних фреймворків QML та дослідження їх придатності до задачі автоматичної класифікації модуляції радіосигналів. Для досягнення цієї мети було поставлено такі завдання: огляд теоретичних засад QML, аналіз сучасних алгоритмічних підходів, формування єдиного протоколу експериментального оцінювання, підготовку сигналових даних, реалізацію квантово-класичних моделей та оцінювання отриманих результатів за метриками якості класифікації, стабільності навчання, ресурсної складності й придатності фреймворків до практичного використання.

РОЗДІЛ 1. АНАЛІЗ СТАНУ ТА НАУКОВИХ ПІДХОДІВ QML

1.1 Взаємозв’язок квантових обчислень та машинного навчання в прикладних задачах

Радіосигнал як об'єкт машинного аналізу має складну математичну структуру, оскільки його інформаційний зміст визначається не лише амплітудою, а й фазою, частотою, часовою динамікою, видом модуляції, рівнем шуму та умовами проходження через канал зв'язку. У цифровій обробці радіосигнал часто подається через комплексну огинаючу:

$$z(t) = I(t) + jQ(t),$$

де $I(t)$ є синфазною складовою, $Q(t)$ – квадратурною складовою, а j – уявною одиницею. Після дискретизації сигнал може бути представлений як послідовність I/Q-відліків, спектральних коефіцієнтів, статистичних ознак або частотно-часових характеристик. У задачі АМС кожному сигналу ставиться у відповідність мітка класу, що визначає тип модуляції або іншу цільову категорію. У загальному вигляді модель класифікації реалізує відображення:

$$f_\theta : R^d \to Y,$$

де R^d є простором ознак сигналу, Y множиною класів, а θ – параметрами моделі. Для радіосигналів значення d може бути великим через використання часових відліків, спектральних перетворень, статистичних характеристик й додаткових ознак, пов'язаних із рівнем шуму або структурою каналу.

Складність задачі автоматичного розпізнавання модуляції визначається кількома чинниками. Сигнали різних класів можуть мати близькі геометричні або спектральні характеристики. Також наявність шуму може змінювати розподіл ознак й зменшувати роздільність класів, а дані можуть бути представлені у різних форматах: сирі I/Q-відліки, спектрограми, статистичні вектори, ознаки після перетворення Фур'є або ознаки після зниження

розмірності. При цьому модель повинна зберігати якість класифікації при зміні SNR, тобто відношення потужності сигналу до потужності шуму:

$$SNR_{dB} = 10log_{10}(\frac{Psignal}{Pnoise}).$$

Наразі існують AMR-дослідження, де рівень SNR використовується як один з головних факторів оцінювання стійкості моделі, оскільки одна й та сама архітектура може демонструвати різні результати при 0 dB, 10 dB і 20 dB [10], [25], [30], [31].

Квантові обчислення базуються на представленні інформації через стани квантових систем. Стан системи з n кубітів описується вектором у гільбертовому просторі розмірності 2^n:

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i \; |i\rangle,$$

де α_i це комплексні амплітуди ймовірностей, що задовольняють умову нормування. Такий спосіб представлення інформації створює математичну основу для побудови складних ознакових просторів, у яких класичні дані можуть бути закодовані через амплітуди, фази та кореляції між кубітами [1], [2], [4], [5], [6].

QML використовує квантові обчислювальні моделі для реалізації окремих етапів машинного навчання: кодування даних, формування ознакового простору, оцінювання подібності між прикладами, оптимізації параметрів або побудови гібридних моделей. У QML класичний вектор ознак x перетворюється у квантовий стан через оператор кодування:

$$|\phi(x)\rangle = U_\phi(x) \; |0\rangle^{\otimes n}.$$

Оператор $U_\phi(x)$ називається квантовою картою ознак. Його структура визначає, як саме класичні ознаки сигналу переносяться у квантовий простір.

Для радіосигналів такими ознаками можуть бути нормалізовані I/Q-відліки, спектральні амплітуди, фазові характеристики, статистичні параметри або ознаки після попереднього зниження розмірності [6], [7], [34]. Одним із ключових наслідків квантового кодування є можливість побудови квантових ядер. Для двох вхідних векторів x_i та x_j квантове ядро визначається як:

$$K(x_i , x_j) = | \langle \phi(x_i) | \phi(x_j) \rangle |^2.$$

Ця величина характеризує подібність між двома прикладами після їх перенесення у квантовий ознаковий простір. У задачах класифікації радіосигналів така подібність може враховувати нелінійні взаємозв'язки між компонентами сигналу, які складно описати за допомогою простих евклідових або кореляційних метрик. Теоретична можливість квантового прискорення під час навчання з учителем була обґрунтована для окремих класів задач з використанням квантового оцінювання ядерної функції, однак практична ефективність таких підходів залежить від доступу до даних, способу кодування, кількості вимірювань та структури конкретного датасету [9], [11], [15], [16].

QML-парадигма значною мірою пов'язана з NISQ-пристроями, оскільки мають обмежену кількість кубітів, піддаються впливу шуму, мають похибки квантових вентилів та не забезпечують повноцінної корекції помилок. З цієї причини переважна частина практичних QML-моделей реалізується як гібридні квантово-класичні системи. У таких системах квантова частина відповідає за кодування даних, виконання параметризованої квантової схеми або оцінювання ядра, а класична частина виконує попередню обробку сигналів, оптимізацію параметрів, обчислення функції втрат і аналіз результатів [7], [8], [19], [20]. Задачі радіосигнальної та радарної класифікації є релевантними для дослідження QML, оскільки вони містять високорозмірні, шумові, частково нестаціонарні та нелінійні дані. До прикладу можна привести роботу з стійкими до атак QML для класифікація радіосигналів, де

досліджувалися квантові варіаційні класифікатори, наближене кодування амплітуди та стійкість до суперечливих атак у сценаріях методу знаків швидкого градієнта (FGSM), прогнозованого методу градієнтного спуску (PGD) та універсальних суперечливих збуреннях (UAP) [24]. Також з класифікацією сигналів радарів було досліджено вплив глибини анзаців на виразність QML-моделі, при чому датасет містив 4800 сигналів трьох типів літаків, а тестування проводилося для різних рівнів SNR [25]. А для відстеження об'єктів за допомогою радіолокації в IoT-enabled середовищах застосовувалося федеративне навчання з використанням квантових технологій, що допомогло розширити контекст QML від ізольованої класифікації до розподіленої обробки сигналових даних [27].

1.2 Сучасні алгоритмічні підходи

Алгоритмічні підходи Quantum Machine Learning формуються навколо кількох основних напрямів, такі як квантові ядрові методи, варіаційні квантові схеми, квантові нейронні мережі, моделі повторного завантаження даних, квантові методи зниження розмірності, квантові автоенкодери, квантові алгоритми k-найближчих сусідів, квантові машини опорних векторів, гібридні квантово-класичні нейронні мережі та програмні фреймворки для реалізації таких моделей. Кожен з цих напрямів має власний математичний апарат, вимоги до кодування даних, обмеження щодо кількості кубітів та різний рівень готовності до застосування на сучасних NISQ-пристроях [2], [4], [5], [6].

Методи квантового ядра є одними з найбільш формалізованих підходів QML, їх основна ідея полягає в тому, що класичні дані кодуються у квантові стани, а подібність між прикладами визначається через перекриття цих станів. У задачах класифікації таке ядро може бути використане в Support Vector Machine або інших ядрових алгоритмах. Для радіосигналів це означає, що класифікатор працює не безпосередньо з I/Q-відліками або спектральними коефіцієнтами, а з їхнім поданням у квантовому ознаковому просторі [2], [6],

[9], [34]. Доцільність методів квантового ядра визначається якістю квантовою картою ознак. Якщо карта ознак недостатньо виразна, то вона не формує корисного розділення між класами. Якщо карта ознак є надмірно складною, значення ядра можуть концентруватися й втрачати інформативність. Наприклад, у 2024 році було окремо розглянуто таке явище як експоненціальна концентрація в методах квантового ядра, за якого значення подібності між станами втрачають здатність відображати структуру даних [15]. А в 2025 році benchmarking-дослідження методів квантового ядра показало необхідність оцінки квантових ядер з точністю та проєкційних квантових ядер на різних типах задач, а не лише на окремих демонстраційних прикладах [16].

Ще одним популярним напрямом є варіаційні квантові алгоритми, що використовують параметризовану квантову схему, значення параметрів якої оптимізується класичним алгоритмом. В загальному вигляді стан такої моделі задається як:

$$| \psi(x, \theta)\rangle = U_\theta U_\phi(x) \mid 0\rangle^{\otimes n},$$

де $U_\phi(x)$ відповідає за кодування вхідних даних, U_θ за параметризовану квантову обробку, x це вхідний вектор ознак, а θ є параметрами моделі. Результат отримується шляхом вимірювання квантового стану, після чого класичний оптимізатор оновлює параметри. Такий цикл є основою VQC, QNN та багатьох гібридних QML-моделей [7], [8], [13]. Варіаційні квантові класифікатори є придатними для задач, у яких необхідно дослідити вплив глибини схеми, кількості параметрів, способу кодування та структури entanglement-шарів на якість класифікації. Для сигналових даних це має пряме значення, оскільки різні структури анзаців можуть по-різному відображати залежності між часовими, частотними та фазовими компонентами сигналу. У вищезгаданому дослідженні про класифікацію радіолокаційних сигналів було

показано, що неглибокі анзаци з менш ніж 70-тю “воротами” достатні для досягнення максимальної доступної продуктивності в умовах одного data-encoding блоку [25].

Окрему групу становлять схеми повторного завантаження даних. У цих моделях класичні ознаки вводяться у квантову схему не один раз, а повторно на кількох шарах. Такий підхід дозволяє збільшити функціональну виразність моделі без обов'язкового збільшення кількості кубітів. Дані схеми є хорошим варіантом для задач класифікації, оскільки кількість ознак сигналу зазвичай перевищує кількість доступних кубітів у симуляторі або на реальному квантовому пристрої. Це підтверджує дослідження QML за межами квантових методів, де показано, що моделі з повторним завантаженням даних мають окреме значення для NISQ-орієнтованих підходів, тому що дозволяють будувати компактні параметризовані схеми [14].

У класичних pipeline для AMR і класифікація радіолокаційних сигналів можуть використовуватися PCA, статистичний відбір ознак, нормалізація, масштабування, FFT-перетворення та формування компактного вектора ознак. У QML це має ще більше значення, оскільки кількість кубітів є обмеженою, а надмірно велика кількість ознак збільшує складність кодування. Quantum Principal Component Analysis [34], квантові автоенкодери та гібридні методи попереднього зниження розмірності можуть використовуватися для формування компактного подання перед класифікацією [2], [3], [5], [6]. У систематичному огляді виявлення аномалій для безпеки IoT було відібрано 124 роботи, а також окремо розглянуто роль розробки ознак, аналізу наборів даних, QML-підходів та експериментальних налаштувань [3]. Хоча цей напрям належить до безпеки IoT, його методологія є релевантною для радіосигнальних задач, оскільки обидві області працюють з потоковими, шумовими, високорозмірними та частково нестаціонарними даними.

Додатково можна привести ще декілька алгоритмічних підходів: квантово-класичні гібридні конволюційні нейронні мережі, де квантова частина використовується як шар обробки або вилучення ознак, а класична частина виконує подільшу класифікацію [26]; квантово-підтримане федеративне навчання для розподіленої обробки даних на основі радіолокації [27]; використання QML для оптичної та SAR-класифікації, що пов’язано з завданнями радіолокації та обробки сигналів [28]; інтелектуальні мережі та ворожі загрози, де QML розглядається в ширшому контексті безпечних та адаптивних бездротових мереж [29].

1.3 Переваги, обмеження та проблемні аспекти

Першою перевагою є можливість формування високорозмірного квантового простору ознак. Система з n кубітів має простір станів розмірності 2^n, що дозволяє компактно описувати складні багатовимірні структури. Для класифікації радіосигналів це означає, що вхідний сигнал може бути перетворений у квантове представлення, у якому взаємозв'язки між ознаками відображаються через амплітуди, фази та entanglement-структури. Такий підхід є теоретично привабливим для сигналів, у яких корисна інформація розподілена між часовими, частотними та фазовими компонентами [6], [9], [13], [14].

Другою перевагою є використання квантових ядер, оскільки ядрові методи дають можливість оцінювати подібність між прикладами після нелінійного відображення в ознаковий простір. Це задається через квантову карту ознак, а ядро обчислюється через перекриття квантових станів. Для сигналів з близькими класами модуляції або складними спектральними структурами квантове ядро може формувати інший критерій подібності, ніж стандартні класичні ядра. Водночас практична корисність залежить від відсутності концентрації, а також стабільності оцінювання ядрової матриці та відповідності карти ознак структурі даних [9], [15], [16], [34]. Ще однією

перевагою є компактність параметризації варіаційних квантових моделей. Параметризована квантова схема може містити відносно невелику кількість тренувальних параметрів, оскільки складність моделі формується не тільки кількістю параметрів, а й квантовою еволюцією стану [24], [25], [26].

До того ж існує потенціал QML у задачах, пов'язаних зі стійкістю до шуму та суперечливих збурень. Оскільки для радіосигналів це має окреме значення, то сам сигнал може бути спотворений як фізичними умовами каналу, так і навмисними збуреннями. У вищезгаданій роботі (див. пункт 1.1) з QML, стійкими до суперечливих даних, для класифікації радіосигналів було досліджено переносимість атак між квантовими варіаційними класифікаторами (QVC) та згортковими нейронними мережами (CNN), а також використано наближене кодування амплітуди для ефективного кодування радіосигнальних даних [24]. Для 6G network intelligence QML також розглядається як напрям, пов'язаний із захищеністю, адаптивністю та інтелектуальною обробкою мережевих даних [29].

Водночас існують суттєві обмеження з QML, зокрема кодування класичних даних у квантові стани. Для сигналів високої розмірності необхідно обрати спосіб кодування, який не створює надмірно глибокої схеми та не втрачає інформативні властивості сигналу. Кодування амплітуди є компактним за кількістю кубітів, але може потребувати складної підготовки стану. Тим часом як кутове кодування є простішим у реалізації, але потребує більшої кількості кубітів або повторного введення. Тобто для задач класифікації вибір стратегії кодування безпосередньо впливає на точність, час навчання, стабільність оптимізації та можливості запуску моделі на NISQ-пристрої [6], [24], [25].

Існують обмеження, що пов’язані безпосередньо з NISQ-пристроями, оскільки вони мають шум, обмежену кількість кубітів, похибки вентилів, короткий час когерентності та обмежену топологію взаємодії кубітів. Це обмежує глибину схем та підвищує роль симуляторів у QML-дослідженнях [8]. Також під час

навчання параметризованих квантових схем однією з проблем є “безплідні плато” (barren plateaus) – це області параметричного простору, у яких градієнти функції втрат стають дуже малими, а оптимізація втрачає ефективність. Це явище залежить від анзаців, початкової ініціалізації, типу функції втрат, глибини схеми, кількості кубітів та апаратного шуму [17]. Для пом'якшення цієї проблеми зазвичай розглядаються локальні функції втрат, спеціальні стратегії ініціалізації та пошарове навчання [18]. А при узагальненні QML-моделей часто виникають проблеми, що залежать від кількості тренувальних ґрат та навчальних прикладів, оскільки це має пряме значення для задач з невеликими датасетами або обмежено. кількістю прикладів для окремих класів [12].

Окремим проблемним аспектом є інтерпретованість QML-моделей через те, що квантові схеми оперують станами, амплітудами, фазами та результатами вимірювань, що ускладнює пряме пояснення причин конкретного рішення класифікатора. Для інженерних задач, пов'язаних із радіозв'язком, це має значення, оскільки практична система повинна бути стабільною, точною, прогнозованою та придатною до аналізу помилок [3], [29].

РОЗДІЛ 2. ОГЛЯД ТА ПОРІВНЯЛЬНИЙ АНАЛІЗ ФРЕЙМВОРКІВ QML

2.1 Критерії порівняльного дослідження

У цьому дослідженні критеріальна база була побудована таким чином, щоб охопити як теоретико-архітектурні, так й інструментально-прикладні властивості фреймворків. Першою групою критеріїв є архітектурні характеристики [4], [5], [6], а саме алгоритмічна репрезентативність фреймворку, тип моделі, яку він природно підтримує, і спосіб інтеграції з класичним стеком машинного навчання. Для частини платформ, зокрема PennyLane та TensorFlow Quantum, центральним є квантово-класичний граф. Для Qibo, Qulacs та Amazon Braket у протестованих конфігураціях головним об’єктом виступав квантовий екстрактор ознак з подальшим класичним класифікатором. TorchQuantum займає проміжне положення, оскільки поєднує PyTorch-стиль end-to-end навчання з явно вираженим гібридним дизайном.

Друга група критеріїв стосувалася динаміки навчання, до якої було віднесено прозорість процесу оптимізації [7], можливість аналізувати функцію втрат, норму градієнта, поведінку параметризованої схеми в часі та наявність діагностичних інструментів, що дозволяють інтерпретувати сам перебіг навчання. Цей критерій важливий для варіаційних квантових моделей, оскільки низька точність може бути пов’язана або зі слабкістю ознакового простору, або з проблемами оптимізації. Тому вибір фреймворків PennyLane, TorchQuantum й TensorFlow Quantum був зроблений і через можливість оцінити поведінку втрат та норму градієнта.

Третя група критеріїв пов’язана з прикладною якістю моделі. Усі реалізації оцінювалися за єдиним набором метрик, який включав accuracy, balanced accuracy, F1-score, AUC, середній час навчання та матрицю помилок для окремого репрезентативного запуску [35]. Вибір метрик зумовлений тим, що

вони охоплюють як жорстку якість класифікації, так і якість ранжування ймовірностей, а також дають змогу оцінити часову вартість серії запусків. Balanced accuracy була обрана як одна з центральних метрик, оскільки вона коректно відображає якість двокласової моделі в збалансованій постановці. AUC дозволяла судити про дискримінаційну здатність побудованого квантового простору ознак у тих випадках, де порогове рішення виявилося не ідеальним.

Окремим складником критеріїв порівняння була організація вхідного простору даних. У ролі єдиного джерела сигналів використовувався датасет RadioML 2016.10A, який офіційно описується як “синтетичний набір даних, створений за допомогою GNU Radio, що складається з 11 модуляцій при різних відношеннях сигнал/шум” [36], [37]. У наданих логах автоматична перевірка структури датасету підтвердила наявність 220 ключів у словнику, 11 класів модуляції, значень SNR від -20 до 18 дБ із кроком 2 дБ, а також масивів форми (1000, 2, 128) для кожної пари (модуляція, SNR). У межах поточного експериментального протоколу було сформовано бінарну вибірку обсягом 720 прикладів, зведену до простору ознак розмірності 8 після PCA, із балансом класів 360/360 і сумарною поясненою дисперсією 0.8872.

Ці характеристики датасету є принциповими для коректного порівняння фреймворків, оскільки вони задають межі інтерпретації результатів. По-перше, усі платформи працювали з однаковим ознаковим описом сигналу. По-друге, кількість компонентів після PCA була навмисно узгоджена з кількістю кубітів, що усунуло додатковий фактор відмінності між реалізаціями. По-третє, для всіх фреймворків використовувався однаковий набір тестових SNR, а саме 0, 10 і 18 дБ, що зробило можливим пряме зіставлення кривих якості, а не тільки окремих чисел. Таким чином, критерії порівняння були сформовані через конкретну схему, де кожен фреймворк опинявся в максимально зіставних умовах.

Четвертим змістовим критерієм стала обчислювальна вартість. У середовищі NISQ-подібних досліджень різниця між моделями часто виявляється в тому, чи можливо повторити запуск достатню кількість разів, не перетворюючи benchmark на надто дорогий обчислювальний процес [7], [8]. Тому середній час навчання був включений до числа центральних порівняльних характеристик, оскільки висока прозорість процесу навчання не завжди супроводжується прийнятною швидкодією, а висока якість класифікації не завжди потребує найбільшої часової вартості.

Останнім критерієм стала стабільність інструментального середовища. У практичних серіях запусків було зафіксовано, що різні екосистеми мають різний рівень чутливості до залежностей. TensorFlow Quantum вимагав суворого дотримання сумісності з TensorFlow 2.18.1 та legacy Keras, TorchQuantum потребував окремого коректного editable install, а Amazon Braket виявився чутливим до змішування з іншими пакетами у спільному runtime. Це означає, що для справедливого порівняння потрібно враховувати не тільки математичні властивості моделі, а й реальну здатність середовища відтворити експеримент без ручного усунення конфліктів [20].

2.2 Характеристики та можливості фреймворків

У сучасному стані галузі найбільш репрезентативними для порівняльного дослідження є ті фреймворки, які належать до різних архітектурних методів у QML. Наприклад, є фреймворки, що роблять ставку на варіаційні схеми й автоматичне диференціювання, насамперед PennyLane. Далі входять платформи, що інтегрують квантові обчислення у вже зрілі системи глибокого навчання, як-от TensorFlow Quantum. Нарешті, окрему інженерно важливу групу утворюють високопродуктивні симулятори й оркестратори на кшталт Qulacs, Qibo та Amazon Braket, які не завжди виступає лише в QML у вузькому сенсі, але визначають реальну вартість симуляції, режим налагодження та можливість масштабованого експерименту.

PennyLane є кросплатформною Python-бібліотекою для квантових обчислень, квантового машинного навчання і квантової хімії. В офіційній документації наголошується, що фреймворк побудований навколо автоматичного диференціювання квантових схем, підтримки гібридних квантово-класичних моделей та можливості запускати один й той самий квантовий workflow на різних бекендах [38], [39]. Така архітектура робить PennyLane одним із найрепрезентативніших середовищ для досліджень, у яких центральне місце займає процес навчання параметризованої квантової схеми, включно з динамікою функції втрат та поведінкою градієнтів.

TorchQuantum є фреймворком для квантових обчислень, побудованим поверх PyTorch і орієнтованим на динамічний обчислювальний граф, автоматичне диференціювання та пакетну обробку. Описується як фреймворк на базі PyTorch, що підтримує динамічний обчислювальний граф, обчислення градієнтів за допомогою autograd, інференцію та навчання в пакетному режимі, а також спрощене створення гібридних класично-квантових моделей [32]. Така архітектура дозволяє вбудовувати квантову частину безпосередньо в типову PyTorch-логіку навчання нейронної мережі. Це зменшує розрив між класичним та квантовим програмуванням, а також робить TorchQuantum природним інструментом для досліджень, де квантовий блок розглядається як компонент більш загальної моделі.

TensorFlow Quantum є бібліотекою для швидкого прототипування гібридних квантово-класичних моделей у середовищі TensorFlow. TensorFlow Quantum “зосереджується на квантових даних та створенні гібридних квантово-класичних моделей” [19], а його базова архітектура побудована навколо інтеграції Cirq-представлення квантових схем у TensorFlow-граф. Це означає, що квантові схеми, оператори спостереження та класичні параметри можуть бути представлені як частини єдиного обчислювального графа, а навчання виконується стандартними засобами Keras API. У основному джерелі TFQ

прямо наголошується, що моделі й операції будуються як потужні гібридні квантово-класичні системи, які поєднують квантові примітиви з типовою TensorFlow-інфраструктурою [19].

Qulacs доцільно розглядати як високопродуктивний симулятор квантових схем, орієнтований на швидке відтворення великих, шумових і параметризованих квантових схем. Це швидкий симулятор квантових схем для моделювання великих, зашумлених або параметричних квантових схем [23]. Серед прикладних сценаріїв у документації окремо представлено підхід Quantum Circuit Learning, тобто використання параметризованих квантових схем як моделі машинного навчання. Це дозволяє трактувати Qulacs як повноцінне середовище для гібридних квантово-класичних експериментів, де квантова схема виконує роль перетворювача ознак, а остаточна класифікація покладається на класичний модуль.

