Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Удосконалений алгоритм ADAM для навчання нейромережі»

Виконав: студент 2-ого курсу групи 1ІСТ-21м
спеціальності 126 – Інформаційні системи та технології_____
Данилевич М. О._____

Керівник: к.т.н., доц. каф. АІІТ_____
Іванов Ю. Ю._____
«____» _____ 2022 р.

Опонент: _____

_____
«____» _____ 2022 р.

Допущено до захисту
в.о. зав. кафедри АІІТ
д.т.н., проф. Бісікало О. В.
«____» _____ 2022 р.

Вінниця ВНТУ – 2022 рік

Вінницький національний технічний університет

Факультет інтелектуальних інформаційних технологій та автоматизації

Кафедра автоматизації та інтелектуальних інформаційних технологій

Рівень вищої освіти II-ий (магістерський)

Галузь знань – 12 – Інформаційні технології

Спеціальність – 126 – Інформаційні системи та технології

Освітньо-професійна програма – Інформаційні технології аналізу даних та зображень

ЗАТВЕРДЖУЮ

в. о. завідувача кафедри АІІТ

д.т.н., проф. Бісікало О. В.

«___»_____ 2022 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Данилевичу Максиму Олександровичу

1. Тема роботи: Удосконалений алгоритм ADAM для навчання нейромережі.
   Керівник роботи: к.т.н., доцент каф. АІІТ Іванов Ю. Ю.
   Затверджені наказом ВНТУ від «14» 09 2022 року № 203.
2. Строк подання роботи студентом: до «до 23» 12 2022 р.
3. Вихідні дані до роботи: вибірка даних; архітектура нейромережі (кількість нейронів та прошарків; структурні особливості); оптимізатор ADAM (гіперпараметри: коефіцієнти навчання α та згладжування ε; оцінки моментів β_1 та β_2).
4. Зміст розрахунково-пояснювальної записки: вступ; огляд предметної області роботи; розробка математичного апарату удосконаленого алгоритму ADAM; розробка програмного забезпечення та експериментальні дослідження; економічний розділ; список використаних джерел; висновки.
5. Перелік графічного матеріалу: структура одношарової штучної нейромережі; структура багатошарової штучної нейромережі; структура для навчання нейромережі; нейромережеві оптимізатори; принцип роботи градієнтних алгоритмів оптимізації; схема програми.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Іванов Ю. Ю., к.т.н., доцент каф. АІІТ	19.09.2022	07.12.2022
4	Лесько О. Й. к.е.н., проф. каф. ЕПтаВМ		

7. Дата видачі завдання: «19»_____09_____ 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Огляд предметної області	10.10.22	
2	Розробка математичної моделі удосконаленого алгоритму ADAM	25.10.22	
3	Розробка програмного забезпечення та експериментальні дослідження	10.11.22	
4	Підготовка економічного розділу	20.11.22	
5	Оформлення пояснювальної записки і графічного матеріалу	30.11.22	
6	Попередній захист роботи	до 07.12.22	
7	Остаточний захист роботи	до 23.12.22	

Студент _____ Данилевич М. О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Іванов Ю. Ю.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.032.26 + 004.942

Данилевич М. О. Удосконалений алгоритм ADAM для навчання нейромережі. Магістерська кваліфікаційна робота зі спеціальності 126 – Інформаційні системи та технології, освітньо-професійна програма – Інформаційні технології аналізу даних та зображень. Вінниця: ВНТУ, 2022. 91 с.

На укр. мові. Бібліогр.: 31 назва; рис.: 21; табл.: 11.

У роботі проаналізовано особливості роботи нейромереж, розглянуто відповідний математичний апарат та удосконалено алгоритм оптимізації ADAM. Також розроблено програмне забезпечення та проведено низку експериментів для підтвердження ефективності запропонованої розробки.

Ключові слова: нейромережі, оптимізація, екстремум, функція помилки, ADAM, корегуюча складова.

ABSTRACT

Danylevich M. O. Improved ADAM algorithm for neural network's learning. Master's thesis in specialty 126 – Information systems and technologies, educational and professional program – Information technologies of data and image analysis. Vinnitsa: VNTU, 2022. 91 p.

In Ukrainian language. Bibliography: 31 titles; fig.: 21; tabl.: 11.

In this work have been analyzed main features of neural networks, also has been considered corresponding mathematical apparatus and has been modified an ADAM optimization algorithm. Software has been developed and a number of experiments have been conducted to confirm the effectiveness of the proposed development.

Keywords: neural networks, optimization, extremum, error function, ADAM, corrective component.

ЗМІСТ

ВСТУП

Штучні нейронні мережі представляють собою перспективну область досліджень, оскільки дозволяють розв'язувати різноманітні прикладні задачі у медицині, економіці, робототехніці тощо. При цьому досить часто нейромережі виявляються ефективніші за інші математичні методи. У ході розв'зання складних прикладних задач все частіше виникає потреба у використанні глибокого навчання, що вимагає великих обчислювальних ресурсів. Тому успіх нейромереж також корелює з розвитком технологій прискорених обчислень [1].

Важливе значення в успіху нейромереж відіграє їх здатність до навчання, що дозволяє виявляти складні залежності в наданій інформації, проводити узагальнення. При цьому нейронні мережі нечутливі до незначних змін вхідних образів, шумів і спотворень вхідної інформації, що дозволяє їм легко адаптуватися в умовах змінного зовнішнього середовища.

У зв'язку з широкою областю застосування нейромереж формуються різноманітні задачі, які відрізняються постановкою і типами вхідних даних: розпізнавання зображень, синтаксичний аналіз текстів, діагностика захворювань тощо [2-4]. Оскільки всі оптимізатори відрізняються своїми властивостями і особливостями реалізації, часто виникає проблема визначення найбільш ефективного алгоритму для пошуку екстремуму функції помилки, що гарантує кращі результати при вирішенні конкретної задачі. Сучасні пакети машинного навчання для розв'язання різноманітних практичних задач використовують різні варіації градієнтного спуску для

оптимізації нейронної мережі, які володіють високою ефективністю за рахунок низки специфічних механізмів [5-11].

Можна зазначити, що відбувається збільшення кількості досліджень за даною тематикою науковими групами по всьому світу, включаючи такі великі компанії, як Google, Facebook, Amazon та інші. Зокрема у наукових працях О.В. Чопорової [5], С. Рудера [7], А. Жанга [8], Д.П. Кінгми та Дж.Л. Ба [9] та інших. У цих роботах виділяється ефективний алгоритм оптимізації на основі адаптивної оцінки моментів (ADAM – adaptive moment estimation), який, за рахунок швидкої збіжності та високої точності, став найбільш поширеним для використання у сучасному програмному забезпеченні для роботи з нейромережами. Отже, актуальною практичною задачею є удосконалення та дослідження поведінки даного алгоритму у ході розв'язання практичних задач.

Мета і задачі дослідження. Метою роботи є підвищення ефективності розв'язання задачі навчання штучної нейромережі за рахунок використання удосконаленого алгоритму оптимізації ADAM.

Для досягнення мети необхідно розв'язати наступні задачі:

- проаналізувати особливості роботи штучних нейромереж;
- розробити математичну модель удосконаленого оптимізатора;
- розробити програмні засоби та оцінити ефективність розробки.

Об'єктом дослідження є процес навчання нейромережі.

Предметом дослідження є моделі, методи та інструментальні засоби для розв'язання задачі навчання штучної нейромережі.

Методи дослідження. У роботі використано елементи прикладної математики, теорії штучного інтелекту та оптимізації для аналізу та

розробки оптимізатора; експериментальне дослідження застосовано для перевірки достовірності отриманих результатів. Основним засобом розробки є мова програмування *Python*.

Наукова новизна отриманих результатів полягає у наступному: запропоновано удосконалення алгоритму оптимізації ADAM, особливістю якого є уточнений розрахунок ваг за рахунок обчислення корегуючої складової, що дозволяє підвищити ефективність роботи нейромережі.

Практичне значення результатів роботи полягає в тому, що розроблено математичне, алгоритмічне та програмне забезпечення, яке дає можливість виконувати навчання нейромереж.

Апробація результатів та публікації. За результатами роботи опубліковано 2 тези доповіді на:

– L науково-технічній конференції підрозділів ВНТУ (м. Вінниця, 2021) [12];

– науково-практичній конференції з міжнародною участю "Актуальні задачі медичної, біологічної фізики та інформатики" (м. Вінниця, 2022) [13].

Основні результати можна використати на практиці, що підтверджено довідкою про можливість впровадження розробки.

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ РОБОТИ

1.1 Огляд особливостей роботи штучних нейромереж

Штучна нейронна мережа (ANN – artificial neural network) представляє собою обчислювальну архітектуру для обробки складних даних за допомогою множини пов'язаних між собою шарів штучних нейронів, які організовують загальну активну структуру та функціонально впливають на роботу один одного. Штучні нейронні мережі здатні навчатися та аналізувати великі та складні набори даних, які за допомогою більш лінійних алгоритмів обробити вкрай складно. Загальний вигляд штучного нейрону наведено на рисунку 1.1 [3, 4].

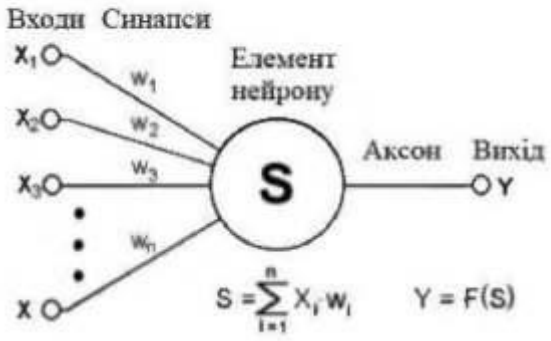

Рисунок 1.1 – Модель штучного нейрона

Основною характеристикою штучного нейрона є його стан, що дозволяє провести аналогію з нервовими клітинами головного мозку. Кожен штучний нейрон володіє групою синапсів та аксоном. Синапс – це односпрямований вхідний зв'язок, який поєднаний з виходом іншого нейрона, а аксон – вихідний зв'язок, що надходить на синапси наступних

нейронів. У сукупності вхідні сигнали позначаються вектором. Кожен синапс характеризується величиною синаптичного зв'язку (іншими словами, його вагою) w_i. Активність штучного нейрона в більшості архітектур визначається, як перетворення зовнішнього сумарного впливу інших нейронів на даний нейрон, та керується функцією активації. Частіше за все функції активації є фіксованими, а ваги синапсів слугують параметрами мережі [8].

