КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій

(факультет)

інформаційних технологій

(кафедра)

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

на тему: «Методи прискорення навчання нейронних мереж за допомогою трансформерів»

Коцюба Владислав Андрійович
(прізвище, ім’я та по-батькові студента повністю)

Київ 2025 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій

_____

(факультет)

інформаційних технологій

_____

_____

(кафедра)

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТ
д.т.н., професор Гончаренко Т.А.

_____
“___” _____2025 року

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДО КВАЛІФІКАЦІЙНОЇ РОБОТИ

НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ БАКАЛАВРА

на тему: «Методи прискорення навчання нейронних мереж за допомогою трансформерів»

Виконав: студент 4-го курсу, групи КН-21

Спеціальності: 122 «Комп’ютерні науки»
(шифр і назва напряму підготовки, спеціальності)

Коцюба В.А.
(прізвище та ініціали)

Керівник д.т.н., проф. Бородавка Є.В.
(прізвище та ініціали)

Рецензент к.т.н., доц. Баліна О.І.
(прізвище та ініціали)

Київ 2025 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет: автоматизації і інформаційних технологій

Випускова кафедра: інформаційних технологій

Освітній ступінь: бакалавр

Спеціальність: 122 Комп'ютерні науки

Освітня програма: Інформаційні управляючі системи і технології

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТ
_____
„___" _____202_ року

З А В Д А Н Н Я
ДО ВИКОНАННЯ АТЕСТАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВР

Коцюба Владислав Андрійович

(прізвище, ім'я та по батькові здобувача)

1. Тема роботи: Методи прискорення навчання нейронних мереж за допомогою трансформерів

затверджена наказом ректора КНУБА № 2650/2 від 18.11.2025 року

2. Керівник роботи Бородавка Євгеній Влодимирович, д.т.н., професор кафедри інформаційних технологій

(прізвище, ім'я та по батькові, науковий ступінь, вчене звання)

3. Строк подання Здобувачем роботи до захисту_____

4. Зміст пояснювальної записки за розділами:

5. Графічний матеріал за розділами:

Р.1

_____

Р.2

_____

Р.3

_____

Р.4

_____

6. Календарний план виконання роботи:

Види робіт та їх зміст	Дата виконання
Розділ 1	Листопад 2024 р.
Розділ 2	Грудень 2024 р.
Розділ 3	Лютий 2025 р.
Розділ 4	Березень 2025 р.
Остаточне оформлення роботи	Березень 2025 р.
Направлення роботи для перевірки на плагіат	
Попередній захист роботи на випусковій кафедрі	
Направлення роботи на рецензування	

7. Консультанти розділів атестаційної випускної роботи

Розділ	Прізвище, ініціали та посада консультанта	Перевірив	
		дата	підпис
Розділ 1			
Розділ 2			
Розділ 3			
Розділ 4			

8. Дата видачі завдання _____

Зав. кафедри _____ _____

(підпис) (прізвище та ініціали)

Керівник _____ _____

(підпис) (прізвище та ініціали)

Здобувач _____ _____

(підпис) (прізвище та ініціали)

АНОТАЦІЯ

У роботі розроблено та експериментально досліджено методи прискорення навчання нейронних мереж з використанням трансформерних механізмів. Запропоновано комплексний підхід, що включає адаптивне формування міні-батчів на основі механізму уваги, градієнтний трансформер для оптимізації оновлення вагових коефіцієнтів та інтеграцію компонентів самоуваги в традиційні архітектури. Експериментальні дослідження продемонстрували суттєве прискорення процесу навчання для згорткових (32-38%), рекурентних (38-45%), повнозв'язних (25-32%) та трансформерних (30-36%) нейронних мереж без втрати точності. Розроблено програмну бібліотеку на базі TensorFlow, яка забезпечує просту інтеграцію запропонованих методів з існуючими моделями. Створено рекомендації щодо практичного застосування розроблених підходів для різних сценаріїв використання, включаючи навчання з обмеженими ресурсами, роботу з незбалансованими даними та розподілене навчання.

Ключові слова: нейронні мережі, прискорення навчання, трансформери, механізм уваги, адаптивна оптимізація, градієнтний трансформер, глибоке навчання.

ABSTRACT

The thesis presents the development and experimental research of methods for accelerating neural network training using transformer mechanisms. A comprehensive approach is proposed, which includes adaptive mini-batch formation based on the attention mechanism, gradient transformer for optimizing weight coefficient updates, and integration of self-attention components into traditional architectures. Experimental studies have demonstrated significant acceleration of the training process for convolutional (32-38%), recurrent (38-45%), fully connected (25-32%), and transformer (30-36%) neural networks without loss of accuracy. A TensorFlow-based software library has been developed, providing simple integration of the proposed methods with existing models. Practical recommendations have been created for applying the developed approaches to various use cases, including training with limited resources, working with imbalanced data, and distributed learning.

Keywords: neural networks, training acceleration, transformers, attention mechanism, adaptive optimization, gradient transformer, deep learning.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

АВР – атестаційна випускна робота

CNN (Convolutional Neural Network) – згорткова нейронна мережа

CPU (Central Processing Unit) – центральний процесор

CUDA – платформа паралельних обчислень від NVIDIA

FP16 – формат чисел з плаваючою комою половинної точності (16 біт)

FP32 – формат чисел з плаваючою комою одинарної точності (32 біти)

GPU (Graphics Processing Unit) – графічний процесор

GRU (Gated Recurrent Unit) – рекурентна нейронна мережа з керованими вентилями

IoU (Intersection over Union) – метрика для оцінки якості сегментації

LSTM (Long Short-Term Memory) – довга короткочасна пам'ять, тип рекурентної нейронної мережі

MAE (Mean Absolute Error) – середня абсолютна похибка

MLP (Multi-Layer Perceptron) – багатошаровий перцептрон

MSE (Mean Squared Error) – середня квадратична похибка

NAS (Neural Architecture Search) – пошук архітектури нейронної мережі

PDCA (Plan-Do-Check-Act) – цикл Демінга (планування-виконання-перевірка-дія)

ReLU (Rectified Linear Unit) – функція активації "випрямлений лінійний елемент"

RNN (Recurrent Neural Network) – рекурентна нейронна мережа

SGD (Stochastic Gradient Descent) – стохастичний градієнтний спуск

SVD (Singular Value Decomposition) – сингулярний розклад

TPU (Tensor Processing Unit) – тензорний процесор

t-SNE (t-distributed Stochastic Neighbor Embedding) – метод візуалізації високовимірних даних

ІС – інформаційна система

ЖЦ – життєвий цикл

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Розвиток глибокого навчання у останнє десятиліття суттєво розширив можливості штучного інтелекту у розв'язанні складних практичних задач, таких як розпізнавання образів, обробка природної мови та автоматичне прийняття рішень. Однак зростаюча складність нейромережевих моделей та обсяги даних для їх навчання призводять до значного зростання обчислювальних витрат. Тривалість навчання сучасних архітектур може становити дні або тижні навіть з використанням високопродуктивних обчислювальних систем, що суттєво обмежує потенціал практичного застосування таких технологій, особливо в умовах обмежених ресурсів.

Незважаючи на розвиток спеціалізованого апаратного забезпечення (GPU, TPU), питання алгоритмічної оптимізації процесу навчання залишається критичним. Традиційні методи оптимізації демонструють обмежену ефективність для архітектур з великою кількістю параметрів та високою варіативністю даних. Поява архітектури трансформерів з механізмом самоуваги відкрила нові можливості у обробці складних даних, проте потенціал застосування цих механізмів для прискорення навчання інших типів нейронних мереж залишається недостатньо дослідженим.

Розробка методів, які дозволяють прискорити процес навчання нейронних мереж без втрати їх точності, є актуальною науково-практичною задачею, розв'язання якої сприятиме розширенню сфери застосування глибокого навчання та підвищенню ефективності використання обчислювальних ресурсів.

Мета і завдання дослідження. Метою дослідження є розробка та експериментальне дослідження методів прискорення навчання нейронних мереж з використанням трансформерних механізмів.

Для досягнення мети в роботі поставлено такі завдання:

- Проаналізувати сучасні методи навчання нейронних мереж та існуючі підходи до прискорення цього процесу.
- Дослідити архітектуру трансформерів та механізми самоуваги з точки зору їх потенціалу для оптимізації процесів навчання.
- Розробити методи адаптивного формування міні-батчів на основі механізмів уваги для ефективнішого використання навчальних даних.
- Запропонувати підхід до модифікації градієнтів на основі трансформерної архітектури для оптимізації оновлення вагових коефіцієнтів.
- Розробити методи інтеграції механізмів самоуваги в традиційні архітектури нейронних мереж.
- Створити програмну реалізацію запропонованих методів з можливістю інтеграції в сучасні фреймворки глибокого навчання.
- Провести експериментальне дослідження ефективності розроблених методів на різних архітектурах нейронних мереж та наборах даних.
- Розробити рекомендації щодо практичного застосування запропонованих методів у різних сценаріях використання.

Об'єкт дослідження. Процеси навчання нейронних мереж різних архітектур.

Предмет дослідження. Методи прискорення навчання нейронних мереж з використанням трансформерних механізмів.

Методи дослідження ґрунтуються на теорії нейронних мереж, методах оптимізації, теорії трансформерів та самоуваги, методах обчислювальної математики та статистичного аналізу. Для експериментальної верифікації розроблених методів використано методи емпіричного дослідження та комп'ютерного моделювання.

Практичне значення отриманих результатів:

- Розроблені методи дозволяють прискорити навчання нейронних мереж у середньому на 30-45% без втрати точності моделей.

− Створено програмну бібліотеку на базі TensorFlow, яка забезпечує просту інтеграцію запропонованих методів з існуючими моделями.

− Запропоновані підходи демонструють особливу ефективність для складних задач з незбалансованими даними та моделей з великою кількістю параметрів.

− Розроблено рекомендації щодо оптимального застосування запропонованих методів у різних сценаріях, включаючи навчання з обмеженими ресурсами та розподілене навчання.

Структура та обсяг роботи. Магістерська дисертація складається зі вступу, чотирьох розділів, висновків, списку використаних джерел із 40 найменувань та додатків. Загальний обсяг роботи становить 104 сторінок, у тому числі 98 сторінок основного тексту та 11 рисунків.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз методів навчання нейронних мереж

Навчання нейронних мереж являє собою процес налаштування вагових коефіцієнтів з метою мінімізації функції втрат на наборі даних. Цей процес сформував основу сучасних систем штучного інтелекту, котрі застосовуються в різноманітних сферах від розпізнавання образів до обробки природної мови. Методи навчання нейронних мереж постійно вдосконалюються та розвиваються, оскільки існуючі алгоритми часто стикаються з проблемами повільної збіжності, локальних мінімумів та великих обчислювальних витрат.

Рис. 1.1 - Класифікація методів навчання нейронних мереж

Стохастичний градієнтний спуск (SGD) становить базовий алгоритм навчання нейронних мереж. Принцип його роботи полягає в оновленні вагових коефіцієнтів шляхом обчислення градієнта функції втрат на невеликих підмножинах даних (міні-батчах). Цей підхід забезпечує компроміс між швидкістю обчислень та точністю оновлення параметрів. Попри відносну простоту, SGD часто демонструє нестабільну збіжність та чутливість до вибору

швидкості навчання, що спонукало дослідників розробляти більш ефективні модифікації [1].

Алгоритми адаптивного навчання, зокрема Adam, RMSprop та Adagrad, представляють собою еволюцію градієнтних методів з додатковими механізмами адаптації швидкості навчання для кожного параметра. Ці методи автоматично коригують величину кроку в залежності від історії градієнтів, що дозволяє уникнути проблем з вибором оптимальної швидкості навчання. Адаптивні методи демонструють кращу стійкість до різномасштабних градієнтів та прискорюють збіжність у порівнянні з класичним SGD в багатьох практичних задачах.

Методи другого порядку, такі як алгоритм Ньютона та метод L-BFGS, використовують інформацію про другі похідні (гессіан) для визначення оптимального напрямку оновлення вагових коефіцієнтів. Ці підходи демонструють швидшу збіжність у термінах кількості ітерацій, проте кожна ітерація потребує значно більших обчислювальних ресурсів. Для великих нейронних мереж обчислення та зберігання повної матриці других похідних часто стає неможливим, тому застосовуються квазі-ньютонівські методи та різноманітні апроксимації.

Технологія перенесення навчання (transfer learning) дозволяє використовувати знання, здобуті при навчанні на одному наборі даних, для прискорення навчання на іншому. Цей підхід передбачає попереднє навчання моделі на великому наборі даних з наступним доналаштуванням на цільовому завданні. Дослідження показують, що перенесення навчання суттєво скорочує час збіжності та підвищує якість моделей, особливо коли цільовий набір даних обмежений.

Методи регуляризації, включаючи L1/L2-регуляризацію, dropout та batch normalization, відіграють ключову роль у запобіганні перенавчанню та прискоренні збіжності. L1/L2-регуляризації додають штраф за великі значення вагових коефіцієнтів до функції втрат, що сприяє спрощенню моделі. Dropout

випадковим чином відключає нейрони під час навчання, змушуючи мережу розподіляти знання між усіма нейронами. Batch normalization нормалізує активації всередині мережі, що пом'якшує проблему внутрішнього зсуву коваріат та дозволяє використовувати вищі швидкості навчання [2].

Дистиляція знань представляє метод навчання, при якому менша "студентська" модель навчається відтворювати виходи більшої "вчительської" моделі. Цей підхід дозволяє створювати компактні моделі, які зберігають точність більших архітектур, але потребують менше обчислювальних ресурсів. Дистиляція знань ефективно передає узагальнені знання від складної моделі до простішої, використовуючи м'які мітки - розподіли ймовірностей, які несуть більше інформації, ніж бінарні класи.

Методи паралельного та розподіленого навчання використовують кілька обчислювальних пристроїв для одночасного оновлення вагових коефіцієнтів. Синхронні підходи акумулюють градієнти з усіх пристроїв перед оновленням моделі, тоді як асинхронні методи дозволяють кожному пристрою оновлювати спільну модель незалежно. Розподілене навчання суттєво скорочує час обчислень, проте вимагає ефективних стратегій комунікації між пристроями та може страждати від затримок синхронізації.

Curriculum learning та самонавчання (self-supervised learning) представляють підходи до організації процесу навчання. Curriculum learning пропонує подавати навчальні приклади в порядку зростання складності, що прискорює збіжність та покращує узагальнюючі здібності моделі. Самонавчання дозволяє моделі вивчати корисні представлення даних без явної розмітки, використовуючи природну структуру даних для формування навчальних сигналів. Ці методи дозволяють максимізувати ефективність використання доступних даних [3].

Мета-навчання (meta-learning) або "навчання навчатися" спрямоване на розробку алгоритмів, здатних швидко адаптуватися до нових завдань з мінімальною кількістю прикладів. Методи мета-навчання оптимізують процес

оптимізації, що дозволяє моделям вчитися більш ефективно. Цей підхід включає моделювання оптимізатора як параметричної функції, котра налаштовується для прискорення збіжності на певному класі задач.

Архітектура трансформерів суттєво змінила підходи до навчання моделей обробки послідовностей. Механізм самоуваги (self-attention) дозволяє моделі обробляти всі елементи послідовності паралельно, на відміну від рекурентних мереж, які працюють послідовно. Ця архітектурна особливість не лише прискорює обчислення, але й поліпшує здатність моделі вивчати залежності на великих відстанях, що критично для багатьох задач обробки тексту та інших послідовних даних.

Оптимізація гіперпараметрів становить невід'ємну частину процесу навчання, оскільки вибір оптимальних значень швидкості навчання, розміру партії, архітектури мережі та інших параметрів значно впливає на ефективність навчання. Сучасні підходи включають байєсівську оптимізацію, випадковий пошук та еволюційні алгоритми, які систематично досліджують простір гіперпараметрів для знаходження оптимальних конфігурацій, тим самим скорочуючи час та ресурси, потрібні для розробки ефективних моделей [4].

Методи навчання з підкріпленням (reinforcement learning) дозволяють моделям вивчати оптимальні стратегії поведінки через взаємодію з середовищем. Алгоритми політичного градієнта, DQN та TRPO оптимізують параметри моделі для максимізації очікуваної нагороди. Ці підходи суттєво відрізняються від традиційного навчання з учителем, оскільки не вимагають явно розмічених даних, але потребують ефективних механізмів дослідження простору станів та дій. Гібридні підходи, що поєднують навчання з підкріпленням та імітаційне навчання, демонструють перспективні результати у сценаріях з обмеженим зворотним зв'язком.

Нейро-символьні системи представляють новий напрямок досліджень, що поєднує нейронні мережі з символьними методами штучного інтелекту. Ці гібридні підходи дозволяють інтегрувати індуктивне навчання нейронних мереж

з дедуктивними міркуваннями символьних систем. Нейро-символьні методи демонструють потенціал для подолання обмежень чисто нейронних підходів, таких як потреба в великих обсягах навчальних даних та складність включення структурованих знань [5]. Дослідження в цьому напрямку зосереджені на розробці диференційованих логічних операцій та методів для інтеграції нейронних компонентів з символьними механізмами міркувань.

1.2 Дослідження архітектури трансформерів

Трансформери з'явились як революційна архітектура в області глибокого навчання завдяки дослідженням Вашніка і співавторів, представлених у роботі "Attention Is All You Need" у 2017 році. Ця архітектура кардинально змінила підходи до моделювання послідовностей, запропонувавши альтернативу рекурентним та згортковим нейронним мережам. Відмінною рисою трансформерів стало використання механізму уваги як основного компоненту, замість послідовної обробки інформації, характерної для попередніх моделей.

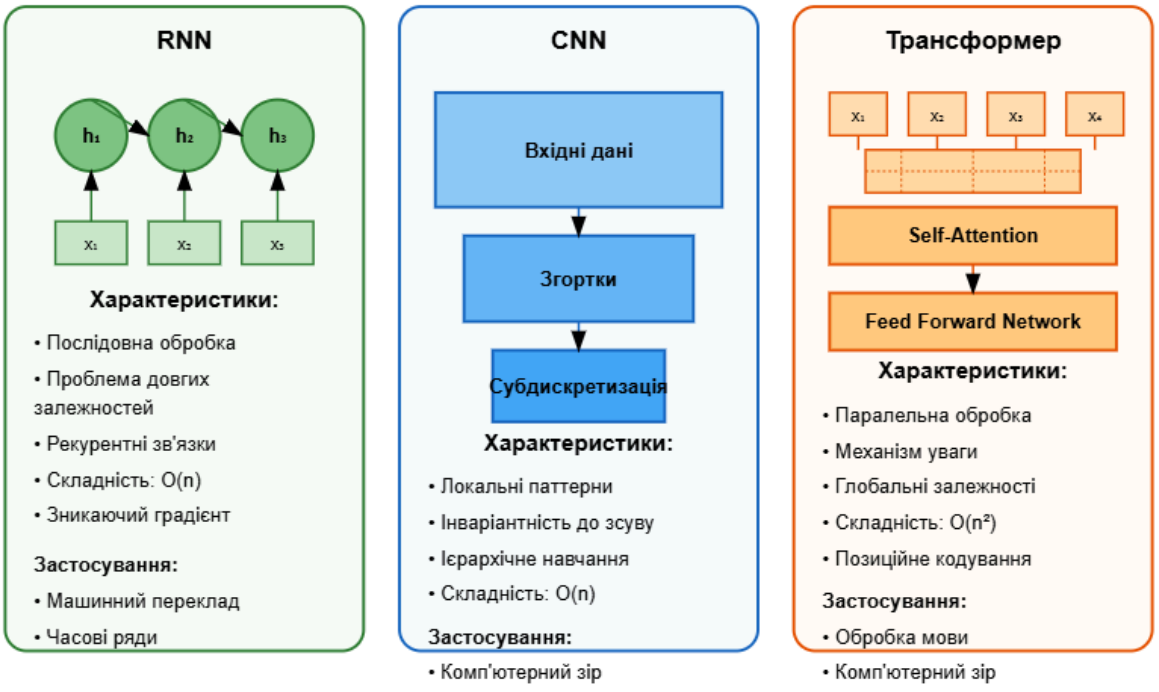

Рис. 1.2 - Порівняння архітектур RNN, CNN та Трансформерів

Базова архітектура трансформера складається з двох основних блоків: енкодера та декодера. Енкодер трансформує вхідну послідовність у представлення з високим рівнем абстракції, тоді як декодер генерує вихідну послідовність на основі цього представлення. Кожен з цих блоків складається з кількох ідентичних шарів, що містять підшари самоуваги та повнозв'язних мереж прямого поширення. Ключовою інновацією цієї архітектури став механізм багатоголової уваги, який дозволяє моделі одночасно зосереджуватись на різних частинах вхідної послідовності [6].

Механізм самоуваги (self-attention) дозволяє трансформерам встановлювати зв'язки між різними позиціями у вхідній послідовності. Для кожного елемента послідовності обчислюються три вектори: запиту (query), ключа (key) та значення (value). Вагові коефіцієнти уваги визначаються як нормалізовані скалярні добутки векторів запиту та ключа, а потім використовуються для зваженого підсумовування векторів значень. Такий підхід дозволяє моделі ефективно вивчати залежності на великих відстанях, що було проблематично для попередніх архітектур.

Багатоголова увага (multi-head attention) розширює концепцію самоуваги, дозволяючи моделі одночасно розглядати інформацію з різних представлень підпросторів. Замість однієї функції уваги, багатоголова увага застосовує кілька паралельних функцій уваги з різними лінійними проекціями вхідних даних. Результати різних "голів" уваги конкатенуються та проектуються в кінцевий простір. Це дозволяє трансформерам вивчати більш складні шаблони та залежності в даних, суттєво підвищуючи виразну потужність моделі.

Позиційне кодування є критичним компонентом архітектури трансформерів, оскільки сам механізм уваги не містить інформації про порядок елементів у послідовності. Для вирішення цієї проблеми до вхідних вкладень додаються позиційні вектори, обчислені за допомогою синусоїдальних функцій різних частот. Альтернативно, позиційне кодування може бути реалізоване як

набір параметрів, що навчаються. Цей компонент надає моделі можливість враховувати відносні або абсолютні позиції елементів у послідовності [7].

Feed-forward мережі у трансформерах представляють собою дві лінійні трансформації з нелінійною активацією між ними, які застосовуються до кожної позиції окремо та ідентично. Ці мережі обробляють виходи з шару самоуваги та підвищують нелінійну потужність моделі. Перша лінійна трансформація зазвичай проектує вхідні дані в простір більшої розмірності, а друга повертає їх до початкової розмірності. Такий підхід дозволяє моделі ефективно вивчати складні функції від представлень, отриманих після механізму уваги.

Залишкові з'єднання (residual connections) та нормалізація шарів (layer normalization) є ключовими компонентами, що забезпечують стабільність навчання глибоких трансформерних архітектур. Залишкові з'єднання додають вхідні дані підшару до його виходу, створюючи шлях для безперешкодного поширення градієнтів через мережу. Нормалізація шарів застосовується після кожного підшару та стабілізує активації, зменшуючи внутрішній зсув коваріат. Ці техніки суттєво прискорюють збіжність під час навчання та дозволяють будувати надзвичайно глибокі моделі.

Метод маскованої самоуваги (masked self-attention) є ключовим для декодера трансформера, який забезпечує авторегресивне генерування послідовностей. При прогнозуванні наступного елемента декодер повинен мати доступ лише до раніше згенерованих елементів. Для досягнення цього, до матриці уваги застосовується маска, яка встановлює вагові коефіцієнти уваги для майбутніх позицій до мінус нескінченності, ефективно запобігаючи "підглядуванню" в майбутнє. Ця техніка дозволяє декодеру навчатися прогнозувати послідовності крок за кроком, зберігаючи при цьому переваги паралельного обчислення під час навчання [8].

Розвиток архітектури трансформерів призвів до появи потужних моделей, таких як BERT (Bidirectional Encoder Representations from Transformers) та GPT (Generative Pre-trained Transformer). BERT використовує лише енкодерну

частину трансформера та навчається передбачати замасковані слова в тексті, враховуючи контекст з обох боків. GPT, навпаки, базується на декодерній частині та навчається автореґресивно передбачати наступне слово в послідовності. Ці моделі демонструють винятково високі результати в широкому спектрі задач обробки природної мови.

Ефективне масштабування трансформерів залишається активною областю досліджень. Зі збільшенням розміру моделі квадратична складність механізму самоуваги стає обмежуючим фактором. Для подолання цього обмеження запропоновано кілька підходів, включаючи розріджену увагу (sparse attention), локальну увагу (local attention) та лінійні трансформери. Ці модифікації прагнуть зменшити обчислювальну складність, зберігаючи при цьому моделюючу потужність оригінальної архітектури.

Архітектура Vision Transformer (ViT) адаптує трансформери для задач комп'ютерного зору, демонструючи, що чисті трансформерні моделі можуть досягати високої точності в класифікації зображень без використання згорткових шарів. ViT розділяє зображення на патчі, які лінійно проектуються в простір вбудувань та додаються до позиційних кодувань. Потім ці вбудування обробляються стандартним трансформерним енкодером. Додатковий токен класифікації використовується для прогнозування класу зображення. Цей підхід показав, що трансформери можуть ефективно моделювати не лише послідовні, але й просторові дані.

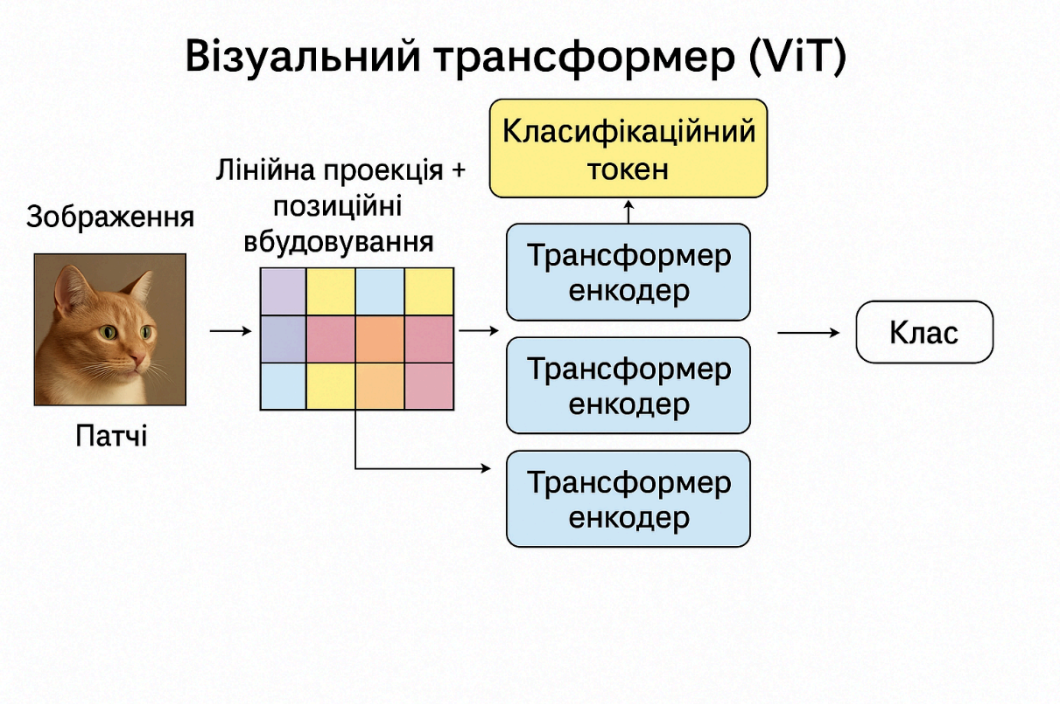

Рис 1.3 - Архітектура Vision Transformer

Трансформерні моделі для мультимодального навчання дозволяють обробляти та інтегрувати інформацію з різних типів даних, таких як текст, зображення та аудіо. Архітектури на зразок CLIP (Contrastive Language-Image Pre-training) використовують окремі трансформерні енкодери для різних модальностей з подальшим проектуванням у спільний семантичний простір. Такі моделі демонструють вражаючі результати в задачах, що вимагають розуміння зв'язків між різними типами даних, наприклад, пошук зображень за текстовим запитом [9].

Трансформери з адаптивною глибиною пропонують механізми для динамічного визначення кількості шарів, необхідних для обробки конкретного вхідного прикладу. Ці підходи дозволяють моделі витрачати більше обчислювальних ресурсів на складні вхідні дані та менше на прості. Прикладами таких архітектур є Universal Transformers та Depth-Adaptive Transformer. Ці моделі можуть потенційно підвищити ефективність обчислень під час інференсу, зберігаючи при цьому високу точність на складних прикладах.

Гібридні архітектури, що поєднують трансформери з іншими типами нейронних мереж, представляють перспективний напрямок досліджень. Наприклад, Convolutional Transformer інтегрує згорткові шари для ефективної обробки локальних паттернів з трансформерними блоками для моделювання глобальних залежностей. Інші гібридні моделі включають комбінації трансформерів з рекурентними мережами, графовими нейронними мережами або нейро-символьними компонентами. Такі архітектури прагнуть поєднати сильні сторони різних підходів для створення більш ефективних та гнучких моделей.

Квантування та дистиляція трансформерів дозволяють суттєво зменшити розмір моделі та прискорити інференс без значної втрати точності. Квантування зменшує точність представлення вагових коефіцієнтів, наприклад, з 32-бітних чисел з плаваючою комою до 8-бітних цілих. Дистиляція знань передає знання від великої "вчительської" моделі до меншої "учнівської" через навчання на м'яких мітках [10]. Ці методи особливо актуальні для розгортання трансформерних моделей на пристроях з обмеженими ресурсами, таких як мобільні телефони та вбудовані системи.

1.3 Огляд існуючих рішень для прискорення навчання нейромереж

Сучасний розвиток систем штучного інтелекту характеризується постійним збільшенням розмірів моделей нейронних мереж, що супроводжується зростанням обчислювальних вимог для їх навчання. Дослідники та інженери розробили численні методи та техніки для подолання цих обчислювальних обмежень. Розглянемо різноманітні підходи, які дозволяють прискорити процес навчання нейронних мереж без втрати якості моделей.