Qibo є open-source middleware для квантових обчислень, яке офіційно позиціонується як комплексна платформа з відкритим кодом для квантового моделювання, автономного керування квантовим обладнанням, його калібрування та визначення характеристик [21], [33]. На відміну від симуляторів, що зосереджені переважно на відтворенні квантових схем, Qibo орієнтується на ширшу екосистему квантових застосувань. У 2025 році було представлено бібліотеку Qiboml як інструмент для узгодження квантових та класичних компонентів у гібридних робочих процесах машинного навчання [22]. Отже, Qibo концептуально добре узгоджується з гібридними сценаріями, де квантова частина використовується для побудови ознак, а класична частина виконує остаточне навчання й класифікацію.

Amazon Braket є хмарною квантовою платформою AWS, яка надає доступ до квантових пристроїв, керованих симуляторів та інфраструктурних засобів для гібридні квантово-класичні робочі процесів. Документація Amazon Braket описує як локальний симулятор для швидкого прототипування, так і як

механізм повноцінної оркестрації гібридних алгоритмів на керованих класичних ресурсах [40]. У цьому полягає його відмінність від більшості академічно орієнтованих фреймворків: Braket є не лише бібліотекою, а й середовищем запуску та інфраструктурою керування експериментом. З прикладного погляду Braket має особливу цінність у задачах, де квантовий блок використовується як feature extractor, а вся схема експерименту повинна бути добре контрольованою з боку логування, часу виконання та повторюваності. Локальний режим дозволяє швидко прототипувати задачу без підключення віддалених ресурсів, а сама логіка Braket добре узгоджується з уявленням про експеримент як про керовану серію запусків.

2.3 Порівняльний аналіз реалізацій експериментів

2.3.1 Qulacs

Реалізація Qulacs (див. “розділ2_qulacs.ipynb”) показала достатньо високі результати при всіх трьох протестованих рівнях SNR, хоча її поведінка виявилася нелінійною. Середні значення для SNR = 0 дБ становили accuracy = 0.775, balanced accuracy = 0.775, F1 = 0.7800, AUC = 0.8622, при середньому часі навчання 0.5773 с. Для SNR = 10 дБ було отримано accuracy = 0.800, balanced accuracy = 0.800, F1 = 0.8027, AUC = 0.8861, а середній час навчання зріс до 1.9109 с. Для SNR = 18 дБ метрики дещо знизилися до accuracy = 0.725, balanced accuracy = 0.725, F1 = 0.7075, AUC = 0.8472, а середній час становив 0.9152 с. На відміну від попередньо розглянутого Amazon Braket, максимальна якість у Qulacs спостерігається не при 0 дБ, а при 10 дБ, тобто в умовах середнього рівня сигналу та шуму.

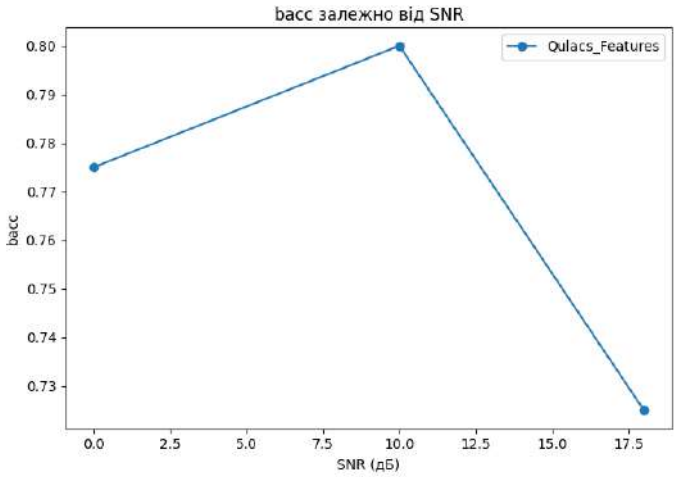


Рис. 2.1 Графік balanced accuracy Qulacs

Графік balanced accuracy для Qulacs демонструє помірно виражену нелінійну залежність від SNR. Значення 0.775 при 0 дБ, підвищення до 0.800 при 10 дБ і спад до 0.725 при 18 дБ свідчать про те, що в поточній конфігурації квантовий простір ознак, побудований за допомогою Qulacs, виявився найінформативнішим саме для середнього рівня шуму. Тобто це можна вважати як наслідок конкретної взаємодії між обраними статистичними та спектральними ознаками, процедурою PCA та характером квантового перетворення ознак у реалізації Qulacs. Графік balanced accuracy показує, що модель не є монотонно чутливою до зміни шуму, а отже її дискримінаційна здатність формується під впливом складнішої структури ознакового простору.

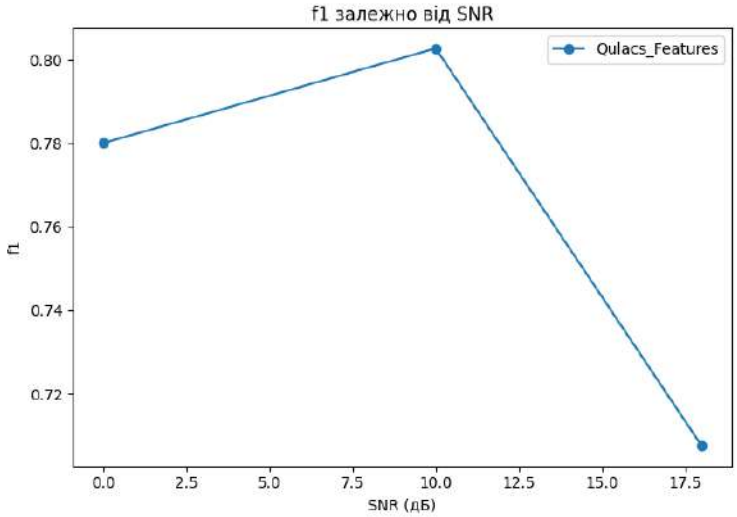


Рис. 2.2 Графік F1 Qulacs

Графік F1 для Qulacs відтворює ту саму загальну тенденцію, але дозволяє точніше оцінити баланс між precision і recall. Значення F1 = 0.7800 при 0 дБ, 0.8027 при 10 дБ і 0.7075 при 18 дБ свідчать про те, що найбільш узгоджена робота моделі щодо позитивного класу спостерігається також при 10 дБ. Зниження F1 при 18 дБ підтверджує, що зі зростанням цього рівня в поточній постановці збільшується частка асиметричних помилок, тобто модель уже не так добре підтримує одночасно і достатню точність, і достатню повноту. Водночас мінімальне значення F1 лишається істотно вищим за випадковий рівень, що дозволяє говорити про реальну корисність побудованого квантового представлення.

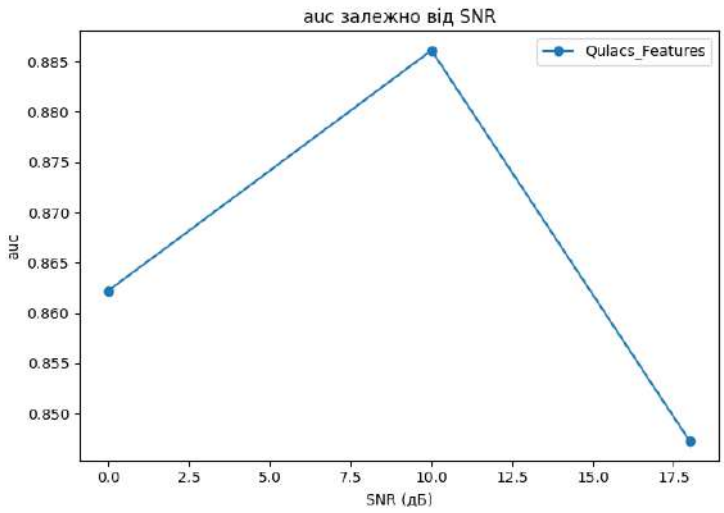


Рис. 2.3 Графік AUC Qulacs

Значення AUC = 0.8622 при 0 дБ, 0.8861 при 10 дБ і 0.8472 при 18 дБ показують, що модель у всіх випадках зберігає достатньо сильну здатність до розділення класів у просторі оцінених ймовірностей. Найкраще ранжування спостерігається при 10 дБ, що підтверджує загальний висновок, отриманий із графіків balanced accuracy та F1. Особливо важливо, що навіть у найслабшій точці кривої AUC суттєво перевищує 0.8, отже модель зберігає добру дискримінаційну здатність на всьому протестованому діапазоні.

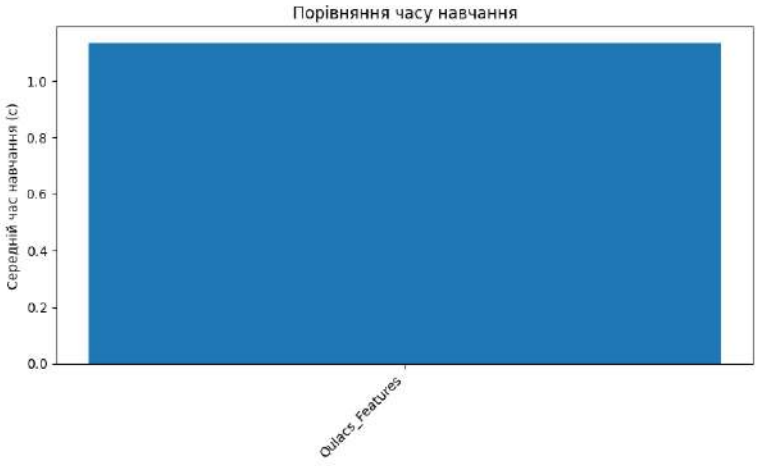


Рис. 2.4 Графік часу навчання Qulacs

Графік часу навчання для Qulacs свідчить про загалом низьку обчислювальну вартість реалізації. Хоча середній час при 10 дБ виявився вищим, ніж при 0 дБ і 18 дБ, абсолютні значення все одно залишаються невеликими, а саме близько однієї-двох секунд. Це є важливою перевагою Qulacs, оскільки в умовах багаторазового повторення експериментів швидкість симуляції стає одним із визначальних критеріїв практичної придатності фреймворку. Якщо зіставити ці результати з метриками якості, можна стверджувати, що Qulacs забезпечує добрий компроміс між точністю й витратами часу.

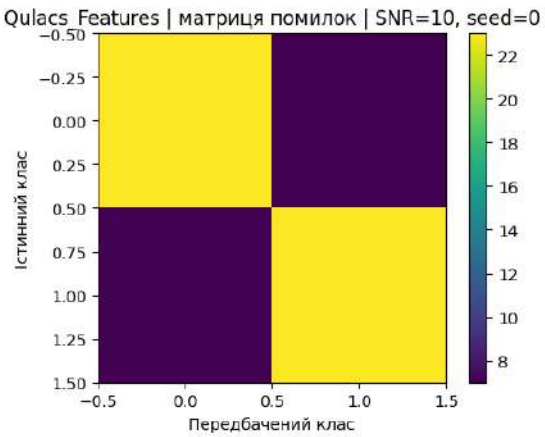


Рис. 2.5 Матриця помилок Qulacs

Матриця помилок для запуску SNR = 10 дБ, seed = 0, де було зафіксовано accuracy = 0.7667, balanced accuracy = 0.7667, F1 = 0.7667, AUC = 0.8456, показує відносно симетричний розподіл помилок між двома класами. Це

важливий результат, тому що за збалансованої вибірки така картина означає не систематичне зміщення в бік одного класу, а радше наявність помірного перекриття між класами в просторі ознак. Іншими словами, Qulacs не формує патологічно упереджену модель, але межа між класами залишається недостатньо різкою, щоб усунути хибні класифікації повністю.

2.3.2 Qibo

У поточному експерименті Qibo (див. “розділ2_qibo.ipynb”) продемонстрував дуже сильні результати. Для SNR = 0 дБ середні метрики становили accuracy = 0.9000, balanced accuracy = 0.9000, F1 = 0.9070, AUC = 0.9739, а середній час навчання дорівнював 0.5476 с. Для SNR = 10 дБ було отримано accuracy = 0.7750, balanced accuracy = 0.7750, F1 = 0.7800, AUC = 0.8839, при середньому часі 0.5474 с. Для SNR = 18 дБ результати становили accuracy = 0.7833, balanced accuracy = 0.7833, F1 = 0.7716, AUC = 0.8567, а середній час навчання зріс до 2.4799 с. Найсильніший результат зафіксовано при 0 дБ, де модель досягла AUC майже 0.974, що є надзвичайно високим показником для цієї постановки.

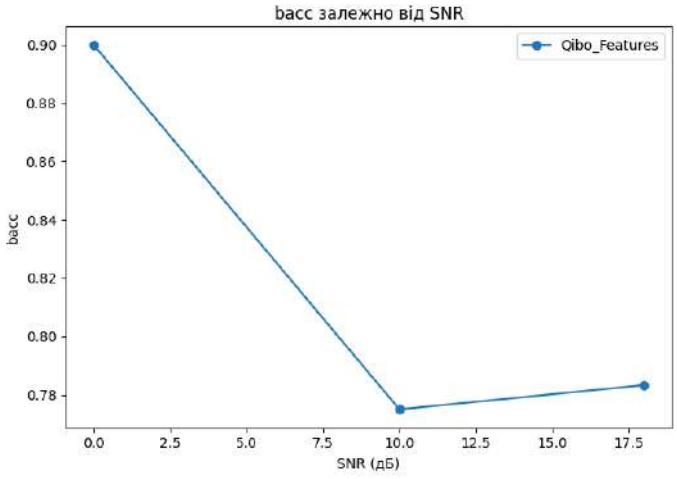


Рис. 2.6 Графік balanced accuracy Qibo

Графік balanced accuracy для Qibo демонструє найвиразнішу перевагу при 0 дБ, де значення досягає 0.9000. При переході до 10 дБ спостерігається зниження до 0.7750, а при 18 дБ показник лише частково відновлюється до

0.7833. Така динаміка свідчить про те, що найбільш інформативне розділення класів у просторі ознак, сформованому за допомогою Qibo, досягається саме при найнижчому з протестованих рівнів SNR. Цей результат узгоджується із загальною картиною, раніше спостереженою для Amazon Braket, і вказує на те, що певні конфігурації статистичних та спектральних ознак можуть краще розділяти обрані класи саме в умовах нижчого співвідношення сигнал/шум, а не в умовах його максимального значення.

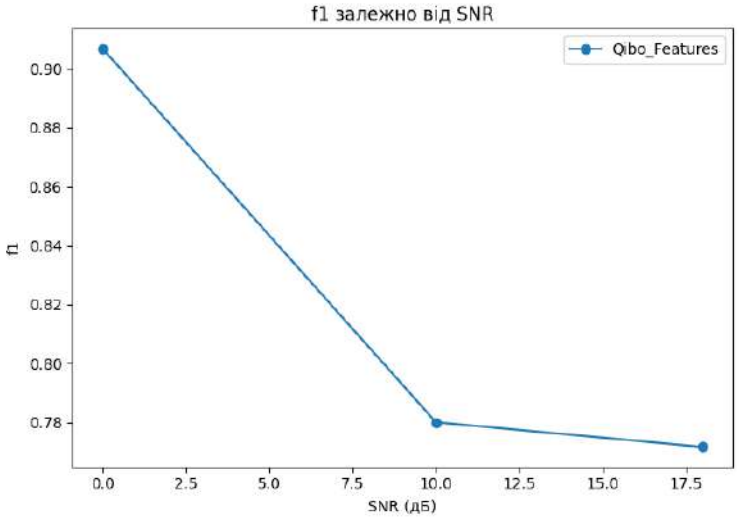


Рис. 2.7 Графік F1 Qibo

Графік F1 для Qibo підтверджує цю тенденцію. Значення 0.9070 при 0 дБ є найкращим серед усіх протестованих конфігурацій цього фреймворку. При 10 дБ F1 зменшується до 0.7800, а при 18 дБ лишається на близькому рівні 0.7716. Це означає, що Qibo особливо добре узгоджує precision і recall саме при 0 дБ, тоді як при вищих SNR баланс між ними послаблюється. У прикладному сенсі це свідчить про те, що простір квантових ознак, побудований у Qibo, найкраще підтримує симетричне розділення класів саме в цій частині тестового діапазону.

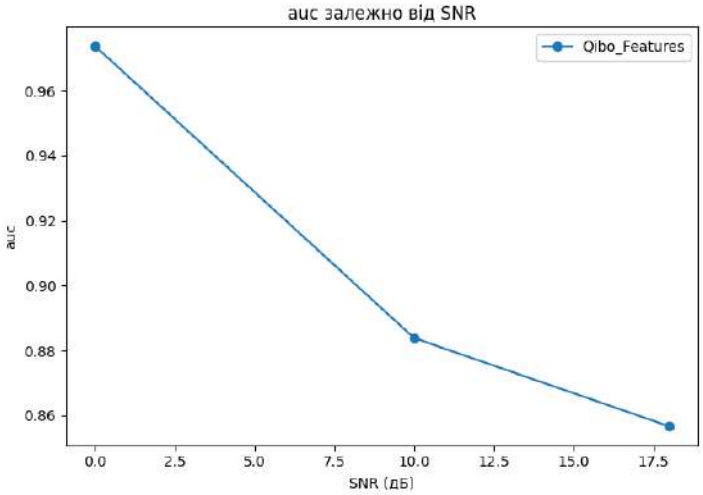


Рис. 2.8 Графік AUC Qibo

Значення AUC = 0.9739 при 0 дБ, 0.8839 при 10 дБ і 0.8567 при 18 дБ свідчать про дуже високу якість ранжування прикладів, особливо на найкращій конфігурації. Такий рівень AUC показує, що навіть якщо порогове рішення класифікатора може бути не абсолютно оптимальним у кожній окремій точці, сама модель формує дуже добре впорядкований простір рішень, у якому приклади різних класів відокремлюються досить чітко.

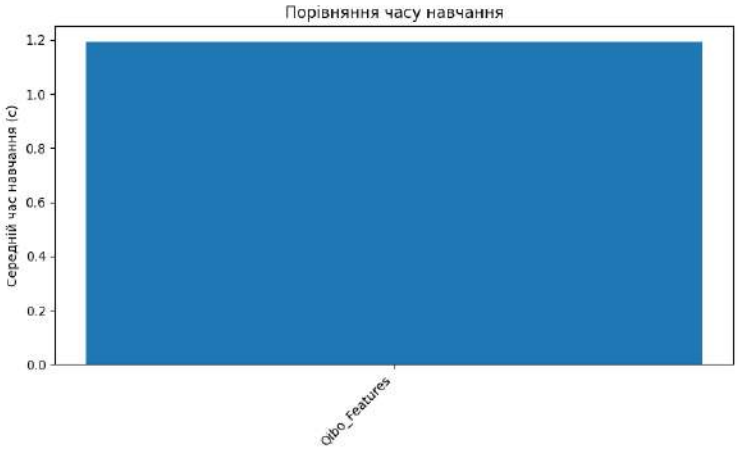


Рис. 2.9 Графік часу навчання Qibo

Графік часу навчання для Qibo виявляє цікаву особливість. Для SNR = 0 дБ і 10 дБ середній час практично однаковий і становить близько 0.55 с, що є дуже сильним результатом з погляду обчислювальної вартості. Однак при 18 дБ середній час різко зростає до 2.48 с. Якщо співставити це з таблицею окремих

запусків, стає видно, що такий ріст зумовлений не систематичним збільшенням складності задачі, а, ймовірно, окремим часовим викидом у одному з запусків. Це означає, що загалом Qibo працює швидко, але часову стабільність у ньому слід аналізувати не лише за середнім значенням, а й з урахуванням дисперсії між окремими повтореннями.

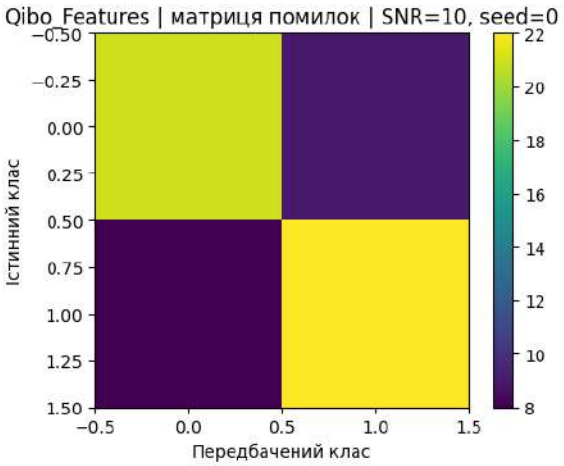


Рис. 2.10 Матриця помилок Qibo

Матриця помилок для запуску SNR = 10 дБ, seed = 0, де було отримано accuracy = 0.7167, balanced accuracy = 0.7167, F1 = 0.7213, AUC = 0.8578, показує, що в цій конкретній конфігурації кількість хибних класифікацій є відчутнішою, ніж у найкращих середніх результатах Qibo. Це важливо, тому що демонструє залежність фреймворку від конкретного розбиття на train/test і підтверджує наявність міжзапускової варіативності. Проте навіть у цьому окремому, не найсильнішому запуску AUC лишається високим, що знову вказує на добру якість побудованого простору ознак, навіть якщо конкретне порогове рішення не є оптимальним.

2.3.3 TorchQuantum

Реалізація TorchQuantum (див. “розділ2_torchquantum.ipynb”) продемонструвала високі результати, особливо при SNR = 0 дБ. Середні значення метрик становили: для 0 дБ accuracy = 0.9000, balanced accuracy = 0.9000, F1 = 0.9035, AUC = 0.9728, при середньому часі навчання 2.1469 с; для

10 дБ accuracy = 0.7750, balanced accuracy = 0.7750, F1 = 0.7765, AUC = 0.8706, при середньому часі 1.8321 с; для 18 дБ accuracy = 0.7500, balanced accuracy = 0.7500, F1 = 0.7371, AUC = 0.8672, при середньому часі 1.9997 с. Отже, TorchQuantum здатний забезпечити помірну обчислювальну вартість. За співвідношенням якості та часу він виглядає сильніше, ніж PennyLane у поточній конфігурації, та наближається до найкращих результатів, отриманих для Amazon Braket і Qibo.

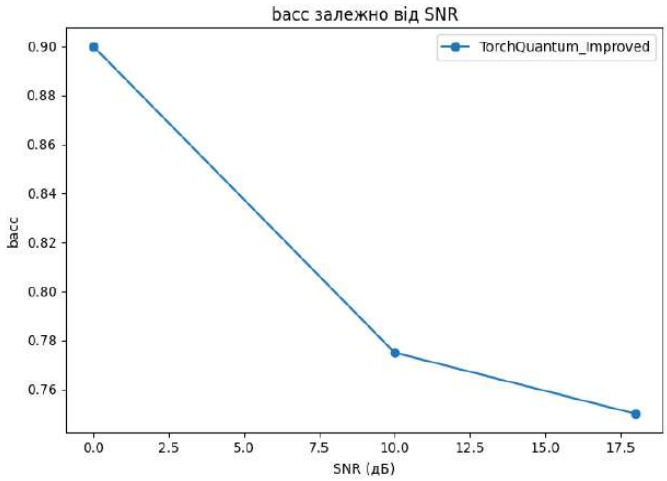


Рис. 2.11 Графік balanced accuracy TorchQuantum

Графік balanced accuracy для TorchQuantum відображає чітко виражену тенденцію спадання якості зі зростанням SNR. Значення 0.9000 при 0 дБ, 0.7750 при 10 дБ і 0.7500 при 18 дБ свідчать про те, що в цій конкретній постановці модель найкраще розділяє класи саме при нижчому рівні сигнал/шум. У загальному вигляді така тенденція не є тривіальною, однак вона вже спостерігалася і для інших гібридних реалізацій на тих самих ознаках. Це дозволяє припустити, що ключову роль тут відіграє не тільки сам фреймворк, а й конфігурація вхідного простору ознак після статистично-спектрального опису та PCA-перетворення. Іншими словами, TorchQuantum не демонструє монотонного погіршення або покращення якості разом із SNR як універсальної закономірності, а відображає специфіку обраної експериментальної схеми.

