Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра комп'ютерних систем та робототехніки

До захисту допущено
Кафедрою комп'ютерних систем та робототехніки
протокол № __ від __ грудня 2025р.

завідувач кафедри _____ Максим ХРУСЛОВ
(підпис)

«___»_____ 2025 р.

Кваліфікаційна робота

здобувача другого (магістерського) рівня вищої освіти

«ФРАКТАЛЬНО-АДАПТИВНІ МЕТОДИ ОПТИМІЗАЦІЇ ГЛИБОКИХ НЕЙРОМЕРЕЖ У МЕДИЧНІЙ ДІАГНОСТИЦІ»

Спеціальність ***174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка***
Освітня програма ***Комп'ютеризовані системи управління та автоматика***

Виконавець _____ **Дмитро СУДАКОВ**
(підпис)

Науковий керівник _____ **Сергій ШМАТКОВ**
(підпис)

Харків – 2025

АНОТАЦІЯ

Пояснювальна записка до магістерської атестаційної роботи складається зі вступу, чотирьох розділів, висновків, списку використаних джерел і одного додатку. Загальний обсяг роботи складає 74 сторінок, із яких 59 сторінок основної частини з 11 рисунками, 3 таблицями, списком використаних джерел із 14 найменувань та чотирма додатками.

Метою кваліфікаційної роботи є підвищення валідаційної точності та стабільності процесу навчання нейромереж шляхом розроблення нового адаптивного оптимізатора, який поєднує принципи класичного алгоритму Adam із фрактальним підходом до багатомасштабного згладжування градієнтів. Досягнення цієї мети передбачає створення методу, здатного забезпечити стійку збіжність навіть за наявності шуму та нерівномірних даних та покращити точність моделей у медичних задачах класифікації.

Об'єкт дослідження: процес оптимізації навчання глибоких нейронних мереж.

Предмет дослідження: фрактальні методи оптимізації процесу навчання нейронних мереж на задачах медичної діагностики, зокрема застосування фрактальної самоподібності до процесу оновлення параметрів.

Проблема, яка вирішується в кваліфікаційній роботі, полягає у підвищенні валідаційної точності та стабільності процесу навчання глибоких нейронних мереж. Досягнення цієї мети передбачає створення методу, здатного забезпечити стійку збіжність за наявності шуму та нерівномірних даних та покращити точність моделей у медичних задачах класифікації.

Область застосування: автоматизація медичної діагностики, зокрема аналіз знімків зрізів тканин у задачах мультикласифікації. Розроблений алгоритм оптимізації може бути використаний у всіх типах медичних закладів.

Ключові слова: фрактальний аналіз, нейромережі, медична діагностика, оптимізаційний алгоритм, машинне навчання.

ABSTRACT

An explanatory note to the master's qualification thesis consists of an introduction, four chapters, conclusions, a list of references, and one appendix. The total volume of the work is 74 pages, including 59 pages of the main part with 11 figures, 3 tables, a list of 14 references, and four appendixes.

The purpose of the qualification work is to improve the validation accuracy and stability of the deep neural network training process by developing a new adaptive optimizer that combines the principles of the classical Adam algorithm with a fractal-based approach to multiscale gradient smoothing. Achieving this goal involves creating a method capable of ensuring stable convergence even in the presence of noise and non-uniform data, as well as improving the accuracy and generalization ability of models in medical classification tasks.

The object of research: the optimization process of deep neural network training.

The subject of research: fractal optimization methods for neural network training in medical diagnostic tasks, in particular, the formation and combination of multilevel gradient moments, as well as the application of fractal self-similarity to the model parameter update process.

The problem solved in the qualification work is enhancing the validation accuracy and stability of deep neural network training. The proposed method aims to ensure robust convergence even under noisy and heterogeneous data conditions and to improve both accuracy medical classification problems.

Application area: automation of medical diagnostics, particularly the analysis of tissue slice images in multiclass classification tasks. The developed optimization algorithm can be applied in all types of medical.

Keywords: fractal analysis, neural networks, medical diagnostics, optimization algorithm, machine learning.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ

КТ – Комп’ютерна томографія

МРТ – Магнітно-резонансна томографія

ООП – Об'єктно-орієнтоване програмування

ШНМ – Штучна нейронна мережа

Adam – Adaptive Moment Estimation – Адаптивна оцінка моментів

AdamW – Adaptive Moment Estimation with Weight Decay – Адаптивна оцінка моментів з регуляризацією ваг

AdaBelief – Adaptive Belief Optimizer – Адаптивний оптимізатор на основі віри

AdaBound – Adaptive Bound Optimizer – Адаптивний обмежений оптимізатор

AdaGrad – Adaptive Gradient Algorithm – Адаптивний алгоритм градієнта

AMSGrad – Adaptive Moment Estimation with Maximum – Адаптивна оцінка моментів з максимальним значенням моменту

API – Application Programming Interface – Інтерфейс прикладного програмування

CNN – Convolutional Neural Network – Згорткова нейронна мережа

FPGA – Field Programmable Gate Array – Програмована вентильна матриця

GDPR – General Data Protection Regulation – Загальний регламент захисту даних (ЄС)

HIPAA – Health Insurance Portability and Accountability Act – Закон про конфіденційність медичних даних (США)

LR – Learning Rate – Швидкість навчання

MedMNIST – Medical Modified National Institute of Standards and Technology – Медична Модифікована база даних Національного інституту стандартів і технологій

ML – Machine Learning – Машинне навчання

MNIST – Modified National Institute of Standards and Technology – Модифікована база даних Національного інституту стандартів і технологій

PathMNIST – Pathology Medical Modified National Institute of Standards and Technology – Модифікована база даних патологій Національного інституту стандартів і технологій

ReLU – Rectified Linear Unit – Виправлена лінійна функція активації

ResNet – Residual Network – Резидуальна мережа

RGB – Red, Green, Blue – Червоний, Зелений, Синій (колірна модель)

RMSProp – Root Mean Square Propagation – Метод середньоквадратичного поширення

SGD – Stochastic Gradient Descent – Стохастичний градієнтний спуск

STT – Speech-to-Text – Перетворення мовлення в текст

TPU – Tensor Processing Unit – Тензорний процесор

Train/val – Training / Validation – Навчання / Валідація

TTS – Text-to-Speech – Перетворення тексту в мовлення

ВСТУП

Стрімкий розвиток сучасних нейромереж впливає на всі сфери людського життя, зокрема, однім з найважливіших напрямів машинного навчання є медицина. За допомогою машинного навчання обробляють рентгенівські знімки, результати МРТ, потокові дані внутрішнього тиску та сердечного ритму, дані попереднього стану пацієнта, та багато інших.

Незважаючи на те, що сучасні неймережі вже вийшли на той рівень, коли вони в змозі самостійно ставити прості діагнози, їх використання ще обмежене ресорсоємністю та нестачею даних для їх навчання, із-за чого більшість з них не можуть вийти на рівень точності, достатній для їх автономної роботи [1]. Окрім того, що наразі існує нестача даних для навчання, ще й існує проблематика того, що медицинські датасети дуже неоднорідні та зашумлені, тобто, наприклад, на рентгенівськіх знімках це можуть бути артефакти плівки, різна тональність та контрастність знімків, що ускладнює процес машинного навчання створюючи неоднорідність градієнтів та природні шуми.

Саме при дефіциті якісних, рівномірних даних для навчання, ключову роль відіграють оптимізаційні алгоритми машинного навчання. Оптимізатори відіграють фундаментальну роль у процесі корекції вагових коефіцієнтів, які визначають здатність моделі до генералізації, збіжності/точності та опору перенавчанню. Впродовж останніх десятиліть було розроблено чимало методів оптимізації, від класичного стохастичного градієнтного спуску (SGD) до складних адаптивних схем на кшталт найпопулярнішого оптимізаційного алгоритму методом адаптивної оцінки моменту Adam. Водночас, більшість популярних оптимізаторів демонструють обмежену стійкість до шумів, локальних градієнтних мінімумів і плато. У зв’язку з цим актуальним є пошук нових підходів, які дозволяють ефективніше аналізувати та використовувати інформацію з неоднорідних та зашумлених градієнтів.

Розвиток нових підходів до оптимізації навчання є необхідним для підвищення надійності систем комп’ютерної діагностики, які мають працювати у реальних клінічних умовах. Одним із перспективних напрямів є використання фрактального аналізу, який дозволяє враховувати багатомасштабну природу процесів зміни градієнтів під час навчання мережі.. Тобто, можливість відслідковування змін градієнта як на коротких, так і на довгих часових інтервалах, поєднуючи інформацію з різних шарів динаміки. Це дозволить адаптивно реагувати як на локальні флуктуації градієнтів, так і на більш стабільні тенденції, що підвищує стійкість моделі до шуму, складного рельєфу функції, та підвищить валідаційну точність і стабільність процесу навчання глибоких нейронних мереж.

РОЗДІЛ 1.
АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ОПТИМІЗАЦІЇ У МЕДИЧНІЙ ДІАГНОСТИЦІ

1.1 Огляд найпопулярніших оптимізаторів у глибинному навчанні

Процес навчання глибоких нейромерж передбачає мінімізацію втрат та максимізацію точності, що напряму залежить від обраного оптимізаційного алгоритму. Оптимізатори відіграють критичну роль у швидкості та збіжності, стабільності тренування та здатності моделі уникати локальних мінімумів або плато втрат, що також називають застряганням оптимізатору.

Оптимізаційних алгоритмів існує величезна кількість, але, можна виділити найпопулярніші, такі як SGD, RMSProp та Adam, які стали де-факто стандартами у більшості фреймворків машинного навчання [2] і входять до переліку вбудованих інструментів у цих фреймворках (до прикладу, Tensorflow, PyTorch).

Одним із базових та найстаріших методів є стохастичний градієнтний спуск (SGD), що оновлює ваги мережі на основі випадкової вибірки (міні-батчу) градієнтів. Незважаючи на свою простоту, чистий SGD часто демонструє повільну збіжність і сильну чутливість до вибору гіперпараметрів. Для покращення його властивостей був запропонований варіант його модифікації моментумом (SGD with Momentum), що вводить інерцію в оновлення градієнтів, дозволяючи уникати сильних коливань у напрямку оновлення.

Проблема коливань і нерівномірності масштабів градієнтів стимулювала розвиток адаптивних оптимізаторів. Одним із перших представників цього класу є Adagrad, який адаптує швидкість навчання для кожного параметра окремо, враховуючи історію його градієнтів. Проте його основний недолік — монотонне зменшення швидкості навчання, призвів до створення

модифікацій, зокрема RMSprop. RMSprop вирішує цю проблему шляхом експоненційного згладжування квадратів градієнтів, що дозволяє підтримувати динамічний баланс між стабільністю та чутливістю до змін.

Найбільш популярним адаптивним оптимізатором сьогодні є Adam, який поєднує ідеї моментуму та RMSprop. Він бере окремі, експоненційно згладжені оцінки першого (середнє значення) та другого (дисперсія) моментів градієнта, дозволяючи досягати швидкої збіжності на складних завданнях без ретельного налаштування гіперпараметрів. Завдяки своїй універсальності Adam став стандартом у багатьох практичних задачах — від обробки зображень до моделювання нейромереж природної мови. Водночас він не позбавлений недоліків — Adam схильний до застрягання в плато мінімумів, а також може демонструвати гіршу здатність до узагальнення порівняно з класичним SGD у деяких сценаріях [3].

З метою подолання цих обмежень наразі запропонована велика кількість модифікацій Adam, наприклад, такі як AMSGrad та AdamW. AMSGrad намагається гарантувати монотонне зменшення функції втрат, а AdamW відокремлює вагову регуляризацію від алгоритму оптимізації, що дозволяє точніше контролювати процес навчання [4].

Незважаючи на велике розмаїття існуючих рішень, вибір оптимізатора залишається емпіричним — жоден з існуючих оптимізаційних алгоритмів не гарантує стабільної переваги у всіх задачах або навіть в одних і тих же типах задач, але на різних датасетах. Це особливо помітно при роботі з шумними, незбалансованими або саме медичними даними, де класичні адаптивні підходи можуть демонструвати деградацію навчання, погану генералізацію та схильність до перенавчання. Це обмеження особливо критичне для медичних застосунків, де похибки оптимізації можуть призводити до помилкових діагностичних рішень. Саме тому наразі зростає інтерес до нових підходів, які можуть поєднувати класичні адаптивні схеми з ідеями з інших галузей,

наприклад, зі стохастичних евристик, хаотичних систем, теорії фракталів або еволюційної оптимізації.

1.2 Особливості медичних датасетів та виклики при їх обробці

Медичні дані становлять один із найбільш цінних, але водночас найскладніших типів інформації для обробки за допомогою машинного навчання. Їхня складність зумовлена як структурними, так і семантичними особливостями. По-перше, більшість медичних датасетів мають високу розмірність — це можуть бути зображення високої роздільної здатності (наприклад, КТ або МРТ), часові ряди (електрокардіограма, моніторинг життєвих показників), клінічні записи, геномні послідовності, лабораторні дані тощо [5]. Такі дані часто мають неоднорідну природу, що потребує комплексної попередньої обробки. По-друге, медичні дані майже завжди супроводжуються значним класовим дисбалансом. Наприклад, у задачах виявлення рідкісних патологій позитивні приклади (хворі пацієнти) можуть становити менше 1% від загального обсягу вибірки. Класичні оптимізатори в таких умовах схильні ігнорувати міноритарний клас, що призводить до високої загальної точності, але нульової чутливості до патологій, тобто, критично неприйнятного результату у клінічному контексті. Простіше кажучи, нейромережа може вірно ставити 99% діагнозів, у випадках коли паталогії немає, ігноруючи їх можливу наявність, так як точність все одно буде 99%.

Ще одна особливість — наявність шуму та артефактів. Медичні зображення можуть містити шуми від апаратного забезпечення, обладнання або артефакти через рух пацієнта (наприклад, рис. 1.1, де видно артефакти від руху під час МРТ) [6]. Текстові записи часто містять пропущені значення, неоднозначні формулювання або абревіатури. У часових рядах спостерігаються пропуски та нерівномірність. Такі фактори ускладнюють не тільки тренування моделей, а й попередню обробку, яка часто вимагає участі експерта-лікаря.

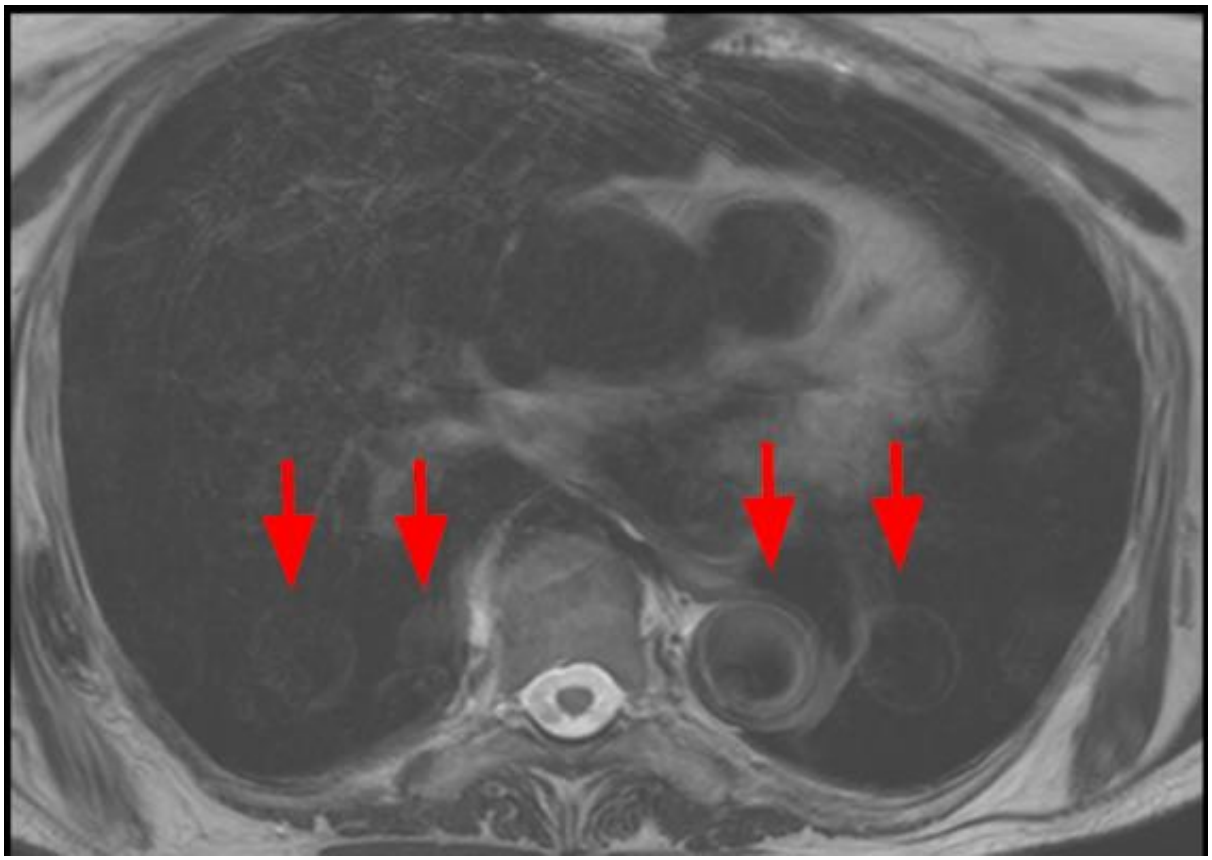


Рисунок 1.1 – Артефакт від мимовільних рухів. Аксіальний зрізок грудної клітки із численними дублюваннями аорти (стрілки).