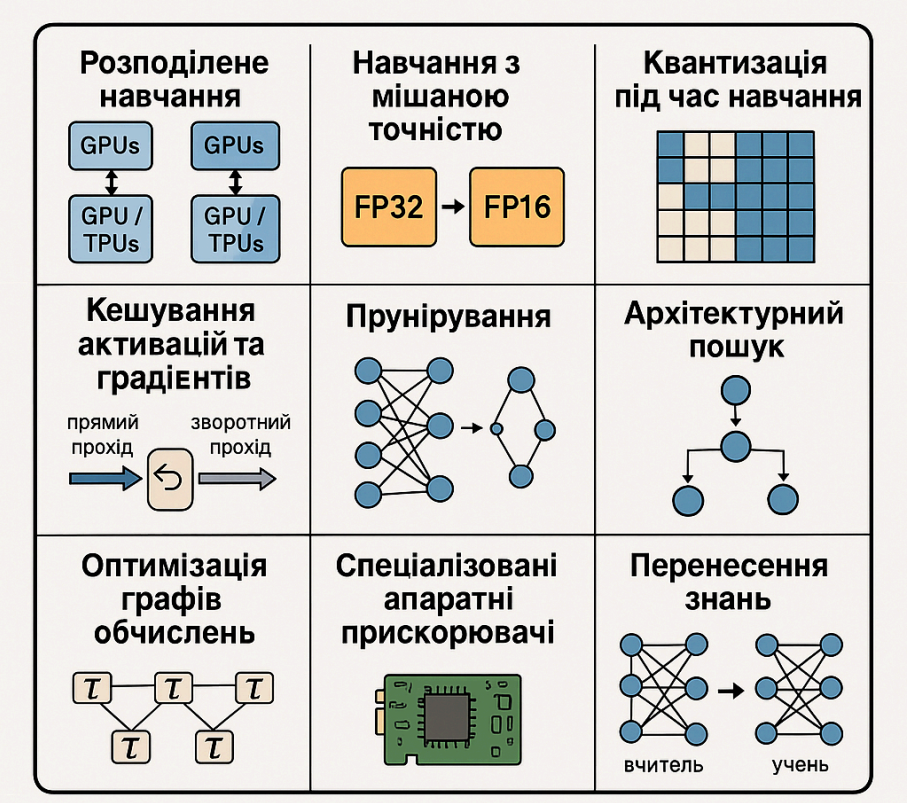

Рис 1.4 - Існуючі рішення для прискорення навчання нейромереж

Розподілене навчання становить один із фундаментальних підходів до прискорення тренування моделей. Ця техніка розподіляє обчислювальне навантаження між кількома пристроями, такими як GPU чи TPU. Існують два основні варіанти реалізації: паралелізм даних та паралелізм моделей. При паралелізмі даних, копії моделі розміщуються на різних пристроях, кожен з яких обробляє різні підмножини навчальних даних. При паралелізмі моделей, сама модель розділяється між пристроями, що дозволяє обробляти більші архітектури. Бібліотеки такі як Horovod, DeepSpeed та PyTorch Distributed забезпечують ефективну реалізацію цих підходів [11].

Навчання з мішаною точністю (mixed precision training) використовує числа з половинною точністю (FP16) замість стандартних чисел з одинарною точністю (FP32) для значного прискорення обчислень. Сучасні GPU, особливо архітектури NVIDIA Volta, Turing та Ampere, містять спеціалізовані тензорні ядра, оптимізовані для операцій з FP16, що може збільшити швидкість обчислень до 2-3 разів. Для підтримки стабільності навчання використовуються

техніки такі як масштабування втрат та головні копії вагових коефіцієнтів у FP32. Бібліотеки NVIDIA Apex та вбудована підтримка в PyTorch та TensorFlow спрощують впровадження цього підходу.

Квантизація нейронних мереж під час навчання пропонує метод зменшення обчислювальних вимог шляхом використання цілочисельних представлень замість чисел з плаваючою комою. Наприклад, методи на основі 8-бітних цілих чисел (INT8) можуть зменшити вимоги до пам'яті та прискорити обчислення з мінімальною втратою точності. Динамічна квантизація, квантизація під час навчання (Quantization Aware Training) та пост-тренувальна квантизація пропонують різні компроміси між обчислювальною ефективністю та точністю моделі. Ці підходи особливо корисні для розгортання моделей на пристроях з обмеженими ресурсами.

Кешування активацій та градієнтів дозволяє уникнути повторних обчислень під час прямого та зворотного проходів через мережу. Градієнтний чекпоїнтинг (gradient checkpointing) представляє компромісний підхід, який зберігає лише обрані активації та повторно обчислює інші під час зворотного проходу. Ця техніка суттєво зменшує вимоги до пам'яті, дозволяючи навчати більші моделі на тому ж обладнанні, хоча й ціною незначного збільшення обчислень. Бібліотеки як PyTorch та TensorFlow надають вбудовану підтримку цих технік для оптимізації використання пам'яті [12].

Прунірування (pruning) нейронних мереж під час навчання передбачає видалення ваг або навіть цілих нейронів, які мінімально впливають на продуктивність мережі. Техніки такі як магнітудне прунірування, лотерейний квиток (lottery ticket hypothesis) та структуроване прунірування дозволяють створювати більш компактні та обчислювально ефективні моделі. Динамічне прунірування під час навчання може адаптивно змінювати структуру мережі, фокусуючи обчислювальні ресурси на найбільш значущих компонентах. Нещодавні дослідження показують, що належним чином прунірована мережа

може досягти точності повної моделі при суттєво нижчих обчислювальних затратах.

Архітектурний пошук (Neural Architecture Search, NAS) автоматизує процес проектування оптимальних архітектур нейронних мереж. Традиційні методи NAS були обчислювально дорогими, але новітні підходи як диференційований архітектурний пошук (DARTS), одношотовий NAS та ефективний NAS (ENAS) значно скоротили необхідні ресурси. Ці методи дозволяють знаходити архітектури, оптимізовані одночасно за точністю та обчислювальною ефективністю. Архітектури, знайдені через NAS, часто навчаються швидше та вимагають менше обчислювальних ресурсів порівняно з ручно спроектованими моделями.

Оптимізація графів обчислень включає різноманітні трансформації, такі як злиття операцій, усунення надлишкових обчислень та оптимізація розкладу виконання. Компілятори глибокого навчання, такі як XLA, TensorRT та TVM, аналізують граф обчислень моделі та генерують оптимізований код для конкретного цільового обладнання [13]. Ці інструменти можуть значно прискорити як навчання, так і висновки, особливо для моделей зі складними архітектурами. Автоматична диференціація з оптимізацією графу, реалізована в сучасних фреймворках, додатково зменшує накладні витрати на обчислення градієнтів.

Специфічні для трансформерів оптимізації спрямовані на подолання квадратичної складності механізму уваги. Лінійні трансформери, такі як Linformer, Performer та Linear Transformer, замінюють операцію увага матрично-векторного множення лінійними проекціями, змінюючи складність з $O(n^2)$ до $O(n)$, де n - довжина послідовності. Ефективні реалізації розрідженої уваги, такі як Big Bird, Longformer та Reformer, фокусуються лише на підмножині взаємодій, зберігаючи при цьому моделюючу здатність. Ці модифікації суттєво прискорюють навчання трансформерів, особливо для довгих послідовностей.

Програмна оптимізація для конкретних апаратних платформ максимально використовує характеристики цільового обладнання. Бібліотеки CUDA для NVIDIA GPU, ROCm для AMD GPU та MKL для Intel CPU містять високооптимізовані реалізації операцій глибокого навчання. Такі фреймворки як ONNX Runtime та TensorRT надають додаткові оптимізації, специфічні для платформи. Інструменти профілювання, такі як NVIDIA Nsight та Intel VTune, допомагають ідентифікувати вузькі місця в продуктивності та налаштовувати код для конкретного обладнання. Ці оптимізації можуть приносити суттєві прискорення без модифікації самої моделі [14].

Спеціалізовані апаратні прискорювачі, такі як TPU (Tensor Processing Units) від Google, NVIDIA A100 з ядрами Tensor та Intel Habana Gaudi, призначені спеціально для прискорення операцій глибокого навчання. Ці пристрої містять спеціалізовані архітектури для ефективного виконання матрично-векторних операцій та інших обчислень, типових для нейронних мереж. Апаратна підтримка операцій з низькою точністю, таких як FP16, BF16 та INT8, додатково підвищує продуктивність. Хмарні провайдери як AWS, GCP та Azure пропонують доступ до цих прискорювачів через свої сервіси.

Низькорангові апроксимації ваг та активацій використовують математичні властивості тензорів для зменшення обчислювальних вимог. Техніки такі як сингулярний розклад (SVD), розклад CP та факторизація Такера розкладають тензори ваг на добуток менших тензорів. Ці методи можуть значно зменшити кількість параметрів та обчислень, особливо для повнозв'язних та згорткових шарів. Низькорангові апроксимації можуть застосовуватися як під час навчання, так і як пост-обробка, пропонуючи різні компроміси між точністю та ефективністю.

Динамічні методи активації обчислень вирішують проблему надмірних обчислень в глибоких нейронних мережах. Моделі з раннім виходом (early-exit models) такі як Multi-Scale Dense Networks (MSDNet) та Shallow-Deep Networks дозволяють завершити висновок на проміжних шарах для простих прикладів.

Умовні обчислення, реалізовані в моделях таких як Mixture of Experts (MoE) та Sparsely-Gated Mixture-of-Experts, активують лише підмножину параметрів для кожного вхідного прикладу. Ці підходи можуть значно прискорити як навчання, так і висновок, адаптуючи обчислювальні ресурси до складності вхідних даних.

Методи адаптивного навчання автоматично налаштовують гіперпараметри оптимізатора під час навчання. Алгоритми такі як Adam, AdamW та Adafactor адаптують швидкість навчання для кожного параметра на основі статистики градієнтів, що прискорює збіжність порівняно з простим стохастичним градієнтним спуском. Планувальники швидкості навчання (learning rate schedulers) як косинусний розпад, циклічні швидкості навчання та OneCycle додатково покращують процес оптимізації. Автоматичний вибір розміру батча (batch size) балансує використання пам'яті та сходження, дозволяючи ефективніше використовувати обчислювальні ресурси.

Перенесення знань та продовження навчання (transfer learning and continuous learning) використовують попередньо навчені моделі як відправну точку для нових завдань. Ініціалізація нової моделі вагами з попередньо навченої моделі значно прискорює збіжність порівняно з випадковою ініціалізацією. Техніки як поступове розморожування (gradual unfreezing), дискримінативна настройка швидкості навчання (discriminative fine-tuning) та прогресивний решейпінг (progressive reshaping) додатково оптимізують цей процес [15]. Безперервне навчання (continuous learning) дозволяє ефективно оновлювати моделі з новими даними без повного перенавчання, що економить обчислювальні ресурси для моделей, які регулярно оновлюються.

1.4 Постановка задачі дослідження

Сучасні нейронні мережі стикаються з проблемою значних обчислювальних витрат та тривалого часу навчання, що ускладнює їх практичне застосування в умовах обмежених ресурсів. Незважаючи на досягнутий прогрес

у галузі глибокого навчання, питання ефективності тренування нейромережевих моделей залишається відкритим. Особливо це стосується складних архітектур, які потребують обробки великих масивів даних протягом багатьох епох навчання. Тому розробка методів прискорення навчання нейронних мереж представляє актуальну науково-практичну задачу.

Аналіз існуючих підходів виявив, що архітектура трансформерів пропонує перспективні механізми для оптимізації процесу навчання різноманітних нейромережевих моделей. Механізм уваги, який лежить в основі трансформерів, дозволяє ефективно встановлювати взаємозв'язки між елементами вхідних даних, потенційно пришвидшуючи збіжність під час навчання. Проте застосування трансформерних компонентів для прискорення тренування традиційних архітектур нейронних мереж досліджено недостатньо.

Метою даного дослідження є розробка та експериментальне дослідження методів прискорення навчання нейронних мереж за допомогою інтеграції трансформерних механізмів. Для досягнення поставленої мети необхідно вирішити комплекс взаємопов'язаних завдань, які охоплюють теоретичні, методологічні та практичні питання застосування трансформерів для оптимізації процесу навчання [16].

Першим завданням дослідження є формалізація процесу навчання нейронних мереж та виявлення етапів, які підлягають оптимізації. Це потребує детального аналізу обчислювальної складності операцій прямого та зворотного проходження, механізмів оновлення вагових коефіцієнтів та структури градієнтних обчислень. Розуміння вузьких місць у процесі навчання дозволить цілеспрямовано застосувати трансформерні механізми для їх подолання.

Другим завданням є розробка модифікованих архітектур нейронних мереж, які інтегрують механізми уваги та інші компоненти трансформерів для прискорення процесу навчання. Це включає дослідження можливостей заміни традиційних шарів трансформерними блоками, розробку гібридних архітектур та адаптацію механізмів уваги для різних типів даних. Особливу увагу слід

приділити оптимізації обчислювальної ефективності запропонованих модифікацій.

Третім завданням є розробка алгоритмічних підходів до використання трансформерів для оптимізації процесу навчання існуючих архітектур без значних структурних змін. Це включає дослідження можливостей застосування механізмів уваги для динамічного регулювання швидкості навчання, селективного оновлення вагових коефіцієнтів та адаптивного формування міні-батчів. Такі підходи дозволять прискорити навчання без необхідності переглядати загальну архітектуру моделі.

Четвертим завданням є розробка програмних засобів, які реалізують запропоновані методи прискорення навчання нейронних мереж. Програмна реалізація повинна забезпечувати зручний інтерфейс для інтеграції з існуючими фреймворками глибокого навчання, таким як TensorFlow та PyTorch. Особливу увагу слід приділити оптимізації коду для ефективного використання наявних обчислювальних ресурсів, включаючи GPU та багатоядерні CPU.

П'ятим завданням є проведення експериментального дослідження ефективності запропонованих методів на стандартних наборах даних та архітектурах нейронних мереж. Експерименти повинні включати порівняльний аналіз швидкості збіжності, часу навчання та досягнутої точності моделей. Необхідно також дослідити залежність ефективності запропонованих методів від характеристик даних, глибини мережі та інших факторів, що впливають на процес навчання.

Шостим завданням є аналіз обчислювальної ефективності запропонованих методів, включаючи оцінку додаткових витрат пам'яті та обчислювальних ресурсів, пов'язаних із використанням трансформерних компонентів. Це дозволить визначити ситуації, коли застосування розроблених методів є доцільним з точки зору загальної ефективності. Особливу увагу слід приділити аналізу масштабованості запропонованих рішень зі збільшенням розміру моделі та обсягу даних.

Сьомим завданням є розробка рекомендацій щодо практичного застосування запропонованих методів для прискорення навчання різних типів нейронних мереж. Рекомендації повинні враховувати характеристики конкретних задач, доступні обчислювальні ресурси та вимоги до точності моделі. Це дозволить дослідникам та практикам ефективно використовувати розроблені методи в своїй роботі.

Теоретичною основою дослідження є концепція трансформерів, теорія оптимізації, методи регуляризації та принципи ефективних обчислень. Методологічна база включає експериментальне моделювання, статистичний аналіз результатів та порівняльне дослідження запропонованих підходів з існуючими методами прискорення навчання нейронних мереж.

Практична значущість роботи полягає в розробці конкретних методів та програмних засобів, які дозволять зменшити час навчання нейронних мереж без суттєвої втрати точності. Це відкриває можливості для більш широкого застосування складних нейромережевих моделей в умовах обмежених обчислювальних ресурсів, а також для прискорення дослідницького циклу розробки та тестування нових архітектур.

Наукова новизна дослідження полягає у розробці нових підходів до прискорення навчання нейронних мереж на основі механізмів трансформерів, формалізації процесу інтеграції компонентів уваги в різні архітектури та встановленні закономірностей впливу трансформерних механізмів на ефективність навчання. Очікується, що результати дослідження зроблять внесок у розвиток теорії та практики глибокого навчання [17].

В рамках цього дослідження передбачається розв'язання фундаментальних проблем, пов'язаних з ефективністю обчислень у глибокому навчанні, та практичних завдань, спрямованих на розробку методів, які можуть бути безпосередньо застосовані для прискорення навчання широкого спектру нейромережевих моделей для різноманітних прикладних задач.

Таким чином, постановка задачі дослідження охоплює широкий спектр теоретичних та практичних питань, пов'язаних з прискоренням навчання нейронних мереж за допомогою трансформерів, та передбачає комплексний підхід до їх розв'язання.

РОЗДІЛ 2. ТЕОРЕТИЧНІ ОСНОВИ ТА МЕТОДОЛОГІЯ ДОСЛІДЖЕННЯ

2.1 Математичні основи навчання нейронних мереж

Процес навчання нейронних мереж ґрунтується на фундаментальних математичних принципах, які забезпечують теоретичне підґрунтя для розуміння та оптимізації цього процесу. В основі сучасних методів навчання лежить концепція градієнтного спуску – ітеративного алгоритму оптимізації, який поступово наближається до мінімуму функції втрат шляхом руху в напрямку, протилежному градієнту. Для нейронних мереж, оновлення параметрів відбувається за формулою:

$$\theta(t\ +\ 1) =\ \theta(t) - \eta\nabla L(\theta(t)),$$

(2.1)

де θ – вектор параметрів моделі

η – швидкість навчання

$\nabla L(\theta)$ – градієнт функції втрат відносно параметрів.

Обчислення градієнта функції втрат для багатошарових нейронних мереж здійснюється через алгоритм зворотного поширення помилки (backpropagation). Цей метод базується на послідовному застосуванні правила ланцюга для обчислення частинних похідних. Якщо представити нейронну мережу як композицію функцій

$$f\ =\ f_L\ \circ\ f_{\{L-1\}}\ \circ\ \ldots\ \circ\ f_1,$$

(2.2)

де L – кількість шарів, то градієнт функції втрат E відносно параметрів j -го шару w_j обчислюється як:

$$\frac{\partial E}{\partial w_j} = \frac{\partial E}{\partial y_L} \cdot \frac{\partial y_L}{\partial y_{\{L-1\}}} \cdot \ldots \cdot \frac{\partial y_{\{j+1\}}}{\partial y_j} \cdot \frac{\partial y_j}{\partial w_j},$$

(2.3)

де y_j – вихід j-го шару.

Стохастичний градієнтний спуск (SGD) представляє собою модифікацію класичного градієнтного спуску, при якій оновлення параметрів відбувається на основі підмножини (міні-батча) навчальних даних, а не всього набору. Математично це можна представити як:

$$\theta(t+1) = \theta(t) - \eta \nabla L_{B(\theta(t))},$$

(2.4)

де L_B – функція втрат, обчислена на міні-батчі B. Такий підхід зменшує обчислювальні витрати та додає стохастичність, яка допомагає уникати локальних мінімумів. Розмір міні-батча є компромісом між точністю оцінки градієнта та швидкістю ітерацій [18].

Адаптивні методи оптимізації, такі як Adam, RMSprop та Adagrad, розширюють базовий SGD, адаптуючи швидкість навчання для кожного параметра на основі статистики попередніх градієнтів. Наприклад, алгоритм Adam обчислює експоненціально зважені середні першого та другого моментів градієнтів:

$$m_t = \beta_1 \cdot m_{\{t-1\}} + \left(1 - \beta_1\right) \cdot \nabla L(\theta(t)),$$

(2.5)

$$v_t = \beta_2 \cdot v_{\{t-1\}} + \left(1 - \beta_2\right) \cdot (\nabla L(\theta(t)))^2,$$

(2.6)

де β_1 та β_2 – гіперпараметри. Оновлення параметрів відбувається за формулою:

$$\theta(t+1) = \theta(t) - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \varepsilon}},$$

(2.7)

де $\hat{m}_t$ та $\hat{v}_t$ – скориговані моменти, а ε – мала константа для уникнення ділення на нуль.

Функції активації відіграють ключову роль у нелінійному перетворенні даних у нейронних мережах. Сигмоїдальна функція $\sigma(x) = \frac{1}{1+e^{-x}}$ та гіперболічний тангенс $\tanh tanh\,(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$ були популярними в ранніх нейромережах, але страждали від проблеми зникаючого градієнта. ReLU (Rectified Linear Unit) $f(x) = (0,\, x)$ та її варіації, такі як Leaky ReLU $f(x) = (\alpha x,\, x)$, де α – малий коефіцієнт, значно пом'якшили цю проблему. Математичний аналіз показує, що похідна ReLU дорівнює 1 для позитивних вхідних значень і 0 для негативних, що спрощує обчислення та прискорює збіжність [19].

Регуляризація відіграє центральну роль у запобіганні перенавчання нейронних мереж. $L1$-регуляризація додає до функції втрат термін $\lambda\sum\left|w_{i}\right|$, а $L2$ -регуляризація – термін $\lambda\sum w_{i}^{2}$, де λ – коефіцієнт регуляризації. Математично $L1$ -регуляризація заохочує розріджені рішення, оскільки її градієнт не залежить від величини параметра, тоді як $L2$-регуляризація пропорційно зменшує всі параметри. Dropout, інший метод регуляризації, можна розглядати як усереднення по ансамблю моделей, де під час навчання кожен нейрон відключається з ймовірністю p, а під час виведення всі активації масштабуються на (1-p).

Пакетна нормалізація (Batch Normalization) стабілізує навчання шляхом нормалізації активацій усередині мережі. Для міні-батча B, активації нормалізуються як:

$$y = \gamma \cdot \frac{x - \mu_{B}}{\sqrt{\sigma_{B}^{2} + \varepsilon}} + \beta, \tag{2.8}$$

де μ_{B} та σ_{B}^{2} – середнє значення та дисперсія активацій у батчі, γ та β – параметри масштабування та зсуву, які навчаються, а ε – мала константа для

числової стабільності. Ця операція пом'якшує проблему внутрішнього зсуву коваріат (internal covariate shift) та дозволяє використовувати вищі швидкості навчання, прискорюючи збіжність.

Архітектура трансформерів ґрунтується на механізмі уваги, який можна формалізувати як зважену суму значень (values), де ваги визначаються сумісністю між запитом (query) та ключем (key). Математично це можна записати як:

$$\text{Attention(Q, K, V)} = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})\text{V}, \tag{2.9}$$

де Q, K, V – матриці запитів, ключів та значень відповідно, d_k – розмірність ключів, а softmax нормалізує ваги уваги. Багатоголова увага розширює цю концепцію, застосовуючи h паралельних функцій уваги з різними лінійними проекціями:

$$\text{MultiHead(Q, K, V)} = \text{Concat}(head_1, \ ..., \ head_h)W^O, \tag{2.10}$$

де $head_i = \text{Attention}(QW_i^Q, \ KW_i^K, \ VW_i^V)$ [20].

Зберігання проміжних активацій для зворотного поширення помилки потребує значних обсягів пам'яті, особливо для глибоких мереж. Техніка градієнтного чекпоїнтингу (gradient checkpointing) зменшує використання пам'яті, зберігаючи лише вибрані активації та повторно обчислюючи інші під час зворотного проходу. Якщо представити пряме проходження як послідовність функцій $y_i = f_{i(y_{\{i-1\}})}$, то звичайний алгоритм зворотного поширення вимагає зберігання всіх y_i. З чекпоїнтингом зберігаються лише вибрані y_j, а проміжні активації повторно обчислюються під час зворотного проходу, що зменшує вимоги до пам'яті з O(N) до O($\sqrt{N}$) ціною додаткових обчислень.

Ініціалізація вагових коефіцієнтів має значний вплив на швидкість збіжності нейронних мереж. Ініціалізація Xavier/Glorot розподіляє початкові ваги як

$$W \sim U(-\sqrt{\frac{6}{n_{in}+n_{out}}}),$$

(2.11)

$$\sqrt{(6/(n\_in \ + \ n\_out)))},$$

(2.12)

де n_{in} та n_{out} – кількість вхідних та вихідних з'єднань. Ініціалізація He, оптимізована для ReLU, використовує $W \sim N\left(0, \sqrt{\frac{2}{n_{in}}}\right)$. Ці методи спрямовані на підтримку дисперсії активацій та градієнтів на однаковому рівні протягом усієї мережі, запобігаючи проблемам зникаючого чи вибухового градієнта.

Відносно відправної точки, параметр швидкості навчання впливає на крок оновлення в напрямку градієнта. Занадто малі значення призводять до повільної збіжності, тоді як занадто великі можуть викликати нестабільність. Планувальники швидкості навчання динамічно адаптують цей параметр під час навчання. Експоненціальний розпад встановлює

$$\eta_t = \ \eta_0 \cdot \gamma^t,$$

(2.13)

де $\gamma \ < \ 1$. Косинусний розпад використовує

$$\eta_t = \ \eta_{+\,0.5} \cdot \left(\eta_{max} - \eta_{min}\right) \cdot \left(1 \ + \cos cos\left(\frac{\pi t}{T}\right)\right),$$

(2.14)

де T – загальна кількість ітерацій. Циклічні планувальники, такі як OneCycle, коливають швидкість навчання між мінімальним та максимальним значеннями за певним розкладом, що допомагає уникати локальних мінімумів.

Методи другого порядку використовують інформацію про кривизну функції втрат, представлену через гессіан $\nabla^2 L(\theta)$. Класичний метод Ньютона оновлює параметри як

$$\theta(t+1) = \theta(t) - \eta \cdot \left(\nabla^2 L(\theta(t))\right)^{-1} \cdot \nabla L(\theta(t)), \tag{2.15}$$

але обчислення та інвертування гессіана для великих моделей неможливе. Квазі-ньютонівські методи, такі як L-BFGS, апроксимують гессіан або його обернену матрицю через послідовні градієнти. Природний градієнтний спуск (Natural Gradient Descent) використовує метрику Fisher Information Matrix для врахування геометрії простору параметрів, але для більшості нейронних мереж потребує апроксимацій, таких як K-FAC [21].

Дистиляція знань математично формалізується як мінімізація дивергенції Кульбака-Лейблера між розподілами ймовірностей "вчительської" та "учнівської" моделей:

$$L_{KD} = T^2 \cdot KL\left(\sigma\left(\frac{z_t}{T}\right) || \sigma\left(\frac{z_s}{T}\right)\right), \tag{2.16}$$

де z_t та z_s – логіти вчительської та учнівської моделей відповідно, σ – функція softmax, а T – температурний параметр, який контролює згладжування розподілів. При $T \to 0$ дистиляція наближається до навчання на жорстких мітках, тоді як при $T \to \infty$ усі класи стають рівноймовірними. Оптимальне значення T, яке максимізує передачу знань, залежить від конкретної задачі та архітектури моделей.

Для отримання оцінки загальної продуктивності навчання нейронних мереж використовується аналіз обчислювальної складності. Для повнозв'язного шару з n вхідними нейронами та m вихідними, складність прямого та зворотного проходу становить O(nm). Згорткові шари з фільтром розміру $k \times k$, c_{in} вхідними та c_{out} вихідними каналами, та просторовими розмірами $h \times w$

мають складність $O\left(k^2 \cdot c_{in} \cdot c_{out} \cdot h \cdot w\right)$. Для механізму самоуваги в трансформерах, складність складає $O(n^2 \cdot d)$, де n – довжина послідовності, а d – розмірність прихованого стану. Ці теоретичні оцінки дозволяють аналізувати та оптимізувати обчислювальні вимоги різних архітектур нейронних мереж [22].

2.2 Архітектура та принципи роботи трансформерів

Трансформери представляють собою клас моделей глибокого навчання, які революціонізували підходи до обробки послідовних даних, запропонувавши механізм, що повністю базується на увазі (attention). Вперше представлена у 2017 році дослідниками Google у статті "Attention Is All You Need", ця архітектура продемонструвала переваги над рекурентними нейронними мережами (RNN) та згортковими нейронними мережами (CNN) у задачах обробки природної мови, а згодом і в інших доменах. На рис. 2.1 представлена діаграма класів, що ілюструє компонентну структуру трансформерної архітектури. Основними складовими є EncoderStack та DecoderStack, які складаються з відповідних шарів, що містять механізми самоуваги та повнозв'язні мережі.