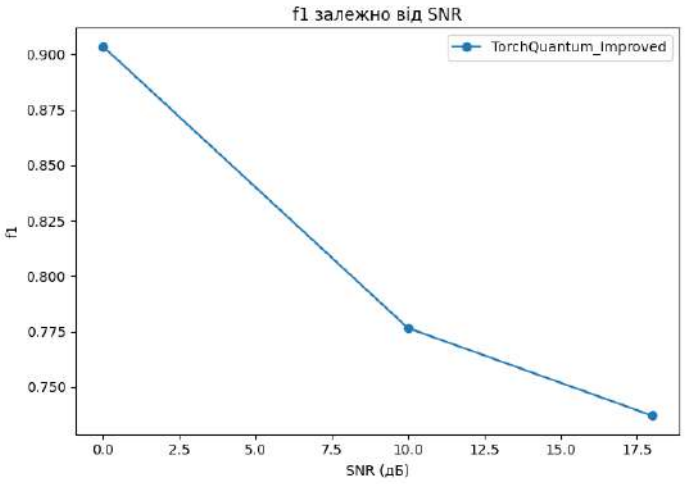


Рис. 2.12 Графік F1 TorchQuantum

Графік F1 для TorchQuantum підтверджує цю картину. Найвище середнє значення F1 = 0.9035 досягається при 0 дБ, після чого показник знижується до 0.7765 при 10 дБ і до 0.7371 при 18 дБ. Це означає, що саме при 0 дБ модель найкраще підтримує баланс між точністю і повнотою щодо позитивного класу. При вищих значеннях SNR баланс стає гіршим, а це свідчить про послаблення роздільної здатності моделі стосовно одного з класів. Водночас навіть мінімальне значення F1 залишається на достатньо високому рівні, що дозволяє вважати модель робочою та інформативною на всьому протестованому діапазоні.

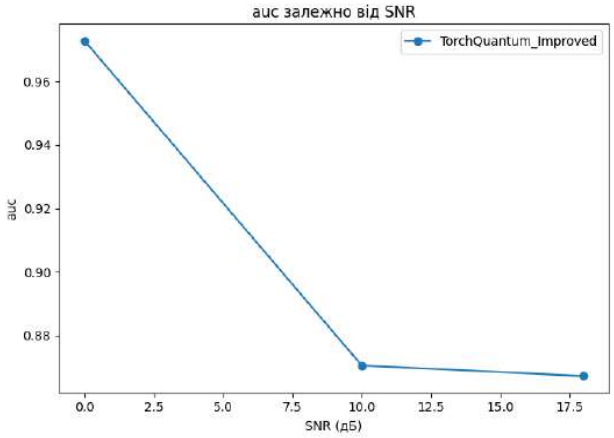


Рис. 2.13 Графік AUC TorchQuantum

Графік AUC є одним із найсильніших аргументів на користь TorchQuantum. Значення AUC = 0.9728 при 0 дБ, 0.8706 при 10 дБ і 0.8672 при 18 дБ свідчать

про дуже добру здатність моделі ранжувати приклади за ймовірністю належності до класу. Особливо важливо, що навіть при 18 дБ AUC залишається на рівні вище 0.86, тобто модель не втрачає дискримінаційної сили у найслабшій з протестованих конфігурацій. З практичного погляду це означає, що квантовий блок у TorchQuantum формує досить якісний ознаковий простір, у якому класи залишаються добре впорядкованими навіть тоді, коли жорстке порогове рішення вже не є оптимальним.

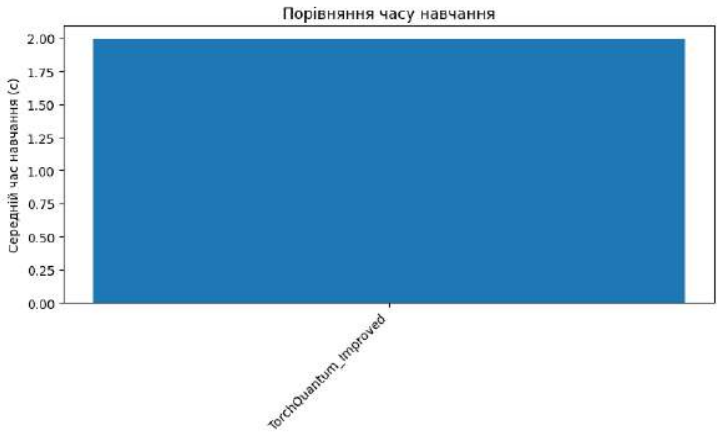


Рис. 2.14 Графік часу навчання TorchQuantum

Графік часу навчання показує, що TorchQuantum потребує в середньому близько двох секунд на запуск. Це суттєво швидше за PennyLane і водночас помітно повільніше за Qibo та більшість запусків Qulacs. Однак різниця не є критичною, оскільки абсолютні значення часу залишаються невеликими. Таким чином, TorchQuantum можна розглядати як компроміс між виразно навчуваною квантово-класичною архітектурою та прийнятною швидкодією.

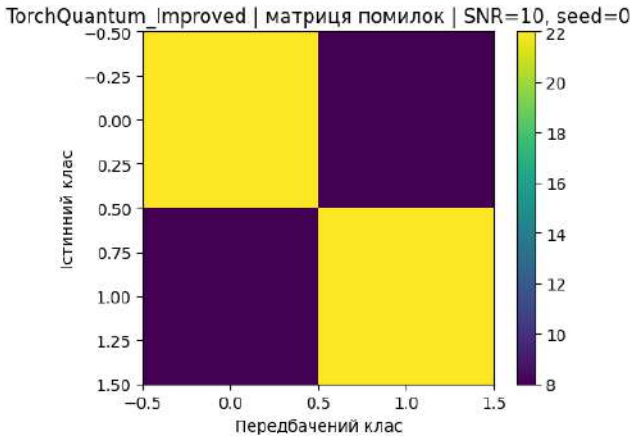


Рис. 2.15 Матриця помилок TorchQuantum

Матриця помилок для запуску SNR = 10 дБ, seed = 0, де було отримано accuracy = 0.7333, balanced accuracy = 0.7333, F1 = 0.7333, AUC = 0.8222, демонструє майже симетричний розподіл правильних і хибних рішень між двома класами. Це означає, що похибка не має вираженого одностороннього зміщення, а пов'язана радше з неповним розділенням класів у побудованому просторі представлень. Відповідно це свідчить, що модель не вироджується в тривіальне віднесення більшості прикладів до одного класу. Наявність помилок при збереженні відносно високого AUC означає, що основна проблема лежить не в самому ранжуванні прикладів, а в межі прийняття рішення.

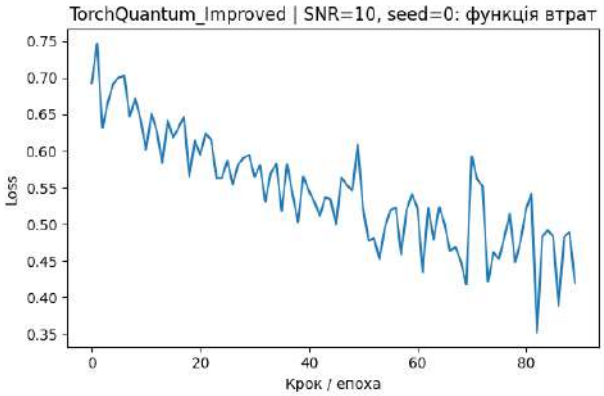


Рис. 2.16 Графік функцій втрат TorchQuantum

Графік функції втрат для TorchQuantum при SNR = 10 дБ і seed = 0 показує загальне зниження loss від приблизно 0.69–0.75 на початкових кроках до

значень близько 0.42–0.50 наприкінці навчання. Динаміка не є абсолютно гладкою, проте тренд зменшення втрат є достатньо чітким. Це вказує на наявність реального процесу навчання, а не на випадкові флуктуації, тому це підтверджує коректну роботу механізму градієнтного оновлення параметрів.

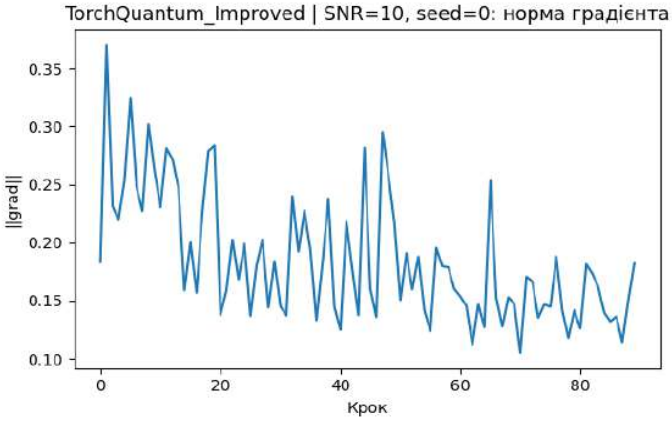


Рис. 2.17 Графік норми градієнта TorchQuantum

Графік норми градієнта також підтверджує стабільність процесу оптимізації. Хоча значення градієнта змінюються нерівномірно й на окремих ітераціях спостерігаються локальні піки, загальний рівень градієнта поступово зменшується, переходячи від приблизно 0.30–0.37 на початкових кроках до значень порядку 0.11–0.18 ближче до завершення навчання. Це означає, що оптимізація не перебуває в режимі числової нестійкості, а наближається до більш стабільної області параметричного простору.

2.3.4 TensorFlow Quantum

Емпіричні результати для TensorFlowQuantum (див. “розділ2_tensorflowquantum.ipynb”) виявилися помітно слабшими, ніж для TorchQuantum, Qibo та Amazon Braket, але при цьому модель продемонструвала працездатність і чітко простежувану динаміку навчання. Середні значення метрик становили: для 0 дБ accuracy = 0.8167, balanced accuracy = 0.8167, F1 = 0.8031, AUC = 0.9000, при середньому часі навчання 4.4446 с; для 10 дБ accuracy = 0.7417, balanced accuracy = 0.7417, F1 = 0.7730, AUC = 0.7650, при середньому часі 3.5989 с; для 18 дБ accuracy = 0.7000,

balanced accuracy = 0.7000, F1 = 0.6905, AUC = 0.7300, а середній час навчання склав 2.0384 с. Ці результати свідчать про те, що TensorFlow Quantum здатний забезпечити робочий рівень класифікації, однак у поточній постановці поступається сильнішим реалізаціям як за якістю, так і за часовою вартістю.

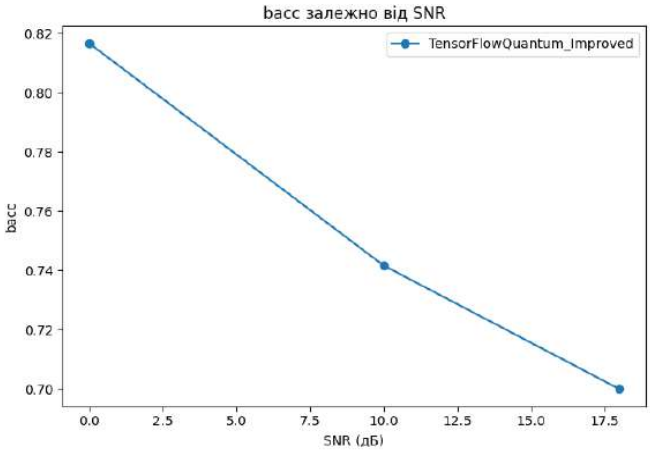


Рис. 2.18 Графік balanced accuracy TensorFlow Quantum

Графік balanced accuracy для TensorFlow Quantum демонструє спадання якості зі зростанням SNR, причому максимум знову спостерігається при 0 дБ. Значення 0.8167 при 0 дБ, 0.7417 при 10 дБ і 0.7000 при 18 дБ показують, що модель здатна досить надійно розділяти класи в найкращій конфігурації, однак при переході до вищих рівнів SNR роздільна здатність зменшується. Порівняно з TorchQuantum і Amazon Braket цей спад є більш вираженим, що дозволяє зробити висновок про нижчу стійкість TFQ-реалізації до зміни умов тестування в межах вибраного простору ознак.

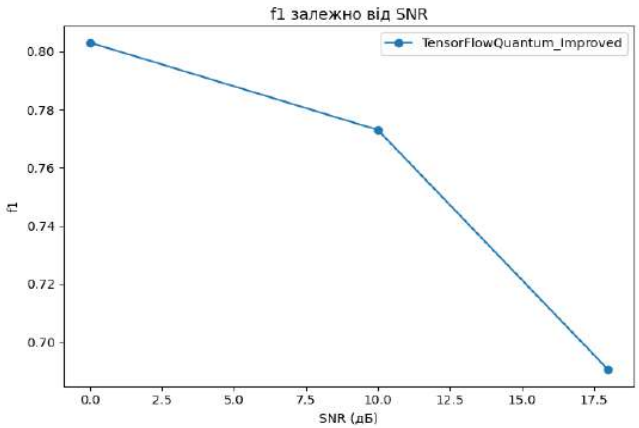


Рис. 2.19 Графік F1 TensorFlow Quantum

Графік F1 відтворює подібну тенденцію. Значення F1 = 0.8031 при 0 дБ, 0.7730 при 10 дБ і 0.6905 при 18 дБ означають, що при зростанні SNR поступово погіршується баланс між точністю і повнотою позитивного класу. Водночас важливо, що навіть при 18 дБ F1 лишається близьким до 0.69, тобто модель не втрачає здатності до змістовної класифікації.

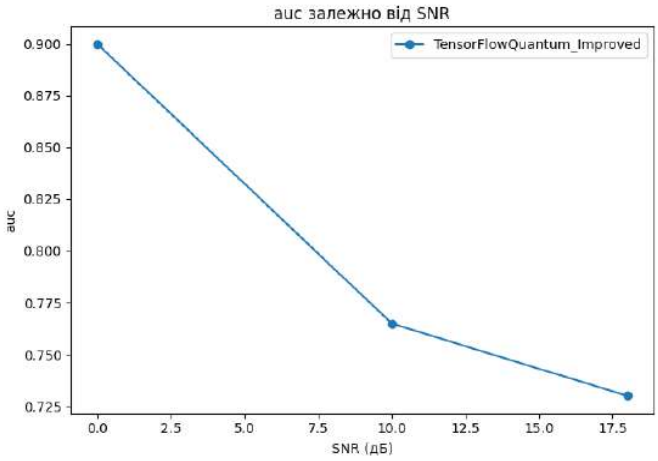


Рис. 2.20 Графік AUC TensorFlow Quantum

Графік AUC для TensorFlow Quantum є більш показовим, ніж самі значення точності. AUC = 0.9000 при 0 дБ свідчить про досить сильну дискримінаційну здатність моделі в найкращій конфігурації. Проте при 10 дБ значення падає до 0.7650, а при 18 дБ до 0.7300. Це означає, що TFQ-реалізація набагато сильніше втрачає якість ранжування прикладів, ніж, наприклад, TorchQuantum чи Qibo. Якщо порівняти саме AUC, який є одним із найбільш інформативних показників у бінарній класифікації, TensorFlow Quantum виявляється менш стабільним у межах усіх протестованих умов.

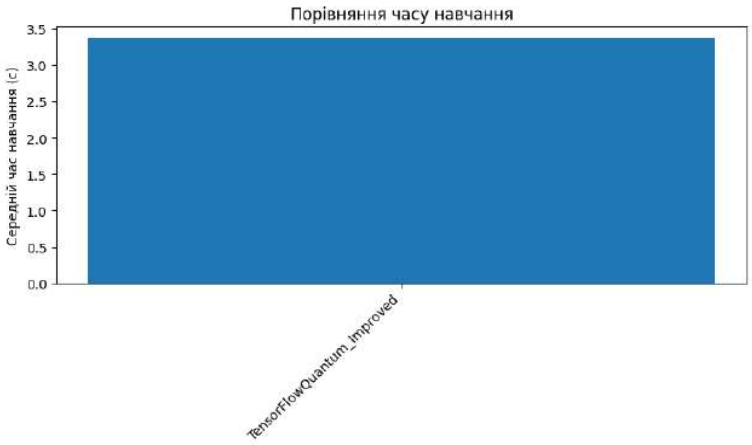


Рис. 2.21 Графік часу навчання TensorFlow Quantum

Графік часу навчання показує, що TensorFlow Quantum є помітно повільнішим, ніж Qibo, Qulacs і TorchQuantum, хоча й швидшим за окремі повільні конфігурації PennyLane. Середній час навчання змінюється від приблизно 2.0 с до 4.4 с. При цьому з таблиці окремих запусків видно, що між різними seed спостерігається відчутна часово-обчислювальна варіативність. Наприклад, для SNR = 10 дБ один запуск тривав близько 1.46 с, а інший понад 5.73 с. Така нестабільність підкреслює чутливість TFQ до деталей конфігурації, що узгоджується з його загальною репутацією як середовища, яке вимагає суворо контрольованої версійної сумісності. Офіційна інсталяційна документація TFQ прямо вказує на вимогу до TensorFlow 2.18.1 і legacy Keras, що саме по собі є показником середовищної вимогливості.

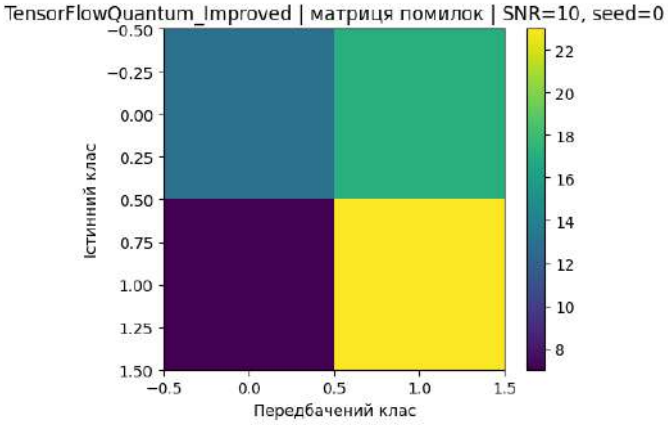


Рис. 2.22 Матриця помилок TensorFlow Quantum

Матриця помилок для SNR = 10 дБ, seed = 0, де було отримано accuracy = 0.6000, balanced accuracy = 0.6000, F1 = 0.6571, AUC = 0.5756, показує, що в цій конкретній конфігурації модель допускає істотну кількість хибних класифікацій. На відміну від більш сильних реалізацій, де навіть окремі запуски демонстрували доволі високу якість, у TFQ розкид між різними запусками є помітнішим. Це означає, що модель є чутливою до конкретного train/test-розбиття і до початкових умов оптимізації.

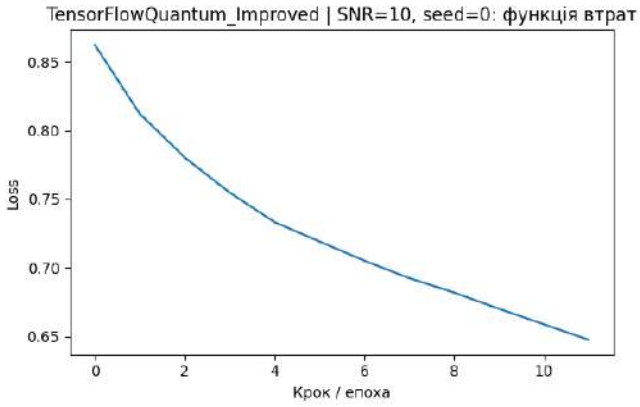


Рис. 2.23 Графік функції втрат TensorFlow Quantum

Графік функції втрат для TensorFlow Quantum, навпаки, є досить акуратним. Він демонструє майже монотонне зменшення loss від приблизно 0.86 до 0.65. Така поведінка свідчить про коректний процес навчання й відсутність очевидних числових вибухів або деградації оптимізації. Тобто, модель навчається формально правильно, але не досягає сильного розділення класів, як інші фреймворки в цій самій постановці.

2.3.5 Amazon Braket SDK

Модель працювала стабільно на всіх трьох значеннях SNR, а саме 0, 10 і 18 дБ, причому для кожного рівня шуму було виконано два незалежні запуски з різними seed (див. “розділ2_amazon.ipynb”). При SNR = 18 дБ одержано результати accuracy = 0.6333 і 0.7000, F1 = 0.6207 і 0.6538, AUC = 0.7278 і 0.8044. При SNR = 10 дБ результати стали вищими, а саме accuracy = 0.8167 і 0.8000, F1 = 0.8070 і 0.7857, AUC = 0.9111 і 0.8922. Найкращі показники були

зафіксовані при SNR = 0 дБ, де окремі запуски дали accuracy = 0.9167 і 0.8167, F1 = 0.9231 і 0.8308, AUC = 0.9789 і 0.9256.

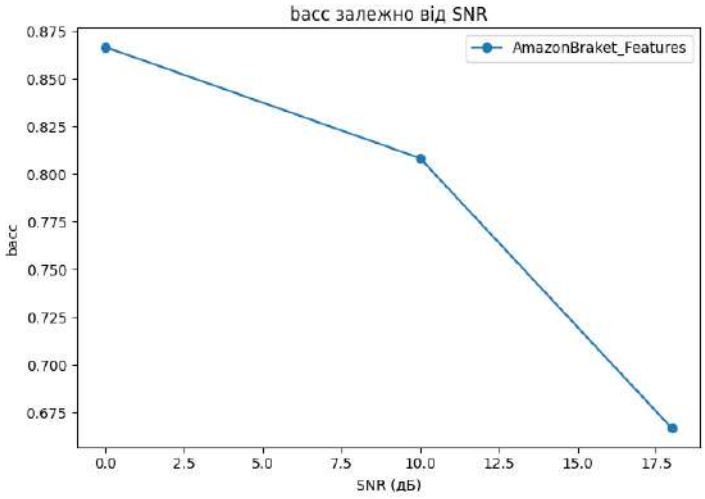


Рис. 2.24 Графік balanced accuracy Amazon Braket SDK

Графік balanced accuracy залежно від SNR демонструє чітко виражену спадну тенденцію. Найвище значення спостерігається при 0 дБ, де bacc = 0.8667. При 10 дБ якість зменшується до 0.8083, а при 18 дБ падає до 0.6667. Формально це означає, що найкраще розділення класів у побудованому квантовому просторі ознак було досягнуте саме в конфігурації з найнижчим із протестованих рівнів SNR. Подібна динаміка не повинна тлумачитися як універсальна властивість самого фреймворку. Набагато коректніше пов’язувати її з конкретною структурою вхідного простору ознак, набором статистичних і спектральних дескрипторів, а також із проєкцією PCA, яка залишила в моделі лише вісім компонент. Іншими словами, графік balanced accuracy у цьому випадку описує не загальну фізичну закономірність, а саме поведінку моделі в заданій експериментальній постановці.