Типи медичних даних та їхні особливості

- Медичні зображення (КТ, МРТ, рентген)
  - Високий рівень шуму: артефакти руху, металічні імпланти, низька якість сканування.
  - Неоднорідність: різна контрастність, роздільна здатність, орієнтація органів на знімках.
  - Класовий дисбаланс: рідкісні патології представлені значно менше, ніж нормальні зразки.
- Електронні медичні записи
  - Неповні дані: пропуски в анамнезі, нестандартизовані формулювання лікарів.

    - Часова залежність: дані змінюються динамічно (наприклад, показники аналізів крові).
- Геномні та біомедичні сигнали
    - Висока розмірність: тисячі ознак при малій кількості пацієнтів (проблема "прокляття розмірності").
    - Біологічний шум: індивідуальні варіації, вплив зовнішніх факторів.

Окремим викликом є невеликий обсяг якісно розмічених даних. Анотація медичних записів вимагає участі кваліфікованих фахівців, що робить процес дорогим і повільним, до того ж, ручна розмітка іноді має показники суб’єктивності, що критично впливає на майбутнє навчання машинної моделі. У результаті датасети часто мають обмежену кількість прикладів або неповні мітки. Крім того, поширеною проблемою є анонімізація та приватність: у багатьох країнах діють жорсткі закони щодо обробки персональних медичних даних (наприклад, HIPAA у США чи GDPR у ЄС), що ускладнює відкритий обмін та використання даних для досліджень. Також є ризик упередженості — датасети можуть бути зміщеними до певних етнічних груп або статі.

Ще однією проблемою є висока міжпацієнтна варіативність. Дві однакові патології можуть мати різні прояви у пацієнтів різної статі, віку, етнічного походження або при наявності супутніх захворювань. Це робить завдання генералізації моделі надзвичайно складним, і ще раз підкреслює важливість вибору стійкого до шуму оптимізатора.

Загалом, робота з медичними даними потребує особливої уваги до кожного етапу — від нормалізації та боротьби з дисбалансом, до правильного налаштування регуляризації та критеріїв зупинки. Саме в таких умовах стандартні оптимізатори (як, наприклад, Adam чи SGD) можуть демонструвати непередбачувані результати — від перенавчання на "здоровому" класі до повного застрягання на плато втрат. Це створює потребу

пошуку нових, більш адаптивних підходів до оптимізації, здатних ефективно працювати в умовах невизначеності, шуму та обмеженої кількості прикладів.

1.3 Сучасні підходи до оптимізації навчання на зашумлених даних

Оптимізація навчання нейронних мереж на зашумлених даних залишається активною дослідницькою областю, особливо у сфері медичних додатків, де помилки можуть мати реальні клінічні наслідки. Останніми роками з'явилося кілька перспективних напрямків, які намагаються подолати обмеження класичних оптимізаційних алгоритмів. Ці підходи можна умовно розділити на три категорії: адаптація існуючих оптимізаторів, використання методів згладжування градієнтів та спроби застосування принципів інших розділів математики та фізики.

Одним з найбільш поширених напрямків є модифікація стандартних оптимізаторів для роботи в умовах шуму. Наприклад, адаптивні методи типу Adam часто демонструють нестабільність на зашумлених даних через свою чутливість до коливань градієнтів. У відповідь на це дослідники запропонували такі варіанти як AdaBound, який поєднує переваги адаптивних методів з обмеженням кроку навчання, що запобігає надмірним коливанням параметрів на пізніх етапах навчання.

Альтернативний шлях полягає у використанні методів згладжування градієнтів. Техніки на кшталт стохастичного згладжування ваг (Stochastic Weight Averaging) або використання ковзного середнього для параметрів моделі демонструють хороші результати на зашумлених даних. Особливо перспективним виглядає підхід, що поєднує методи згладжування з аналізом геометрії функції втрат, дозволяючи адаптивно оновлювати рівень згладжування в залежності від інтенсивності шуму в даних.

Також, треба наголосити, що проблематику неоднорідності та шумів в даних намагаються вирішувати і за межами саме оптимізаційних алгоритмів. Наприклад, через зміни архітектури нейромережі та підходів до попередньої

обробки даних. Можна виділити такі основні прийоми боротьби проти шумів та неоднорідності за межами оптимізаційного алгоритму [7]:

- Focal Loss зменшує вагу легких прикладів і фокусується на важких або рідкісних.
- Dropout – випадкове "вимикання" нейронів під час тренування знижує залежність від окремих вхідних ознак та знижує вірогідність перенавчання.
- Stochastic Depth у ResNet – випадкове пропускання шарів під час тренування, що запобігає перенавчанню.
- Label Smoothing – замість жорстких міток (наприклад, булінових 1 і 0) використовується розподіл ймовірностей, що пом'якшує ефект хибно промаркованих даних.
- Deep Ensembles – кілька моделей навчаються незалежно одна від одної, і прогнози усереднюються. При цьому можна оцінити невпевненість моделі (наприклад, високу дисперсію між учасниками ансамблю).

Проте, незважаючи на значний прогрес, більшість сучасних методів все ще мають суттєві обмеження при роботі з реальними медичними даними. Основні проблеми пов'язані з недостатнім врахуванням специфіки медичних джерел шуму, які часто мають складну кореляційну структуру і не піддаються стандартним статистичним описам. Крім того, існуючі підходи рідко враховують когнітивні аспекти обробки медичних даних, які можуть бути критично важливими для забезпечення стійкості моделей у реальних клінічних умовах.

Ці обмеження вказують на необхідність розробки нових спеціалізованих методів оптимізації, які б враховували не лише статистичні властивості шуму, але й структурні особливості медичних даних. Перспективним напрямком є розвиток гібридних підходів, що поєднують переваги адаптивних оптимізаторів з методами нелінійної динаміки, зокрема з використанням

фрактального аналізу, що має на меті підвищення валідаційної точності та стабільності процесу навчання глибоких нейронних мереж у задачах медичної діагностики. Такі методи могли б забезпечити більш стійку роботу моделей у реальних умовах, де якість даних часто значно відрізняється від ідеалізованих умов лабораторних досліджень.

Висновки за розділом 1

Перший розділ роботи присвячений аналізу існуючих підходів до оптимізації в контексті глибокого навчання, зокрема їх застосування у медичних задачах. Було встановлено, що сучасні оптимізаційні алгоритми, такі як SGD, Adam та RMSProp, демонструють значні обмеження при роботі з реальними медичними даними. Ці обмеження пов’язані з високим рівнем шуму, нерівномірністю розподілу класів, наявністю артефактів, а також малою кількістю якісно розмічених даних. Медичні набори даних характеризуються високою міжпацієнтною варіативністю, що ускладнює процес генералізації моделей і вимагає спеціалізованих підходів до попередньої обробки та навчання.

Особливу увагу приділено тому, що класичні адаптивні оптимізатори, такі як Adam, часто демонструють нестабільність на зашумлених даних, схильні до перенавчання та застрягання в локальних мінімумах. Це обумовлено їхньою чутливістю до флуктуацій градієнтів та нездатністю враховувати мультимасштабну природу медичних даних. У відповідь на ці виклики було розглянуто сучасні підходи до покращення стійкості оптимізаторів, зокрема методи згладжування градієнтів, використання ансамблів моделей, а також інтеграцію ідей з нелінійної динаміки, теорії фракталів та хаотичних систем.

Враховуючи складність та специфіку медичних даних, було підкреслено необхідність розробки нових оптимізаційних методів, які б поєднували переваги адаптивних підходів із здатністю до багатомасштабного аналізу градієнтів. Саме ця потреба лягла в основу подальшої розробки нового типу

оптимізатора, що мав би потенціал для підвищення стійкості, швидкості збіжності та якості узагальнення моделей у реальних клінічних задачах.

РОЗДІЛ 2.
РОЗРОБКА НОВОГО ОПТИМІЗАЦІЙНОГО АЛГОРИТМУ НА ОСНОВІ ФРАКТАЛЬНОЇ САМОПОДІБНОСТІ

2.1 Загальні вимоги до сучасного оптимізатора

Оптимізатор — це ключовий компонент у процесі навчання штучних нейронних мереж, який визначає, як саме будуть оновлюватися параметри моделі з метою мінімізації функції втрат. Від вибору оптимізатора значною мірою залежить не лише швидкість збіжності, але й кінцева якість моделі, її здатність до узагальнення, а також стабільність під час навчання. У контексті складних задач, таких як аналіз медичних даних, де наявний шум, нерівномірний розподіл класів або нестача прикладів, вимоги до оптимізатора зростають у кілька разів.

Сучасний оптимізатор має відповідати низці загальних вимог, без виконання яких застосування його у практичних задачах глибокого навчання є неефективним або нестабільним. По-перше, оптимізатор повинен забезпечувати стійку та швидку збіжність навіть у випадках, коли початкові ваги мережі обрані випадково або невдало. Це означає, що метод має вміти долати локальні мінімуми, плато та інші особливості втрат, характерні для високовимірних функцій, які описують простір параметрів глибокої мережі. По-друге, важливою є адаптивність до різних масштабів градієнтів. У багатьох сучасних мережах різні параметри можуть оновлюватися в різних масштабах, у той час як одна частина моделі вимагає великих кроків градієнтного спуску, інша потребує обережніших змін. Оптимізатор має вміти автоматично підлаштовуватись під ці умови, зберігаючи стабільність навчання. По-третє, оптимізатор повинен бути стійким до шуму у градієнтах, що особливо актуально при роботі з реальними, неідеальними датасетами, такими як медичні. Випадкові флуктуації у даних не повинні призводити до деградації збіжності або до розгойдування процесу навчання [8].

Крім того, ефективний оптимізатор має враховувати нерівномірність топології ландшафту втрат, тобто повинен працювати добре як у гладких, так і в «рваних» або ізольованих областях простору параметрів. Ще однією ключовою вимогою є здатність до генералізації, тобто побудови такої моделі, яка б добре працювала не тільки на тренувальних даних, а й на невідомих їй прикладах. З практичної точки зору це означає, що оптимізатор не повинен підлаштовувати модель під шум тренувального набору, а навпаки, сприяти пошуку більш гладкого і стабільного рішення.

До вимог сучасних оптимізаторів додається також обчислювальна ефективність. Оптимізатори повинні масштабуватись на великі об’єми даних, працювати швидко у батчевому режимі та бути сумісними з апаратним прискоренням. Тобто, не можна бескінечно ускладнювати архітектуру оптимізаційного алгоритму, бо збільшення складності буде означати підвищення витрат ресурсів та часу, що особливо важливо для деяких медицинських датасетів в яких є знімки високої роздільної здатності.

Таким чином, створення сучасного оптимізатора є балансуванням між низкою, іноді суперечливих вимог: швидкою адаптивною збіжністю, стійкістю до шуму, здатністю працювати в умовах складного ландшафту втрат і збереженням генералізуючих властивостей моделі та ресурсною ефективністю. Саме такі властивості покладені в основу подальшої розробки фрактального оптимізатора, що поєднує адаптивні методи (як у Adam) з нелінійною динамікою та ідеєю масштабної самоподібності, характерної для фрактальних структур.

2.2 Стохастичні оптимізаційні методи та підходи з нелінійної динаміки

У контексті оптимізаційних алгоритмів нейронних мереж стохастичні методи виявилися значно ефективнішими за класичні детерміновані алгоритми завдяки своїй здатності працювати в умовах великої розмірності, шуму та сильного перенавчання. Найбільш відомими прикладами таких

методів є SGD та його численні модифікації, зокрема Adam, RMSprop, AdaGrad та інші. Усі вони ґрунтуються на ідеї обчислення градієнта втрат не по всій вибірці, а по її випадкових міні-батчах, що зменшує обчислювальні витрати та сприяє виходу з деяких невеличких локальних мінімумів завдяки шумовому компоненту.

Проте, традиційні стохастичні методи залишаються вразливими до багатьох проблем, зокрема плато та частих або тривалих локальних мінімумів. У відповідь на це дослідники почали звертатися до методів, натхненних фізикою складних систем і нелінійною динамікою. Зокрема, з’являються підходи які використовують хаотичні системи, нелінійні карти, або принципи з теорії флуктуацій, ентропії, фрактальності та самоорганізації.

Окремий клас алгоритмів в цьому напрямку — це так звані «хаотичні оптимізатори», які використовують нестандартні траєкторії оновлення ваг (наприклад, логістичну карту) для збереження варіативності та втечі з повторюваних циклів при навчанні. У подібних підходах стохастичність стає не просто випадковістю, а керованою багатомасштабною флуктуацією, що дозволяє досліджувати простір параметрів більш різноманітними траєкторіями.

Таким чином, інтеграція елементів з нелінійної динаміки, від хаотичних карт до фрактальних механізмів, стає перспективним напрямком для вдосконалення існуючих стохастичних оптимізаторів. Це відкриває шлях до побудови нових алгоритмів, які зберігають адаптивність Adam чи SGD, але мають покращену поведінку у складних, шумних або багатомодальних ландшафтах функцій втрат. На цьому тлі постає ідея об'єднання класичного оптимізатора Adam із фрактальною структурою моментів, що буде докладніше розглянуто в наступних підрозділах.

2.3 Фрактали як засіб багатомасштабного аналізу в оптимізації

Фрактали — це математичні об’єкти, які характеризуються властивістю самоподібності, тобто, структура об’єкта повторюється на різних масштабах. Вони не лише естетично привабливі, але й мають глибокий практичний зміст у природничих науках, біології, фізиці, обробці сигналів [9]. Головна перевага фрактального підходу — це здатність виявляти закономірності та структури, приховані у складних залежностях. У контексті оптимізаційних алгоритмів фрактальний підхід пропонує унікальну можливість одночасно враховувати структуру функції втрат на різних масштабах — від глобальних особливостей до локальних деталей.

Багато даних, з якими працюють нейромережі (особливо в біомедичній сфері), мають фрактальну природу. Наприклад:

- Електроенцефалографія (ЕЕГ) та електрокардіографія (ЕКГ) – їх сигнали містять паттерни, що повторюються на різних часових масштабах. Швидкі коливання можуть відповідати миттєвим змінам (наприклад, артефактам руху), тоді як повільні — довготривалим фізіологічним станам (наприклад, стану стресу або фазі сну).
- Медичні зображення (МРТ, КТ) — мають фрактальні текстури через ієрархічну будову тканин (наприклад, судинні мережі, які повторюють свою структуру на різних масштабах).

Якщо градієнти функції втрат формуються на таких даних, то вони також успадковують мультишкальність. Традиційні оптимізатори згладжують градієнти лише в одному масштабі, що може призводити до нестабільності навчання та застряганню в локальних мінімумах.

У класичних оптимізаторах, таких як Adam чи RMSProp, використовується експоненціальне згладжування градієнтів та їх квадратів. Це згладжування передбачає один домінуючий часовий масштаб для динаміки оновлення ваг. Такий підхід може бути ефективним у задачах із регулярними

або добре структурованими даними, але виявляється менш стійким у випадках, коли дані змінюються на декількох часових, або просторових масштабах одночасно, що характерно для реальних медичних датасетів.

Фрактальні методи, на відміну від традиційних оптимізаційних алгоритмів, оперують не одним, а множиною масштабів. У контексті оптимізації це означає можливість врахування як короткострокових, так і довгострокових трендів у градієнтах. Саме ця ідея покладена в основу фрактального підходу до модифікації оптимізаторів, — замість однієї експоненціальної згортки вводиться багаторівневе фрактальне згладжування, що дозволяє зберігати більше інформації про динаміку змін у параметрах моделі.

Одним із підходів до реалізації фрактального згладжування є використання декількох експоненціальних фільтрів із різними коефіцієнтами згасання або ж побудова ієрархічної системи моментів, де кожен рівень відповідає за різний масштаб змін. Такий підхід дозволяє не лише точніше відстежувати складну динаміку градієнтів, а й зменшити ризик "застрягання" оптимізатора в локальних мінімумах або на плато.

Крім того, фрактальні підходи можуть бути природно адаптивними, тобто, кожен масштаб реагує на своєму рівні чутливості, що дозволяє системі адаптуватися до змін у навчальних даних навіть при високому рівні шуму. В задачах з неповними або непослідовними медичними записами така властивість є особливо цінною.

Таким чином, використання фракталів, як інструменту багатомасштабного аналізу, відкриває нові можливості для створення більш ефективних оптимізаційних алгоритмів, особливо при роботі зі складними, зашумленими даними. Це дозволяє моделювати багатомасштабну динаміку більш природно та гнучко, ніж це роблять стандартні методи, що й було

реалізовано у розробленому в межах цієї роботи оптимізаторі *FractalMomentAdam*.