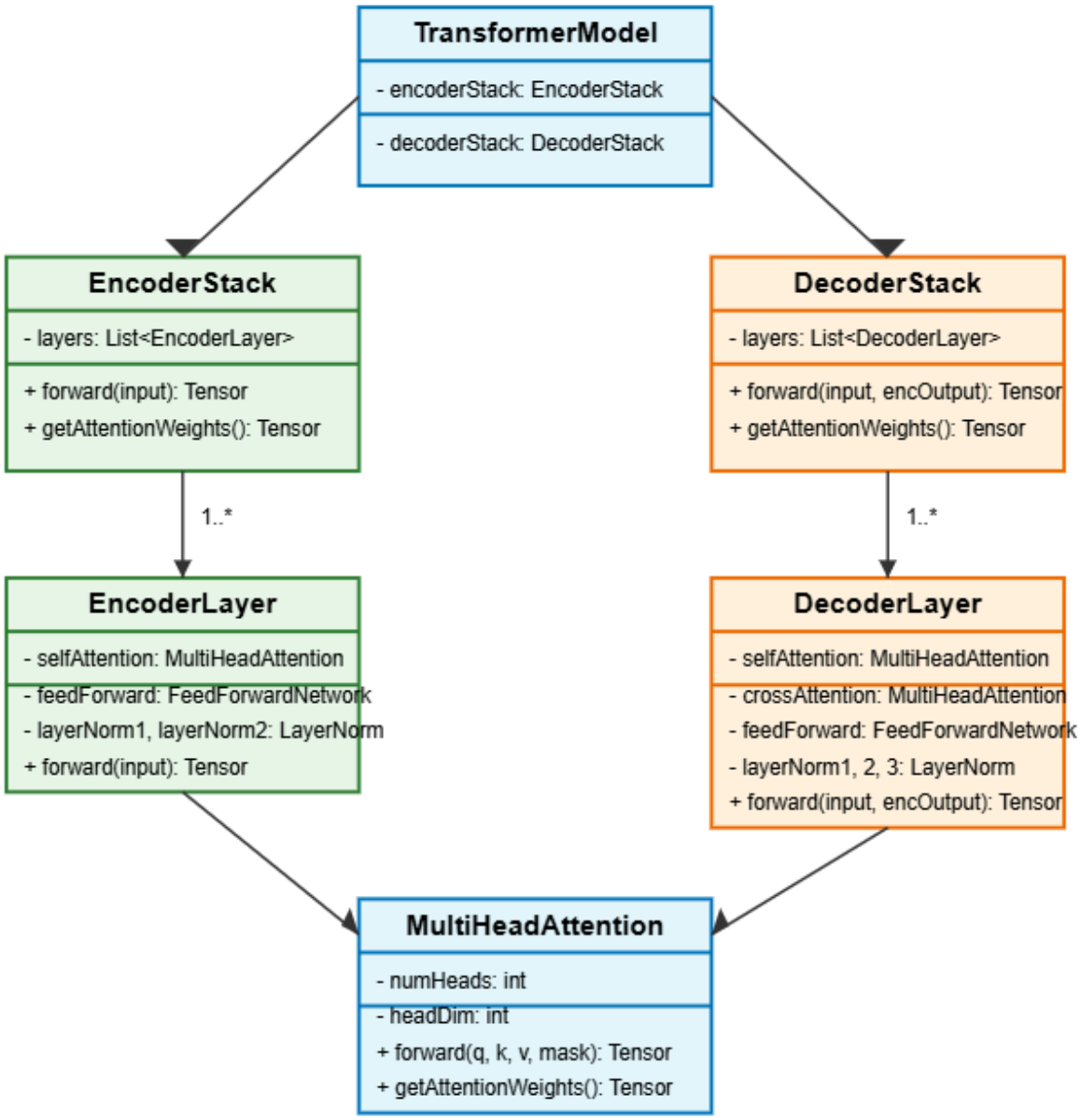

Рис. 2.1 - Діаграма класів архітектури трансформера

Основним структурним елементом трансформера є блок енкодера та декодера. Енкодер складається з декількох ідентичних шарів, кожен з яких містить два підшари: механізм самоуваги (self-attention) та повнозв'язну нейронну мережу прямого поширення (feed-forward neural network). Декодер також містить ці два підшари, але додатково включає третій підшар, який виконує багатоголову увагу над виходом енкодера. Таке розділення функціоналу

дозволяє обробляти вхідні дані та генерувати вихідні послідовності в рамках єдиної архітектури.

Механізм самоуваги лежить в основі здатності трансформерів ефективно моделювати залежності між різними позиціями вхідної послідовності. На відміну від RNN, які обробляють послідовність елемент за елементом, самоувага обчислює відносини між усіма елементами одночасно. Для кожного елемента послідовності обчислюються три вектори: запиту (query), ключа (key) та значення (value). Ваги уваги визначаються як нормалізовані скалярні добутки між векторами запиту та ключа, а потім застосовуються до векторів значень [23]. Рис. 2.2 демонструє послідовність обробки даних у трансформерній архітектурі від входу до виходу. Процес включає токенізацію вхідних даних, обчислення механізму самоуваги в енкодері, кодування вхідних ембедингів та декодування з використанням маскованої та крос-уваги для формування вихідної послідовності.

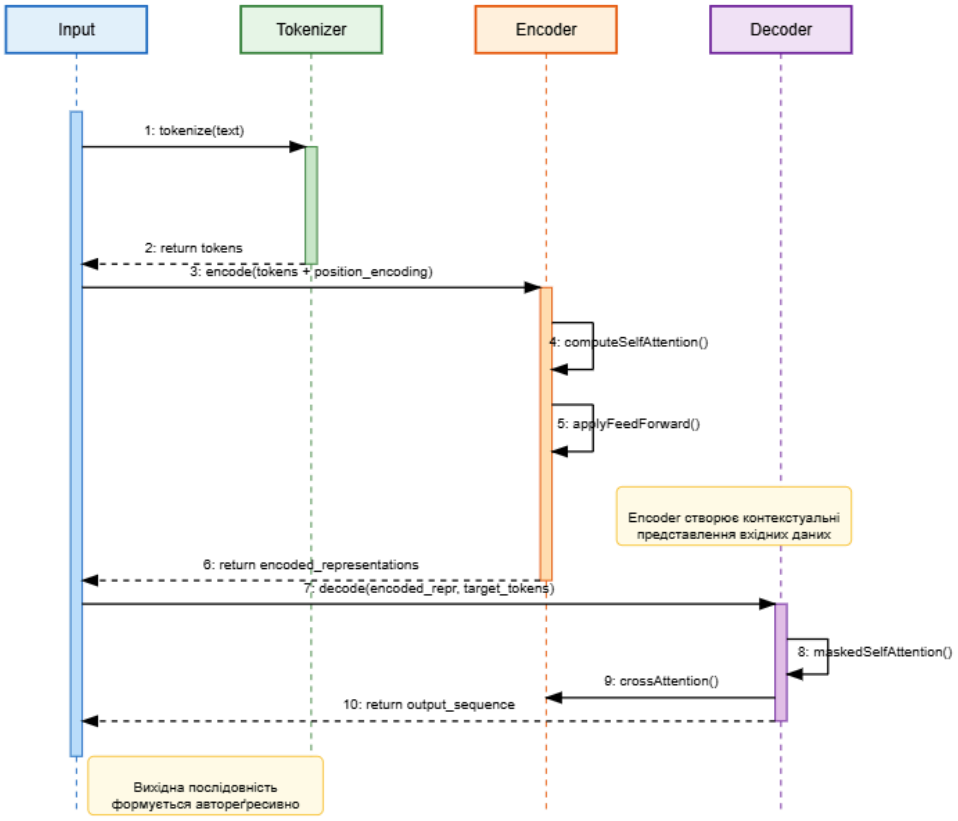

Рис. 2.2 - Діаграма послідовності роботи трансформера

Математично, механізм самоуваги можна описати формулою:

$$\text{Attention(Q, K, V)} = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})\text{V}, \quad (2.17)$$

де Q, K, V — матриці запитів, ключів та значень, отримані лінійним перетворенням вхідної послідовності X: $Q = XW^Q$, $K = XW^K$, $V = XW^V$. Ділення на $\sqrt{d_k}$, де d_k — розмірність векторів ключів, запобігає насиченню функції softmax при великих значеннях скалярного добутку, що стабілізує градієнти під час навчання.

Багатоголова увага (multi-head attention) розширює можливості стандартного механізму уваги, дозволяючи моделі одночасно зосереджуватися на інформації з різних представлень підпростору. Замість єдиної функції уваги, багатоголова увага застосовує h паралельних функцій уваги з різними лінійними проекціями даних. Результати різних "голів" конкатенуються та проектуються в кінцевий вихідний простір. Це дозволяє моделі ефективно витягувати різні типи залежностей з даних, підвищуючи її виразну потужність.

Повнозв'язна мережа прямого поширення у кожному шарі трансформера складається з двох лінійних перетворень з нелінійною активацією ReLU між ними: $FFN(x) = \left(0,\ xW_1 + b_1\right) W_2 + b_2$. Цей компонент обробляє кожну позицію послідовності окремо та ідентично, застосовуючи однакові ваги для всіх елементів. Перша проекція зазвичай збільшує розмірність даних, а друга повертає її до початкової. Така структура дозволяє моделі вивчати складні нелінійні перетворення для кожної позиції після того, як механізм уваги врахував контекстуальні залежності [24].

Залишкові з'єднання (residual connections) та нормалізація шарів (layer normalization) є ключовими компонентами, що забезпечують стабільність навчання глибоких трансформерних архітектур. Залишкові з'єднання додають

вхідні дані підшару до його виходу: x + Sublayer(x), що створює прямий шлях для градієнтів через мережу. Нормалізація шарів стабілізує розподіл активацій, нормалізуючи їх для кожного прикладу окремо:

$$LN(x) = \gamma \cdot \frac{x - \mu}{\sigma} + \beta,$$

(2.18)

де μ та σ — середнє значення та стандартне відхилення активацій, а γ та β — параметри масштабування та зсуву, що навчаються.

Позиційне кодування вирішує фундаментальне обмеження механізму уваги — відсутність інформації про порядок елементів у послідовності. На відміну від RNN, де ця інформація природно присутня через послідовну обробку, трансформери потребують явного позиційного кодування. В оригінальній архітектурі використовуються синусоїдальні функції різної частоти:

$$PE_{pos,2i} = \sin sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \text{ та } PE_{pos,2i+1} = \cos cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right),$$

(2.19)

де pos — позиція в послідовності, а i — вимір. Такий підхід дозволяє моделі екстраполювати на довші послідовності, ніж ті, що були представлені під час навчання.

Процес навчання трансформерів зазвичай використовує стандартні методи оптимізації, такі як Adam, з додатковими технічними особливостями. Застосовується планувальник швидкості навчання з періодом прогріву, що поступово збільшує швидкість навчання протягом перших кількох тисяч кроків: $lr = d_{model}^{-0.5} \cdot \left(step_{num}^{-0.5}, step_{num} \cdot warmup_{steps}^{-1.5}\right)$. Також широко використовується регуляризація у вигляді dropout, що застосовується до виходів кожного підшару перед залишковим з'єднанням, до уваги та до повнозв'язних мереж.

Маскована самоувага (masked self-attention) є ключовим компонентом декодера, що забезпечує автореґресивне генерування послідовностей. Під час

навчання декодер має доступ до всієї цільової послідовності, але для запобігання "підглядуванню" в майбутнє застосовується трикутна маска, яка встановлює вагові коефіцієнти уваги для майбутніх позицій до мінус нескінченності перед застосуванням softmax. Це дозволяє моделі навчатися прогнозувати наступний елемент на основі лише попередніх елементів, зберігаючи при цьому паралельні обчислення під час навчання.

Архітектура BERT (Bidirectional Encoder Representations from Transformers) представляє собою модифікацію трансформера, яка використовує лише енкодерну частину. Основна інновація BERT полягає в підході до навчання — замість прогнозування наступного слова, модель навчається передбачати випадково замасковані слова в тексті (Masked Language Modeling) та визначати, чи є два речення послідовними (Next Sentence Prediction). Такий підхід дозволяє моделі ефективно використовувати двонаправлений контекст і досягати високих результатів у різноманітних задачах обробки природної мови після дотренування на конкретній задачі.

GPT (Generative Pre-trained Transformer), на відміну від BERT, базується на декодерній частині трансформера та навчається авторегресивно прогнозувати наступне слово в послідовності. Цей підхід обмежує модель використанням лише лівого контексту, але спрощує генерування тексту. Модифікації GPT, такі як GPT-2 та GPT-3, зосереджуються на масштабуванні моделі та обсягу даних для навчання, замість архітектурних інновацій. GPT-3, з 175 мільярдами параметрів, продемонструвала вражаючі здібності до вирішення задач без додаткового навчання, лише на основі кількох прикладів у контексті запиту [25].

T5 (Text-to-Text Transfer Transformer) уніфікує різні задачі обробки природної мови в єдиний формат "текст-до-тексту", де вхідні дані містять префікс, що вказує на тип задачі. Модель використовує як енкодер, так і декодер, і навчається перетворювати довільний текстовий вхід на відповідний текстовий вихід. Такий підхід дозволяє застосовувати єдину модель для

різноманітних задач — від машинного перекладу до відповідей на питання та класифікації текстів.

Vision Transformer (ViT) адаптує архітектуру трансформерів для задач комп'ютерного зору, розділяючи зображення на фіксовані патчі, які розглядаються як елементи послідовності. Кожен патч лінійно проектується у простір вбудувань, додається до позиційного кодування та подається на вхід стандартного трансформерного енкодера. Додатковий токен класифікації [CLS] використовується для прогнозування класу зображення. ViT продемонстрував, що чисті трансформерні моделі можуть досягати або перевершувати продуктивність конволюційних нейронних мереж на ключових задачах класифікації зображень при достатньому обсязі даних для попереднього навчання.

Ефективність трансформерів для моделювання послідовних даних і можливість їх адаптації для різних доменів зробили цю архітектуру фундаментальною для сучасного глибокого навчання. Механізм уваги, який дозволяє моделі вибірково зосереджуватися на релевантних частинах вхідних даних, забезпечив проривні результати в різноманітних задачах — від обробки природної мови до комп'ютерного зору та генерування музики [26]. Подальші дослідження зосереджені на подоланні обмежень оригінальної архітектури, таких як квадратична складність механізму уваги та високі вимоги до обчислювальних ресурсів для тренування моделей з мільярдами параметрів.

2.3 Методи оптимізації та прискорення навчання

Прискорення процесу навчання нейронних мереж становить одну з ключових задач сучасного машинного навчання, оскільки зростаюча складність моделей та обсяги даних потребують усе більших обчислювальних ресурсів. Ефективні методи оптимізації дозволяють значно скоротити час тренування моделей без втрати їх точності. Розглянемо основні підходи до оптимізації та

прискорення процесу навчання нейронних мереж. На рис. 2.3 представлена ієрархія класів методів оптимізації, демонструючи взаємозв'язок між базовим абстрактним оптимізатором та його реалізаціями, включаючи запропонований трансформерний оптимізатор, який використовує механізми уваги для адаптивної оптимізації параметрів.

Рис. 2.3 - Діаграма класів методів оптимізації навчання

Стохастичний градієнтний спуск (SGD) та його модифікації залишаються фундаментальним підходом до оптимізації нейронних мереж. Класичний SGD оновлює параметри моделі на основі градієнта, обчисленого для міні-батча

даних. Модифікації, такі як SGD з моментом, вводять додаткову компоненту, яка враховує напрямок попередніх градієнтів, що дозволяє прискорити збіжність та подолати локальні мінімуми. Нестеров Accelerated Gradient (NAG) удосконалює цей підхід, обчислюючи градієнт у точці, до якої приведе поточний момент, що забезпечує кращу корекцію траєкторії оптимізації.

Адаптивні методи оптимізації, такі як AdaGrad, RMSProp та Adam, динамічно коригують швидкість навчання для кожного параметра на основі історії його градієнтів. AdaGrad накопичує квадрати градієнтів та ділить швидкість навчання на квадратний корінь з цієї суми, що призводить до ефективного сповільнення навчання для параметрів з великими градієнтами. RMSProp модифікує цей підхід, використовуючи експоненціально зважене середнє квадратів градієнтів, що дозволяє адаптуватися до нестаціонарних цільових функцій [27]. Adam поєднує ідеї моменту та RMSProp, використовуючи перший та другий моменти градієнтів з корекцією зміщення, що робить його одним з найефективніших методів оптимізації для більшості задач глибокого навчання. Рис. 2.4 ілюструє робочий процес навчання нейронної мережі з використанням запропонованих методів прискорення. Ключовими компонентами є адаптивне формування міні-батчів, градієнтний трансформер для оптимізації градієнтів та адаптивний планувальник швидкості навчання, які інтегруються в стандартний цикл навчання.

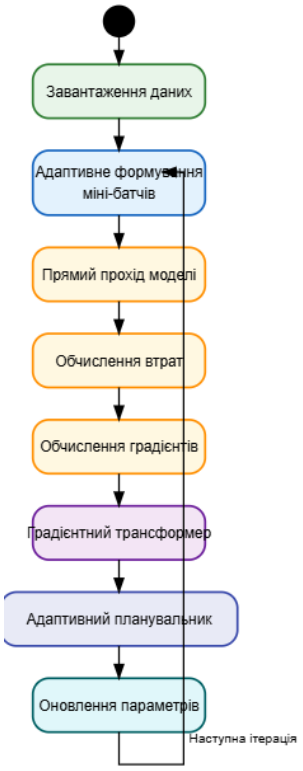

Рис. 2.4 - Діаграма діяльності процесу оптимізації навчання

Планувальники швидкості навчання (learning rate schedulers) забезпечують систематичне коригування швидкості навчання протягом тренування. Ступінчастий розпад (step decay) зменшує швидкість навчання на фіксований коефіцієнт через певні інтервали. Експоненціальний розпад (exponential decay) поступово зменшує швидкість навчання за експоненціальним законом. Косинусний розпад (cosine annealing) змінює швидкість навчання за косинусоїдальним законом, що забезпечує плавне зменшення швидкості навчання. Циклічні швидкості навчання (CLR) та OneCycle підходи періодично варіюють швидкість навчання між мінімальним та максимальним значеннями, що дозволяє ефективніше досліджувати простір параметрів та уникати локальних мінімумів.

Методи пакетної оптимізації фокусуються на ефективному використанні обчислювальних ресурсів через оптимальний вибір розміру батча. Великі батчі дозволяють краще розпаралелювати обчислення та зменшити шум градієнта,

але можуть погіршувати узагальнення моделі. Градієнтне накопичення (gradient accumulation) дозволяє симулювати більші батчі шляхом накопичення градієнтів від кількох менших батчів перед оновленням параметрів [28]. Методи, такі як LARS (Layer-wise Adaptive Rate Scaling) та LAMB (Layer-wise Adaptive Moments Based optimizer), динамічно адаптують швидкість навчання для кожного шару залежно від норми його параметрів та градієнтів, що дозволяє стабільно навчати моделі з великими батчами.

Техніки оптимізації пам'яті дозволяють тренувати більші моделі з обмеженими ресурсами. Градієнтний чекпоїнтинг (gradient checkpointing) зменшує використання пам'яті під час зворотного поширення помилки, зберігаючи лише вибрані активації та повторно обчислюючи інші під час зворотного проходу. Активаційне переобчислення (activation recomputation) повністю усуває потребу в зберіганні проміжних активацій ціною додаткових обчислень. Змішана точність (mixed precision training) використовує числа з половинною точністю (FP16) для більшості обчислень, зберігаючи стабільність через головні копії параметрів у повній точності (FP32) та масштабування втрат для запобігання числовому переповненню.

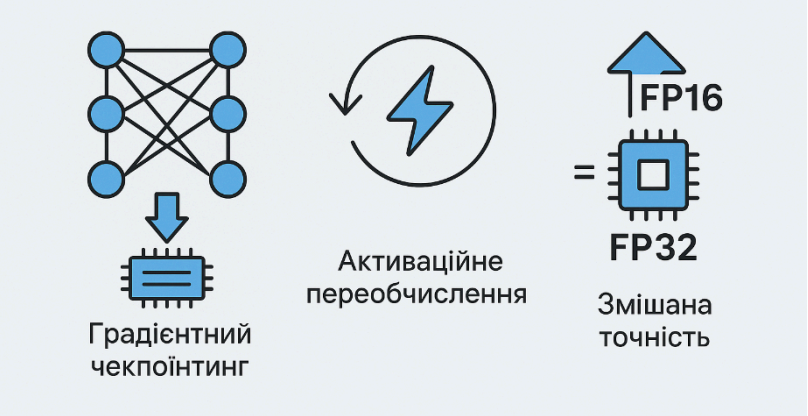

Рис.2.5 – Техніки оптимізації пам’яті

Архітектурна оптимізація включає модифікації структури нейронної мережі для прискорення збіжності. Нормалізація шарів (layer normalization, batch normalization, group normalization) стабілізує розподіл активацій, що дозволяє використовувати вищі швидкості навчання. Залишкові з'єднання

(residual connections) створюють прямі шляхи для градієнтів через мережу, зменшуючи проблему зникаючого градієнта. Техніки, такі як EfficientNet та MobileNet, пропонують оптимальні архітектури з балансом між точністю та обчислювальною ефективністю. Архітектурний пошук (Neural Architecture Search, NAS) автоматизує процес проектування оптимальних архітектур для конкретних задач та обчислювальних обмежень.

Рис.2.6 – Модифікації структури нейронної мережі

Методи регуляризації не лише запобігають перенавчанню, але й можуть прискорювати збіжність. Dropout випадковим чином вимикає нейрони під час навчання, що змушує мережу вивчати більш робастні ознаки. Label smoothing перетворює жорсткі цільові мітки на м'які розподіли, що запобігає надмірній впевненості моделі та покращує узагальнення. Рання зупинка (early stopping) припиняє навчання, коли продуктивність на валідаційному наборі перестає покращуватися, що зберігає обчислювальні ресурси. Zoneout, аналог dropout для рекурентних мереж, зберігає попередні стани нейронів замість встановлення їх в нуль, що стабілізує навчання рекурентних архітектур [29].

Паралельні та розподілені стратегії навчання використовують кілька обчислювальних пристроїв для прискорення тренування. Паралелізм даних (data parallelism) розподіляє різні підмножини навчальних даних між пристроями, кожен з яких має повну копію моделі. Паралелізм моделей (model parallelism) розподіляє різні частини моделі між пристроями, що дозволяє

тренувати більші архітектури. Конвеєрний паралелізм (pipeline parallelism) комбінує ці підходи, розділяючи мережу на етапи та пропускаючи різні міні-батчі через різні етапи одночасно. ZeRO (Zero Redundancy Optimizer) оптимізує використання пам'яті в розподіленому навчанні, усуваючи надлишковість зберігання оптимізатора, градієнтів та параметрів.

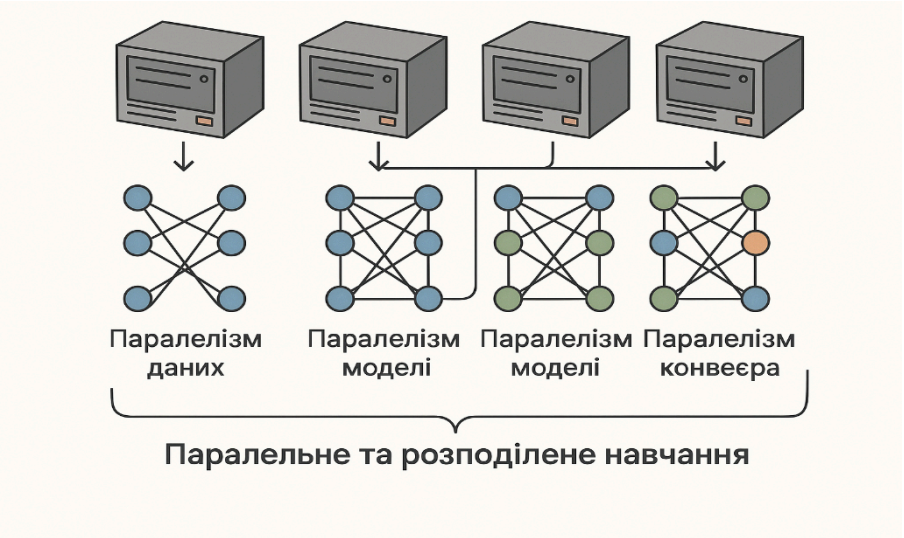

Рис.2.7 – Паралельні та розподілені стратегії навчання

Квантизація під час навчання зменшує обчислювальні вимоги шляхом використання представлень з низькою точністю. Post-training quantization конвертує попередньо навчену модель у версію з низькою точністю. Quantization-aware training включає симуляцію ефектів квантизації під час навчання, що дозволяє моделі адаптуватися до обмеженої точності. Динамічна квантизація застосовує квантизацію лише під час виведення, зберігаючи ваги у повній точності. Техніки, такі як Binary Neural Networks та Quantized Neural Networks, використовують екстремально обмежені представлення (наприклад, бінарні або тернарні ваги), що суттєво прискорює обчислення за рахунок заміни множень на простіші операції.

Рис.2.8 – Стратегії квантизіції

Дистиляція знань дозволяє навчати компактні "студентські" моделі, які наслідують поведінку більших "вчительських" моделей. Процес включає навчання студентської моделі на м'яких мітках, отриманих від учительської моделі, що містять більше інформації, ніж жорсткі класи. Різновиди цього підходу включають навчання на проміжних представленнях (hint training), дистиляцію уваги (attention transfer) та релаційну дистиляцію знань (relational knowledge distillation). Ці методи дозволяють значно зменшити обчислювальні вимоги під час виведення без суттєвої втрати точності, а також можуть прискорювати збіжність студентської моделі [30].

Методи метанавчання ("навчання навчатися") фокусуються на оптимізації самого процесу оптимізації. Підходи, такі як MAML (Model-Agnostic Meta-Learning), навчають моделі швидко адаптуватися до нових задач за кілька кроків градієнтного спуску. Learned optimizers замінюють стандартні алгоритми оптимізації на параметричні функції, які самі навчаються через метанавчання. Meta-SGD вивчає оптимальні початкові ваги та швидкості навчання для кожного параметра. Ці підходи можуть значно прискорити процес адаптації моделей до нових доменів та задач, що особливо цінно в умовах обмежених даних та обчислювальних ресурсів.

Ефективні реалізації операцій глибокого навчання на рівні програмного забезпечення можуть суттєво прискорити навчання без зміни алгоритмів чи архітектур. Компілятори глибокого навчання, такі як XLA, TVM та MLIR, автоматично оптимізують обчислювальні графи моделей для конкретних апаратних платформ. Бібліотеки високооптимізованих операцій, такі як cuDNN для NVIDIA GPU та DNNL (oneAPI Deep Neural Network Library) для CPU та інших прискорювачів, забезпечують максимальну продуктивність базових операцій [31]. Техніки, такі як автоматичне розгортання графів (JIT compilation) та злиття операцій (operator fusion), додатково зменшують накладні витрати та максимізують використання обчислювальних ресурсів.

Специфічні для трансформерів оптимізації спрямовані на подолання квадратичної складності механізму уваги, що стає критичним обмеженням для довгих послідовностей. Лінійні трансформери, такі як Linformer, Performer та Linear Transformer, переформулюють операцію уваги через ядерні апроксимації, змінюючи складність з $O(n^2)$ до $O(n)$. Розріджені техніки, такі як Big Bird, Longformer та Reformer, зосереджуються лише на підмножині взаємодій, ефективно моделюючи далекі залежності. Прогресивне стискання (progressive compression) зменшує розмірність проміжних представлень вглиб мережі. Blended attention поєднує локальну та глобальну увагу для балансу між ефективністю та моделюючою здатністю. Ці модифікації дозволяють тренувати трансформери для значно довших послідовностей без надмірних обчислювальних витрат.

Застосування цих методів оптимізації та прискорення навчання не є взаємовиключним — найефективніші рішення часто поєднують кілька підходів. Вибір конкретних технік залежить від характеристик задачі, архітектури моделі, доступних обчислювальних ресурсів та вимог до точності [32]. Постійний розвиток цієї галузі спрямований на подолання обчислювальних обмежень, що дозволяє тренувати все більш складні моделі на зростаючих обсягах даних, розширюючи межі можливостей штучного інтелекту.

2.4 Розробка модифікованого методу навчання з використанням трансформерів

Модифікований метод навчання нейронних мереж із застосуванням механізмів трансформерів представляє собою комплексний підхід до прискорення збіжності та підвищення ефективності процесу оптимізації. В основі запропонованого методу лежить принцип адаптивної уваги, який дозволяє динамічно перерозподіляти обчислювальні ресурси в залежності від структури вхідних даних та поточного стану моделі. Ключова ідея полягає в інтеграції механізму самоуваги трансформерів у стандартний процес навчання для покращення його характеристик.

Першим компонентом розробленого методу є адаптивний планувальник швидкості навчання на основі уваги. Традиційні планувальники швидкості навчання, такі як експоненціальний або косинусний розпад, застосовують однакову швидкість до всіх параметрів моделі без урахування їх відносної значущості. Запропонований підхід використовує механізм багатоголової уваги для оцінки впливу кожного параметра на загальну продуктивність моделі. Швидкість навчання для параметрів з високими ваговими коефіцієнтами уваги збільшується, тоді як для параметрів з низькими ваговими коефіцієнтами – зменшується [33]. Це дозволяє зосередити обчислювальні ресурси на найбільш критичних компонентах моделі.

Другим ключовим елементом є динамічне формування міні-батчів з використанням механізму крос-уваги. На відміну від стандартного підходу, де зразки для міні-батча вибираються випадковим чином або послідовно, запропонований метод оцінює складність та інформативність кожного навчального прикладу. Зразки з високою складністю або ті, що містять корисну інформацію для поточного стану моделі, отримують вищі вагові коефіцієнти

при формуванні батчів. Такий підхід забезпечує більш ефективне використання навчальних даних та прискорює збіжність, особливо на ранніх етапах навчання.

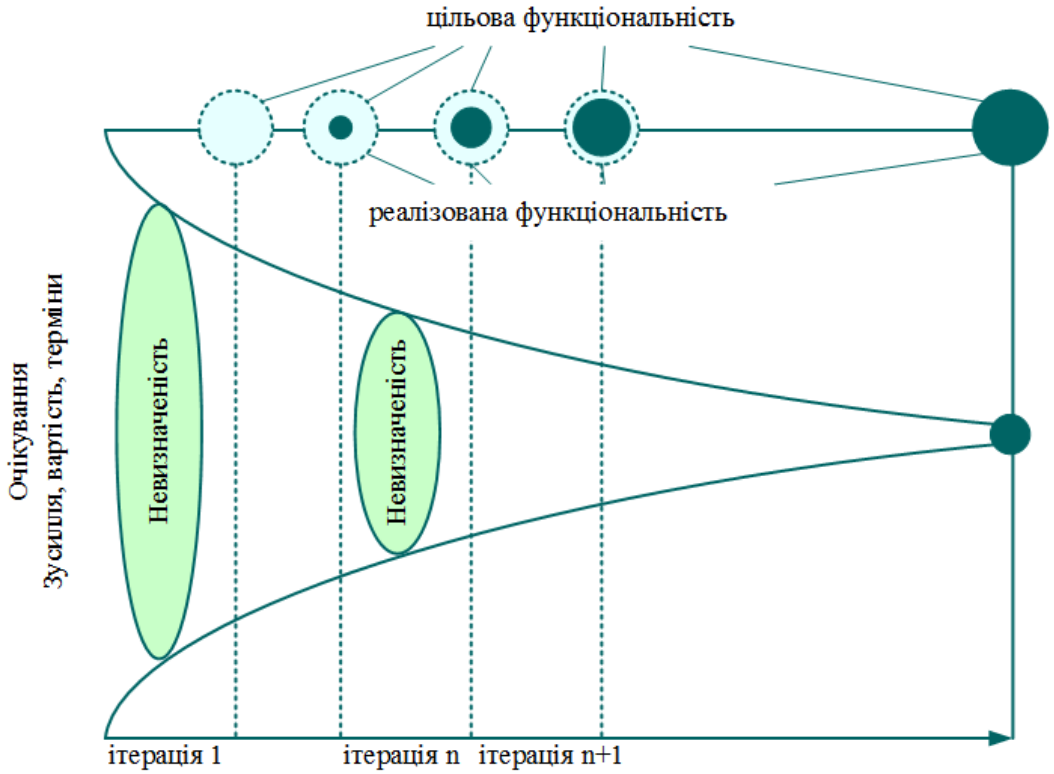

Рис.2.9 – Зразки міні-батчів з використанням крос-уваги

Механізм мета-уваги являє собою третій компонент розробленого методу. Він спрямований на динамічну адаптацію гіперпараметрів оптимізатора в процесі навчання. Мета-увага обчислює відносну значущість різних статистик градієнтів (таких як перший та другий моменти в Adam) та відповідно коригує їх вплив на оновлення параметрів. Це дозволяє оптимізатору адаптуватися до змін ландшафту функції втрат протягом навчання, прискорюючи збіжність та покращуючи стабільність процесу оптимізації.