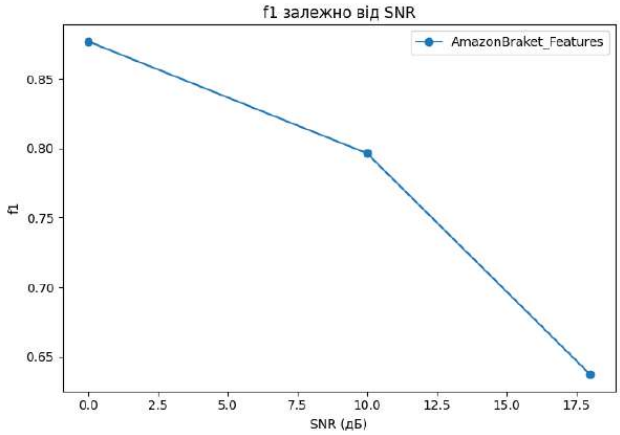


Рис. 2.25 Графік F1 Amazon Braket SDK

Графік F1 підтверджує ту саму загальну тенденцію, але дозволяє глибше оцінити баланс між точністю і повнотою щодо позитивного класу. Значення F1 = 0.8769 при 0 дБ, 0.7964 при 10 дБ і 0.6373 при 18 дБ показують, що модель найкраще узгоджує precision та recall в успішній конфігурації, тоді як зі зростанням SNR баланс поступово погіршується. Особливо показовим є значення при 18 дБ, де F1 уже знижується до рівня, що свідчить про істотне перекриття між класами. Проте навіть у цій точці значення лишається достатньо високим, щоб модель можна було вважати працездатною та інформативною.

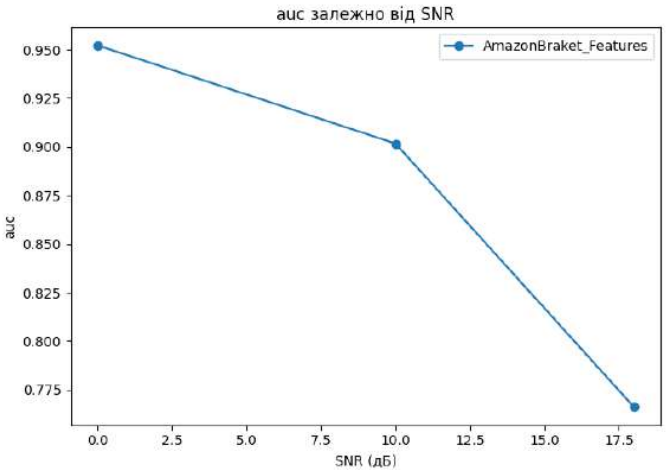


Рис. 2.26 Графік AUC Amazon Braket SDK

Значення AUC = 0.9522 при 0 дБ, 0.9017 при 10 дБ і 0.7661 при 18 дБ означають, що модель дуже добре ранжує приклади за ступенем належності

до цільового класу, особливо в двох перших конфігураціях. Рівень AUC понад 0.95 при 0 дБ є дуже сильним результатом, тому що він свідчить не просто про коректне порогове рішення, а про формування справді інформативного простору рішень. Навіть при 18 дБ, де спостерігається найгірший середній результат, AUC залишається істотно вищим за випадковий рівень. Це означає, що дискримінаційна здатність моделі не руйнується повністю, хоча її сила вже помітно зменшується.

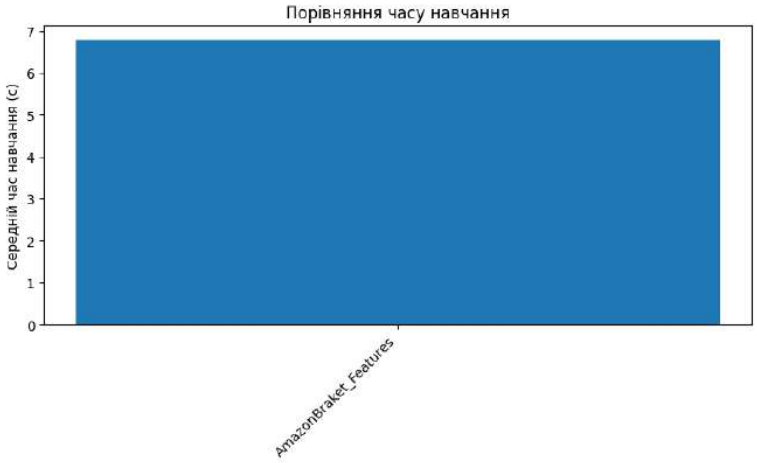


Рис. 2.27 Графік часу навчання Amazon Braket SDK

Час навчання виявився майже незмінним для всіх трьох протестованих конфігурацій. Усереднені значення становили 6.7378 с при 0 дБ, 6.8497 с при 10 дБ і 6.7594 с при 18 дБ. Така стабільність є принципово важливою, тому що вона свідчить про добру передбачуваність обчислювальної вартості. Інакше кажучи, зміна характеристик сигналу не призводила до суттєвого збільшення часу виконання. У серійному бенчмаркінгу це є вагомою перевагою, оскільки дозволяє планувати повторні експерименти без ризику різкого росту витрат часу на окремих конфігураціях.

Особливо показовим є окремий запуск при SNR = 10 дБ, seed = 0, для якого було отримано accuracy = 0.8167, balanced accuracy = 0.8167, F1 = 0.8070, AUC = 0.9111, train_time_s = 7.1570. Цей запуск є репрезентативним, оскільки він

не належить до ні найсильнішої, ні найслабшої конфігурації, але все одно демонструє високу якість класифікації.

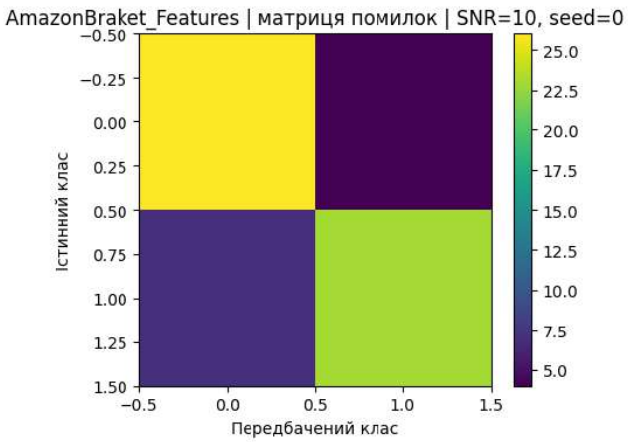


Рис. 2.28 Матриця помилок Amazon Braket SDK

Матриця помилок для цього випадку показує, що більшість прикладів обох класів було класифіковано правильно. Водночас певна кількість помилок зберігається, причому їхня структура не є повністю хаотичною. Це означає, що між класами лишається зона часткового перекриття, однак побудований квантовий простір ознак уже є достатньо виразним для надійного розділення більшості прикладів.

З погляду інтерпретації матриця помилок важлива ще й тому, що вона підтверджує відсутність тривіального сценарію, за якого модель штучно завищує якість за рахунок домінування одного класу. Наявне співвідношення правильних і помилкових передбачень свідчить про відносно симетричну поведінку моделі щодо обох класів. Це добре узгоджується з тим, що balanced accuracy та accuracy збігаються, а AUC перевищує 0.9. Тобто модель не лише формує коректний поріг класифікації, а й загалом будує добре впорядкований простір рішень.

2.3.6 PennyLane

У проведеному експерименті реалізація PennyLane (див. “розділ2_pennylane.ipynb”) показала змішані результати. Середні значення для

SNR = 0 дБ становили accuracy = 0.7833, balanced accuracy = 0.7833, F1 = 0.8042, AUC = 0.8933, при середньому часі навчання 64.0749 с. Для SNR = 10 дБ було отримано accuracy = 0.5583, balanced accuracy = 0.5583, F1 = 0.4305, AUC = 0.6872, а середній час навчання склав 64.0089 с. Для SNR = 18 дБ значення дорівнювали accuracy = 0.6750, balanced accuracy = 0.6750, F1 = 0.6042, AUC = 0.7811, при середньому часі навчання 64.2022 с. Такі результати свідчать, що PennyLane у поточній постановці здатний досягати змістовної якості класифікації, але його ефективність суттєво залежить від конкретної конфігурації SNR, а обчислювальна вартість залишається значно вищою, ніж у більшості інших протестованих фреймворків.

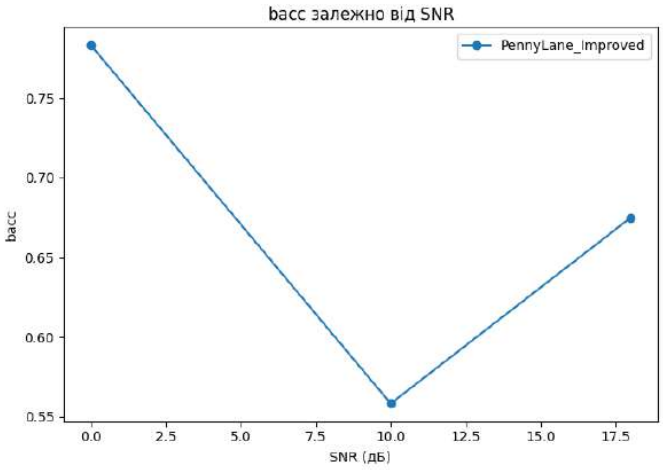


Рис. 2.29 Графік balanced accuracy PennyLane

Графік balanced accuracy для PennyLane показує різко немонотонну залежність від SNR. При 0 дБ модель досягає 0.7833, при 10 дБ значення падає до 0.5583, після чого при 18 дБ частково відновлюється до 0.6750. Така форма кривої означає, що модель не демонструє стабільного узагальнення на всьому діапазоні тестових умов. Найслабшою виявляється саме конфігурація 10 дБ, де класифікаційна здатність майже наближається до межі слабко інформативного рішення. Це є важливим результатом, оскільки показує, що сама наявність диференційовної квантової схеми ще не гарантує високої стійкості до змін у структурі вхідних даних.

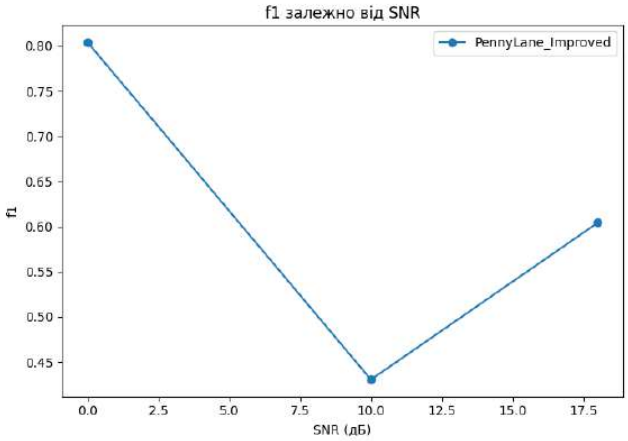


Рис. 2.30 Графік F1 PennyLane

Графік F1 підсилює цю інтерпретацію. Значення F1 = 0.8042 при 0 дБ, 0.4305 при 10 дБ і 0.6042 при 18 дБ свідчать, що найбільші труднощі модель має саме з підтриманням балансу між точністю і повнотою на проміжному рівні SNR. Падіння до 0.4305 є істотним, оскільки воно означає не просто незначне послаблення класифікації, а фактичну втрату стійкого балансу між правильними виявленнями позитивного класу і хибними рішеннями. Така поведінка добре узгоджується з таблицею окремих запусків, де один із запусків при SNR = 10 дБ показав F1 = 0.2727, що є найслабшим індивідуальним результатом у наборі.

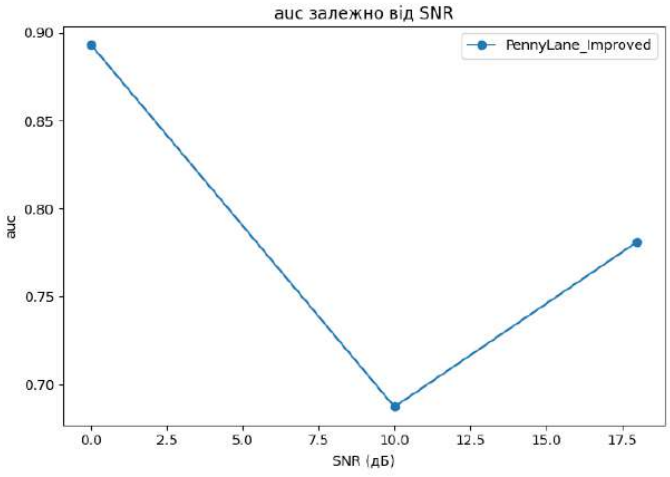


Рис. 2.31 Графік AUC PennyLane

Графік AUC для PennyLane демонструє ту саму загальну тенденцію, але у формі, що краще відображає якість ранжування прикладів. Значення AUC =

0.8933 при 0 дБ свідчить про досить сильну дискримінаційну здатність моделі в найкращій конфігурації. При 10 дБ AUC падає до 0.6872, а при 18 дБ частково зростає до 0.7811. Це означає, що навіть у найслабшій точці модель не повністю втрачає здатність розрізняти класи, однак якість побудованого простору рішень стає помітно нижчою, ніж у Qibo, Amazon Braket або TorchQuantum. Саме AUC тут показує, що головна проблема полягає не тільки у виборі порога рішення, а й у загальному ослабленні роздільної здатності моделі при певних умовах.

Рис. 2.32 Графік часу навчання PennyLane

Середній час стабільно тримається на рівні приблизно 64 с, причому ця величина майже не змінюється між різними значеннями SNR. Це означає, що обчислювальна вартість визначається не специфікою окремого піднабору сигналів, а самою природою реалізації. Фактично PennyLane в цій конфігурації є на порядок повільнішим за Qibo, Qulacs, TorchQuantum і Amazon Braket. Для багаторазового серійного бенчмаркінгу це є істотним обмеженням, оскільки навіть за відносно невеликих вибірок сумарна вартість експериментів швидко зростає.

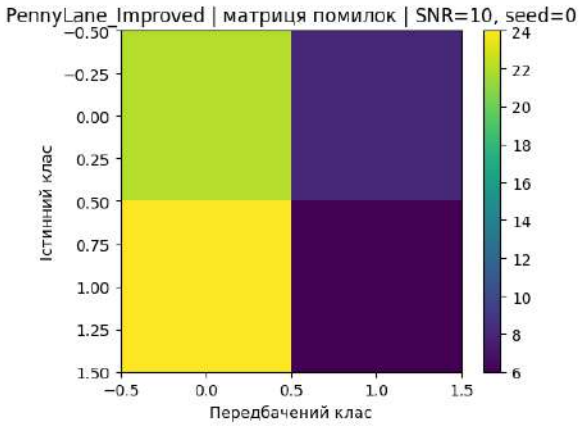


Рис. 2.33 Матриця помилок PennyLane

Матриця помилок для запуску SNR = 10 дБ, seed = 0, де було отримано accuracy = 0.4667, balanced accuracy = 0.4667, F1 = 0.2727, AUC = 0.5911, показує, що модель у цій конфігурації класифікує дані нестабільно й допускає значну кількість систематичних помилок. Структура матриці свідчить про те, що один із класів розпізнається значно гірше, ніж інший. Це пояснює різке падіння F1 і підтверджує, що найслабший режим роботи моделі пов'язаний не тільки з випадковими коливаннями, а з реальною деградацією якості класифікаційної межі.

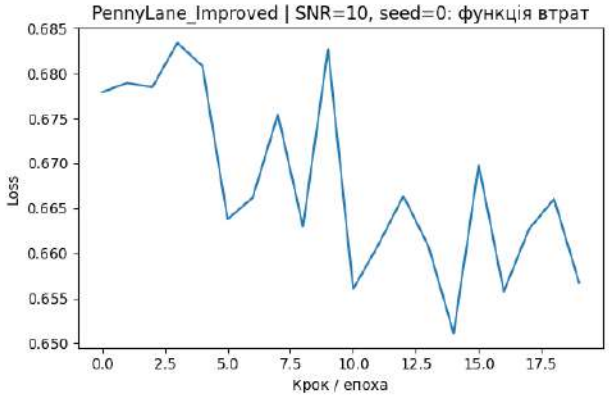


Рис. 2.34 Графік функції втрат PennyLane

Графік функції втрат при SNR = 10 дБ, seed = 0 демонструє помірне зниження loss, але без вираженого гладкого монотонного тренду. Значення коливаються приблизно в межах від 0.651 до 0.683, а амплітуда коливань лишається помітною протягом усього навчання. Це означає, що оптимізація не є

катастрофічно нестабільною, однак не входить у чіткий режим впевненої збіжності. На відміну від TorchQuantum або TensorFlow Quantum, де функція втрат зменшувалася більш послідовно, у PennyLane спостерігається відчутно більш нерівна траєкторія оптимізації.

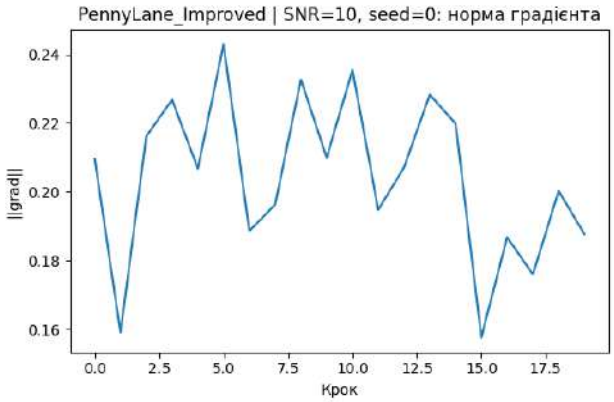


Рис. 2.35 Графік норми градієнта PennyLane

Графік норми градієнта також відображає цю особливість. Значення ||grad|| коливаються приблизно від 0.157 до 0.243 без стійкої тенденції до зменшення. Це означає, що модель не входить у режим різкого затухання градієнтів, але і не демонструє впорядкованого зближення до стабільної області параметричного простору. З одного боку, це свідчить про відсутність повного градієнтного виродження. З іншого боку, така поведінка узгоджується з високою часовою вартістю та нестабільною якістю на окремих підмножинах даних.

Узагальнюючи, PennyLane у цьому експерименті варто розглядати як фреймворк із високою методологічною цінністю, але не як лідера за сукупністю емпіричних характеристик. Його головна перевага полягає в тому, що він надає найбільш прозорий доступ до аналізу trainability, функції втрат, градієнтів і загальної поведінки варіаційної квантової моделі. Саме це робить його дуже корисним для дослідження внутрішньої динаміки навчання. Однак за співвідношенням якості класифікації, стабільності між різними SNR і часу виконання PennyLane поступається сильнішим практичним кандидатам.

Документація PennyLane прямо підкреслює сильні сторони автоматичного диференціювання, гібридних моделей і device-independent execution, однак отримані результати показують, що в прикладному сценарії класифікації радіосигналів цих архітектурних переваг недостатньо для виходу в лідери без додаткової оптимізації моделі.

2.4 Обґрунтування вибору інструментів та середовищ

Якщо узагальнити результати серії запусків, найбільш переконливу групу утворюють Qibo, TorchQuantum та Amazon Braket. Qibo виявився найсильнішим за інтегральним поєднанням bacc, AUC і середнього часу навчання. TorchQuantum продемонстрував майже таку саму силу за AUC, при цьому залишаючись повноцінною гібридною архітектурою, що підтримує налаштування від початку до кінця. Amazon Braket поступився їм за швидкодією, але зберіг дуже високі метрики якості й водночас репрезентує окремий клас інфраструктурно орієнтованих квантових середовищ з можливістю подальшого переходу до hybrid jobs та керованого виконання.

Qulacs, попри дуже сильну швидкодію та стабільно високий AUC, був віднесений до резервної групи. Причина полягає не в слабкості самого інструмента, а в тому, що при виборі лише трьох фреймворків необхідно було поєднати дві логіки. Перша логіка стосується максимальної якості та обчислювальної ефективності. Друга стосується архітектурної репрезентативності. Якщо залишити тільки найшвидші симуляторні середовища, вибір виявиться надто однорідним й не покаже різниці між end-to-end тренованою моделлю, гібридною моделлю на основі вилучення ознак та інфраструктурним хмарним workflow.

TensorFlow Quantum та PennyLane мають високий методологічний статус, але за сукупністю показників не входять до ядра подальшого етапу. TensorFlow Quantum важливий тому, що він репрезентує інтеграцію квантових шарів у TensorFlow-граф та дає змогу аналізувати квантові моделі в межах звичного

tf.keras pipeline. Проте його середовищна вимогливість, залежність від конкретної версії TensorFlow та помітно слабші результати за якістю та стабільністю часу запуску знижують його пріоритет. PennyLane, своєю чергою, залишається одним з найцінніших фреймворків для аналізу диференційовного квантового програмування, але у поточній постановці його надзвичайно висока часова вартість не компенсується відповідним приростом якості.

Практична серія запусків показала, що для частини фреймворків спільне середовище є джерелом систематичних конфліктів залежностей. TensorFlow Quantum потребує окремої сумісної конфігурації з TensorFlow 2.18.1 і legacy Keras. TorchQuantum вимагає правильного editable install і коректного шляху до пакета. Amazon Braket, попри стабільну роботу після ізоляції, змінює залежності наукового стеку таким чином, що сумісне виконання з іншими системами в одному runtime стає ненадійним. З цього випливає, що коректне порівняння QML-фреймворків потребує не універсального ноутбука для всіх платформ, а набору ізольованих середовищ із максимально контрольованими залежностями. Це є не технічною деталлю, а частиною методологічної вимоги до відтворюваності.

Щодо самого набору даних, використання RadioML 2016.10A і стандартизованої схеми попередньої обробки є ще одним елементом обґрунтування вибору інструментів. Оскільки всі фреймворки працювали з одним й тим самим підпростором ознак, з однаковою кількістю компонент після PCA, з однаковими SNR і зі збалансованою вибіркою однакового обсягу, відмінності між результатами не можна пояснювати різницею у вхідних даних.

РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ЗАДАЧІ ОБРОБКИ РАДІОСИГНАЛІВ

3.1 Постановка задачі

Для виконання наступного етапу була взята задача автоматичної класифікації типу модуляції радіосигналу на основі відліків комплексної огинаючої, поданих у вигляді пар I/Q. Дана задача належить до класу automatic modulation classification, які використовуються в інтелектуальних бездротових системах, оскільки дозволяють визначати тип модуляції без попереднього знання параметрів передавача і, таким чином, підтримують сценарії когнітивного радіо, спектрального моніторингу та адаптивного керування каналом зв'язку. AMC вважається однією з базових задач фізичного рівня в системах, де необхідно приймати рішення щодо характеру сигналу в умовах неповної або відсутньої апріорної інформації [30].

В межах роботи сформульовано багатокласову задачу розпізнавання чотирьох типів модуляці, а саме BPSK, QPSK, 8PSK та QAM16. Вибір саме цих класів зумовлений тим, що вони охоплюють як відносно прості схеми модуляції, так і більш складні фазові та фазово-амплітудні формати, які є показовими для перевірки здатності моделі розрізняти сигнали зі схожими структурними характеристиками. Додатково задача аналізувалася за трьома рівнями відношення сигнал/шум, а саме 0, 10 та 18 ДБ. Тому це дозволило оцінити поведінку моделей як у складних шумових умовах, так і в режимах з вищою якістю сигналу. Далі було побудувано та порівняно дві квантово-класичні моделі обробки сигналів, реалізовані у фреймворках TorchQuantum та Qibo (див. "розділ3_torchquantum_finetuned.ipynb" та "розділ3_qibo_finetuned.ipynb"), за однакових умов попередньої підготовки даних, навчання й оцінювання. Вибір даних фреймворків випливає з розділу 2, де попередньо був проведений порівняльний аналіз, в якому дані фреймворки було визначено як найкращі.

3.2 Попередня підготовка вхідних даних

Для попередньої підготовки вхідних даних використано раніше зазначений датасет RadioML 2016.10A, який завантажувався через kagglehub у вигляді файлу RML2016.10a_dict.pkl. У ноутбуці цей файл зчитується як Python-словник, де ключем виступає пара (тип модуляції, SNR), а значенням є масив сигналів. Перевірка структури показала, що датасет містить 220 ключів, 11 типів модуляції та 20 рівнів SNR у діапазоні від -20 до 18 дБ. Для кожної комбінації модуляції та SNR сигнали були представлені масивами форми (1000, 2, 128), тобто кожен приклад складався з двох компонентів I та Q й мав довжину 128 відліків [31].