2.4 Формалізація фрактального підходу на основі алгоритму Adam

Алгоритм Adam є одним із найпопулярніших методів оптимізації в глибинному навчанні завдяки поєднанню адаптивного масштабування градієнтів (як у RMSProp) з корекцією моментів першого порядку (як у Momentum SGD). У класичній формі Adam підтримує два експоненційно згладжених моментів [10] — середнє значення градієнтів m_t і середнє квадратичне значення градієнтів v_t, які оновлюються за формулами 2.1 та 2.2, де g_t – градієнт на ітерації *t*, а $\beta_1, \beta_2 \in [0,1)$ – коефіцієнти згладжування.

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \qquad (2.1)$$

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \qquad (2.2)$$

Хоча цей підхід забезпечує стабільну і швидку збіжність у багатьох задачах, він має обмеження при роботі з даними, що мають мультишкальні властивості або високу змінність (наприклад, у нашому випадку, медичні сигнали, клінічні записи, часові ряди з патологіями тощо). Одношкальна експоненціальна фільтрація не здатна одночасно вловити як короткострокові, так і довгострокові тенденції.

Для подолання подібних обмежень було вирішено розробити модифікацію оптимізатору Adam за допомогою додавання йому механізмів фрактальної самоподібності, тобто, багатомасштабного згладжування.

Створений алгоритм *FractalMomentAdam* розширює звичайний Adam шляхом введення багатомасштабного згладжування, що відображає ключовий принцип фракталів — самоподібність на різних масштабах. Замість одного набору моментів (m_t, v_t), використовується множина моментів із різними масштабами згладжування.

- g_t — градієнт у момент часу t
- β_1, β_2 — коефіцієнти експоненційного згладжування швидких моментів
- β_{1s}, β_{2s} — коефіцієнти експоненційного згладжування повільних моментів
- $\gamma \in [0,1]$ — вага шкали при об'єднанні
- $m_t^{(f)},\ v_t^{(f)}$ — швидкі моменти першого та другого порядку
- $m_t^{(s)},\ v_t^{(s)}$ — повільні моменти
- $\hat{m}_t,\ \hat{v}_t$ — ефективні моменти після об'єднання та баіас-корекції

Кроки:

Оновлення швидких моментів (fast scale):

$$m_t^{(f)} = \beta_1 \cdot m_{t-1}^{(f)} + (1-\beta_1) \cdot g_t \qquad (2.3)$$

$$v_t^{(f)} = \beta_2 \cdot v_{t-1}^{(f)} + (1-\beta_2) \cdot g_t^2 \qquad (2.4)$$

Оновлення повільних моментів (slow scale):

$$m_t^{(s)} = \beta_{1s} \cdot m_{t-1}^{(s)} + (1-\beta_{1s}) \cdot g_t \qquad (2.5)$$

$$v_t^{(s)} = \beta_{2s} \cdot v_{t-1}^{(s)} + (1-\beta_{2s}) \cdot g_t^2 \qquad (2.6)$$

Фрактальне об'єднання (ефективні моменти):

$$m_t^{(eff)} = \gamma \cdot m_t^{(f)} + (1-\gamma) \cdot m_t^{(s)} \qquad (2.7)$$

$$v_t^{(eff)} = \gamma \cdot v_t^{(f)} + (1-\gamma) \cdot v_t^{(s)} \qquad (2.8)$$

Біас-корекція:

$$\widehat{m}_t = \frac{m_t^{(eff)}}{1-\beta_1^t}\ ,\ \ \widehat{v}_t = \frac{v_t^{(eff)}}{1-\beta_2^t} \qquad (2.9)$$

Оновлення ваг:

$$\theta_t = \theta_{t-1} - \eta \cdot \frac{\widehat{m}_t}{\sqrt{\widehat{v}_t}+\epsilon} \qquad (2.10)$$

Цей підхід реалізує багатомасштабне згладжування градієнтів через два незалежні шари моментів — швидких та повільних (fast та slow).

Ключова перевага фрактального підходу — його універсальність. На відміну від методів, що вимагають ручного підбору швидкості навчання або налаштувань адаптації, FractalMomentAdam за замовчуванням враховує різні масштаби даних. Це особливо важливо в задачах, де дані нестаціонарні (наприклад, часові ряди зі змінами статистики) або градієнти мають "важкі хвости" (великі рідкісні зміни, поширені в медичних даних, особливо за наявності рідкісних паталогій або в задачах мультикласифікації).

Фрактальний підхід у FractalMomentAdam — це не просто механічне об’єднання моментів, а відображення глибшої властивості даних і динаміки навчання. Він дозволяє алгоритму "бачити" одночасно деталі та загальну картину, що критично для складних мультимасштабних задач.

2.5 Програмна реалізація оптимізатора FractalMomentAdam

Для практичного використання фрактального підходу до оптимізації було реалізовано власний оптимізатор FractalMomentAdam, який базується на модифікації класичного алгоритму Adam із урахуванням багатомасштабного фрактального згладжування моментів. Реалізація виконана на мові Python у вигляді кастомного класу для TensorFlow/Keras, що успадковує базовий клас tf.keras.optimizers.Optimizer [11]. Це дозволяє інтегрувати його в стандартний пайплайн навчання нейронних мереж, зберігаючи при цьому гнучкість у керуванні параметрами.

Архітектура класу та ініціалізація:

Клас FractalMomentAdam ініціалізується з набором параметрів, кожен з яких відіграє визначену роль у процесі оптимізації:

- **learning_rate**: Швидкість навчання
- **beta_1/beta_2**: Коефіцієнти для швидких моментів (аналогічно класичному Adam)
- **beta_1_slow/beta_2_slow**: Коефіцієнти для повільних моментів (довший часовий горизонт)
- **gamma**: Параметр змішування між швидкими та повільними моментами

Оптимізатор реалізує дві пари словників експоненційно згладжених моментів для зберігання стану:

- швидкі моменти m_fast, v_fast (оновлюються з класичними коефіцієнтами β_1 та β_2);
- повільні моменти m_slow, v_slow (оновлюються з повільнішими коефіцієнтами β_1^{slow} та β_2^{slow}).

Ключові етапи роботи оптимізатора

1. **Ініціалізація стану (build)**:
    - Створює змінні для моментів кожного параметра моделі
    - Ініціалізує лічильник кроків _step
2. **Оновлення параметрів (update_step)**:
    - Обчислення моментів:
        - Швидкі моменти оновлюються з більшою чутливістю до локальних змін (менші beta)
        - Повільні моменти інтегрують інформацію за довший період (більші beta)

- Фрактальне змішування:
    - Ефективні моменти обчислюються як зважена сума швидких і повільних компонентів
    - Параметр gamma контролює баланс між реакцією на локальні зміни та глобальними тенденціями
- Корекція зсуву:
    - Виправлення початкового зсуву моментів (аналогічно Adam)
- Оновлення параметрів:
    - Класичне оновлення з адаптивним кроком на основі ефективних моментів

Технічні особливості реалізації

- Всі моменти зберігаються у вигляді слотів (slots) для кожного параметра моделі, як це передбачено API TensorFlow.
- Кожен параметр отримує окремі змінні для m_fast, v_fast, m_slow, v_slow.
- Для забезпечення сумісності з механізмами TensorFlow 2.x, ітерація кроків (self._step) реалізується як окрема змінна, що оголошується при кожному оновленні.
- Ініціалізація моментів відбувається у методі build, який викликається перед першим кроком оптимізації.
- Користувач може налаштувати параметри згладжування та змішування через аргументи: beta_1, beta_2, beta_1_slow, beta_2_slow, gamma, learning_rate.

Також, що до особливостей реалізації, треба наголосити про те, що одразу, «за замовчуванням», створено гнучкість налаштування, тобто реалізація дозволяє керувати двома шкалами адаптивності окремо, що надає змогу краще адаптуватися до структур шуму або динаміки навчання. А також

розширюваність: архітектура оптимізатора відповідає принципу OCP (Open-Closed Principle) з ООП, тобто дозволяє у подальшому додати нові типи фрактального згладжування (наприклад, нелінійне змішування, логарифмічну інтерполяцію, більшу кількість масштабів тощо), при цьому не змінюючи його архітектуру та типи вхідних-вихідних даних.

Таким чином, реалізований оптимізатор FractalMomentAdam поєднує ідеї адаптивної оптимізації та фрактального аналізу, що створює основу для більш стабільного і гнучкого навчання глибоких моделей, особливо в задачах з нерівномірним або шумовим ландшафтом функції втрат.

Псевдокод для можливості реалізації оптимізатору на будь-якій мові програмування (Рис. 2.1)

```
ВХІД:
  - learning_rate: швидкість навчання (за замовчуванням 1e-3)
  - beta_1, beta_2: параметри експоненційного згладжування для швидких
моментів (за замовчуванням 0.9, 0.999)
  - beta_1_slow, beta_2_slow: параметри для повільних моментів (за
замовчуванням 0.99, 0.9999)
  - gamma: коефіцієнт змішування швидких/повільних моментів (за
замовчуванням 0.3)
  - epsilon: мала константа для чисельної стабільності (за
замовчуванням 1e-7)

ІНІЦІАЛІЗАЦІЯ:
  - Ініціалізувати словники для зберігання моментів:
    * _m_fast, _v_fast - швидкі моменти
    * _m_slow, _v_slow - повільні моменти
  - Ініціалізувати лічильник кроків _step = 0

МЕТОД build(var_list):
  ДЛЯ КОЖНОЇ змінної var у var_list:
    Створити тензори для моментів (швидких та повільних) з нульовою
ініціалізацією
    Додати до відповідних словників (_m_fast, _v_fast, _m_slow,
_v_slow)

МЕТОД update_step(grad, var):
  ЯКЩО var ще не має моментів у словниках:
    Ініціалізувати моменти для var нулями

  Отримати поточні значення:
    m_f, v_f - швидкі моменти
    m_s, v_s - повільні моменти
```

```
    lr - поточну швидкість навчання
    t - поточний крок (_step + 1)

# Оновлення моментів
  Оновити швидкі моменти:
    m_fast = beta_1 * m_f + (1 - beta_1) * grad
    v_fast = beta_2 * v_f + (1 - beta_2) * grad²

  Оновити повільні моменти:
    m_slow = beta_1_slow * m_s + (1 - beta_1_slow) * grad
    v_slow = beta_2_slow * v_s + (1 - beta_2_slow) * grad²

# Фрактальне змішування моментів
  Обчислити ефективні моменти:
    m_eff = gamma * m_fast + (1 - gamma) * m_slow
    v_eff = gamma * v_fast + (1 - gamma) * v_slow

  # Корекція зміщення
  m_eff_hat = m_eff / (1 - beta_1^t)
  v_eff_hat = v_eff / (1 - beta_2^t)

  # Оновлення ваг
  var = var - lr * m_eff_hat / (sqrt(v_eff_hat) + epsilon)

# Збереження оновлених моментів
  m_f = m_fast
  v_f = v_fast
  m_s = m_slow
  v_s = v_slow

  # Оновлення кроку
  _step += 1
```

Рисунок 2.1 – псевдокод реалізації оптимізаційного алгоритму

Повний код оптимізаційного алгоритму на Python 3.10/ TensorFlow 2.19/ Keras 3.10 розміщено у репозиторії на GitHub за посиланням: https://github.com/beardedcatfox/FractalAdaptiveOptimizer

Висновки за розділом 2

Другий розділ роботи присвячений розробці та реалізації нового оптимізаційного алгоритму FractalMomentAdam, що поєднує принципи адаптивної оптимізації з ідеєю фрактальної самоподібності. Основною метою було створення методу, здатного ефективно працювати в умовах складного, шумного та неоднорідного ландшафту функції втрат, характерного для медичних даних, та підвищення валідаційної точності та стабільності процесу навчання глибоких нейронних мереж. В основу алгоритму покладено

концепцію багатомасштабного аналізу градієнтів, яка дозволяє одночасно враховувати як короткострокові флуктуації, так і довгострокові тенденції в динаміці навчання.

Формалізація підходу полягає у модифікації класичного алгоритму Adam шляхом введення двох незалежних пар моментів — швидких та повільних, що відповідають різним часовим масштабам. Їхнє подальше фрактальне змішування за допомогою параметра γ дозволяє створити ефективні моменти, які інтегрують інформацію з різних рівнів динаміки. Це забезпечує більш гнучку та стабільну реакцію на зміни градієнтів, зменшуючи схильність до застрягання в локальних мінімумах та на плато.

Програмна реалізація алгоритму була виконана у вигляді кастомного класу для TensorFlow/Keras, що забезпечує його легку інтеграцію в стандартні пайплайни навчання. Архітектура оптимізатора була розроблена з урахуванням вимог гнучкості та масштабованості, дозволяючи в подальшому розширювати кількість масштабів або вдосконалювати механізм змішування. Важливою перевагою реалізації є її обчислювальна ефективність та збереження звичного інтерфейсу, що робить алгоритм практичним для використання у реальних задачах.

Розроблений алгоритм FractalMomentAdam представляє собою синтез сучасних адаптивних технологій із перспективними ідеями з нелінійної динаміки та фрактального аналізу, що створює основу для експериментальної перевірки його ефективності у порівнянні з традиційними підходами.

РОЗДІЛ 3.
ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ FRACTALMOMENTADAM

3.1 Вибір і опис медичного датасету

Для експериментальної перевірки ефективності розробленого оптимізатору FractalMomentAdam було вирішено використати медицинський датасет PathMNIST, який є частиною комплексного проекту MedMNIST – спеціально розробленої колекції даних для досліджень у галузі медичного машинного навчання [12]. Вибір саме цього набору даних обумовлений його клінічною релевантністю, технічною доступністю та тими викликами, які він ставить перед традиційними методами оптимізації.

Проект MedMNIST виник як відповідь на потребу у стандартизованих, але клінічно значущих датасетах для швидкої перевірки алгоритмів. На відміну від класичного MNIST, який містить зображення рукописних цифр, MedMNIST агрегує реальні медичні зображення різних модальностей, попередньо оброблені до формату 28×28 пікселів . Він був вперше представлений у статті “MedMNIST v2: A Large-Scale Lightweight Benchmark for 2D and 3D Biomedical Image Classification” (Yang et al., 2021) і являє собою облегшену, зручну для використання платформу для тестування алгоритмів діагностики на основі глибокого навчання. Це дозволяє дослідникам зосередитися на розробці методів, а не на трудомісткій ручній підготовці даних. Однак, незважаючи на спрощену форму, набори MedMNIST зберігають ключові особливості та виклики медичних зображень: анатомічну варіабельність, артефакти зображення та великий класовий дисбаланс. Також, треба наголосити, що усі дані в MedMNIST є реальними випадками/знімками, які мають чітке ліцензування та документовані джерела.

Для цього дослідження було обрано саме PathMNIST — один із найскладніших та інформативних піднаборів MedMNIST, який орієнтований

на задачу мультикласової класифікації гістопатологічних зображень. PathMNIST базується на датасеті з публікації Kather et al., 2019 — "Predicting survival from colorectal cancer histology slides using deep learning", де було використано дані з гістопатології пацієнтів, хворих на колоректальний рак [13]. З оригінальних Whole-Slide Images (WSI) було витягнуто фрагменти зображень розміром 224×224 пікселів, а для створення PathMNIST їх було перетворено до 28×28 та збережено в RGB-форматі. Кожне зображення являє собою зріз тканини, забарвлений гематоксиліном і еозином (рис. 3.1).