Активність нейрона є нелінійною. Для реалізації нелінійності використовується функція активації, яка застосовується до зваженої суми постсинаптичних сигналів на вході нейрона. Існує ряд функцій активації, які використовуються в залежності від об'єкта, що аналізується нейронною мережею, типу нейронної мережі тощо [14].

Наприклад, простим варіантом є функція Хевісайда, яка використовується в класичному перцептроні. На даний момент вона не є поширеною на практиці, а застосовується лише для вивчення теорії нейронних мереж та моделювання найпростіших випадків. Ще однією простою та рідко застосовуваною в практичних реалізаціях є лінійна функція активації [14-16].

Однією з найпоширеніших передавальних функцій є сигмоїдна функція активації. За рахунок того, що сигмоїд – монотонно зростаюча, диференційовна на всьому проміжку нелінійна функція з насиченням, дана функція активації забезпечує посилення слабких сигналів та запобігає насиченню сильних. Її введення було зумовлено недостатньою гнучкістю класифікаторів, які працювали на основі порогових передавальних функцій. Використання сигмоїди дозволило перейти від жорсткої

однорозрядної логіки до більш гнучкої параметризації нейромереж.

На даний момент однією з найефективніших і найпростіших з точки зору обчислювальної складності є випрямлена лінійна функція ReLu, яка використовується при навчанні глибоких нейромереж, оскільки, на відміну від нелінійних (сигмоїдна, гіперболічний тангенс) функцій не призводить до проблем із згасанням та збільшенням градієнтів.

Для забезпечення кращого результату роботи нейрони об'єднуються в мережі. Найпростішою є одношарова мережа, яка складається з сукупності нейронів, утворення яких формує шар (рисунок 1.2).

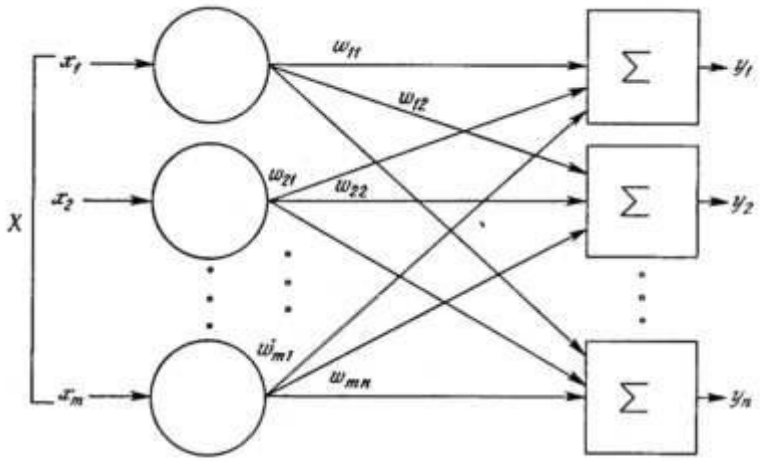

Рисунок 1.2 – Проста одношарова штучна нейронна мережа

Ліві вершини слугують для розподілу сигналів X, що подаються на вхід. Оскільки за ними не закріплено жодних обчислень, то вони не вважаються шаром і позначаються колами. Обчислювальні нейрони позначаються квадратами. Між окремою вагою кожного елементу з множини входів та кожним штучним нейроном існує сполучення. Ваги представляються елементами двовимірної матриці W розмірністю $n \times m$, де n – кількість нейронів, а m – кількість виходів.

Обчислення вихідного вектора зводиться до матричного множення:

$$Y = XW\ . \tag{1.1}$$

Для розв'язання більш складних задач використовуються багатошарові нейронні мережі, тобто ті, які містять багато шарів нейронів. Спосіб взаємозв'язків нейронів часто визначає відмінності в обчислювальних процесах, які відбуваються в нейронній мережі. За однією з класифікацій сучасні багатошарові архітектури поділяються на статичні та динамічні [8].

Архітектура згорткових нейронних мереж використовується для ефективного розпізнавання зображень. Вона базується на чергуванні згорткових шарів з нелінійними функціями активації та шарів об'єднання або підвибірки. В згорткових мережах для отримання вхідних значень застосовується операція згортки, що і стало причиною такої назви даної архітектури нейронної мережі [17-19].

Операція згортки використовує матрицю ваг невеликого розміру, яка проходить через поточний шар та формує, після кожного зсуву, сигнал активації для нейрона наступного шару з аналогічною позицією. Матриця носить назву ядра згортки і використовується для різних нейронів вихідного шару. Згорткові нейронні мережі:

- мають вхід фіксованого розміру;
- генерують виходи фіксованого розміру;
- використовують зворотний зв'язок;

– використовують схему взаємодії між нейронами, подібно до організації зорової кори тварин.

За рахунок наведених особливостей згорткові нейронні мережі використовуються для обробки відео та зображень. Звичайні штучні нейронні мережі (рисунок 1.3) погано масштабуються для роботи з зображеннями великих розмірів, повнозв'язність нейронів у даному випадку призводить до швидкого переповнення пам'яті та, як наслідок, недієздатності нейронної мережі в межах існуючих комп'ютерних архітектур. Крім того, величезна кількість параметрів швидко призводить до проблеми перенавчання [19].

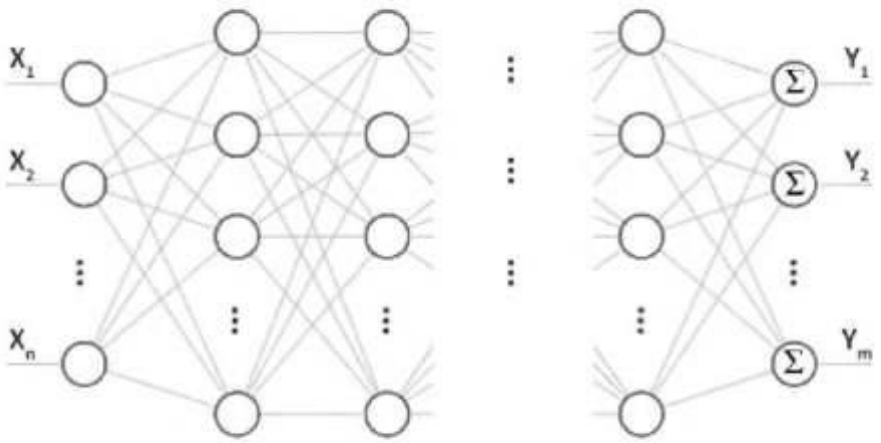

Рисунок 1.3 – Багатошарова нейронна мережа

При розробці згорткових нейронних мереж (рисунок 1.4) було враховано, що вхідними даними є зображення, тому побудова мережі обмежується. Шари згорткової нейронної мережі складаються з нейронів, розташованих у тривимірному просторі (у вимірах ширини, висоти і глибини, тобто формується об'єм). Нейрони підключаються лише до невеликої області шару, а до кінця побудови нейромережі зображення

перетворюється в єдиний вектор оцінок класу, розташованих за вимірюванням глибини [12].

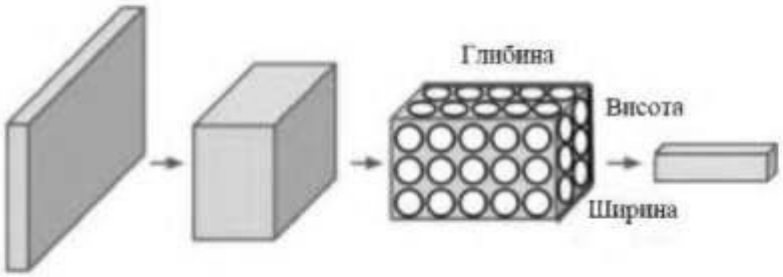

Рисунок 1.4 – Згорткова нейронна мережа

У згортковій нейронній мережі також використовуються і повнозв'язні шари, присутні на рівні класифікатора. Нейрони повнозв'язного шару мають зв'язок з усіма функціями активації попереднього шару. Основною відмінністю шару класифікації та шару згортки є те, що нейрони останнього шару об'єднані лише з локальною областю на вході і можуть спільно використовувати параметри.

В загальному, при використанні згорткових нейронних мереж, якщо зробити припущення, що зображення є двовимірними функціями, то різні перетворення, які здійснюються над ними, є операцією згортки функції зображення з локальною функцією (ядром згортки). Операція згортки чергується з операцією підвибірки, що, в свою чергу, призначена для зменшення загального розміру зображення і збільшення ступеня інваріантності застосованих до нього фільтрів. Проходження кількох шарів перетворює карту ознак в вектор або скаляр, що значно спрощує обробку карт ознак з допомогою повнозв'язної нейронної мережі [3].

Першою запропонованою згортковою моделлю була модель Яна Лекуна – *LeNet*. Вона містила два згорткові шари, шари субдискретизації, які чергуються, а також три повнозв'язних шари. Дана архітектура з параметрами, наведеними на рисунку 1.5, вважається класичною [15, 18].

