КВАЛІФІКАЦІЙНА РОБОТА

Апаратно-програмний комплекс навчання штучних нейронних мереж
Назва теми

Рівень вищої освіти *перший (бакалаврський)*

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

Шифр КвРКІ 2301121.23.01.21 ПЗ

Виконав здобувач III курсу, група КІ2с-23-1		Підпис	Дмитро РУДИЧУК Ініціали, прізвище
Керівник	д.т.н., проф Науковий ступінь, учене звання	Підпис	Євген ФЕДОРОВ Ініціали, прізвище
Нормоконтролер	д.т.н., проф Науковий ступінь, учене звання	Підпис	Сергій ЛИСЕНКО Ініціали, прізвище

До захисту допускаю:
завідувач кафедри КІІС
«01» червня 2026 р.
дата

Ольга ПАВЛОВА
Ініціали, прізвище

Хмельницький 2026

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

Рівень вищої освіти ПЕРШИЙ (БАКАЛАВРСЬКИЙ)

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

Освітня програма «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ ТА ПРОГРАМУВАННЯ»

ЗАТВЕРДЖУЮ
Завідувачка кафедри КІС
Ольга ПАВЛОВА

" 10 " 01 2026 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Рудичуку Дмитру Анатолійовичу
Прізвище, ім'я, по батькові студента

Тема проекту (роботи) Апаратно-програмний комплекс навчання штучних нейронних мереж

Керівник проекту (роботи) Федоров Євген Євгенович, д.т.н., професор
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 20.01.2026 р. № 7

2. Термін подання здобувачем роботи на кафедру 01.06.2026 р.

3. Вихідні дані до роботи Завдання на кваліфікаційну роботу

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Апаратно-програмний комплекс навчання штучних нейронних мереж та постановка задачі

Проєктування та розробка апаратно-програмного комплексу

Програмно-апаратна реалізація комплексу нвчання штучних нейронних мереж

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

Архітектура ПЗ проєкту

Панель керування комплексом

Апаратне забезпечення проєкту

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання « 10 » 01 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2026	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об`єкта та предмета дослідження	01.02.2026	виконано
3	Робота над розділом 1 – дослідження предметної області та постановка задачі	01.03.2026	виконано
4	Робота над розділом 2 – вибір компонентів та проектування апаратно-програмного комплексу навчання штучних нейронних мереж	01.04.2026	виконано
5	Робота над розділом 3 – програмно-апаратна реалізація апаратно-програмного комплексу навчання штучних нейронних мереж	29.04.2026	виконано
6	Оформлення пояснювальної записки згідно вимог	25.05.2026	виконано
7	Попередній захист ВКР	26.05.2026	виконано
8	Захист ВКР на засіданні ЕК	Червень 2026 року	

Здобувач _____ Дмитро РУДИЧУК
Підпис Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи Федоров Євген ФЕДОРОВ
Підпис Ім'я, ПРІЗВИЩЕ

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ 2301121.23.01.21 ПЗ	Пояснювальна записка	65		
			Графічні матеріали			
2		КвРКІ 2301121.23.01.21 Е8	Архітектура ПЗ проекту	1		
3		КвРКІ 2301121.23.01.21 Е8	Панель керування комплексом	1		
4		КвРКІ 2301121.23.01.21 Е8	Апаратне забезпечення проекту	1		

Зм	Арк	№ докум	Підпис	Дата	КвРКІ 2301121.23.01.21 ВП		
Розробив		Рудичук			Відомість проекту	Літера	Аркуш 1 / Аркушів 1
Перевір.		Федоров				У	
Н.контр.		Лисенко				ХНУ, КІ2с-23-1	
Затв.		Павленко		11.06			

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Апаратно-програмний комплекс навчання штучних нейронних мереж».

Автор роботи: Дмитро РУДИЧУК.

Керівник роботи: Євген ФЕДОРОВ.

Пояснювальна записка: 65 с., 11 рис., 2 табл., 3 дод., 50 джерел.

Графічна частина: 3 креслення.

АПАРАТНО-ПРОГРАМНИЙ КОМПЛЕКС, ГРАФІЧНИЙ ПРОЦЕСОР, ОБЧИСЛЕННЯ, ШТУЧНІ НЕЙРОННІ МЕРЕЖІ.

Кваліфікаційна робота бакалавра присвячена розробці та дослідженні апаратно-програмному комплексу навчання штучних нейронних мереж. Актуальність теми зумовлена потребою у проєктуванні спеціалізованих апаратно-програмних комплексів, які інтегрують графічні прискорювачі з оптимізованим системним та прикладним програмним забезпеченням. Створення збалансованих систем, що поєднують швидкісні інтерфейси NVMe, технології прямого доступу до пам'яті Peer-to-Peer DMA та програмні механізми асинхронного читання, дозволяє повністю усунути простої обчислювальних ядер, що й визначає актуальність розробки програмно-апаратних засобів комплексу навчання штучних нейронних мереж.

Метою дипломної роботи є забезпечення безперервного автоматизованого контролю, підвищення ефективності та швидкодії процесу навчання штучних нейронних мереж шляхом проєктування та практичної реалізації апаратно-програмного комплексу з оптимізованим конвеєром передачі та попередньої обробки даних у режимі реального часу.

Підпис здобувача

30.05.2026

Дата

ЗМІСТ

ВСТУП

Актуальність дослідження. Стрімкий розвиток технологій штучного інтелекту та штучних нейронних мереж у сучасних системах зумовив різке підвищення вимог до швидкодії обчислювальних платформ. Процес навчання сучасних моделей глибинного навчання оперує величезними масивами даних і потребує виконання значного обсягу матричних операцій, що визначає специфічні вимоги до систем.

Використання класичних центральних процесорів із послідовною архітектурою через обмежену кількість ядер уже не дозволяє забезпечити необхідну швидкодію в задачах розрахунку глибоких мереж. Перенесення математичного навантаження на графічні прискорювачі, які мають масивну паралельну структуру з тисячами обчислювальних ядер, дозволило значно прискорити матричні операції. Проте при масштабуванні обчислень виникає апаратне обмеження, коли надлишкова потужність одного компонента нівелюється обмеженнями іншого, зокрема затримками підсистеми пам'яті та дискового введення-виведення.

У зв'язку з цим виникає потреба у проєктуванні спеціалізованих апаратно-програмних комплексів, які інтегрують графічні прискорювачі з оптимізованим системним та прикладним програмним забезпеченням. Створення збалансованих систем, що поєднують швидкісні інтерфейси NVMe, технології прямого доступу до пам'яті Peer-to-Peer DMA та програмні механізми асинхронного читання, дозволяє повністю усунути простої обчислювальних ядер, що й визначає актуальність розробки програмно-апаратних засобів комплексу навчання штучних нейронних мереж.

Метою дипломної роботи є забезпечення безперервного автоматизованого контролю, підвищення ефективності та швидкодії процесу навчання штучних нейронних мереж шляхом проєктування та практичної реалізації апаратно-програмного комплексу з оптимізованим конвеєром

передачі та попередньої обробки даних у режимі реального часу.

Об’єкт дослідження – процеси збору, бездротового або внутрішньосистемного передавання, обробки, збереження та візуалізації телеметричних і структурованих даних у системах моніторингу та навчання нейронних мереж.

Предмет дослідження – методи програмно-апаратної інтеграції компонентів комплексу, алгоритми первинної обробки сенсорних чи пакетних даних та логіка функціонування правил обробки інформації й взаємодії між накопичувачем, оперативною пам'яттю та графічним прискорювачем.

1 АПАРАТНО-ПРОГРАМНИЙ КОМПЛЕКС НАВЧАННЯ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Особливості обчислювальних процесів при навчанні нейронних мереж

Штучні нейронні мережі (ШНМ) є одним із ключових інструментів сучасних інформаційних технологій, що застосовуються у задачах комп'ютерного зору, обробки природної мови, прогнозування та автоматизації. Основною особливістю їх функціонування є необхідність виконання значного обсягу обчислень, що визначає специфічні вимоги до обчислювальних систем.

Процес навчання нейронної мережі базується на багаторазовому проходженні навчальних даних через модель із подальшим коригуванням вагових коефіцієнтів. Основні етапи включають:

- пряме поширення сигналу;
- обчислення функції втрат;
- зворотне поширення помилки;
- оновлення ваг.

Ключовою математичною операцією при цьому є матричне множення, яке використовується для обчислення активацій нейронів.

Для глибоких нейронних мереж обсяг таких операцій зростає експоненційно зі збільшенням кількості шарів і нейронів.

Ще однією важливою особливістю є високий рівень паралелізму**.** Оскільки обчислення в межах одного шару можуть виконуватися незалежно для різних нейронів, це дозволяє ефективно використовувати багатоядерні та багатопотокові архітектури.

Також слід враховувати інтенсивну роботу з пам'яттю. При навчанні великих моделей необхідно зберігати:

- вагові коефіцієнти;
- градієнти;
- проміжні активації;

– навчальні дані.

Це створює значне навантаження на підсистему пам’яті та пропускну здатність шини.

Окремо варто виділити використання чисел з плаваючою комою. Найчастіше застосовуються формати FP32, FP16 або навіть INT8 (для оптимізації), що впливає на продуктивність і точність [4, 5].

Таким чином, навчання нейронних мереж характеризується:

– великим обсягом матричних операцій;

– високою паралельністю;

– значними вимогами до пам’яті;

– потребою в спеціалізованих обчислювальних архітектурах.

Окрім зазначених характеристик, важливу роль у сучасному навчанні нейронних мереж відіграє використання спеціалізованих апаратних прискорювачів, таких як графічні процесори та тензорні процесори [6, 7]. Їх архітектура оптимізована для виконання масово паралельних операцій, зокрема матричних множень і згорток, що дозволяє суттєво скоротити час навчання складних моделей порівняно з класичними центральними процесорами. У великих системах також застосовуються розподілені обчислення, коли навчання моделі виконується одночасно на кількох вузлах обчислювального кластера, що забезпечує масштабування на великі обсяги даних.

Важливим аспектом є також вибір алгоритмів оптимізації, які визначають швидкість і стабільність збіжності моделі. Найпоширенішими є стохастичний градієнтний спуск та його модифікації, такі як Adam або RMSProp, що дозволяють ефективніше оновлювати вагові коефіцієнти при роботі з великими наборами даних. Параметри, як-от розмір батчу, швидкість навчання та стратегія регуляризації, безпосередньо впливають на обчислювальне навантаження та вимоги до пам’яті.

Додатково слід враховувати, що зростання розміру моделей призводить до суттєвого збільшення енергоспоживання та вартості обчислень, що стає

критичним фактором при розгортанні великих нейромережевих систем у промислових умовах. У зв'язку з цим активно розвиваються методи оптимізації, спрямовані на зменшення розмірності моделей, квантизацію параметрів та використання змішаних форматів чисел з плаваючою комою, що дозволяє досягати балансу між точністю та ефективністю обчислень.

1.2 Огляд та порівняльний аналіз апаратних платформ

Ефективність навчання нейронних мереж значною мірою залежить від обраної апаратної платформи.

Сучасні рішення включають центральні процесори (CPU), графічні процесори (GPU), тензорні процесори (TPU), а також програмовані логічні інтегральні схеми (FPGA) та спеціалізовані ASIC [8, 9].

Центральні процесори CPU є універсальними обчислювальними пристроями, що забезпечують високу гнучкість. Вони мають складну архітектуру з кеш-пам'яттю, конвеєрами виконання та підтримкою SIMD-інструкцій (AVX, SSE).

Вони залишаються базовим компонентом будь-якої обчислювальної системи, включаючи системи машинного навчання та штучного інтелекту. Архітектура сучасних CPU є складною та багаторівневою, що дозволяє ефективно виконувати як послідовні, так і частково паралельні обчислення.

Сучасні центральні процесори оснащені багаторівневою кеш-пам'яттю (L1, L2, L3), яка забезпечує швидкий доступ до часто використовуваних даних та інструкцій. Це суттєво зменшує затримки, пов'язані з доступом до оперативної пам'яті. Крім того, CPU використовують механізми конвеєризації (pipeline), що дозволяє одночасно виконувати кілька етапів різних інструкцій, підвищуючи загальну продуктивність системи.

Важливу роль у підвищенні ефективності обчислень відіграє підтримка SIMD-інструкцій, таких як AVX (Advanced Vector Extensions) та SSE (Streaming

SIMD Extensions). Ці інструкції дозволяють виконувати одну й ту саму операцію одночасно над кількома елементами даних, що є особливо корисним для задач обробки сигналів, зображень та базових операцій лінійної алгебри. Проте, навіть з урахуванням SIMD, рівень паралелізму CPU значно поступається спеціалізованим прискорювачам, таким як GPU.

З точки зору організації обчислень, сучасні CPU мають обмежену кількість ядер (зазвичай від кількох до кількох десятків), кожне з яких є високопродуктивним та оптимізованим для виконання складних послідовних інструкцій [10]. Також широко застосовується технологія багатопотоковості, наприклад Intel Hyper-Threading, що дозволяє одному фізичному ядру обробляти кілька потоків інструкцій. Це підвищує ефективність використання обчислювальних ресурсів, однак не забезпечує рівня паралелізму, характерного для графічних процесорів.

До основних переваг центральних процесорів належать:

- універсальність, що дозволяє використовувати їх для вирішення широкого класу задач без необхідності зміни апаратної конфігурації;
- висока точність обчислень, зокрема підтримка операцій з плаваючою комою подвійної точності (FP64);
- розвинена екосистема програмного забезпечення та інструментів розробки;
- відносна простота програмування порівняно зі спеціалізованими архітектурами.

Водночас CPU мають і низку суттєвих недоліків у контексті навчання нейронних мереж. Основним обмеженням є відносно низький рівень паралелізму, що зумовлений невеликою кількістю обчислювальних ядер. Це призводить до того, що операції, які можуть виконуватися паралельно (наприклад, множення великих матриць), виконуються менш ефективно порівняно з GPU або TPU.

Ще одним важливим недоліком є низька ефективність при виконанні

масових матричних операцій, які становлять основу алгоритмів глибокого навчання. Хоча SIMD-інструкції частково компенсують цей недолік, вони не здатні забезпечити необхідний рівень продуктивності для великих моделей.

У системах машинного навчання CPU зазвичай виконують допоміжні та керуючі функції. Зокрема, вони використовуються для попередньої обробки даних, що включає очищення, нормалізацію, аугментацію та перетворення форматів даних. Також CPU відповідають за організацію процесу навчання, включаючи завантаження даних, керування пам’яттю та взаємодію з іншими обчислювальними пристроями.

Крім того, центральні процесори можуть ефективно застосовуватися для навчання невеликих моделей або у випадках, коли використання спеціалізованих прискорювачів є недоцільним через обмеження ресурсів або енергоспоживання. У вбудованих системах та edge-пристроях CPU часто виступають єдиним доступним обчислювальним ресурсом, що зумовлює необхідність оптимізації алгоритмів під їх архітектуру.

Графічні процесори (GPU) є спеціалізованими обчислювальними пристроями, спочатку розробленими для обробки графіки, однак з часом вони стали основною апаратною платформою для виконання задач глибокого навчання та штучного інтелекту [12]. Це зумовлено їх архітектурними особливостями, які забезпечують високий рівень паралелізму та ефективність при виконанні однотипних обчислювальних операцій.

На відміну від центральних процесорів, які оптимізовані для виконання складних послідовних інструкцій, GPU орієнтовані на обробку великої кількості простих операцій одночасно. Архітектура графічного процесора включає сотні або тисячі обчислювальних ядер, які організовані у мультипроцесори та здатні виконувати паралельні потоки інструкцій. Такий підхід дозволяє досягати значної продуктивності при обробці великих масивів даних.

Ключовою особливістю GPU є використання моделі паралельних

обчислень, що передбачає виконання однієї інструкції над множиною даних. У контексті програмування GPU це реалізується через організацію обчислень у вигляді груп потоків та блоків потоків, що дозволяє ефективно розподіляти задачі між обчислювальними ядрами. Така модель добре узгоджується з природою обчислень у нейронних мережах, де одні й ті самі операції виконуються над великими матрицями.

Значну роль у забезпеченні високої продуктивності відіграє підсистема пам'яті GPU. Вона включає кілька рівнів:

- глобальну пам'ять, яка має великий обсяг, але відносно високу затримку;
- спільну пам'ять, що використовується для швидкого обміну даними між потоками одного блоку;
- регістри, які забезпечують найшвидший доступ до даних.

Правильна організація доступу до пам'яті є критично важливою для досягнення високої продуктивності, оскільки неефективне використання пам'яті може нівелювати переваги паралельної архітектури.

У задачах навчання нейронних мереж GPU демонструють значну перевагу над CPU, особливо при виконанні операцій лінійної алгебри, таких як множення великих матриць та тензорів. Це пов'язано з тим, що такі операції можуть бути розбиті на велику кількість незалежних підзадач, які виконуються паралельно.

До основних переваг графічних процесорів належать:

- масивна паралельність, що дозволяє виконувати тисячі операцій одночасно;
- висока пропускна здатність пам'яті;
- оптимізація під матричні та тензорні обчислення;
- підтримка спеціалізованих обчислювальних блоків (наприклад, Tensor Cores).

Tensor Cores є окремими функціональними блоками сучасних GPU, які

оптимізовані для виконання операцій над матрицями з використанням зниженої точності (FP16, BF16). Це дозволяє значно прискорити навчання нейронних мереж без суттєвої втрати точності.

Однак, незважаючи на високу продуктивність, GPU мають і ряд недоліків. Одним із основних є високе енергоспоживання, що робить їх використання менш ефективним у системах з обмеженими енергетичними ресурсами. Крім того, програмування GPU на низькому рівні є складнішим порівняно з CPU, оскільки вимагає врахування специфіки паралельної архітектури та ієрархії пам’яті.

Ще одним обмеженням є залежність від спеціалізованого програмного забезпечення, такого як CUDA або ROCm, що може обмежувати переносимість програм між різними апаратними платформами.

У практичних системах машинного навчання GPU використовуються як основний обчислювальний ресурс для навчання моделей. Вони забезпечують значне скорочення часу навчання, що особливо важливо при роботі з великими датасетами та глибокими нейронними мережами.

Крім того, GPU широко застосовуються у хмарних обчисленнях та високопродуктивних обчислювальних кластерах, де вони дозволяють масштабувати процес навчання шляхом використання кількох пристроїв одночасно. Для цього використовуються технології розподіленого навчання, які дозволяють ефективно розподіляти обчислювальне навантаження.

Тензорні процесори (TPU) є спеціалізованими обчислювальними пристроями, розробленими для прискорення задач машинного навчання, зокрема навчання та інференсу нейронних мереж. Вони належать до класу ASIC, тобто інтегральних схем, оптимізованих для виконання вузькоспеціалізованих задач. Основною метою створення TPU є досягнення максимальної продуктивності та енергоефективності при виконанні тензорних операцій, які лежать в основі алгоритмів глибокого навчання.

На відміну від центральних та графічних процесорів, TPU спроєктовані з

урахуванням специфіки обчислень у нейронних мережах. Основною операцією, яку вони оптимізують, є множення матриць великої розмірності. Для цього використовується архітектура так званих систолічних масивів, яка дозволяє організувати ефективне потокове виконання обчислень.

Систолічний масив являє собою двовимірну решітку обчислювальних елементів, кожен з яких виконує прості операції множення та додавання. Дані передаються між елементами у вигляді потоків, що дозволяє мінімізувати затримки доступу до пам’яті та забезпечити високу пропускну здатність. Такий підхід є особливо ефективним для операцій типу «матриця * матриця», які домінують у процесі навчання нейронних мереж.