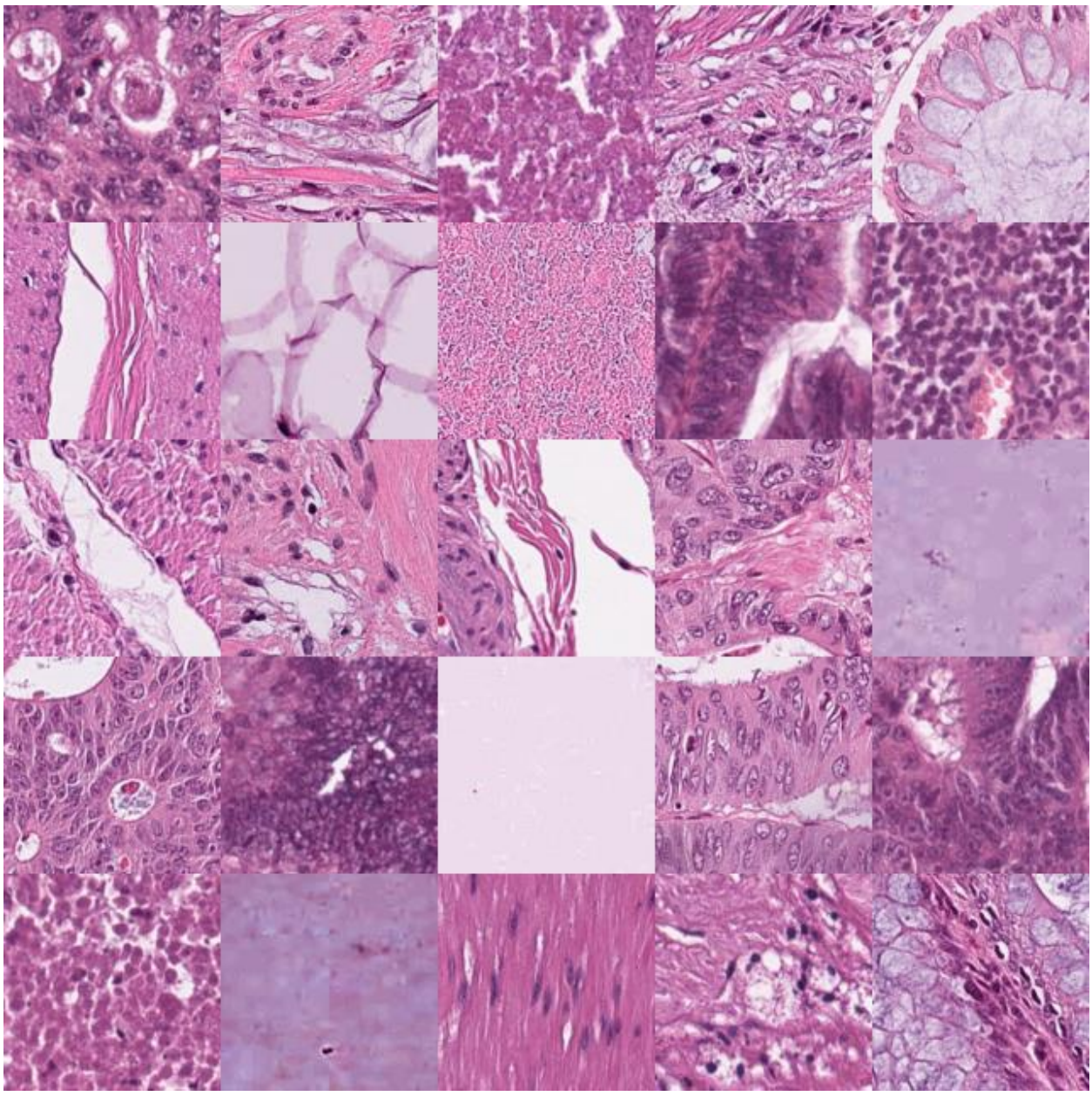


Рисунок 3.1 – приклади забарвлених зрізів тканин датасету PathMNIST

Характеристики датасету (Таблиця 3.1)

Таблиця 3.1

Характеристики датасету PathMNIST

Ознака	**Значення**
Розмір зображення	28 × 28 × 3 (RGB)
Кількість класів	9
Тип задачі	Мультикласова класифікація
Кількість тренувальних прикладів	89 996
Кількість тестових прикладів	7 180
Кількість валідаційних прикладів	10 004

Опис класів

PathMNIST містить 9 класів тканин, які відповідають типам клітин чи структур у гістологічних зрізах товстої кишки (Таблиця 3.2).

Таблиця 3.2

Класи тканин датасету PathMNIST

Клас	Назва
0	Епітеліальна тканина
1	З'єднувальна тканина
2	М'язова тканина
3	Жирові клітини
4	Пухлинна тканина

5	Некротизована тканина
6	Лімфоцити
7	Плазматичні клітини
8	Фон / інше

Чому саме PathMNIST:

- Медична релевантність — дані відображають реальні гістологічні зрізи, які використовуються в онкологічній діагностиці.
- Високий класовий дисбаланс, що створює реалістичну ситуацію для тестування стійкості оптимізаторів. На відміну від чітко розділених цифр у звичайному MNIST, різні типи тканин мають тонкі відмінності, що вимагає від моделі та оптимізатору високої чутливості до деталей.
- Невеликі зображення (28×28×3) — дають змогу проводити експерименти локально, без залучення хмарних обчислень, та без шкоди для наукової достовірності.
- Якість анотацій — лейбли класів сформовані експертами на основі досліджень оригінальних мікроскопічних зображень.
- Присутність артефактів — забарвлення, розрізи та якість зразків варіюються, створюючи "шум", типовий для реальних медичних даних.
- Клінічна релевантність: гістопатологія — галузь, де автоматизація особливо затребувана через високе навантаження на лікарів.

Таким чином, використання PathMNIST у якості тестового датасету дозволяє перевірити адаптивність, стабільність і загальну ефективність оптимізатора FractalMomentAdam в умовах реалістичної, багатокласової медичної задачі.

3.2 Реалізація нейромережевої архітектури та середовище навчання

З метою об'єктивного порівняльного аналізу ефективності розробленого оптимізатора FractalMomentAdam було реалізовано повноцінне середовище навчання нейронної мережі, орієнтованої на класифікацію гістопатологічних

зображень із датасету PathMNIST. Навчання та тести проводились із використанням глибокої згорткової нейронної мережі (CNN). Реалізація була виконана на мові Python із використанням бібліотеки TensorFlow та API Keras.

Передобробка даних

Дані були завантажені безпосередньо з бібліотеки MedMNIST, яка забезпечує доступ до стандартизованих медичних датасетів. Для створення єдиного набору було об'єднано як train, так і test частини датасету PathMNIST, після чого виконано поділ на:

- навчальну вибірку (70%),
- валідаційну (15%),
- тестову (15%).

Кожне зображення має розмірність 28×28 пікселів і три канали кольорів (RGB). Перед подачею до моделі зображення було масштабовано до розмірності 32×32×3 пікселі з використанням шару Resizing у TensorFlow для уніфікації розмірів. Мітки (цільові значення) були закодовані у форматі one-hot для багатокласової класифікації (всього 9 класів, що відповідають типам тканин).

Архітектура моделі

Модель побудована на основі глибокої згорткової нейромережі [14] (CNN) з трьома основними блоками обробки зображень та двома повнозв'язними шарами для класифікації. Кожен блок включає:

- Два згорткових шари з ядром 3x3, активацією ReLU та L2-регуляризацією (λ=1e-4)
- Шар BatchNormalization для стабілізації навчання
- MaxPooling (2x2) для зменшення розмірності
- Dropout з прогресивно зростаючим коефіцієнтом (0.2 → 0.4) для запобігання перенавчанню

Фінальна частина моделі містить:

- Flatten для переходу до повнозв'язних шарів
- Dense шар (256 нейронів) з ReLU-активацією
- Фінальний класифікаційний шар з 9 нейронами (за кількістю класів) та softmax-активацією

Середовище навчання

Модель компілюється з вказаним оптимізатором (у залежності від експерименту — FractalMomentAdam, Adam, SGD, RMSprop), функцією втрат categorical_crossentropy та метрикою точності accuracy. Для контролю за навчанням застосовано EarlyStopping (автоматична зупинка при відсутності покращення),

Параметри навчання:

- batch size: 128,
- кількість епох: 10,
- валідація: на відкладеній частині x_val, y_val.

Після завершення навчання виконується оцінка моделі на тестовій вибірці x_test, y_test, а також візуалізуються графіки зміни точності та втрат на тренувальній та валідаційній вибірках.

3.3 Порівняльний аналіз результатів навчання з різними оптимізаторами

Метою цього експериментального етапу було провести комплексне порівняння ефективності різних оптимізаторів при розв’язанні задачі багатокласової класифікації медичних зображень з набору даних PathMNIST. Завдання передбачає точну класифікацію кожного зображення до одного з класів на основі візуальних ознак.

У ході дослідження було зроблено порівняння поведінки та продуктивності наступних оптимізаторів:

- Adam (базовий адаптивний оптимізатор),
- SGD з моментумом 0.9 (стохастичний градієнтний спуск з інерцією),
- SGD без моментуму,
- RMSProp,
- Fractal Adam з різними значеннями параметра γ (0.2 та 0.6).

Навчання для кожного з оптимізаторів проводилося на одній і тій самій архітектурі глибокої нейронної мережі, без зміни моделі чи структури даних. Зберігався однаковий розмір батчу, функції втрат (кросентропія), початкове значення коефіцієнта навчання та кількість епох (10). Початкова швидкість навчання однакова для всіх (LR 0.001), окрім SGD momentum, для нього було вирішено підняти LR до 0.01 на основі емпіричних результатів запуску SGD без momentum на LR 0.001, і його дуже низьких показниках точності на такому рівні швидкості навчання.

Основними метриками, що використовувалися для оцінки ефективності оптимізаторів, були:

- Точність на тренувальній вибірці (train accuracy)
- Точність на валідаційній вибірці (val accuracy)
- Функція втрат на тренувальній вибірці (train loss)
- Функція втрат на валідаційній вибірці (val loss)
- Швидкість збіжності (конвергенція)
- Стабільність навчання (наявність/відсутність різких стрибків або деградації результату)

Окремий акцент був зроблений на аналізі того, як поводить себе Fractal Adam із різними параметрами γ, і чи дійсно фрактальна компонента дає покращення порівняно з класичним Adam або іншими відомими підходами.

3.3.1 Загальні результати навчання на різних оптимізаторах

Нижче представлена таблиця 3.3 з усіма результатами навчання на різних оптимізаторах по кінцевій епосі. Жирним позначено найкращі результати в кожній метриці.

Таблиця 3.3

Результати навчання ШНМ на різних оптимізаторах

Оптимізатор	Train Acc	Train Loss	Val Acc	Val Loss
Fractal Adam (γ=0.6)	0.9606	0.2017	0.9627	**0.2246**
Fractal Adam (γ=0.2)	**0.9641**	0.2051	**0.9644**	0.2251
Adam	0.9570	0.2093	0.9401	0.2662
SGD + Momentum (0.9, LR=0.01)	0.9419	0.2559	0.6877	2.1697
SGD (без моментуму, LR=0.001)	0.7087	0.8861	0.5762	1.2159
RMSProp	0.9610	**0.1813**	0.7567	0.8732

Також, графічні кінцеві метрики тренувальної точності (рис 3.2), валідаційної точності (рис 3.3), тренувальних втрат (рис 3.4) та валідаційних втрат (рис 3.5).