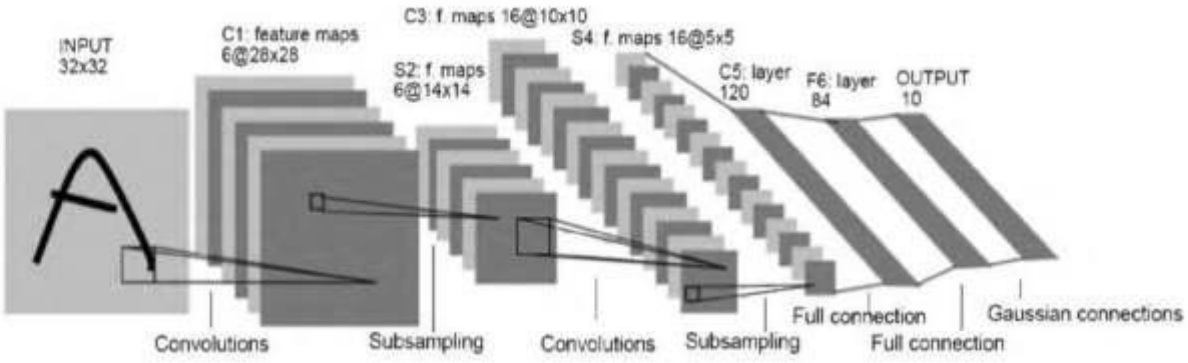

Рисунок 1.5 – Схема архітектури LeNet

Наступною значною мережею була мережа AlexNet, запропонована Алексом Крижевським [15, 18]. Вона була дуже схожою на мережу LeNet, однак відрізнялася від неї більш масивною і складною архітектурою. У її складі був лише один згортковий шар і кілька шарів субдискретизації з операцією вибору максимального значення, 96 фільтрів (рисунок 1.6).

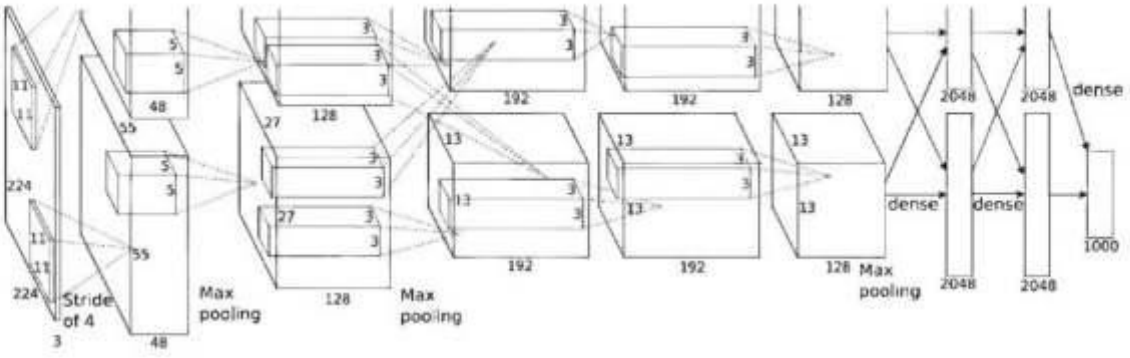

Рисунок 1.6 – Схема архітектури AlexNet

Наступний значний внесок в розвиток архітектур внесла мережа VGGNet. Дана мережа складається з великої кількості шарів, які чергуються і мають малі розміри (3x3 – розмір фільтрів, 2x2 – розмір шару субдискретизації). Негативною стороною є те, що вона зберігає до 140 мільйонів параметрів, а це робить її дуже громіздкою і низькопродуктивною (рисунок 1.7) [15, 18].

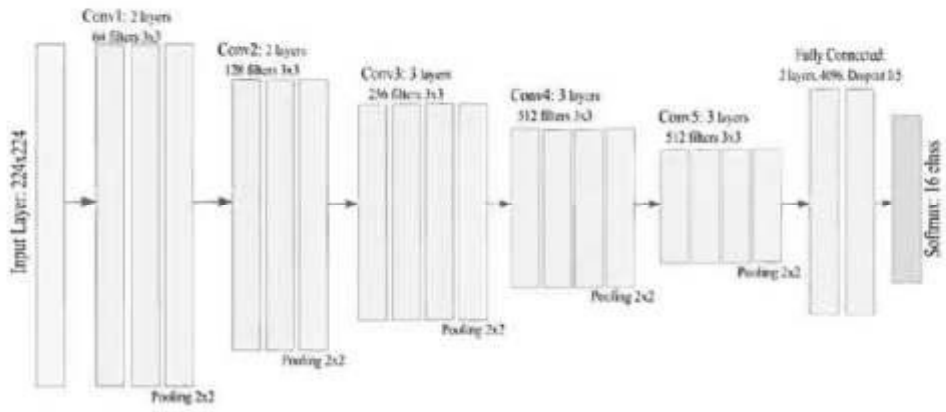

Рисунок 1.7 – Архітектура мережі VGGNet

Отже, оскільки архітектура згорткових нейронних мереж призначена для ефективного розпізнавання зображень, доцільним буде її застосування в межах розв'язання задачі мультикласифікації зображень, що розглядається в даній роботі. Математичний апарат згорткової нейронної мережі та алгоритмів її навчання наведено у наступному розділі.

1.2 Нейромережева оптимізація

Нейронні мережі часто навчаються (тобто оптимізується цільова функція помилки), використовуючи на кожній новій ітерації різні частини

даних (батчі). Це мотивовано, як мінімум, двома причинами: по-перше, набори даних, які використовуються для навчання, часто дуже великі, щоб зберігати їх повністю в оперативній пам'яті і / або робити обчислення ефективно; по-друге, цільова функція зазвичай невипукла, таким чином, використання різних частин даних на кожній ітерації може допомогти від застрявання моделі в локальному мінімумі. Існує три основні групи нейромережевих оптимізаторів [5-11]:

– методи оптимізації першого порядку;

– методи оптимізації другого порядку;

– стохастичні методи оптимізації.

Методи оптимізації першого порядку (рисунок 1.8) знаходять екстремум функції втрат E, використовуючи значення градієнта по відношенню до параметрів.

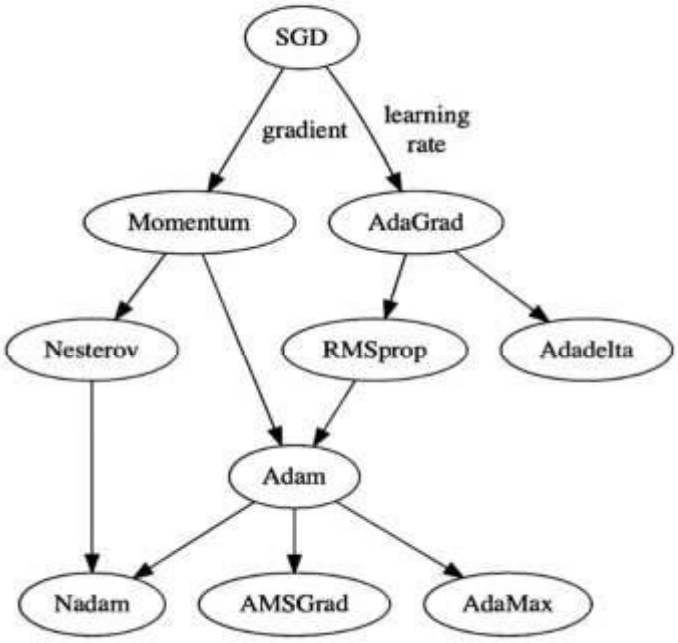

Рисунок 1.8 – Градієнтні нейромережеві оптимізатори першого порядку

Градієнтом $\nabla E(x)$ неперервної диференційовної функції $E(x)$ називається вектор, який містить частинні похідні першого порядку, обчислені в точці x:

$$\nabla E(x) = \left(\frac{\partial E(x)}{\partial x_1} \quad \frac{\partial E(x)}{\partial x_2} \quad \dots \quad \frac{\partial E(x)}{\partial x_n} \right)^T . \tag{1.2}$$

Градієнт $\nabla E(x)$ показує напрям до максимуму, антиградієнт $-\nabla E(x)$ – до мінімуму. Довжина градієнта (норма) дорівнює швидкості зміни $\nabla E(x)$, тобто

$$\left\| \nabla E(x) \right\| = \sqrt{\sum_{i=1}^{n} \left(\frac{\partial E(x)}{\partial x_i} \right)^2} . \tag{1.3}$$

Найбільш відомим представником даної групи є стохастичний градієнтний спуск (SGD – Stochastic Gradient Descent), який виконує оновлення ваг нейромережі за виразом

$$w^{(k+1)} = w^{(k)} + \eta \cdot \nabla E(w^{(k)}), \tag{1.4}$$

де $w^{(k)}$, $w^{(k+1)}$ – поточна та обчислена вага синапсу; η – довжина кроку пошуку; k – номер ітерації.

Але даний метод може збігатися дуже повільно і не отримати прийнятного результату, якщо швидкість навчання η налаштована неакуратно. Тому існує багато альтернативних методів, які розроблені з

метою прискорення збіжності навчання і позбавлення користувача від необхідності ретельного налаштування гіперпараметрів мережі (значення глобальних параметрів, які встановлюються перед запуском процесу навчання мережі). Загальна ідея роботи подібних методів зображена на рисунку 1.9.