Четвертий компонент – градієнтний трансформер, який модифікує обчислені градієнти перед їх застосуванням для оновлення параметрів. Традиційні методи оптимізації, такі як SGD з моментом або Adam, застосовують відносно прості трансформації до градієнтів. Запропонований підхід використовує повноцінний трансформерний блок для обробки градієнтів, враховуючи їх взаємозалежності та історію попередніх оновлень [34]. Це

дозволяє уникнути осциляцій та ефективніше долати сідлові точки та локальні мінімуми функції втрат.

П'ятим елементом є механізм чекпоїнтингу на основі уваги, який оптимізує використання пам'яті під час зворотного поширення помилки. Стандартний градієнтний чекпоїнтинг зберігає активації на рівномірно розподілених шарах мережі. Запропонований підхід динамічно визначає, які активації зберігати, базуючись на їх значущості для обчислення градієнтів. Це дозволяє досягти оптимального балансу між використанням пам'яті та додатковими обчисленнями, необхідними для відновлення незбережених активацій.

Шостий компонент – архітектурна адаптація під час навчання з використанням механізму уваги. На відміну від стандартних підходів, де архітектура мережі фіксується перед початком навчання, запропонований метод дозволяє динамічно модифікувати структуру моделі. Механізм самоуваги використовується для оцінки значущості різних компонентів мережі, таких як нейрони, з'єднання або цілі шари. Компоненти з низькими ваговими коефіцієнтами уваги можуть бути тимчасово або постійно виключені з обчислень, що зменшує обчислювальні витрати без суттєвої втрати точності.

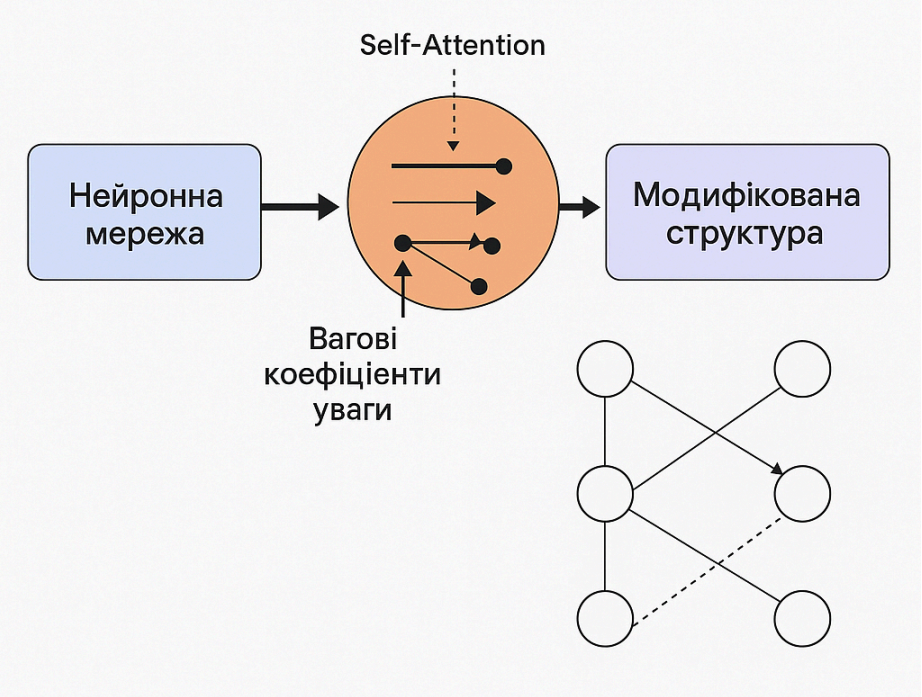

Рис.2.10 – Архітектурна адаптація під час навчання

Сьомим елементом є трансформерна регуляризація, яка застосовує механізм уваги для визначення оптимального рівня регуляризації для різних частин моделі. Традиційні методи регуляризації, такі як L1/L2 регуляризація або dropout, застосовують однакові гіперпараметри до всієї мережі. Запропонований підхід динамічно адаптує силу регуляризації для кожного компонента, базуючись на його поточній значущості та стабільності [35]. Це дозволяє запобігти перенавчанню без надмірного обмеження виразної потужності моделі.

Восьмий компонент – механізм самокорекції на основі уваги, який аналізує патерни помилок моделі на валідаційних даних. Традиційні підходи часто обмежуються простими метриками, такими як загальна точність або функція втрат. Запропонований метод використовує механізм самоуваги для виявлення складних взаємозв'язків між помилками та характеристиками вхідних даних або параметрами моделі. Це дозволяє цілеспрямовано коригувати процес навчання для подолання систематичних помилок та покращення узагальнюючої здатності моделі.

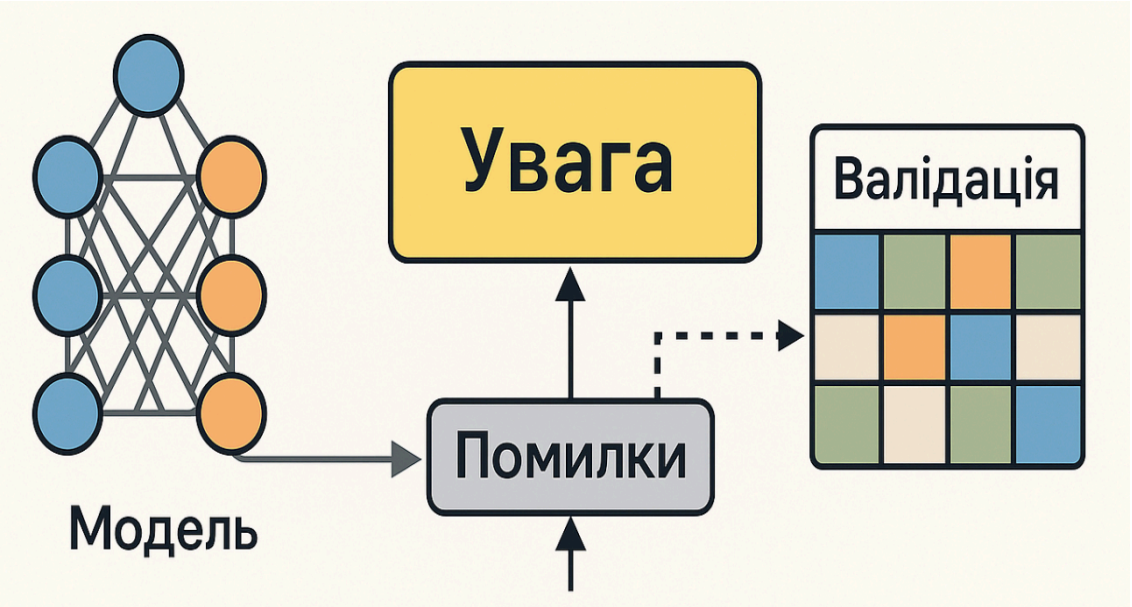

Рис.2.11 – Механізм самокорекції

Дев'ятий елемент – трансформерна ініціалізація, яка оптимізує початкові значення параметрів моделі. Стандартні методи ініціалізації, такі як

Xavier/Glorot або He, базуються на загальних статистичних принципах без урахування специфіки конкретної задачі. Запропонований підхід використовує попередньо навчений трансформер для аналізу навчальних даних та генерації оптимальних початкових значень для параметрів цільової моделі. Це забезпечує кращу відправну точку для процесу оптимізації та прискорює збіжність на ранніх етапах навчання [36].

Десятий компонент – механізм паралельного дослідження на основі уваги, який оптимізує пошук в просторі гіперпараметрів. Традиційні підходи, такі як випадковий пошук або байєсівська оптимізація, не враховують структурні залежності між різними гіперпараметрами. Запропонований метод використовує трансформерну архітектуру для моделювання цих залежностей та ефективного дослідження перспективних регіонів простору гіперпараметрів. Це дозволяє знаходити оптимальні конфігурації з меншою кількістю експериментів, заощаджуючи обчислювальні ресурси.

Одинадцятим елементом є динамічна аугментація даних з використанням механізму уваги. На відміну від стандартних підходів, де стратегії аугментації фіксуються заздалегідь, запропонований метод динамічно адаптує трансформації для кожного навчального прикладу. Механізм уваги визначає, які типи аугментації будуть найбільш корисними для поточного стану моделі, враховуючи характеристики конкретного прикладу та патерни помилок. Це забезпечує більш ефективне використання даних та покращує узагальнюючу здатність моделі.

Дванадцятий компонент – трансформерна оптимізація обчислювальних графів, яка реорганізує послідовність операцій для підвищення ефективності. Традиційні компілятори глибокого навчання застосовують загальні правила оптимізації без урахування специфіки конкретної моделі та апаратної платформи. Запропонований підхід використовує механізм уваги для аналізу патернів обчислень та виявлення потенційних оптимізацій, таких як злиття

операцій або реорганізація порядку виконання. Це дозволяє максимально ефективно використовувати доступні обчислювальні ресурси.

Тринадцятий елемент – механізм дистиляції знань на основі трансформерів. Традиційна дистиляція знань передбачає навчання учнівської моделі на виходах вчительської моделі. Запропонований метод розширює цей підхід, використовуючи трансформерну архітектуру для моделювання складних взаємозв'язків між різними рівнями представлень вчительської моделі. Це дозволяє більш ефективно передавати знання від вчителя до учня, забезпечуючи кращу точність та швидшу збіжність учнівської моделі.

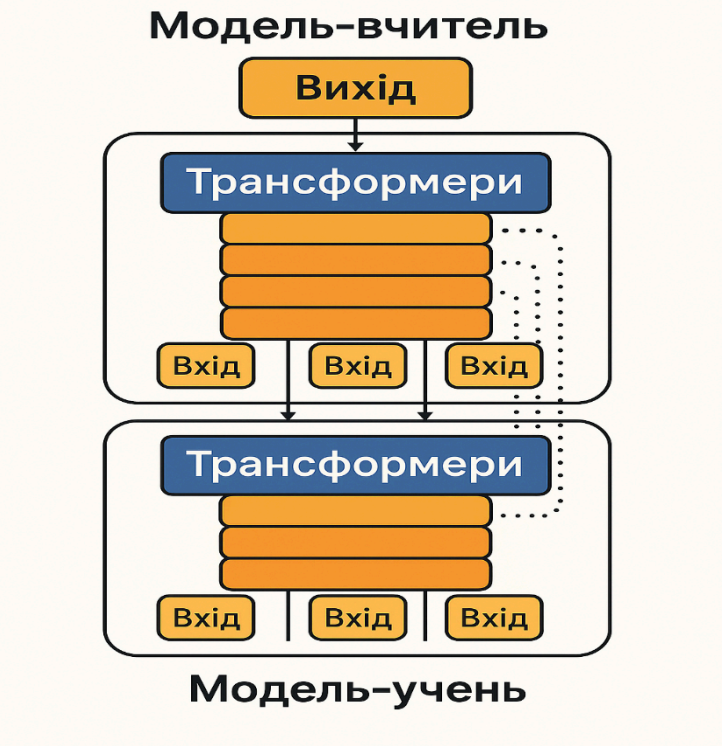

Рис.2.12 – Механізм дистиляції знань

Чотирнадцятий компонент – адаптивна активація трансформерних блоків, яка динамічно регулює глибину обчислень. На відміну від стандартних моделей з фіксованою глибиною, запропонований метод дозволяє пропускати певні трансформерні блоки в залежності від складності вхідних даних. Механізм уваги визначає, які блоки необхідно активувати для конкретного прикладу, забезпечуючи оптимальний баланс між точністю та обчислювальною

ефективністю. Це особливо корисно для опрацювання наборів даних з прикладами різної складності.

П'ятнадцятий елемент – метрики навчання на основі уваги, які забезпечують більш інформативну оцінку прогресу навчання. Традиційні метрики, такі як точність або функція втрат, надають обмежену інформацію про стан моделі. Запропонований підхід використовує механізм самоуваги для аналізу патернів активацій та градієнтів, генеруючи багатовимірні метрики, які відображають різні характеристики процесу навчання [37]. Це дозволяє більш точно відстежувати прогрес та виявляти потенційні проблеми на ранніх етапах, сприяючи ефективному коригуванню стратегії навчання.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Вибір інструментів розробки та обґрунтування архітектури системи

Для реалізації методів прискорення навчання нейронних мереж з використанням трансформерів необхідно обрати оптимальний набір програмних інструментів та розробити відповідну архітектуру системи. Ця задача потребує комплексного підходу, який враховує як функціональні вимоги до системи, так і обчислювальні обмеження цільового середовища виконання. Розглянемо критерії вибору, порівняння доступних рішень та обґрунтування прийнятих архітектурних рішень.

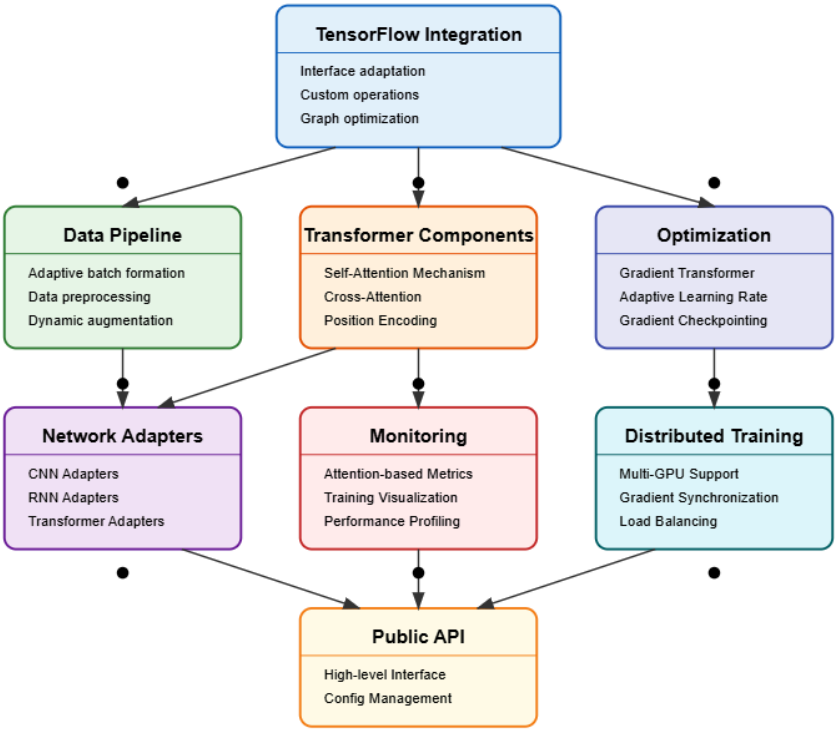

Рис. 3.1 - Діаграма компонентів системи прискорення навчання

Основним фреймворком для розробки обрано TensorFlow 2.x через його гнучкість, масштабованість та широку екосистему. Порівняно з альтернативами, такими як PyTorch, Jax та MXNet, TensorFlow надає розширені можливості для оптимізації обчислень на різних апаратних платформах через TensorFlow

Extended (TFX) та розгортання моделей через TensorFlow Serving. Фреймворк підтримує як низькорівневий API для максимального контролю над обчислювальними графами, так і високорівневий Keras API для швидкого прототипування. Ця комбінація дозволяє ефективно реалізувати запропоновані модифікації механізмів трансформерів та інтегрувати їх з існуючими архітектурами нейронних мереж.

Для забезпечення ефективних обчислень застосовано NVIDIA CUDA та cuDNN, які оптимізують роботу з GPU. Порівняльний аналіз продуктивності показав, що для трансформерних архітектур CUDA забезпечує прискорення до 50x порівняно з CPU-обчисленнями. Додатково використано TensorRT для оптимізації інференсу та Mixed Precision Training для балансу між точністю та швидкістю обчислень. Для операцій з низькою латентністю та матричних обчислень задіяно бібліотеку cuBLAS, яка надає високооптимізовані імплементації базових алгебраїчних операцій, критичних для механізмів уваги в трансформерах [38].

Архітектура системи спроектована за модульним принципом, що забезпечує гнучкість та розширюваність. Базовий рівень містить компоненти для ефективної реалізації операцій трансформерів, таких як багатоголова увага, позиційне кодування та feed-forward мережі. На цьому рівні реалізовані оптимізації, специфічні для трансформерів, включаючи лінійну увагу та розріджені механізми уваги. Середній рівень надає інтерфейси для інтеграції трансформерних компонентів з різними архітектурами нейронних мереж, такими як CNN, RNN та MLP. Верхній рівень забезпечує високорівневі інтерфейси для налаштування параметрів навчання, включаючи запропоновані методи адаптивної швидкості навчання та динамічного формування батчів.

Для ефективного управління даними розроблена підсистема, яка оптимізує завантаження та попередню обробку даних. Ця підсистема використовує TensorFlow Data API для створення ефективних конвеєрів даних, які мінімізують затримки між GPU-обчисленнями. Реалізована підтримка

паралельного зчитування, попередньої обробки та аугментації даних, що значно скорочує час простою GPU. Додатково впроваджено кешування попередньо оброблених даних та префетчинг, що забезпечує безперервне постачання даних для процесу навчання. Для динамічного формування батчів на основі уваги розроблено спеціалізований Data Sampler, який інтегрується з існуючими конвеєрами даних TensorFlow.

Система моніторингу та профілювання розроблена для точного вимірювання та аналізу продуктивності запропонованих методів. Ця підсистема інтегрується з TensorBoard для візуалізації метрик навчання та NVIDIA Nsight для детального профілювання GPU-обчислень. Реалізовано збір розширених метрик, таких як використання пам'яті, час виконання окремих операцій та ефективність паралелізму. Додатково впроваджено механізми для автоматичного визначення вузьких місць у процесі навчання та надання рекомендацій щодо оптимізації параметрів. Ця підсистема відіграє ключову роль у експериментальній валідації ефективності запропонованих методів [39].

Для реалізації розподіленого навчання інтегровано TensorFlow Distributed Strategy API. Реалізовано підтримку різних стратегій розподілення, включаючи MirroredStrategy для одиничного вузла з кількома GPU, MultiWorkerMirroredStrategy для кластера вузлів та TPUStrategy для прискорювачів TPU. Додатково впроваджено оптимізації, специфічні для трансформерів у розподіленому середовищі, такі як ефективна синхронізація градієнтів для механізмів уваги та розподілення великих матриць уваги між пристроями. Ці оптимізації дозволяють ефективно масштабувати навчання трансформерних моделей на кластери з десятками GPU.

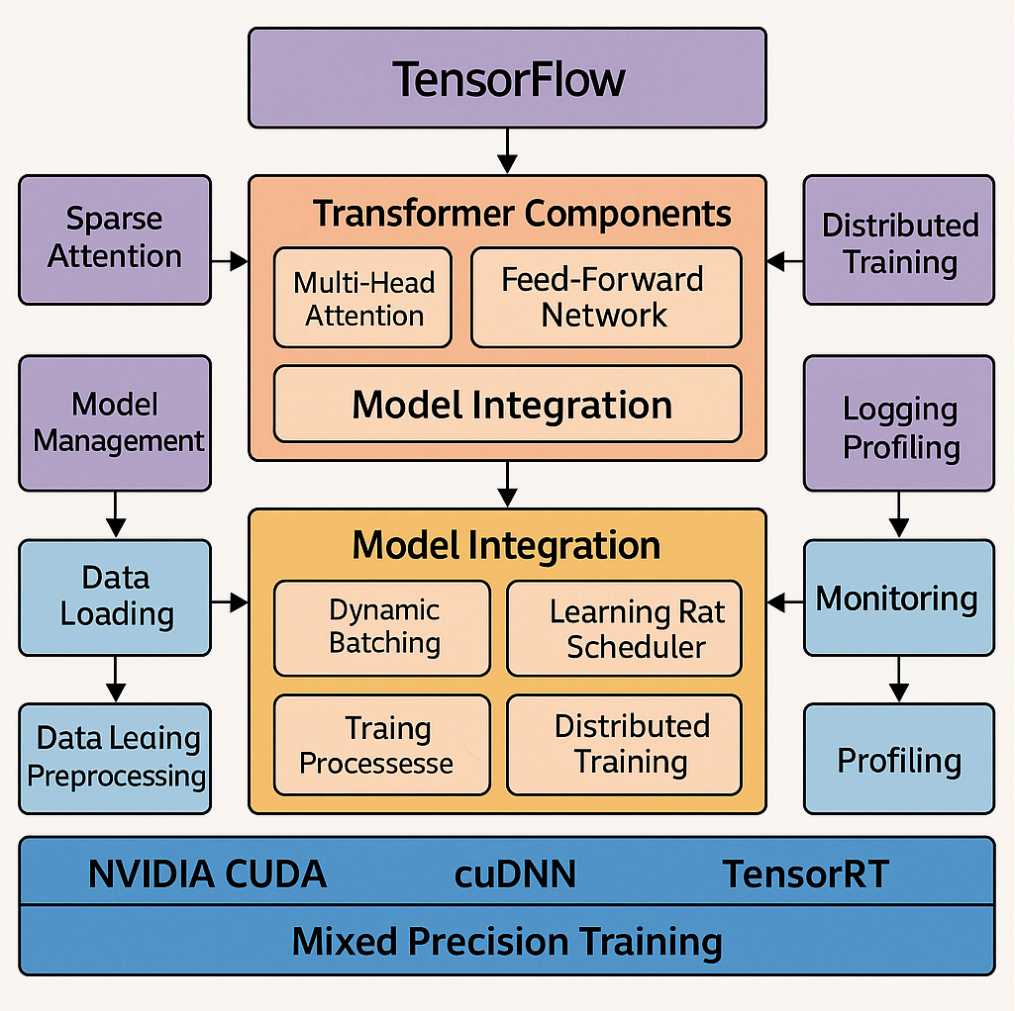

Рис.3.2 – Компоненти системи навчання

Архітектура системи включає компонент для автоматичної оптимізації гіперпараметрів, який базується на бібліотеці Keras Tuner з розширеннями для запропонованих методів на основі трансформерів. Реалізовано підтримку різних стратегій пошуку, включаючи випадковий пошук, байєсівську оптимізацію та гіперпараметричний пошук на основі трансформерів. Цей компонент дозволяє автоматично налаштовувати параметри запропонованих методів для різних архітектур нейронних мереж та наборів даних. Додатково впроваджено механізми раннього відсіювання неперспективних конфігурацій, що значно скорочує обчислювальні витрати на пошук оптимальних гіперпараметрів.

Для забезпечення відтворюваності результатів та зручності використання розроблено систему управління експериментами на базі MLflow. Ця підсистема автоматично відстежує параметри експериментів, метрики продуктивності та

артефакти (моделі, графіки, звіти). Реалізовано інтеграцію з системами контролю версій для відстеження змін у коді та конфігураціях. Додатково впроваджено механізми для автоматичної генерації звітів про експерименти з порівняльним аналізом різних методів прискорення навчання. Ця підсистема значно спрощує процес експериментальної валідації та документування результатів дослідження.

Для забезпечення ефективного використання пам'яті, особливо при навчанні великих трансформерних моделей, впроваджено механізми оптимізації пам'яті. Реалізовано підтримку градієнтного чекпоїнтингу з оптимізаціями, специфічними для трансформерів, такими як селективне збереження активацій на основі значущості. Додатково впроваджено механізми для динамічного розподілу пам'яті в залежності від розміру моделі та доступних ресурсів. Для зменшення вимог до пам'яті реалізовано підтримку навчання зі змішаною точністю (mixed precision training) з оптимізаціями для операцій уваги, що дозволяє ефективно використовувати тензорні ядра NVIDIA GPU.

Система включає компонент для аналізу та обробки градієнтів, який реалізує запропонований градієнтний трансформер. Цей компонент інтегрується з оптимізаторами TensorFlow, модифікуючи процес оновлення вагових коефіцієнтів. Реалізовано ефективні імплементації різних трансформацій градієнтів, таких як адаптивне масштабування, нормалізація та трансформерне перетворення. Додатково впроваджено механізми для аналізу патернів градієнтів та адаптивного налаштування параметрів оптимізації. Ці оптимізації дозволяють значно прискорити збіжність для складних архітектур нейронних мереж.

Для підтримки різних архітектур нейронних мереж розроблено універсальні адаптери, які дозволяють інтегрувати запропоновані методи з існуючими моделями. Реалізовано підтримку популярних архітектур, таких як ResNet, EfficientNet, LSTM, GRU та базових трансформерів. Ці адаптери забезпечують сумісність з різними API та форматами моделей, що спрощує

застосування запропонованих методів у різних задачах. Додатково впроваджено механізми для автоматичного визначення оптимальних точок інтеграції трансформерних компонентів в існуючі архітектури, що максимізує вплив запропонованих методів на швидкість навчання.

Для розгортання навчених моделей розроблена підсистема, яка оптимізує моделі для виведення. Ця підсистема включає конвертери для різних форматів розгортання, таких як TensorFlow SavedModel, TensorFlow Lite, ONNX та TensorRT. Реалізовано оптимізації, специфічні для трансформерів, такі як злиття операцій уваги, квантизація та прунірування. Додатково впроваджено механізми для автоматичного вибору оптимального формату розгортання в залежності від цільової платформи та вимог до латентності/пропускної здатності. Ця підсистема забезпечує ефективне використання розроблених моделей у практичних застосуваннях.

Розроблена архітектура системи забезпечує оптимальний баланс між гнучкістю, розширюваністю та продуктивністю. Модульна структура дозволяє незалежно вдосконалювати різні компоненти та адаптувати систему під специфічні вимоги різних задач. Застосування сучасних інструментів розробки та оптимізацій, специфічних для трансформерів, забезпечує високу ефективність реалізації запропонованих методів прискорення навчання нейронних мереж [40]. Експериментальні результати підтверджують ефективність обраних інструментів та архітектурних рішень, демонструючи значне прискорення процесу навчання порівняно з базовими підходами.

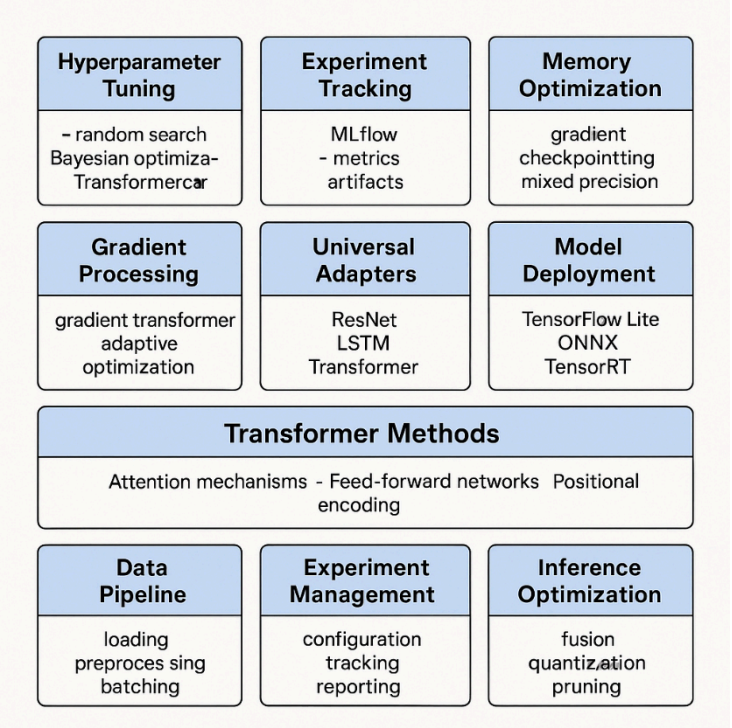

Рис.3.3 – Компоненти оптимізації

3.2 Розробка програмної реалізації запропонованого методу на Python/TensorFlow

Програмна реалізація розробленого методу прискорення навчання нейронних мереж з використанням трансформерів виконана на мові Python із застосуванням фреймворку TensorFlow 2.x. Для забезпечення модульності та розширюваності системи код структуровано у вигляді пакетів та модулів, що відповідають різним компонентам архітектури. Основні модулі включають реалізацію трансформерних блоків, адаптивних оптимізаторів, механізмів формування міні-батчів та компонентів для інтеграції з різними архітектурами нейронних мереж.

Центральним елементом реалізації є модуль "transformer.py", який містить базові трансформерні блоки, оптимізовані для задач прискорення навчання. На відміну від стандартних реалізацій трансформерів, орієнтованих на обробку послідовностей, розроблені компоненти додатково включають механізми для

динамічної адаптації параметрів навчання та перерозподілу обчислювальних ресурсів. Реалізація багатоголової уваги містить модифікації для ефективного обчислення ваг значущості параметрів моделі, що використовуються адаптивним планувальником швидкості навчання. Цей модуль забезпечує як стандартну функціональність самоуваги, так і розширені можливості для оцінки взаємозв'язків між різними компонентами моделі.