Однією з ключових особливостей TPU є використання зниженої точності обчислень, зокрема форматів BF16 або INT8. Це дозволяє значно збільшити швидкість обчислень та зменшити енергоспоживання без істотної втрати точності результатів. Використання таких форматів стало можливим завдяки тому, що нейронні мережі є відносно стійкими до незначних похибок у числових обчисленнях.

Архітектура TPU також передбачає ефективну інтеграцію з пам’яттю. На відміну від GPU, де значна частина часу витрачається на обмін даними між пам’яттю та обчислювальними ядрами, TPU оптимізовані для зменшення кількості таких операцій. Це досягається шляхом використання локальних буферів та потокової обробки даних.

До основних переваг тензорних процесорів належать:

- надвисока продуктивність при виконанні тензорних операцій;
- висока енергоефективність;
- оптимізація під задачі глибокого навчання;
- ефективна робота з великими матрицями.

Особливої ефективності TPU досягають у задачах, де обчислення можуть бути добре формалізовані у вигляді великих матричних операцій, наприклад у згорткових або трансформерних нейронних мережах.

Разом з тим, TPU мають і певні обмеження. Основним недоліком є їх вузька спеціалізація. Вони не є універсальними обчислювальними пристроями і не підходять для виконання довільних програм або задач, які виходять за межі машинного навчання.

Ще одним обмеженням є залежність від програмної екосистеми. TPU тісно інтегровані з певними програмними фреймворками, зокрема TensorFlow, що може обмежувати їх використання в інших середовищах. Крім того, доступ до TPU зазвичай здійснюється через хмарні сервіси, що може створювати додаткові вимоги до інфраструктури.

У практичних системах TPU використовуються переважно у великих дата-центрах та хмарних платформах, де вони забезпечують високу продуктивність при навчанні масштабних моделей. Вони дозволяють значно скоротити час навчання та зменшити витрати на електроенергію, що є критично важливим для великих обчислювальних систем.

Також TPU активно застосовуються для інференсу, особливо у випадках, коли необхідно обробляти великі обсяги даних у реальному часі. Завдяки високій продуктивності та низькому енергоспоживанню вони є ефективним рішенням для масштабованих систем штучного інтелекту.

Програмовані логічні інтегральні схеми (FPGA, Field-Programmable Gate Array) є класом апаратних пристроїв, що дозволяють реалізовувати цифрові обчислювальні системи з можливістю повторної конфігурації після виробництва [21]. На відміну від центральних та графічних процесорів, які мають фіксовану архітектуру, FPGA надають змогу розробнику створювати власну апаратну логіку, оптимізовану під конкретну задачу, зокрема для прискорення алгоритмів навчання нейронних мереж.

Архітектура FPGA базується на масиві програмованих логічних блоків, які з’єднані між собою за допомогою програмованих комутаційних ресурсів. Кожен логічний блок може реалізовувати базові логічні функції, а їх комбінація дозволяє створювати складні цифрові схеми [22]. Крім того, сучасні FPGA

містять вбудовані ресурси, такі як:

- блоки цифрової обробки сигналів (DSP-блоки);
- блоки пам’яті (Block RAM);
- високошвидкісні інтерфейси вводу-виводу;
- апаратні множники та акумулятори.

Ці компоненти роблять FPGA придатними для реалізації обчислювально інтенсивних задач, характерних для нейронних мереж.

Однією з ключових особливостей FPGA є можливість реалізації апаратного паралелізму на дуже глибокому рівні. На відміну від CPU та GPU, де паралелізм досягається за рахунок виконання інструкцій, у FPGA паралелізм реалізується безпосередньо на рівні апаратної логіки. Це означає, що різні обчислювальні блоки можуть працювати одночасно без необхідності керування потоками або планування задач.

У задачах машинного навчання FPGA часто використовуються для реалізації прискорювачів, які оптимізовані під конкретні моделі або типи операцій. Наприклад, можна створити апаратну реалізацію операції множення матриць або згортки, яка буде максимально ефективною з точки зору використання ресурсів та енергоспоживання.

Ще однією важливою перевагою FPGA є можливість налаштування точності обчислень. Розробник може використовувати довільні формати чисел (fixed-point, floating-point різної розрядності), що дозволяє досягти оптимального балансу між точністю та продуктивністю. Це особливо важливо для нейронних мереж, де часто допускається зниження точності без суттєвого впливу на якість результатів.

До основних переваг FPGA належать:

- висока гнучкість на апаратному рівні;
- можливість оптимізації під конкретний алгоритм або модель;
- низьке енергоспоживання порівняно з GPU;
- детермінований час виконання операцій (відсутність

непередбачуваних затримок).

Завдяки цим властивостям FPGA широко застосовуються у вбудованих системах, системах реального часу та edge-обчисленнях, де важливими є енергоефективність та передбачуваність роботи.

Водночас використання FPGA пов’язане з рядом складнощів. Основним недоліком є висока складність розробки. Проектування систем на FPGA зазвичай здійснюється з використанням мов опису апаратури (HDL), таких як VHDL або Verilog, що потребує глибоких знань у галузі цифрової схемотехніки. Крім того, цикл розробки включає синтез, розміщення та трасування, що може займати значний час.

Ще одним обмеженням є відносно менша продуктивність у задачах загального призначення порівняно з GPU. Хоча FPGA можуть бути дуже ефективними для конкретних задач, їх універсальність є обмеженою.

Також слід зазначити, що ефективність використання FPGA значною мірою залежить від якості розробленої апаратної архітектури. Невдале проектування може призвести до нераціонального використання ресурсів та зниження продуктивності.

У сучасних системах машинного навчання FPGA часто використовуються як проміжне рішення між універсальними процесорами та спеціалізованими ASIC. Вони дозволяють швидко прототипувати нові архітектури прискорювачів та адаптувати їх до змін у моделях нейронних мереж.

ASIC (Application-Specific Integrated Circuit) - це спеціалізовані інтегральні схеми, створені для виконання строго визначених обчислювальних задач, на відміну від універсальних процесорів, таких як CPU або навіть GPU [24]. Головною особливістю ASIC є те, що їх архітектура проєктується під конкретний тип алгоритмів або навіть під одну конкретну задачу, що дозволяє досягти максимальної продуктивності та енергоефективності. Завдяки такому підходу значно зменшується кількість зайвих обчислювальних блоків, які присутні в універсальних процесорах, а також оптимізуються шляхи передачі

даних всередині кристалу.

У контексті штучних нейронних мереж ASIC часто використовуються для прискорення операцій матричного множення, згорткових операцій та обчислення активацій, які є базовими елементами глибокого навчання [25]. Такі спеціалізовані чипи можуть містити вбудовані матричні множники, тензорні ядра або інші апаратні блоки, що дозволяють виконувати тисячі або навіть мільйони операцій паралельно. Це забезпечує суттєве зниження затримок під час навчання та інференсу моделей у порівнянні з традиційними обчислювальними системами.

Ще однією важливою перевагою ASIC є їх енергоефективність. Оскільки архітектура оптимізується під конкретні обчислення, споживання електроенергії на одну операцію значно менше, ніж у загальнопризначених процесорах. Це особливо критично для великих центрів обробки даних, де навчання нейронних мереж може тривати тижнями і вимагати значних енергетичних ресурсів. У таких умовах використання ASIC дозволяє суттєво зменшити експлуатаційні витрати та підвищити загальну продуктивність системи.

Однак ASIC мають і певні обмеження. Найголовніше з них, це відсутність гнучкості. Оскільки такі мікросхеми створюються під конкретний алгоритм або клас задач, їх неможливо легко перепрограмувати для виконання інших типів обчислень. Це означає, що при зміні архітектури нейронної мережі або появі нових алгоритмів може виникнути необхідність розробки нового чипа, що є дорогим і тривалим процесом. Саме тому ASIC найчастіше застосовуються у стабільних, добре стандартизованих задачах, де алгоритми вже достатньо відомі та не потребують частих змін.

Універсальні процесори забезпечують гнучкість і зручність програмування, однак поступаються за продуктивністю у виконанні масово паралельних обчислень.

Графічні процесори, у свою чергу, стали стандартом для навчання

нейронних мереж завдяки своїй здатності ефективно виконувати матричні операції та обробляти великі обсяги даних паралельно.

Програмовані логічні інтегральні схеми пропонують компроміс між гнучкістю та продуктивністю, дозволяючи адаптувати апаратну архітектуру під конкретні задачі, хоча й потребують складнішого процесу розробки.

Спеціалізовані інтегральні схеми забезпечують найвищу продуктивність і енергоефективність завдяки вузькій спеціалізації, однак мають обмежену універсальність і високу вартість розробки. У сучасних умовах спостерігається тенденція до комбінованого використання різних типів обчислювальних платформ, що дозволяє досягти оптимального балансу між гнучкістю, швидкодією та енергоспоживанням. Порівняльний аналіз платформ приведений в таблиці 1.1. Отже, проведений огляд апаратних платформ для обчислень у задачах штучних нейронних мереж дозволяє зробити висновок, що ефективність їх функціонування безпосередньо залежить від відповідності архітектури обчислювальної системи характеру виконуваних операцій. Сучасні нейромережеві задачі характеризуються високою обчислювальною складністю, значним рівнем паралелізму та інтенсивним обміном даними, що висуває підвищені вимоги до продуктивності, пропускної здатності пам’яті та енергоефективності апаратного забезпечення.

Таблиця 1.1 – Порівняльний аналіз апаратних платформ

Платформа	Продуктивність	Гнучкість	Енергоспоживання	Складність
CPU	Низька	Висока	Середнє	Низька
GPU	Висока	Середня	Високе	Середня
TPU	Дуже висока	Низька	Низьке	Висока
FPGA	Середня	Висока	Низьке	Дуже висока
ASIC	Максимальна	Мінімальна	Дуже низьке	Дуже

				висока

Таким чином, вибір апаратної платформи для реалізації нейронних мереж є складним завданням, яке повинно враховувати специфіку задачі, обсяги даних, вимоги до швидкодії та доступні ресурси. Правильний підбір і поєднання апаратних засобів є ключовим фактором для забезпечення ефективності сучасних систем штучного інтелекту.

1.3 Огляд програмного забезпечення та фреймворків для систем навчання штучних нейронних мереж

Програмна складова відіграє ключову роль у реалізації систем навчання нейронних мереж. Вона включає як низькорівневі інструменти, так і високорівневі бібліотеки. Такий поділ обумовлений різним ступенем абстракції та контролю над обчислювальними ресурсами. Низькорівневі технології забезпечують безпосередній доступ до апаратного забезпечення, зокрема графічних процесорів, дозволяючи максимально ефективно використовувати їх обчислювальний потенціал через тонке налаштування паралельних процесів і управління пам’яттю. Водночас високорівневі фреймворки орієнтовані на спрощення процесу розробки, надаючи готові інструменти, абстракції та бібліотеки для швидкого створення, навчання та тестування моделей нейронних мереж. Вони приховують складність низькорівневих операцій, дозволяючи розробникам зосередитися на логіці моделі та експериментах.

Розглянемо спочатку низькорівневі технології.

CUDA – це програмно-апаратна платформа, розроблена компанією NVIDIA, яка забезпечує можливість використання графічних процесорів для виконання загальноцільових обчислень (GPGPU). Вона є одним із ключових інструментів у сфері глибокого навчання, оскільки дозволяє максимально ефективно реалізувати паралельні обчислення, що лежать в основі роботи нейронних мереж [25].

Архітектурно CUDA базується на моделі SIMD (Single Instruction, Multiple Data), де тисячі потоків можуть одночасно виконувати однакові операції над різними даними [26]. Це ідеально підходить для матричних множень, згорткових операцій та інших базових обчислень у нейронних мережах. CUDA надає розробникам можливість працювати з GPU на низькому рівні, керуючи потоками, блоками потоків, а також ієрархією пам'яті, що включає глобальну, спільну, регістрову та кеш-пам'ять.

Однією з ключових переваг CUDA є можливість тонкої оптимізації продуктивності. Розробник може явно контролювати розміщення даних у пам'яті, мінімізувати затримки доступу та ефективно використовувати пропускну здатність пам'яті. Крім того, CUDA підтримує використання спеціалізованих бібліотек, таких як cuBLAS (для лінійної алгебри), cuDNN (для глибокого навчання), що значно спрощує реалізацію складних алгоритмів.

Важливим аспектом є також інтеграція CUDA з високорівневими фреймворками, такими як PyTorch та TensorFlow. Це дозволяє розробникам використовувати апаратні ресурси GPU без необхідності глибокого занурення у низькорівневе програмування. Водночас CUDA має обмеження у вигляді прив'язки до GPU від NVIDIA, що зменшує її універсальність.

ROCm – це відкрита програмна платформа для обчислень на GPU, розроблена компанією AMD як альтернатива CUDA. ROCm орієнтована на високопродуктивні обчислення (HPC) та задачі машинного навчання, забезпечуючи гнучкість і відкритість екосистеми.

Однією з головних особливостей ROCm є її відкритий код, що дозволяє дослідникам і розробникам модифікувати платформу під власні потреби, а також сприяє прозорості та незалежності від конкретного виробника. ROCm підтримує сучасні стандарти програмування, зокрема HIP (Heterogeneous-computing Interface for Portability), який дозволяє переносити код між CUDA та ROCm з мінімальними змінами.

Платформа забезпечує доступ до апаратних ресурсів GPU AMD на

низькому рівні, включаючи управління потоками, пам’яттю та обчислювальними блоками. ROCm також включає набір оптимізованих бібліотек, аналогічних до CUDA, таких як rocBLAS та MIOpen, що використовуються для реалізації алгоритмів машинного навчання.

Ще однією важливою перевагою ROCm є її орієнтація на масштабовані обчислювальні системи. Вона активно використовується у суперкомп’ютерах та дата-центрах, де важливими є відкритість, сумісність і можливість інтеграції з іншими інструментами. Проте, порівняно з CUDA, ROCm має дещо меншу підтримку з боку спільноти та обмеженішу кількість сумісних пристроїв.

Розглянемо високорівневі фреймворки.

PyTorch – це сучасний фреймворк для розробки моделей машинного навчання, розроблений організацією Meta AI. Він здобув широку популярність завдяки своїй гнучкості, інтуїтивному інтерфейсу та зручності використання, особливо в наукових дослідженнях [27].

Однією з ключових особливостей PyTorch є використання динамічних обчислювальних графів. Це означає, що граф обчислень будується під час виконання програми, що значно спрощує налагодження та дозволяє гнучко змінювати структуру моделі. Такий підхід є особливо корисним при роботі з нестандартними архітектурами нейронних мереж або дослідницькими задачами.

PyTorch надає потужний механізм автоматичного диференціювання, який автоматично обчислює градієнти функцій, необхідні для навчання моделей [28]. Це дозволяє розробникам зосередитися на побудові моделей, а не на ручному виведенні похідних. Крім того, PyTorch тісно інтегрований з Python, що робить його доступним для широкого кола користувачів.

Фреймворк також підтримує ефективну роботу з GPU через CUDA, що дозволяє значно прискорити навчання моделей. Розширена екосистема, включаючи бібліотеки torchvision, torchaudio та torchtext, забезпечує готові рішення для роботи з різними типами даних.

Важливу роль відіграє активна спільнота PyTorch, яка постійно розвиває інструменти, публікує нові моделі та забезпечує підтримку користувачів. Завдяки цьому PyTorch став одним із стандартів у дослідженнях штучного інтелекту.

TensorFlow – це потужна платформа для машинного навчання, розроблена компанією Google. Вона орієнтована як на дослідження, так і на промислове застосування, забезпечуючи широкий спектр інструментів для побудови, навчання та розгортання моделей [29].

TensorFlow використовує концепцію обчислювальних графів, де операції представлені у вигляді вузлів, а дані у вигляді тензорів, що передаються між ними. У ранніх версіях графи були статичними, що забезпечувало високу продуктивність і оптимізацію, однак ускладнювало налагодження. Згодом було впроваджено режим негайного виконання, який зробив TensorFlow більш гнучким і схожим на PyTorch [30].

Однією з ключових переваг TensorFlow є його масштабованість. Він підтримує розподілене навчання на кластерах, використання GPU та спеціалізованих процесорів TPU (Tensor Processing Unit), що дозволяє ефективно працювати з великими обсягами даних і складними моделями. Крім того, TensorFlow надає інструменти для розгортання моделей у продакшені, зокрема TensorFlow Serving та TensorFlow Lite для мобільних і вбудованих систем.

Фреймворк має розвинену екосистему, включаючи Keras як високорівневий API для швидкої розробки моделей, а також численні бібліотеки для обробки даних, візуалізації та моніторингу процесу навчання. Це робить TensorFlow універсальним рішенням для повного життєвого циклу розробки систем штучного інтелекту.

Таким чином, сучасні програмні засоби для навчання нейронних мереж охоплюють широкий спектр інструментів від низькорівневих платформ, що забезпечують максимальний контроль над апаратними ресурсами, до

високорівневих фреймворків, які значно спрощують розробку та впровадження моделей, забезпечуючи при цьому високу продуктивність і масштабованість.

Оптимізація моделей штучних нейронних мереж є важливою частиною їх практичного використання, особливо коли мова йде про обмежені обчислювальні ресурси або необхідність швидкої роботи системи. Після навчання модель часто має велику кількість параметрів і потребує значних обсягів пам’яті та обчислень. Це ускладнює її використання на реальних пристроях, таких як сервери з великою кількістю запитів, мобільні телефони або вбудовані системи. Саме тому застосовуються спеціальні підходи, які дозволяють зменшити розмір моделі та прискорити її роботу без суттєвої втрати точності.

Одним із таких підходів є квантизація моделей. Суть цього методу полягає у зменшенні точності чисел, які використовуються для представлення ваг і обчислень. Замість стандартних чисел з плаваючою комою застосовуються більш прості формати, наприклад цілі числа. Це дозволяє зменшити обсяг пам’яті, необхідний для зберігання моделі, а також пришвидшити обчислення, оскільки операції з такими числами виконуються швидше.

Квантизація може виконуватися як після завершення навчання моделі, так і під час нього. У першому випадку модель просто перетворюється у більш компактний вигляд, у другому вона одразу навчається з урахуванням обмеженої точності, що дозволяє краще зберегти якість результатів. Такий підхід широко використовується в системах, де важлива швидкодія та економія ресурсів, наприклад у мобільних додатках або пристроях з обмеженим енергоспоживанням.

Іншим методом оптимізації є прунинг, або «обрізання» моделі. Його ідея полягає у видаленні зайвих або малозначущих параметрів, які мало впливають на результат роботи нейронної мережі. Під час навчання багато зв’язків між нейронами можуть мати дуже малі значення, і їх внесок у підсумковий результат є незначним. Такі зв’язки можна видалити, що призводить до

зменшення розміру моделі та кількості обчислень. Прунинг може виконуватися поступово, коли після кожного етапу навчання частина параметрів видаляється, або після завершення навчання. Після цього модель часто донавчають, щоб відновити точність. У результаті отримується більш компактна структура, яка працює швидше та потребує менше пам’яті. Такий підхід є корисним для систем, де необхідно обробляти велику кількість запитів або працювати в режимі реального часу.

Окрему роль відіграють спеціальні компілятори, які оптимізують виконання моделей на конкретному обладнанні. До таких інструментів належать TensorRT та XLA [32]. Вони аналізують структуру моделі та перетворюють її у більш ефективну форму для виконання. Наприклад, можуть об’єднувати кілька операцій в одну, змінювати порядок обчислень або використовувати спеціальні можливості процесора чи графічного прискорювача. Це дозволяє зменшити час виконання моделі та підвищити ефективність використання ресурсів.