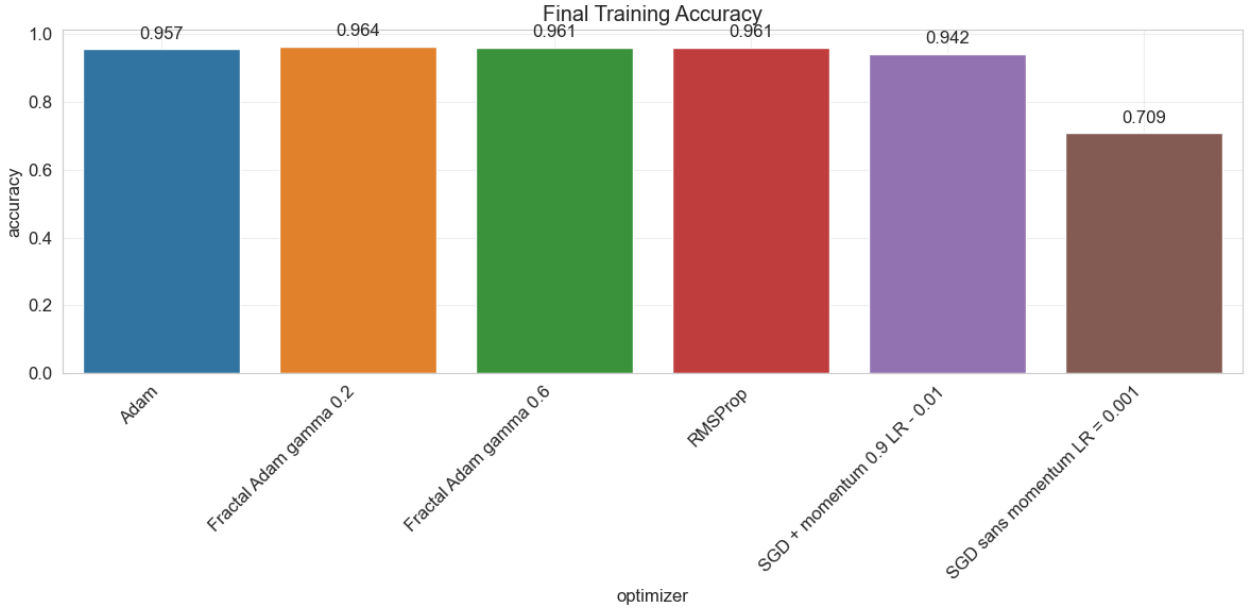


Рисунок 3.2 – Кінцева тренувальна точність оптимізаторів

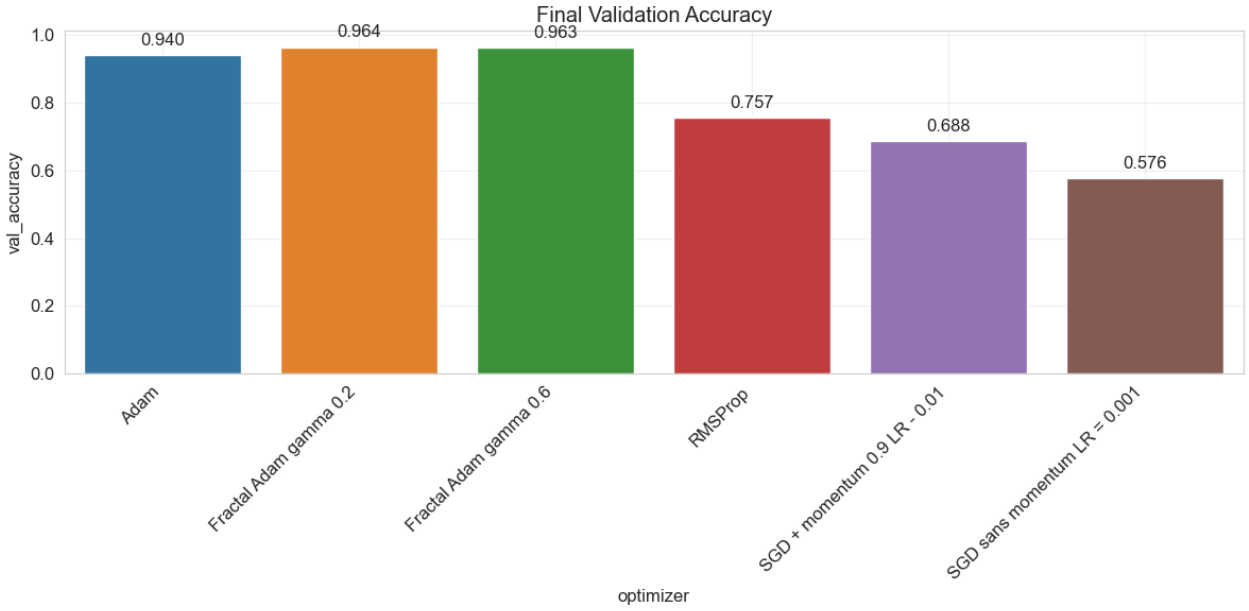


Рисунок 3.3 – Кінцева валідаційна точність оптимізаторів

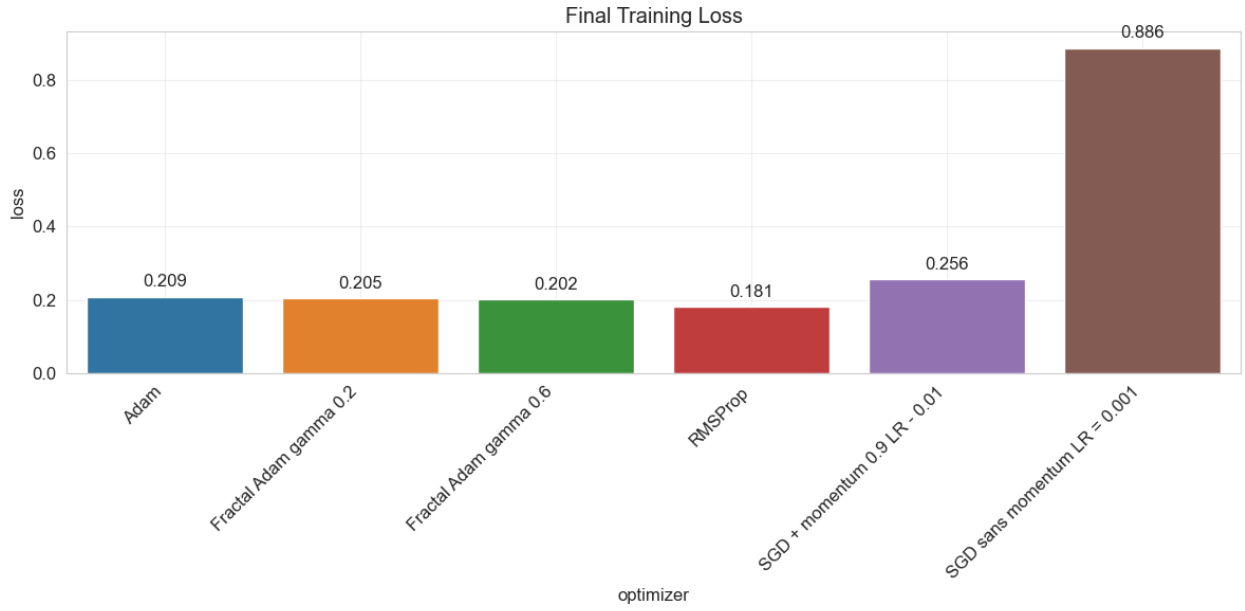


Рисунок 3.4 – Кінцеві тренувальні втрати

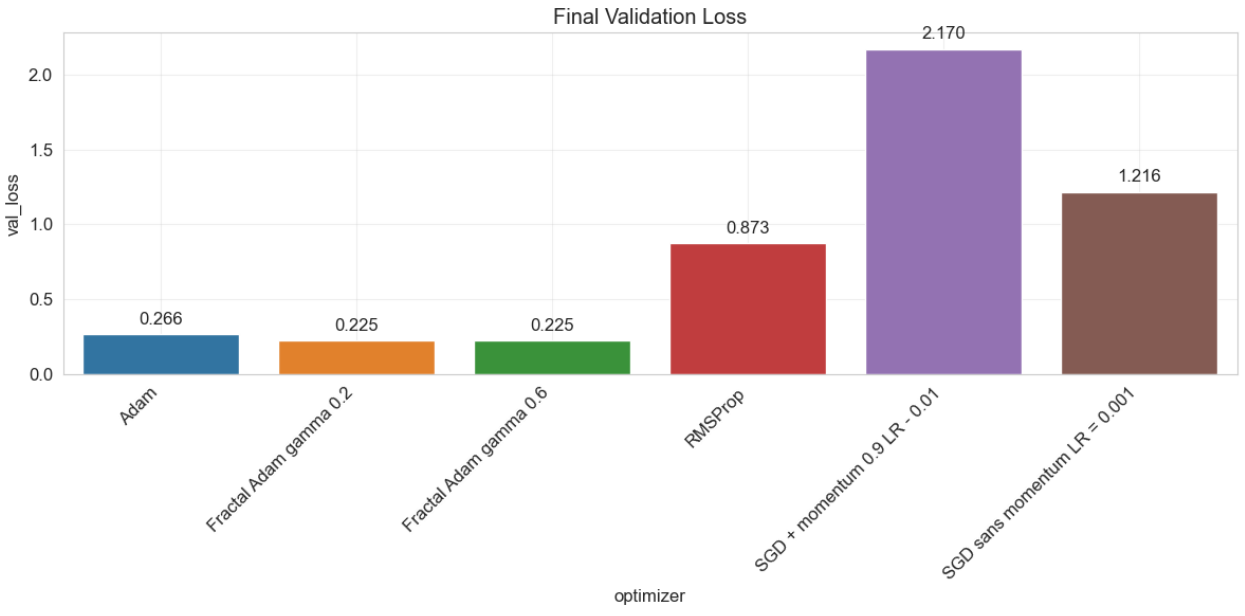


Рисунок 3.5 – Кінцеві валідаційні втрати

3.3.2 Тестова та валідаційна точність, швидкість сходження, стабільність, аналіз та порівняння

Експериментальний аналіз виявив суттєві відмінності у поведінці оптимізаторів на датасеті PathMNIST. На тренувальній вибірці усі оптимізатори показали стабільне навчання. Найбільш показовим критерієм стала динаміка точності на валідаційному наборі, яка демонструє здатність моделі до узагальнення.

Нижче представлені графіки навчання на тренувальній (рис. 3.6) та валідаційній (рис. 3.7) вибірках.

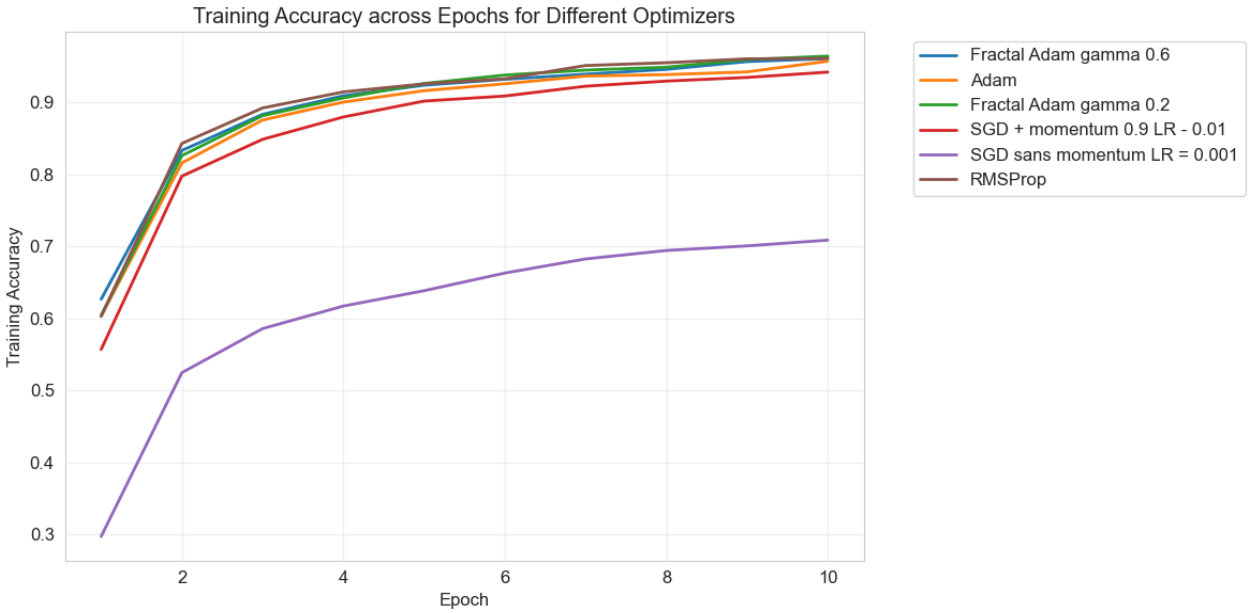


Рисунок 3.6 – Динаміка навчання на тренувальній вибірці.

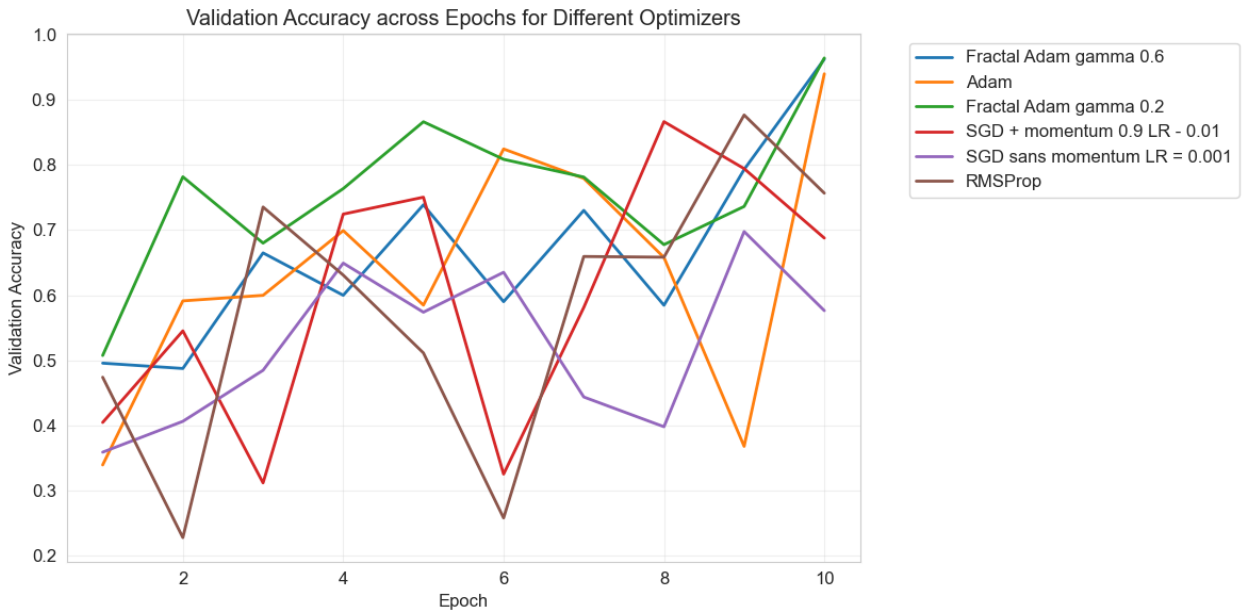


Рисунок 3.7 – Динаміка навчання на валідаційній вибірці.

1. **FractalMomentAdam (γ=0.6)**:
    - Досяг пікової val_accuracy = 96.27%
    - Швидкий старт: вже на 3-й епосі val_accuracy = 66.49%
    - Стабільне зростання без різких коливань (відхилення ±13.8%)
2. **Adam**:
    - Пікова val_accuracy = 94.01%
    - Виражені коливання між епохами
    - Повільніший старт (59.97% на 3-й епосі)
3. **FractalMomentAdam (γ=0.2)**:
    - Найвища стабільність (±11.3%)
    - Плавне зростання до пікового значення 96.44%
4. **SGD з моментом**:
    - Різкі стрибки (від 22.50% до 76.66%)
    - Середня стабільність значно нижча (±20.1%)

5. **Чистий SGD**:

    - Найгірші результати (макс. val_accuracy = 69.77%)
    - Навчання в 2-3 рази повільніше

6. **RMSProp**:

    - Найвища тренувальна точність (96.10%) серед усіх методів
    - Катастрофічні провали валідації (наприклад, падіння до 25.76% на 6-й епосі)
    - Низька фінальна точність (75.67%)

3.3.3 Аналіз та порівняння тестових та валідаційних втрат

Як і в огляді росту точності, на графіках втрат на тренувальній (рис. 3.8) та валідаційній (рис. 3.9) вибірках, зниження втрат відбувається аналогічно стабільно на тренувальній вибірці, але показує нестабільні результати на валідації.

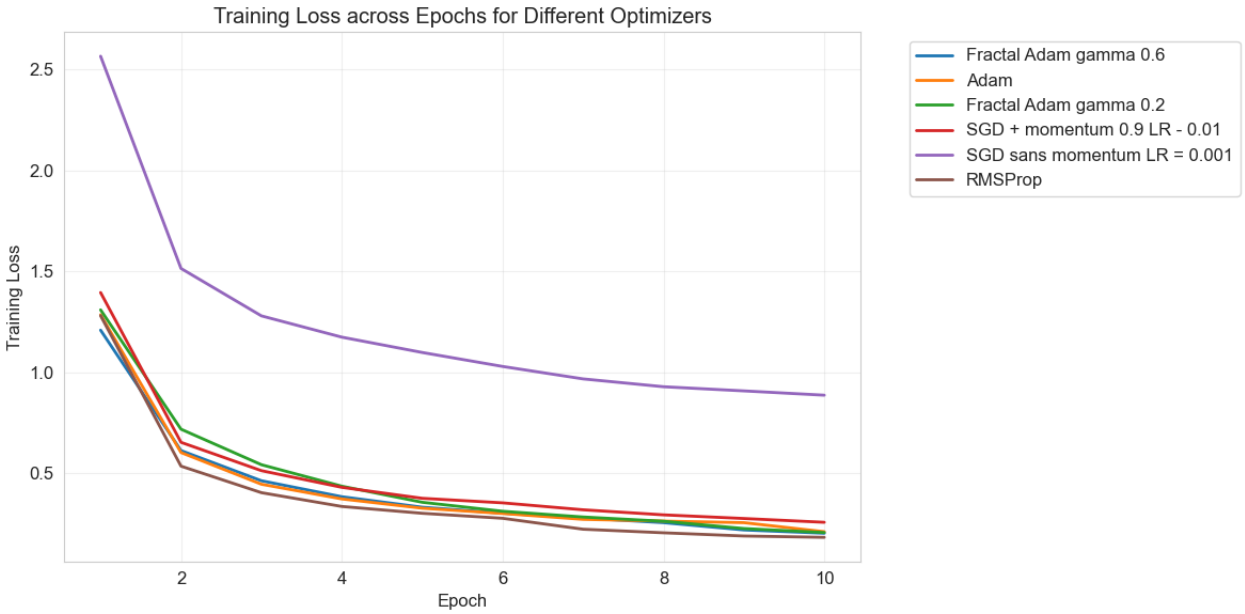


Рисунок 3.8 – Динаміка втрат на тренувальній вибірці.

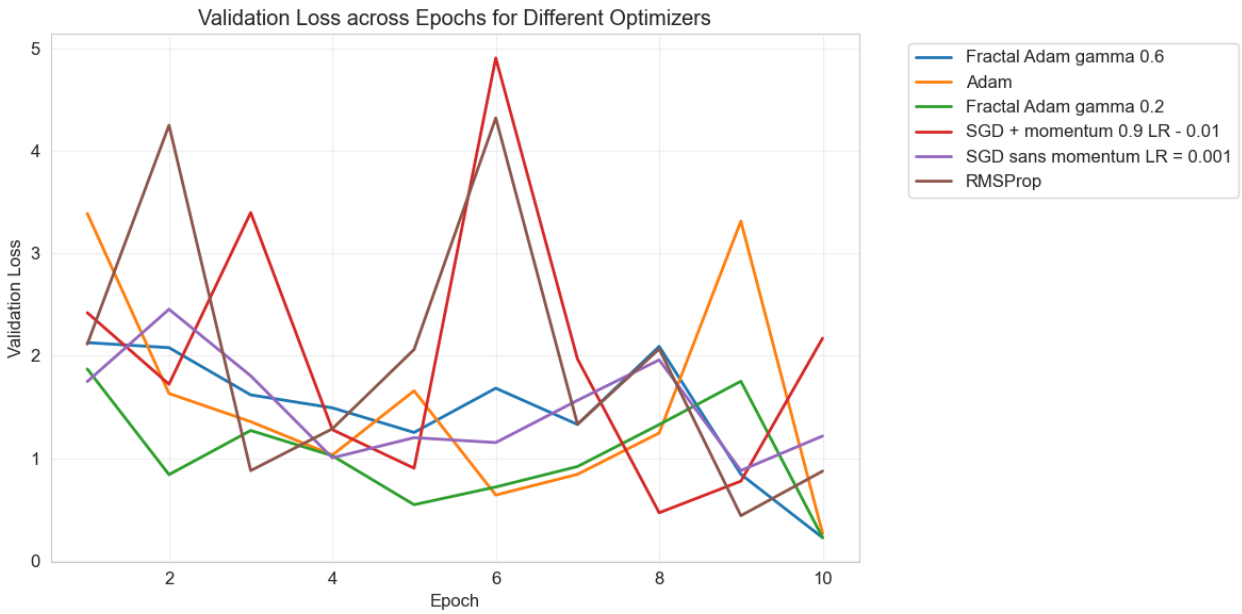


Рисунок 3.9 – Динаміка втрат на валідаційній вибірці.

Аналіз кривих втрат на валідації демонструє помітні розбіжності у поведінці оптимізаторів. Усі методи демонструють загальну тенденцію до зменшення втрат, однак ступінь стабільності, швидкість сходження та амплітуда флуктуацій значно відрізняються. Ці характеристики є критично важливими для розуміння здатності оптимізатора не лише ефективно тренувати модель, але й забезпечити її загальну здатність до узагальнення.

FractalMomentAdam ($\gamma = 0.2$ та $\gamma = 0.6$):

- Обидві версії демонструють стабільну і гладку динаміку зниження втрат.
- Для $\gamma = 0.2$ спостерігається найменша дисперсія на валідації — втрати зменшуються поступово, без стрибків, що вказує на високу узагальнюючу здатність та хорошу регуляризацію.
- Для $\gamma = 0.6$ фіксується невеликий пік на 8-ій епосі, проте швидке повернення до низьких втрат свідчить про ефективну адаптацію після локального мінімуму.

FractalMomentAdam демонструє найкращу стабільність, стійкість до перенавчання та застрягання у локальних мінімумах.

Adam:

- Спостерігаються коливання втрат у широкому діапазоні. Наприклад, на 9-ій епосі — раптове зростання втрат в декілька разів.
- Це може свідчити про періодичні проблеми з узагальненням: модель переобучається або застряє в локальному мінімумі.
- Незважаючи на добру точність, нестабільність втрат робить Adam менш передбачуваним та стабільним під час навчання/роботи.

Adam — потужний, але потребує додаткових регуляризаційних заходів (додаткових до тих, які вже були вбудовані в архітектуру ШНМ) для стабільного узагальнення.

RMSProp:

- Один із найнестабільніших алгоритмів: на 2-ій та 6-ій епосі — різке зростання втрат (більше ніж у 3 рази).
- Це вказує на нестійку динаміку оновлень і чутливість до зміни градієнтів.
- Незважаючи на хорошу тренувальну продуктивність, RMSProp сильно перенавчається.

RMSProp не забезпечує достатньої стійкості до зашумленних даних.

SGD (з моментумом і без):

- Обидва варіанти демонструють високий рівень втрат.
- Із моментумом видно хаотичні стрибки (наприклад, на 3-й та 6-й епосі).
- Чистий SGD демонструє повільне зниження втрат з меншими катастрофічними піками.

Хоча SGD може бути стабільним на більш «рівних» даних, але на зашумленних даних він не забезпечує конкурентних результатів.