Було сформовано окрему підвибірку з чотирьох вищезгаданих модуляцій, також додатково було обмежено набір рівнів шуму трьома значеннями, а саме 0, 10 та 18 дБ. Для кожної пари (клас, SNR) з доступних 1000 прикладів випадково відбиралося по 120 сигналів. У результаті вийшла збалансована вибірка розміром 1440 прикладів, тобто по 360 прикладів на кожен із чотирьох класів й по 480 прикладів на кожне з використаних рівнів SNR. Після формування підвибірки сирі сигнали не подавалися безпосередньо в модель, замість цього для кожного прикладу обчислювався набір числових ознак, який стисло описував часові, фазові та спектральні властивості сигналу (функція extract_signal_features). Спочатку з двох каналів I та Q формувався комплексний сигнал $s = I+jQ$. Далі на його основі обчислювалися амплітуда, фаза та миттєва частота. Після цього будувався розширений вектор ознак, який включав середні значення і стандартні відхилення для I та Q, характеристики амплітуди, середнє та стандартне відхилення фази, середнє та стандартне відхилення миттєвої частоти, коефіцієнти асиметрії та ексцесу, спектральну ентропію, автокореляційні характеристики, а також додаткові величини, пов’язані з потужністю та взаємозв’язком сусідніх комплексних відліків.

Внаслідок цього для кожного сигналу формувався вектор з 50 числових ознак, а для всієї підвибірки було отримано матрицю ознак розмірності (1440, 50).

Наступним кроком було розбиття вибірки на навчальну, валідаційну та тестову частини. Щоб зберегти однакове співвідношення класів та рівнів SNR у всіх частинах вибірки, використовувалася стратифікація за об’єднаною міткою виду class_id + SNR (функція make_joint_strat_labels). Спочатку з усієї вибірки відокремлювалася тестова частина обсягом 15%, після чого з решти даних виділялася валідаційна підмножина еквівалентного масштабу. У результаті було сформовано три підмножини: навчальну (1008, 50), валідаційну (216, 50) і тестову (216, 50). Це забезпечило коректний режим оцінювання, у якому навчання, налаштування гіперпараметрів та фінальне тестування здійснювалися на взаємно відокремлених наборах даних. Оскільки отримані ознаки мали різні масштаби й різну дисперсію, перед подальшою обробкою використовувалася стандартизація за допомогою StandardScaler. Скейлер навчався лише на тренувальній підмножині, після чого ті самі параметри застосовувалися до валідаційних та тестових даних. Повний стандартизований простір ознак одразу зберігався як окреме класичне представлення сигналу, а паралельно до нього застосовувався метод головних компонентів PCA, зменшуючи розмірність вхідного простору до 8 компонентів. Це скорочення було пов'язане з кількістю кубітів у квантовій частині моделей, оскільки обидві фінальні реалізації працювали на восьмикубітній схемі. У результаті для квантового блоку було сформовано матриці X_train, X_val Та X_test розмірності (1008, 8), (216, 8) і (216, 8) відповідно, тоді як повний стандартизований простір ознак використовувався як додатковий класичний канал у гібридній архітектурі. Крім того, у підсумкове представлення було включено ще нормалізоване значення SNR. Це дало змогу моделі враховувати шумовий режим як частину вхідного опису сигналу, не змінюючи при цьому постановки задачі. Для моделі на базі

TorchQuantum дані були додатково обгорнуті у власний клас AMCDataset та подавалися в мережу через DataLoader із фіксованим розміром пакета 32. Для моделі Qibo ті самі підготовлені дані використовувалися у вигляді матриць ознак для побудови гібридного представлення.

3.3 Реалізація моделей обробки сигналів

Як зазначалося вище, дві моделі обробки радіосигналів побудовані на двох різних фреймворках QML. Перша модель, TorchQuantumFineTuned, реалізовувалася у середовищі TorchQuantum, яке підтримує побудову гібридних квантово-класичних моделей у стилі PyTorch, пакетне навчання, автоматичне диференціювання та інтеграцію квантової частини в загальний процес оптимізації [32]. Друга модель, QiboFineTuned, реалізовувалася за допомогою Qibo, який надає засоби для моделювання параметризованих квантових схем і побудови гібридних workflow на основі квантового екстрактора ознак та класичної голови [33].

Модель TorchQuantumFineTuned була побудована як end-to-end гібридна квантово-класична архітектура. У цій реалізації восьмивимірний вектор після PCA подавався на квантову частину після попередньої нормалізації та нелінійного перетворення. Далі використовувалася восьмикубітна схема з двома повторними введеннями даних й трьома параметризованими квантовими шарами. Кожен шар містив кільцеву схему заплутування через слот та параметризовані обертання RX, RY, RZ. Вимірювання операторів Паулі на всіх кубітах формували восьмивимірний квантовий вектор. Після цього він конкатенувався з вектором після PCA, повиним стандартизованим вектором ручних ознак та нормалізованим значенням SNR. Отримане гібридне представлення надходило до класичної голови, яка складалася з двох повнозв'язних шарів розмірності 64 і 32, функцій активації ReLU, шару Dropout та фінального класифікаційного шару на чотири класи. Навчання моделі здійснювалося за допомогою функції втрат CrossEntropyLoss,

оптимізатора AdamW, адаптивного scheduler типу ReduceLROnPlateau, а також з використанням gradient clipping. Раннє зупинення контролювалося за валідаційним macro-F1.

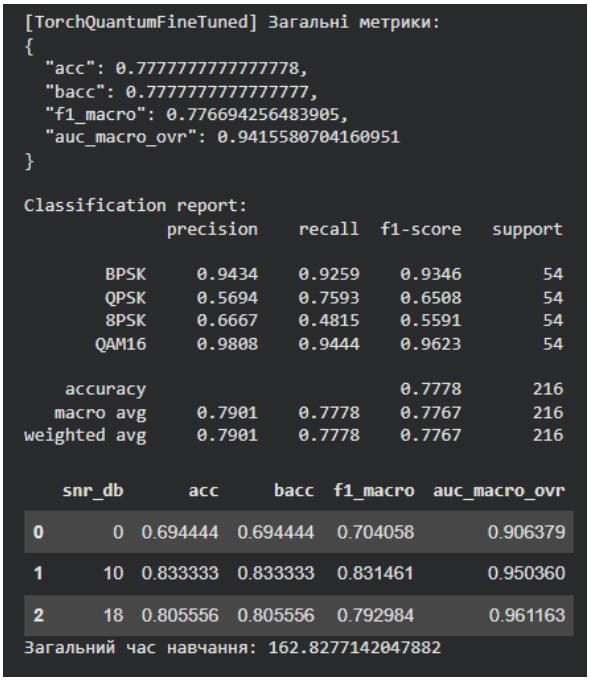
Модель QiboFineTuned мала іншу архітектурну логіку та була реалізована як гібридна модель на основі вилучення ознак. У цій схемі квантова частина не навчалася end-to-end разом з класифікатором, а використовувалася як параметризований екстрактор ознак. На вхід восьмикубітної квантової схеми подавалися компоненти вектора після PCA, які кодувалися через RY-embedding. Далі використовувалися три параметризовані шари з кільцевою схемою заплутування через CNOT і параметрами RX, RY, RZ. Після виконання схеми обчислювалися очікувані значення операторів z на всіх кубітах, що формувало квантовий вектор ознак розмірності 8. Цей вектор додатково стандартизувався та конкатенувався з повним стандартизованим простором ручних ознак, вектором після PCA та нормалізованим SNR. Фінальна класифікація, на відміну від моделі TorchQuantumFineTuned, виконувалася багатошаровим перцептроном MLPClassifier з двома прихованими шарами розмірності 128 та 64. Навчання класичної голови також контролювалося за валідаційним macro-F1, а зупинка відбувалася за early stopping.

У результаті обидві моделі отримували на вхід однакову інформацію про сигнал, але використовували різні принципи інтеграції квантової частини в загальну систему класифікації. У першому випадку квантовий блок був повністю включений у процес навчання параметрів через backpropagation, у другому випадку квантова схема працювала як генератор додаткового представлення сигналу, на основі якого вже навчалася класична модель.

3.4 Порівняльний аналіз отриманих результатів

За підсумками внутрішнього тестування обидві моделі продемонстрували позитивні результати, однак з QiboFineTuned фінальні показники виявилися

дещо вищими за сукупністю основних. Для моделі TorchQuantumFineTuned на тестовій вибірці було отримано accuracy=0.7778, balanced accuracy=0.7778, macro-F1=0.7767, macro-AUC(OVR)=0.9416. Загальний час навчання цієї моделі становив 162.83 с.



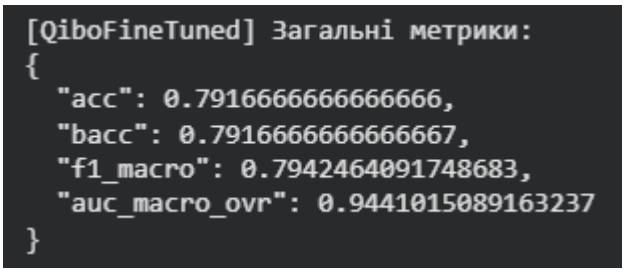
```
[TorchQuantumFineTuned] Загальні метрики:
{
  "acc": 0.7777777777777778,
  "bacc": 0.7777777777777777,
  "f1_macro": 0.776694256483905,
  "auc_macro_ovr": 0.9415580704160951
}

Classification report:
              precision    recall  f1-score   support

        BPSK     0.9434    0.9259    0.9346        54
        QPSK     0.5694    0.7593    0.6508        54
        8PSK     0.6667    0.4815    0.5591        54
       QAM16     0.9808    0.9444    0.9623        54

    accuracy                         0.7778       216
   macro avg     0.7901    0.7778    0.7767       216
weighted avg     0.7901    0.7778    0.7767       216
```

	snr_db	acc	bacc	f1_macro	auc_macro_ovr
0	0	0.694444	0.694444	0.704058	0.906379
1	10	0.833333	0.833333	0.831461	0.950360
2	18	0.805556	0.805556	0.792984	0.961163

```
Загальний час навчання: 162.8277142047882
```

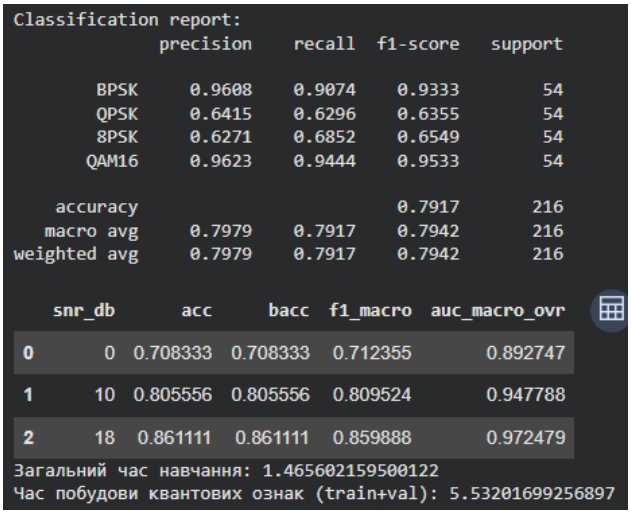
Рис. 3.1 Класифікаційний рапорт моделі TorchQuantumFineTuned

Для QiboFineTuned відповідні значення дорівнювали accuracy=0.7917, balanced accuracy=0.7917, macro-F1=0.7942 за підсумками класифікаційного звіту, macro-AUC(OVR)=0.9441. Час навчання класичної голови становив 1.47 с, а побудова квантових ознак для навчальної та валідаційної частин вимагала 5.53 с. Навіть з урахуванням цього етапу повний час роботи моделі QiboFineTuned залишився істотно меншим, ніж у TorchQuantumFineTuned.



```
[QiboFineTuned] Загальні метрики:
{
  "acc": 0.7916666666666666,
  "bacc": 0.7916666666666667,
  "f1_macro": 0.7942464091748683,
  "auc_macro_ovr": 0.9441015089163237
}
```

Рис. 3.2 Загальні метрики моделі QiboFineTuned



```
Classification report:
              precision    recall  f1-score   support

        BPSK     0.9608    0.9074    0.9333        54
        QPSK     0.6415    0.6296    0.6355        54
        8PSK     0.6271    0.6852    0.6549        54
       QAM16     0.9623    0.9444    0.9533        54

    accuracy                         0.7917       216
   macro avg     0.7979    0.7917    0.7942       216
weighted avg     0.7979    0.7917    0.7942       216
```

	snr_db	acc	bacc	f1_macro	auc_macro_ovr
0	0	0.708333	0.708333	0.712355	0.892747
1	10	0.805556	0.805556	0.809524	0.947788
2	18	0.861111	0.861111	0.859888	0.972479

```
Загальний час навчання: 1.465602159500122
Час побудови квантових ознак (train+val): 5.53201699256897
```

Рис. 3.3 Класифікаційний рапорт моделі QiboFineTuned

Структура помилок в обох моделях показала, що найкраще розпізнавалися класи BPSK та QAM16. Для TorchQuantumFineTuned значення F1-score становили 0.9346 для BPSK та 0.9623 для QAM16. Для QiboFineTuned відповідні значення дорівнювали 0.9333 й 0.9533. Це означає, що обидві моделі надійно відокремлювали найбільш виразні сигнали як з точки зору фазової структури, так й з точки зору амплітудно-фазової конфігурації. Основні труднощі в обох випадках були пов'язані з розмежуванням QPSK І 8PSK, що є очікуваним для модуляцій зі схожою фазовою організацією. Для TorchQuantumFineTuned F1-score для QPSK становив 0.6508, а для 8PSK – 0.5591. Для QiboFineTuned ці значення дорівнювали 0.6355 й 0.6549 відповідно. Тобто модель на базі Qibo забезпечила більш збалансоване розділення саме найбільш складної фазової пари, тоді як TorchQuantum показав дещо кращу якість для QPSK, але слабший результат для 8PSK.

Більше деталей дає рапорт про поведінку моделей за різних рівнів SNR. Для TorchQuantumFineTuned при 0 дб було отримано accuracy=0.6944, balanced accuracy=0.6944, macro-F1=0.7041, macro-AUC(OVR)=0.9064. При 10 дБ ці показники зросли до 0.8333, 0.8333, 0.8315 та 0.9504 відповідно, а при 18 дБ становили 0.8056, 0.8056, 0.7930 й 0.9612. Для QiboFineTuned при 0 дБ було отримано accuracy=0.7083, balanced accuracy=0.7083, macro-F1=0.7124, macro-

AUC(OVR)=0.8927. При 10 дБ значення дорівнювали 0.8056, 0.8056, 0.8095 та 0.9478 відповідно, а при 18 дБ досягали 0.8611, 0.8611, 0.8599 й 0.9725. Для обох моделей спостерігається закономірне зростання якості зі збільшенням SNR, однак у випадку QiboFineTuned це зростання є більш виразним у верхньому діапазоні, де модель досягає найкращих підсумкових значень.

Матриці помилок підтверджують даний висновок, оскільки у випадку з TorchQuantumFineTuned найбільша кількість хибних класифікацій припадала на змішування QPSK і 8PSK, причому клас 8PSK досить часто помилково відносився до QPSK.

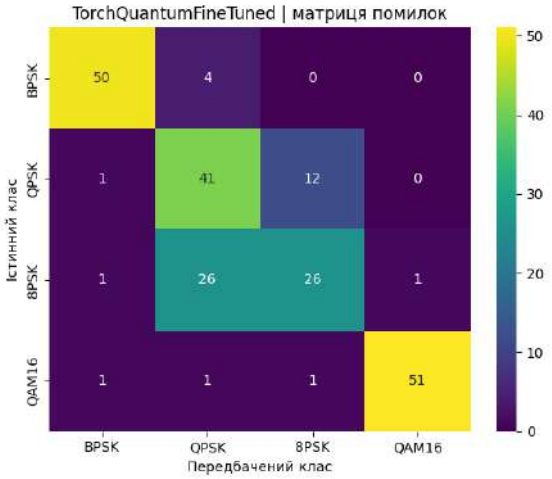


Рис. 3.4 Матриця помилок моделі TorchQuantumFineTuned

У моделі QiboFineTuned така тенденція також зберігалася, однак розподіл помилок був більш рівномірним, а кількість правильних класифікацій для 8PSK була вищою. Тобто наразі в поточній постановці модель на базі Qibo краще формує простір ознак для подальшого розділення близьких фазових модуляцій.

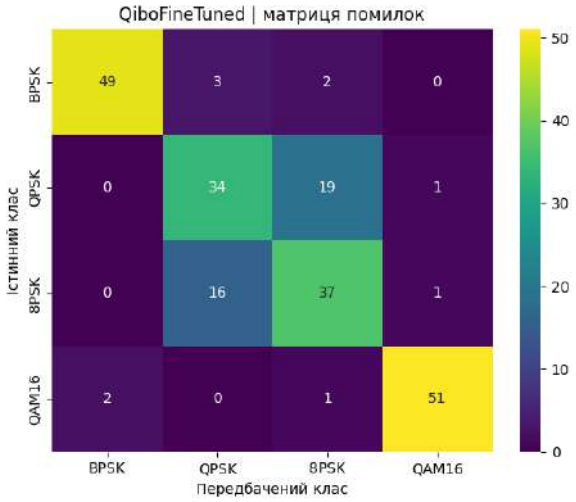


Рис. 3.5 Матриця помилок моделі QiboFineTuned

При аналізі графіків втрат та macro-F1 виявлено, що для TorchQuantumFineTuned спостерігається інтенсивне зменшення функції втрат у перших епохах та зростання валідаційного macro-F1 до рівня близько 0.79, після чого раннє зупинення фіксує найкращий стан моделі. Це підтверджує ефективність навчання квантово-класичної мережі, але також показує наявність певної нестійкості в пізніших епохах, що узгоджується з коливаннями норми градієнта.

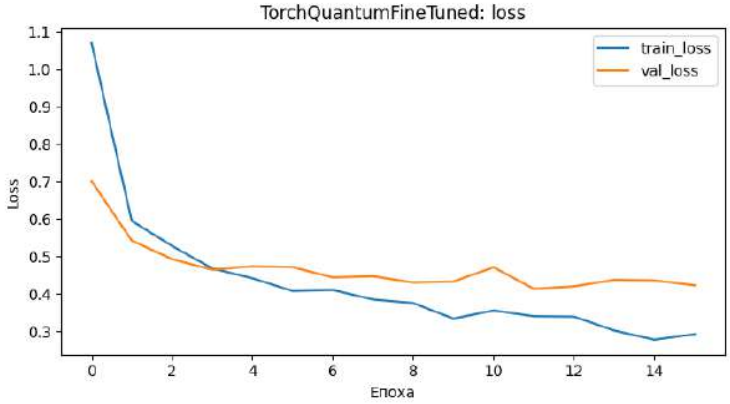


Рис. 3.6 Графік втрат модель TorchQuantumFineTuned

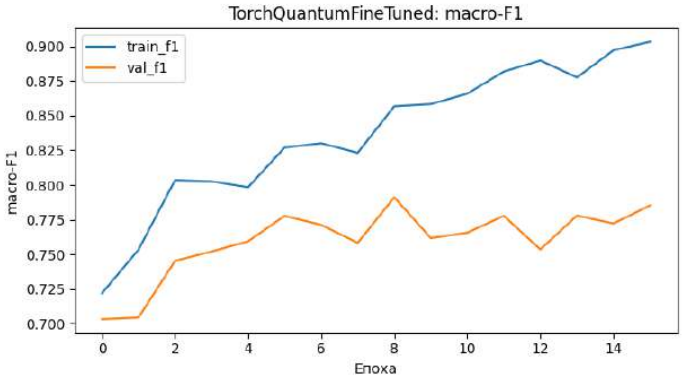


Рис. 3.7 Графік macro-F1 TorchQuantumFineTuned

Для QiboFineTuned навчання класичної голови відбувалося значно швидше. Валідаційна функція втрат поступово зменшувалася до області найкращих значень у середині процесу навчання, а валідаційний macro-F1 стабілізувався в інтервалі близько 0.74-0.75. Водночас наприкінці навчання між train та validation уже спостерігався помітний розрив.

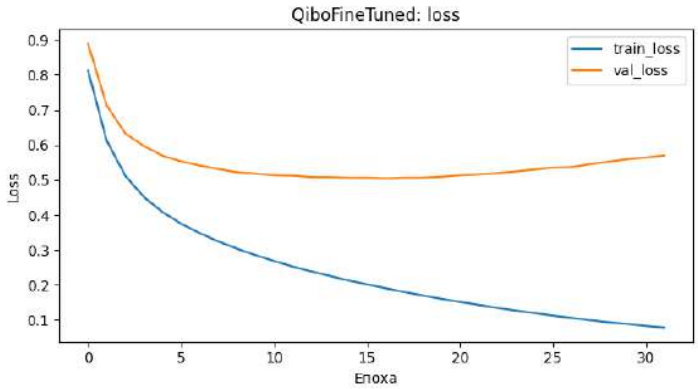


Рис. 3.8 Графік втрат модель QiboFineTuned

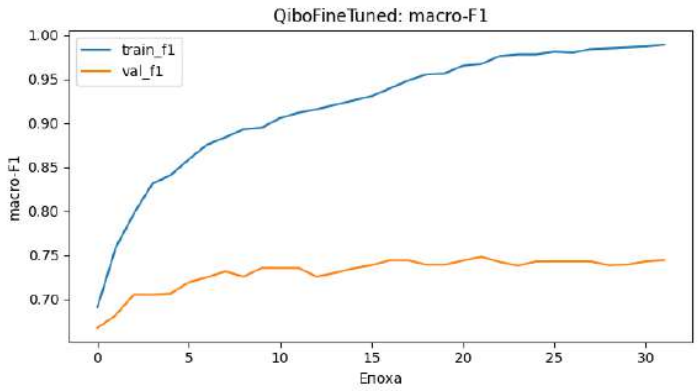


Рис. 3.9 Графік macro-F1 QiboFineTuned

Для перевірки узагальнювальної здатності обох моделей було також проведено зовнішнє тестування на датасеті RadioML 2016.10B [41], який завантажувався через kagglehub у вигляді файлу RML2016.10b.dat. Перевірка структури показала, що зовнішній датасет містить 200 ключів. Для зовнішнього оцінювання було сформовано окрему збалансовану підвибірку розміром 600 прикладів, яка включала ті самі чотири класи модуляції BPSK, QPSK, 8PSK, QAM16 та ті самі три рівні SNR, а саме 0, 10 та 18 дБ. Для кожного класу використовувалося по 150 прикладів, а для кожного значення SNR по 200 прикладів. Над зовнішнім набором застосовувався той самий pipeline попередньої обробки, що й для внутрішнього тестування, тобто побудова 50 ознак, стандартизація за вже навченим feature_scaler, перетворення через PCA до 8 компонентів та додавання нормалізованого значення SNR.

За результатами зовнішнього тестування модель TorchQuantumFineTuned продемонструвала accuracy=0.8000, balanced accuracy=0.8000, macro-F1=0.7996, macro-AUC(OVR)= 0.9457. Для класів BPSK та QAM16 значення F1-score становили 0.9508 й 0.9559 відповідно, тоді як для QPSK та 8PSK дорівнювали 0.6584 й 0.6331. При 0 дБ було отримано accuracy

balanced accuracy=0.7550, macro-F1=0.7570, macro-AUC(OVR)=0.9117; при 10 дБ 0.8100, 0.8100, 0.8061 та 0.9559; при 18 дБ 0.8350, 0.8350, 0.8344 та 0.9616 відповідно. Отже, зовнішнє тестування підтвердило здатність моделі зберігати високу якість класифікації і за межами початкового набору RadioML 2016.10A.