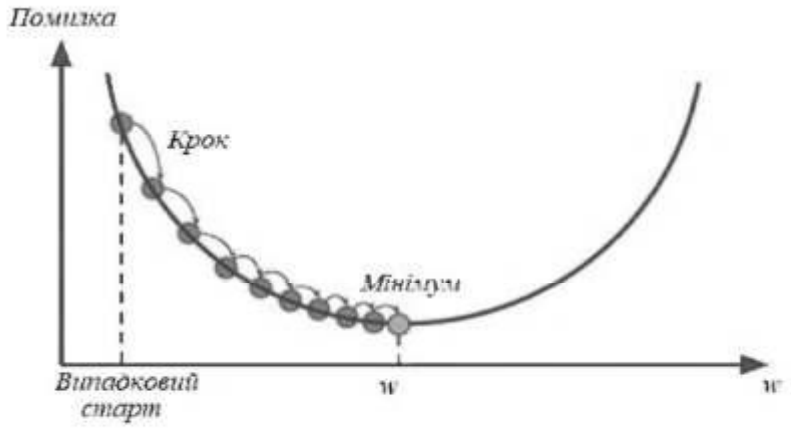

Рисунок 1.9 – Принцип роботи градієнтних методів оптимізації

Алгоритм Adagrad використовує адаптивну швидкість навчання η на основі минулих градієнтів, які були обчислені для даного параметра, тобто основною перевагою є те, що не потрібно налаштовувати швидкість навчання вручну. Основним недоліком є те, що швидкість навчання постійно зменшується, що у результаті спричиняє зупинку в певному локальному оптимумі без можливості з нього вибратися [7].

Алгоритм AdaDelta розроблений для усунення проблеми погіршення швидкості навчання Adagrad. Замість того щоб накопичувати всі попередні квадратні градієнти, Adadelta обмежує накопичення минулих градієнтів до вікна деякого фіксованого розміру. Алгоритми RMSProp і Adadelta були розроблені практично одночасно і незалежно один від одного. Вони схожі,

за винятком того, що коефіцієнт швидкості навчання в Adadelta замінюється на корінь із середньої суми квадратів зміни ваги. У зв'язку з цим, немає необхідності встановлювати швидкість навчання за замовчуванням [7].

Алгоритм адаптивної оцінки моментів (Adam – Adaptive Moment Estimation) – найпопулярніший на даний момент алгоритм оптимізації нейромереж, який вирізняється двома ідеями: по-перше, оцінка першого моменту обчислюється як ковзне середнє; по-друге, через те, що оцінки першого і другого моментів ініціалізуються нулями, використовується невелика корекція, щоб результуючі оцінки не були зміщені до нуля. Алгоритм також інваріантний до масштабування градієнтів і збігається до розв'язку дуже швидко [7, 9].

Алгоритм Adamax використовує ідею заглядання вперед по вектору оновлення. Ідея полягає в тому, щоб змусити квадрат градієнта збільшуватися, додавши ще одну змінну для відстеження максимуму.

Методи другого порядку використовують похідну другого порядку для того, щоб мінімізувати або максимізувати функцію втрат. Відомими представниками цієї групи є алгоритми Левенберга-Марквардта та Бройдена- Флетчера-Гольдфарба-Шанно. Похідна другого порядку показує досліднику, збільшується або зменшується перша похідна, що вказує на кривизну функції. Оскільки друга похідна є досить коштовною для обчислень на комп'ютері, то такі алгоритми вкористовуються рідко. У даних алгоритмах необхідно знайти гессіан H, який представляє собою матрицю частинних похідних другого порядку [17]:

$$H(x)=\begin{pmatrix} \frac{\partial^2 E(x)}{\partial x_1^2} & \frac{\partial^2 E(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 E(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 E(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 E(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 E(x)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 E(x)}{\partial x_n \partial x_1} & \frac{\partial^2 E(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 E(x)}{\partial x_n^2} \end{pmatrix}. \quad (1.5)$$

Оскільки задача полягає в пошуку глобального оптимум функції втрат E серед величезної кількості локальних оптимумів, то доцільно застосовувати алгоритми стохастичної оптимізації (випадковий пошук рішень), яка часто дозволяє знайти хороші рішення за прийнятний час. Наприклад, алгоритм модельного загартування, еволюційні алгоритми, ройовий інтелект, який включає алгоритми кажанів, бджолиної колонії, зграї сірих вовків, котів тощо [19].

1.3 Висновки

У даному розділі розглянуто основні особливості нейромереж, із яких виділяється згорткова нейронна мережа. Визначено, що оптимізація функції помилки нейромережі зазвичай виконується за допомогою градієнтних методів першого порядку, оскільки через велику кількість параметрів нейромережі неможливо ефективно застосовувати методи більш високих порядків. Слід зазначити, що іноді можна застосовувати стохастичні методи оптимізації на основі ройового інтелекту.

2 РОЗРОБКА МАТЕМАТИЧНОГО АПАРАТУ УДОСКОНАЛЕНОГО АЛГОРИТМУ ADAM

2.1 Математичний апарат для розробки згорткової нейромережі

Процес функціонування нейронної мережі поділяється на навчання та використання вже натренованої нейромережі з метою отримання певного результату (наприклад, до якого класу належить зображення, подане на вхід мережі, прогнозування значення тощо). У першому випадку для згорткової нейронної мережі передбачається два етапи: пряме проходження, яке абсолютно ідентичне другому варіанту використання мережі (за винятком порівняння отриманої відповіді з відомою), та процес коригування значень ваг, тобто навчання нейронної мережі або оптимізація функції втрат E (помилок) [1].

Розглянемо архітектуру згорткової нейронної мережі та процес прямого проходження, який є обов'язковим при будь-якому варіанті використання нейронної мережі. Основу згорткових нейронних мереж складають шари, кожен з яких характеризується простим набором задач: перетворює вхідні дані у вигляді тривимірного об'єму у вихідний тривимірний об'єм з деякою диференційованою функцією з набором параметрів або без них.

Згорткова нейронна мережа (Convolutional Neural Network) – це особлива архітектура штучної нейронної мережі, яка імітує особливості

зорової кори головного мозку, складається з декількох багатовимірних шарів і призначена для ефективного розпізнавання складних зображень.

Вперше така модель була запропонована Я. Лекуном у 1998 році і призначалася для розпізнавання рукописних символів. Особливість даної моделі нейронної мережі полягала у введенні в архітектуру багатошарового перцептрона низки згорткових шарів, в яких кожен нейрон був пов'язаний тільки з невеликою кількістю нейронів попереднього шару [13].

Для задач розпізнавання об'єктів на зображеннях вхідний шар найчастіше представляється у вигляді тривимірної сітки, розміри якої залежать від вхідного зображення, що можна представити у вигляді

$$I = W \cdot H \cdot D, \tag{2.1}$$

де I – розмірність вхідного шару мережі; W та H – ширина та висота вхідного зображення; D – кількість колірних каналів зображення.

На рисунку 2.1 представлено умовний вигляд вхідного шару згорткової нейронної мережі, для якого зображення задано з використанням кольорової моделі *RGB*.

Згортковий шар нейронної мережі призначений для виділення ознак зображення та їх перетворень, які, в свою чергу, на більш глибоких шарах мережі, використовуються для отримання більш складних ознак, а у підсумку визначають клас об'єкта.

Основною характеристикою даного шару є так звані фільтри – багатовимірні (зазвичай двовимірні або тривимірні) матриці ваг зв'язків нейронів попереднього шару з нейронами згорткового шару. Фільтрами

вони називаються тому, що операція згортки, тобто отримання вихідного сигналу нейрона згорткового шару, дуже схожа на операцію фільтрації зображення [8].

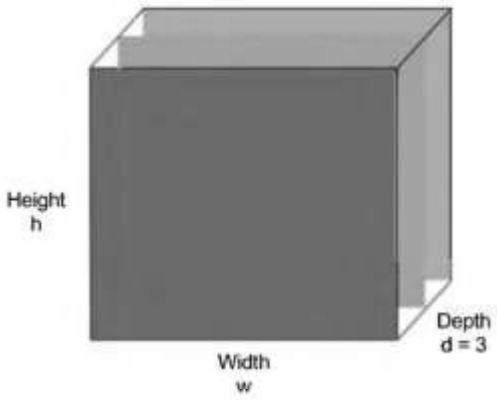

Рисунок 2.1 – Умовний вигляд вхідного шару нейромережі

Кожне значення сигналу нейрона попереднього шару, розташованого в певній області, множиться на відповідне значення ядра фільтра (в нейромережі значеннями ядра фільтра є ваги зв'язків нейронів згорткового шару w_{00}, w_{01}, …, $w_{(F-1)(F-1)}$, де F – розмір квадратного фільтра), а результатом фільтрації є сума всіх отриманих добутків [20].

Шар згортки є головним шаром нейронної мережі. Його параметрами є набір фільтрів для навчання. Просторові габарити фільтра (тобто, ширина та висота) є невеликими, але фільтр проходить по всій глибині об'єму. Наприклад, стандартним фільтром першого шару згорткової нейронної мережі може бути фільтр розмірністю $[5\times5\times3]$. Під час проходження вперед фільтр рухається по ширині та висоті вхідних даних. У процесі цього руху обчислюється скалярний добуток між записами фільтра і входом для будь-якого положення.

Результатом роботи фільтра є двовимірна активаційна карта, яка вказує на результат обчислень фільтра для кожної просторової позиції. Під час навчання нейронної мережі значення ваг фільтру коригується для розпізнавання конкретного виду зображень, з якими працює дана нейронна мережа. Приклад операції згортки наведено на рисунку 2.2.

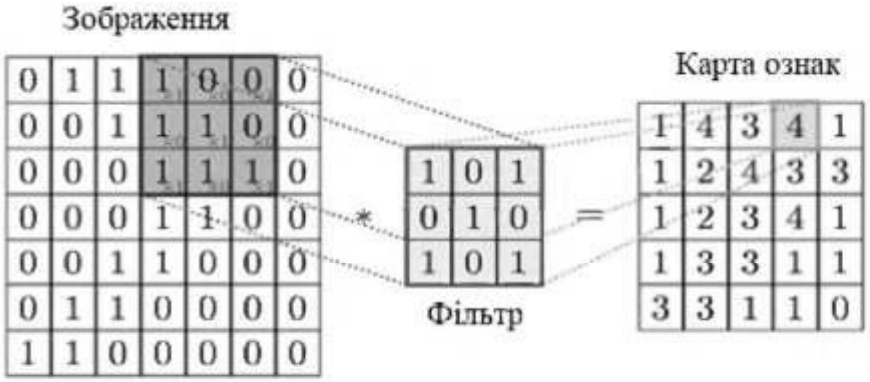

Рисунок 2.2 – Операція згортки

Формула згортки відображає "рух" ядра w^l по вхідному зображенню або по карті ознак y^{l-1}. В загальному вигляді формула згортки виглядає наступним чином

$$x_{ij}^l = \sum_{a=-\infty}^{\infty} \sum_{b=-\infty}^{\infty} w_{ab}^l \cdot y_{(i \cdot s-a)(j \cdot s-b)}^{l-1} + b^l, \tag{2.2}$$

де i, j, a, b – індекси елементів в матрицях; $s\,(stride)$ – величина кроку згортки; l – індекс шару мережі; x^{l-1} – вихід попередньої функції або вхідне зображення мережі; y^{l-1} – це x^{l-1} після проходження функції активації; w^l – ядро згортки; b^l – зсув, x^l – результат операції згортки.