Оптимізатори FractalMomentAdam, особливо з γ=0.2, показали найкращу поведінку за критеріями зниження втрат та стабільності. У той час як RMSProp та Adam можуть давати непогані результати в окремих випадках, їхня нестабільність робить їх менш надійними. Класичний SGD потребує ретельного налаштування або комбінування з сучасними преднавченними моделями.

3.4 Висновки по порівнянню оптимізаторів

Проведене експериментальне дослідження виявило фундаментальні відмінності в поведінці оптимізаційних алгоритмів. Аналізуючи динаміку навчання, ми спостерігаємо цілісні картини, що розкривають внутрішню логіку кожного методу та його взаємодію зі складними медичними даними.

FractalMomentAdam проявив себе як якісно новий підхід до оптимізації. Його унікальність полягає у здатності одночасно оперувати на різних масштабах даних — від дрібних текстурних особливостей до глобальних змін. Ця властивість виникає завдяки глибокому синтезу трьох ключових принципів: адаптивності моментів, фрактального аналізу градієнтів та динамічного змішування часових горизонтів. На практиці це виражається в плавних, передбачуваних траєкторіях навчання, де кожен наступний крок не просто мінімізує поточні втрати, але й узгоджується з довгостроковою структурою задачі. Тобто, мультишкальність, яка дає нам можливість покращувати локальні значення через погляд на довгострокові зміни.

Традиційні методи (Adam, RMSProp) демонструють принципово іншу філософію — вони починають більш агресивно досліджувати весь простір, але, із-за шумів та плато, швидко застрягають та втрачають точність., що також призводить до періодичних "катастроф", коли модель раптово втрачає

раніше набуті знання. Особливо показово це видно в середині навчання, коли раптово втрачається здобута точність.

Чистий SGD займає проміжне положення — його простота дає певну стабільність, але мінусом є його нездатність вийти із локальних мінімумів, що особливо критично для медичних задач з їх складним багатомодальним ландшафтом втрат. Цей метод буквально "застрягає" в проміжних станах, не маючи механізмів для оцінки глобального контексту.

SGD з імпульсом виявився абсолютно нежиттєздатним у цих умовах — його траєкторія навчання хаотична та має високі втрати та найбільший рівень «катастроф».

Таким чином, експерименти підтвердили, що для досягнення нового рівня якості в медичному машинному навчанні є необхідність відійти від классичних оптимізаційних підходів і зосередитись на розробці нових методів, які враховують нелінійні залежності та стійкість до шумів.

Висновки за розділом 3

Третій розділ роботи присвячений експериментальному дослідженню ефективності розробленого оптимізатора FractalMomentAdam у порівнянні з низкою популярних алгоритмів оптимізації. Експерименти проводились на медичному датасеті PathMNIST, що містить гістопатологічні зображення для задачі мультикласової класифікації, що дозволило імітувати реальні умови роботи з складними, зашумленими даними, які характеризуються високою міжкласовою подібністю та анатомічною варіативністю.

Результати експериментів чітко продемонстрували переваги запропонованого підходу. Оптимізатор FractalMomentAdam, особливо з параметром змішування $\gamma=0.2$, показав найкращі результати за критеріями стабільності та якості узагальнення. Він забезпечив найвищу валідаційну точність (96.44%) та продемонстрував плавну, передбачувану динаміку навчання з мінімальними коливаннями втрат і точності між епохами. Це

свідчить про здатність алгоритму ефективно балансувати між реакцією на локальні зміни градієнтів та врахуванням глобальних трендів, що є ключовим для успішного навчання на медичних даних.

Порівняно з ним, класичний Adam, незважаючи на високу тренувальну точність, показав виражені коливання на валідації та нижчий фінальний результат, що підтверджує його схильність до нестабільності та перенавчання в умовах шуму. Ще більш проблемною виявилася поведінка RMSProp, яка супроводжувалася катастрофічними провалами валідаційної точності, що робить цей метод ненадійним для складних медичних задач. Стохастичні методи (SGD з моментумом та без нього) продемонстрували або повільну збіжність, або хаотичну динаміку з високими значеннями втрат, що підтверджує їхню неефективність при роботі з складним багатомодальним ландшафтом функції втрат.

Таким чином, експериментальна частина роботи стала переконливим підтвердженням гіпотези про те, що фрактальний підхід до оптимізації забезпечує якісно новий рівень стабільності та ефективності. Здатність FractalMomentAdam до швидкої збіжності, мінімізації коливань та досягнення високої узагальнюючої здатності робить його перспективним інструментом для вирішення реальних клінічних завдань, де передбачуваність та надійність роботи моделі є критично важливими. Отримані результати створюють міцну основу для подальшого впровадження та розвитку даного методу.

РОЗДІЛ 4.
ПЕРСПЕКТИВИ ПОДАЛЬШОГО ВИКОРИСТАННЯ ТА АДАПТАЦІЇ FRACTALMOMENTADAM

4.1 Застосування в інших типах задач машинного навчання

Розроблений оптимізатор FractalMomentAdam демонструє стабільну збіжність на складних медичних даних та має високий потенціал до перенесення у значно ширший клас задач, де класичні методи оптимізації (наприклад, Adam) можуть зазнавати труднощів. Йдеться насамперед про ті задачі, у яких нестабільні градієнти та багато шумів, або поверхня функції втрат містить численні плоскі регіони, локальні мінімуми або розгалуження, що утруднюють навігацію у просторі параметрів.

Окрім застосування до задач багатокласової класифікації медичних зображень (наприклад, датасет PathMNIST), FractalMomentAdam може бути корисним у глибокому аналізі знімків гістології, дерматоскопії, рентгену або МРТ, де часто зустрічаються великі варіативності структур тканин, нечіткі межі патологічних регіонів або інших уражень здорової тканини, нерівномірна якість анотацій/журналів спостереження. Зокрема, в онкології, дерматології та пульмонології, часто містять високовимірні зображення з нечіткими межами класів, варіативністю анотацій між фахівцями та значною кількістю неструктурованих або помилкових зразків. У таких умовах використання оптимізатора, здатного виявляти глобальні закономірності без надмірної залежності від локальних змін у градієнтному полі, має особливу цінність. FractalMomentAdam має потенціал покращити не лише точність моделей на таких наборах даних, а й, що більш важливо, зробити процес навчання більш стабільним та контрольованим, що є критичним у медичних системах прийняття рішень.

Ще одним напрямом, який потребує стабільної адаптивної оптимізації, є контроль якості мікрочіпів та наноструктур за допомогою аналізу зображень

електронної мікроскопії. У цих випадках глибокі згорткові нейронні мережі повинні навчатися на вкрай деталізованих зображеннях з високою щільністю інформації, де дефекти або мікроскопічні порушення структури можуть проявлятися у вигляді слабких аномалій. Особливістю таких задач є сильна диспропорція класів, а також високий рівень візуального шуму, викликаного як технічними характеристиками апаратури, так і особливостями самого матеріалу. Фрактальний підхід до оновлення градієнтів дозволяє моделі бути чутливішою до тонких змін у втраті навіть при слабкому сигналі, зберігаючи при цьому глобальну стабільність.

Також, має сенс розглянути тестування фрактального алгоритму оптимізації у глибокому навчанні з підкріпленням. У таких задачах часто спостерігається складна динаміка середовища і традиційні оптимізатори часто дають нестабільні результати через високу дисперсію градієнтів та змінні винагороди для ШНМ, FractalMomentAdam може забезпечити кращу стабільність функції цінності в умовах флуктуацій градієнтного ландшафту, оскільки він є менш чутливим до одиничних вибухів градієнтів і краще зберігає корисний тренд.

Ще однією з перспективних сфер застосування є обробка супутникових знімків, де моделі повинні виявляти об'єкти, класифікувати типи земного покриву або відслідковувати зміни на місцевості в умовах суттєвого шуму, хмарності, атмосферних спотворень та різної роздільності зображень. У таких задачах поверхня помилки часто має фрагментарний характер, а області локальних мінімумів можуть вводити в оману класичні оптимізатори. Застосування FractalMomentAdam дозволить уникати застрягання у локальних мінімумах завдяки нелінійній корекції моментів, особливо в ранніх фазах навчання.

Також, одним з напрямів застосування є біоінформатика. Завдання аналізу геномних послідовностей, передбачення структури білків, обробки даних однонуклеотидного поліморфізму — усе це вимагає роботи з

надзвичайно довгими, фрагментованими, структуровано-шумовими послідовностями. Зокрема, подібні задачі часто характеризуються високошумовими та слабкими градієнтами, а вхідні дані мають багаторівневу фрактальну організацію. Тут фрактальні методи прямо відповідають фрактальній природі вхідних даних, і тому FractalMomentAdam може дозволити більш адаптивне навчання без потреби складного препроцесингу.

Також зараз однією з найпопулярніших та найперспективніших сфер є обробка природної мови, але, часто, вхідні дані в таких задачах зашумлені та мають багато артефактів, наприклад, граматичні помилки, скорочення, фонетичні особливості, багатозначність контексту, містять спотворення, спричинені попередніми обробками (наприклад, при роботі з модулями TTS/STT). FractalMomentAdam може ефективніше адаптуватися до динаміки втрат при обробці такого "брудного" або слабко формалізованого тексту, надаючи моделі більше свободи на ранніх фазах і маючи можливість коректування локальних моментів з оглядом на глобальний тренд.

4.2 Потенційні напрямки подальшого розвитку FractalMomentAdam

Розроблений оптимізатор FractalMomentAdam, що поєднує класичну ефективність адаптивного алгоритму Adam із гнучкістю багатомасштабного фрактального згладжування, відкриває перспективні напрями для подальших досліджень і вдосконалення. Його фундамент базується на спостереженні, що навчання нейронної мережі у складних умовах, таких як шумні, нерівномірно збалансовані чи високорозмірні дані, дає кращі результати точності та втрат при адаптивній динаміці, що враховує нелінійні залежності та багаторівневу структуру втрат.

Першочерговим напрямком може стати поглиблена адаптація параметрів фрактального згладжування під конкретні класи задач. Поточна реалізація використовує фіксовані коефіцієнти фрактального моменту другого порядку, однак можна запровадити динамічне коригування параметрів $\beta 1s$ та

β2s у часі, чи залежно від топології градієнтного ландшафту. Застосування механізмів уваги та саморегуляції для керування цими параметрами може зробити FractalMomentAdam ще більш адаптивним для конкретних фаз навчання.

Також, перспективним є збільшення кількості моментів (β) для коректування оптимізації на більшій кількості масштабів. Але, у цьому випадку треба враховувати, що додавання нових моментів це додатковий слот пам’яті, що збільшує час виконання та витрати обчислювальних ресурсів. Тобто, для дуже складних датасетів треба буде емпірично дізнаватися максимальну кількість додаткових масштабів для спостереження.

Крім того, перспективною є інтеграція фрактального підходу в інші класи оптимізаторів, зокрема в методи на основі Natural Gradient, Sharpness-Aware Minimization або в гібридні, такі як Lion чи AdaBelief. Це дозволить розширити застосування фрактальної згладженої динаміки на випадки, де необхідна підвищена чутливість до кривизни функції втрат або контроль за площинами локальних мінімумів.

У контексті апаратного прискорення, варто дослідити ефективність FractalMomentAdam на різних архітектурах (наприклад, TPU та FPGA). Завдяки своїй структурі, розроблений оптимізатор добре масштабується і може демонструвати переваги в енергозбереженні при роботі на edge-пристроях або в умовах обмеженого обчислювального ресурсу.

Таким чином, запропонований оптимізатор є не лише цінним практичним інструментом для глибокого навчання на зашумлених і складних даних, а й потенційною відправною точкою для розробки нової парадигми адаптивної фрактальної оптимізації, яка поєднує ідеї з нелінійної динаміки, стохастичних процесів і багатомасштабного аналізу.

Висновки за розділом 4

Четвертий розділ роботи присвячений аналізу перспектив подальшого використання та вдосконалення розробленого оптимізаційного алгоритму FractalMomentAdam. Запропонований підхід має значний потенціал для застосування не лише в задачах класифікації медичних зображень, але й в інших складних галузях машинного навчання, де традиційні оптимізатори демонструють обмежену ефективність. Перспективними напрямами для адаптації алгоритму визначено аналіз мікроскопічних зображень у нанотехнологіях, а також задачі глибокого навчання з підкріпленням, де висока дисперсія градієнтів є типовими явищем. Окрім того, фрактальний підхід може виявитися корисним при роботі з супутниковими даними, біоінформатичними послідовностями та завданнями обробки природної мови, які характеризуються високим рівнем шуму, багатомасштабністю та структурною складністю вхідних даних.

Важливим аспектом розділу став огляд потенційних напрямів подальшого розвитку самого алгоритму. Серед них — впровадження механізмів динамічної адаптації параметрів фрактального згладжування, таких як коефіцієнти $\beta 1s$ та $\beta 2s$, на основі поточної топології градієнтного ландшафту або фаз навчання. Іншим перспективним кроком може стати розширення кількості масштабів моменту, що дозволить ще точніше враховувати багаторівневу структуру даних, хоча це вимагатиме додаткових обчислювальних ресурсів. Також актуальною є інтеграція фрактального підходу в інші класи оптимізаторів, що дозволить створити нові гібридні методи з покращеною збіжністю. Окремо варто дослідити ефективність FractalMomentAdam на спеціалізованих апаратних архітектурах, таких як TPU та FPGA, що може відкрити можливості для його використання в системах з обмеженими обчислювальними потужностями, зокрема в edge-пристроях.

ВИСНОВКИ

У результаті проведеного дослідження було розроблено, реалізовано та експериментально перевірено новий оптимізаційний алгоритм FractalMomentAdam, який поєднує класичні принципи адаптивного градієнтного навчання з додатковими механізмами фрактальної стабілізації моментів. Запропонований підхід має на меті подолання деяких фундаментальних обмежень традиційних оптимізаторів типу Adam та підвищення валідаційної точності та стабільності процесу навчання глибоких нейронних мереж у задачах медичної діагностики.

На практиці було інтегровано оптимізатор у типовий пайплайн глибинного навчання та проведено експериментальне порівняння з класичними оптимізаторами на основі реального медичного набору даних PathMNIST із колекції MedMNIST. Цей датасет було обрано з огляду на його реалістичність, багатокласовість і важливість для медичної діагностики, що забезпечує надійну основу для порівняння ефективності оптимізаторів у складному середовищі.

Аналіз результатів показав, що FractalMomentAdam демонструє стабільнішу поведінку впродовж навчання, знижує коливання показників точності між епохами, а також досягає високих показників як тестової, так і валідаційної точності швидше, ніж базовий Adam. Зокрема, фрактальна модуляція дозволяє краще адаптуватися до мінливих топологій простору втрат, що особливо важливо у медичних задачах із високою міжкласовою подібністю та слабко вираженими градієнтами. Графіками та метриками було показано, що FractalMomentAdam швидше виходить із фаз початкового хаосу, зменшуючи ймовірність «застрягання» в локальних мінімумах. Показники втрат при цьому не лише нижчі, а й демонструють плавну та контрольовану динаміку протягом навчання, що є ознакою покращеної узагальнюючої здатності моделі. На практиці це виражається в плавних, передбачуваних

траєкторіях навчання, де кожен наступний крок не просто мінімізує поточні втрати, але й узгоджується з довгостроковою структурою задачі.

FractalMomentAdam — це принципово новий інструмент, створений спеціально для роботи з реальними медичними даними. Його переваги найяскравіше проявляються саме в тих аспектах, які найбільш важливі для клінічного застосування — передбачуваність, стабільність і здатність до узагальнення у роботі зі складними датасетами, розмір яких, іноді, може перевищувати десятки гігабайт даних.

Головну ціль роботи було досягено — було покращено показники точності, втрат та стабільності навчання ШНМ за наявності шуму та нерівномірних даних на прикладі реального медичного датасету.

Таким чином, можна зробити висновок, що FractalMomentAdam є перспективним напрямом подальшого розвитку оптимізаційних методів у глибинному навчанні, особливо в галузі медичних даних, де вимоги до стабільності, надійності та точності особливо високі. У подальших дослідженнях доцільно перевірити ефективність даного підходу на інших типах задач, зокрема сегментації та багатокласової класифікації. FractalMomentAdam відкриває шлях до нових типів гібридних оптимізаторів, які здатні поєднувати гнучкість фрактальних структур з адаптивністю сучасних градієнтних методів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bodner B. 10 PyTorch Optimizers Everyone Is Using. 2024. URL: https://medium.com/@benjybo7/10-pytorch-optimizers-you-must-know-c99cf3390899 (date of access: 21.05.2025).
2. Robin M. Schmidt, Schneider F., Hennig P. Descending through a Crowded Valley — Benchmarking Deep Learning Optimizers. 2021. URL: https://arxiv.org/pdf/2007.01547 (date of access: 21.05.2025).
3. GfG, Optimization Algorithms in Machine Learning. 2025. URL: https://www.geeksforgeeks.org/machine-learning/optimization-algorithms-in-machine-learning/ (date of access: 25.05.2025).
4. Reyad M., Amany M. Sarhan, Arafa M. A modified Adam algorithm for deep neural network optimization. 2023. URL: https://link.springer.com/article/10.1007/s00521-023-08568-z (date of access: 28.05.2025).
5. Salmi M., Atif D., Oliva D., Abraham A., Ventura S. Handling imbalanced medical datasets: review of a decade of research. 2024. URL: https://link.springer.com/article/10.1007/s10462-024-10884-2 (date of access: 18.06.2025).
6. Philip Ward, MRI artifacts still require significant care and attention. 2023. URL: https://www.auntminnieeurope.com/clinical-news/article/15657705/mri-artifacts-still-require-significant-care-and-attention (date of access: 18.06.2025).
7. G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov Improving neural networks by preventing co-adaptation of feature detectors. 2012. URL: https://arxiv.org/pdf/1207.0580 (date of access: 21.05.2025).
8. Koushik, Optimization Algorithms in Machine Learning: A Comprehensive Guide to Understand the concept and Implementation. 2023. URL:

https://medium.com/@koushikkushal95/optimization-algorithms-in-machine-learning-a-comprehensive-guide-to-understand-the-concept-and-3db1df7a2f59 (date of access: 21.05.2025).