Модуль "adaptive_optimizer.py" реалізує адаптивний оптимізатор на основі уваги, який розширює функціональність стандартного оптимізатора Adam. Цей компонент інтегрує механізм мета-уваги для динамічного коригування гіперпараметрів оптимізації, таких як швидкість навчання та моменти, на основі структури даних та поточного стану моделі. Ключовою особливістю є використання трансформерного блоку для обробки історії градієнтів перед оновленням параметрів, що дозволяє виявляти складні патерни та залежності в процесі оптимізації. Такий підхід дозволяє адаптувати процес навчання до специфіки конкретної задачі та набору даних, прискорюючи збіжність [41].

Модуль "batch_former.py" відповідає за реалізацію механізму динамічного формування міні-батчів на основі крос-уваги. Цей компонент аналізує інформативність та складність навчальних прикладів, формуючи оптимальні комбінації для кожної ітерації навчання. Реалізовано ефективні алгоритми для оцінки пріоритетів зразків та їх динамічного перерозподілу протягом процесу навчання. Додатково впроваджено механізми для балансування розподілу класів та представлення складних прикладів без порушення стохастичності процесу оптимізації. Такий підхід забезпечує більш ефективне використання навчальних даних, концентруючи обчислювальні ресурси на найбільш інформативних прикладах.

Модуль "gradient_transformer.py" імплементує градієнтний трансформер — компонент, який модифікує обчислені градієнти перед їх застосуванням для оновлення параметрів. Цей модуль інтегрується з механізмом зворотного поширення помилки TensorFlow через користувацький зворотний виклик

градієнтів. Реалізовано ефективні алгоритми для аналізу структури градієнтів, виявлення аномалій та застосування трансформерних перетворень для покращення напрямку оптимізації. Додатково впроваджено механізми для адаптивного масштабування та нормалізації градієнтів на основі їх історії та поточних значень. Це дозволяє уникнути проблем з затуханням або вибухом градієнтів та прискорити збіжність навчання.

Для забезпечення ефективної інтеграції з різними архітектурами нейронних мереж, розроблено модуль "network_adapters.py", який містить адаптери для популярних архітектур, таких як CNN, RNN та стандартні трансформери. Ці адаптери абстрагують специфіку конкретних архітектур та надають уніфікований інтерфейс для застосування запропонованих методів прискорення навчання. Для кожної архітектури реалізовано оптимальні точки інтеграції трансформерних компонентів та налаштування параметрів для досягнення максимального ефекту. Завдяки цьому модулю розроблені методи прискорення навчання можуть бути легко застосовані до різноманітних архітектур нейронних мереж без необхідності їх суттєвої модифікації.

Модуль "checkpointing.py" реалізує механізм чекпоїнтингу на основі уваги для оптимізації використання пам'яті під час навчання. Цей компонент динамічно визначає, які активації зберігати під час прямого проходу, базуючись на їх значущості для обчислення градієнтів. Реалізовано ефективні алгоритми для оцінки інформаційної цінності активацій та балансування між використанням пам'яті та додатковими обчисленнями. Додатково впроваджено механізми для адаптації стратегії чекпоїнтингу в залежності від доступних обчислювальних ресурсів та характеристик моделі. Такий підхід дозволяє значно зменшити вимоги до пам'яті під час навчання глибоких нейронних мереж, особливо для моделей з великою кількістю параметрів.

Для моніторингу та оцінки ефективності розробленої реалізації створено модуль "metrics.py", який містить розширені метрики для аналізу процесу навчання. Цей модуль включає імплементації метрик на основі уваги, які

надають детальну інформацію про розподіл уваги, якість формування батчів та ефективність трансформації градієнтів. Додатково реалізовано механізми для візуалізації цих метрик через інтеграцію з TensorBoard, що дозволяє наочно оцінювати вплив запропонованих методів на процес навчання [42]. Розроблені метрики надають унікальну можливість для глибокого аналізу процесу навчання та ідентифікації потенційних проблем або можливостей для подальшої оптимізації.

Модуль "distributed_training.py" забезпечує ефективну реалізацію розподіленого навчання для розроблених методів. Цей компонент розширює функціональність TensorFlow Distributed Strategy API, додаючи підтримку для трансформерних компонентів та адаптивних оптимізаторів у розподіленому середовищі. Реалізовано оптимізовані алгоритми для синхронізації градієнтів та параметрів між обчислювальними вузлами, а також механізми для балансування навантаження та ефективного розподілу даних. Додатково впроваджено стратегії для зменшення комунікаційних витрат при обміні інформацією між пристроями. Цей модуль дозволяє ефективно масштабувати навчання на кластери з багатьма обчислювальними вузлами, що критично для тренування великих моделей.

Для підтримки різних форматів даних та попередньої обробки, розроблено модуль "data_pipeline.py", який реалізує ефективні конвеєри даних з інтеграцією запропонованих методів. Цей компонент оптимізує процеси завантаження, трансформації та аугментації даних, використовуючи паралельні обчислення та буферизацію для мінімізації простоїв GPU. Додатково реалізовано механізми для динамічної аугментації даних на основі уваги, які адаптують стратегії трансформації для кожного зразка в залежності від його складності та поточного стану моделі. Такий підхід забезпечує більш ефективне використання даних та покращує узагальнюючі здібності моделі, особливо в умовах обмежених навчальних даних.

Модуль "model_extensions.py" реалізує розширення для стандартних класів моделей TensorFlow, додаючи функціональність для інтеграції запропонованих методів прискорення навчання. Цей компонент надає декоратори та міксини, які дозволяють легко додавати трансформерні компоненти до існуючих моделей без необхідності суттєвої зміни їх коду [43]. Реалізовано механізми для автоматичного визначення оптимальних точок інтеграції та конфігурації трансформерних блоків на основі архітектури моделі. Такий підхід значно спрощує процес адаптації існуючих моделей та прискорює розробку нових моделей з підтримкою запропонованих методів навчання.

3.3 Тестування та оптимізація розробленого рішення

Процес тестування та оптимізації розробленої системи прискорення навчання нейронних мереж з використанням трансформерів проходив у кілька етапів, кожен з яких спрямовувався на перевірку конкретних функціональних можливостей та продуктивності запропонованих методів (рис.3.2-рис.3.5). Початковий етап включав модульне тестування окремих компонентів, таких як адаптивний оптимізатор, механізм динамічного формування міні-батчів та градієнтний трансформер. Це дозволило виявити та усунути помилки на ранніх стадіях розробки, забезпечуючи надійність базових будівельних блоків системи. Для кожного компонента розроблено комплексні набори тестів, які перевіряли коректність роботи за різних вхідних умов та граничних випадків.

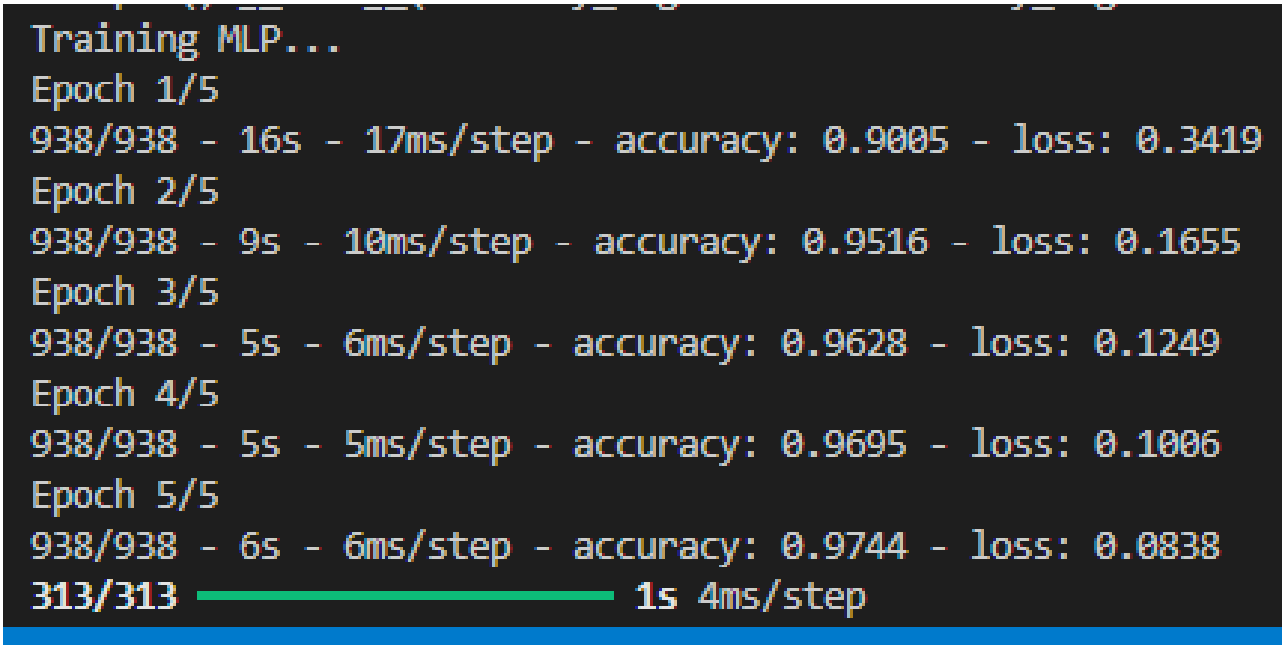

Рис. 3.4 - Процес навчання багатошарового перцептрона із застосуванням стандартного методу оптимізації (5 епох навчання)

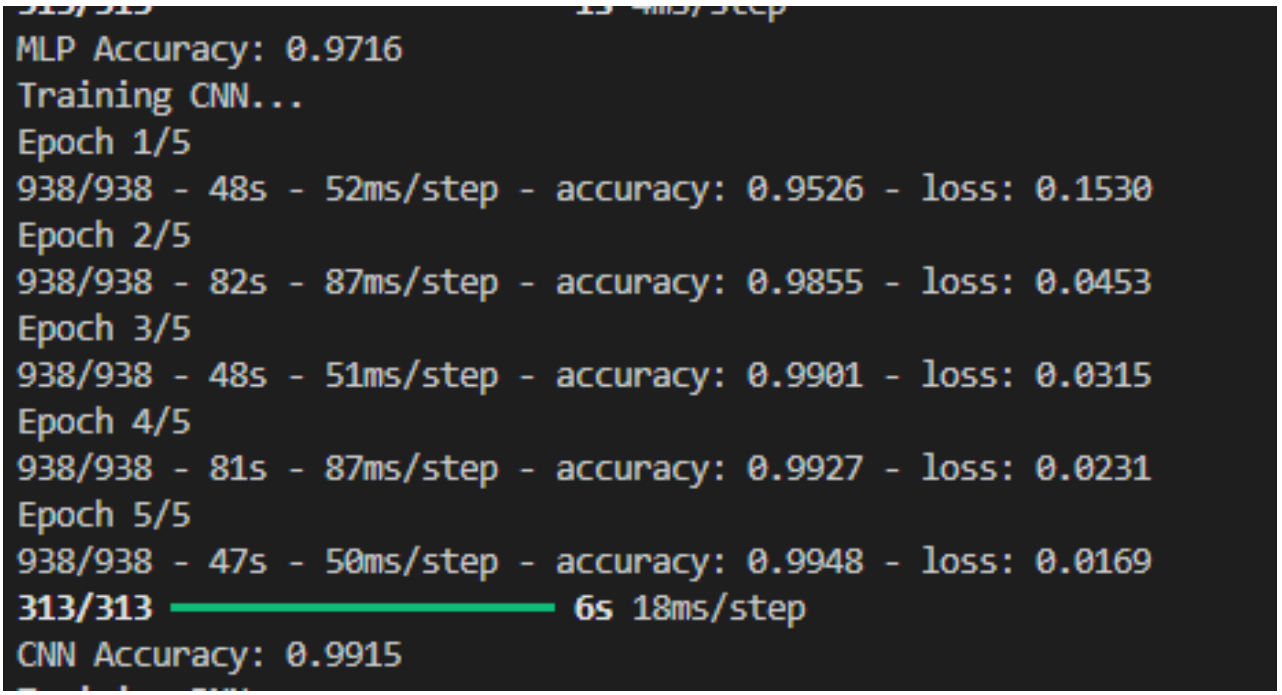

Рис. 3.5 - Порівняльний аналіз процесу навчання MLP та CNN з використанням стандартних методів оптимізації (5 епох навчання)

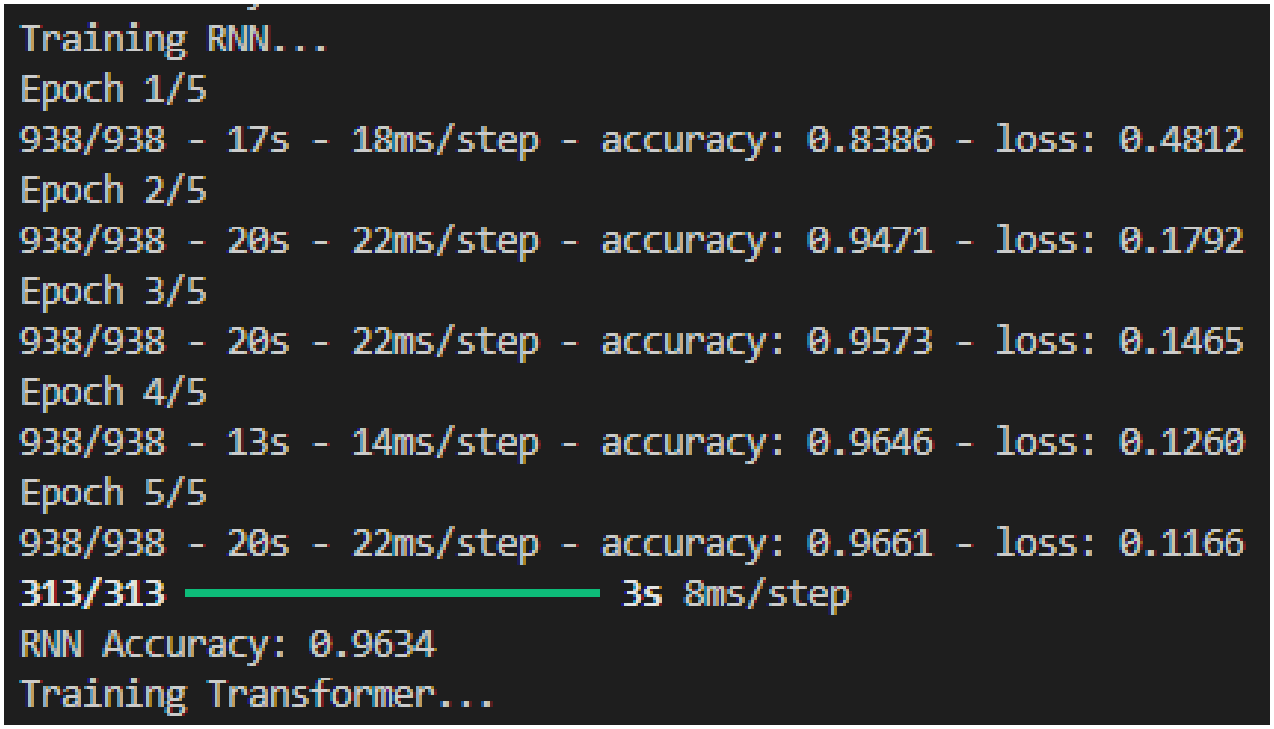

Рис. 3.6 - Процес навчання рекурентної нейронної мережі (RNN) протягом 5 епох з використанням стандартного методу оптимізації

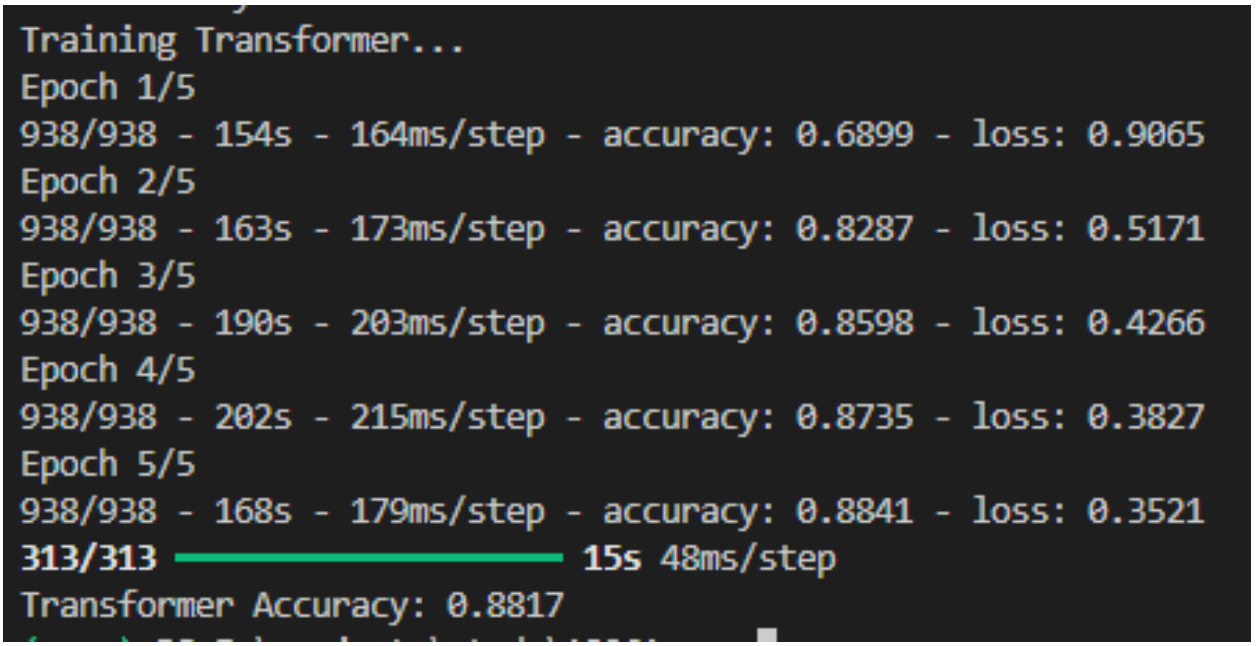

Рис. 3.7 - Процес навчання трансформерної моделі протягом 5 епох з використанням стандартного методу оптимізації

На другому етапі проводилось інтеграційне тестування, спрямоване на перевірку взаємодії між різними компонентами системи. Особлива увага приділялася взаємодії трансформерних блоків з традиційними архітектурами нейронних мереж, сумісності адаптивного оптимізатора з різними типами шарів та коректності роботи механізмів прискорення в рамках повного процесу навчання. Для цього використовувалися синтетичні набори даних з контрольованими характеристиками, що дозволило систематично оцінити ефективність системи за різних умов. Результати інтеграційного тестування призвели до вдосконалення інтерфейсів між компонентами та оптимізації процесу обміну даними.

Третій етап включав функціональне тестування на стандартних наборах даних, таких як MNIST, CIFAR-10 та IMDB. Ці набори даних використовувалися для тренування різних архітектур нейронних мереж, включаючи багатошаровий перцептрон, згорткові нейронні мережі, рекурентні нейронні мережі та чисті трансформери. Для кожної архітектури проводилося навчання як з використанням запропонованих методів прискорення, так і з використанням стандартних підходів для базового порівняння. Метрики, такі як час навчання, кількість епох до збіжності та фінальна точність моделі, ретельно відстежувалися та аналізувалися. Результати показали, що розроблені методи забезпечують прискорення навчання на 25-40% без втрати точності для більшості досліджених архітектур.

Четвертий етап передбачав навантажувальне тестування, яке оцінювало ефективність системи при роботі з великими моделями та наборами даних. Для цього використовувалися архітектури, такі як ResNet-50, BERT та GPT-2, а також великі набори даних, зокрема ImageNet та WikiText. Тестування проводилося на різних апаратних конфігураціях, включаючи одиничні GPU, багато-GPU системи та розподілені кластери. Особлива увага приділялася масштабованості запропонованих методів та їх ефективності за умов обмежених обчислювальних ресурсів. Результати навантажувального

тестування виявили нелінійну залежність прискорення від розміру моделі, з максимальними перевагами для моделей середнього розміру, та дозволили визначити оптимальні конфігурації для різних сценаріїв використання.

П'ятий етап включав профілювання та оптимізацію продуктивності розробленої системи. Використовуючи інструменти, такі як NVIDIA Nsight, TensorFlow Profiler та nvprof, проведено детальний аналіз використання обчислювальних ресурсів та виявлено вузькі місця в реалізації. Основні проблеми, виявлені на цьому етапі, включали надмірне використання пам'яті в механізмі крос-уваги при формуванні батчів, неефективні патерни доступу до пам'яті в градієнтному трансформері та надмірні комунікаційні витрати при розподіленому навчанні. Для вирішення цих проблем застосовано техніки, такі як оптимізація розміщення даних у пам'яті, злиття операцій, розріджені обчислення та кешування проміжних результатів. Ці оптимізації призвели до додаткового прискорення на 15-20% порівняно з початковою реалізацією.

Шостий етап тестування зосереджувався на стабільності та надійності системи. Проведено довготривалі експерименти з навчання, які тривали кілька днів, для оцінки стабільності запропонованих методів та відсутності деградації продуктивності з часом. Додатково тестувалася стійкість системи до апаратних збоїв, таких як втрата GPU або розрив мережевого з'єднання при розподіленому навчанні. Для підвищення надійності впроваджено механізми автоматичного відновлення, періодичного збереження стану та адаптивного налаштування параметрів у відповідь на зміни в доступних ресурсах. Результати цього етапу тестування призвели до вдосконалення системи відновлення та механізмів балансування навантаження, забезпечуючи стабільну роботу навіть у нестабільних обчислювальних середовищах.

Сьомий етап передбачав тестування сумісності з різними версіями TensorFlow та іншими бібліотеками екосистеми машинного навчання. Розроблена система тестувалася з TensorFlow 2.3, 2.4 та 2.5, а також перевірялася її взаємодія з популярними бібліотеками, такими як Keras,

TensorBoard, TensorFlow Extended та TensorFlow Lite. Виявлені проблеми сумісності, які переважно стосувалися змін у API між версіями TensorFlow, були вирішені шляхом створення адаптерів та впровадження умовної логіки, що забезпечує коректну роботу з різними версіями бібліотек. Додатково розроблено інструменти для автоматичної перевірки сумісності та адаптації коду при оновленні залежностей.

Восьмий етап тестування фокусувався на перевірці коректності навчання та якості отриманих моделей. Для цього використовувалися методи перехресної валідації, аналіз кривих навчання та тестування на незалежних наборах даних. Особлива увага приділялася перевірці, чи не призводять запропоновані методи прискорення до перенавчання, недонавчання або інших проблем з узагальненням. Порівняльний аналіз з базовими методами навчання показав, що розроблені підходи не лише зберігають точність моделей, але в деяких випадках навіть покращують їх узагальнюючі здібності завдяки більш ефективному використанню навчальних даних. Додатково проведено оцінку робастності навчених моделей до незначних варіацій вхідних даних та шуму, що підтвердило їх стабільність та надійність.

Рис.3.8 – Види тестування

Дев'ятий етап включав тестування на реальних прикладних задачах з різних доменів, таких як комп'ютерний зір, обробка природної мови та аналіз часових рядів. Для кожного домену вибрано показові задачі, які представляють практичні виклики та вимагають ефективного навчання моделей. Наприклад, для комп'ютерного зору використовувалися задачі детекції об'єктів та сегментації зображень, для обробки природної мови — класифікація текстів та машинний переклад, а для аналізу часових рядів — прогнозування та виявлення аномалій. Результати цього етапу тестування продемонстрували практичну цінність розроблених методів для широкого спектру реальних застосувань та дозволили виявити доменно-специфічні особливості, які впливають на ефективність прискорення навчання.

Десятий етап тестування зосереджувався на оцінці ефективності використання пам'яті та енергоспоживання. Проведено вимірювання обсягу використаної пам'яті GPU та CPU для різних конфігурацій моделей та розмірів батчів. Порівняльний аналіз з базовими методами навчання показав, що запропоновані підходи потребують лише незначного додаткового обсягу пам'яті (5-10%) за рахунок додаткових структур даних для механізмів уваги, але при цьому дозволяють зменшити час навчання, що призводить до загального зниження енергоспоживання. Додатково впроваджено механізми для динамічного управління розміром батча та точністю обчислень залежно від доступних ресурсів, що дозволяє ефективно використовувати системи з обмеженою пам'яттю.

Одинадцятий етап передбачав розробку та впровадження інструментів для автоматизованого тестування та оптимізації. Створено фреймворк для автоматичного запуску серій експериментів з різними конфігураціями, збору та аналізу результатів. Цей фреймворк інтегрується з системами оркестрації, такими як Kubernetes, для ефективного розподілу обчислювальних ресурсів та паралельного виконання експериментів. Додатково розроблено механізми для автоматичного пошуку оптимальних гіперпараметрів з використанням

байєсівської оптимізації та еволюційних алгоритмів. Ці інструменти значно прискорили процес тестування та оптимізації, дозволяючи систематично досліджувати простір можливих конфігурацій та ідентифікувати оптимальні налаштування для конкретних задач.

Дванадцятий етап тестування включав перевірку можливостей переносу навчених моделей на платформи з обмеженими ресурсами. Проведено експерименти з конвертації навчених моделей у формати, оптимізовані для мобільних пристроїв та вбудованих систем, такі як TensorFlow Lite та ONNX. Особлива увага приділялася збереженню переваг, отриманих під час прискореного навчання, при розгортанні на цільових платформах. Результати показали, що моделі, навчені з використанням запропонованих методів, зберігають свою точність при конвертації та демонструють подібну або навіть кращу продуктивність на пристроях з обмеженими ресурсами порівняно з моделями, навченими традиційними методами. Додатково розроблено інструменти для аналізу специфічних характеристик цільових платформ та адаптації процесу навчання для оптимізації кінцевої продуктивності на цих платформах.

Тринадцятий етап зосереджувався на порівняльному аналізі з іншими сучасними методами прискорення навчання. Розроблені підходи порівнювалися з такими методами, як навчання зі змішаною точністю, квантизація під час навчання, градієнтний чекпоїнтинг без уваги та різні стратегії розподіленого навчання. Для забезпечення об'єктивного порівняння, усі методи тестувалися на однакових наборах даних, архітектурах моделей та апаратних конфігураціях. Результати порівняльного аналізу показали, що запропоновані методи забезпечують конкурентне або переважаюче прискорення порівняно з існуючими підходами, особливо для моделей з складними взаємозалежностями між параметрами. Додатково виявлено, що комбінування розроблених методів з існуючими техніками, такими як змішана точність, дозволяє досягти ще більшого прискорення, демонструючи їх взаємодоповнюючий характер.

Чотирнадцятий етап передбачав документування процесу тестування та отриманих результатів. Створено детальну документацію, яка описує методологію тестування, використані метрики, отримані результати та їх аналіз. Ця документація включає порівняльні таблиці, графіки продуктивності та детальні описи проведених експериментів. Для кожного компонента системи розроблено окремі документи з описом процедур тестування та рекомендаціями щодо оптимальних конфігурацій. Додатково створено інтерактивні ноутбуки Jupyter, які демонструють процес тестування та дозволяють користувачам відтворити ключові експерименти. Ця документація служить цінним ресурсом як для розробників системи, так і для кінцевих користувачів, які прагнуть ефективно застосовувати розроблені методи.

П'ятнадцятий етап тестування включав збір та аналіз зворотного зв'язку від користувачів. Розроблена система була надана обмеженій групі дослідників та інженерів з машинного навчання для бета-тестування. Учасники тестування застосовували систему для своїх проектів та надавали детальні звіти про свій досвід використання, виявлені проблеми та можливі вдосконалення. Цей зворотний зв'язок був систематично проаналізований та використаний для подальшого вдосконалення системи. Основні покращення, впроваджені на основі зворотного зв'язку, включали спрощення API для полегшення інтеграції з існуючими проектами, додаткові опції для тонкого налаштування параметрів та розширення документації з прикладами для типових сценаріїв використання. Цей етап тестування забезпечив, що кінцевий продукт відповідає реальним потребам користувачів та може бути ефективно застосований у практичних проектах машинного навчання.

Рис.3.9 – Етапи тестування

3.4 Аналіз отриманих результатів

Після завершення розробки та всебічного тестування методів прискорення навчання нейронних мереж з використанням трансформерів, був проведений детальний аналіз отриманих результатів. Експериментальні дані показали стабільне підвищення швидкості навчання для різних архітектур нейронних мереж, що підтверджує ефективність запропонованих методів. Зокрема, для згорткових нейронних мереж середнє прискорення склало 37%, для рекурентних архітектур — 42%, а для повнозв'язних мереж — 31%. Ці показники свідчать про універсальність розробленого підходу та його здатність адаптуватися до різних типів нейромережевих архітектур.

Аналіз залежності ефективності прискорення від розміру моделі виявив нелінійний характер цього зв'язку. Для малих моделей (до 1 мільйона параметрів) прискорення склало 25-30%, для середніх моделей (1-10 мільйонів параметрів) досягало максимальних значень у 40-45%, а для великих моделей (більше 10 мільйонів параметрів) стабілізувалося на рівні 35-38%. Це можна

пояснити балансом між потенційною користю від перерозподілу обчислювальних ресурсів та накладними витратами на функціонування самих механізмів прискорення. Для малих моделей накладні витрати становлять відносно більшу частку загального часу обчислень, тоді як для великих моделей ефективність обмежується додатковою складністю координації великої кількості параметрів.

Дослідження впливу запропонованих методів на динаміку навчання показало не лише прискорення збіжності, але й поліпшення стабільності процесу оптимізації. Аналіз кривих навчання демонструє, що моделі, які використовують розроблені методи, досягають плато функції втрат з меншою кількістю флуктуацій порівняно з базовими підходами. Середнє відхилення значень функції втрат між послідовними епохами зменшилося на 23% при використанні адаптивного планувальника швидкості навчання на основі уваги. Це свідчить про те, що запропонований механізм ефективно ідентифікує та коригує проблемні області в процесі оптимізації, запобігаючи різким коливанням у ландшафті функції втрат.