Зазначимо, що виділяються три гіперпараметри мережі, які контролюють розмір вихідної інформації [20]:

Гіперпараметр "глибина" (deep) є еквівалентним кількості використовуваних фільтрів, причому кожен фільтр навчається пошуку різних ознак на вході.

Гіперпараметр "крок" (stride) визначає величину зсуву фільтру на кожному кроці та виконує безпосередній вплив на згортку. Чим більшим є значення даного гіперпараметру, тим меншим є розмір вихідної матриці.

Гіперпараметр "заповнення нулями" (padding) передбачає ситуацію, коли необхідні дані поза межами вхідного зображення. В такому випадку матриця пікселів зображення може доповнюватись нульовими елементами до потрібної розмірності.

Параметрами згорткового шару є: кількість фільтрів K, розмір фільтра F, крок S та кількість заповнень нулями P. Дані параметри контролюють кількість вихідної інформації. Розмірність згорткового шару $[W_2, H_2, D_2]$ визначається його параметрами і розмірами попереднього шару мережі, тобто [21]

$$\begin{cases} W_2 = \dfrac{(W_1 - F + 2P)}{S} + 1, \\ H_2 = \dfrac{(H_1 - F + 2P)}{S} + 1, \\ D_2 = K, \end{cases} \tag{2.3}$$

де W_2 і H_2 – ширина і висота згорткового шару відповідно; W_1 і H_1 – ширина і висота попереднього шару відповідно; F – розмір квадратного фільтру

згорткового шару мережі; P – кількість доданих нулів по краях попереднього шару; S – зміщення фільтра.

Для формування вихідного сигналу нейронів згорткового шару в нейромережі використовуються функції активації нейрона, які за певними правилами перетворюють сигнал x.

Зворотно-згортковий шар є протилежним шаром до згорткового шару, оскільки він використовується для збільшення розмірності попереднього шару та відновлення виділених попередніми шарами ознак.

Саме тому даний тип шарів широко застосовується у автокодерах та згорткових нейронних мережах, які використовуються для сегментації та детектування об'єктів на зображеннях. Зворотно-згортковий шар має ті самі параметри, що і звичайний згортковий шар, тому впродовж тренування мережі теж "навчається" [21]. Розмірність зворотно-згорткового шару визначається його параметрами і розмірами попереднього шару мережі, а отримані вихідні значення подаються, як аргументи, у функції активації, тобто

$$W = (W_p - 1) \cdot S + F - 2 \cdot P, \tag{2.4}$$

$$H = (H_p - 1) \cdot S + F - 2 \cdot P, \tag{2.5}$$

де W і H – ширина і висота зворотно-згорткового шару відповідно; W_p і H_p – ширина і висота попереднього шару відповідно; F – розмір квадратного фільтру зворотно-згорткового шару мережі; P – значення доповнення; S – зміщення фільтра.

У згорткових мережах для класифікації використовується повнозв'язний шар. Повнозв'язний шар є одновимірним, у ньому кожен нейрон пов'язаний з кожним нейроном попереднього шару на всіх рівнях, якщо у попередньому шарі присутній параметр глибини. Даний шар також може використовуватися в якості останнього (вихідного) шару згорткової нейронної мережі, результатом якого є ймовірність приналежності вхідного зображення до певного класу. Єдиним параметром даного шару є кількість нейронів D_n. Зазвичай вона вибирається, виходячи з розміру попереднього шару і кількості класів (необхідної кількості виходів) мережі. Схема розглянутого шару представлена на рисунку 2.3.

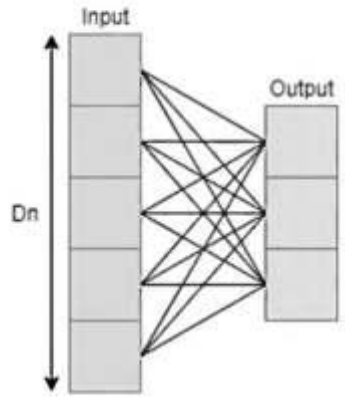

Рисунок 2.3 – Два повнозв'язних шари нейромережі

Шар виключення використовується для регуляризації даних, що допомагає запобігти перенавчанню мережі шляхом випадкового виключення нейронів (вузлів) у певному шарі на кожній епосі навчання (рисунок 2.4) [22].

Шар субдискретизації згорткової нейронної мережі використовується для скорочення розмірності даних з метою зменшення ймовірності

швидкого перенавчання, а також для зниження обчислювальної складності і витрат пам'яті. В даному шарі використовується вікно певного розміру (U) для виділення областей нейронів попереднього шару, які пов'язані з нейроном шару субдискретизації. Потім, за певними правилами вибирається і розраховується один сигнал нейрона шару субдискретизації.

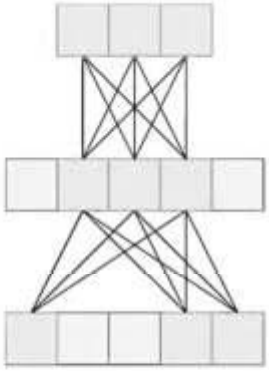

Рисунок 2.4 – Формування шару виключення

Розмірність шару субдискретизації визначається його параметрами і розмірами попереднього шару мережі, відповідно до формул [22]

$$W = \frac{W_p - P_s}{S} + 1, \tag{2.6}$$

$$H = \frac{H_p - P_s}{S} + 1, \tag{2.7}$$

де W і H – ширина і висота шару субдискретизації; W_p і H_p – ширина і висота попереднього шару відповідно; P_s – розмір квадратного вікна шару субдискретизації мережі; S – зміщення вікна шару субдискретизації.

Важливим параметром шару субдискретизації є розмір вікна P_s. Чим він більший, тим сильніша руйнівна дія операції субдискретизації для

виділених згортковим шаром мережі ознак, а чим він менший, тим сильніше перенавчання мережі. Тому необхідно знаходити компромісне значення.

У випадку максимального шару підвибірки для такого розміру фільтру обиратиметься максимальне (max) значення з квадратної області розміром $A = 4$. Розмір по глибині залишається незмінним, тобто

$$f(A) = \max_{i=1,\dots,|A|} \{a_i \in A\}. \tag{2.8}$$

Крім максимального, існують інші шари підвибірки, наприклад, усереднена (avg) підвибірка та l_2 підвибірка, які обчислюються за формулами

$$f(A) = \frac{1}{|A|} \sum_{\substack{a_i \in A \\ i=1,\dots,|A|}} a_i, \tag{2.9}$$

$$f(A) = \sqrt{\sum_{\substack{a_i \in A \\ i=1,\dots,|A|}} a_i}. \tag{2.10}$$

Вибір математичної функції, яка формує основу шару підвибірки, залежить від конкретної задачі. Наприклад, на рисунку 2.5 наведено приклад операції maxpooling.

Одним з популярних класифікаторів є класифікатор типу log-softmax, який представляє собою узагальнену логістичну функцію для багатовимірного випадку. Вираз для обчислення можна задати у такій формі (x – вхідний сигнал) [8]

$$L_i = -\log\left(\frac{\exp(x_i)}{\sum_{i=1}^{m}\exp(x_i)}\right). \qquad (2.11)$$

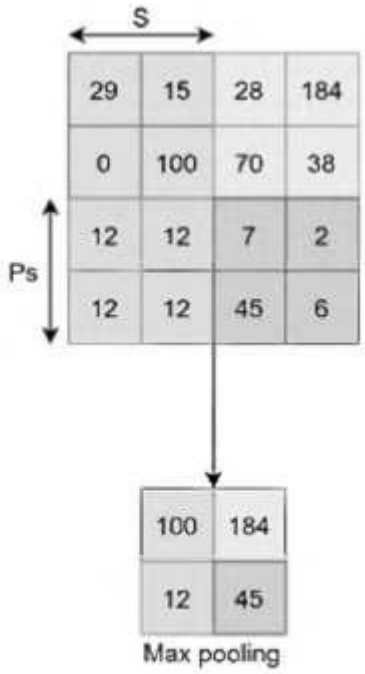

Рисунок 2.5 – Формування шару субдискретизації
за допомогою операції max

Він, на відміну, наприклад, від методу опорних векторів, керується більш інтуїтивним підходом та має ймовірнісну інтерпретацію. Далі застосовується функція втрат, представлена у вигляді перехресної ентропії (кросс-ентропії), тобто [8]

$$E(X,Y) = -\frac{1}{n}\sum_{i=0}^{m}(y_i \ln a(x_i) + (1-y_i)\ln(1-a(x_i))), \qquad (2.12)$$

де X та Y – набори прикладів для навчання (вхідний набір) та міток (бажаних або правильних значень) відповідно; y – бажане значення на

виході мережі із набору Y; $a(x)$ – вихідні значення нейронної мережі з вхідним значенням x; n – розмір тренувальних даних; m – кількість виходів мережі.