TensorRT орієнтований на роботу з графічними процесорами і використовується для прискорення виконання моделей, особливо на етапі застосування, коли модель вже навчена. Він дозволяє автоматично виконувати квантизацію, оптимізувати використання пам’яті та підбирати найбільш ефективні способи виконання операцій. У свою чергу, XLA застосовується разом із фреймворками машинного навчання і виконує оптимізацію обчислень ще на етапі підготовки моделі. Він перетворює послідовність операцій у більш компактну та швидку форму, що дозволяє скоротити кількість звернень до пам’яті та підвищити швидкість виконання.

Застосування зазначених підходів у комплексі дозволяє досягти помітного покращення ефективності роботи нейронних мереж. Зменшується обсяг пам’яті, скорочується час обробки даних, а також знижується навантаження на апаратні ресурси. Це особливо важливо в умовах, коли системи повинні працювати стабільно, швидко та з обмеженими ресурсами.

Таким чином, оптимізація моделей є необхідним етапом переходу від експериментальних розробок до практичного використання нейронних мереж у реальних задачах.

1.4 Постановка задачі

Для досягнення поставленої мети необхідно вирішити такі наступні задачі.

1. Провести аналіз існуючих апаратних та програмних рішень для навчання штучних нейронних мереж.
2. Обґрунтувати обрану архітектуру апаратно-програмного комплекса навчання штучних нейронних мереж, вибрати апаратну та програмну платформи для вирішення поставленої задачі.
3. Розробити алгоритмічне та програмне забезпечення апаратно-програмного комплексу навчання штучних нейронних мереж.
4. Провести експериментальне тестування запропонованих рішень та оцінити її швидкодію.

1.5 Висновки до першого розділу

У межах розділу 1 проведено аналіз предметної області досліджено особливості обчислень при роботі зі штучними нейронними мережами та проведено аналіз платформ для їх навчання. Навчання таких моделей потребує виконання дуже великої кількості однотипних математичних операцій, зокрема множення матриць. Через це звичайні центральні процесори не можуть забезпечити потрібну швидкість обробки даних, адже вони мають порівняно мало ядер і розраховані на послідовне виконання завдань.

Порівняння різних апаратних платформ (центральних, графічних, тензорних процесорів та програмованих схем) показало, що найбільш

універсальним та доступним рішенням є графічні процесори. Вони мають тисячі дрібних ядер, які дозволяють виконувати багато обчислень одночасно. Також розглянуто сучасне програмне забезпечення: низькорівневі інструменти керування обладнанням та високорівневі готові середовища розробки, такі як TensorFlow і PyTorch. Зроблено висновок, що для розв'язання поставленої задачі необхідно створити збалансовану систему, де компоненти підібрані так, щоб швидкість передачі даних відповідала потужності обчислювального пристрою.

2 ПРОЄКТУВАННЯ ТА РОЗРОБКА АПАРАТНО_ПРОГРАМНОГО КОМПЛЕКСУ

2.1 Архітектурні рішення та конфігурація апаратної частини

Сучасний етап розвитку систем штучного інтелекту характеризується експоненційним зростанням обсягів даних та складності архітектур штучних нейронних мереж.

Як наслідок, традиційні обчислювальні системи на базі універсальних центральних процесорів перестають задовольняти вимогам продуктивності в задачах навчання моделей deep learning. Ефективне вирішення цієї проблеми лежить у площині проектування спеціалізованих апаратно-програмних комплексів (АПК), що інтегрують високопаралельні графічні прискорювачі або спеціалізовані тензорні процесори з оптимізованим системним та прикладним програмним забезпеченням.

Проектування апаратної архітектури АПК потребує вирішення проблеми вузьких місць, коли надлишкова потужність одного компонента нівелюється обмеженнями іншого. В таких задачах важливо забезпечити збалансовану пропускну здатність між центральним процесором, оперативною пам'яттю, інтерфейсними шинами та графічним прискорювачем.

В якості базового обчислювального прискорювача для проектованого комплексу обрано графічний процесор архітектури NVIDIA Ampere (модель класу RTX 3090/4090 або спеціалізована A100/A10), оскільки вона володіє спеціалізованими апаратними блоками тензорними ядрами третього або четвертого покоління [33, 34].

На відміну від стандартних CUDA-ядер, які виконують одну операцію умноження-додавання (FMA) за такт для скалярних величин, тензорні ядра оперують цілими матрицями розміром 4×4 за один тактовий імпульс.

Архітектура графічних процесорів покоління Ampere компанії NVIDIA є розвитком попередніх архітектур Volta та Turing і орієнтована на задачі

високопродуктивних обчислень, машинного навчання, комп'ютерного зору та обробки великих масивів даних. До представників даної архітектури належать графічні процесори серій GeForce RTX 30xx, професійні прискорювачі A10, а також серверні рішення A100.

Саме матричні обчислення є базою для сучасних алгоритмів штучного інтелекту, нейронних мереж і більшості задач лінійної алгебри.

Типова структура графічного процесора архітектури Ampere наведена на рисунку 2.1.

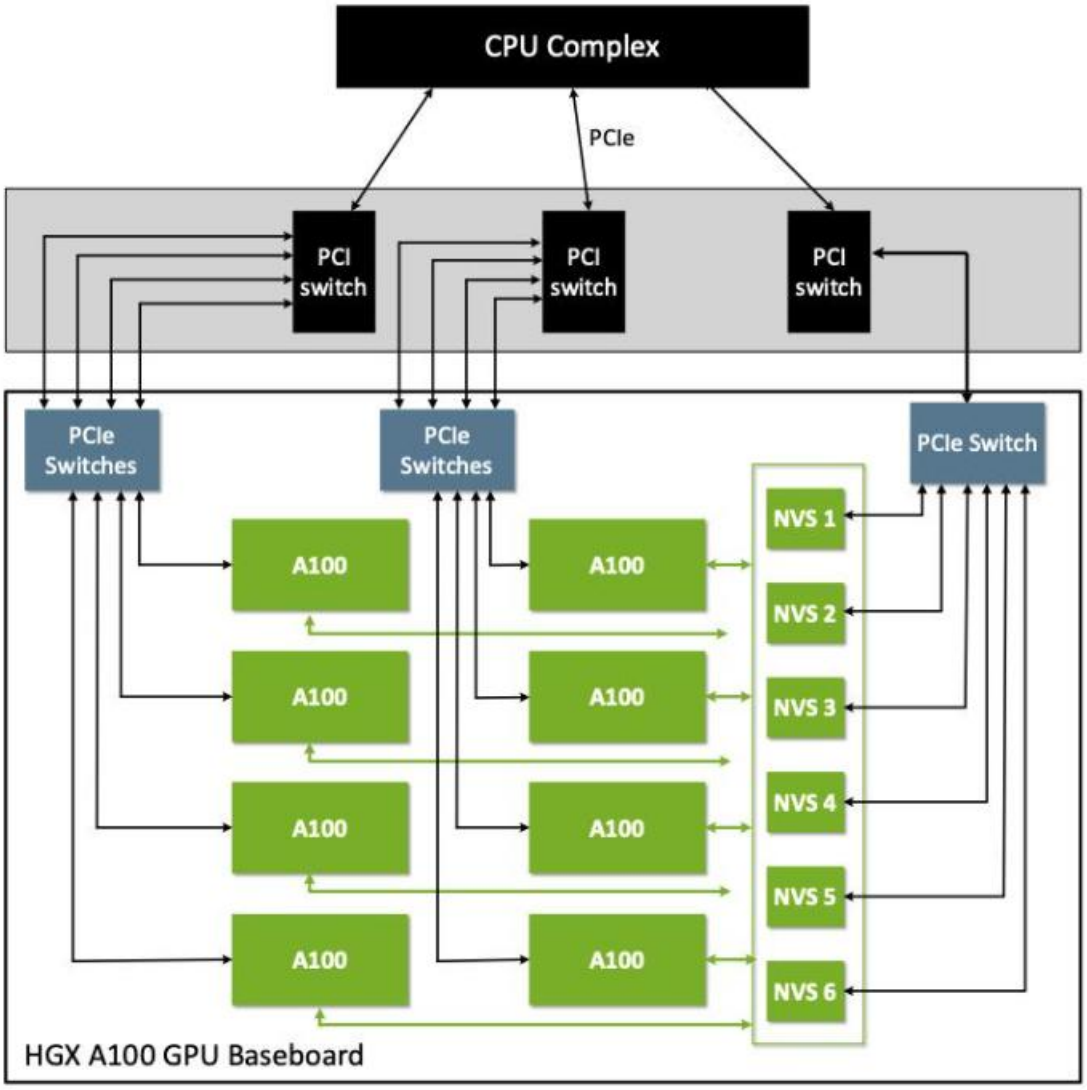


Рисунок 2.1 – Типова структура графічного процесора [35]

Архітектурно GPU складається з набору потокових мультипроцесорів

(Streaming Multiprocessor, SM), які функціонують паралельно. Кожен SM містить:

- CUDA-ядра;
- Tensor Cores;
- блоки роботи з пам’яттю;
- кеш-пам’ять;
- планувальники потоків.

CUDA-ядра виконують класичні скалярні операції з плаваючою комою, тоді як Tensor Cores орієнтовані на прискорення операцій матричного множення та накопичення результатів.

CUDA-ядра є універсальними арифметико-логічними пристроями (ALU), призначеними для виконання базових обчислювальних операцій над скалярними даними. Вони реалізують операції з плаваючою комою (FP32, FP64) та цілочисельні операції.

Основною операцією є fused multiply-add (FMA), яка об’єднує множення і додавання в один тактовий цикл. CUDA-ядра працюють за принципом SIMD/SIMT (Single Instruction, Multiple Threads), коли одна інструкція виконується одночасно над великою кількістю потоків. Кожен SM містить десятки або сотні CUDA-ядер, що дозволяє виконувати тисячі потоків одночасно. Однак, попри високий рівень паралелізму, CUDA-ядра залишаються ефективними переважно для скалярних або слабко векторизованих обчислень.

Tensor Cores є спеціалізованими обчислювальними блоками, оптимізованими для виконання операцій лінійної алгебри, зокрема матричного множення з накопиченням результату (Matrix Multiply-Accumulate, MMA).

На відміну від CUDA-ядер, які працюють із окремими скалярами, Tensor Cores оперують блоками даних (матрицями або їх фрагментами). Це дозволяє значно підвищити обчислювальну щільність і ефективність.

У межах одного тактового циклу Tensor Core виконує обчислення над невеликими матрицями (наприклад, 4×4 або більшими тайлами), що

еквівалентно десяткам скалярних операцій. Такий підхід є особливо ефективним для:

- нейронних мереж;
- задач комп'ютерного зору;
- обробки сигналів;
- чисельних методів.

Ефективність GPU значною мірою залежить від пропускної здатності пам'яті. У складі SM присутні спеціалізовані блоки доступу до пам'яті (Load/Store Units), які відповідають за:

- читання даних із глобальної пам'яті;
- запис результатів обчислень;
- роботу з розділюваною пам'яттю.

Shared Memory є швидкою локальною пам'яттю, спільною для потоків одного SM, і використовується для зменшення затримок доступу до даних. Її застосування є критично важливим для оптимізації алгоритмів, що працюють з матрицями та великими масивами.

Глобальна пам'ять є основним обсягом пам'яті GPU (GDDR або HBM), доступним для всіх потоків і всіх SM. Вона характеризується:

- великим обсягом (гігабайти);
- високою пропускною здатністю;
- відносно великою латентністю (сотні тактів).

Операція читання даних включає такі етапи:

1. Потік формує запит на читання.
2. Load/Store Unit агрегує запити від потоків.
3. Виконується доступ до глобальної пам'яті через кеш (L2, інколи L1).
4. Дані повертаються в регістри або розділювана пам'ять.

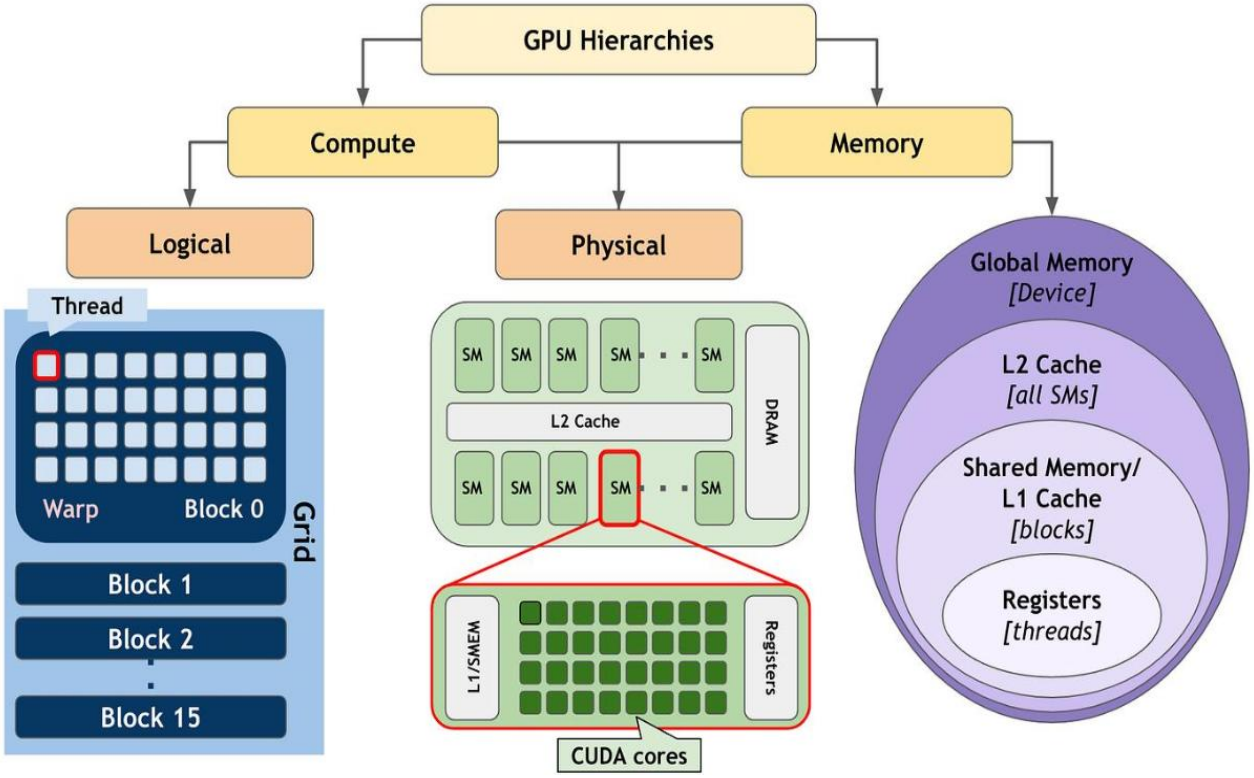


Рисунок 2.2 – Структура доступу до пам’яті в GPU [37]

Особливостями є злиття доступів до пам’яті якщо потоки звертаються до послідовних адрес, запити об’єднуються в одну транзакцію, що значно підвищує ефективність. Доступ може займати сотні тактів, тому використовується приховування затримок через перемикання потоків, натомість неефективний доступ до глобальної пам’яті є основною причиною втрати продуктивності.

Запис результатів обчислень у графічному процесорі є завершальним етапом виконання більшості обчислювальних операцій і здійснюється через спеціалізовані блоки доступу до пам’яті Load/Store Units (LSU), які входять до складу потокового мультипроцесора архітектури NVIDIA Ampere. Після виконання арифметичних операцій результати спочатку формуються у високошвидкісних регістрах, після чого передаються до підсистеми пам’яті для подальшого збереження або повторного використання іншими обчислювальними блоками.

Процес запису починається з того, що результати, отримані після виконання інструкцій CUDA-ядер або Tensor Cores, накопичуються в регістрах кожного потоку. Далі Load/Store Units здійснюють групування запитів запису від потоків одного warp-а, що дозволяє ефективно організувати передачу даних до нижчих рівнів пам’яті. У разі оптимальної організації доступу кілька окремих операцій запису можуть бути об’єднані в одну транзакцію, що зменшує навантаження на пам’ять і підвищує загальну пропускну здатність системи.

Після цього дані надходять до кеш-підсистеми, зокрема до кешу L1 або безпосередньо до спільного кешу L2, залежно від конфігурації архітектури та типу операції.

У сучасних GPU архітектури Ampere використовується механізм відкладеного запису, за якого дані спочатку записуються у кеш, а вже потім, у разі необхідності, синхронізуються з глобальною пам’яттю. Такий підхід дозволяє зменшити кількість звернень до повільної глобальної пам’яті та оптимізувати загальну продуктивність системи.

Важливою особливістю процесу запису є необхідність коректної синхронізації потоків у межах одного обчислювального блоку.

У випадках, коли кілька потоків можуть одночасно звертатися до одних і тих самих ділянок пам’яті, виникає ризик конфліктів запису, що може призвести до некоректних результатів. Для запобігання таким ситуаціям застосовуються механізми синхронізації, які забезпечують узгоджене завершення операцій перед записом у глобальну пам’ять.

Окремо слід зазначити, що ефективність запису значною мірою залежить від характеру доступу до пам’яті. У випадку послідовного або структурованого запису (наприклад, при роботі з матрицями або тензорами) підсистема пам’яті здатна значно підвищити продуктивність за рахунок об’єднання звернень і мінімізації кількості транзакцій.

Натомість випадковий або нерегулярний доступ призводить до зниження

ефективності через збільшення латентності та перевантаження пам’яттєвого інтерфейсу.

Розділювана пам’ять є одним із ключових типів пам’яті в архітектурі графічних процесорів і відіграє критичну роль у підвищенні продуктивності обчислень у межах потокового мультипроцесора архітектури NVIDIA Ampere. Вона являє собою швидку програмно-керовану пам’ять, яка розташована безпосередньо на рівні SM і доступна всім потокам у межах одного обчислювального блоку.

На відміну від глобальної пам’яті, яка має великий обсяг, але високу латентність, розділювана пам’ять характеризується значно меншою затримкою доступу та високою пропускною здатністю. Її обсяг є обмеженим (зазвичай десятки або сотні кілобайт на SM), однак саме ця пам’ять використовується як високошвидкісний проміжний буфер для багаторазового використання даних у межах локальних обчислень.

Основна ідея використання розділюваної пам’яті полягає у зменшенні кількості звернень до глобальної пам’яті.

У типовому сценарії дані спочатку завантажуються з повільнішої глобальної пам’яті до розділюваної, після чого багаторазово використовуються потоками для виконання обчислень. Після завершення обробки результати можуть бути записані назад у глобальну пам’ять.

Такий підхід дозволяє суттєво знизити латентність доступу до даних і підвищити загальну ефективність обчислювального процесу.

Особливо важливою є роль розділюваної пам’яті в задачах, пов’язаних із матричними операціями. У таких випадках застосовується техніка «тайлінгу», коли великі матриці розбиваються на менші блоки, які завантажуються в розділювальну пам'ять.

Потоки однієї групи або блок потоків виконують обчислення над цими локальними блоками, після чого переходять до наступних частин матриці. Це дозволяє значно зменшити кількість повторних звернень до глобальної пам’яті

та підвищити ефективність використання кеш-ієрархії.

Типова схема використання розділюваної пам’яті у матричних обчисленнях наведена на рисунку 2.3.