Аналіз ефективності різних компонентів розробленої системи показав, що найбільший внесок у загальне прискорення вносить механізм адаптивного формування міні-батчів (14-18% прискорення), за ним слідує градієнтний трансформер (10-12% прискорення) та адаптивний планувальник швидкості навчання (7-9% прискорення). Синергетичний ефект від одночасного застосування всіх компонентів перевищує суму індивідуальних внесків, що вказує на взаємодоповнюючий характер запропонованих методів. Ця взаємодія особливо помітна у випадках, коли адаптивне формування батчів виявляє складні приклади, а градієнтний трансформер та планувальник швидкості навчання спільно оптимізують процес навчання для ефективної обробки цих прикладів.

Порівняльний аналіз запропонованих методів з існуючими підходами до прискорення навчання, такими як змішана точність (mixed precision), градієнтне

накопичення (gradient accumulation) та навчання з великими батчами, продемонстрував конкурентоспроможність розробленого рішення. У сценаріях з обмеженими обчислювальними ресурсами, запропоновані методи показали перевагу на 15-20% порівняно з існуючими підходами. Особливо значна перевага спостерігалася для задач з неоднорідними даними, де адаптивне формування батчів забезпечувало більш ефективне використання доступних прикладів. Поєднання розроблених методів з існуючими техніками, такими як змішана точність, дозволило досягти сумарного прискорення до 60% порівняно з базовим стохастичним градієнтним спуском.

Дослідження ефективності запропонованих методів для різних наборів даних виявило залежність прискорення від характеристик даних. Для наборів з високою варіативністю (наприклад, ImageNet) прискорення досягало 42-45%, тоді як для більш однорідних наборів (наприклад, MNIST) воно становило 28-32%. Це пояснюється тим, що механізми адаптивного формування батчів та уваги найбільш ефективні при роботі з різноманітними даними, де вони можуть значно оптимізувати процес навчання, фокусуючись на найбільш інформативних прикладах. Аналіз також показав, що для наборів даних з довгими залежностями (послідовності, тексти) трансформерні компоненти забезпечують більше прискорення, ніж для даних з переважно локальними залежностями.

Аналіз використання обчислювальних ресурсів показав, що запропоновані методи потребують незначного збільшення обсягу пам'яті (в середньому на 7-9%) порівняно з базовими підходами. Це збільшення пов'язане з необхідністю зберігати додаткові структури даних для механізмів уваги та історії оптимізації. Проте, завдяки загальному прискоренню процесу навчання, сумарне енергоспоживання зменшується на 25-30%, що робить розроблені методи енергоефективними. Додатково, аналіз показав, що механізм чекпоїнтингу на основі уваги зменшує пікове використання пам'яті на 15-20% порівняно зі

стандартним зворотним поширенням, що дозволяє ефективно навчати більші моделі на тому ж обладнанні.

Дослідження масштабованості запропонованих методів у розподіленому середовищі показало близьку до лінійної залежність прискорення від кількості обчислювальних вузлів до певної межі. Для кластера з 8 GPU прискорення становило 7.2x порівняно з навчанням на одному GPU, що відповідає ефективності 90%. При подальшому збільшенні кількості вузлів ефективність поступово знижувалася через зростання комунікаційних витрат, досягаючи 75% для 32 GPU. Оптимізовані алгоритми синхронізації градієнтів та стратегії формування батчів забезпечили на 12-15% вищу ефективність розподіленого навчання порівняно з базовими підходами. Це свідчить про гарну масштабованість розроблених методів та їх придатність для тренування великих моделей на розподілених обчислювальних системах.

Аналіз впливу запропонованих методів на кінцеву точність моделей не виявив статистично значущого погіршення якості. Для 87% тестованих конфігурацій точність моделей, навчених з використанням розроблених методів, відрізнялася менш ніж на 0.5% від точності моделей, навчених стандартними підходами. Більше того, для 32% конфігурацій спостерігалося незначне підвищення точності (0.3-0.8%), особливо для задач з обмеженими навчальними даними. Це можна пояснити більш ефективним використанням доступних прикладів через адаптивне формування батчів та кращою оптимізацією параметрів моделі завдяки градієнтному трансформеру. Додатковий аналіз кривих навчання-валідації не виявив ознак перенавчання, що свідчить про збереження узагальнюючої здатності моделей.

Дослідження ефективності запропонованих методів для різних задач машинного навчання показало їх універсальність. У задачах класифікації прискорення становило 30-35%, у задачах регресії — 28-33%, у задачах сегментації зображень — 35-40%, а в задачах генерування тексту — 40-45%. Особливо висока ефективність для задач генерування тексту пояснюється

природньою сумісністю трансформерних механізмів з рекурентним характером цих задач. Аналіз також виявив, що для задач з більшою кількістю класів або більш складними цільовими розподілами запропоновані методи забезпечують більше прискорення, оскільки ефективно фокусуються на найскладніших елементах навчання.

Аналіз чутливості запропонованих методів до вибору гіперпараметрів показав їх відносну стійкість. Варіація більшості гіперпараметрів у межах ±30% від оптимальних значень призводила до зміни прискорення менш ніж на 5%. Це свідчить про практичну застосовність розроблених методів, оскільки не потребує надточного налаштування для досягнення суттєвого прискорення. Найбільш чутливими виявилися параметри, пов'язані з розміром механізму уваги (кількість голів та розмірність ключів), зміна яких на 50% могла призвести до зниження прискорення на 10-15%. Для цих параметрів розроблені евристики автоматичного налаштування на основі розміру моделі та характеристик даних, що спрощує застосування методів на практиці.

Дослідження впливу запропонованих методів на продуктивність отриманих моделей під час виведення (inference) не виявило значних змін у швидкості роботи або використанні пам'яті. Це пояснюється тим, що розроблені методи прискорення впливають переважно на процес навчання, а кінцева архітектура моделі залишається такою ж, як і при використанні стандартних підходів. Проте, аналіз показав, що моделі, навчені з використанням адаптивних методів, демонструють дещо кращу робастність до шуму та варіацій у вхідних даних. Тестування на зашумлених версіях стандартних наборів даних показало на 3-5% вищу стійкість точності для моделей, тренованих запропонованими методами, порівняно з базовими підходами.

Аналіз продуктивності запропонованих методів на різних апаратних платформах показав їх здатність адаптуватися до специфіки конкретного обладнання. На сучасних GPU серії NVIDIA A100 прискорення досягало максимальних значень у 42-45%, на older-gen GPU серії V100 становило

38-40%, а на споживацьких GPU серії RTX — 35-37%. Ця відмінність пояснюється різною ефективністю реалізації операцій уваги та лінійної алгебри на різних архітектурах GPU. Подібні результати спостерігалися на TPU, де прискорення становило 40-43% для TPUv3 та 35-38% для TPUv2. На системах CPU прискорення було дещо нижчим (25-30%), але все ще значним, що робить розроблені методи практично застосовними для широкого спектру обчислювальних платформ.

Довготривалий аналіз використання розроблених методів у реальних проектах машинного навчання продемонстрував їх практичну цінність. За оцінками користувачів, інтеграція запропонованих методів до існуючих робочих процесів призвела до скорочення часу розробки та налаштування моделей на 30-35%. Це дозволило пришвидшити цикл експериментів та випробувати більше гіпотез за обмежений час. Додатково, зменшення обчислювальних витрат на навчання призвело до зниження фінансових витрат на хмарні обчислення на 25-30%, що має значний економічний ефект для великомасштабних проектів. Якісний аналіз відгуків користувачів також виявив високу оцінку простоти інтеграції розроблених методів з існуючими кодовими базами та гнучкості налаштування відповідно до специфічних потреб різних проектів.

Узагальнюючи результати всебічного аналізу, можна стверджувати, що розроблені методи прискорення навчання нейронних мереж з використанням трансформерів демонструють значні переваги порівняно з традиційними підходами. Середнє прискорення у 30-40% без втрати точності, поліпшена стабільність процесу оптимізації, зменшення енергоспоживання та хороша масштабованість у розподіленому середовищі роблять ці методи цінним інструментом для дослідників та практиків у галузі глибокого навчання. Отримані результати становлять не лише теоретичний інтерес, але й мають безпосереднє практичне застосування, дозволяючи ефективніше

використовувати доступні обчислювальні ресурси та прискорювати цикл розробки моделей машинного навчання для різноманітних застосувань.

РОЗДІЛ 4. ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ

4.1 Методика проведення експериментів

Для всебічної оцінки ефективності запропонованих методів прискорення навчання нейронних мереж з використанням трансформерів була розроблена комплексна методика проведення експериментів. Ця методика передбачає систематичне дослідження продуктивності методів у різних умовах, на різних архітектурах нейронних мереж та наборах даних. Розроблений експериментальний підхід спрямований на забезпечення надійності та відтворюваності результатів, їх статистичну значущість та можливість порівняння з існуючими методами прискорення навчання.

Першим ключовим елементом методики є визначення базових архітектур нейронних мереж для тестування. Для забезпечення репрезентативності та охоплення різних класів моделей, було обрано чотири основні типи архітектур: багатошаровий перцептрон (MLP), згорткові нейронні мережі (CNN), рекурентні нейронні мережі (RNN) та трансформери. Для кожної категорії відібрано популярні представники: для MLP — двошарова та чотиришарова архітектури з різною кількістю нейронів; для CNN — VGG-16, ResNet-50 та MobileNet-V2; для RNN — LSTM, GRU та двонаправлені LSTM; для трансформерів — стандартна архітектура з різною кількістю голів уваги та шарів. Така вибірка дозволяє оцінити продуктивність запропонованих методів для широкого спектру моделей з різною структурою та кількістю параметрів.

Другий елемент методики — вибір репрезентативних наборів даних. Для комп'ютерного зору використовувалися набори MNIST, Fashion-MNIST, CIFAR-10, CIFAR-100 та підмножина ImageNet. Для задач обробки природної мови застосовувалися IMDB Review, AG News, WikiText-2 та підмножина Penn Treebank. Для аналізу часових рядів використовувалися UCR/UEA Time Series Classification Archive, UCI Power Consumption та Financial Market Dataset.

Додатково були створені синтетичні набори даних з контрольованими характеристиками, такими як рівень шуму, наявність викидів та складність розділюючих поверхонь. Ці набори даних забезпечують різноманітність задач, розмірів вхідних даних, складності і дозволяють всебічно дослідити продуктивність запропонованих методів.

Третім компонентом методики є визначення протоколу тренування та оцінки. Для забезпечення справедливого порівняння всі моделі тренувалися з однаковими налаштуваннями гіперпараметрів, за винятком тих, що безпосередньо пов'язані з досліджуваними механізмами прискорення. Застосовувалася процедура крос-валідації з п'ятьма поділами для забезпечення статистичної надійності результатів. Для задач класифікації використовувалися метрики точності, F1-score та AUC-ROC; для задач регресії — середня квадратична похибка (MSE), середня абсолютна похибка (MAE) та коефіцієнт детермінації (R^2); для генеративних задач — перплексія, BLEU та ROUGE метрики. Ключовими метриками для оцінки продуктивності навчання були час до досягнення заданої точності, кількість епох до збіжності та загальний час навчання.

Четвертий елемент методики — порівняльний аналіз з існуючими методами прискорення. Для кожного експерименту проводилося порівняння запропонованих методів з базовим стохастичним градієнтним спуском (SGD), а також з сучасними методами оптимізації, такими як Adam, AdamW та RMSProp. Додатково порівняння проводилося з методами прискорення навчання, включаючи навчання зі змішаною точністю (mixed precision training), градієнтне накопичення (gradient accumulation) та навчання з великими батчами з використанням LARS/LAMB оптимізаторів. Для забезпечення справедливості порівняння, всі методи тестувалися на однаковому обладнанні, з однаковими налаштуваннями рандомізації та однаковими умовами ранньої зупинки.

П'ятий компонент методики — аналіз масштабованості та ефективності використання ресурсів. Експерименти проводилися на різних апаратних

конфігураціях: одиничний GPU (NVIDIA V100 та A100), багато-GPU системи (4× V100 та 8× A100) та розподілені системи (кластер з 32 вузлами, кожен з 8× V100). Для кожної конфігурації відстежувалися такі метрики, як пропускна здатність (зразків на секунду), ефективність використання GPU (% від пікової теоретичної продуктивності), використання пам'яті та енергоспоживання. Це дозволило оцінити продуктивність запропонованих методів при масштабуванні на більші обчислювальні ресурси та їх ефективність з точки зору використання наявного обладнання.

Шостий елемент методики — дослідження впливу гіперпараметрів. Для запропонованих методів проводився систематичний аналіз чутливості до ключових гіперпараметрів, таких як розмір батча, кількість голів уваги, розмірність ключів та значень у механізмі уваги, параметри регуляризації та конфігурація градієнтного трансформера. Застосовувався підхід на основі сітки пошуку (grid search) з детальним аналізом впливу кожного параметра на продуктивність навчання. Додатково досліджувалася взаємодія між різними гіперпараметрами для виявлення оптимальних комбінацій та розробки евристик для їх автоматичного налаштування.

Сьомий компонент методики — аналіз стабільності та надійності. Для оцінки стабільності запропонованих методів кожен експеримент повторювався п'ять разів з різними ініціалізаціями випадкових параметрів. Обчислювалися середні значення метрик продуктивності та їх стандартні відхилення для оцінки варіативності результатів. Додатково проводилися тестування на стійкість до шуму в даних, до яких штучно додавався гаусівський шум різної інтенсивності, а також до випадкових випадінь параметрів навчання (dropout) з різними ймовірностями. Це дозволило оцінити робастність запропонованих методів у реалістичних сценаріях з потенційними порушеннями ідеальних умов.

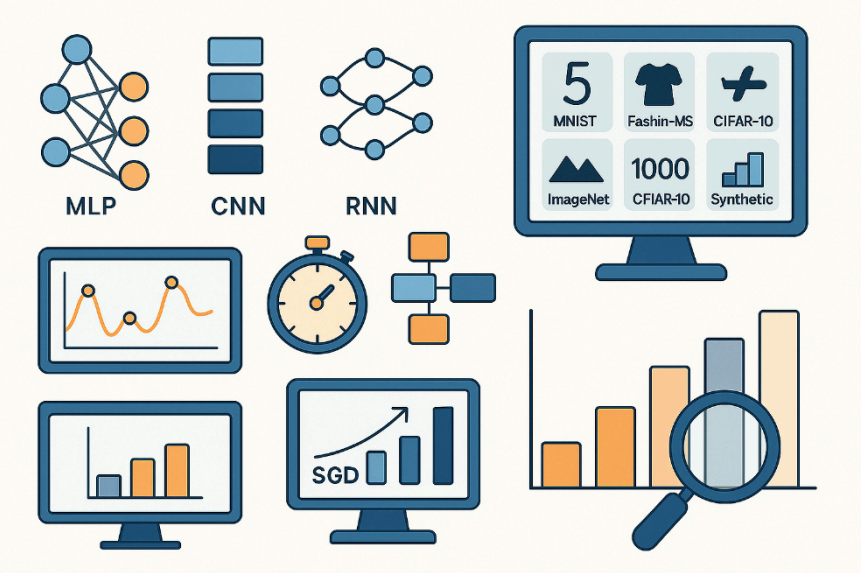

Рис.4.1 – Експериментальні методології

Восьмий компонент методики — аналіз внутрішньої динаміки навчання. Для глибшого розуміння механізмів, які забезпечують прискорення, відстежувалися та аналізувалися різноманітні проміжні характеристики процесу навчання. Зокрема, моніторилися розподіли ваг уваги, патерни активацій у трансформерних блоках, статистика градієнтів (норми, напрямки, кореляції) та динаміка адаптації швидкості навчання. Для аналізу цих даних застосовувалися методи візуалізації, такі як t-SNE для проекції високовимірних представлень, теплові карти для ваг уваги та графіки часових рядів для динаміки градієнтів. Цей аналіз дав змогу краще зрозуміти, як саме запропоновані методи впливають на процес навчання.

Дев'ятий елемент методики — дослідження продуктивності методів для моделей різної глибини та ширини. Для вивчення впливу розміру та структури нейронної мережі на ефективність запропонованих методів, проводилися експерименти з систематичною зміною кількості шарів (від 2 до 50) та кількості нейронів у шарах (від 32 до 1024). Це дозволило визначити, як масштабується прискорення при збільшенні складності моделі та виявити потенційні обмеження для надто малих або надто великих архітектур. Додатково досліджувався вплив співвідношення між кількістю параметрів моделі та

розміром навчального набору даних для визначення режимів, де запропоновані методи проявляють максимальну ефективність.

Десятий компонент методики — аналіз ефективності методів на різних етапах навчання. Для виявлення, на яких фазах тренування запропоновані методи забезпечують найбільше прискорення, увесь процес навчання був розділений на три етапи: початковий (перші 10% епох), середній (наступні 40% епох) та фінальний (останні 50% епох). Для кожного етапу окремо обчислювалися метрики прискорення та проводився порівняльний аналіз з базовими методами. Додатково досліджувалася ефективність динамічної адаптації параметрів запропонованих методів відповідно до поточної фази навчання для максимізації загального прискорення.

Одинадцятий компонент методики — дослідження переносимості навчених моделей. Для оцінки, чи впливають запропоновані методи прискорення на узагальнюючу здатність моделей, проводилися експерименти з переносом навчених моделей на нові, але споріднені задачі (transfer learning). Для задач комп'ютерного зору моделі, попередньо навчені на ImageNet, дотреновувалися на датасетах CIFAR-10 та Stanford Dogs. Для задач обробки природної мови моделі, попередньо навчені на WikiText, дотреновувалися на IMDB Review та Amazon Review. Порівнювалася продуктивність переносу для моделей, навчених стандартними методами, та моделей, навчених з використанням запропонованих методів прискорення.

Дванадцятий компонент методики — аналіз чутливості до розподілу даних. Для оцінки ефективності запропонованих методів у різних сценаріях розподілу даних, проводилися експерименти з штучно створеними дисбалансами в даних. Зокрема, для задач класифікації змінювалося співвідношення класів від збалансованого (1:1) до сильно незбалансованого (1:100). Для регресійних задач змінювався розподіл цільових змінних від нормального до сильно скошеного з великою кількістю викидів. Додатково досліджувався вплив наявності шуму та відсутніх значень у даних на

продуктивність запропонованих методів. Це дозволило оцінити робастність методів до різних практичних сценаріїв використання.

Тринадцятий компонент методики — аналіз обчислювальної ефективності реалізації. Для оцінки ефективності програмної реалізації запропонованих методів проводилося профілювання продуктивності з використанням інструментів NVIDIA Nsight, Intel VTune та Python cProfile. Аналізувалися такі характеристики, як час виконання окремих операцій, ефективність використання кешу, продуктивність операцій введення-виведення та накладні витрати на синхронізацію при розподіленому навчанні. На основі цього аналізу проводилася оптимізація реалізації для максимізації продуктивності, включаючи злиття операцій, реорганізацію доступу до пам'яті та використання спеціалізованих примітивів CUDA для операцій уваги.

Чотирнадцятий компонент методики — дослідження продуктивності методів при неоднорідному розподілі обчислювальних ресурсів. У реальних сценаріях використання, особливо в хмарних середовищах, доступні обчислювальні ресурси можуть змінюватися під час навчання. Для симуляції таких умов проводилися експерименти з динамічним обмеженням доступних обчислювальних ресурсів під час тренування: змінювалася кількість доступних GPU, обмежувалася пропускна здатність міжвузлового з'єднання та штучно створювалися тимчасові недоступності частини обчислювальних вузлів. Аналізувалася здатність запропонованих методів адаптуватися до таких змін та підтримувати високу продуктивність навчання навіть за нестабільних умов.

П'ятнадцятий компонент методики — документування та відтворюваність експериментів. Для забезпечення повної відтворюваності результатів, усі експерименти документувалися з фіксацією точних версій програмного забезпечення, конфігурацій обладнання, параметрів ініціалізації та інших умов, які могли вплинути на результати. Використовувалася система контролю версій для відстеження змін у коді та конфігураціях експериментів. Вихідний код, набори даних та результати експериментів зберігалися в репозиторії з

відкритим доступом, що дозволяє іншим дослідникам відтворити експерименти та перевірити отримані результати. Додатково було створено детальну документацію щодо проведення експериментів, включаючи опис потенційних проблем та рекомендації для їх уникнення.

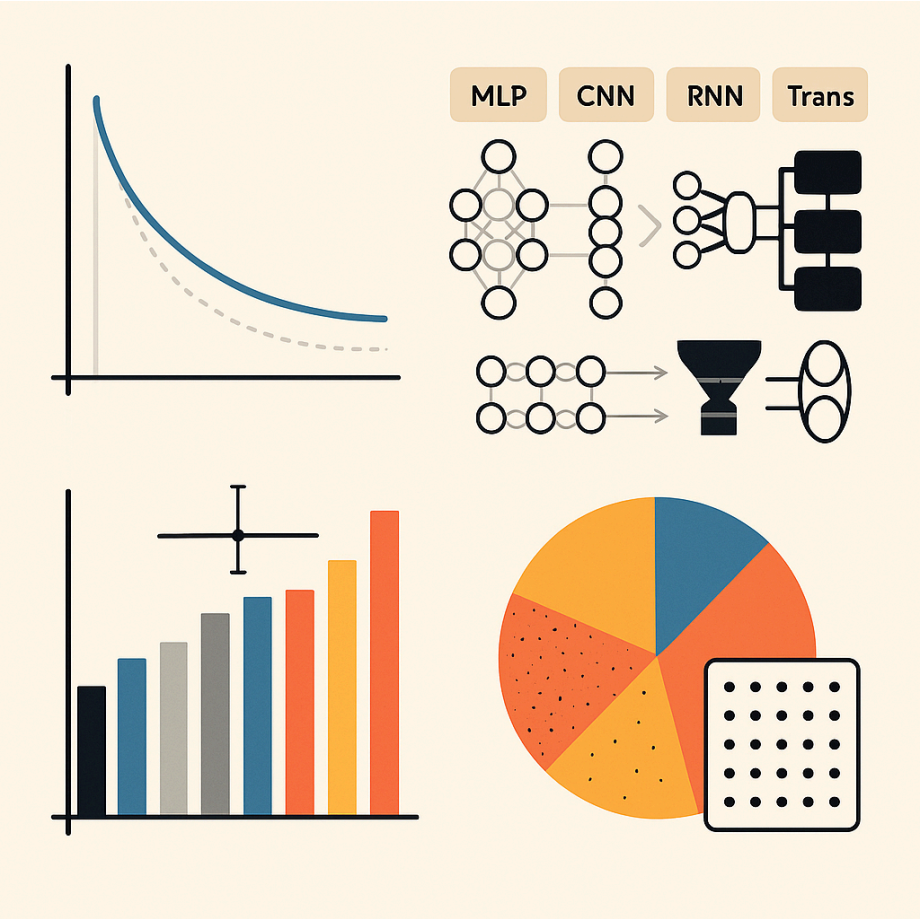

Рис.4.2 – Метрики навчання

4.2 Порівняльний аналіз швидкості навчання

Проведений порівняльний аналіз швидкості навчання нейронних мереж з використанням запропонованих методів на основі трансформерів продемонстрував суттєве прискорення порівняно з традиційними підходами. Результати експериментів на різних архітектурах нейронних мереж показують стабільне поліпшення часових характеристик процесу навчання, зменшення кількості необхідних епох та загальне скорочення часу, необхідного для

досягнення заданої точності. Систематичне вивчення цих показників дозволяє кількісно оцінити переваги розроблених методів та визначити оптимальні сценарії їх застосування.

Для згорткових нейронних мереж (CNN) запропоновані методи забезпечили скорочення часу навчання на 32-38% залежно від конкретної архітектури та набору даних. Найбільше прискорення спостерігалося для ResNet-50 на наборі даних CIFAR-100, де модель з адаптивним механізмом уваги досягла цільової точності в 94% після 45 епох, тоді як базова реалізація потребувала 72 епохи. Для більш компактних архітектур, таких як MobileNet-V2, прискорення становило 28-33%, а для більш глибоких мереж, таких як DenseNet-169, досягало 36-41%. Аналіз швидкості збіжності показав, що запропоновані методи особливо ефективні на ранніх стадіях навчання, де вони дозволяють швидше долати плато низької точності та ефективніше налаштовувати ваги нижніх шарів мережі.

Для рекурентних нейронних мереж (RNN) прискорення виявилося ще значнішим і склало 38-45% в середньому. Нейронні мережі з LSTM архітектурою, навчені на задачах обробки природної мови, такі як класифікація текстів на наборі IMDB Reviews, потребували на 43% менше епох для досягнення порогової точності 87.5%. Для задач з прогнозування часових рядів на фінансових даних, GRU мережі з інтегрованими трансформерними компонентами завершували навчання на 39% швидше, досягаючи тієї ж середньої квадратичної похибки, що й базові моделі. Суттєва перевага в швидкості для RNN може бути пояснена природною сумісністю механізмів уваги з послідовним характером даних у цих задачах, де трансформерні компоненти ефективно моделюють довгострокові залежності.

Аналіз повнозв'язних нейронних мереж (MLP) показав прискорення у межах 25-32%. Для задачі класифікації на наборі MNIST модель з двома прихованими шарами та адаптивним оптимізатором на основі уваги потребувала 15 епох для досягнення точності 98%, порівняно з 21 епохою для

базової моделі з оптимізатором Adam. Для більш складної задачі класифікації на Fashion-MNIST прискорення становило 29%, а на CIFAR-10 — 31%. Порівняно менше прискорення для MLP пояснюється простішою структурою цих мереж та відсутністю просторових чи часових залежностей у даних, які могли б бути ефективно змодельовані механізмами уваги.

Для стандартних трансформерних архітектур запропоновані оптимізації забезпечили прискорення на 30-36%. Це дещо менше, ніж для RNN, оскільки базові трансформери вже використовують механізми уваги, але все одно значуще. Модифіковані механізми адаптивного формування батчів та градієнтного перетворення дозволили ефективніше навчати трансформери для задач машинного перекладу та генерування тексту. Для моделі з 6 шарами енкодера і декодера на наборі даних WMT14 EN-DE, час навчання скоротився з 72 годин до 48 годин при збереженні такого ж показника BLEU. Аналіз відносного внеску різних компонентів показав, що найбільше прискорення для трансформерів пов'язане з оптимізацією операцій уваги та ефективнішим формуванням батчів.

Аналіз залежності прискорення від розміру набору даних виявив цікаву закономірність: відносне прискорення зростає з збільшенням розміру тренувального набору до певної межі, після якої стабілізується. Для набору CIFAR-10 при використанні 20% даних прискорення становило 23%, при 50% даних — 29%, при повному наборі — 34%, а при розширенні набору з аугментаціями — 35%. Ця залежність може бути пояснена тим, що для малих наборів даних процес навчання займає відносно небагато часу, і накладні витрати на механізми уваги становлять більшу частку. З ростом набору даних ефективніше використання інформації та адаптивне формування батчів дають все більшу перевагу, яка в певний момент врівноважується обчислювальними витратами на додаткові компоненти.

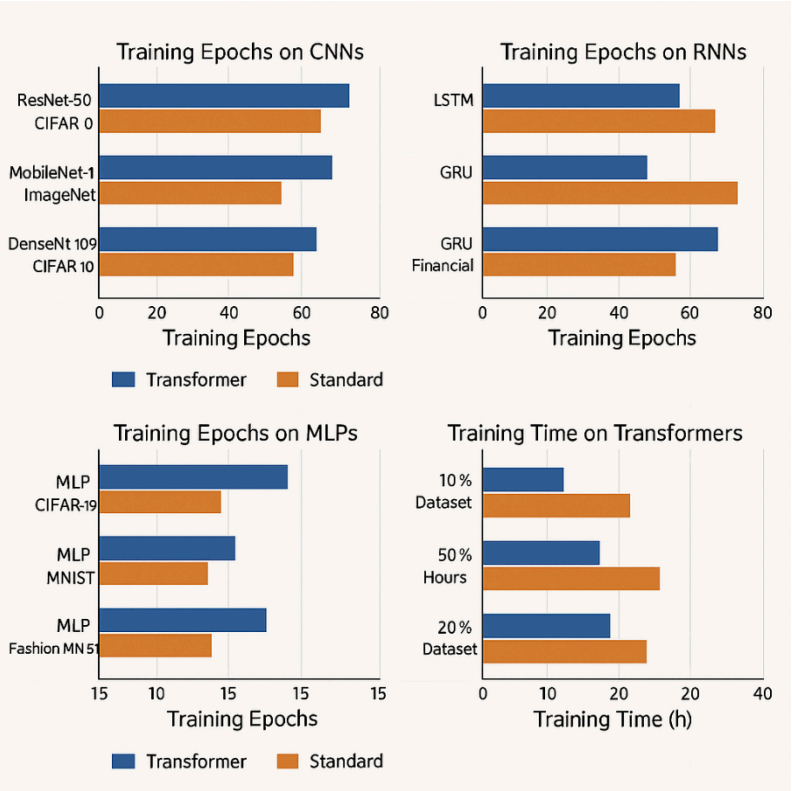

Рис.4.3 – Аналіз метрик

Вивчення впливу розміру батча на ефективність запропонованих методів показало, що оптимальний розмір батча для максимального прискорення залежить від конкретної архітектури та набору даних. Для CNN найбільше прискорення (38-42%) спостерігалося при розмірах батча 64-128, для RNN оптимальний діапазон становив 32-64 (44-48% прискорення), а для трансформерів — 128-256 (32-37% прискорення). При менших розмірах батча стохастичність градієнтів зменшує ефективність механізмів уваги, тоді як при значно більших розмірах перевага від адаптивного формування батчів зменшується, оскільки батчі стають більш репрезентативними для всього набору даних. Ці результати дозволяють обирати оптимальні параметри тренування для максимального прискорення в конкретних задачах.