Також можна використовувати звичайну квадратичну функцію (метод найменших квадратів) у вигляді [8]

$$E(X,Y)=\frac{1}{n}\sum_{i=0}^{m}(y_i^{truth}-y_i)^2 . \quad (2.13)$$

Рекомендації до архітектури нейронної мережі в цілому [20]:

а) процес формування архітектури мережі повинен враховувати глобальні параметри архітектури, наведені на рисунку 2.6, де значення змінних n, m, k і d – додатні цілі числа, причому $n \in [1;3]$, $m \in [0;1]$, $k \in [0;2]$, $d >> 0$.

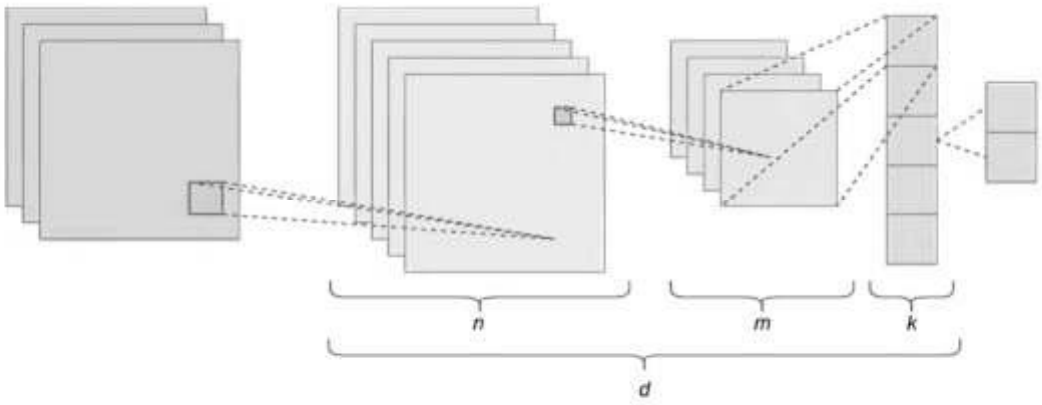

Рисунок 2.6 – Шаблон архітектури нейромережі

б) бажано, щоб мережа містила якомога більше шарів (велике значення d), тобто була глибшою;

в) чим більше і складніше зображення, тим більше значення n;

г) бажано використовувати малі значення k.

Рекомендації до архітектури вхідного шару мережі:

а) вхідні зображення повинні бути одного розміру (бажано квадратними) та мати однакову кількість каналів;

б) ширина і висота вхідного зображення повинні багаторазово ділитися на 2.

Рекомендації до архітектури згорткового шару:

а) параметри шару P (додавання нулів) і S (зміщення фільтрів) повинні бути обрані таким чином, щоб розміри згорткового шару були цілочисельними і, бажано, дорівнювали розмірності попереднього шару, за умови, що він не є згортковим шаром;

б) переважно використовувати фільтри малого розміру: 3*3 або 5*5, однак, для великих вхідних зображень розміри фільтрів першого згорткового шару можуть бути 7*7;

в) чим більший вхідний шар мережі, тим більше фільтрів рекомендується використовувати в згортковому шарі, при цьому, бажано, щоб кількість фільтрів для глибших згорткових шарів збільшувалася.

Рекомендації до архітектури шару субдискретизації:

а) використовувати вибір максимального значення замість середнього;

б) переважно застосовувати розміри вікна 2*2, оскільки великі розміри вікна надають значний вплив на руйнування ознак, виділених згорткових шарів мережі.

Наведені рекомендації є основною для розробки алгоритму формування найбільш ефективної архітектури згорткової нейронної мережі.

Під навчанням (у контексті нейронних мереж) розглядається налаштування архітектури мережі, ваг нейронів та їх зв'язків, які здійснюють вплив на ефективне виконання поставленої задачі [13].

Для згорткових нейронних мереж найчастіше використовуються методи навчання з учителем. У такому випадку задача машинного навчання полягає у реалізації функції виведення на підставі відомих навчальних даних.

Структура даних складається з набору прикладів для розповсюдження навчання. Кожен приклад являє собою особливу пару, яка складається з вхідного об'єкта (вектора) і бажаного вихідного значення (контрольного сигналу).

При навчанні з учителем дані, вибрані для навчання мережі, розділяють на дві частини: навчальну та тестову вибірки. Стандартним співвідношенням поділу є 70 на 30, проте можливі й інші варіанти. Для навчальної вибірки розраховуються значення функції, визначаються значення функції помилки. Після цього проводиться коригування ваг синапсів у мережі. Ці дії повторюються до тих пір, поки значення функції помилки не мінімізується. Повторювати дії можна як для виділених екземплярів навчальної вибірки, так і для всієї вибірки в цілому. Після здійснення навчання нейронна мережа запускається на тестовій вибірці, щоб зрозуміти, наскільки добре система навчилася.

2.2 Алгоритм зворотного поширення помилки

Оскільки згорткові нейронні мережі за типом поширення активаційного сигналу між нейронами є прямими, то для таких мереж доцільним є застосування алгоритму зворотного поширення помилки. Він представляє собою алгоритм градієнтного спуску (стратегія підбору ваг нейронів багатошарової нейронної мережі базується на градієнтному алгоритмі) [1, 17].

Алгоритм зворотного поширення помилки використовує два поширення мережею – пряме та зворотнє. Фактично генерується набір вихідних сигналів, які визначають реакцію мережі на вхідні дані. Всі синаптичні ваги при прямому проходженні є фіксованими.

При зворотному проходженні параметри (синаптичні ваги) налаштовуються відповідно до правила коригування помилок, суть якого полягає в тому, що від очікуваних вхідних значень віднімається отримане (результуюче) значення фактичного входу. В результаті такої операції отримується сигнал помилки, який поширюється в протилежному напрямі по синаптичним зв'язкам. Синаптичні ваги підлаштовуються з метою максимального наближення вихідного сигналу мережі до очікуваного результата.

Недоліком даного алгоритму є те, що він не дозволяє в загальному випадку досягти глобального мінімуму. При навчанні мережі можуть спостерігатись такі негативні явища, як параліч мережі та повільна збіжність до результату [1].

На першому кроці алгоритму зворотного поширення помилки

вихідний нейрон y_i^{truth} $(i=1,2,...m)$ отримує цільове значення – вихідне значення, яке є правильним, а йому відповідає значення y_i^l $(i=1,2,...m)$. Помилка розраховується за формулою [17]

$$\sigma_i = (y_i^{truth} - y_i^l) \cdot f'(y^l), \tag{2.14}$$

де $f'(y^l)$ – похідна функції втрат.

Для функції з рівняння (2.18) похідну можна розрахувати так

$$\begin{aligned} \frac{\partial E}{\partial y_i^l} &= \frac{\partial\left(\frac{1}{2}\sum_{i=0}^{m}(y_i^{truth} - y_i^l)^2\right)}{\partial y_i^l} \\ &= \frac{\partial\left(\frac{1}{2}((y_0^{truth} - y_0^l)^2 + \ldots + (y_i^{truth} - y_i^l)^2 + \ldots + (y_n^{truth} - y_n^l)^2)\right)}{\partial y_i^l} \\ &= \frac{\partial\left(\frac{1}{2}(y_i^{truth} - y_i^l)^2\right)}{\partial y_i^l} = \frac{1}{2}\cdot 2\cdot (y_i^{truth} - y_i^l)^{2-1}\cdot\frac{\partial(y_i^{truth} - y_i^l)}{\partial y_i^l} \\ &= (y_i^{truth} - y_i^l)\cdot(-1) = y_i^l - y_i^{truth}. \end{aligned} \tag{2.15}$$

Виходячи з формул (2.14) та (2.15), маємо

$$\sigma_i = -(y_i^{truth} - y_i^l)^2. \tag{2.16}$$

У відповідності до значення, отриманого за виразом (2.15) або (2.16) та похідною будь-якої іншої функції втрат E, розраховується величина, на яку змінюється вага зв'язку першого з кінця шару та величину коригування

зсуву, тобто

$$\Delta w_{ji} = a \cdot \sigma_i \cdot z_j, \tag{2.17}$$

$$\Delta w_{0i} = a \cdot \sigma_i, \tag{2.18}$$

де z_j $(j = 1,2,...p)$ – нейрони прихованого шару; a – швидкість навчання нейронної мережі.

Отримані значення відправляються на нейрони попереднього шару. Кожен нейрон прихованого шару z_j $(j = 1,2,...p)$ знаходить суму вхідних помилок (від нейронів наступного шару), тобто

$$\sigma_j = \sum_{i=1}^{m} \sigma_i \cdot w_{ij}, \tag{2.19}$$

а також розраховує величину зміни ваг синапсів w та зміщення за формулами

$$\Delta w_{kj} = a \cdot \sigma_j \cdot x_k, \tag{2.20}$$

$$\Delta v_{0j} = a \cdot \sigma_j. \tag{2.21}$$

Наступним кроком є зміна ваг. Кожен вихідний y_i $(i = 1,2,...m)$ та прихований нейрон z_j $(j = 1,2,...p)$ змінює ваги своїх зв'язків

$$w_{ji}^{new} = w_{ji}^{old} + \Delta w_{ji}. \tag{2.22}$$

$$v_{ji}^{new} = v_{ji}^{old} + \Delta v_{ji}. \quad (2.23)$$

Процедура є аналогічною для більш складних нейронних мереж (у тому числі для глибоких згорткових нейронних мереж), проте передбачається розрахунок з більшою кількістю етапів, оскільки оновлення ваг здійснюється на кожному з шарів нейронної мережі.

Варто зазначити, що для великих мереж дана процедура може виявитись достатньо трудомісткою та вимагати значних часових затрат. Крім того, існує ряд недоліків, таких як застрягання в локальних мінімумах або сідлових точках, складний ландшафт цільової функції, неоднорідне оновлення параметрів та занадто мала швидкість навчання. Це стало поштовхом для подальших досліджень та розробки різноманітних оптимізаторів, які дозволяють покращити збіжність алгоритму та пришвидшити його роботу.