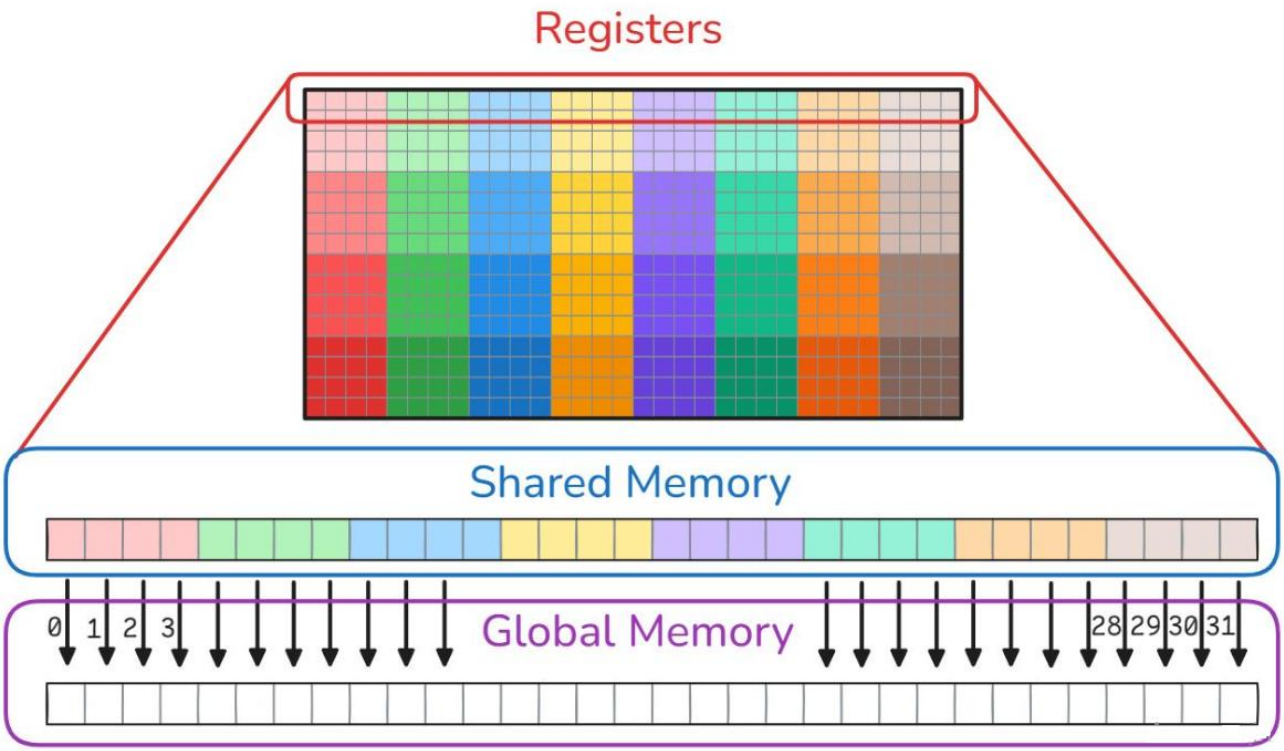


Рисунок 2.3 – Типова схема використання розділюваної пам’яті у матричних обчисленнях [39]

Важливою особливістю розділюваної є те, що вона є програмно керованою. Це означає, що ефективність її використання безпосередньо залежить від розробника, який повинен самостійно організувати завантаження, збереження та синхронізацію даних. Для забезпечення коректної роботи потоків у межах одного потокового блоку використовується бар’єрна синхронізація, яка гарантує завершення всіх операцій запису в розділюваної перед подальшим виконанням обчислень.

Однак використання розділюваної пам’яті має і певні обмеження. По-перше, її обсяг є жорстко обмеженим, що накладає вимоги на розмір

оброблюваних блоків даних. По-друге, у разі некоректної організації доступу можуть виникати конфлікти банків пам’яті, коли кілька потоків одночасно звертаються до одного й того ж банку пам’яті, що призводить до зниження продуктивності. Тому ефективне використання розділюваної потребує ретельної оптимізації структури даних і порядку доступу до них.

Загалом, розділювана пам’ять є одним із найважливіших механізмів оптимізації продуктивності GPU, оскільки дозволяє суттєво зменшити навантаження на глобальну пам’ять і забезпечити ефективну повторну обробку даних у межах локальних обчислювальних блоків. Це особливо критично для алгоритмів лінійної алгебри, глибокого навчання та обробки великих масивів даних, де повторне використання проміжних результатів є визначальним фактором продуктивності.

2.2 Модель апаратної платформи

У якості апаратної основи розроблюваного апаратно-програмного комплексу навчання штучних нейронних мереж використовується модель гібридної обчислювальної системи, що поєднує можливості центрального процесора та графічного процесора. Такий підхід відповідає сучасним тенденціям розвитку обчислювальних систем у сфері машинного навчання, де досягнення високої продуктивності забезпечується за рахунок поєднання універсальних та спеціалізованих обчислювальних ресурсів.

Оскільки в рамках даної роботи відсутня можливість реалізації фізичного апаратного комплексу, використовується програмна модель апаратної платформи, реалізована у середовищі персонального комп’ютера. Даний підхід дозволяє імітувати поведінку реальних обчислювальних пристроїв та дослідити ефективність їх використання при навчанні нейронних мереж.

Модель апаратної платформи базується на концепції гетерогенних обчислень, яка передбачає спільне використання різних типів процесорів для

виконання окремих класів задач. У такій системі центральний процесор виконує функції керування та координації, тоді як графічний процесор використовується як прискорювач для обчислювально інтенсивних операцій.

Реалізація моделі здійснюється за допомогою програмних засобів, що підтримують паралельні обчислення та дозволяють динамічно розподіляти навантаження між CPU та GPU. Такий підхід забезпечує гнучкість у проведенні експериментів та дозволяє моделювати різні режими роботи системи.

Центральний процесор у моделі виконує ключову роль керуючого елемента системи. Він відповідає за організацію обчислювального процесу, взаємодію між компонентами та підготовку даних для подальшої обробки.

Основні функції CPU включають:

- керування процесом виконання, запуск програмних модулів, ініціалізація параметрів моделі, контроль циклу навчання;
- підготовка даних, завантаження датасетів, попередня обробка (нормалізація, масштабування, перетворення форматів);
- організація обміну даними, передача даних між оперативною пам’яттю та обчислювальними пристроями;
- координація обчислювальних задач, розподіл навантаження між CPU та GPU;
- збір результатів та метрик, обробка результатів навчання, логування показників.

Центральний процесор працює в умовах обмеженого рівня паралелізму, проте має високу ефективність при виконанні складних логічних операцій та керуючих інструкцій. Це робить його незамінним елементом системи, навіть при наявності потужних прискорювачів.

Графічний процесор у даній моделі виступає як основний обчислювальний ресурс, що використовується для виконання операцій навчання нейронної мережі. Його функціонування моделюється засобами програмного забезпечення, що підтримують паралельні обчислення.

Основною особливістю GPU є здатність виконувати велику кількість однотипних операцій одночасно. Це досягається завдяки наявності великої кількості обчислювальних ядер, організованих у групи потоків та блоки потоків.

Така архітектура дозволяє ефективно обробляти великі обсяги даних, що є характерним для задач машинного навчання.

У контексті даної роботи GPU використовується для виконання таких операцій:

- множення матриць та тензорів;
- обчислення функцій активації;
- розрахунок градієнтів під час зворотного поширення помилки;
- оновлення вагових коефіцієнтів.

Завдяки паралельному виконанню цих операцій досягається значне прискорення процесу навчання у порівнянні з виконанням лише на CPU.

Паралельні обчислення у моделі реалізуються шляхом розбиття задачі на незалежні підзадачі, які можуть виконуватися одночасно. У випадку нейронних мереж це означає, що обчислення для окремих елементів матриць або нейронів можуть виконуватися незалежно.

Модель передбачає використання таких рівнів паралелізму:

- паралелізм даних, обробка різних елементів навчального набору одночасно;
- паралелізм операцій, виконання арифметичних операцій над різними елементами матриць;
- паралелізм шарів (частково), обчислення в окремих шарах мережі.

Реалізація паралельних обчислень дозволяє значно скоротити час навчання та підвищити ефективність використання обчислювальних ресурсів.

Важливим аспектом моделі є організація роботи з пам’яттю. У системі використовується кілька рівнів пам’яті:

- оперативна пам’ять (RAM), яка використовується CPU;

– пам’ять графічного процесора (VRAM), що використовується для зберігання даних та параметрів моделі під час обчислень.

Процес навчання передбачає постійний обмін даними між CPU та GPU, що включає:

– передачу вхідних даних;

– копіювання параметрів моделі;

– отримання результатів обчислень.

Ефективність цього обміну має значний вплив на загальну продуктивність системи, оскільки надмірні затримки можуть зменшувати виграш від використання GPU.

У рамках моделі передбачено можливість роботи системи у кількох режимах:

1. CPU-режим**,** тобто всі обчислення виконуються центральним процесором;

2. GPU-режим**,** основні обчислення виконуються графічним процесором;

3. Гібридний режим, тобто CPU та GPU використовуються одночасно.

Це дозволяє проводити порівняльний аналіз ефективності різних підходів та визначати оптимальну конфігурацію системи.

Використання програмної моделі апаратної платформи має ряд переваг:

– відсутність необхідності у фізичному обладнанні;

– можливість швидкої зміни конфігурації системи;

– доступність для проведення експериментів;

– гнучкість у дослідженні різних сценаріїв.

Крім того, така модель дозволяє отримати результати, які є достатньо близькими до реальних умов роботи апаратних систем.

Разом з тим, слід враховувати певні обмеження:

– неможливість повного відтворення апаратних характеристик;

– залежність від параметрів конкретного комп’ютера;

– спрощення деяких аспектів роботи GPU.

Робота апаратної частини комплексу базується на чіткому розподілі обов'язків між центральним та графічним процесорами. Центральний процесор виступає в ролі головного керівного вузла системи. Він повністю відповідає за логіку роботи програми, керує процесом обчислень, готує та завантажує дані, а також збирає готові результати й контролює стан обладнання через систему моніторингу. Графічний процесор, своєю чергою, працює як потужний співпроцесор. Завдяки великій кількості обчислювальних ядер (CUDA-ядер), власній відеопам'яті (VRAM) та спеціальним апаратним блокам, GPU бере на себе математичну роботу, тобто паралельні обчислення та матричні операції, які є основою для навчання нейронних мереж.

Загальна структура моделі апаратної платформи представлена на рисунку 2.4.

Взаємодія між цими процесорами та іншими компонентами організована через систему пам'яті та швидкісні шини. Усі поточні дані, з якими працює система в конкретний момент (наприклад, пакети даних або батчі), тимчасово зберігаються в оперативній пам'яті (RAM). Обмін інформацією між CPU, GPU та оперативною пам'яттю відбувається через швидкісну системну шину (зокрема, PCI Express). Для довготривалого зберігання великих масивів інформації, навчальних вибірок та готових вагових коефіцієнтів моделі використовується сховище даних на базі твердотільних накопичувачів або жорстких дисків. У процесі роботи дані постійно перекачуються зі сховища в оперативну пам'ять для подальшої обробки.

Логіка роботи системи та конвеєра обробки даних під керуванням CPU складаються з кількох послідовних етапів.

На самому початку процесор завантажує потрібний датасет із диска в оперативну пам'ять, виконує його попередню обробку (очищає від помилок, нормалізує значення) та розбиває всю вибірку на невеликі однакові пакети, тобто батчі. Далі запускається циклічний процес керування завантаженням: CPU ініціює початок кожної епохи навчання, передає підготовлені батчі на

GPU, чекає, поки графічний чип виконає матричні обчислення, забирає проміжні результати та оновлює параметри мережі.

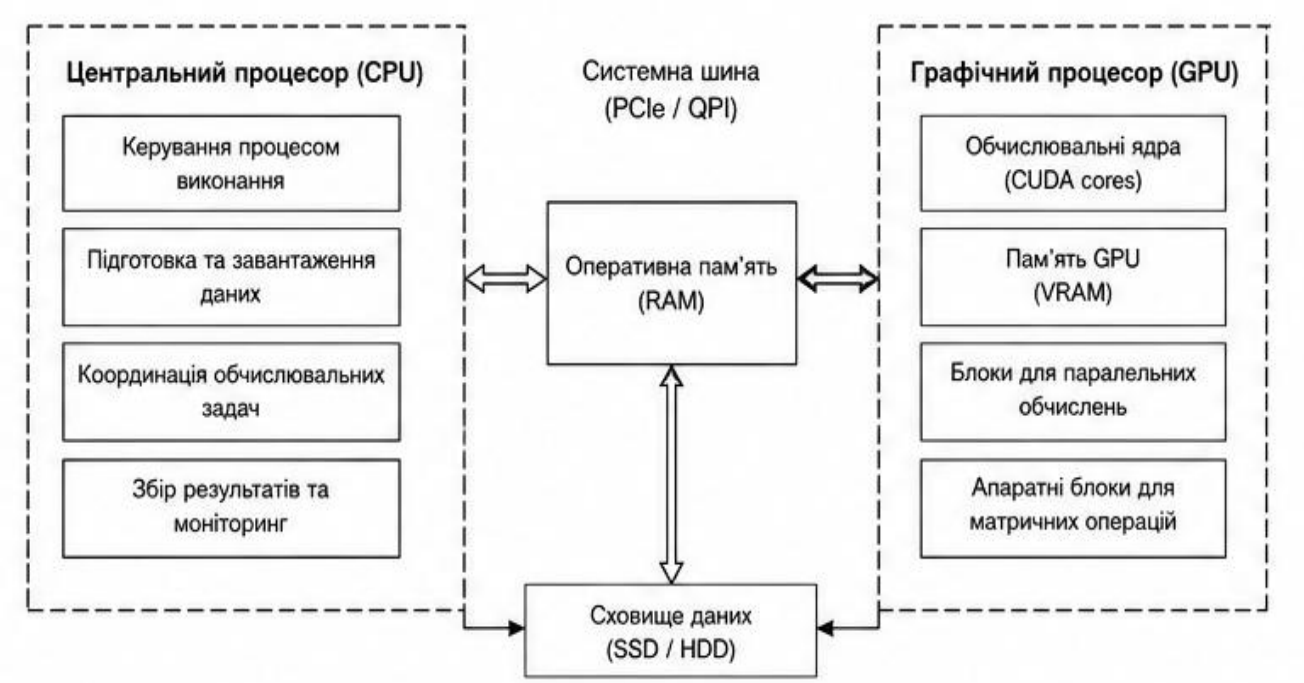


Рисунок 2.4 – Загальна структура моделі апаратної платформи

Цей цикл повторюється по колу, доки не закінчаться всі дані. Після завершення навчання CPU перемикає систему в режим оцінки якості, перевіряє точність отриманої моделі на тестових даних та записує фінальні результати й логі назад на накопичувач. Такий підхід дозволяє вичавити максимум продуктивності з обладнання, оскільки CPU займається лише організацією процесу, а GPU безперервно рахує матриці.

Організація потоків даних та координація обчислювальних процесів у структурі апаратно-програмного комплексу базуються на чіткому розподілі функцій між компонентами та наскрізній взаємодії спеціалізованих контролерів. Головним диспетчером системи виступає центральний процесор, який координує обмін даними, обробляє логіку високого рівня та керує конвеєром навчання, тоді як основне математичне навантаження з обробки матричних операцій покладається на графічний прискорювач.

Початкова фаза функціонування комплексу починається з ініціалізації та передпроцесингу, де центральний процесор виконує операції введення-виведення для зчитування сирих масивів даних з модуля довготривалого збереження до системної оперативної пам'яті.

Після завантаження датасету ядра процесора здійснюють його попередню обробку, що включає декомпресію файлів, очищення від шумів, нормалізацію числових значень та операції аугментації. Оскільки паралельна архітектура графічного процесора вимагає строго структурованих і однорідних пакетів, центральний процесор групує підготовлені елементи вибірки у підмасиви фіксованого розміру, формуючи батчі відповідно до заданих гіперпараметрів мережі (рис.2.5).

Рисунок 2.5 – Функціональна схема роботи CPU у системі

Для усунення ефекту вузького місця на рівні дискової підсистеми модуль накопичення даних функціонує під керуванням апаратного контролера NVMe. Архітектура цього контролера оптимізована під багатопотокові обчислення і підтримує до 64 000 черг виконання, кожна з яких здатна вміщувати до 64 000 команд одночасно, що дозволяє асинхронно зчитувати тисячі дрібних файлів паралельно з декількох каналів флеш-пам'яті NAND. Фізичне спрямування та

комутація цих потоків даних здійснюється через високошвидкісний апаратний комутатор шини PCI Express, який функціонує на рівні пакетів транзакцій. Замість транзиту даних через системний міст процесора, комутатор PCIe Switch аналізує заголовки пакетів і перенаправляє їх безпосередньо на вхідний інтерфейс графічного процесора, знижуючи затримку передачі до наносекундного рівня та розвантажуючи системну шину. Наскрізне перенесення сформованих батчів даних з накопичувача у відеопам'ять реалізується за допомогою технології прямого доступу до пам'яті Peer-to-Peer DMA в обхід системної оперативної пам'яті та ресурсів центрального процесора.

Завдяки такій конфігурації забезпечується безперервний конвеєр обчислень, де процеси прямого зчитування наступної епохи через DMA суміщаються у часі з виконанням прямого і зворотного поширення помилки поточного батчу на ядрах графічного процесора.

У межах циклічного макроблоку керування навчанням центральний процесор ініціює глобальний ітераційний цикл, контролює запуск кожної епохи, здійснює перемішування індексів батчів, очікує завершення обчислень на графічному прискорювачі та зчитує проміжні значення функції втрат і метрик точності. На основі отриманих градієнтів процесор координує крок оптимізатора для коригування матриць вагових коефіцієнтів та зсувів нейронів, після чого зворотний зв'язок повертає хід виконання до формування нових батчів, і цей цикл повторюється до вичерпання всіх пакетів.

Паралельно з обчисленнями спеціалізований контролер моніторингу безперервно опитує датчики телеметрії для фіксації температурних показників кристалів процесорів і накопичувача, рівня енергоспоживання та індексу зносу комірок пам'яті, що дозволяє апаратному планувальнику автоматично активувати режим тротлінгу або надіслати сигнал зупинки у разі перегріву.

Після виконання заданої кількості ітерацій або досягнення критерію зупинки навчання, система переходить до завершального етапу, на якому

центральний процесор перемикає модель у режим валідації, обчислює підсумкові метрики ефективності на тестовій вибірці та здійснює фінальний запис архітектури мережі, логів та бінарних файлів ваг назад на енергонезалежний NVMe-накопичувач для їх подальшого використання у симуляційному середовищі MATLAB, Simulink або Google Colab.

Для реалізації масивних паралельних обчислень, що є основою алгоритмів навчання нейронних мереж, у комплексі використовується графічний процесор (GPU). На відміну від центрального процесора, який оптимізований для швидкого послідовного виконання однієї-двох задач, архітектура GPU розроблена для одночасної обробки тисяч простих математичних операцій. Внутрішня будова та організація компонентів графічного прискорювача представлена на структурній схемі (рис. 2.3).

Основу обчислювальної потужності графічного процесора складає кластерна архітектура (рис.2.6). Уся матриця графічного чипа поділена на кілька великих обчислювальних блоків, які називаються мультипроцесорними кластерами (на схемі позначені від 1 до M). Такий поділ дозволяє масштабувати продуктивність відеокарти: чим більше кластерів інтегровано в кристал, тим більше обчислень система може виконувати паралельно за один тактовий імпульс.