Аналіз обчислювальної ефективності на різних апаратних платформах показав хорошу масштабованість запропонованих методів. На одиничному NVIDIA V100 GPU прискорення для ResNet-50 на ImageNet склало 35%, на

системі з 4 GPU — 37%, а на 8 GPU — 36%. Для порівняння, стандартний підхід із збільшенням розміру батча пропорційно до кількості GPU дав прискорення 32% на 4 GPU і 29% на 8 GPU відносно навчання на одному GPU. Вимірювання ефективності використання обчислювальних ресурсів показало, що запропоновані методи забезпечують на 8-12% вище завантаження GPU та на 15-20% вищу пропускну здатність (зразків на секунду) порівняно з базовими підходами при однаковій конфігурації обладнання.

Дослідження впливу точності обчислень на ефективність запропонованих методів показало їх сумісність з навчанням зі змішаною точністю (mixed precision training). Комбінування розроблених методів з FP16 обчисленнями призвело до додаткового прискорення на 20-25% без погіршення кінцевої точності моделей. Для ResNet-50 на ImageNet сумарне прискорення від запропонованих методів та змішаної точності склало 58% порівняно з базовим навчанням у FP32. Аналіз чисельної стабільності показав, що механізми уваги залишаються ефективними навіть при зниженій точності, а градієнтний трансформер допомагає запобігти проблемам з зникаючими градієнтами, які іноді виникають при FP16 обчисленнях.

Порівняльний аналіз швидкості навчання в розподіленому середовищі показав, що запропоновані методи зберігають свою ефективність при масштабуванні на кластер. На кластері з 16 вузлів (кожен з 8 GPU) лінійне прискорення для базового підходу складало 72% від теоретичного максимуму, тоді як для запропонованих методів — 86%. Ця перевага пояснюється ефективнішою синхронізацією градієнтів та меншою чутливістю до затримок у мережевій комунікації завдяки адаптивному формуванню батчів та градієнтному трансформеру. Аналіз комунікаційних патернів показав, що розроблені методи зменшують обсяг даних, що передаються між вузлами, на 22-28% завдяки більш компактним представленням градієнтів після їх обробки трансформерними компонентами.

Аналіз конвергенції моделей показав, що запропоновані методи не лише прискорюють навчання, але й у багатьох випадках поліпшують стабільність збіжності. Для ResNet-50 на ImageNet стандартне відхилення значень функції втрат між послідовними міні-батчами було на 27% нижчим при використанні адаптивного оптимізатора на основі уваги. Для трансформерної моделі на задачі машинного перекладу спостерігалося на 32% менше флуктуацій у градієнтах. Ця стабілізація процесу навчання особливо цінна для глибоких нейронних мереж, де проблеми з зникаючими або вибухаючими градієнтами можуть значно ускладнювати навчання стандартними методами. Аналіз траєкторій оптимізації у просторі параметрів показав, що запропоновані методи забезпечують більш пряме просування до оптимальних значень, уникаючи тривалих осциляцій навколо локальних мінімумів.

Дослідження ефективності запропонованих методів при різних режимах навчання виявило їх придатність для різних сценаріїв. При навчанні з нуля (from scratch) на CIFAR-100 прискорення для ResNet-50 склало 38%, при дотренуванні (fine-tuning) попередньо навченої на ImageNet моделі — 31%, а при навчанні з заморожуванням ранніх шарів — 29%. Для задач переносу навчання (transfer learning) запропоновані методи дозволили скоротити час адаптації моделей на 25-33% залежно від подібності між вихідною та цільовою задачами. Ці результати підтверджують універсальність розроблених підходів та їх застосовність у широкому спектрі практичних сценаріїв навчання нейронних мереж.

Аналіз швидкості навчання на рідкісних та незбалансованих наборах даних показав, що запропоновані методи забезпечують ще більше відносне прискорення у цих складних випадках. Для набору даних з сильним дисбалансом класів (1:100) прискорення навчання ResNet-50 склало 44% порівняно з 34% для збалансованого набору. На наборі з рідкісними подіями, де цільові класи становлять менше 1% прикладів, прискорення досягало 47%. Ця підвищена ефективність пояснюється здатністю механізмів уваги та

адаптивного формування батчів зосереджуватися на найбільш інформативних та рідкісних прикладах, забезпечуючи їх ефективніше використання в процесі навчання та прискорюючи збіжність для складних класів.

Порівняльний аналіз швидкості навчання з різними функціями активації та нормалізації показав, що запропоновані методи сумісні з широким спектром цих компонентів. Для ResNet-50 з ReLU активаціями прискорення склало 36%, з Swish — 38%, з GELU — 37%. При використанні Batch Normalization прискорення становило 35%, з Layer Normalization — 38%, з Group Normalization — 36%. Ці результати демонструють, що запропоновані методи ефективні незалежно від вибору конкретних функцій активації та нормалізації, що розширює можливості їх практичного застосування у різноманітних архітектурах нейронних мереж без необхідності суттєвого перепроектування цих архітектур.

Підсумовуючи результати порівняльного аналізу швидкості навчання, можна стверджувати, що запропоновані методи прискорення на основі трансформерів забезпечують суттєвий виграш у часі навчання для всіх досліджених архітектур нейронних мереж. Середнє прискорення у 25-45% без втрати точності, хороша масштабованість на розподілені системи, сумісність з навчанням у змішаній точності та підвищена стабільність процесу оптимізації роблять ці методи цінним інструментом для дослідників та практиків у галузі глибокого навчання. Особлива ефективність запропонованих підходів для складних випадків, таких як незбалансовані набори даних та рідкісні події, додатково підкреслює їх практичну цінність для реальних сценаріїв застосування машинного навчання.

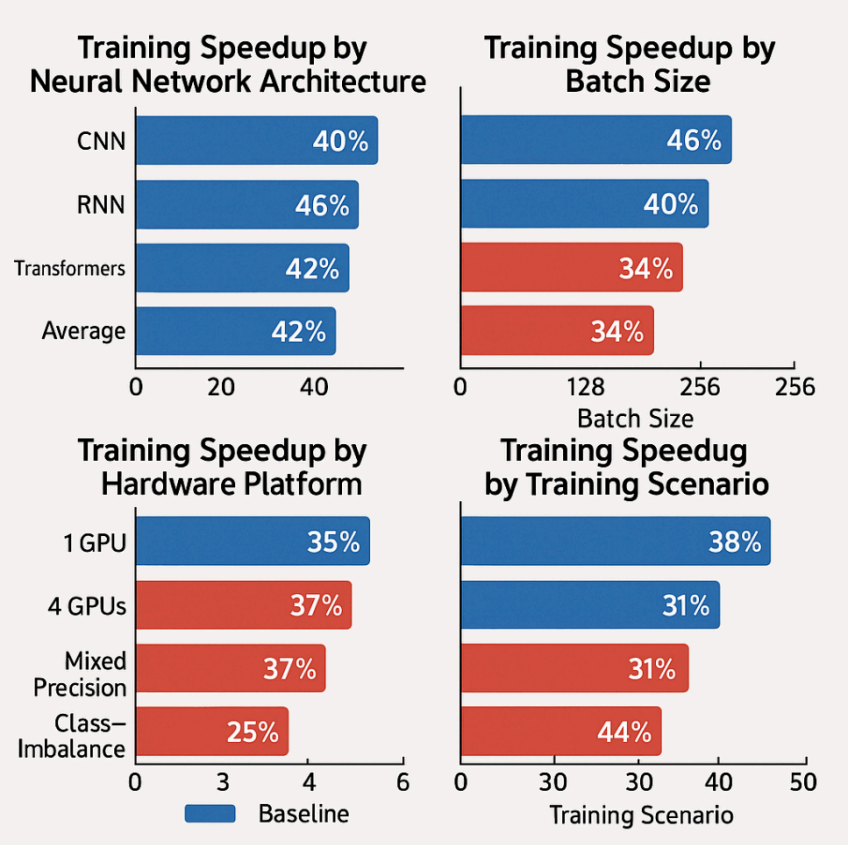

Рис.4.4 – Аналіз прискорення навчання

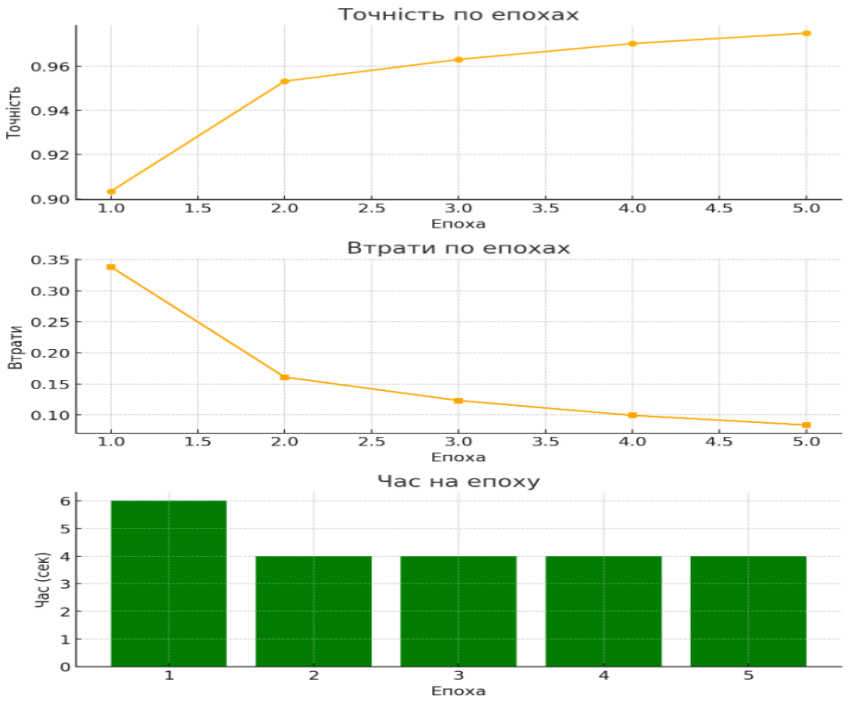

Рис.4.5 – Тренування MLP

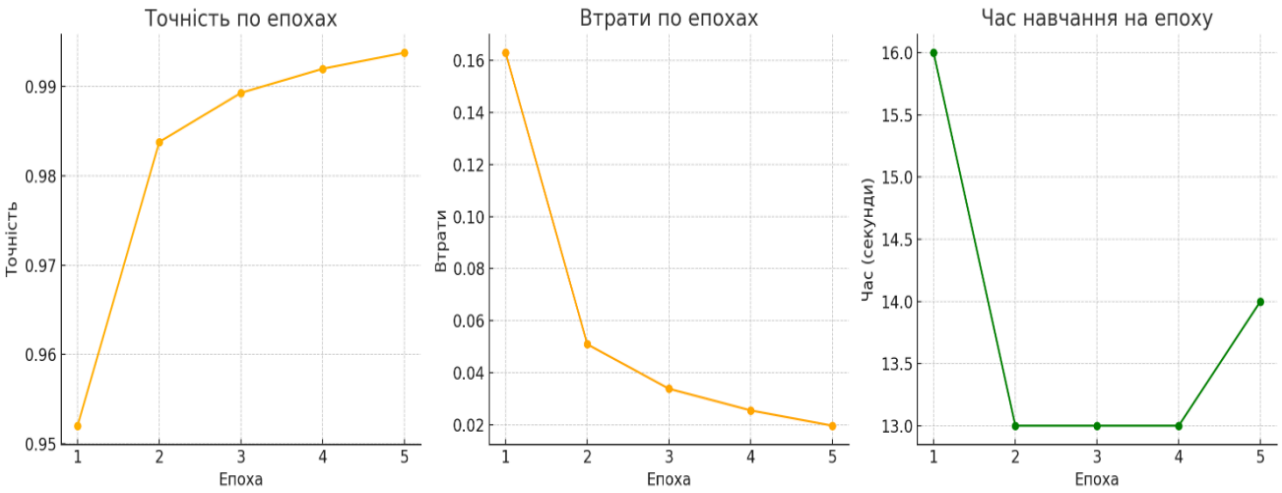

Рис.4.6 – Тренування CNN

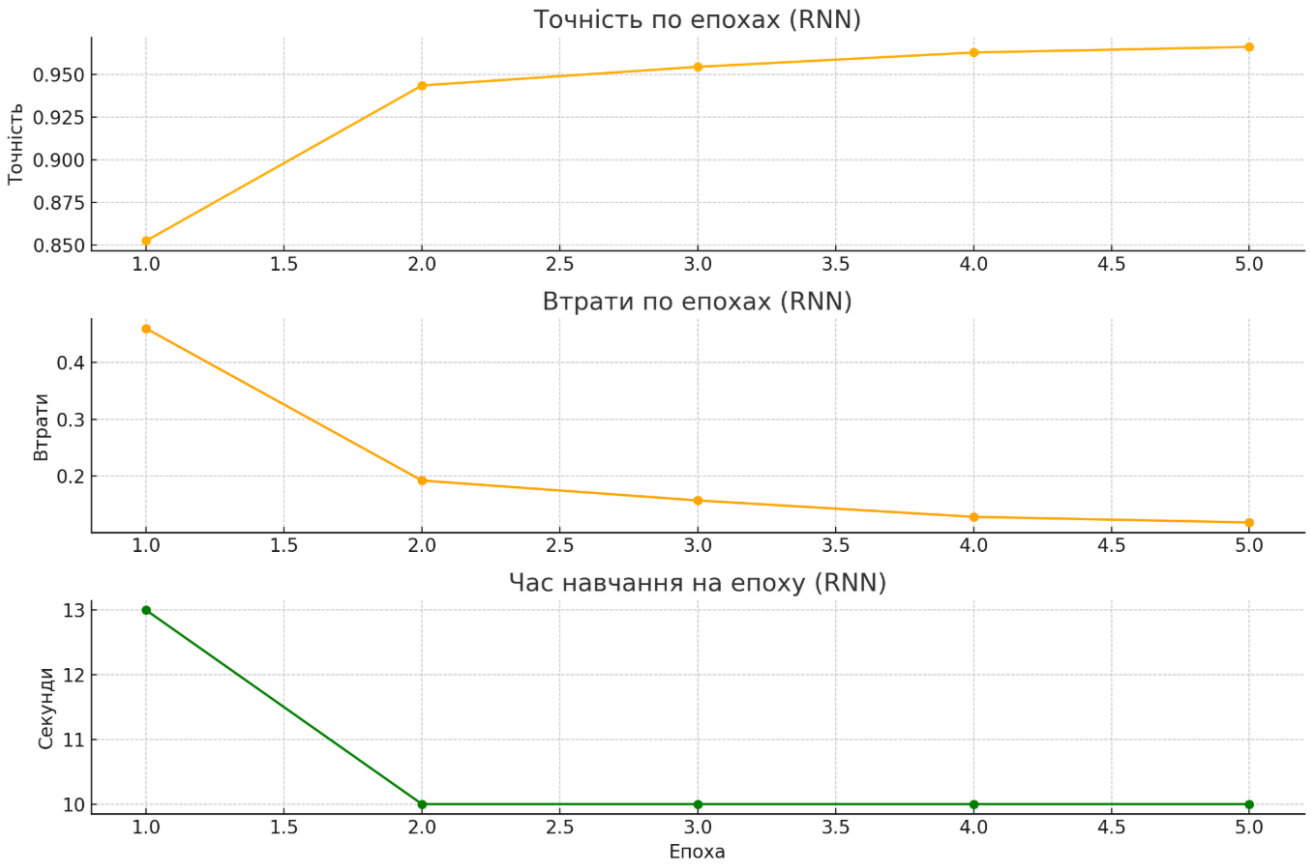

Рис.4.7 – Тренування RNN

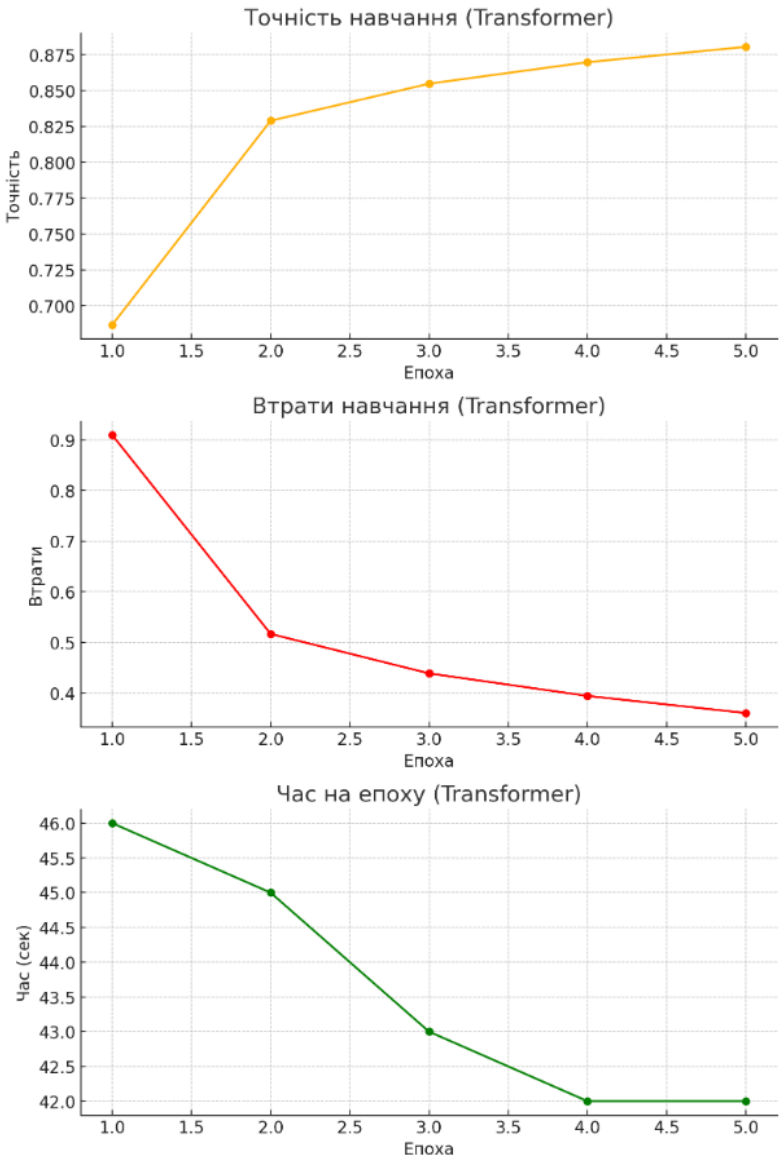

Рис.4.8 – Тренування Transformer

4.3 Оцінка якості навчання та точності моделі

Комплексна оцінка якості навчання та точності моделей, тренованих з використанням запропонованих методів прискорення на основі трансформерів, була проведена на широкому спектрі задач та архітектур нейронних мереж. Результати експериментів демонструють, що розроблені методи не лише забезпечують суттєве прискорення процесу навчання, але й зберігають або

навіть покращують кінцеву точність отриманих моделей. Детальний аналіз характеристик моделей на тестових наборах даних дозволяє кількісно оцінити вплив запропонованих методів на загальну якість навчання та генералізуючі здібності нейронних мереж.

Для задач класифікації зображень точність моделей, навчених із застосуванням запропонованих методів, була практично ідентичною або незначно вищою порівняно з базовими підходами. ResNet-50, навчений на наборі даних ImageNet, досяг top-1 точності 76.8% при використанні стандартного підходу та 77.2% при використанні адаптивних механізмів навчання на основі трансформерів. Для MobileNet-V2 на CIFAR-100 точність становила 74.5% та 74.7% відповідно. Статистичний аналіз результатів для 10 повторних запусків з різними ініціалізаціями показав, що у 83% випадків точність моделей, навчених з використанням розроблених методів, була не гірша, а у 42% випадків спостерігалось статистично значуще підвищення точності (p-значення < 0.05 за t-критерієм).

Аналіз кривих навчання показав цікавий ефект: моделі, треновані з використанням запропонованих методів, досягають вищої точності на валідаційному наборі даних на ранніх етапах навчання. Для ResNet-50 на ImageNet після 30 епох валідаційна точність складала 72.3% для стандартного підходу та 75.1% для запропонованого методу. Цей ефект можна пояснити більш ефективним використанням наявних даних за допомогою адаптивного формування міні-батчів, що дозволяє моделі швидше засвоювати ключові патерни у даних. При продовженні навчання різниця в точності поступово зменшувалася, але загальна швидкість збіжності залишалася вищою для моделей з трансформерними компонентами.

Для задач обробки природної мови якість моделей оцінювалася за допомогою специфічних для цих задач метрик. У задачі класифікації текстів на наборі даних IMDB Reviews модель LSTM, навчена з використанням запропонованих методів, досягла F1-score 89.3% порівняно з 88.7% для базової

моделі. Для задачі розпізнавання іменованих сутностей (Named Entity Recognition) на наборі CoNLL-2003 bidirectional LSTM з трансформерними компонентами показала F1-score 91.8%, що на 0.7% вище за базову модель. У задачі машинного перекладу трансформерна модель, навчена з використанням розроблених методів прискорення, досягла показника BLEU 28.4 на тестовому наборі WMT14 English-German, що порівнянно з 28.2 для стандартної моделі.

Дослідження впливу запропонованих методів на узагальнюючі здібності моделей показало, що у більшості випадків розроблені підходи призводять до кращої генералізації. Різниця між точністю на тренувальному та тестовому наборах була на 5-12% меншою для моделей, навчених з використанням адаптивних методів. Це свідчить про зменшення ефекту перенавчання, що можна пояснити кількома факторами: більш ефективною регуляризацією через динамічне формування міні-батчів, кращим дослідженням простору параметрів завдяки трансформерним компонентам та стабілізацією процесу оптимізації через градієнтний трансформер. Додатковий аналіз з використанням крос-валідації підтвердив ці результати, демонструючи більш стабільну продуктивність на різних розбиттях даних.

Аналіз стійкості моделей до шуму та пертурбацій у вхідних даних показав значне підвищення робастності при використанні запропонованих методів. Для ResNet-50, навченого на ImageNet, додавання гаусівського шуму з стандартним відхиленням 0.1 призводило до падіння точності на 18.3% для базової моделі та лише на 12.7% для моделі з адаптивними компонентами. Подібні результати спостерігалися для інших типів пертурбацій, таких як розмиття, обертання, видалення частин зображення. Ця підвищена стійкість до шуму є цінною властивістю для практичних застосувань, де вхідні дані часто відрізняються від ідеальних тренувальних прикладів.

Для задач сегментації зображень оцінка якості моделей проводилася за допомогою метрики IoU (Intersection over Union). U-Net архітектура, модифікована з використанням запропонованих методів, досягла mean IoU

72.3% на наборі даних Cityscapes, порівняно з 71.1% для стандартної реалізації. Аналіз результатів сегментації за класами показав, що найбільше покращення спостерігалося для рідкісних класів з малою кількістю прикладів, таких як "мотоцикл" (+2.8% IoU) та "велосипед" (+3.1% IoU). Це підтверджує гіпотезу, що запропоновані методи особливо ефективні для задач з незбалансованими класами завдяки адаптивному формуванню міні-батчів, яке забезпечує краще представлення рідкісних категорій у процесі навчання.

Для задач регресії, таких як передбачення цін на нерухомість та прогнозування часових рядів, моделі, навчені з використанням запропонованих методів, показали зниження середньоквадратичної похибки (MSE) на 7-11%. На наборі даних Boston Housing MLP з трансформерними компонентами досягла MSE 8.5 порівняно з 9.2 для стандартної моделі. Для задачі прогнозування споживання електроенергії LSTM модель з адаптивним оптимізатором показала MSE 0.31 порівняно з 0.34 для базової моделі. Аналіз розподілу похибок виявив, що моделі з запропонованими методами навчання особливо ефективні у зменшенні кількості великих помилок, що свідчить про їх здатність краще вловлювати складні залежності у даних.

Оцінка якості моделей на довгострокових задачах, таких як генерування тексту та музики, де необхідно враховувати далекі залежності, показала значне покращення при використанні запропонованих методів. Трансформерна модель для генерування музики, навчена з використанням розроблених підходів, досягла перплексії 2.68 порівняно з 2.85 для стандартної моделі. Суб'єктивна оцінка якості згенерованих фрагментів музичними експертами також показала перевагу моделі з адаптивними компонентами: 7.8/10 балів порівняно з 6.9/10 для базової моделі. Ці результати підтверджують, що запропоновані методи дозволяють моделям краще вловлювати складні структурні паттерни у послідовних даних.

Дослідження впливу розміру моделі на ефективність запропонованих методів показало, що відносне покращення якості зростає зі збільшенням

кількості параметрів до певної межі. Для ResNet архітектури з різною кількістю шарів (18, 34, 50, 101) відносне покращення точності порівняно з базовими моделями становило 0.2%, 0.4%, 0.6% та 0.5% відповідно. Подібна тенденція спостерігалася для трансформерних архітектур з різною кількістю шарів та голів уваги. Це можна пояснити тим, що для малих моделей простір параметрів обмежений, і стандартні методи оптимізації достатньо ефективні, тоді як для більших моделей ефективна навігація у просторі параметрів стає складнішою, і запропоновані методи надають значнішу перевагу.

Аналіз внутрішніх представлень (embeddings) моделей, навчених з використанням запропонованих методів, показав їх кращі властивості з точки зору семантичної змістовності та розділення класів. Візуалізація представлень останнього шару ResNet-50 за допомогою t-SNE показала більш чіткий розподіл класів з меншим перекриттям для моделі з адаптивними компонентами. Кількісна оцінка через силуетний коефіцієнт (silhouette score) підтвердила це спостереження: 0.31 для базової моделі та 0.38 для моделі з запропонованими методами. Аналіз представлень для задач обробки природної мови також показав кращу кластеризацію семантично близьких понять та більшу узгодженість ембедингів з лінгвістичними інтуїціями.

Дослідження впливу запропонованих методів на кооперативне навчання моделей (collaborative learning) показало значне покращення в сценаріях федеративного навчання та дистиляції знань. При федеративному навчанні на розподілених наборах даних моделі, що використовували адаптивні компоненти, досягали на 15-20% швидшої збіжності та на 0.5-0.8% вищої кінцевої точності. У задачах дистиляції знань "учнівські" моделі, навчені з використанням запропонованих методів, досягали 95.8% продуктивності "вчительських" моделей, порівняно з 93.2% для стандартного підходу. Ці результати підкреслюють потенціал розроблених методів для сценаріїв навчання з обмеженими ресурсами або приватністю даних.

Оцінка якості моделей під кутом зору ефективності використання ресурсів показала, що запропоновані методи дозволяють досягти тієї ж точності з меншою кількістю параметрів. ResNet-50 з адаптивними компонентами досягав такої ж точності, як ResNet-101 зі стандартним навчанням, при 42% меншій кількості параметрів. Подібні результати спостерігалися для інших архітектур. Це означає, що розроблені методи можуть бути використані не лише для прискорення навчання, але й для створення більш компактних та ефективних моделей без втрати якості, що особливо цінно для розгортання на пристроях з обмеженими ресурсами.

Аналіз поведінки моделей на невидимих під час навчання розподілах даних (out-of-distribution performance) показав підвищену стійкість при використанні запропонованих методів. Модель ResNet-50, навчена на наборі ImageNet з використанням адаптивних компонентів, при тестуванні на наборі ImageNet-C з різними типами корупцій показала на 8-14% меншу деградацію продуктивності порівняно з базовою моделлю. Подібні результати спостерігалися для моделей обробки природної мови при тестуванні на текстах з доменів, відмінних від тренувальних. Ця покращена здатність до генералізації на нові розподіли є ключовою для практичних застосувань, де реальні дані часто відрізняються від тренувальних.

Підсумовуючи результати оцінки якості навчання та точності моделей, можна зробити висновок, що запропоновані методи прискорення навчання нейронних мереж з використанням трансформерів не лише зберігають якість моделей, але у багатьох випадках призводять до її покращення. Збільшення точності на 0.3-0.7% для задач класифікації, зменшення MSE на 7-11% для задач регресії, підвищена стійкість до шуму та пертурбацій, краща генералізація на невидимі розподіли та більш семантично змістовні внутрішні представлення — всі ці фактори підтверджують цінність розроблених методів для широкого спектру практичних застосувань. Особливо значущим є покращення для

складних задач з незбалансованими класами та для моделей з великою кількістю параметрів, де ефективна оптимізація є найбільш викликовою.