9. Barnsley, Michael F.; Rising Hawley; Fractals Everywhere. Boston: Academic Press Professional, 1993. ISBN 0-12-079061-0
10. Diederik P. Kingma, Jimmy Lei Ba, ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION. 2015. URL: https://arxiv.org/pdf/1412.6980 (date of access: 10.06.2025).
11. Keras 3 API documentation. 2025. URL: https://keras.io/api/ (date of access: 18.06.2025).
12. Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister & Bingbing Ni. MedMNIST v2 - A large-scale lightweight benchmark for 2D and 3D biomedical image classification. 2023. URL: https://www.nature.com/articles/s41597-022-01721-8 (date of access:18.06.2025).
13. Jakob Nikolas Kather. Predicting survival from colorectal cancer histology slides using deep learning: A retrospective multicenter study. 2019. URL: https://journals.plos.org/plosmedicine/article?id=10.1371/journal.pmed.1002730 (date of access: 18.06.2025).
14. Convolutional Neural Network (CNN) TensorFlow Guide for CIFAR. 2025. URL: https://www.tensorflow.org/tutorials/images/cnn (date of access: 25.06.2025).

ДОДАТКИ

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра комп'ютерних систем та робототехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) ***Магістр***
Галузь знань: 17 – Електроніка, автоматизація та електронні комунікації
Спеціальність: 174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка
Освітня програма «Комп'ютеризовані системи управління та автоматика»

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних
систем та робототехніки
к. ф.-м. н., доц. ХРУСЛОВ М. М.
«02» жовтня 2024року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

СУДАКОВА Дмитра Геннадійовича
(прізвище, ім'я, по батькові студента)

1. Тема роботи «**ФРАКТАЛЬНО-АДАПТИВНІ МЕТОДИ ОПТИМІЗАЦІЇ ГЛИБОКИХ НЕЙРОМЕРЕЖ У МЕДИЧНІЙ ДІАГНОСТИЦІ»**
_____

керівник роботи **ШМАТКОВ Сергій Ігорович, доктор технічних наук, професор**
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету ***від 30 вересня 2025 року № 4101-5/3554***

2. Строк подання студентом роботи ***30 листопада 2025 року***

3. Перелік питань, які потрібно розробити

1) Аналіз підходів та методів оптимізації глибоких нейромереж у медичній діагностиці.
2) Аналіз особливостей та проблематики медичних датасетів.
3) Розробка алгоритму оптимізації на основі фрактальної самоподібності.
4) Програмна реалізація оптимізаційного алгоритму та моделі глибокого навчання.
5) Тестування та оцінка ефективності розробленого алгоритму.

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1	Огляд літератури щодо розробки оптимізаційних алгоритмів та їх впровадження в згорткові нейромережі.	2.09.2025-10.09.2025
2	Огляд та аналіз існуючих проблем з даними в медичних датасетах та вплив їх особливостей на процес навчання.	11.09.2025-15.09.2025
3	Розробка оптимізаційного алгоритму та програмного середовища для його тестування.	16.09.2025-29.09.2025
4	Пошук даних для навчання розробленої моделі машинного навчання.	30.09.2025-5.10.2025
5	Тестування та оцінка ефективності розробленого оптимізаційного алгоритму.	6.10.2025-20.10.2025
6	Розробка рекомендацій щодо вдосконалення оптимізаційного алгоритму та впровадження його у інші типи задач.	21.10.2025-30.10.2025
7	Оформлення пояснювальної записки та звітних матеріалів та додатків кваліфікаційної роботи відповідно вимогам до звітів про НДР.	31.10.2025-14.11.2025
8	Представлення кваліфікаційної роботи керівнику та рецензенту.	15.11.2025-30.11.2025

5. Дата видачі завдання ***02 жовтня 2025 року.***

Студент **Д. Г. Судаков** _____
ініціали, прізвище підпис

Керівник роботи **С. І. Шматков** _____
ініціали, прізвище підпис

Додаток Б

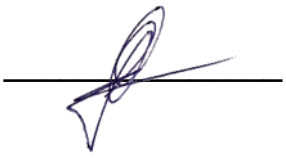
Технічне завдання
на розробку програмного виробу «Фрактально-адаптивні методи оптимізації глибоких нейромереж у медичній діагностиці»

1.	Введення	1.1. Назва: Оптимізаційний алгоритм на основі фрактальної самоподібності для ШНМ медичної діагностики . 1.2. Галузь застосування: Автоматизація аналізу та класифікації медичних знімків, машинне навчання.
2.	Підстава для розробки	2.1. Навчальний план за спеціальністю 174 – Автоматизація та комп'ютерно-інтегровані технології 2.2. Завдання на дипломну роботу магістра № 4101-5/3554 від «30» вересня 2025 (представити як Додаток А до пояснювальної записки до дипломної роботи).
3.	Призначення розробки	3.1. Мета розробки: покращення результатів точності та стабільності навчання ШНМ в задачах класифікації в медичній діагностиці. 3.2.Призначення розробки надає можливість підвищити точність та передбачуваність навчання у задачах медичної класифікації на зашумлених датасетах. 3.3. Вихідні дані розробки: експериментальні дані, а саме, знімки гістологічних зрізів тканин товстої кишки.
4.	Технічні вимоги до програмного виробу	4.1. Вимоги до функціональних характеристик: покращення точності класифікації на медичних датасетах. 4.2. Вимоги до надійності: забезпечувати точність та достовірність визначення ознак та стабільність вищу за існуючі методи та програмне забезпечення. 4.3.Вимоги до умов експлуатації: немає 4.4. Вимоги до складу і параметрів технічних засобів: звичайне обчислювальне обладнання, ПК, тощо 4.5.Вимоги до інформаційної та програмної сумісності: Python 3.8+, Tensorflow 2.19, Keras 3.10. 4.6. Вимоги до маркування та упаковки: немає 4.7. Вимоги до транспортування і зберігання: на звичайних носіях інформації та/або хмарних сервісах. 4.8. Спеціальні вимоги: немає.

<table>
<tr><td>5.</td><td>Вимоги до програмної документації</td><td colspan="2">Програмною документацією до виробу «Оптимізаційний алгоритм на основі фрактальної самоподібності для ШНМ медичної діагностики» вважати:
1) Справжнє Технічне завдання на розробку виробу (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи).
2) Методику розрахунку інформативності змінних стану (у вигляді глав 2.4 та 2.5 пояснювальної записки до кваліфікаційної роботи).
3) Програму і методику випробувань (Додаток В до пояснювальної записки).</td></tr>
<tr><td rowspan="8">6.</td><td rowspan="8">Стадії і етапи розробки</td><td>Дата</td><td>Назва етапу</td></tr>
<tr><td>02.09.2025-
10.09.2025</td><td>Аналіз профільної та наукової літератури.</td></tr>
<tr><td>11.09.2025-
18.09.2025</td><td>Аналіз існуючих підходів до програмної оптимізації навчання ШНМ на медичних даних.</td></tr>
<tr><td>19.09.2025-
30.09.2025</td><td>Пошук та підготовка набору даних для навчання ШНМ.</td></tr>
<tr><td>01.10.2025-
15.10.2025</td><td>Розробка та експериментальне доопрацювання оптимізаційного алгоритму на медичному датасеті.</td></tr>
<tr><td>15.10.2025-
31.10.2025</td><td>Тестування розробленої методики на реальній вибірці з медичної практики.</td></tr>
<tr><td>01.11.2025-
15.11.2025</td><td>Оформлення результатів. Написання пояснювальної записки.</td></tr>
<tr><td>15.11.2025-
25.11.2025</td><td>Представлення дипломного проекту керівнику дипломної роботи та рецензенту.</td></tr>
<tr><td>7.</td><td>Порядок контролю і</td><td colspan="2">1. Перевірку ходу розробки програми виконувати раз в 3 тижні.</td></tr>
</table>

	приймання програмного продукту (моделі)	2. Захист розробленої моделі провести на засіданні Атестаційної комісії. 3. Пояснювальну записку подати на паперових носіях в 1 примірнику і в електронному вигляді в 1 примірнику.

Виконавець
студент групи КУ- 61
Судаков Д.Г.



Замовник
д. т. н., проф.
Шматков С.І.

Додаток В

Програма і методика випробувань програмного виробу

«Оптимізаційний алгоритм на основі фрактальної самоподібності для ШНМ медичної діагностики»

1. Об'єкт випробувань

1. Назва програмного виробу: «Фрактально-адаптивний оптимізатор FractalMomentAdam для навчання нейронних мереж у медичних задачах».

2. Галузь застосування: Програмний виріб призначений для підвищення стабільності та ефективності навчання глибоких нейронних мереж у задачах комп’ютерної діагностики, класифікації медичних зображень та біомедичних сигналів. Реалізовано у середовищі TensorFlow/Keras (Python).

2. Мета випробувань

Метою випробувань є перевірка:

- коректності реалізації алгоритму FractalMomentAdam;
- стабільності роботи оптимізатора в процесі навчання;
- відповідності до технічного завдання (Додаток Б);
- підвищення точності та швидкості збіжності порівняно з класичними оптимізаторами (Adam/SGD/RMSProp).

3. Загальні положення

3.1. Підстави для проведення випробувань

Підставою для проведення випробувань є наказ про призначення атестаційної комісії.

3.2. Місце і тривалість випробувань

Приймальні (приймально-здавальні) випробування проводяться на базі комп'ютерного класу кафедри або приватного приміщення в період роботи атестаційної комісії.

3.3. Обсяг випробувань

Приймальні випробування програмного вироби проводяться в обсязі відповідному цієї програми і методики випробувань.

3.4. Організації, які беруть участь у випробуваннях

Приймальні випробування проводяться атестаційною комісією напередодні засідання (або в процесі засідання) за участю Замовника, Виконавця та інших осіб, присутніх на засіданні.

4. Вимоги до програмного виробу

Програмний виріб повинен:

- забезпечувати коректну реалізацію двомасштабного фрактального згладжування моментів;
- сумісно працювати з API TensorFlow/Keras;
- стабільно навчати модель без перепадів градієнтів;
- підвищувати валідаційну точність моделі порівняно з базовими оптимізаційними алгоритмами;

5. Вимоги до програмної документації

До програмної документації належать:

− Технічне завдання на розробку програми (Додаток Б). − Програма і методика випробувань (Додаток В). − Код програмного виробу (Додаток Г). − Звіти про результати тестування (Розділ 3 пояснювальної записки).

6. Засоби і порядок випробувань

6.1. Засоби випробувань:

- Операційна система: Windows 10
- Мова програмування: Python 3.11
- Фреймворки та API: TensorFlow 2.19, Keras 3.10
- Набір даних: PathMNIST
- Обладнання: ПК з CPU не нижче Intel i3, 8 ГБ ОЗП

6.2. Порядок проведення випробувань

Тестування оптимізатора

Мета: оцінити працездатність і точність FractalMomentAdam у процесі навчання.

Кроки тестування:

1. Навчити CNN протягом 10 епох з використанням базових оптимізаторів.
2. Повторити навчання тієї ж моделі з FractalMomentAdam та різними значеннями γ.
3. Порівняти метрики точності (accuracy) та функції втрат (loss) на тренувальній і валідаційній вибірках.
4. Побудувати графіки збіжності.

Очікувані результати:

- Тренувальний процес відбувається стабільно.
- Валідаційна точність FractalMomentAdam перевищує кращі показники базових оптимізаційних алгоритмів
- Графік втрат демонструє плавну збіжність без різких коливань.

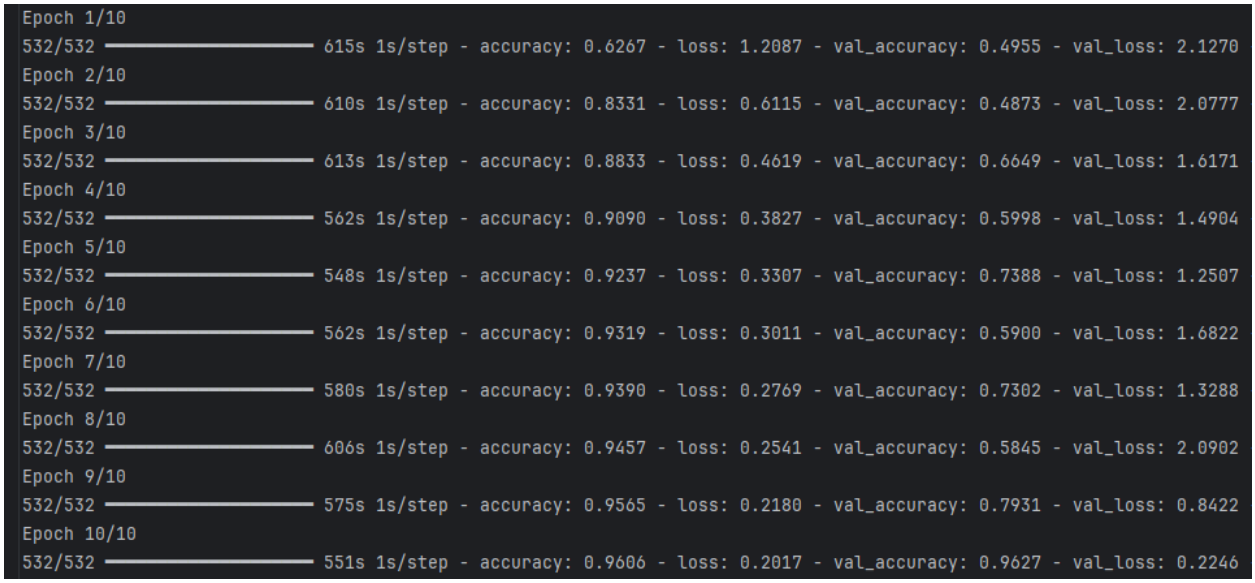


Рисунок В.1 – Результати навчання на оптимізаторі FractalMomentAdam gamma=0.6

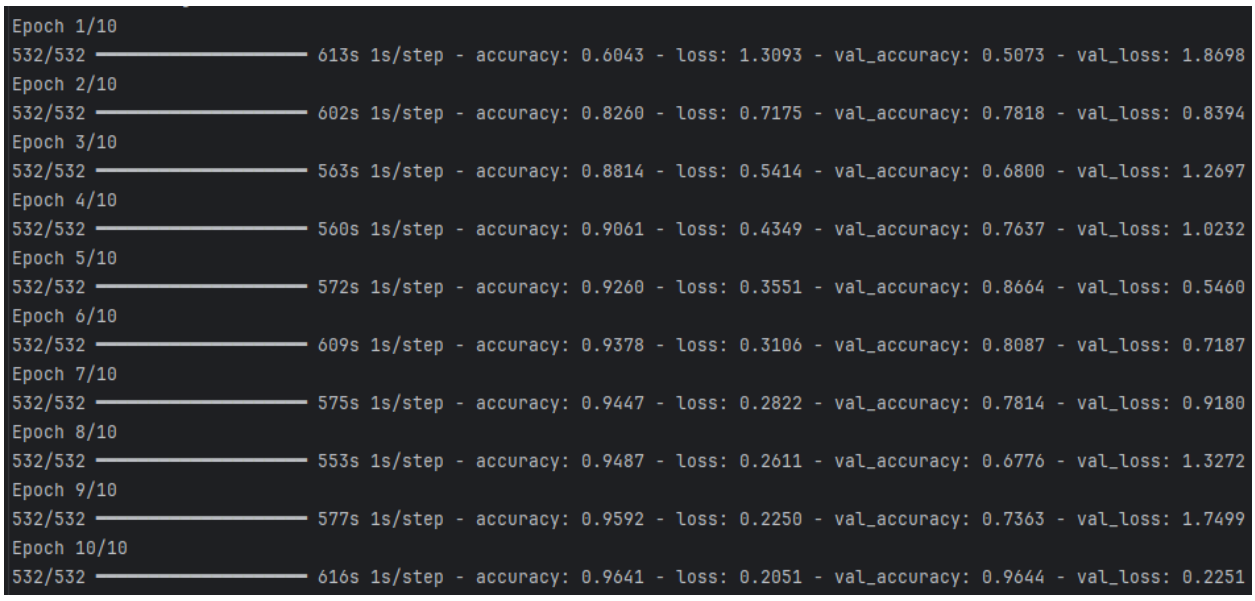


Рисунок В.2 – Результати навчання на оптимізаторі FractalMomentAdam gamma=0.2

```
Epoch 1/30
532/532 ━━━━━━━━━━━━━━━━━━━━ 620s 1s/step - accuracy: 0.6034 - loss: 1.2849 - val_accuracy: 0.3391 - val_loss: 3.3870
Epoch 2/30
532/532 ━━━━━━━━━━━━━━━━━━━━ 592s 1s/step - accuracy: 0.8158 - loss: 0.6017 - val_accuracy: 0.5912 - val_loss: 1.6306
Epoch 3/30
532/532 ━━━━━━━━━━━━━━━━━━━━ 617s 1s/step - accuracy: 0.8754 - loss: 0.4446 - val_accuracy: 0.5997 - val_loss: 1.3560
Epoch 4/30
532/532 ━━━━━━━━━━━━━━━━━━━━ 564s 1s/step - accuracy: 0.9003 - loss: 0.3717 - val_accuracy: 0.6992 - val_loss: 1.0302
Epoch 5/30
532/532 ━━━━━━━━━━━━━━━━━━━━ 569s 1s/step - accuracy: 0.9161 - loss: 0.3257 - val_accuracy: 0.5847 - val_loss: 1.6568
Epoch 6/30
532/532 ━━━━━━━━━━━━━━━━━━━━ 729s 1s/step - accuracy: 0.9257 - loss: 0.2996 - val_accuracy: 0.8246 - val_loss: 0.6400
Epoch 7/30
532/532 ━━━━━━━━━━━━━━━━━━━━ 610s 1s/step - accuracy: 0.9364 - loss: 0.2705 - val_accuracy: 0.7794 - val_loss: 0.8424
Epoch 8/30
532/532 ━━━━━━━━━━━━━━━━━━━━ 651s 1s/step - accuracy: 0.9384 - loss: 0.2619 - val_accuracy: 0.6579 - val_loss: 1.2453
Epoch 9/30
532/532 ━━━━━━━━━━━━━━━━━━━━ 604s 1s/step - accuracy: 0.9422 - loss: 0.2541 - val_accuracy: 0.3677 - val_loss: 3.3115
Epoch 10/30
532/532 ━━━━━━━━━━━━━━━━━━━━ 566s 1s/step - accuracy: 0.9570 - loss: 0.2093 - val_accuracy: 0.9401 - val_loss: 0.2662
```