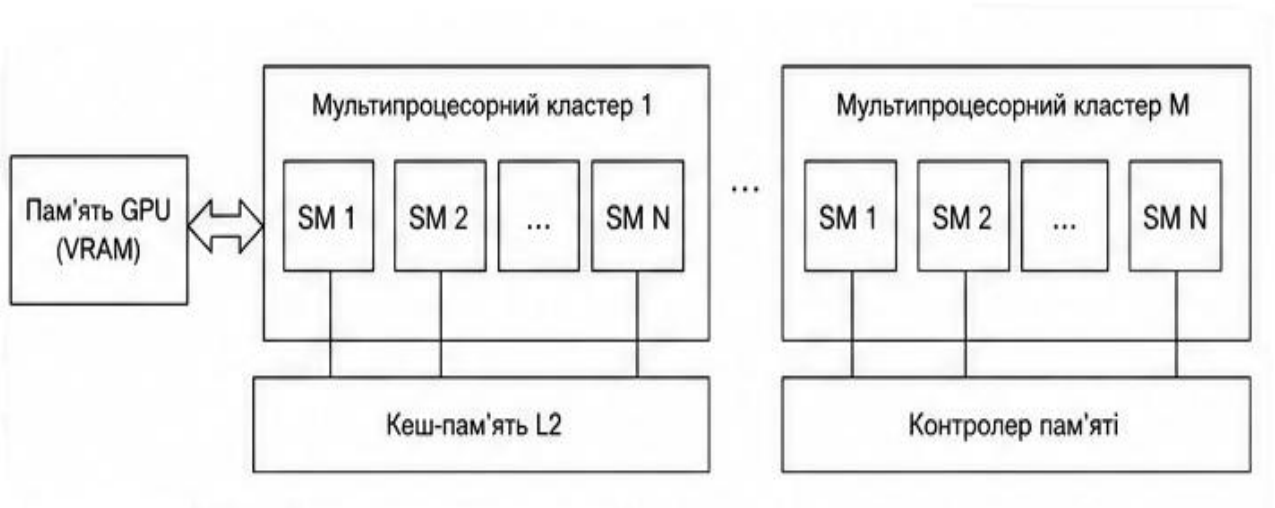


Рисунок 2.6 – Структурна схема моделі GPU

Кожен мультипроцесорний кластер містить у своєму складі набір потокових мультипроцесорів (Streaming Multiprocessors, або SM, позначені на схемі від 1 до N). Потоковий мультипроцесор є базовим обчислювальним модулем GPU. Саме всередині кожного SM розташовані сотні дрібних ядер (зокрема, CUDA-ядра та спеціалізовані тензорні блоки), які безпосередньо виконують арифметичні дії над матрицями та векторами ваг нейронної мережі.

Усі мультипроцесори всередині кластера працюють незалежно один від одного, але за єдиним командним принципом, що дозволяє ефективно розподіляти між ними великі пакети даних (батчі).

Важливою частиною структури GPU є підсистема пам'яті, оскільки швидкість математичних розрахунків напряму залежить від того, як швидко обчислювальні блоки отримують потрібні цифри.

Головним масивом для збереження поточної інформації є власна пам'ять GPU, або відеопам'ять (VRAM). У VRAM завантажуються вагові коефіцієнти, архітектура мережі та поточні навчальні батчі, передані від центрального процесора. Обмін даними між VRAM та обчислювальними кластерами є двостороннім: під час прямого ходу навчання з відеопам'яті зчитуються вхідні дані, а під час зворотного ходу туди записуються оновлені значення градієнтів.

Оскільки пряме звернення до відеопам'яті VRAM вимагає певного часу і може створювати затримки, всередині графічного процесора реалізовано швидкісну буферну зону, тобто кеш-пам'ять другого рівня (L2 Cache).

Кеш-пам'ять L2 є спільною для всіх обчислювальних кластерів системи. У ній тимчасово зберігаються ті дані, до яких мультипроцесори звертаються найчастіше (наприклад, константи або проміжні матричні коефіцієнти). Використання L2-кешу дозволяє значно скоротити кількість тривалих звернень до основної відеопам'яті, що суттєво підвищує загальну швидкість розрахунків.

Кординацію та фізичний розподіл інформаційних потоків між усіма елементами GPU здійснює апаратний контролер пам'яті. Він керує чергами

запитів на читання та запис, стежить за правильністю адресації й забезпечує максимальну пропускну здатність шини пам'яті.

Контролер пам'яті виступає зв'язковою ланкою, яка балансує роботу між повільною (відносно швидкості самих ядер) відеопам'яттю VRAM, швидким кешем L2 та потоковими мультипроцесорами SM.

2.3 Проєктування програмного забезпечення та інформаційного конвеєра комплексу

Ефективність функціонування розроблюваного апаратно-програмного комплексу навчання штучних нейронних мереж безпосередньо залежить від раціональної організації його програмної складової. Оскільки основні апаратні обчислювальні ресурси комплексу, представлені тензорними та потоковими мультипроцесорами графічного прискорювача, мають високу швидкодію, головним завданням при проєктуванні програмного забезпечення є усунення затримок під час передачі та попередньої обробки інформації.

Програмне забезпечення комплексу проектується за модульним принципом, що дозволяє відокремити завдання низькорівневої взаємодії з апаратними компонентами від високорівневої логіки керування симуляцією та навчання моделей.

Центральною концепцією архітектури софту є створення наскрізного інформаційного конвеєра, який забезпечує безперервний рух даних від енергонезалежного накопичувача NVMe через системну оперативну пам’ять безпосередньо у виділену відеопам’ять графічного процесора.

Це дозволяє мінімізувати час простою ядер GPU та оптимізувати загальне споживання системних ресурсів під час виконання тривалих ітераційних обчислень.

На етапі проєктування інформаційного конвеєра розробляється

трирівнева архітектура програмних модулів, яка включає рівень збереження та асинхронного зчитування, рівень системного передпроцесингу та рівень апаратних тензорних обчислень.

Логічну структуру спроєктованого інформаційного конвеєра та схему розподілу потоків даних між апаратними рівнями комплексу представлено на рисунку 2.7.

Перший рівень, рівень збереження даних базується на розробці програмних інтерфейсів для асинхронного читання великих масивів інформації з твердотільного накопичувача. Традиційні методи послідовного зчитування створюють значні затримки, тому в розроблюваному комплексі програмний модуль проектується з використанням неблокуючих I/O-операцій та системних викликів, які дозволяють паралельно заповнювати декілька черг запитів. Це забезпечує повну сумісність із апаратною структурою NVMe-контролера, що підтримує багатопотоковість, та дозволяє зчитувати інформацію окремими сегментами без залучення обчислювальних ресурсів центрального процесора для очікування завершення дискових операцій.

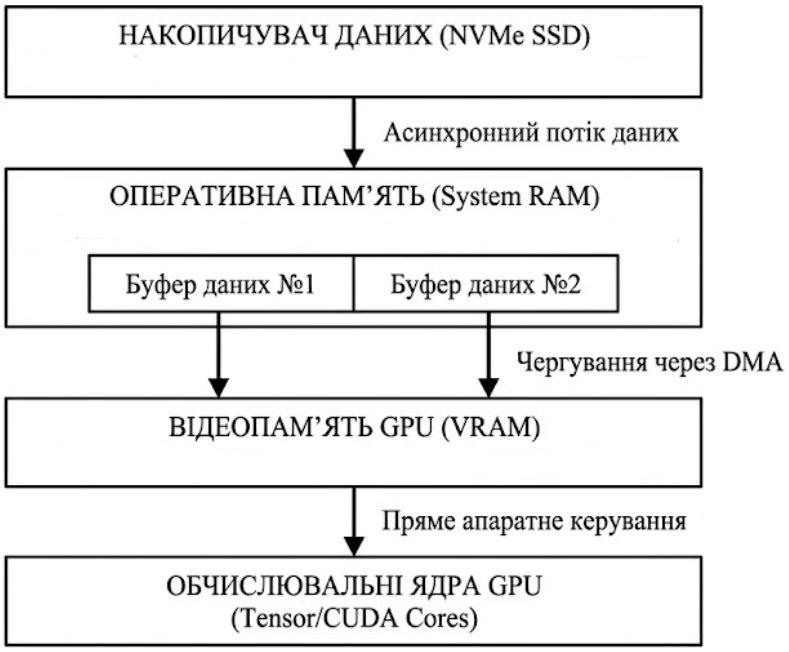


Рисунок 2.7 – Схема інформаційних потоків та рівнів проектування ПЗ

Другий рівень конвеєра, рівень системного передпроцесингу проектується для виконання в системній оперативній пам’яті (RAM) під безпосереднім керуванням ядер центрального процесора.

Основна задача цього програмного модуля полягає у трансформації сирих вхідних даних до вигляду, оптимізованого для обробки нейронною мережею.

Програмні алгоритми цього рівня реалізують функції декомпресії файлів, очищення від аномальних значений, нормалізації числових векторів у заданий діапазон (наприклад, від 0 до 1), а також математичні перетворення аугментації даних.

Для запобігання ситуаціям, коли графічний процесор очікує на підготовку наступного пакета даних, у програмному забезпеченні проектується механізм подвійної буферизації. Згідно з цим підходом, в оперативній пам’яті виділяються два незалежні програмні буфери: поки перший буфер передає поточний сформований батч у відеопам’ять для обчислення поточної епохи, другий буфер автономно заповнюється новими даними з накопичувача та обробляється алгоритмами CPU.

Третій рівень, рівень апаратних тензорних обчислень є безпосереднім ядром програмного комплексу, яке взаємодіє з обчислювальними кластерами GPU. Проєктування цього рівня передбачає використання спеціалізованих бібліотек низькорівневого програмування NVIDIA CUDA та cuDNN, інтегрованих у високорівневі фреймворки машинного навчання.

Програмний модуль ініціалізує виділення простору у відеопам’яті (VRAM) для розміщення архітектурного графа нейронної мережі, матриць вагових коефіцієнтів, векторів зсувів та градієнтів. Важливою архітектурною особливістю проектування є мінімізація зворотного копіювання даних між GPU та CPU в межах однієї епохи навчання.

Усі проміжні кроки, включаючи пряме поширення сигналу, розрахунок значення функції втрат, зворотне поширення помилки та коригування ваг за допомогою обраного оптимізатора, виконуються виключно всередині

відеопам'яті та обчислювальних ядер прискорювача, що усуває потребу у використанні відносно повільної системної шини для проміжних транзакцій.

2.4 Висновки до другого розділу

У другому розділі кваліфікаційної роботи проведено аналіз, проєктування та обґрунтування рішень для створення апаратно-програмного комплексу навчання штучних нейронних мереж.

Встановлено, що поділ структури GPU на мультипроцесорні кластери, тензорні та стандартні обчислювальні ядра дозволяє паралельно виконувати матричні операції. Це забезпечує збільшення швидкості обчислень порівняно з послідовною обробкою завдань на центральному процесорі.

Розроблено та описано модель апаратної платформи комплексу. Визначено логіку взаємодії та зв'язки між центральним процесором, графічним прискорювачем, оперативною пам'яттю та накопичувачем даних. Визначено роль CPU як керівного елемента системи, який відповідає за загальну логіку програми, передачу даних та моніторинг стану обладнання, тоді як GPU виконує функцію співпроцесора для математичних розрахунків.

Обґрунтовано впровадження інтерфейсів обміну даними для запобігання затримкам при зчитуванні інформації. Використання протоколу NVMe та технології прямого доступу до пам'яті дозволило організувати передачу даних із накопичувача у відеопам'ять GPU через комутатор в обхід основних ресурсів процесора та оперативної пам'яті. Спроєктовано трирівневу структуру програмного забезпечення комплексу, яка складається з рівня збереження даних, рівня попередньої обробки на CPU та рівня обчислень на GPU. Завдяки розробці програмного механізму подвійної буферизації забезпечено конвеєрну обробку інформації, де процес підготовки наступного пакета даних виконується одночасно з розрахунками поточної епохи навчання, що зменшує час простою обчислювальних ядер.

3 ПРОГРАМНО-АПАРАТНА РЕАЛІЗАЦІЯ КОМПЛЕКСУ НАВЧАННЯ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ

3.1 Опис модулів апаратного та програмного забезпечення апаратно-програмного комплексу

Апаратно-програмний комплекс навчання штучних нейронних мереж призначений для забезпечення повного циклу створення, навчання, тестування та використання моделей штучного інтелекту.

Комплекс складається з взаємопов’язаних апаратних і програмних модулів, які забезпечують ефективну обробку даних та виконання обчислювально складних операцій (рис. 3.1).

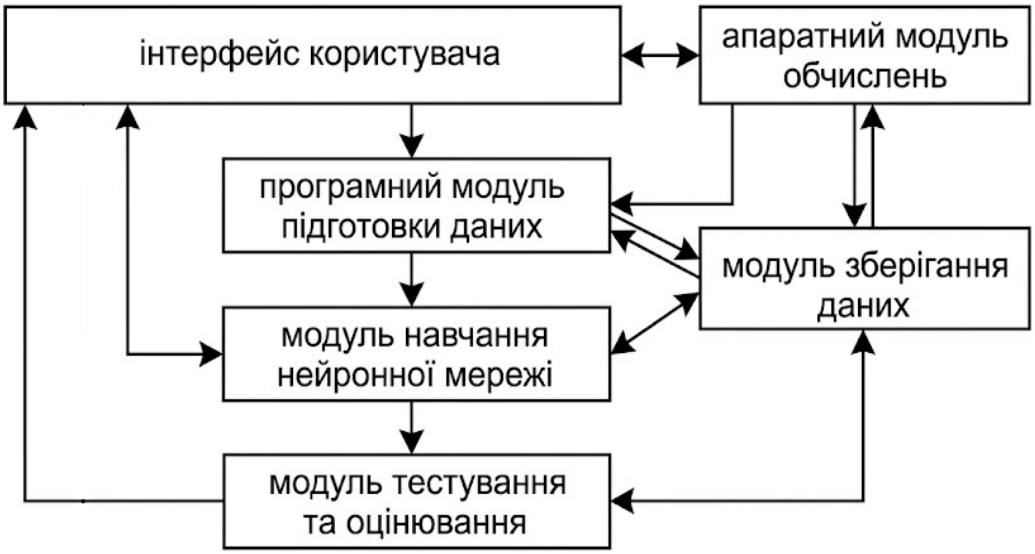


Рисунок 3.1 – Структурна схема апаратно-програмного комплексу навчання штучних нейронних мереж

Основою комплексу є апаратний модуль обчислень, який забезпечує виконання всіх ресурсоємних операцій, пов’язаних із навчанням нейронних мереж. До складу цього модуля входять центральний процесор, графічний процесор, оперативна пам’ять та інші обчислювальні ресурси. Саме цей модуль

відповідає за швидкість обробки даних, виконання математичних операцій, обчислення градієнтів та оновлення вагових коефіцієнтів моделі. Ефективність роботи всього комплексу значною мірою залежить від продуктивності апаратної складової, особливо при роботі з великими наборами даних та глибокими нейронними мережами.

Наступним компонентом є модуль зберігання даних, який виконує функції накопичення, організації та управління інформацією. У цьому модулі зберігаються навчальні та тестові вибірки, проміжні результати обробки, параметри моделей, а також журнали роботи системи. Дані можуть зберігатися як у локальних сховищах, так і у віддалених або хмарних середовищах. Важливою характеристикою цього модуля є забезпечення швидкого доступу до даних, їх цілісності та можливості масштабування обсягів зберігання.

Програмний модуль підготовки даних є початковим етапом обробки інформації в системі. Він відповідає за перетворення сирих даних у формат, придатний для навчання нейронної мережі. До його функцій входять очищення даних від шуму та помилок, нормалізація або стандартизація значень, кодування категоріальних ознак, а також розподіл даних на навчальні, валідаційні та тестові підмножини. Якість роботи цього модуля безпосередньо впливає на ефективність навчання моделі, оскільки некоректно підготовлені дані можуть призвести до зниження точності або перенавчання.

Центральним елементом комплексу є модуль навчання нейронної мережі, який реалізує основні алгоритми машинного навчання.

У цьому модулі здійснюється побудова архітектури нейронної мережі, ініціалізація її параметрів, виконання прямого та зворотного поширення сигналу, а також оптимізація вагових коефіцієнтів.

Модуль забезпечує можливість використання різних типів нейронних мереж (наприклад, згорткових або рекурентних), а також різних алгоритмів оптимізації. Крім того, він підтримує налаштування гіперпараметрів, таких як швидкість навчання, кількість епох, розмір батчу тощо. Саме цей модуль

визначає інтелектуальні можливості всієї системи.

Після завершення процесу навчання використовується модуль тестування та оцінювання, який дозволяє визначити якість побудованої моделі. У цьому модулі виконуються обчислення різноманітних метрик ефективності, таких як точність, повнота, F-міра та інші показники, залежно від поставленої задачі. Також може проводитися аналіз помилок, побудова матриці неточностей та порівняння результатів різних моделей. Це дозволяє зробити обґрунтовані висновки щодо придатності моделі до практичного використання.

Завершальним компонентом структури є інтерфейс користувача, який забезпечує взаємодію людини з апаратно-програмним комплексом. Через інтерфейс користувач має можливість завантажувати дані, налаштовувати параметри навчання, запускати процеси обробки та аналізувати отримані результати. Інтерфейс може бути реалізований у вигляді командного рядка, графічного додатку або веб-інтерфейсу. Його основним завданням є забезпечення зручності та доступності використання системи навіть для користувачів без глибоких технічних знань.

Взаємодія між усіма модулями комплексу реалізується за допомогою внутрішніх програмних інтерфейсів (API) та механізмів обміну даними. Це дозволяє організувати чіткий потік інформації між компонентами системи: від завантаження та підготовки даних до отримання кінцевих результатів.

Такий підхід забезпечує незалежність модулів, що спрощує їх тестування, модернізацію та повторне використання. Крім того, модульна архітектура дозволяє інтегрувати додаткові компоненти, наприклад, нові алгоритми навчання або зовнішні сервіси обробки даних.

Наступним етапом є побудова функціональної схеми, яка описує алгоритм роботи комплексу, рух даних між модулями та основні етапи навчання нейронної мережі (рис.3.2).

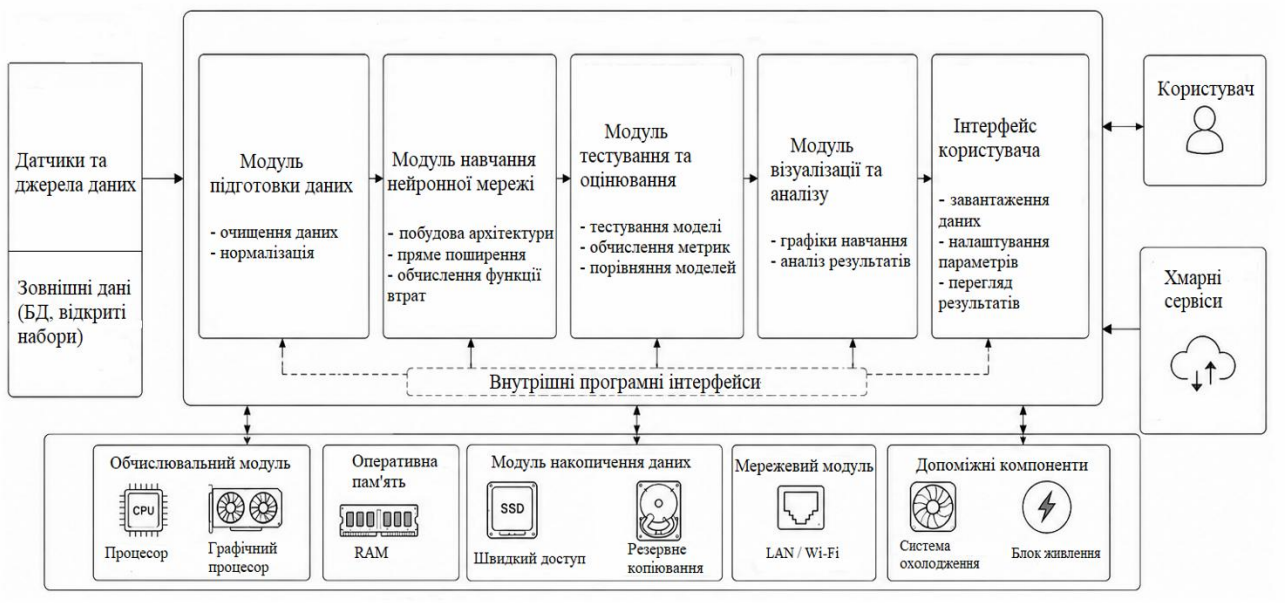


Рисунок 3.2 – Функціональна схема апаратно-програмного комплексу навчання штучних нейронних мереж

Апаратне забезпечення апаратно-програмного комплексу навчання штучних нейронних мереж є складовою, що визначає продуктивність, швидкодію та ефективність виконання обчислювальних процесів.