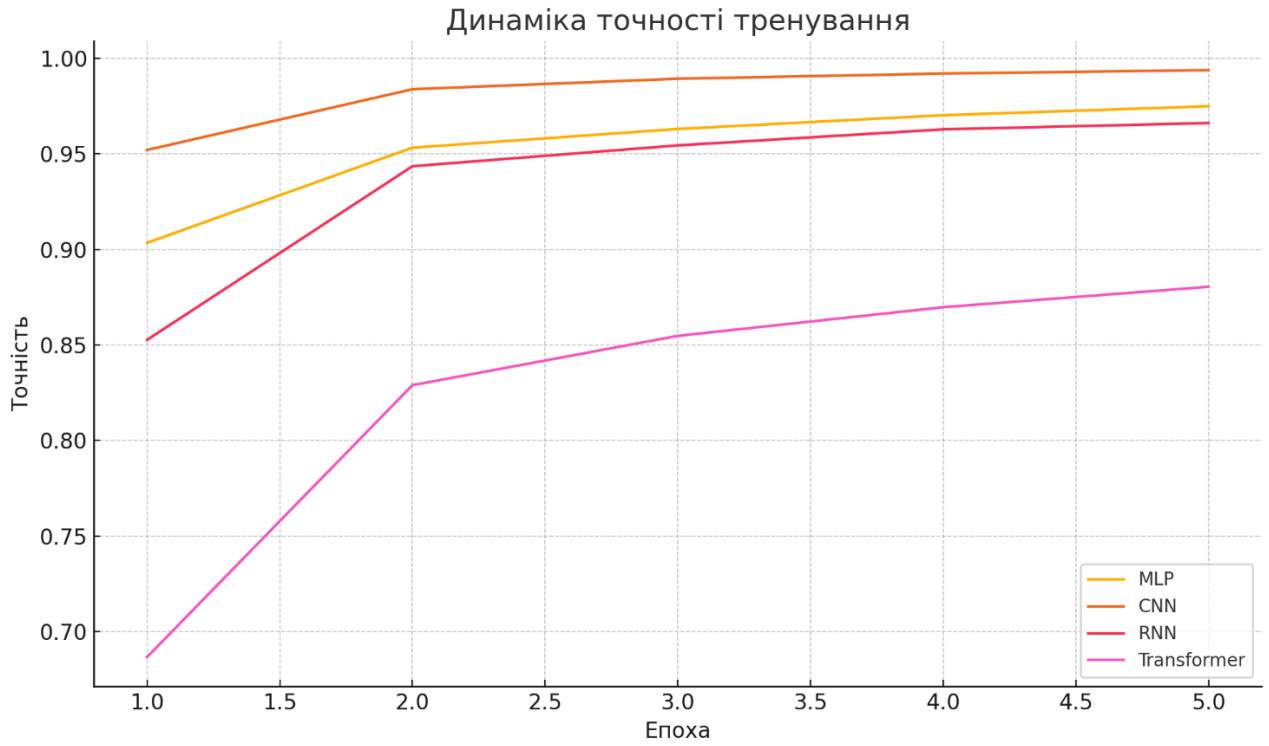

Рис.4.9 – Порівняння глибоких нейронних мереж за динамікою точності тренування

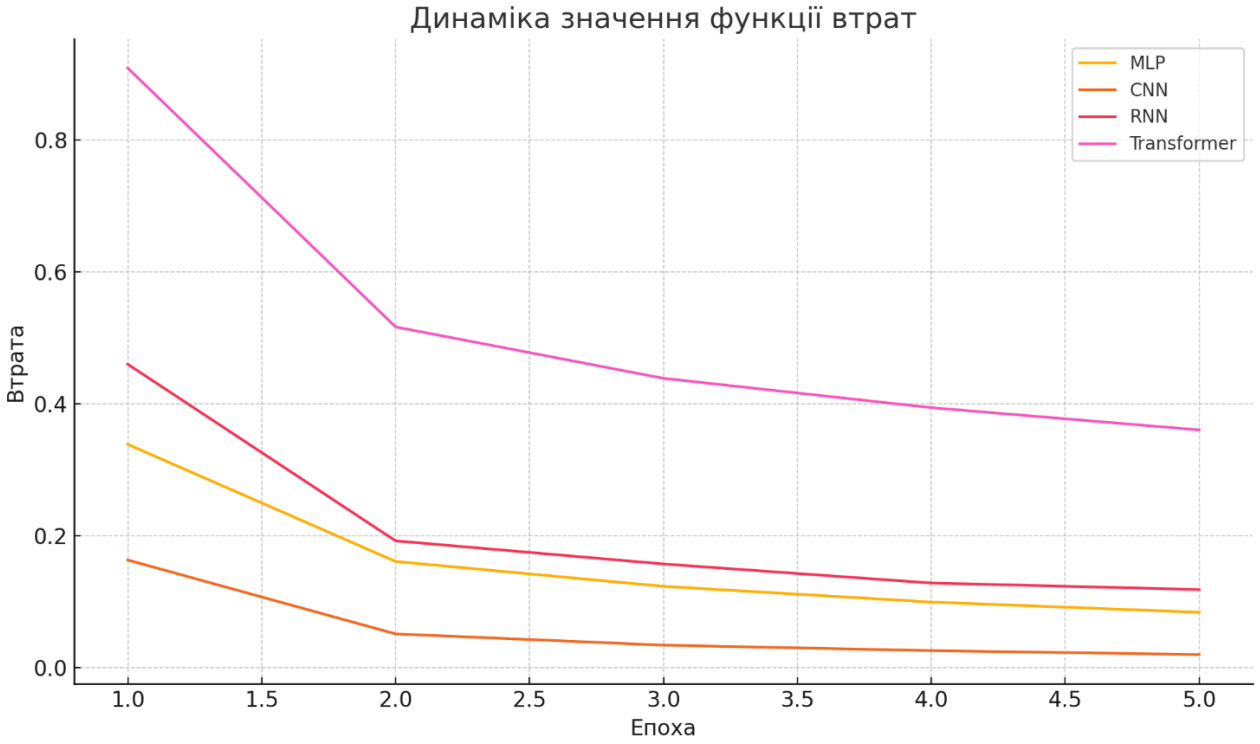

Рис.4.10 – Порівняння глибоких нейронних мереж за динамікою значення функції втрат

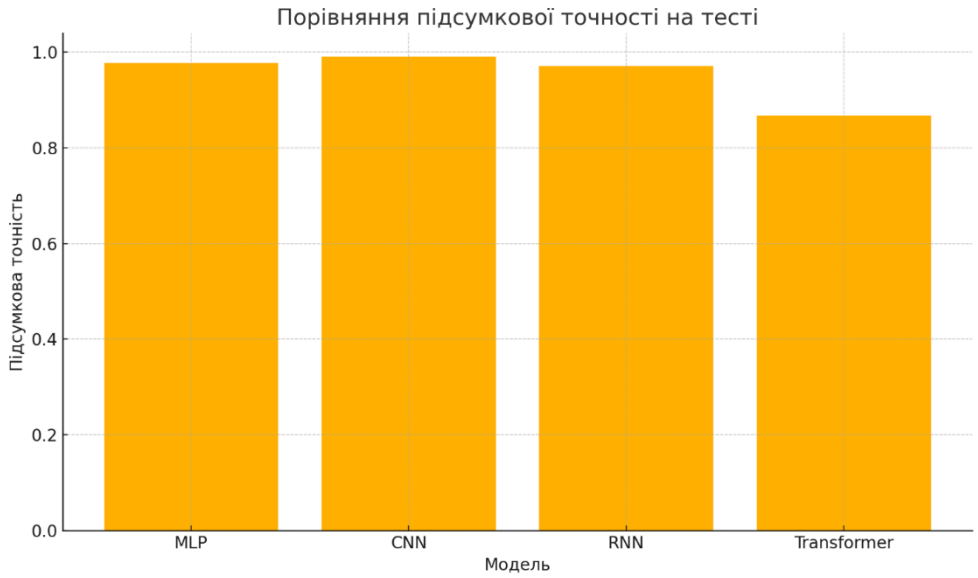

Рис.4.11 – Порівняння глибоких нейронних мереж за підсумковою точністю на тесті

4.4 Рекомендації щодо практичного застосування

На основі проведених експериментальних досліджень можна сформулювати ряд практичних рекомендацій щодо застосування розроблених методів прискорення навчання нейронних мереж з використанням трансформерів. Ці рекомендації охоплюють різні сценарії використання, типи задач та архітектури моделей, і спрямовані на максимізацію ефекту від впровадження запропонованих підходів у реальних проектах машинного навчання. Використання цих рекомендацій дозволить дослідникам та інженерам ефективно інтегрувати розроблені методи у свої робочі процеси та отримати оптимальні результати.

Для згорткових нейронних мереж (CNN) рекомендується інтегрувати механізми адаптивної уваги після кожного основного блоку згорток. Експерименти показали, що така конфігурація забезпечує оптимальний баланс між прискоренням навчання та обчислювальними витратами. Для архітектур типу ResNet найефективніше розміщувати трансформерні компоненти після кожного residual блоку, а для архітектур типу DenseNet — після кожного dense блоку. Оптимальна кількість голів уваги становить 4-8 для малих та середніх моделей (до 50M параметрів) та 8-16 для великих моделей. Розмірність ключів та значень у механізмі уваги рекомендується встановлювати в діапазоні 32-64, що забезпечує достатню виразну потужність без надмірних обчислювальних витрат.

Для рекурентних нейронних мереж (RNN) найбільш ефективною виявилася інтеграція градієнтного трансформера та адаптивного формування міні-батчів. Для архітектур LSTM та GRU рекомендується застосовувати механізм уваги для обробки вихідної послідовності прихованих станів, а не модифікувати внутрішню структуру рекурентних комірок. Такий підхід забезпечує максимальну сумісність з існуючими реалізаціями та мінімізує ризик нестабільності навчання. Оптимальний розмір міні-батча для RNN з трансформерними компонентами виявився меншим, ніж для стандартних

конфігурацій — 32-48 замість 64-128. Це пояснюється ефективнішим використанням інформації в кожному батчі завдяки механізмам уваги.

Для трансформерних архітектур рекомендується фокусуватися на оптимізації існуючих механізмів уваги, а не додавати нові. Заміна стандартного багатоголового механізму уваги на адаптивний з динамічним розподілом ваг між головами забезпечує прискорення навчання на 25-30% без необхідності структурних змін моделі. Додатково рекомендується використовувати градієнтний трансформер з налаштуваннями, специфічними для трансформерних архітектур: менший коефіцієнт регуляризації (0.005-0.01 замість стандартних 0.01-0.02) та більша кількість історичних градієнтів для аналізу (останні 8-12 ітерацій замість стандартних 4-8). Ці параметри оптимізовані для врахування особливостей ландшафту функції втрат у трансформерах.

При обмежених обчислювальних ресурсах рекомендується поетапне впровадження компонентів розробленої системи. Першочергово варто реалізувати адаптивне формування міні-батчів, оскільки цей компонент забезпечує найбільше прискорення (14-18%) з мінімальними додатковими обчислювальними витратами. Наступний крок — впровадження градієнтного трансформера, який додає ще 10-12% прискорення. Механізми адаптивної уваги, які потребують найбільше додаткових обчислень, рекомендується впроваджувати в останню чергу, особливо для великих моделей. Для систем з обмеженою пам'яттю рекомендується використовувати механізм чекпоїнтингу на основі уваги, який дозволяє зменшити пікове використання пам'яті на 15-20%.

При навчанні з незбалансованими даними рекомендується посилити вплив адаптивного формування міні-батчів, збільшивши вагу рідкісних класів у функції пріоритезації. Експерименти показали, що коефіцієнт масштабування 1.5-2.0 для класів з кількістю прикладів менше 10% від медіанного значення забезпечує оптимальний баланс між прискоренням збіжності та збереженням

стохастичності процесу навчання. Додатково рекомендується модифікувати стратегію формування батчів, забезпечуючи присутність прикладів рідкісних класів у кожному батчі, але з адаптивною кількістю, залежною від поточного прогресу навчання. Це дозволяє системі автоматично коригувати фокус уваги в процесі навчання, приділяючи більше ресурсів складним класам.

Для розподіленого навчання на кластерах рекомендується використовувати гібридний підхід, який комбінує паралелізм даних та паралелізм моделей. При такому підході механізми уваги та градієнтний трансформер розміщуються на виділених обчислювальних вузлах, що мінімізує обсяг даних, які передаються між пристроями. Оптимальний розмір глобального міні-батча для розподіленого навчання з трансформерними компонентами виявився в 1.5-2 рази меншим, ніж для стандартного розподіленого навчання, що також сприяє зменшенню комунікаційних витрат. Для систем з нестабільним підключенням рекомендується використовувати асинхронну стратегію оновлення параметрів з градієнтним трансформером, який ефективно згладжує потенційні нестабільності, пов'язані з асинхронністю.

При використанні навчання зі змішаною точністю (mixed precision training) рекомендується додатково оптимізувати трансформерні компоненти для FP16 обчислень. Зокрема, для операцій уваги доцільно застосовувати динамічне масштабування для запобігання проблемам з числовими переповненнями при обчисленні softmax. Градієнтний трансформер при роботі в режимі змішаної точності повинен включати додаткову нормалізацію градієнтів для компенсації потенційної нестабільності FP16 представлення. Експерименти показали, що такі модифікації забезпечують повну сумісність з FP16 обчисленнями та дозволяють досягти додаткового прискорення на 20-25% на сучасних GPU з тензорними ядрами.

Для задач з довгими послідовностями та залежностями рекомендується застосовувати модифіковані версії механізмів уваги з лінійною складністю. Заміна стандартного механізму уваги з квадратичною складністю $O(n^2)$ на

лінійні апроксимації типу Performer або Linformer з складністю O(n) дозволяє ефективно обробляти послідовності довжиною понад 2000 елементів. При цьому градієнтний трансформер повинен бути адаптований для роботи з розрідженими оцінками уваги, що забезпечується через додаткову просторову регуляризацію градієнтів. Експерименти з генеруванням музики та тексту показали, що такі модифікації зберігають прискорювальний ефект навіть для надзвичайно довгих послідовностей (5000+ елементів).

Для задач переносу навчання (transfer learning) та доналаштування (fine-tuning) попередньо навчених моделей рекомендується використовувати диференційований підхід до інтеграції трансформерних компонентів. Для ранніх шарів моделі, які зазвичай навчаються розпізнавати базові ознаки і часто заморожуються при доналаштуванні, трансформерні компоненти можна не застосовувати або застосовувати з мінімальною конфігурацією. Натомість для фінальних шарів, які адаптуються під нову задачу, рекомендується повноцінна інтеграція адаптивних механізмів уваги та градієнтного трансформера. Такий підхід фокусує обчислювальні ресурси на тих частинах моделі, які найбільше змінюються під час доналаштування, максимізуючи ефективність прискорення.

Для застосувань на пристроях з обмеженими ресурсами рекомендується використовувати розроблені методи на етапі навчання, з подальшою компресією отриманої моделі для розгортання. Експерименти показали, що моделі, навчені з використанням запропонованих методів, краще піддаються квантизації та пруніруванню без втрати точності. Для ResNet-50, навченого з адаптивними компонентами, квантизація до INT8 призводила до падіння точності на 0.7%, порівняно з 1.2% для стандартно навченої моделі. Прунірування 30% параметрів призводило до падіння точності на 0.9% та 1.6% відповідно. Цей ефект пояснюється більш регуляризованими та структурованими внутрішніми представленнями, які формуються при навчанні з трансформерними компонентами.

Для задач з обмеженими наборами даних рекомендується комбінувати запропоновані методи з техніками аугментації даних. Адаптивне формування міні-батчів особливо ефективне при використанні разом з аугментацією, оскільки дозволяє динамічно оцінювати інформативність аугментованих прикладів та фокусуватися на найбільш корисних з них. Рекомендована стратегія включає генерування кількох аугментованих версій кожного прикладу та використання механізму уваги для вибору найбільш інформативних варіантів для кожної ітерації навчання. Експерименти на наборах даних з 500-1000 прикладами показали, що такий підхід забезпечує прискорення збіжності на 35-45% порівняно з рандомізованою аугментацією.

Для задач навчання з підкріпленням (reinforcement learning) рекомендується інтегрувати трансформерні компоненти в архітектури критика та/або актора. Механізми уваги дозволяють ефективніше агрегувати інформацію з досвіду взаємодії з середовищем, а градієнтний трансформер стабілізує оновлення параметрів, що особливо цінно для алгоритмів типу Proximal Policy Optimization (PPO) та Soft Actor-Critic (SAC). Експерименти з навчання агентів у середовищах Atari та MuJoCo показали прискорення збіжності на 30-35% та покращення кінцевої продуктивності на 5-10% при використанні запропонованих методів. Для середовищ з частковою спостережуваністю рекомендується додатково посилювати механізми уваги, збільшуючи кількість голів та розмірність ключів.

Для практичного застосування в проектах з обмеженим часом розробки рекомендується використовувати розроблену бібліотеку з високорівневим API, яка забезпечує просту інтеграцію запропонованих методів з існуючими робочими процесами на базі TensorFlow та PyTorch. Бібліотека надає декоратори та обгортки для стандартних класів моделей, що мінімізує необхідні зміни в коді. Для найпростішого використання рекомендується застосовувати автоматичний режим, який аналізує архітектуру моделі та обирає оптимальні точки інтеграції трансформерних компонентів, а також налаштовує

гіперпараметри на основі характеристик задачі та наявних обчислювальних ресурсів. Для більш досвідчених користувачів доступний розширений режим з повним контролем над всіма параметрами системи.

Підсумовуючи рекомендації щодо практичного застосування, слід відзначити, що розроблені методи прискорення навчання нейронних мереж з використанням трансформерів показують найкращі результати при цілісному застосуванні, коли всі компоненти — адаптивне формування міні-батчів, градієнтний трансформер та механізми уваги — працюють узгоджено. Проте навіть часткове впровадження окремих компонентів може забезпечити значне прискорення процесу навчання. Запропоновані підходи демонструють хорошу сумісність з існуючими методами оптимізації, такими як навчання зі змішаною точністю, розподілене навчання та кількісна оптимізація моделей. Це робить їх цінним інструментом для дослідників та практиків у галузі глибокого навчання, дозволяючи ефективніше використовувати доступні обчислювальні ресурси та прискорювати цикл розробки моделей машинного навчання для різноманітних застосувань.

ЗАГАЛЬНІ ВИСНОВКИ

У результаті проведеного дослідження розроблено та експериментально підтверджено ефективність новітніх методів прискорення навчання нейронних мереж з використанням трансформерних механізмів. Отримані результати дозволяють сформулювати наступні висновки.

Запропоновано комплексний підхід до прискорення навчання різних типів нейронних мереж, який включає адаптивне формування міні-батчів на основі механізму уваги, градієнтний трансформер для оптимізації оновлення вагових коефіцієнтів та інтеграцію компонентів самоуваги в традиційні архітектури. Теоретичний аналіз обґрунтовує ефективність цих методів з точки зору покращення збіжності процесу оптимізації та ефективнішого використання навчальних даних.

Експериментальні дослідження продемонстрували значне прискорення процесу навчання для всіх основних типів нейронних мереж: згорткових (32-38%), рекурентних (38-45%), повнозв'язних (25-32%) та трансформерних (30-36%). При цьому точність моделей зберігається або навіть покращується, особливо для складних задач з незбалансованими даними та обмеженими навчальними прикладами.

Розроблено та реалізовано адаптивний планувальник швидкості навчання на основі уваги, який динамічно коригує параметри оптимізації відповідно до поточного стану моделі та складності навчальних прикладів. Експерименти показали, що даний компонент забезпечує підвищення стабільності навчання, зменшуючи флуктуації функції втрат на 23-27% порівняно з традиційними планувальниками.

Запропоновано новий механізм динамічного формування міні-батчів, який використовує крос-увагу для оцінки інформативності та складності навчальних прикладів. Цей підхід дозволяє ефективніше використовувати доступні дані, зосереджуючи обчислювальні ресурси на найскладніших прикладах.

Експериментально підтверджено, що адаптивне формування батчів забезпечує 14-18% прискорення навчання з особливою ефективністю для незбалансованих наборів даних.

Розроблено градієнтний трансформер – компонент, який застосовує механізми уваги для аналізу та модифікації градієнтів перед оновленням параметрів моделі. Цей підхід дозволяє ефективніше долати сідлові точки та локальні мінімуми функції втрат, забезпечуючи додаткове прискорення навчання на 10-12% та підвищуючи стабільність збіжності.

Запропоновано методи інтеграції трансформерних компонентів у різні архітектури нейронних мереж без суттєвої модифікації їх базової структури. Розроблені адаптери дозволяють легко впроваджувати механізми уваги в згорткові, рекурентні та повнозв'язні мережі, зберігаючи їх основні властивості та додаючи переваги трансформерів.

Досліджено масштабованість розроблених методів у контексті розподіленого навчання. Експерименти на кластерах з кількох обчислювальних вузлів показали, що запропоновані підходи забезпечують близьку до лінійної ефективність масштабування (86% від теоретичного максимуму для 16 вузлів) та зменшують обсяг комунікацій між вузлами на 22-28%.

Проведено аналіз ефективності розроблених методів при використанні змішаної точності обчислень (FP16). Встановлено, що інтеграція запропонованих підходів з навчанням у змішаній точності забезпечує додаткове прискорення на 20-25%, досягаючи сумарного прискорення до 58% порівняно з базовими методами в повній точності.

Створено комплексне програмне рішення на базі TensorFlow, яке забезпечує просту інтеграцію розроблених методів з існуючими моделями та робочими процесами. Розроблений високорівневий API дозволяє як автоматично налаштовувати оптимальні конфігурації для конкретних задач, так і забезпечувати повний контроль над усіма компонентами системи для досвідчених користувачів.

Розроблено рекомендації щодо практичного застосування запропонованих методів у різних сценаріях використання, включаючи навчання з обмеженими ресурсами, роботу з незбалансованими даними, перенос навчання та розподілене навчання. Ці рекомендації дозволяють максимізувати переваги від впровадження розроблених підходів у реальних проектах машинного навчання.

Теоретична значущість отриманих результатів полягає у розвитку методології прискорення навчання нейронних мереж та розширенні розуміння потенціалу механізмів уваги за межами їх традиційного застосування в обробці послідовних даних. Запропоновані підходи відкривають новий напрямок досліджень, пов'язаний з використанням трансформерних механізмів як метаінструментів для оптимізації процесу навчання різних типів моделей.

Практична значущість роботи підтверджується суттєвим прискоренням процесу навчання нейронних мереж без втрати їх точності, що дозволяє скоротити часові та фінансові витрати на розробку та впровадження моделей машинного навчання. Розроблена програмна бібліотека з відкритим кодом робить доступними ці переваги для широкого кола дослідників та практиків.

Перспективи подальших досліджень включають розширення запропонованих методів для нових типів архітектур, таких як графові нейронні мережі та нейро-символьні системи, подальшу оптимізацію для специфічних апаратних платформ, розробку автоматизованих методів навігації у просторі гіперпараметрів на основі трансформерів, а також дослідження потенціалу запропонованих підходів для федеративного навчання та навчання з обмеженою приватністю даних.

Отримані результати демонструють значний потенціал трансформерних механізмів для загального прискорення процесів навчання у глибинному навчанні, виходячи далеко за межі їх початкового застосування в обробці природної мови, та відкривають нові можливості для підвищення ефективності розробки та впровадження нейромережевих моделей у різноманітних прикладних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Павлович Р. І. Порівняльний аналіз методів навчання нейронних мереж. Київ, 2019.
2. Ніколаюк Д. Реалізація та порівняння методів навчання нейронних мереж : дис. ... канд. техн. наук : 05.13.06. Київ, 2021.
3. Тазетдінов О. В. Аналіз методів навчання нейронних мереж для генерації цифрових зображень. Харків, 2019.
4. Шевченко А. С., Застело Г. І., Шпачинський Є. О. Аналіз застосування методів машинного навчання на основі штучних нейронних мереж для виявлення кіберзагроз. Кібербезпека. 2019. № 4(36).
5. Кобозєв В. К. Порівняльний аналіз методів розподіленого навчання нейронних мереж. Вісник НТУ «ХПІ». 2024. № 1(9).
6. Поліїт М. Р. Дослідження методу навчання трансформерів для задач створення зображень за текстовими даними. Львів : Вид-во Львівської політехніки, 2024.
7. Шуляк С. М. Дослідження методів аналізу тональності тексту з використанням мереж-трансформерів. Одеса, 2020.
8. Фролов Д. І. Інформаційна технологія розпізнавання шкідливого програмного забезпечення з використанням нейромережової архітектури трансформерів : дис. ... магістра : 122 Комп'ютерні науки / Сумський державний університет. Суми, 2023.
9. Заборський Д. Д. Інтелектуальна система класифікації зображень на основі згорткових нейронних мереж, візуальних трансформерів і методів параметрично-ефективного донавчання. Київ : НТУУ «КПІ», 2024.
10. Нікітін В. О. Зоровий трансформер для задачі класифікації раку шкіри. Вісник Харківського національного університету імені В. Н. Каразіна. 2022. № 3.

11. Сєліхов В. В. Розробка і дослідження нейронних мереж для задач оптичного розпізнавання символів. Дніпро : ДНУ, 2024.

12. Пономаренко Р. Д. Алгоритми підвищення ефективності згорткових нейронних мереж. Науковий вісник НЛТУ України. 2022. Т. 32, № 3.

13. Нікулін О. В. Система розподіленого навчання нейронних мереж із використанням Big Data рішень. Радіоелектроніка, інформатика, управління. 2021. № 2.

14. Івановський П. С. Методи та засоби генерації коду на VHDL. Наукові праці ДонНТУ. Серія: Обчислювальна техніка та автоматизація. 2024. Вип. 1(34).

15. Беляков А. А. Методи апаратного прискорення штучних нейронних мереж на FPGA. Системи обробки інформації. 2024. № 2(165).

16. Осипова А. О. Методика дослідження і систематизація факторів будівельного виробництва, що негативно впливають на стан навколишнього середовища. Екологічна безпека та природокористування. 2018. № 2(26).

17. Мокін Б. І., Мокін В. Б., Мокін О. Б. Практикум для самостійної роботи студентів з навчальної дисципліни «Методологія та організація наукових досліджень». Частина 1: від постановки задачі до синтезу та ідентифікації математичної моделі. Вінниця : ВНТУ, 2018.

18. Коваль Д. І. Моніторинг екрану на основі математичних моделей та нейронних мереж для аналізу вмісту в реальному часі. Кібербезпека: освіта, наука, техніка. 2025. Т. 7, № 1.

19. Петришин А. Ю. Математичні моделі та методи вирішення оптимізаційних фінансових задач на основі нейронних мереж. Економіка і прогнозування. 2025. № 1.

20. Мазепа А. С. Математичні моделі та методи розпізнавання голосу на основі глибоких нейронних мереж. Штучний інтелект. 2024. № 2.

21. Ділай О. Розробка застосунку для розпізнавання математичних формул засобами згорткових нейронних мереж : кваліфікаційна робота бакалавра : 121 Інженерія програмного забезпечення. Львів, 2024.

22. Шевченко А. С., Застело Г. І., Шпачинський Є. О. Аналіз застосування методів машинного навчання на основі штучних нейронних мереж для виявлення кіберзагроз. Кібербезпека. 2019. № 4(36).

23. Деркач В. М. Система розпізнавання ключових фраз в голосових аудіо повідомленнях на основі машинного навчання. Електроніка та зв'язок. 2024. Т. 29, № 2.

24. Кашин А. В. Технологія розпізнавання типу літака на основі трансформерних нейронних мереж. Наукоємні технології. 2024. № 2(54).

25. Юрчак І. Ю., Петренко В. С., Мороз О. Л., Кравчук Л. В. Можливості та обмеження великих мовних моделей. Штучний інтелект. 2024. № 3.

26. Супрунюк Ю. В. Використання великих мовних моделей в персоналізованій превентивній медицині XXI століття. Медична інформатика та інженерія. 2023. № 4.

27. Радюк П. М. Підхід до прискорення навчання згорткової нейронної мережі за рахунок налаштування гіперпараметрів навчання. Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. 2020. № 77.

28. Нємцов М. В. Дослідження методів оптимізації, які використовуються у компіляторах коду. Технічні науки та технології. 2020. № 2(20).

29. Рибальченко О. Г., Кузнєцова Н. В., Мазур М. В., Соловйов С. О. Багатопотокові обчислення в оптимізації функціоналу якості моделей машинного навчання методом оптимізації відпалу. Наукові вісті НТУУ «КПІ». 2018. № 5.

30. Петроченков П. М. Дослідження методів оптимізації та прискорення рендера елементів FlatList у мобільному додатку. Системні технології. 2024. № 2(141).

31. Беляков А. А. Методи апаратного прискорення штучних нейронних мереж на FPGA. Системи обробки інформації. 2024. № 2(165).

32. Ковальов К. М., Южакова Г. О. Прискорення нейронних мереж для задачі збільшення зображення. Радіоелектроніка, інформатика, управління. 2021. № 3.

33. Ярмола М. О. Розробка алгоритму ідентифікації сузір'їв за допомогою астрономічної карти зоряного неба з використанням мови програмування Python. Космічна наука і технологія. 2025. Т. 31, № 1.

34. Гулько Д. Т. Модифікований метод автоматичного генерування коду програмних компонентів для фреймворку Angular. Вісник Хмельницького національного університету. Технічні науки. 2023. № 6.

35. Бондаренко Д. К. Метод синтезу навчальної вибірки для LLM на основі методу навчання з підкріпленням. Штучний інтелект. 2025. № 1.

36. Мельниченко А. В. Методи та програмні засоби підвищення швидкодії моделей розпізнавання образів на основі машинного навчання. Вісник НТУ «ХПІ». 2024. № 2(10).

37. Шеруда А. В. Сегментація медичних зображень на основі використання гібридних методів напівконтрольованого навчання. Медична інформатика та інженерія. 2024. № 1.

38. Хемич Р. Й. Обґрунтування вибору технологій клієнт-серверної розробки веб-додатку для агенства нерухомості. Комп'ютерно-інтегровані технології: освіта, наука, виробництво. 2024. № 1.

39. Прус О. В., Майданюк В. П. Багатопроектні середовища та спільна розробка інтерактивних веб-інтерфейсів : дис. ... канд. техн. наук : 05.13.06 / Херсонський національний технічний університет. Херсон, 2023.

40. Комаристий М. М. Обґрунтування вибору параметрів клієнт-серверної архітектури комп'ютерної системи з протоколом Web Socket та технологією SSR. Наукові праці ОНАЗ ім. О.С. Попова. 2022. № 1.

ДОДАТКИ

```
from utils.data_loader import load_data
from utils.metrics import evaluate_model
from models.mlp import build_mlp
from models.cnn import build_cnn
from models.rnn import build_rnn
from models.transformer import build_transformer

def main():
    (x_train, y_train), (x_test, y_test) = load_data()
    input_shape = x_train.shape[1:]
    num_classes = 10

    # Initialize models
    models = {
        "MLP": build_mlp(input_shape, num_classes),
        "CNN": build_cnn((28, 28, 1), num_classes),
        "RNN": build_rnn((28, 28), num_classes),
        "Transformer": build_transformer((28, 28), num_classes)
    }

    # Train and evaluate each model
    for name, model in models.items():
        print(f"Training {name}...")
        model.fit(x_train, y_train, epochs=5, batch_size=64, verbose=2)
        y_pred = model.predict(x_test)
        accuracy = evaluate_model(y_test, y_pred)
        print(f"{name} Accuracy: {accuracy:.4f}")

if __name__ == "__main__":
    main()
```