Рисунок В.3 – Результати навчання на оптимізаторі Adam

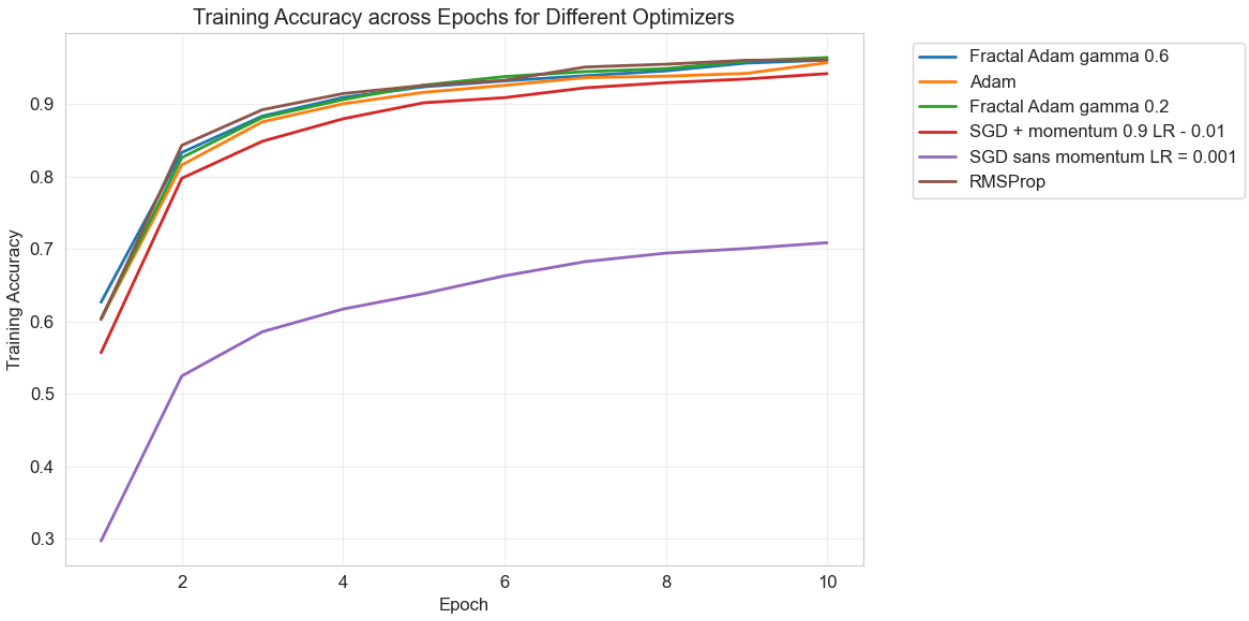


Рисунок В.4 – Графіки збіжності на тренувальній вибірці

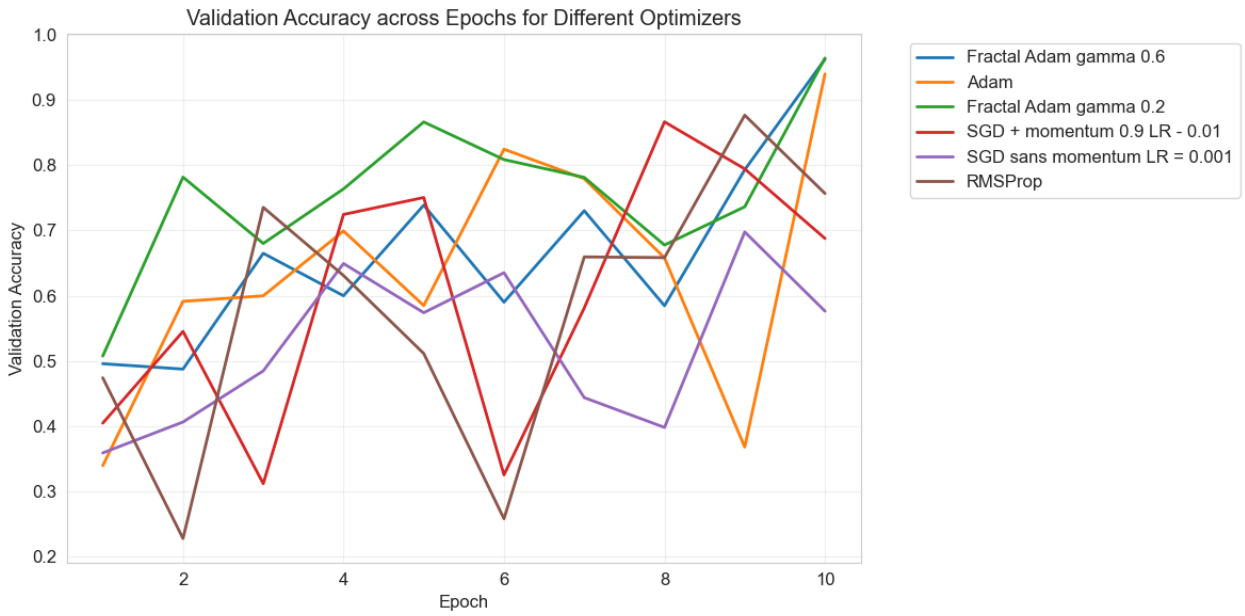


Рисунок В.5 – Графіки збіжності на валідаційній вибірці

7. Висновки

Результати випробувань показали, що програмний виріб FractalMomentAdam відповідає вимогам технічного завдання, забезпечує стабільне навчання нейронних мереж і демонструє покращену збіжність порівняно з класичними оптимізаторами.

Випробування успішно пройдені.

Виконавець:

студент групи КУ-61, Судаков Д.Г.

Додаток Г

Код реалізаціх середовища згорткової нейромережі для випробувань оптимізаторів

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, regularizers
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from medmnist import PathMNIST
from .FractalAdam import FractalMomentAdam

# Заватаження датасету
pathmnist = PathMNIST(split="train", download=True)
train_images = pathmnist.imgs
train_labels = pathmnist.labels

pathmnist_test = PathMNIST(split="test", download=True)
test_images = pathmnist_test.imgs
test_labels = pathmnist_test.labels

all_images = np.concatenate([train_images, test_images])
all_labels = np.concatenate([train_labels, test_labels])

# Розподіл вибірки на train/val/test (70%/15%/15%)
x_train, x_temp, y_train, y_temp = train_test_split(
    all_images, all_labels, test_size=0.3, random_state=42)
x_val, x_test, y_val, y_test = train_test_split(
    x_temp, y_temp, test_size=0.5, random_state=42)

# Нормалізація
x_train = x_train.astype("float32") / 255.0
x_val = x_val.astype("float32") / 255.0
x_test = x_test.astype("float32") / 255.0

# one-hot encoding
y_train = keras.utils.to_categorical(y_train, 9)
y_val = keras.utils.to_categorical(y_val, 9)
y_test = keras.utils.to_categorical(y_test, 9)

# Перевірка форми даних
print("Train shapes:", x_train.shape, y_train.shape)
print("Validation shapes:", x_val.shape, y_val.shape)
print("Test shapes:", x_test.shape, y_test.shape)


# Створення моделі
def create_model():
    model = keras.Sequential([
        # Збільшуємо розмір зображення
        layers.Resizing(32, 32),

        # Блок 1
        layers.Conv2D(32, (3, 3), activation='relu', padding='same',
                      kernel_regularizer=regularizers.l2(1e-4),
                      input_shape=(32, 32, 3)),
        layers.BatchNormalization(),
        layers.Conv2D(32, (3, 3), activation='relu', padding='same',
                      kernel_regularizer=regularizers.l2(1e-4)),
```

```
        layers.BatchNormalization(),
        layers.MaxPooling2D((2, 2)),
        layers.Dropout(0.2),

        # Блок 2
        layers.Conv2D(64, (3, 3), activation='relu', padding='same',
                      kernel_regularizer=regularizers.l2(1e-4)),
        layers.BatchNormalization(),
        layers.Conv2D(64, (3, 3), activation='relu', padding='same',
                      kernel_regularizer=regularizers.l2(1e-4)),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2, 2)),
        layers.Dropout(0.3),

        # Блок 3
        layers.Conv2D(128, (3, 3), activation='relu', padding='same',
                      kernel_regularizer=regularizers.l2(1e-4)),
        layers.BatchNormalization(),
        layers.Conv2D(128, (3, 3), activation='relu', padding='same',
                      kernel_regularizer=regularizers.l2(1e-4)),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2, 2)),
        layers.Dropout(0.4),

        # Повнозв'язні
        layers.Flatten(),
        layers.Dense(256, activation='relu',
                     kernel_regularizer=regularizers.l2(1e-4)),
        layers.BatchNormalization(),
        layers.Dropout(0.5),
        layers.Dense(9, activation='softmax')
    ])

    return model


model = create_model()

'''
Коментуємо/розкоментуємо оптимізатор на вибір
'''
# optimizer = FractalMomentAdam(
#                 learning_rate=1e-3,
#                 beta_1=0.9,
#                 beta_1_slow=0.995,
#                 gamma=0.2                           # first 0.6
#             )

# optimizer = tf.keras.optimizers.SGD(
#                 learning_rate=0.001,      # learning rate
#                 momentum=0.0,             # імпульс (0 для чистого SGD)
#                 nesterov=False            # Nesterov momentum
#             )

optimizer = tf.keras.optimizers.RMSprop(
                learning_rate=0.001,     # learning rate
                rho=0.9,                 # затухання
                momentum=0.0,            # імпульс
                epsilon=1e-07,           # стабілізатор
                centered=False           # centered RMSprop
            )
```

```
model.compile(optimizer=optimizer,
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Callbacks
callbacks = [
    keras.callbacks.EarlyStopping(patience=10, restore_best_weights=True),
    keras.callbacks.ReduceLROnPlateau(factor=0.5, patience=3)
]

# Навчання
history = model.fit(
    x_train, y_train,
    batch_size=128,
    epochs=10,
    validation_data=(x_val, y_val),
    callbacks=callbacks,
    verbose=1
)

# Оцінка моделі
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=0)
print(f"\nTest accuracy: {test_acc:.4f}")

# Графіки
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Accuracy over epochs')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Loss over epochs')
plt.legend()
plt.show()

sample_idx = 0
sample_image = x_test[sample_idx]
sample_label = y_test[sample_idx]

prediction = model.predict(np.expand_dims(sample_image, axis=0))
predicted_class = np.argmax(prediction)

print(f"\nSample prediction:")
print(f"True class: {np.argmax(sample_label)}")
print(f"Predicted class: {predicted_class}")
print(f"Confidence: {prediction[0][predicted_class]:.2%}")
```

Код реалізації оптимізаційного алгоритму FractalMomentAdam

```
import tensorflow as tf
from tensorflow.keras.optimizers import Optimizer


class FractalMomentAdam(Optimizer):
    def __init__(
        self,
        learning_rate=1e-3,
        beta_1=0.9,
        beta_2=0.999,
        beta_1_slow=0.99,
        beta_2_slow=0.9999,
        gamma=0.3,
        epsilon=1e-7,
        name="FractalMomentAdam",
        **kwargs,
    ):
        super().__init__(learning_rate=learning_rate, name=name, **kwargs)
        self.beta_1 = beta_1
        self.beta_2 = beta_2
        self.beta_1_slow = beta_1_slow
        self.beta_2_slow = beta_2_slow
        self.gamma = gamma
        self.epsilon = epsilon

        self._m_fast = {}
        self._v_fast = {}
        self._m_slow = {}
        self._v_slow = {}

    def build(self, var_list):
        for var in var_list:
            name = var.name
            self._m_fast[name] = self.add_variable_from_reference(var,
"m_fast", initializer="zeros")
            self._v_fast[name] = self.add_variable_from_reference(var,
"v_fast", initializer="zeros")
            self._m_slow[name] = self.add_variable_from_reference(var,
"m_slow", initializer="zeros")
            self._v_slow[name] = self.add_variable_from_reference(var,
"v_slow", initializer="zeros")

        self._step = self.add_variable((), dtype=tf.int64,
initializer="zeros", name="iter")
        super().build(var_list)

    def update_step(self, grad, var, learning_rate=None):
        name = var.name

        if name not in self._m_fast:
            self._m_fast[name] = self.add_variable_from_reference(var,
"m_fast", initializer="zeros")
            self._v_fast[name] = self.add_variable_from_reference(var,
"v_fast", initializer="zeros")
            self._m_slow[name] = self.add_variable_from_reference(var,
"m_slow", initializer="zeros")
            self._v_slow[name] = self.add_variable_from_reference(var,
```

```
"v_slow", initializer="zeros")

        m_f, v_f = self._m_fast[name], self._v_fast[name]
        m_s, v_s = self._m_slow[name], self._v_slow[name]

        lr = tf.cast(learning_rate if learning_rate is not None else
self.learning_rate, var.dtype)
        eps = tf.cast(self.epsilon, var.dtype)
        beta_1 = tf.cast(self.beta_1, var.dtype)
        beta_2 = tf.cast(self.beta_2, var.dtype)
        beta_1_slow = tf.cast(self.beta_1_slow, var.dtype)
        beta_2_slow = tf.cast(self.beta_2_slow, var.dtype)
        gamma = tf.cast(self.gamma, var.dtype)

        # Step
        self._step.assign_add(1)
        t = tf.cast(self._step, var.dtype)

        # Fast moments
        m_fast = beta_1 * m_f + (1 - beta_1) * grad
        v_fast = beta_2 * v_f + (1 - beta_2) * tf.square(grad)

        # Slow moments
        m_slow = beta_1_slow * m_s + (1 - beta_1_slow) * grad
        v_slow = beta_2_slow * v_s + (1 - beta_2_slow) * tf.square(grad)

        # Merge by scale
        m_eff = gamma * m_fast + (1 - gamma) * m_slow
        v_eff = gamma * v_fast + (1 - gamma) * v_slow

        # bias correction
        m_eff_hat = m_eff / (1 - tf.pow(beta_1, t))
        v_eff_hat = v_eff / (1 - tf.pow(beta_2, t))

        var.assign_sub(lr * m_eff_hat / (tf.sqrt(v_eff_hat) + eps))

        # Renew slots
        m_f.assign(m_fast)
        v_f.assign(v_fast)
        m_s.assign(m_slow)
        v_s.assign(v_slow)

    def get_config(self):
        base_config = super().get_config()
        return {
            **base_config,
            "learning_rate": self._serialize_hyperparameter("learning_rate"),
            "beta_1": self.beta_1,
            "beta_2": self.beta_2,
            "beta_1_slow": self.beta_1_slow,
            "beta_2_slow": self.beta_2_slow,
            "gamma": self.gamma,
            "epsilon": self.epsilon,
        }
```