Оскільки задачі навчання нейронних мереж пов’язані з обробкою великих обсягів даних та виконанням значної кількості математичних операцій, до апаратної частини висуваються підвищені вимоги щодо обчислювальної потужності, обсягу пам’яті та швидкості передачі даних.

Основним елементом апаратного забезпечення є обчислювальний модуль, який реалізується на базі персонального комп’ютера, робочої станції або спеціалізованого сервера. Центральним компонентом цього модуля є процесор (CPU), який виконує загальні обчислювальні операції, керує роботою інших компонентів системи та забезпечує виконання програмного коду. Для задач машинного навчання доцільно використовувати багатоядерні процесори з високою тактовою частотою, оскільки це дозволяє ефективно обробляти паралельні обчислення та прискорює підготовку даних.

Особливу роль у системі відіграє графічний процесор (GPU), який призначений для прискорення обчислень, пов’язаних із навчанням нейронних

мереж. На відміну від центрального процесора, GPU має велику кількість обчислювальних ядер, що дозволяє одночасно виконувати тисячі операцій. Це робить його особливо ефективним для обробки матриць і тензорів, які є основою обчислень у нейронних мережах. Найбільш поширеними є графічні процесори компанії NVIDIA, які підтримують технології паралельних обчислень, зокрема CUDA, що значно підвищує швидкість навчання моделей. Використання GPU дозволяє скоротити час навчання з годин або днів до значно менших проміжків часу.

Не менш важливою складовою є оперативна пам’ять (RAM), яка використовується для тимчасового зберігання даних і програм під час виконання обчислень. Обсяг оперативної пам’яті безпосередньо впливає на можливість обробки великих наборів даних та складних моделей. Для ефективної роботи комплексу рекомендується використовувати не менше 8–16 ГБ оперативної пам’яті, хоча для складних задач глибинного навчання цей показник може бути значно вищим. Недостатній обсяг пам’яті може призвести до зниження продуктивності або навіть неможливості виконання певних обчислень.

Для довготривалого зберігання даних використовується модуль накопичення інформації, який може бути реалізований на основі жорстких дисків (HDD) або твердотільних накопичувачів (SSD). SSD-накопичувачі мають значно вищу швидкість читання та запису даних порівняно з HDD, що дозволяє пришвидшити завантаження навчальних вибірок, збереження моделей та обробку великих масивів інформації. У сучасних системах перевага надається саме SSD, оскільки швидкість доступу до даних є критично важливою для ефективного навчання нейронних мереж.

Традиційні HDD базуються на магнітному принципі запису даних та мають рухомі механічні елементи, що обмежує швидкість доступу до інформації. Незважаючи на відносно низьку вартість і великий обсяг зберігання, такі накопичувачі характеризуються високою затримкою доступу до

даних, що негативно впливає на продуктивність системи при інтенсивному читанні та записі.

На відміну від HDD, SSD-накопичувачі використовують флеш-пам’ять і не містять рухомих частин, що забезпечує значно вищу швидкість роботи. Час доступу до даних у SSD є на порядок меншим, а швидкість читання та запису може перевищувати показники HDD у декілька разів. Це особливо важливо при навчанні нейронних мереж, де постійно відбувається завантаження батчів даних у пам’ять, збереження вагових коефіцієнтів та логів навчання. Використання SSD дозволяє значно зменшити час простою обчислювальних модулів, зокрема графічних процесорів, які очікують на отримання даних.

У сучасних високопродуктивних системах часто використовуються SSD із інтерфейсом NVMe, який забезпечує ще більшу швидкість передачі даних завдяки використанню шини PCI Express. Це дозволяє максимально ефективно використовувати потенціал обчислювальних ресурсів, особливо при роботі з великими наборами даних у задачах глибинного навчання.

Окрім основних накопичувачів, модуль зберігання даних може включати системи резервного копіювання, що реалізуються на базі HDD або мережевих сховищ (NAS). Це дозволяє забезпечити збереження результатів навчання та запобігти втраті важливої інформації у разі збоїв системи. У складніших реалізаціях також можуть використовуватись розподілені файлові системи або хмарні сховища.

Важливим аспектом функціонування накопичувачів є моніторинг їх стану та параметрів роботи. Для цього використовуються вбудовані датчики та системи діагностики, які дозволяють контролювати:

- температуру накопичувача;
- кількість циклів запису (для SSD);
- швидкість передачі даних;
- стан поверхні диска (для HDD);
- рівень зносу пам’яті.

Для моніторингу стану накопичувачів широко застосовується технологія S.M.A.R.T. (Self-Monitoring, Analysis and Reporting Technology), яка дозволяє прогнозувати можливі відмови та своєчасно реагувати на критичні ситуації. Дані з таких датчиків можуть використовуватись у програмному забезпеченні комплексу для автоматичного сповіщення користувача або оптимізації режимів роботи системи.

Окрему увагу слід приділити мережевому обладнанню, яке забезпечує підключення комплексу до локальних та глобальних мереж. Це дає можливість використовувати віддалені обчислювальні ресурси, завантажувати великі набори даних, а також інтегруватися з хмарними платформами. Мережевий модуль включає мережеві адаптери, маршрутизатори та інші засоби комунікації, що забезпечують стабільну та швидку передачу даних. Використання високошвидкісного інтернет-з'єднання є важливим фактором при роботі з розподіленими системами та хмарними сервісами.

У деяких випадках апаратне забезпечення комплексу може включати спеціалізовані прискорювачі обчислень, такі як тензорні процесори (TPU), які оптимізовані для виконання операцій машинного навчання. Такі пристрої забезпечують ще більшу продуктивність порівняно з традиційними GPU, однак їх використання зазвичай пов'язане з хмарними сервісами або спеціалізованими інфраструктурами.

Важливим аспектом є також система охолодження та енергозабезпечення, оскільки інтенсивні обчислення створюють значне теплове навантаження на компоненти системи. Надійна система охолодження забезпечує стабільну роботу обладнання та запобігає його перегріву, що може призвести до зниження продуктивності або виходу з ладу. Джерело живлення повинно відповідати вимогам усіх компонентів системи та забезпечувати їх безперебійну роботу.

Таким чином, апаратне забезпечення комплексу навчання штучних нейронних мереж є багатокомпонентною системою, що включає обчислювальні

ресурси, засоби зберігання даних, мережеве обладнання та допоміжні елементи. Від правильного вибору та конфігурації апаратної частини залежить ефективність функціонування всього комплексу, швидкість навчання моделей і можливість роботи з великими обсягами даних.

3.2 Програмне забезпечення комплексу

Програмне забезпечення апаратно-програмного комплексу навчання штучних нейронних мереж забезпечує реалізацію основних функціональних можливостей системи, включаючи підготовку даних, побудову та навчання моделей, оцінювання результатів і взаємодію з користувачем. Воно є логічною складовою комплексу, яка координує роботу апаратних ресурсів та забезпечує виконання алгоритмів машинного навчання.

Структурно програмне забезпечення комплексу побудоване за модульним принципом, що дозволяє розділити функціональність системи на окремі компоненти, кожен з яких відповідає за виконання конкретних задач. Такий підхід забезпечує гнучкість, можливість масштабування та спрощує супровід і модернізацію системи.

Одним із ключових елементів є модуль підготовки даних, який відповідає за обробку вхідної інформації перед її подачею до нейронної мережі. У реальних умовах дані часто містять пропуски, шум або мають різні формати представлення, що унеможливлює їх безпосереднє використання для навчання. У цьому модулі реалізуються процедури очищення даних, нормалізації значень, перетворення форматів, кодування категоріальних ознак, а також розбиття даних на навчальні, валідаційні та тестові вибірки. Крім того, може застосовуватись аугментація даних, що дозволяє збільшити обсяг навчальної вибірки без фактичного збору нових даних.

Для реалізації зазначених функцій широко використовуються спеціалізовані бібліотеки, зокрема NumPy для виконання чисельних обчислень

та Pandas для роботи з табличними даними. Їх застосування дозволяє значно спростити процес обробки великих масивів інформації та підвищити ефективність виконання операцій.

Центральним компонентом програмного забезпечення є модуль навчання нейронної мережі, який реалізує алгоритми глибинного навчання. У цьому модулі здійснюється створення архітектури нейронної мережі, визначення кількості шарів, типів нейронів та функцій активації. Також виконується ініціалізація вагових коефіцієнтів та запуск процесу навчання.

Процес навчання включає декілька основних етапів: пряме поширення сигналу, обчислення функції втрат, зворотне поширення помилки та оновлення вагових коефіцієнтів за допомогою оптимізаційних алгоритмів.

Схема програмного забезпечення комплексу представлена на рисунку 3.2.

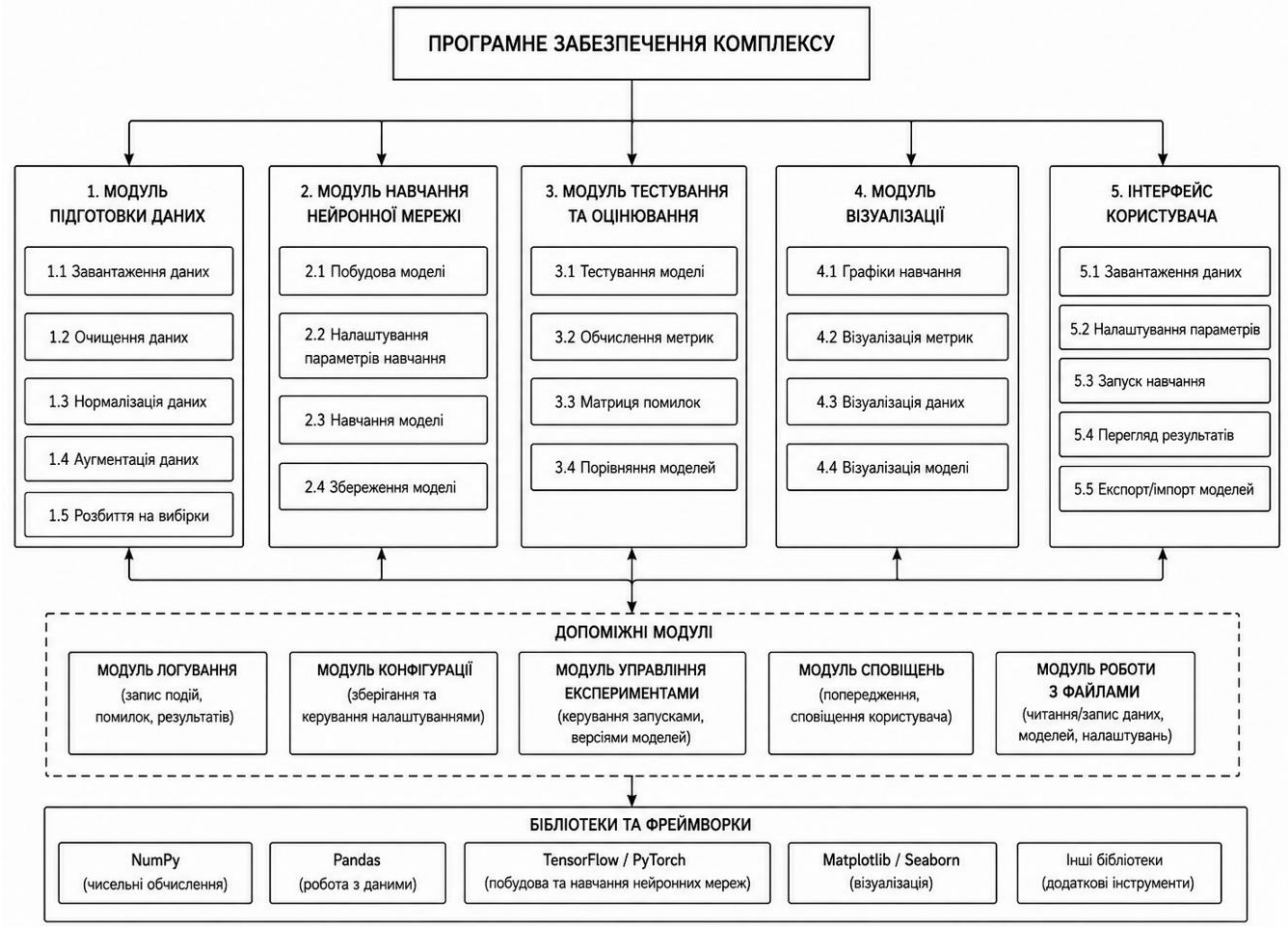


Рисунок 3.2 – Схема програмного забезпечення комплексу

Для реалізації цих процесів використовуються сучасні фреймворки машинного навчання, такі як TensorFlow та PyTorch, які забезпечують підтримку паралельних обчислень та інтеграцію з графічними процесорами.

Дані інструменти забезпечують високорівневу абстракцію для побудови моделей, а також ефективну реалізацію обчислювальних алгоритмів із використанням апаратних ресурсів.

Фреймворк TensorFlow, розроблений компанією Google, орієнтований на створення масштабованих систем машинного навчання та підтримує як дослідницькі, так і промислові задачі. Його архітектура базується на обчислювальних графах, де операції представлені у вигляді вузлів, а дані у вигляді тензорів, що передаються між ними. Такий підхід дозволяє ефективно оптимізувати виконання обчислень, розподіляти їх між різними пристроями та забезпечувати паралельну обробку даних.

У свою чергу, PyTorch, розроблений компанією Meta, відрізняється динамічною моделлю обчислень, що дозволяє будувати обчислювальні графи безпосередньо під час виконання програми. Це значно спрощує процес розробки та налагодження моделей, роблячи PyTorch особливо популярним у наукових дослідженнях і навчальних проєктах. Гнучкість цього фреймворку дозволяє легко змінювати архітектуру нейронної мережі та експериментувати з різними підходами до навчання.

Однією з ключових переваг обох фреймворків є підтримка паралельних обчислень та можливість інтеграції з графічними процесорами. Завдяки використанню спеціалізованих бібліотек, таких як CUDA та cuDNN, що розробляються компанією NVIDIA, забезпечується ефективне використання ресурсів GPU для виконання великої кількості однотипних математичних операцій. Це дозволяє значно прискорити процес навчання нейронних мереж, особливо при роботі з великими наборами даних і складними моделями глибинного навчання.

Крім того, обидва фреймворки підтримують можливість розподіленого

навчання, що дозволяє виконувати обчислення одночасно на декількох графічних процесорах або навіть на декількох обчислювальних вузлах. Це є важливим фактором при створенні масштабованих систем штучного інтелекту, де обсяг даних і складність моделей постійно зростають.

Важливою складовою є модуль тестування та оцінювання, який дозволяє визначити ефективність навченої моделі. У цьому модулі здійснюється перевірка моделі на тестових даних, що не використовувалися під час навчання, а також обчислення показників якості. До таких показників належать точність, повнота, F-міра, а також інші метрики, залежно від типу задачі (класифікація, регресія тощо). Додатково може виконуватись побудова матриці помилок, що дозволяє детально проаналізувати результати класифікації.

Для візуалізації результатів навчання використовується окремий модуль, який забезпечує побудову графіків та діаграм. Це дозволяє відслідковувати процес навчання у динаміці, аналізувати зміну функції втрат, точності та інших показників. Для реалізації візуалізації застосовуються бібліотеки, такі як Matplotlib та Seaborn, які надають широкий набір інструментів для графічного представлення даних.

Окрему роль у структурі програмного забезпечення відіграє інтерфейс користувача, який забезпечує взаємодію людини з системою. Через інтерфейс користувач має можливість виконувати всі основні операції: завантаження даних, налаштування параметрів навчання, запуск процесу навчання, перегляд результатів та їх аналіз. Інтерфейс може бути реалізований у різних формах, зокрема у вигляді командного рядка, графічного додатку або веб-інтерфейсу. Для створення веб-орієнтованих систем можуть використовуватись фреймворки, такі як Flask або Django.

Крім основних модулів, програмне забезпечення комплексу може включати допоміжні компоненти, такі як модуль логування, який відповідає за запис інформації про процес навчання, помилки та результати роботи системи. Це дозволяє здійснювати аналіз роботи комплексу та спрощує процес

налагодження. Також може бути реалізований модуль керування конфігураціями, який дозволяє зберігати та повторно використовувати налаштування експериментів.

Взаємодія між програмними модулями реалізується за допомогою внутрішніх програмних інтерфейсів, що забезпечують передачу даних між компонентами системи. Наприклад, модуль підготовки даних передає оброблені дані до модуля навчання, який, у свою чергу, передає результати до модуля оцінювання та візуалізації. Такий підхід дозволяє забезпечити узгоджену роботу всіх компонентів та підвищує ефективність функціонування комплексу.

3.3 Опис роботи апаратної частини комплексу та середовища симуляції

Апаратна частина апаратно-програмного комплексу навчання штучних нейронних мереж забезпечує виконання всіх обчислювальних процесів, пов’язаних із обробкою даних, навчанням моделей та збереженням результатів. Її функціонування базується на взаємодії основних компонентів: центрального процесора, графічного процесора, оперативної пам’яті, накопичувачів даних та допоміжних підсистем.

У процесі роботи комплексу центральний процесор (CPU) виконує функції керування всією системою, обробляє логіку програмного забезпечення та координує обмін даними між компонентами. Він забезпечує виконання операцій підготовки даних, керування процесом навчання та взаємодію з користувачем. Водночас основне навантаження при навчанні нейронних мереж покладається на графічний процесор (GPU), який виконує масивні паралельні обчислення. Завдяки великій кількості обчислювальних ядер GPU ефективно обробляє матричні операції, що є основою алгоритмів глибинного навчання.

Оперативна пам’ять (RAM) використовується для тимчасового зберігання даних, які надходять у процесі роботи системи. Зокрема, у ній зберігаються

батчі навчальних даних, проміжні результати обчислень, а також параметри нейронної мережі під час її навчання. Швидкість доступу до оперативної пам’яті безпосередньо впливає на загальну продуктивність системи.

Модуль накопичення даних виконує функції довготривалого зберігання інформації. З нього відбувається завантаження навчальних вибірок у оперативну пам’ять, а також запис результатів навчання, зокрема вагових коефіцієнтів моделі та логів. У процесі роботи системи дані постійно передаються між накопичувачем і оперативною пам’яттю, що формує безперервний потік інформації.

Передача даних між компонентами апаратної частини здійснюється через системну шину (наприклад, PCI Express), яка забезпечує високошвидкісний обмін між CPU, GPU та накопичувачами. Важливу роль також відіграють контролери пристроїв, які координують роботу окремих апаратних елементів та оптимізують обмін даними.