2.3 Алгоритм оптимізації ADAM

Одним з ефективних оптимізаторів є ADAM – оптимізаційний алгоритм, який поєднує в собі ідеї накопичення руху та більш слабкого оновлення ваг для типових ознак [9]. Його робота заснована на обчисленнях моментів. Математичне очікування (перший момент) – середнє значення випадкової величини. Дисперсія (другий момент) – міра розкиду значень випадкової величини відносно математичного очікування.

У даному оптимізаторі здійснюється накопичення значень градієнта.

Крім того, передбачається визначення частоти зміни градієнта. Для цього автори алгоритму пропонують оцінювати математичне очікування m_t та середню нецентровану дисперсію v_t, що можна представити такими формулами [7, 9]

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \tag{2.24}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2. \tag{2.25}$$

де g_t – градієнт; $m_0 = v_0 = 0$; β_1 та β_2 – коефіцієнти (в якості значень за замовчуванням пропонується $\beta_1 = 0{,}9$, $\beta_2 = 0{,}999$).

Основною відмінністю ADAM від інших оптимізаторів є початкове калібрування параметрів m_t та v_t. Якщо на початку роботи задавати нульове значення, то вони будуть дуже довго накопичуватись, особливо при великому вікні накопичення (при $0 << \beta_1 < 1$, $0 << \beta_2 < 1$), а будь-які початкові значення будуть новими гіперпараметрами, що є небажаним при програмуванні нейронної мережі. З цієї причини на перших кроках m_t та v_t штучно збільшуються за правилом калібрування

$$\begin{cases} \hat{m}_t = \dfrac{m_t}{1-\beta_1^t}, \\ \hat{v}_t = \dfrac{v_t}{1-\beta_2^t}. \end{cases} \tag{2.26}$$

Враховуючи вираз (2.26) та правило оновлення ваг для стохастичного градієнтного спуску, маємо

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} \hat{m}_t, \tag{2.27}$$

де $\epsilon = 10^{-8}$ – параметр згладжування.

З аналізу роботи авторів алгоритму [9] формула (2.26) отримується з рекурсивних виразів (2.24) та (2.25), причому наближення до реальних умов змушує рекурентний ряд "затухати" швидше, що дозволяє швидше збігтися до правильного результату. Стверджується, що за таких умов алгоритм працює краще, або приблизно так само, як більшість алгоритмів оптимізації на широкому наборі датасетів, за рахунок початкового калібрування. Варто зазначити, оскільки рівняння (2.27) не є єдиним можливим виглядом затухання, то допускаються експерименти з формулами калібрування.

Оптимізатор ADAM є хорошим вибором для навчання нейронної мережі, оскільки він долає ряд недоліків алгоритму зворотного поширення помилки та дозволяє значно прискорити навчання для нейронних мереж зі складною архітектурою.

2.4 Удосконалення алгоритму

У даній роботі застосовано удосконалення, яке на новому кроці алгоритму враховує інформацію з попередніх кроків (напрямок градієнта, оновлення параметрів). Основна ідея в тому, щоб прискорити рух по тим координатам, в яких градієнт послідовно вказує один і той же напрям руху.

Відбувається накопичення імпульсу за правилом: якщо ми деякий час рухаємося в певному напрямку, то, ймовірно, нам слід туди рухатися деякий час і в майбутньому (заглядання вперед – look ahead). Така зміна дозволяє швидше "котитися", якщо в бік, куди ми прямуємо, похідна збільшується, і повільніше, якщо навпаки. На основі виразу (2.27) можна сформувати наступну формулу [13]

$$w_{t+1} = w_t - \eta \cdot \Delta_t = w_t - \eta \cdot \left(\beta_1 \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t + c}} + \psi_t \right), \tag{2.28}$$

де $\Delta_t = \beta_1 \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t + c}} + \psi_t$; а $\psi_{t+1} = \Delta_t \cdot \text{sign}(g_t) \cdot (1 - \beta_1)$ – корегуюча складова, яка визначає напрямок руху.

2.5 Висновки

У даному розділі розглянуто математичну модель для розробки згорткової нейронної мережі, описано її складові компоненти, наведено рекомендації. Представлено математичну основу навчання нейронних мереж на основі алгоритму зворотного поширення помилки та оптимізатора ADAM. Запропоновано удосконалений алгоритм оптимізації нейромережі, який використовує корегуючу складову та аналізує інформацію з попередніх кроків. У наступному розділі необхідно підтвердити ефективність розробки.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

3.1 Обгрунтування вибору мови програмування

Однією з найпоширеніших мов для програмування нейронних мереж є Python. Це інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією, розроблена в 1990 році Г. ван Россумом [23].

Перевагами, які сприяють збільшенню популярності даної мови, є структури даних високого рівня в поєднанні з динамічною семантикою, можливість динамічного зв'язування, що дозволяє швидко розробляти програми та легко поєднувати вже існуючі компоненти.

Програми, написані на мові Python, можуть бути виконані на будь-якому персональному комп'ютері під керуванням десктопних операційних систем Windows, MacOS, Linux. Крім того, відсутні механізми, які потенційно призводять до помилок (наприклад, перетворення типів з втратою точності) та присутній обов'язковий контроль виняткових ситуацій.

Функціонування програми повністю визначається виконавчим середовищем. Відсутні покажчики та інші механізми для безпосередньої роботи з фізичною пам'яттю й іншим апаратним забезпеченням комп'ютера. Зручним є механізм автоматичного генерування документації на основі коду.

Висока модульність програмного забезпечення та легке повторне використання коду допускається завдяки тому, що Python підтримує пакети модулів, причому стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. Окрім цього, Python підтримує кілька парадигм програмування, а саме об'єктно-орієнтовану, процедурну, функціональну та аспектно-орієнтовану.

Перевагою використання Python для розробки нейронних мереж є те, що в ньому є можливість реалізації нового продукту з використанням великої кількості бібліотек для роботи з нейронними мережами [23].

У машинному навчанні мова програмування Python вважається однією з найпопулярніших та має повноцінний інструментарій для реалізації згорткової нейронної мережі з використанням різних оптимізаторів функції втрат.

Багато допоміжних задач, які зустрічаються при розробці програмного забезпечення, вже вирішені в рамках стандартних та користувацьких бібліотек. Існує безліч платформ для реалізації нейромереж, наприклад Caffe, TensorFlow, Theano, PyTorch та інші. На нашу думку, найкращим рішенням для реалізації експериментального програмного стенда і оцінювання ефективності запропонованого класифікатора є бібліотека TensorFlow від Google, яка розповсюджується за відкритою ліцензією [24].

Необхідні обчислення проводяться за допомогою потоків даних через графи станів (data-flow графи). У них вершини представляють собою математичні операції, а ребра є даними, які зазвичай подаються у вигляді багатовимірних масивів або тензорів. Бібліотеки TensorFlow помітно

спрощують вбудовування в додатки елементів, які навчаються, і функцій штучного інтелекту, призначених для розпізнавання мови, організації комп'ютерного зору, обробки природної мови тощо.

Отже, мова програмування Python досить зручна, надійна та потужна, що і обумовило необхідність її застосування у даному проекті.

3.2 Результати експериментів

У межах роботи розв'язується задача мультикласифікації зображень з використанням бази даних MNIST (Modified National Institute of Standards and Technology – Вибірка Національного інституту стандартів і технологій США), яка є колекцією з 70 000 написаних від руки чисел розміром 28*28 пікселів (від 0 до 9), що поділяється на навчальну (60 000) та тестову (10 000) вибірки [25]. Приклад даних наведено на рисунку 3.1.

Рисунок 3.1 – Приклад даних датасету MNIST

Відповідно вхідними даними для навчання та тестування нейронної мережі є дані вибірки, кількість епох навчання, назви файлів для збереження параметрів нейронної мережі (для процесу навчання та тестування). Вихідними даними для програми є значення функції втрат E, розрахованих при навчанні нейронної мережі, точність розпізнавання даних з тестової вибірки та часові характеристики роботи програми.

У роботі розроблене програмне забезпечення, яке реалізує роботу згорткової нейронної мережі з використанням оптимізатора ADAM та його модифікації cor_ADAM. У програмному забезпеченні повністю відображена архітектура згорткової нейронної мережі, тобто такі її елементи, як:

– згортки, які працюють на основі фільтрів, що займаються розпізнаванням певних характеристик зображення;

– функції активації;

– шари макспулінгу (максимальної підвибірки);

– повнозв'язний шар.

Програмування згорток можна виконувати за допомогою бібліотеки NumPy. Згортка здійснюється за допомогою зовнішнього циклу for, який застосовує фільтри до вхідного зображення, та двох циклів while для забезпечення переміщення фільтру вздовж зображення. Поелементне множення пікселів здійснюється за допомогою стандартного оператора множення.

Після низки згорткових шарів застосовується редукція (зменшення розміру) проміжного результату, що в машинному навчанні носить назву даунсемплінгу. В даному випадку використано найбільш розповсюджений

варіант даунсемплінгу – макспулінг (максимальне об'єднання). Процес максимального об'єднання передбачає вибір максимального значення пікселя з певного вікна просіювання (частини матриці, яка стискається до 1 пікселя). В даному випадку для визначення максимального значення використовується функція max.

На етапі повнозв'язного шару передбачається перетворення тривимірної матриці сигналів у вектор, який можна пропустити через повнозв'язну нейронну мережу. Для розгортання у вектор використовується метод reshape бібліотеки NumPy. Після цього утворені вектори пропускаються через повнозв'язний шар, який має велику кількість зв'язків $n \times m$, де n – кількість нейронів, з яких складається повнозв'язний шар, а m – кількість вихідних сигналів (кількість елементів вектора). Взаємодія згорткових шарів з повнозв'язним шаром зображена на рисунку 3.2.