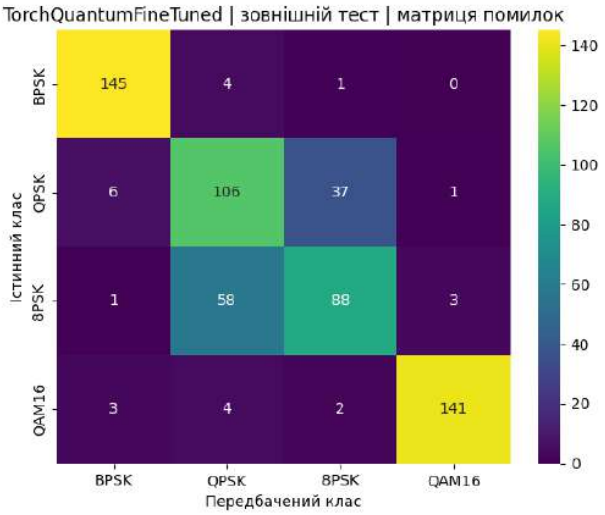


Рис. 3.10 Матриця помилок моделі TorchQuantumFineTuned на зовнішньому тесті

Матриця помилок моделі TorchQuantumFineTuned на зовнішньому наборі даних показує, що найкраще розпізнавалися класи BPSK та QAM16, для яких кількість правильних класифікацій становила 145 та 141 відповідно з 150 можливих. Найбільша частина похибок зосереджена між класами QPSK та 8PSK, оскільки QPSK у 37 випадках було віднесено до 8PSK, а 8PSK у 58 випадках – до QPSK . Це підтверджує, що модель достатньо добре відокремлює сигнали з більш виразною структурою, проте має складнощі при розділенні близьких фазових модуляцій на зовнішньому наборі.

Модель QiboFineTuned на зовнішньому наборі RadioML 2016.10B показала accuracy=0.7917, balanced accuracy=0.7917, macro-F1=0.7906, macro-AUC(OVR)=0.9473. Для класів BPSK та QAM16 значення F1-score становили 0.9439 й 0.9533, а для QPSK і 8PSK – 0.6202 та 0.6452 відповідно. При 0 дБ було отримано accuracy=0.7050, balanced accuracy=0.7050, macro-F1=0.7041, macro-AUC(OVR)=0.9052; при 10 дБ 0.8250, 0.8250, 0.8213 й 0.9663; при 18 дБ 0.8450, 0.8450, 0.8468 та 0.9615 відповідно. Таким чином, зовнішній експеримент підтвердив, що модель QiboFineTuned також зберігає високу якість класифікації на іншому наборі даних та демонструє дещо стабільніше розділення складної пари QPSK–8PSK.

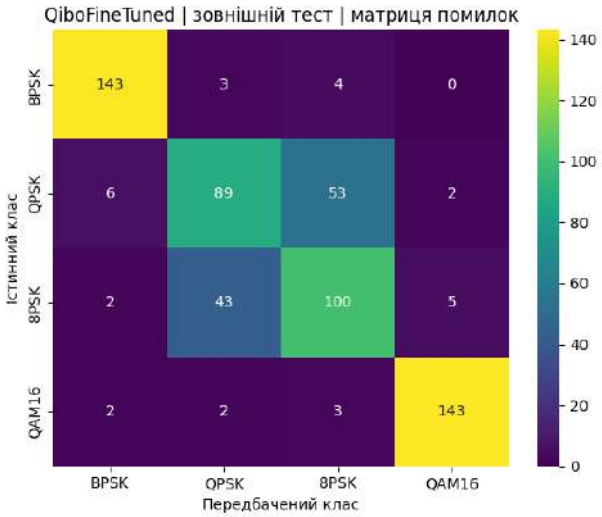


Рис. 3.11 Матриця помилок моделі QiboFineTuned на зовнішньому тесті

Матриця помилок моделі QiboFineTuned на зовнішньому тесті також демонструє високу якість розпізнавання класів BPSK та QAM16, для яких було отримано 143 правильних класифікації в обох випадках із 150. Основні похибки, як і у попередній моделі, зосереджені між QPSK та 8PSK полягали в тому, що клас QPSK у 53 випадках було помилково віднесено до 8PSK, а 8PSK у 43 випадках до QPSK. Водночас розподіл помилок у цій моделі є більш збалансованим, а кількість правильних класифікацій для 8PSK є вищою, ніж у TorchQuantumFineTuned, що свідчить про кращу здатність моделі QiboFineTuned до розділення складної пари фазових модуляцій на зовнішньому наборі даних.

Порівняння результатів зовнішнього тестування показує, що обидві моделі зберегли близький рівень якості при переході до нового датасету. Для TorchQuantumFineTuned зовнішні показники навіть дещо перевищили результати внутрішнього тесту за accuracy, macro-F1 та macro-AUC(OVR), що свідчить про достатню узагальнювальну здатність моделі в межах близького за структурою набору даних. У випадку QiboFineTuned значення accuracy й balanced accuracy на зовнішньому тесті залишилися практично на тому самому рівні, а macro-AUC(OVR) навіть зріс до 0.9473. . Це означає, що обидві моделі не були переналаштовані лише під один конкретний набір прикладів, а здатні

переносити отримані закономірності на інший датасет тієї самої предметної області.

Загалом результати внутрішнього і зовнішнього тестування показують, що обидві фінальні моделі є працездатними для задачі багатокласової класифікації модуляцій на основі I/Q-сигналів. Модель QiboFineTuned продемонструвала дещо вищу якість на внутрішньому тесті та значно меншу обчислювальну вартість, тоді як TorchQuantumFineTuned показала близькі результати на зовнішньому наборі та високу якість для класів BPSK і QAM16. В обох випадках найбільш складним аспектом задачі залишилося розділення близьких фазових модуляцій QPSK і 8PSK, особливо за нижчих значень SNR.

Ноутбуки з усіма експериментами розташовані в даному репозиторії: https://github.com/xrendezvous/QML_framework_analysis

ВИСНОВКИ

В цій кваліфікаційній роботі було виконано порівняльний аналіз сучасних алгоритмів та фреймворків QML та досліджено їх придатність до задачі автоматичної класифікації модуляції радіосигналів. В межах дослідження розглянуто теоретичні основи QML, сучасні квантово-класичні алгоритмічні підходи, особливості програмних фреймворків, а також практичні обмеження, пов'язані з кодуванням даних, NISQ-пристроями, стабільністю навчання та ресурсною складністю.

У першому розділі проведено аналіз взаємозв'язку квантових обчислень та машинного навчання в задачах обробки радіосигналів. Визначено, що QML є перспективним напрямом для дослідження високорозмірних, шумових й нелінійних сигналових даних, оскільки дає змогу використовувати квантові простори ознак, квантові ядра, параметризовані квантові схеми та гібридні моделі. Водночас встановлено, що практична ефективність QML залежить від способу кодування даних, структури схеми, кількості кубітів, глибини моделі, обчислювального середовища та характеристик датасету.

У другому розділі було сформовано критерії порівняльного аналізу QML-фреймворків, зокрема архітектурні можливості, підтримку гібридних моделей, якість класифікації, динаміку навчання, час виконання та стабільність середовища. Було розглянуто PennyLane, TorchQuantum, TensorFlow Quantum, Qulacs, Qibo та Amazon Braket. За результатами попереднього аналізу найбільш перспективними для подальшої практичної реалізації було визначено Qibo, TorchQuantum та Amazon Braket, оскільки вони продемонстрували найкраще поєднання якості, швидкодії та архітектурної репрезентативності.

У третьому розділі було реалізовано практичну задачу багатокласової класифікації модуляцій BPSK, QPSK, 8PSK та QAM16 на основі I/Q-сигналів.

Для внутрішнього тестування використовувався датасет RadioML 2016.10A, а для зовнішньої перевірки узагальнювальної здатності – RadioML 2016.10B. Було виконано попередню обробку сигналів, стандартизацію, PCA-перетворення до 8 компонентів, додано нормалізоване значення SNR та реалізовано дві фінальні моделі: TorchQuantumFineTuned й QiboFineTuned.

За результатами внутрішнього тестування TorchQuantumFineTuned досягла accuracy=0.7778, balanced accuracy=0.7778, macro-F1=0.7767 та macro-AUC=0.9416. Модель QiboFineTuned показала дещо вищі результати: accuracy=0.7917, balanced accuracy=0.7917, macro-F1=0.7942 та macro-AUC=0.9441. При цьому QiboFineTuned мала значно меншу обчислювальну вартість, тоді як TorchQuantumFineTuned забезпечила оцінну end-to-end гібридну архітектуру.

Зовнішнє тестування підтвердило працездатність обох моделей на іншому наборі даних тієї самої предметної області. TorchQuantumFineTuned на RadioML 2016.10B досягла accuracy=0.8000, balanced accuracy=0.8000, macro-F1=0.7996 та macro-AUC=0.9457. QiboFineTuned показала accuracy=0.7917, balanced accuracy=0.7917, macro-F1=0.7906 та macro-AUC=0.9473. Це свідчить про здатність моделей узагальнювати отримані закономірності поза межами одного тестового набору.

Аналіз матриць помилок показав, що моделі найкраще розпізнавали класи BPSK та QAM16, тоді як основні труднощі виникали при розмежуванні QPSK та 8PSK. Це пояснюється близькістю фазової структури цих типів модуляції. Отримані результати підтверджують, що QML-фреймворки можуть бути використані для експериментального дослідження задач автоматичної класифікації модуляції, однак їх практична цінність визначається не лише точністю, а й повним експериментальним пайплайном: підготовкою даних, вибором фреймворку, способом інтеграції квантової частини, стабільністю середовища, часом виконання та здатністю моделі до узагальнення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Fahmy, J. M. The Intersection of Quantum Computing and Machine Learning: A Review of Fundamental Concepts and Applications. Unpublished. 2025. URL: https://doi.org/10.13140/RG.2.2.16053.95205

[2] Olaitan, O. F., Ayeni, S. O., Olosunde, A., Okeke, F. C., Okonkwo, U. U., Ochieze, C. G., Chukwujama, O. V., & Akatakpo, O. N. Quantum Computing in Artificial Intelligence: a Review of Quantum Machine Learning Algorithms. *Path of Science.* 2025. Vol. 11, no. 5. P. 7001. URL: https://doi.org/10.22178/pos.117-25

[3] Aparcana-Tasayco, A. J., Deng, X., & Park, J. H. A systematic review of anomaly detection in IoT security: towards quantum machine learning approach. *EPJ Quantum Technology*. 2025. Vol. 11, no. 5 P. 7001. URL: https://doi.org/10.1140/epjqt/s40507-025-00414-6

[4] Peral-García, D., Cruz-Benito, J., García-Peñalvo F. J. Systematic literature review: Quantum machine learning and its applications. *Computer Science Review*. 2024. Vol. 51. Article 100619. URL: https://doi.org/10.1016/j.cosrev.2024.100619

[5] Devadas, R. M., T, S. Quantum machine learning: A comprehensive review of integrating AI with quantum computing for computational advancements. *MethodsX*. 2025. Vol. 14. Article 103318. URL: https://doi.org/10.1016/j.mex.2025.103318

[6] Rodríguez-Díaz F., Gutiérrez-Avilés D., Troncoso A., Martínez-Álvarez F. A Survey of Quantum Machine Learning: Foundations, Algorithms, Frameworks, Data and Applications. *ACM Computing Surveys.* 2025. URL: https://doi.org/10.1145/3764582

[7] Cerezo M., Arrasmith A., Babbush R. et al. Variational quantum algorithms. *Nature Reviews Physics*. 2021. Vol. 3, no. 9. P. 625–644. URL: https://doi.org/10.1038/s42254-021-00348-9

[8] Bharti K., Cervera-Lierta A., Kyaw T. H. et al. Noisy intermediate-scale quantum algorithms. *Reviews of Modern Physics.* 2022. Vol. 94, no. 1. Article 015004. URL: https://doi.org/10.1103/RevModPhys.94.015004

[9] Liu Y., Arunachalam S., Temme K. A rigorous and robust quantum speed-up in supervised machine learning. *Nature Physics.* 2021. Vol. 17. P. 1013–1017. URL: https://doi.org/10.1038/s41567-021-01287-z

[10] El-Haryqy N., Kharbouche A., Ouamna H., Madini Z., Zouine Y. Improved automatic modulation recognition using deep learning with additive attention. *Results in Engineering.* 2025. Vol. 26. Article 104783. URL: https://doi.org/10.1016/j.rineng.2025.104783

[11] Huang H.-Y., Broughton M., Mohseni M. et al. Power of data in quantum machine learning. *Nature Communications.* 2021. Vol. 12. Article 2631. URL: https://doi.org/10.1038/s41467-021-22539-9

[12] Caro M. C., Huang H.-Y., Cerezo M. et al. Generalization in quantum machine learning from few training data. *Nature Communications.* 2022. Vol. 13. Article 4919. URL: https://doi.org/10.1038/s41467-022-32550-3

[13] Abbas A., Sutter D., Zoufal C. et al. The power of quantum neural networks. *Nature Computational Science.* 2021. Vol. 1. P. 403–409. URL: https://doi.org/10.1038/s43588-021-00084-1

[14] Jerbi S., Fiderer L. J., Poulsen Nautrup H. et al. Quantum machine learning beyond kernel methods. *Nature Communications.* 2023. Vol. 14. Article 517. URL: https://doi.org/10.1038/s41467-023-36159-y

[15] Thanasilp S., Wang S., Cerezo M. et al. Exponential concentration in quantum kernel methods. *Nature Communications*. 2024. Vol. 15. Article 5200. URL: https://doi.org/10.1038/s41467-024-49287-w

[16] Schnabel J., Roth M. Quantum kernel methods under scrutiny: a benchmarking study. *Quantum Machine Intelligence*. 2025. Vol. 7. Article 58. URL: https://doi.org/10.1007/s42484-025-00273-5

[17] Larocca M., Thanasilp S., Wang S. et al. Barren plateaus in variational quantum computing. *Nature Reviews Physics*. 2025. Vol. 7. P. 174–189. URL: https://doi.org/10.1038/s42254-025-00813-9

[18] Sack S. H., Medina R. A., Michailidis A. A., Kueng R., Serbyn M. Avoiding Barren Plateaus Using Classical Shadows. *PRX Quantum*. 2022. Vol. 3, no. 2. Article 020365. URL: https://journals.aps.org/prxquantum/abstract/10.1103/PRXQuantum.3.020365

[19] Broughton M., Verdon G., McCourt T. et al. TensorFlow Quantum: A Software Framework for Quantum Machine Learning. *arXiv*. 2020. URL: https://arxiv.org/abs/2003.02989 (date of access: 07.05.2026)

[20] Sahin M. E., Altamura E., Wallis O. et al. Qiskit Machine Learning: an open-source library for quantum machine learning tasks at scale on quantum hardware and classical simulators. *arXiv*. 2025. URL: https://arxiv.org/abs/2505.17756 (date of access: 01.05.2026)

[21] Efthymiou S., Ramos-Calderer S., Bravo-Prieto C. et al. Qibo: a framework for quantum simulation with hardware acceleration. *Quantum Science and Technology*. 2021. Vol. 7, no. 1. Article 015018. URL: https://doi.org/10.1088/2058-9565/ac39f5

[22] Robbiati M., Papaluca A., Pasquale A. et al. Qiboml: towards the orchestration of quantum-classical machine learning. *arXiv*. 2025. URL: https://arxiv.org/abs/2510.11773 (date of access: 07.05.2026)

[23] Suzuki Y., Kawase Y., Masumura Y. et al. Qulacs: a fast and versatile quantum circuit simulator for research purpose. *Quantum*. 2021. Vol. 5. Article 559. URL: https://ar5iv.labs.arxiv.org/html/2011.13524 (date of access: 07.05.2026)

[24] Wu Y., Adermann E., Thapa C., Camtepe S., Suzuki H., Usman M. Radio Signal Classification by Adversarially Robust Quantum Machine Learning. *arXiv*. 2023. URL: https://arxiv.org/abs/2312.07821 (date of access: 01.05.2026)

[25] Martinez G. F., Croci A., Drago F., Niccolai A., Mussetta M., Zich R. E. Radar Signal Classification with Quantum Machine Learning: Ansatz Depth Impact on Expressibility. *Electronics*. 2026. Vol. 15, no. 2. Article 370. URL: https://doi.org/10.3390/electronics15020370

[26] Fan L., Wang L., Mao Y., Zhang Z., Yu X. Radar Signal Classification Based on Quantum-Classical Hybrid Convolutional Neural Network. *Advanced Quantum Technologies*. 2026. Vol. 9, no. 1. Article e00323. URL: https://doi.org/10.1002/qute.202500323

[27] Jabbar A., Jianjun H., Jabbar M. K. et al. Quantum-assisted federated learning for radar-based object tracking in IoT-enabled environments. *EPJ Quantum Technology*. 2025. Vol. 12. Article 141. URL: https://doi.org/10.1140/epjqt/s40507-025-00440-4

[28] Miller L., Uehara G., Sharma A., Spanias A. Quantum Machine Learning for Optical and SAR Classification. *DSP 2023*. URL: https://2025.ic-dsp.org/wp-content/uploads/2023/05/DSP2023-94.pdf (date of access: 10.05.2026)

[29] Nguyen V.-L., Nguyen L.-H., Hwang R.-H. et al. Quantum Machine Learning for 6G Network Intelligence and Adversarial Threats. *IEEE Communications Standards Magazine*. 2025. P. 1–1. DOI: 10.1109/MCOMSTD.2025.3575261. URL: https://cerc-ngct.ca/wp-content/uploads/2025/06/J-2025-IEEE-COMSTD-Quantum-Machine-Learning-for-6G-Network-Intelligence-and-Adversarial-Threats.pdf (date of access: 01.05.2026)

[30] Liu X., Mao Z., Liu M., Wang C., Cai Z. Automatic Modulation Classification Based on a Dynamic Graph Architecture. *Applied Sciences*. 2025. Vol. 15, no. 21. Article 11782. URL: https://doi.org/10.3390/app152111782

[31] Kassri N., Ennouaary A., Bah S. Efficient Squeeze-and-Excitation-Enhanced Deep Learning Method for Automatic Modulation Classification. *International Journal of Advanced Computer Science and Applications*. 2024. Vol. 15, no. 6. URL: http://dx.doi.org/10.14569/IJACSA.2024.0150655

[32] TorchQuantum: Quantum Computing in PyTorch. *GitHub*. URL: https://github.com/mit-han-lab/torchquantum (date of access: 01.05.2026)

[33] API reference. *Qibo Documentation*. URL: https://qibo.science/qibo/stable/api-reference/qibo.html (date of access: 01.05.2026)

[34] Quantum Principal Component Analysis (QPCA). *Qniverse*. URL: https://qniverse.in/docs/qpca-algorithm/ (date of access: 01.05.2026)

[35] Metrics and scoring: quantifying the quality of predictions. *scikit-learn Documentation*. URL: https://scikit-learn.org/stable/api/sklearn.metrics.html (date of access: 01.05.2026)

[36] DeepSig Dataset. *GitHub*. URL: https://github.com/ianblenke/deepsig_dataset (date of access: 01.05.2026)

[37] Jafarigol E., Alaghband B., Gilanpour A., Hosseinipoor S., Mirmozafari M. AI/ML-Based Automatic Modulation Recognition: Recent Trends and Future Possibilities. *arXiv*. 2025. DOI: 10.48550/arXiv.2502.05315. URL: https://arxiv.org/abs/2502.05315 (date of access: 01.05.2026)

[38] PennyLane Documentation. *PennyLane*. URL: https://docs.pennylane.ai/en/stable/ (date of access: 01.05.2026)

[39] Interfaces. *PennyLane Documentation*. URL: https://docs.pennylane.ai/en/stable/introduction/interfaces.html (date of access: 01.05.2026)

[40] Amazon Braket Documentation. *Amazon Web Services*. URL: https://aws.amazon.com/documentation-overview/braket (date of access: 01.05.2026)

[41] Abudeeb M. RML2016.10b. *Kaggle*. URL: https://www.kaggle.com/datasets/marwanabudeeb/rml201610b (date of access: 12.05.2026)