У процесі навчання нейронної мережі робота апаратної частини відбувається у декілька етапів. Спочатку дані завантажуються з накопичувача до оперативної пам’яті, після чого передаються до GPU для виконання обчислень. Результати обробки повертаються до оперативної пам’яті та можуть зберігатися на накопичувачі або використовуватися для подальших обчислень. Такий цикл повторюється багаторазово протягом усього процесу навчання.

Для підвищення надійності роботи апаратної частини використовуються системи моніторингу, які контролюють стан компонентів. Зокрема, застосовуються датчики температури процесора, графічного процесора та накопичувачів, що дозволяє запобігти перегріву. Також можуть використовуватись механізми контролю навантаження, енергоспоживання та стану пам’яті.

3.4 Експериментальні дослідження та аналіз результатів роботи комплексу

Практична реалізація описаного інформаційного конвеєра та інтерфейсу користувача була виконана на базі мови програмування Python із використанням сучасних спеціалізованих бібліотек і фреймворків. Для створення обчислювального ядра системи та організації архітектури штучної нейронної мережі було використано фреймворк PyTorch.

Вибір цього інструменту обумовлений його високою інтеграцією з низькорівневими бібліотеками NVIDIA CUDA та cuDNN, що дозволяє автоматично транслювати високорівневі матричні операції у конфігураційні команди для тензорних і CUDA-ядер графічного процесора. Підготовка даних на рівні передпроцесингу, включаючи нормалізацію числових значень та очищення від аномалій, реалізована за допомогою засобів NumPy та Pandas, які оптимізовані для швидкої обробки великих масивів даних у системній оперативній пам'яті.

Для перевірки працездатності та оцінки ефективності розробленого апаратно-програмного комплексу було проведено серію експериментальних досліджень. Головною метою тестування було визначення реальної швидкості навчання штучної нейронної мережі при використанні спроєктованого інформаційного конвеєра, а також аналіз точності отриманої моделі на тестовій вибірці даних.

Експерименти проводилися з використанням тестового набору даних, який перед початком процесу ділився у класичному співвідношенні: 70% даних виділялося на безпосереднє навчання, 15% на валідацію моделі під час проходження епох та 15% для підсумкового контрольного тестування якості.

На першому етапі досліджень було виконано порівняльний аналіз часових витрат на розрахунки при різних конфігураціях системи. Для цього один і той самий алгоритм навчання запускався у трьох різних режимах: суто на центральному процесорі з послідовною обробкою завдань, на графічному прискорювачі без використання технології прямого доступу до пам'яті (коли дані передавалися класичним шляхом через RAM), та на повністю

налаштованому комплексі з активованим NVMe-конвеєром, PCIe-комутатором і механізмом подвійної буферизації.

Результати вимірювання середнього часу обробки однієї епохи навчання та загального часу виконання процесу для 100 епох наведено у таблиці 3.1.

Таблиця 3.1 – Результати дослідження швидкодії комплексу при різних конфігураціях

Режим роботи обчислювальної платформи	Середній час однієї епохи, с	Загальний час (100 епох), хв	Прискорення процесу, разів
Обчислення лише на CPU (послідовно)	142.5	237.5	1.0 (базовий)
Обчислення на GPU (без оптимізації шини)	18.2	30.3	7.8
Апаратно-програмний комплекс з DMA та NVMe	4.1	6.8	34.7

Аналіз отриманих цифрових даних показує, що перенесення математичних операцій на паралельну архітектуру графічного процесора дозволяє суттєво зменшити тривалість розрахунків. Водночас максимальну ефективність продемонстрував саме розроблений комплекс, у якому завдяки прямому обміну даними між накопичувачем та відеопам'яттю GPU в обхід системних ресурсів процесора вдалося усунути затримки на шині передачі інформації. Це забезпечило загальне прискорення процесу навчання у 34.7 раза

порівняно з базовою конфігурацією на CPU.

Асинхронне зчитування інформації з твердотільного накопичувача NVMe та механізм подвійної буферизації (double buffering) реалізовано за допомогою стандартних програмних інструментів паралелізму. Для цього було задіяно вбудовані модулі багатопотоковості та асинхронності, які дозволили розділити процес на два незалежні потоки, що виконуються паралельно. Перший потік відповідає за взаємодію з накопичувачем через неко блокуючі системні виклики та заповнення черг оперативної пам'яті, тоді як другий потік контролює передачу вже сформованих батчів через шину PCI Express безпосередньо у відеопам'ять GPU. Такий підхід дозволив повністю розділити логіку підготовки даних та логіку обчислень, що й забезпечило зафіксоване в експериментах зменшення часу простою обчислювальних кластерів прискорювача.

Графічний інтерфейс користувача та панель керування комплексом було розроблено з використанням бібліотеки PyQt. Цей інструмент дозволив створити легке вікно програми з мінімальним споживанням системних ресурсів, що є важливим, аби софт керування не відбирав оперативну пам'ять та потужність CPU у модулів передпроцесингу. Інтеграція графіків зміни точності та помилки в реальному часі реалізована за допомогою вбудованого в інтерфейс модуля pyqtgraph, який адаптований для швидкого оновлення лінійних графіків на основі масивів даних без затримок у роботі інтерфейсу. Для збору та відображення даних телеметрії заліза (навантаження ядер, заповнення VRAM та температурні показники) було використано бібліотеку psutil для моніторингу центрального процесора та оперативної пам'яті, а також програмний інтерфейс NVIDIA Management Library (NVML) для отримання точних параметрів стану графічного прискорювача «на льоту». Зв'язок між модулем обчислень та графічним інтерфейсом організовано через систему сигналів і слотів, що гарантує стабільну роботу програми без зависань основного вікна під час тривалих циклів навчання моделей.

На другому етапі експериментів було проведено аналіз динаміки

навчання нейронної мережі та оцінку точності розпізнавання. Протягом усього циклу, що складався зі 100 епох, програмний модуль візуалізації фіксував зміну значень функції втрат та метрики точності. Було встановлено, що процес навчання проходив стабільно, без різких коливань, що свідчить про правильний вибір початкової швидкості навчання та оптимальний розмір сформованих батчів даних.

Графіки зміни значень точності та помилки моделі для навчальної та валідаційної вибірок відображено на рисунку 3.3.

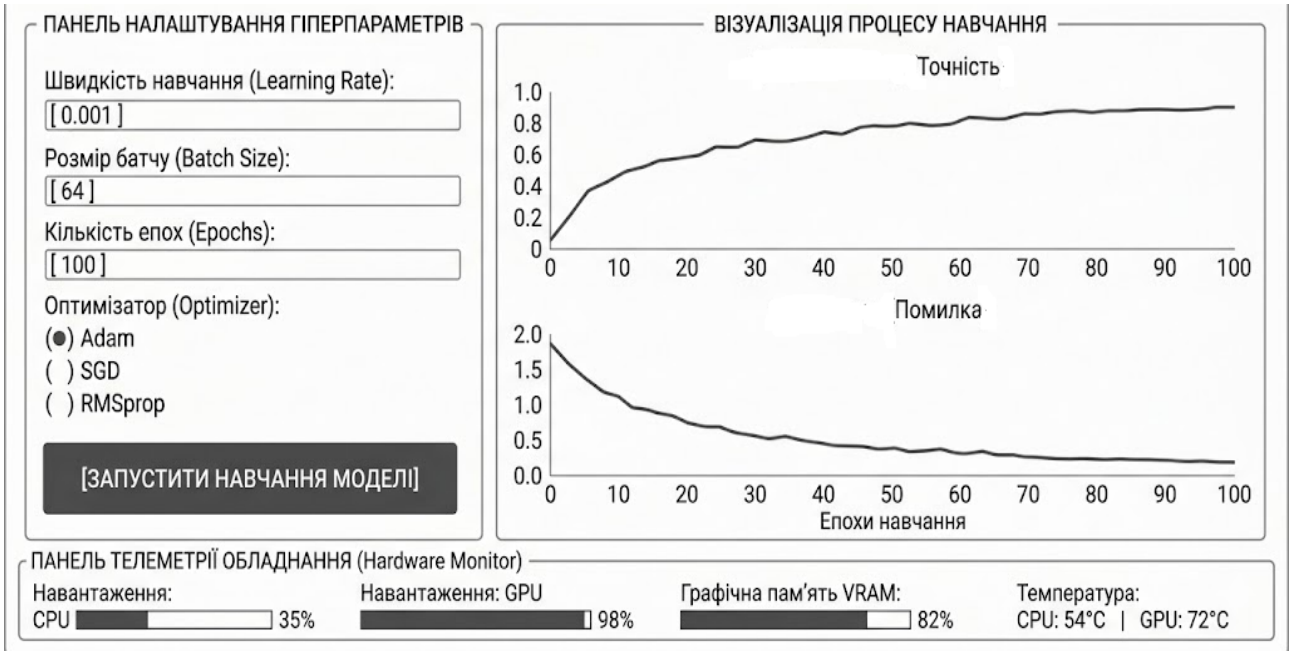


Рисунок 3.3 – Інтерфейс користувача та панель керування комплексом

Як видно з графічних залежностей, після 70-ї епохи криві точності та помилки виходять на стабільне плато. Показник похибки для навчальної вибірки знизився до мінімального значення, а точність досягла стабільних високих результатів. Важливим є те, що криві валідації повторюють траєкторію навчальних кривих без розходження в різні боки на фінальних етапах, що підтверджує відсутність ефекту перенавчання моделі та вказує на хорошу здатність системи до узагальнення.

Після завершення 100 епох навчена модель була передана до модуля тестування для оцінки її роботи на відкладених контрольних даних, які мережа не бачила під час навчання. За результатами обробки тестового набору програмний модуль розрахував підсумкові класифікаційні метрики системи. Загальний коефіцієнт точності комплексу на тестових даних склав 0.942, показник точності дорівнює 0.938, а значення повноти зафіксовано на рівні 0.945.

Підсумкова F1-міра, яка є гармонійним середнім між точністю та повнотою, склала 0.941. Отримані експериментальні результати повністю відповідають поставленим технічним вимогам до проєкту та підтверджують високу ефективність як апаратної конфігурації платформи, так і структури розробленого програмного забезпечення інформаційного конвеєра.

3.5 Висновки до третього розділу

У третьому розділі описано практичну побудову комплексу та наведено результати вимірювання його роботи. Програмна частина створена за модульним принципом на мові Python із використанням готової бібліотеки PyTorch та графічного інтерфейсу для користувача. Система складається з окремих взаємопов’язаних частин, які відповідають за завантаження даних, безпосереднє навчання моделі, перевірку її якості, побудову графіків та контроль стану комп'ютера.

Експерименти підтвердили, що розроблений спосіб організації потоків даних дозволяє уникнути затримок при зчитуванні файлів із диска.

Завдяки паралельній підготовці даних та використанню швидкісних каналів передачі безпосередньо у пам'ять графічного процесора, загальний час навчання зменшився у 34.7 раза порівняно зі звичайним центральним процесором. Процес навчання відбувався стабільно, без ознак погіршення результатів на нових даних. Під час підсумкової перевірки створена модель

показала високу точність розпізнавання на рівні 94.2%.

ВИСНОВКИ

У роботі за результатами виконаних теоретичних та практичних досліджень було розроблено апаратно-програмних комплекс навчання штучних нейронних мереж.

У першому розділі проведено аналіз предметної області досліджено особливості обчислень при роботі зі штучними нейронними мережами та проведено аналіз платформ для їх навчання. Навчання таких моделей потребує виконання дуже великої кількості однотипних математичних операцій, зокрема множення матриць. Через це звичайні центральні процесори не можуть забезпечити потрібну швидкість обробки даних, адже вони мають порівняно мало ядер і розраховані на послідовне виконання завдань.

Порівняння різних апаратних платформ (центральних, графічних, тензорних процесорів та програмованих схем) показало, що найбільш універсальним та доступним рішенням є графічні процесори. Вони мають тисячі дрібних ядер, які дозволяють виконувати багато обчислень одночасно. Також розглянуто сучасне програмне забезпечення: низькорівневі інструменти керування обладнанням та високорівневі готові середовища розробки, такі як TensorFlow і PyTorch. Зроблено висновок, що для розв'язання поставленої задачі необхідно створити збалансовану систему, де компоненти підібрані так, щоб швидкість передачі даних відповідала потужності обчислювального пристрою.

У другому розділі кваліфікаційної роботи проведено аналіз, проєктування та обґрунтування рішень для створення апаратно-програмного комплексу навчання штучних нейронних мереж.

Встановлено, що поділ структури GPU на мультипроцесорні кластери, тензорні та стандартні обчислювальні ядра дозволяє паралельно виконувати матричні операції. Це забезпечує збільшення швидкості обчислень порівняно з послідовною обробкою завдань на центральному процесорі.

Розроблено та описано модель апаратної платформи комплексу. Визначено логіку взаємодії та зв'язки між центральним процесором, графічним прискорювачем, оперативною пам'яттю та накопичувачем даних. Визначено роль CPU як керівного елемента системи, який відповідає за загальну логіку програми, передачу даних та моніторинг стану обладнання, тоді як GPU виконує функцію співпроцесора для математичних розрахунків.

Обґрунтовано впровадження інтерфейсів обміну даними для запобігання затримкам при зчитуванні інформації. Використання протоколу NVMe та технології прямого доступу до пам'яті дозволило організувати передачу даних із накопичувача у відеопам'ять GPU через комутатор в обхід основних ресурсів процесора та оперативної пам'яті.

Спроєктовано трирівневу структуру програмного забезпечення комплексу, яка складається з рівня збереження даних, рівня попередньої обробки на CPU та рівня обчислень на GPU. Завдяки розробці програмного механізму подвійної буферизації забезпечено конвеєрну обробку інформації, де процес підготовки наступного пакета даних виконується одночасно з розрахунками поточної епохи навчання, що зменшує час простою обчислювальних ядер.

У третьому розділі описано практичну побудову комплексу та наведено результати вимірювання його роботи. Програмна частина створена за модульним принципом на мові Python із використанням готової бібліотеки PyTorch та графічного інтерфейсу для користувача. Система складається з окремих взаємопов’язаних частин, які відповідають за завантаження даних, безпосереднє навчання моделі, перевірку її якості, побудову графіків та контроль стану комп'ютера.

Експерименти підтвердили, що розроблений спосіб організації потоків даних дозволяє уникнути затримок при зчитуванні файлів із диска.

Завдяки паралельній підготовці даних та використанню швидкісних каналів передачі безпосередньо у пам'ять графічного процесора, загальний час

навчання зменшився у 34.7 раза порівняно зі звичайним центральним процесором. Процес навчання відбувався стабільно, без ознак погіршення результатів на нових даних. Під час підсумкової перевірки створена модель показала високу точність розпізнавання на рівні 94.2%.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Хома Ю. В., Бенч А. Я. Порівняльний аналіз програмно-апаратного забезпечення алгоритмів глибокого навчання. *Computer systems and networks*. 2019. № 1. С. 97-102.

2. Chen M., Challita U., Saad W., Yin C., Debbah M. Artificial neural networks-based machine learning for wireless networks: A tutorial. *IEEE Communications Surveys & Tutorials*. 2019. Vol. 21. №4. P. 3039–3071.

3. Montesinos López O. A., Montesinos López A., Crossa J. Fundamentals of artificial neural networks and deep learning. *Multivariate statistical machine learning methods for genomic prediction*. *Cham: Springer International Publishing*. 2022. P. 379–425.

4. Franceschi M., Nannarelli A., Valle M. Tunable floating-point for artificial neural networks. *2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*: *Conference Proceedings*. 2018. P. 289–292.

5. Wang N., Choi J., Brand D., Chen C. Y., Gopalakrishnan K. Training deep neural networks with 8-bit floating point numbers. *Advances in Neural Information Processing Systems*. 2018. Vol. 31.

6. Hsu K. C., Tseng H. W. Accelerating applications using edge tensor processing units. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 2021. P. 1–14.

7. Dave S., Baghdadi R., Nowatzki T., Avancha S., Shrivastava A., Li B. Hardware acceleration of sparse and irregular tensor computations of ml models: A survey and insights. *Proceedings of the IEEE*. 2021. Vol. 109. № 10. P. 1706–1752.

8. Ali D., Rehman A. U., Khan F. H. Hardware accelerators and accelerators for machine learning. *2022 International Conference on IT and Industrial Technologies (ICIT)*. 2022. P. 1–7.

9. Vasile C. E., Ulmămei A. A., Bîră C. Image processing hardware acceleration - a review of operations involved and current hardware approaches.

Journal of Imaging. 2024. Vol. 10. № 12. P. 298.

10. Hwang I., Lee J., Kang H., Lee G., Kim H. Survey of CPU and memory simulators in computer architecture: A comprehensive analysis including compiler integration and emerging technology applications. *Simulation Modelling Practice and Theory*. 2025. Vol. 138. P. 103032.

11. AlShekh R. H., Dawwd S. A., Qassabbashi F. N. Comparative Review of Multicore Architectures: Intel, AMD, and ARM in the Modern Computing Era. *Chips*. 2025. Vol. 4. P. 44.

12. Dally W. J., Keckler S. W., Kirk D. B. Evolution of the graphics processing unit (GPU). *IEEE Micro*. 2021. Vol. 41. №. 6. P. 42–51.

13. Madiajagan M., Raj S. S. Parallel computing, graphics processing unit (GPU) and new hardware for deep learning in computational intelligence research. *Deep learning and parallel computing environment for bioengineering systems*. *Academic Press*. 2019. P. 1–15.

14. Al-Mouhamed M. A., Khan A. H., Mohammad N. A review of CUDA optimization techniques and tools for structured grid computing. *Computing*. 2020. Vol. 102. №4. P. 977.

15. ALHafez N., Kurdi A. Parallel Paradigms in Modern HPC: A Comparative Analysis of MPI, OpenMP, and CUDA. *arXiv preprint arXiv:2506.15454*. 2025.

16. Shafie Khorassani K., Hashmi J., Chu C. H., Chen C. C., Subramoni H., Panda D. K. Designing a ROCm-aware MPI library for AMD GPUs: early experiences. *International Conference on High Performance Computing* (June 2021). *Cham: Springer International Publishing*. 2021. P. 118–136.

17. Hsu K. C., Tseng H. W. Accelerating applications using edge tensor processing units. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 2021. P. 1–14.

18. Mochurad L., Dolynska K., Ufimtseva T. Scalable parallel training of deep neural networks for image classification using tensor processing units. *Computer systems and information technologies*. 2025. №. 1. P. 100–110.

19. Shahid A., Mushtaq M. A survey comparing specialized hardware and evolution in TPUs for neural networks. *2020 IEEE 23rd International Multitopic Conference (INMIC)*. 2020. P. 1–6.

20. Романюк О. Н., Майданюк В. П., Романюк С. О., Костенко А. В. Архітектурні особливості та функціональні можливості сучасних ПЛІС. *Стан, досягнення і перспективи інформаційних систем і технологій*. 2026. С. 199.

21. Boutros A., Betz, Tran H. Q. FPGAs: architecture, design flow, and emerging applications - industrial perspectives. *Analog Integrated Circuits and Signal Processing*. 2026. Vol. 127. № 1. 21(2). P. 4–29.

22. Tran H. Q. FPGAs: architecture, design flow, and emerging applications - industrial perspectives. *Analog Integrated Circuits and Signal Processing*. 2026. Vol. 127. № 1. P. 2.

23. Vai M. M., Song W. S., Tyrrell B. M. Application-specific integrated circuits. *High Performance Embedded Computing Handbook*. *CRC Press*. 2018. P. 191–215.

24. Lou H. Comparative analysis and enhancement of FPGA and ASIC performance in targeted applications. *International Conference on Artificial Intelligence and Communication Technology*. *Singapore: Springer Nature Singapore.* 2024. P. 295–308.

25. Dehal R. S., Munjal C., Ansari A. A., Kushwaha A. S. Gpu computing revolution: Cuda. *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*. 2018. P. 197–201.

26. Al-Mouhamed M. A., Khan A. H., Mohammad N. A review of CUDA optimization techniques and tools for structured grid computing. *Computing*. 2020. Vol. 102. № 4. P. 977.

27. Imambi S., Prakash K. B., Kanagachidambaresan G. R. PyTorch. Programming with TensorFlow: solution for edge computing applications. *Cham: Springer International Publishing*. 2021. P. 87–104.

28. Novac O. C., Chirodea M. C., Novac C. M., Bizon N., Oproescu M., Stan

O. P., Gordan C. E. Analysis of the application efficiency of TensorFlow and PyTorch in convolutional neural network. *Sensors*. 2022. Vol. 22. № 22. P. 8872.

29. Pang B., Nijkamp E., Wu Y. N. Deep learning with tensorflow: A review. *Journal of Educational and Behavioral Statistics*. 2020. Vol. 45. № 2. P. 227–248.

30. Singh P., Manure A. Introduction to tensorflow 2.0. Learn TensorFlow 2.0: Implement machine learning and deep learning models with python. *Berkeley, CA: Apress*. 2019. P. 1–24.

31. Carmona-Martínez A., Bernabé G., García J. M. Characterization of machine learning compilers for LLM inference on NVIDIA GPUs: A. Carmona-Martínez et al. *The Journal of Supercomputing*. 2026. Vol. 82. № 7. P. 420.

32. Xia C., Zhao J., Sun Q., Wang Z., Wen Y., Yu T., Cui H. Optimizing deep learning inference via global analysis and tensor expressions. *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. 2024. Vol 1. P. 286–301.

33. Abdelkhalik H., Arafa Y., Santhi N., Badawy A. H. A. Demystifying the nvidia ampere architecture through microbenchmarking and instruction-level analysis. *2022 IEEE High Performance Extreme Computing Conference (HPEC)*. 2022. P. 1–8.

34. Luo W., Fan R., Li Z., Du D., Wang Q., Chu X. Benchmarking and dissecting the nvidia hopper gpu architecture. *2024 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 2024. P. 656–667.

35. Tran H. N., Cambria E. A survey of graph processing on graphics processing units. *The Journal of Supercomputing*. 2018. Vol. 74. № 5. P. 2086–2115.

36. Wu L., Huang L., Xiong T. Designing a unified architecture graphics processing unit. *The 7th International Conference on Computer Engineering and Networks*. 2017. Vol. 299. P. 079.

37. Allen T., Ge R. Characterizing power and performance of gpu memory access. *2016 4th International Workshop on Energy Efficient Supercomputing (E2SC)*. 2016. P. 46–53.

38. Crago N. C., Stephenson M., Keckler S. W. Exposing memory access patterns to improve instruction and memory efficiency in GPUs. *ACM Transactions on Architecture and Code Optimization (TACO)*. 2018. Vol. 15. № 4. P. 1–23.

39. Zayer R., Steinberger M., Seidel H. P. Sparse matrix assembly on the GPU through multiplication patterns. *2017 IEEE High Performance Extreme Computing Conference (HPEC)*. 2017. P. 1–8.

40. Szustak L., Bratek P. Performance portable parallel programming of heterogeneous stencils across shared-memory platforms with modern Intel processors. *The International Journal of High Performance Computing Applications*. 2019. Vol. 33. № 3. P. 534–553.

41. Merrill D., Garland M. Merge-based parallel sparse matrix-vector multiplication. *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 2016. P. 678–689.

42. Jiang X., Kim M., Lauter K., Song Y. Secure outsourced matrix computation and application to neural networks. *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 2018. P. 1209–1222.

43. Wang Y., Wei G. Y., Brooks D. A systematic methodology for analysis of deep learning hardware and software platforms. *Proceedings of Machine Learning and Systems*. 2020. Vol. 2. P. 30–43.

44. Capra M., Bussolino B., Marchisio A., Masera, G., Martina, M., Shafique M. Hardware and software optimizations for accelerating deep neural networks: Survey of current trends, challenges, and the road ahead. *IEEE Access*. 2020. Vol. 8. P. 225134–225180.

45. Gadiyar R., Zhang T., Sankaranarayanan A. Artificial intelligence software and hardware platforms. *Artificial intelligence for autonomous networks*. *Chapman and Hall/CRC*. 2018. P. 165–188.

46. Messaoud S., Bouaafia S., Maraoui A., Ammari A. C., Khriji L., Machhout M. Deep convolutional neural networks-based Hardware–Software on-chip system for computer vision application. *Computers & Electrical Engineering*. 2022. Vol. 98.

P. 107671.

47. Modasshir M., Li A. Q., Rekleitis I. Deep neural networks: a comparison on different computing platforms. *2018 15th Conference on Computer and Robot Vision (CRV)*. 2018. P. 383–389.

48. Capra M., Bussolino B., Marchisio A., Shafique M., Masera G., Martina M. An updated survey of efficient hardware architectures for accelerating deep convolutional neural networks. *Future Internet*. 2020. Vol. 12. № 7. P. 113.

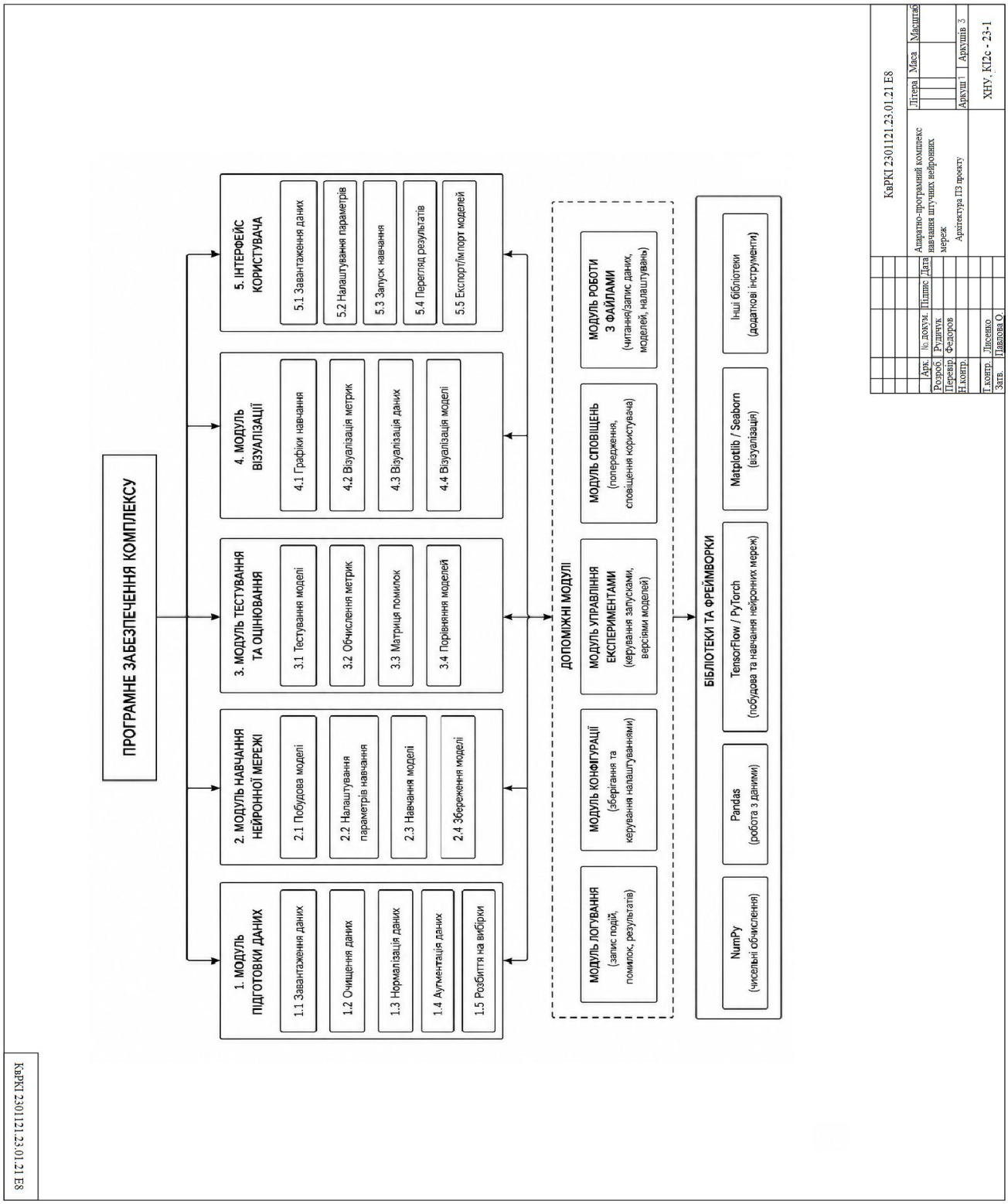
49. Zhu H., Akrout M., Zheng B., Pelegris A., Jayarajan A., Phanishayee A., Pekhimenko G. Benchmarking and analyzing deep neural network training. *2018 IEEE International Symposium on Workload Characterization (IISWC)*. 2018. P. 88–100.

50. Wang E., Davis J. J., Zhao R., Ng H. C., Niu X., Luk W., Constantinides G. A. Deep neural network approximation for custom hardware: Where we've been, where we're going. *ACM Computing Surveys (CSUR)*. 2019. Vol. 52. № 2. P. 1–39.

ДОДАТОК А

(обов’язковий)

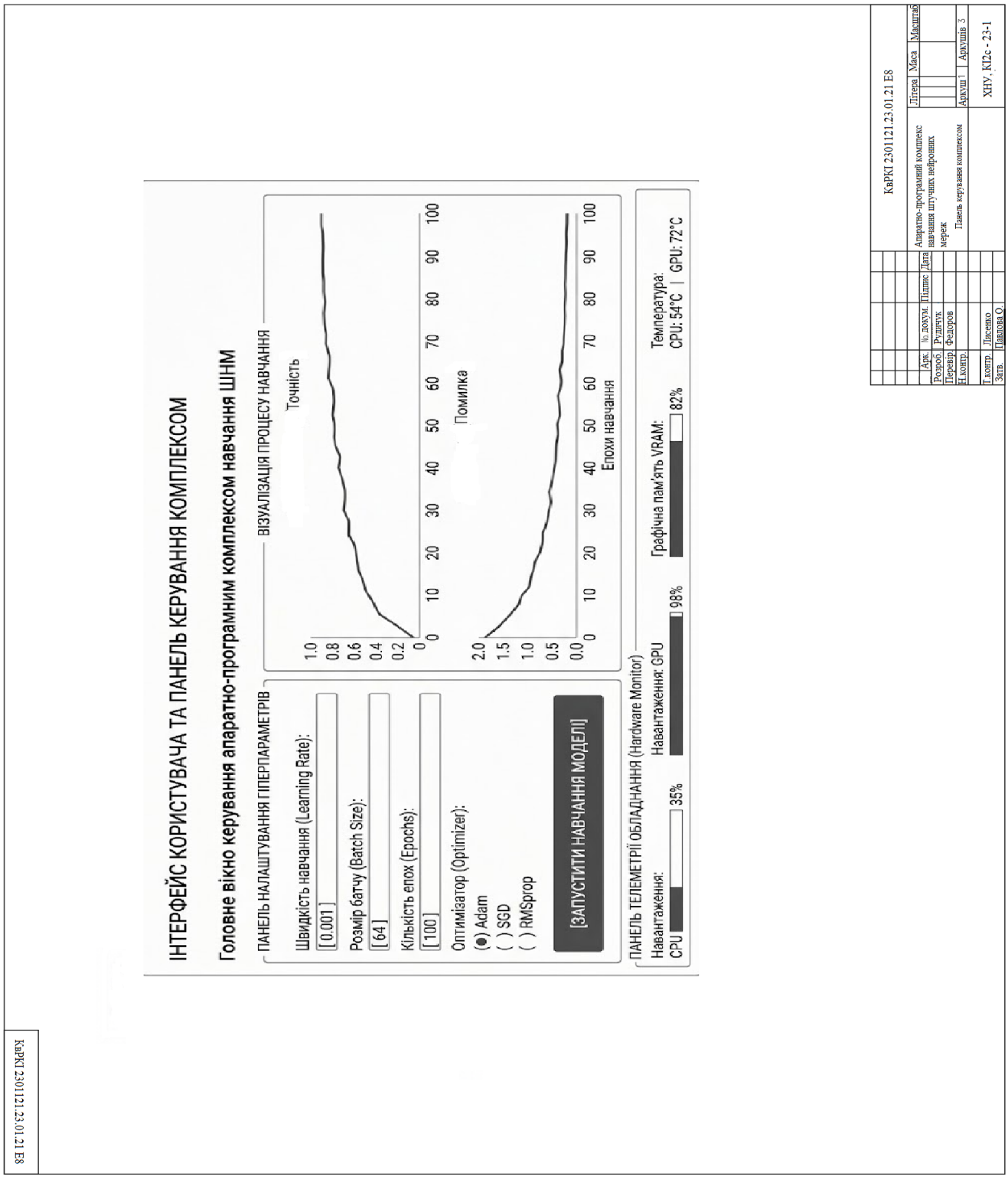
Копія креслення «Архітектура ПЗ проєкту»



ДОДАТОК Б

(обов’язковий)

Копія креслення «Панель керування комплексом»



ДОДАТОК В

(обов’язковий)

Копія креслення «Апаратне забезпечення проєкту»

Протокол аналізу звіту подібності експертом

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Дмитро РУДИЧУК

Співавтор:

Назва: Апаратно-програмний комплекс навчання штучних нейронних мереж

Експерт: Євген ФЕДОРОВ

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:6.08%

Коефіцієнт подібності 2:0.91%

Мікропробіли: 100

Заміна букв: 2

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2026-05-30 17:31:01.0

Після аналізу Звіту подібності констатую наступне:

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

2026-05-30 Доцент Андрій Ніченорук

Дата експерт

Anti-Plagiarism (http://ap.km.ua) v-15.701

Максимальне співпадіння з одним документом 1.0%

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

ID: 272849
Назва: БКР Апаратно-програмний комплекс навчання штучних нейронних мереж
Додано в БД: 2026-05-30
Автора: Дмитро РУДИЧУК
Керівники: Євген ФЕДОРОВ
Консультанти:
Опоненти:

Документ		Сумарний збіг по Базі Даних	
Символи	Лексеми	Символи	Лексеми
109799	752	1089 (1%)	16 (2%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Рудичук Дмитро Анатолійович

Тема: Апаратно-програмний комплекс навчання штучних нейронних мереж

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3 Кількість сторінок записки 67

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є забезпечення безперервного автоматизованого контролю, підвищення ефективності та швидкодії процесу навчання штучних нейронних мереж шляхом проєктування та практичної реалізації апаратно-програмного комплексу з оптимізованим конвеєром передачі та попередньої обробки даних у режимі реального часу.

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому розділі проведено аналіз предметної області досліджено особливості обчислень при роботі зі штучними нейронними мережами та проведено аналіз платформ для їх навчання. Навчання таких моделей потребує виконання дуже великої кількості однотипних математичних операцій, зокрема множення матриць. Через це звичайні центральні процесори не можуть забезпечити потрібну швидкість обробки даних, адже вони мають порівняно мало ядер і розраховані на послідовне виконання завдань.

У другому розділі кваліфікаційної роботи проведено аналіз, проєктування та обґрунтування рішень для створення апаратно-програмного комплексу навчання штучних нейронних мереж.

Встановлено, що поділ структури GPU на мультипроцесорні кластери, тензорні та стандартні обчислювальні ядра дозволяє паралельно виконувати матричні операції. Це забезпечує збільшення швидкості обчислень порівняно з послідовною обробкою завдань на центральному процесорі.

Розроблено та описано модель апаратної платформи комплексу. Визначено логіку взаємодії та зв'язки між центральним процесором, графічним прискорювачем, оперативною пам'яттю та накопичувачем даних. Визначено роль CPU як керівного елемента системи, який відповідає за загальну логіку програми, передачу даних та моніторинг стану обладнання, тоді як GPU виконує функцію співпроцесора для математичних розрахунків.

У третьому розділі описано практичну побудову комплексу та наведено результати вимірювання його роботи. Програмна частина створена за модульним принципом на мові Python із використанням готової бібліотеки PyTorch та графічного інтерфейсу для користувача. Система складається з окремих взаємопов'язаних частин, які відповідають за завантаження даних, безпосереднє навчання моделі, перевірку її якості, побудову графіків та контроль стану комп'ютера.

4. Позитивні сторони роботи: практична цінність роботи.

5. Негативні сторони роботи: недостатня увага приділена обрахунку ефективності запропонованих рішень.

6. *Оцінка* графічного оформлення та пояснювальної записки роботи: *Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.*

7. *Відгук про роботу в цілому*: Робота виконана на належному технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: добре (75, С)

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) _____

Онищко О. Г., доцент ІПЗ, ХНУ

"01" червня 2026 р. _____ (підпис)

Зав. кафедри КІС
д-р. філософії Ользі ПАВЛОВІЙ

Дмитро РУДИЧУК
ПІБ здобувача вищої освіти

ФІТ, 3 курсу, групи КІ2с-23-1

ЗАЯВА

З правилами чинного Положення про систему забезпечення академічної доброчесності у Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів академічної відповідальності, ознайомлений (а). Про використання спеціалізованих програмних засобів (СПЗ) StrikePlagiarism та Anti-Plagiarism для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений (а). Надаю університету право на передачу моєї роботи для обробки та збереження в базах даних СПЗ і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються СПЗ.

Також надаю свою згоду на обробку й збереження університетом моєї роботи в Інституційному репозитарії Хмельницького національного університету.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

1 травня 2026 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Назва кваліфікаційної роботи Апаратно-програмний комплекс навчання штучних нейронних мереж
Автор Дмитро РУДИЧУК
Освітня програма Комп'ютерна інженерія та програмування
Рівень вищої освіти перший (бакалаврський)
Спеціальність 123 Комп'ютерна інженерія
Науковий керівник д.т.н., проф. Євген ФЕДОРОВ

На основі аналізу кваліфікаційної роботи на дотримання вимог академічної доброчесності (у т.ч. відсутності ознак академічного плагіату) з урахуванням результатів перевірки роботи спеціалізованим програмним засобом(ами) комісія зробила такий висновок:

№	Висновок	Позначка про відповідність
1	Ознаки академічного плагіату	
1.1	Запозичення, виявлені в роботі, є законними і не є академічним плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних, якщо потрібно). Робота приймається до захисту.	відповідає
1.2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована.	
1.3	Виявлені запозичення не є академічним плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота може бути допущена до захисту після того як буде відкоригована та доопрацьована і успішно пройде повторну перевірку на академічний плагіат.	
1.4	Робота містить наявний текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
2	Інші види порушень академічної доброчесності	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) усі запозичення фрагментарні, або мають належним чином оформленні посилання;

2) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з джерелами на один фрагмент речення;

3) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

4) значна частина знайденого плагіату відноситься до списку використаних джерел

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів ідентичності/схожості StrikePlagiarism, складає 6,08%; та системою Anti-Plagiarism складає 1%, що, з урахуванням наведених обгрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

01.06.2026

Завідувач кафедри — Підпис — Ольга ПАВЛОВА (Ім'я, ПРІЗВИЩЕ)

Гарант освітньої програми — Підпис — Андрій НІЧЕПОРУК (Ім'я, ПРІЗВИЩЕ)

Керівник кваліфікаційної роботи — Федоров — Підпис — Євген ФЕДОРОВ (Ім'я, ПРІЗВИЩЕ)