ДОДАТКИ

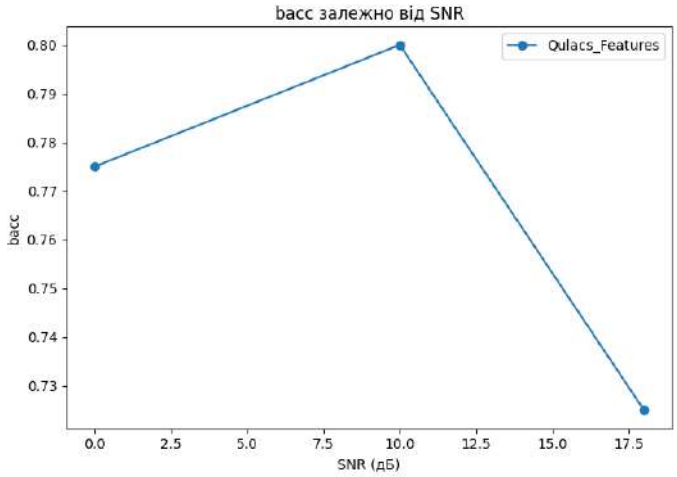


Рис. 2.1 Графік balanced accuracy Qulacs

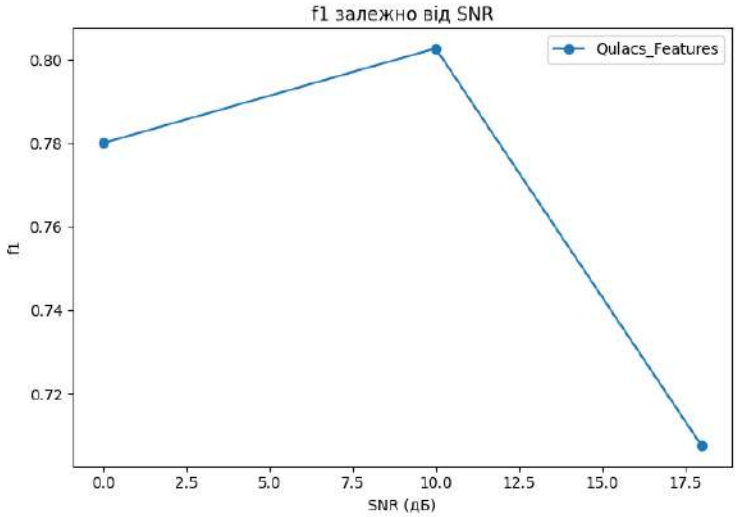


Рис. 2.2 Графік F1 Qulacs

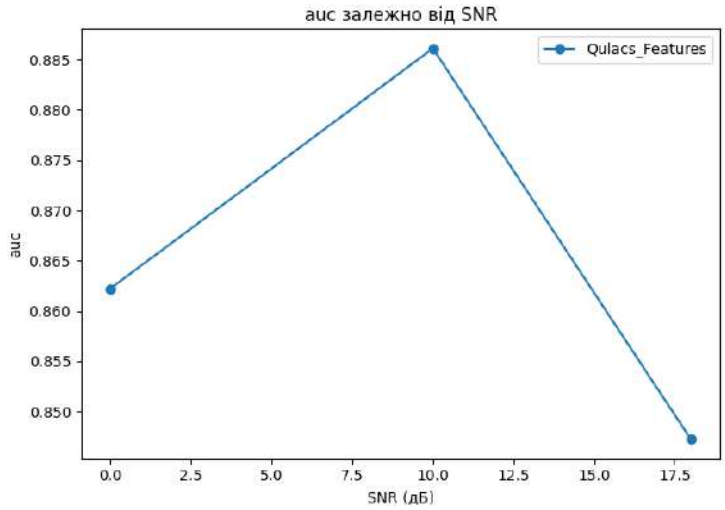


Рис. 2.3 Графік AUC Qulacs

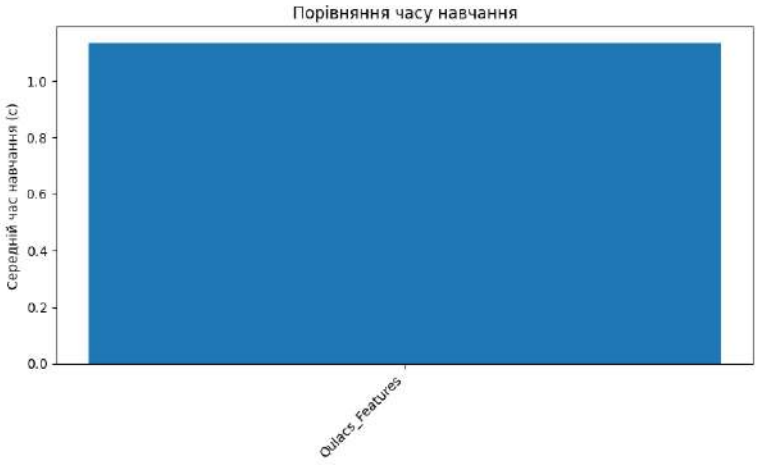


Рис. 2.4 Графік часу навчання Qulacs

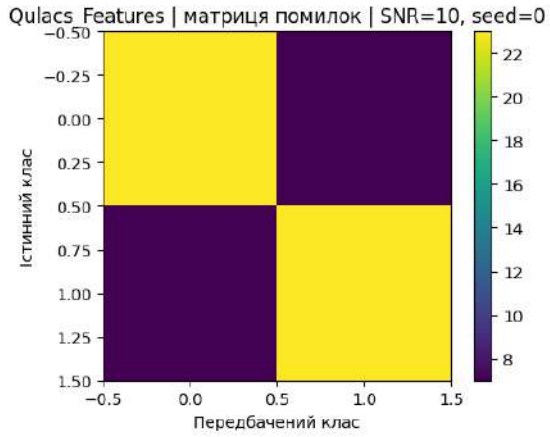


Рис. 2.5 Матриця помилок Qulacs

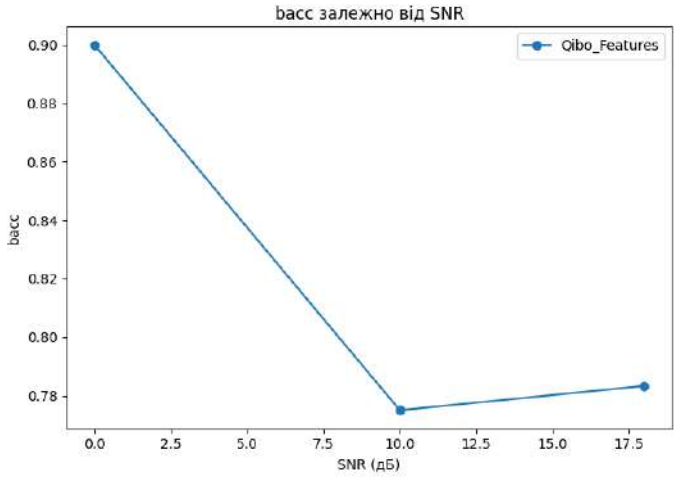


Рис. 2.6 Графік balanced accuracy Qibo

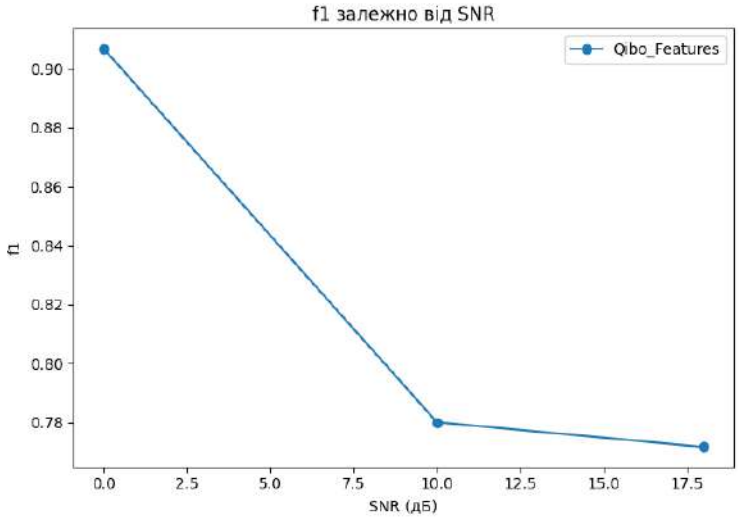


Рис. 2.7 Графік F1 Qibo

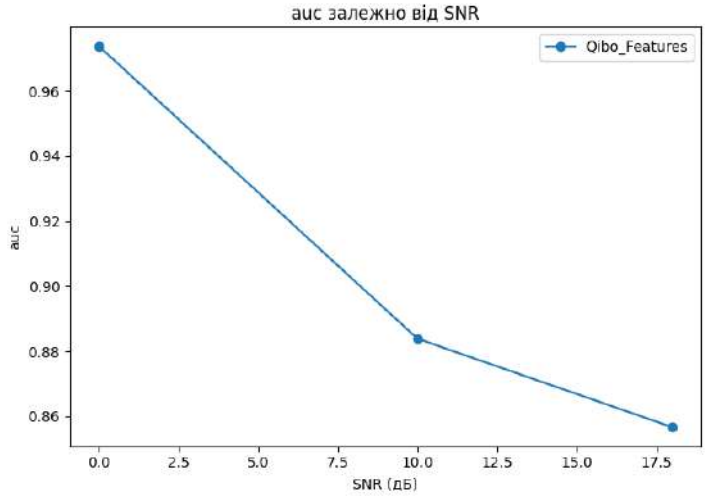


Рис. 2.8 Графік AUC Qibo

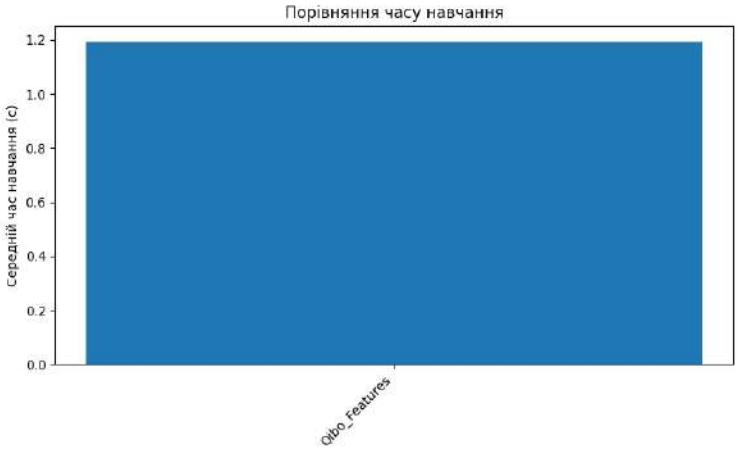


Рис. 2.9 Графік часу навчання Qibo

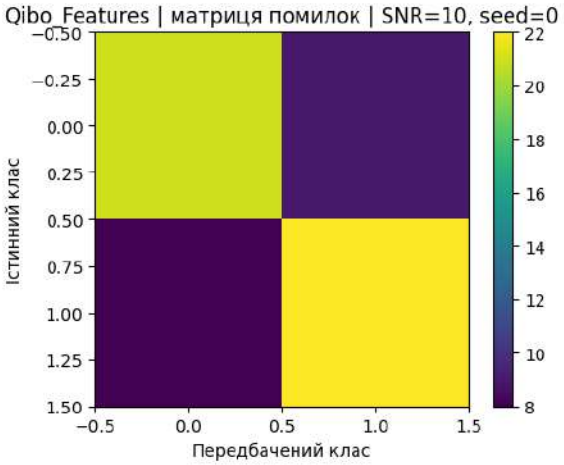


Рис. 2.10 Матриця помилок Qibo

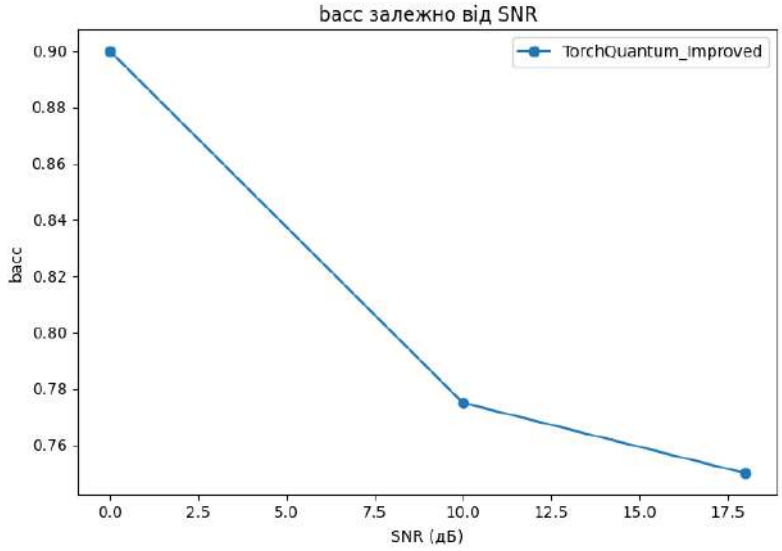


Рис. 2.11 Графік balanced accuracy TorchQuantum

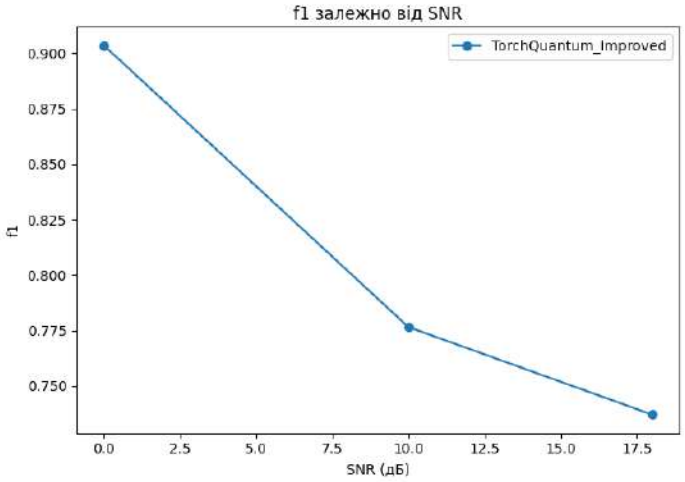


Рис. 2.12 Графік F1 TorchQuantum

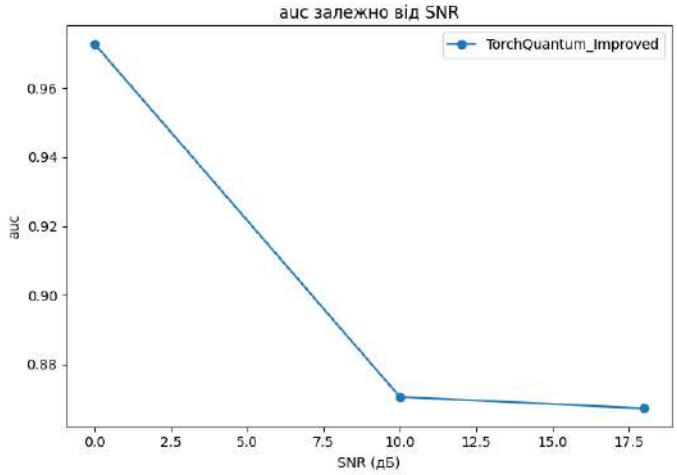


Рис. 2.13 Графік AUC TorchQuantum

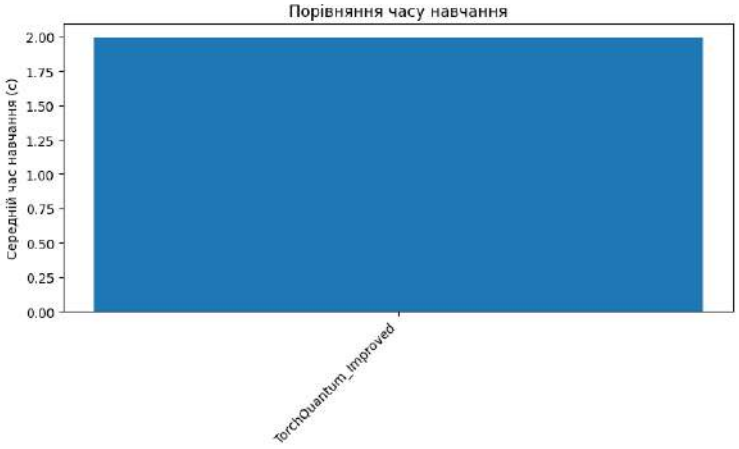


Рис. 2.14 Графік часу навчання TorchQuantum

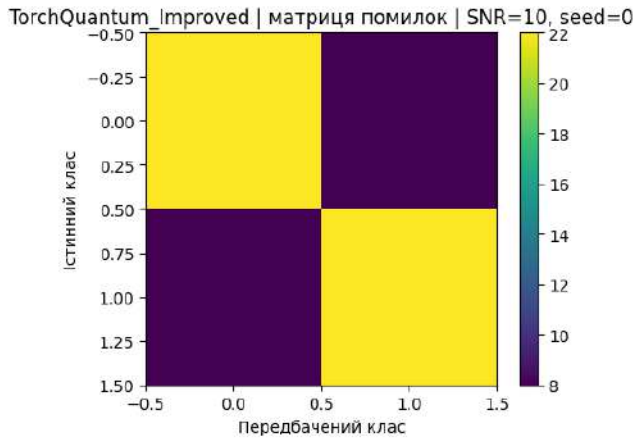


Рис. 2.15 Матриця помилок TorchQuantum

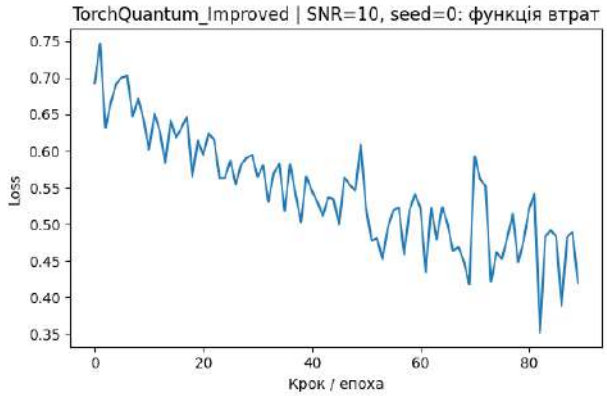


Рис. 2.16 Графік функцій втрат TorchQuantum

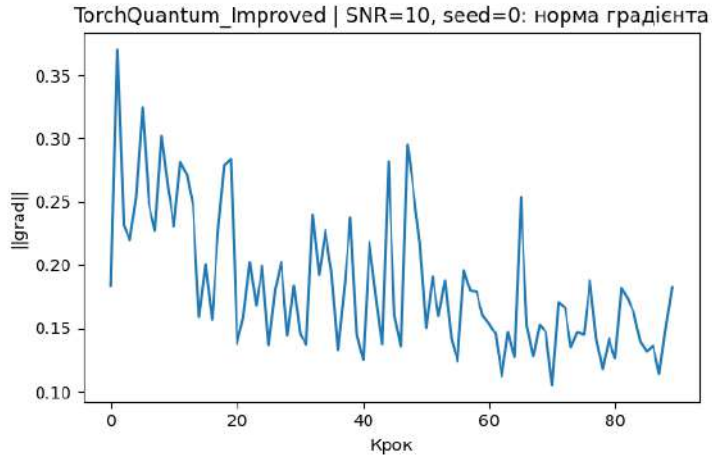


Рис. 2.17 Графік норми градієнта TorchQuantum

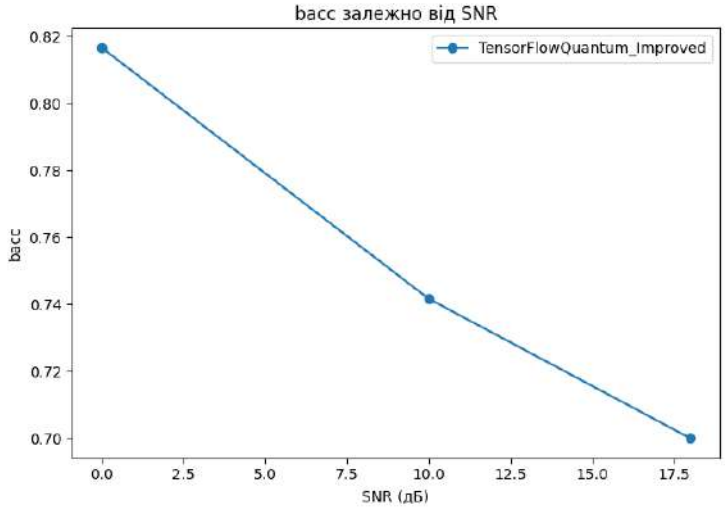


Рис. 2.18 Графік balanced accuracy TensorFlow Quantum

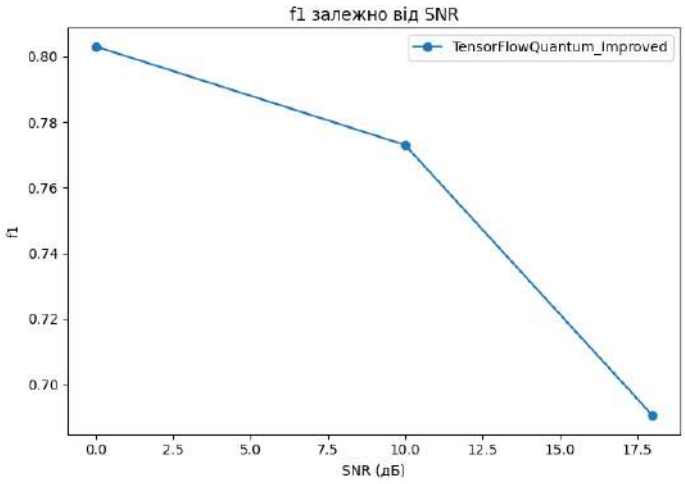


Рис. 2.19 Графік F1 TensorFlow Quantum

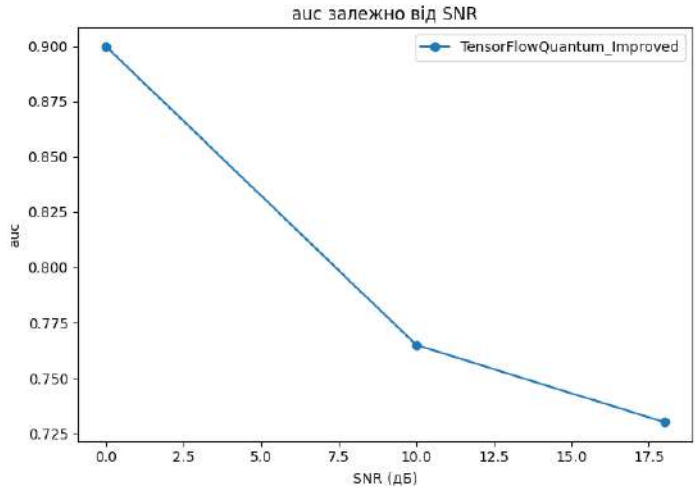


Рис. 2.20 Графік AUC TensorFlow Quantum

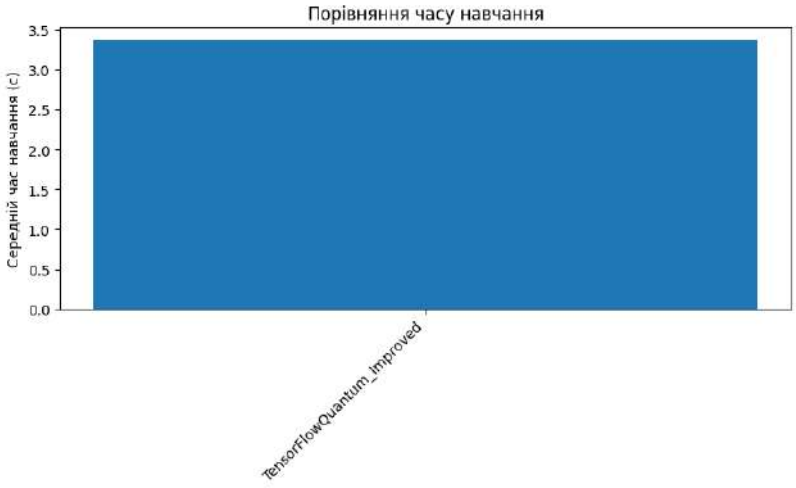


Рис. 2.21 Графік часу навчання TensorFlow Quantum

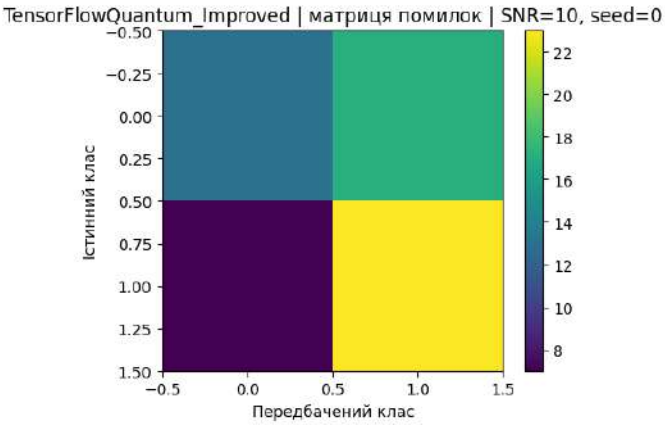


Рис. 2.22 Матриця помилок TensorFlow Quantum

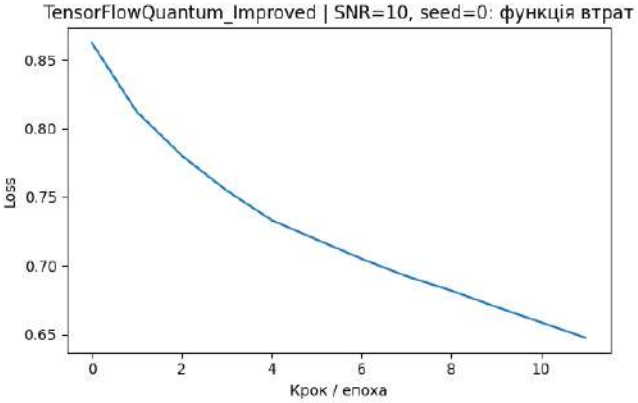


Рис. 2.23 Графік функції втрат TensorFlow Quantum

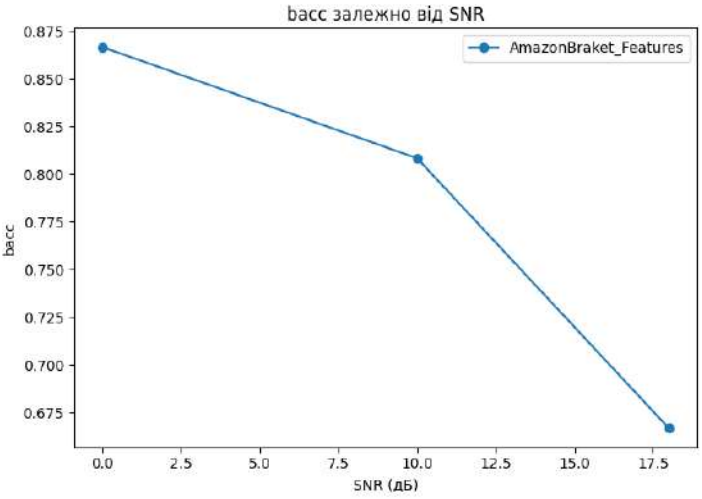


Рис. 2.24 Графік balanced accuracy Amazon Braket SDK

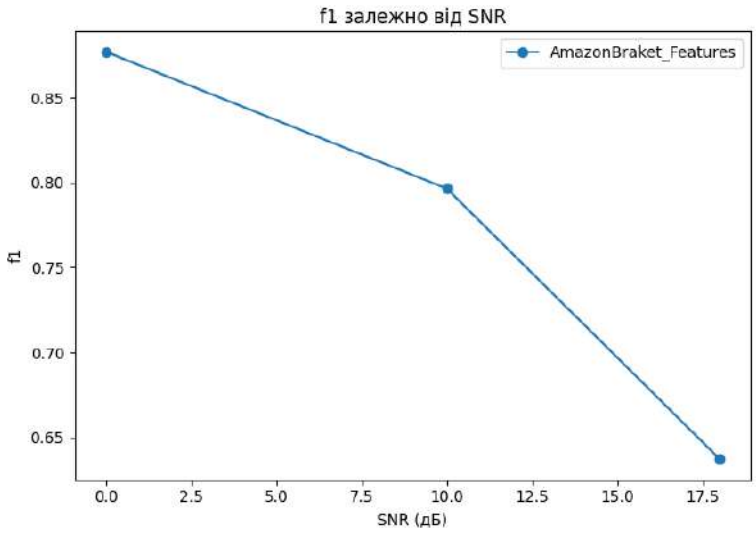


Рис. 2.25 Графік F1 Amazon Braket SDK

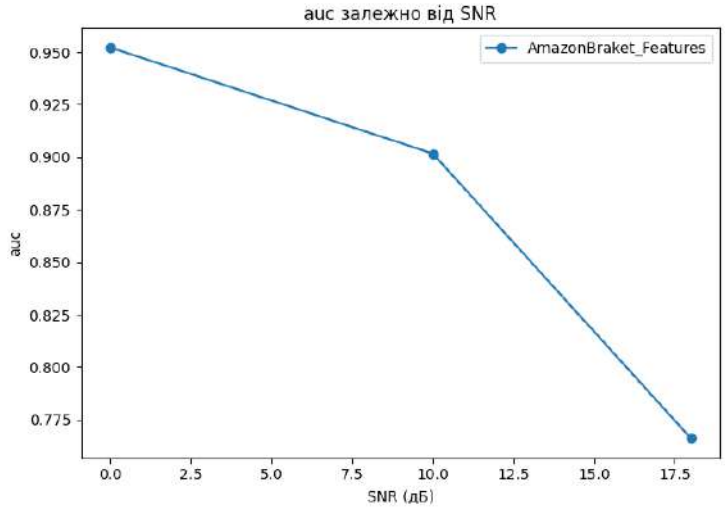


Рис. 2.26 Графік AUC Amazon Braket SDK

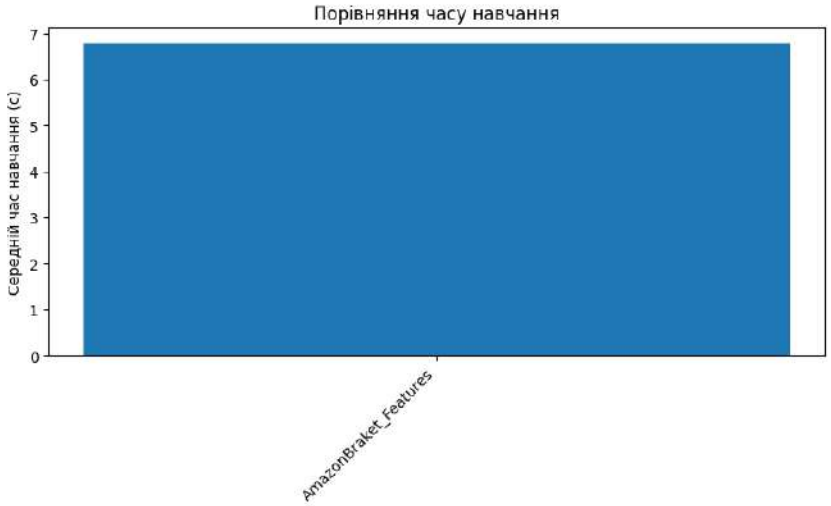


Рис. 2.27 Графік часу навчання Amazon Braket SDK

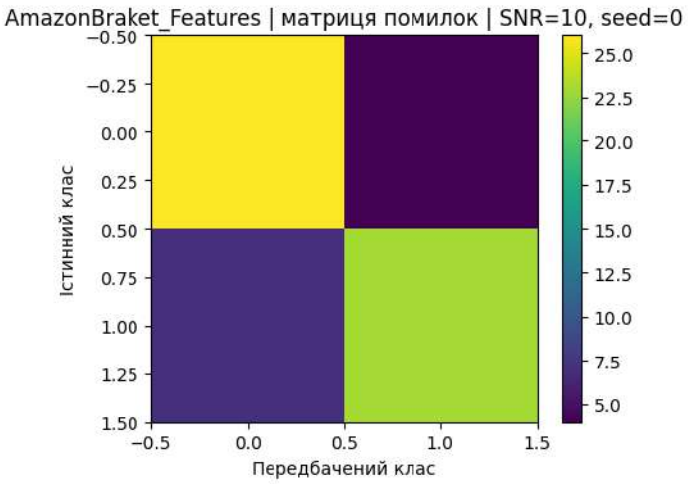


Рис. 2.28 Матриця помилок Amazon Braket SDK

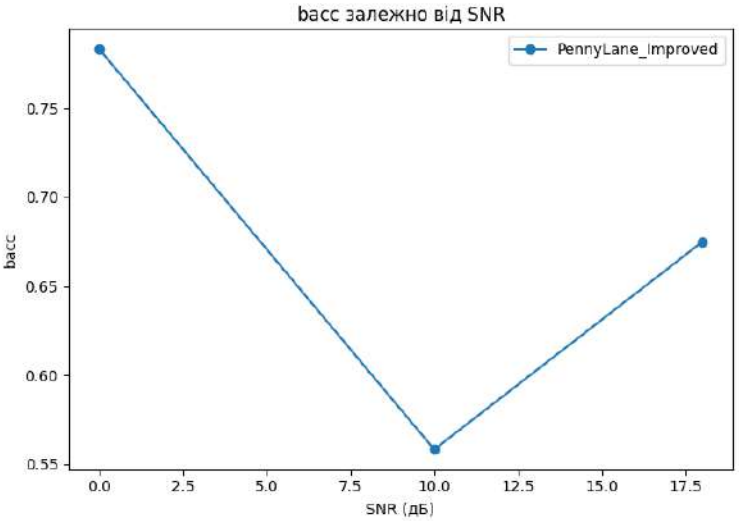


Рис. 2.29 Графік balanced accuracy PennyLane

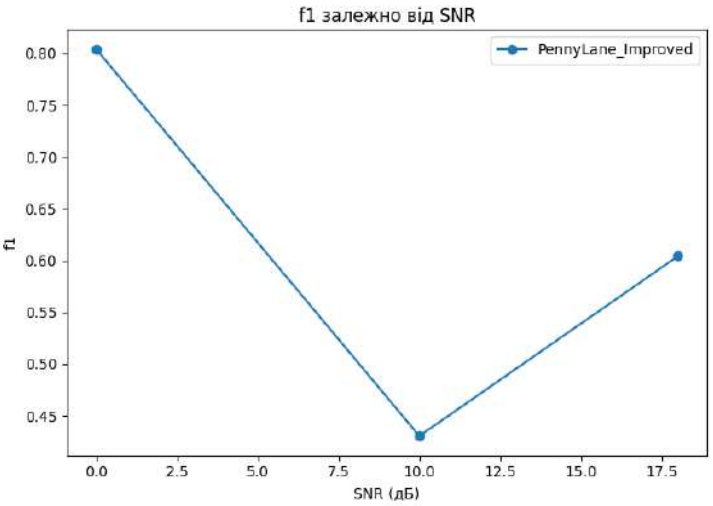


Рис. 2.30 Графік F1 PennyLane

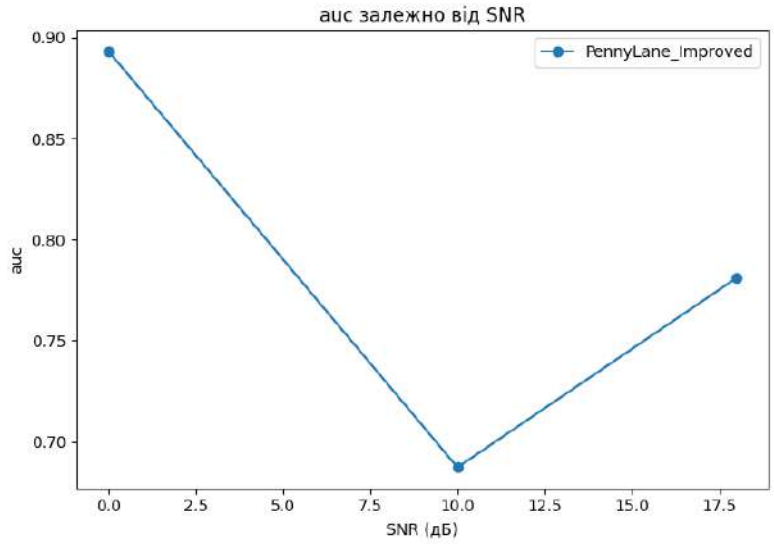


Рис. 2.31 Графік AUC PennyLane

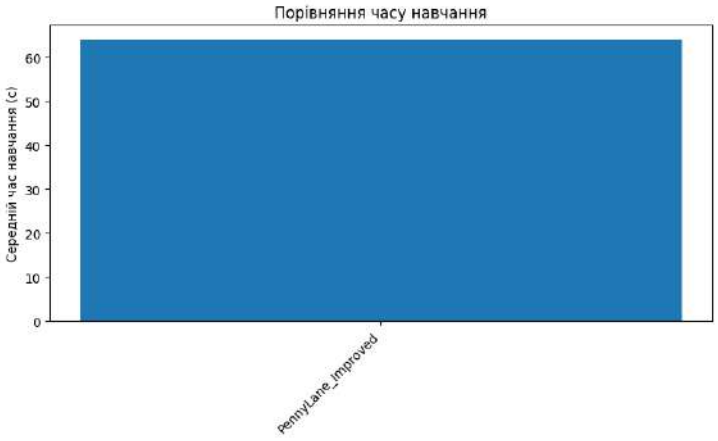


Рис. 2.32 Графік часу навчання PennyLane

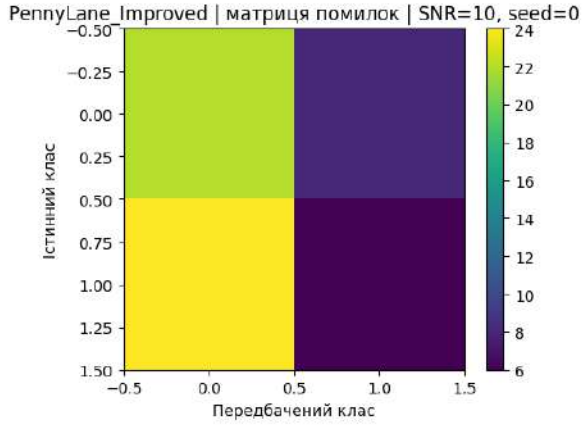


Рис. 2.33 Матриця помилок PennyLane

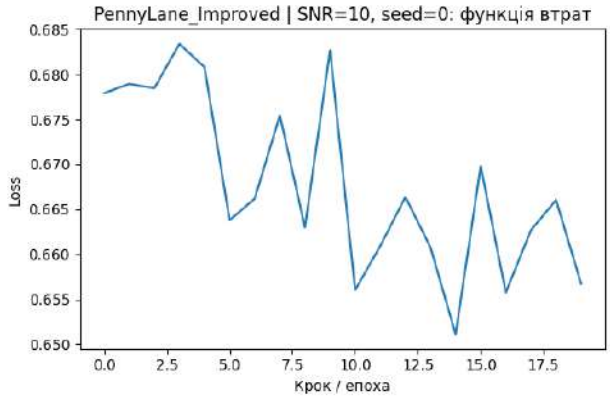


Рис. 2.34 Графік функції втрат PennyLane

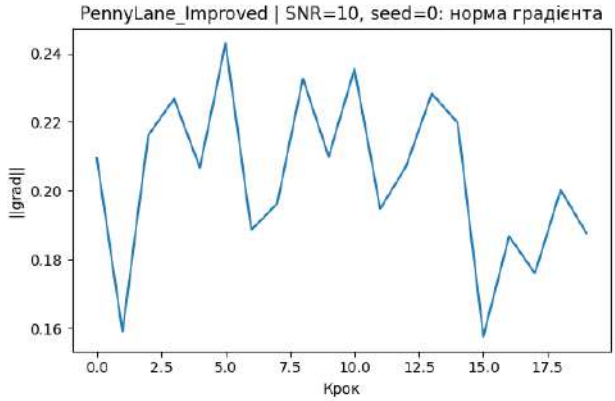


Рис. 2.35 Графік норми градієнта PennyLane

```
[TorchQuantumFineTuned] Загальні метрики:
{
  "acc": 0.7777777777777778,
  "bacc": 0.7777777777777777,
  "f1_macro": 0.776694256483905,
  "auc_macro_ovr": 0.9415580704160951
}

Classification report:
              precision    recall  f1-score   support

        BPSK     0.9434    0.9259    0.9346        54
        QPSK     0.5694    0.7593    0.6508        54
        8PSK     0.6667    0.4815    0.5591        54
       QAM16     0.9808    0.9444    0.9623        54

    accuracy                         0.7778       216
   macro avg     0.7901    0.7778    0.7767       216
weighted avg     0.7901    0.7778    0.7767       216
```

	snr_db	acc	bacc	f1_macro	auc_macro_ovr
0	0	0.694444	0.694444	0.704058	0.906379
1	10	0.833333	0.833333	0.831461	0.950360
2	18	0.805556	0.805556	0.792984	0.961163

```
Загальний час навчання: 162.8277142047882
```

Рис. 3.1 Класифікаційний рапорт моделі TorchQuantumFineTuned

```
[QiboFineTuned] Загальні метрики:
{
  "acc": 0.7916666666666666,
  "bacc": 0.7916666666666667,
  "f1_macro": 0.7942464091748683,
  "auc_macro_ovr": 0.9441015089163237
}
```

Рис. 3.2 Загальні метрики моделі QiboFineTuned

```
Classification report:
              precision    recall  f1-score   support

        BPSK     0.9608    0.9074    0.9333        54
        QPSK     0.6415    0.6296    0.6355        54
        8PSK     0.6271    0.6852    0.6549        54
       QAM16     0.9623    0.9444    0.9533        54

    accuracy                         0.7917       216
   macro avg     0.7979    0.7917    0.7942       216
weighted avg     0.7979    0.7917    0.7942       216
```

	snr_db	acc	bacc	f1_macro	auc_macro_ovr
0	0	0.708333	0.708333	0.712355	0.892747
1	10	0.805556	0.805556	0.809524	0.947788
2	18	0.861111	0.861111	0.859888	0.972479

```
Загальний час навчання: 1.465602159500122
Час побудови квантових ознак (train+val): 5.53201699256897
```

Рис. 3.3 Класифікаційний рапорт моделі QiboFineTuned

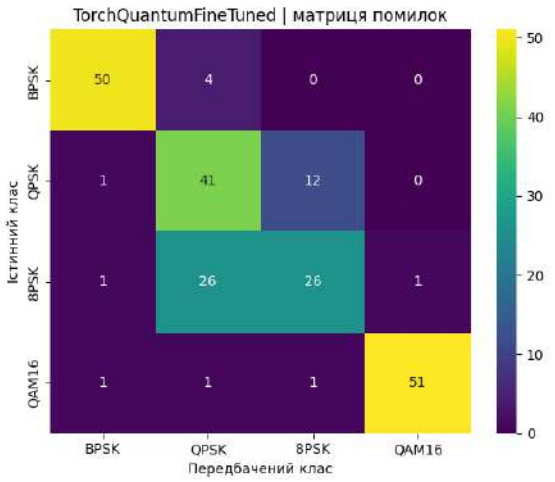


Рис. 3.4 Матриця помилок моделі TorchQuantumFineTuned

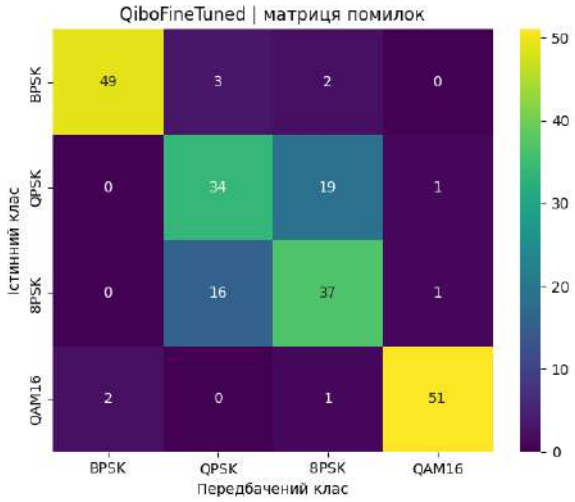


Рис. 3.5 Матриця помилок моделі QiboFineTuned

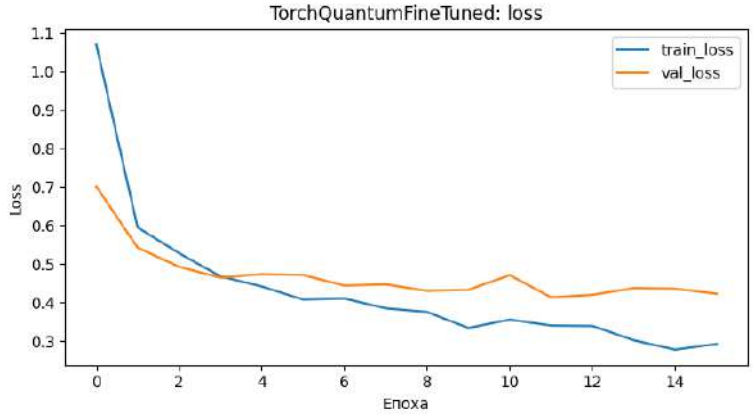


Рис. 3.6 Графік втрат модель TorchQuantumFineTuned

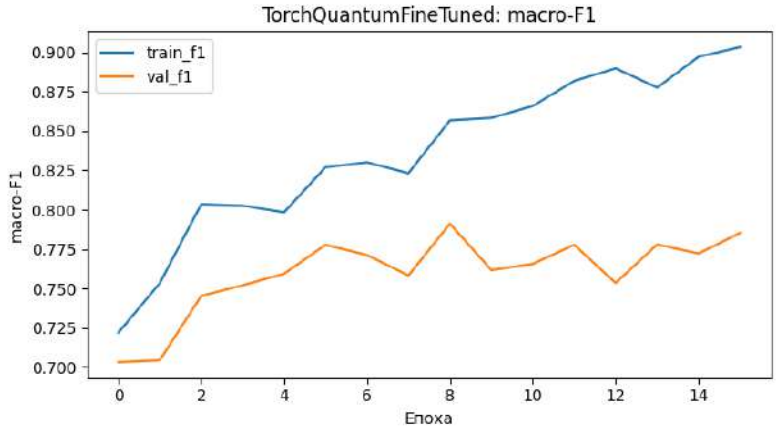


Рис. 3.7 Графік macro-F1 TorchQuantumFineTuned

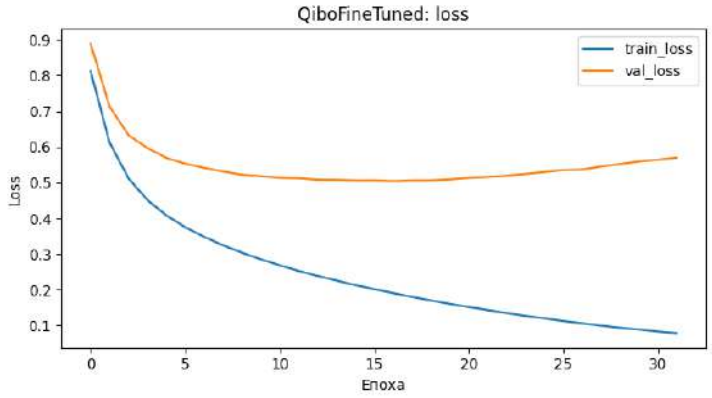


Рис. 3.8 Графік втрат модель QiboFineTuned

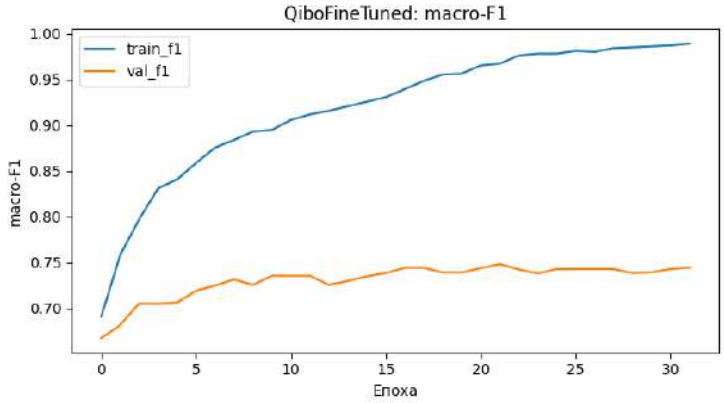


Рис. 3.9 Графік macro-F1 QiboFineTuned

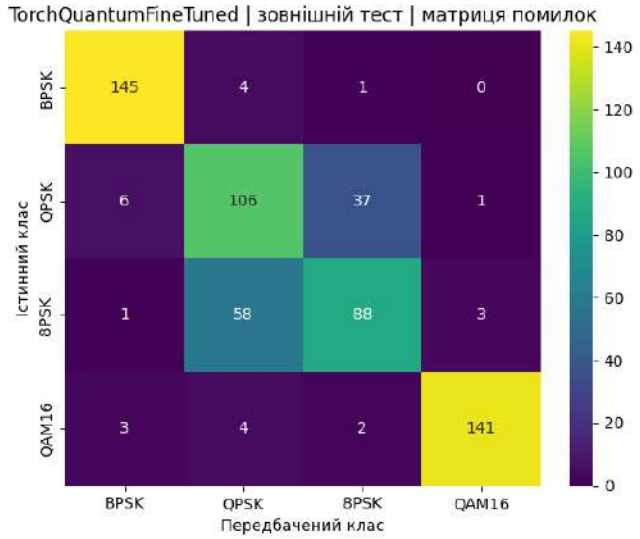


Рис. 3.10 Матриця помилок моделі TorchQuantumFineTuned на зовнішньому тесті

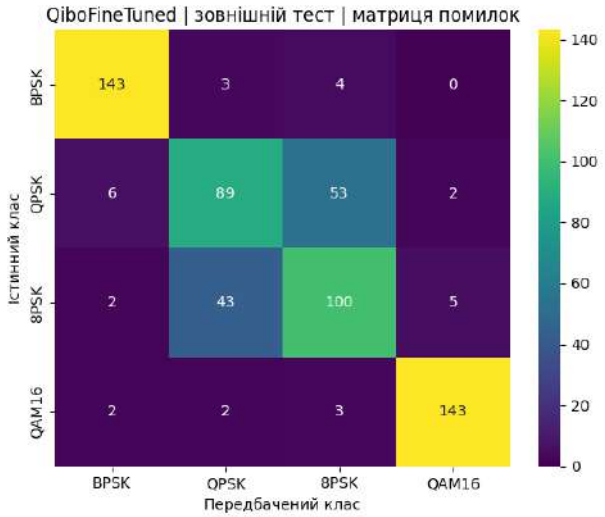


Рис. 3.11 Матриця помилок моделі QiboFineTuned на зовнішньому тесті