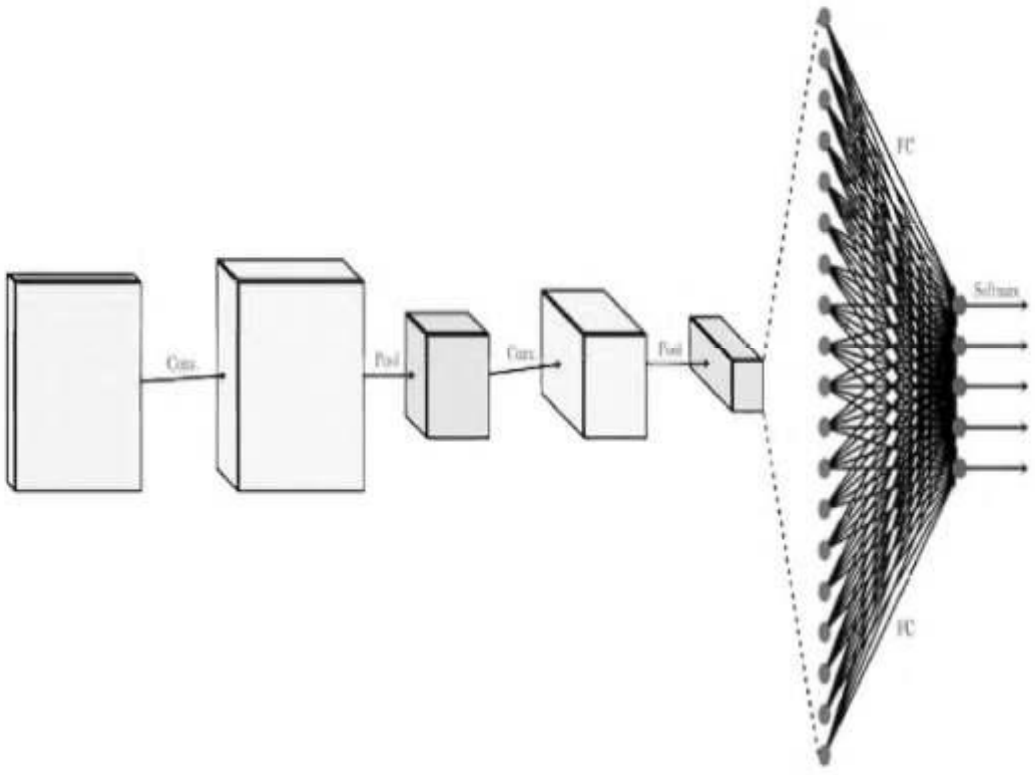

Рисунок 3.2 – Згорткова нейронна мережа для експериментів

Вихідний шар відповідає за формування ймовірностей належності вхідного образу тому або іншому класу (в даному випадку, класів всього 10, як і арабських цифр). Зважені та просумовані сигнали модифікуються з допомогою функції активації.

У розробленій програмі використовується функція softmax, яка реалізується за допомогою експоненціальної функції з бібліотеки NumPy. В якості функції втрат використовується функція квадратів помилок.

Оскільки файли з колекції рукописних чисел та відповідні їм маркери зберігаються у вигляді тензорів, то робота з даними, на відміну від реальних зображень, значно спрощується. Для зчитування даних було розроблено функції extract_data (для отримання самих файлів) та extract_labels (отримання маркерів).

Було розроблено функції зворотного поширення помилки через пулінговий та згортковий шари (convolutionBackward для згорткового шару, maxpoolBackward для шару макспулінгу). Для спрощення та підвищення модульності програмного забезпечення розроблено функцію conv, яка комбінує операції прямого та зворотного розповсюдження сигналів у згорткових шарах.

Для навчання та оптимізації нейронної мережі було використано алгоритм оптимізації ADAM та його модифіковану версію cor_ADAM. Для зручного використання нейронної мережі дозволено коригування файлів для зміни вхідних гіперпараметрів нейронної мережі.

Для тестування роботи оптимізаторів встановимо гіперпараметри за роботою Кінгма та Ба [9], тоді встановимо наступні параметри алгоритмів оптимізації $\eta = 0{,}002$, $\beta_1 = 0{,}9$, $\beta_2 = 0{,}999$, $m_0 = 0$, $v_0 = 0$.

Попередня обробка до навчальних зображень розміром 28*28 пікселів у відтінках сірого кольору не застосовувалась. Мережа навчалася на міні-батчах розміром 300 зображень. Усі ваги ініціалізувались за допомогою стандартного нормального закону розподілу зі стандартним відхиленням 0,1. Значення зсувів встановлені на значенні 0,1.

Проведено низку експериментів з різною архітектурою згорткової нейронної мережі та визначено, що cor_ADAM дає кращі результати в більшості випадків. У відповідності до рисунку 3.3 значення функції втрат E максимальне на найперших ітераціях. Поступово нейронна мережа адаптується до вже оброблених даних (в моментах, коли значення функції втрат наближається до нуля) та реагує на нові типи (в моментах, коли значення функції втрат зростає). Результати ілюструють те, що дані оптимізатори мають практично однакову ефективність роботи, але cor_ADAM працює дещо краще за ADAM більшу частину часу.

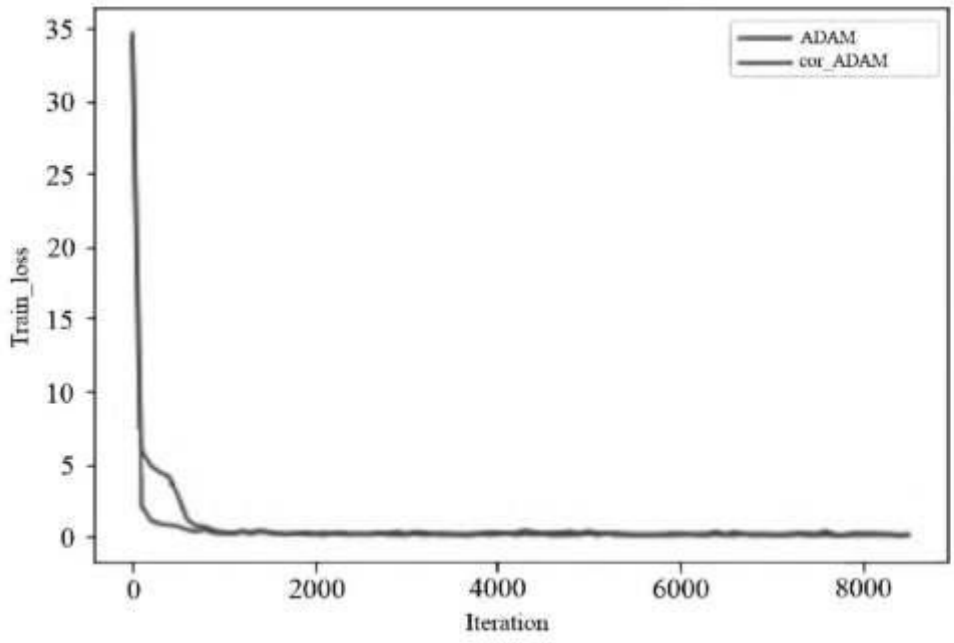

Рисунок 3.3 – Зміна значення функції втрат на навчальних даних

Оптимізатор cor_ADAM працює краще за ADAM на тестових даних з відносно великим відривом за однакових параметрів алгоритмів (рисунок 3.4). Крім того, він має кращу здатність до узагальнення.

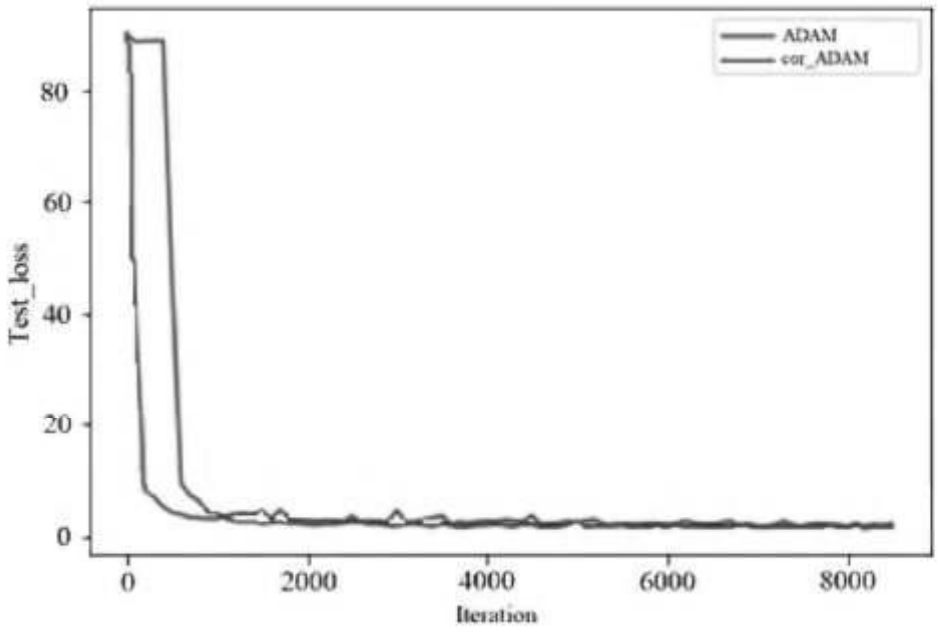

Рисунок 3.4 – Зміна значення функції втрат на тестових даних

Між 7500 і 7800 ітераціями, коли величина функції втрат E для оптимізатора ADAM зростає, значення помилки для cor_ADAM навпаки зменшується (рисунок 3.5). Це досить хороша поведінка оптимізатора, оскільки його метою є максимальне зменшення значення помилки на тестових даних.

Слід зазначити, що на 7708-ій ітерації оптимізатор cor_ADAM дозволяє отримати найнижче значення помилки, яке становить 0,0258 порівняно з 0,1282 для ADAM на 7812 ітерації, що краще на 0,1024.

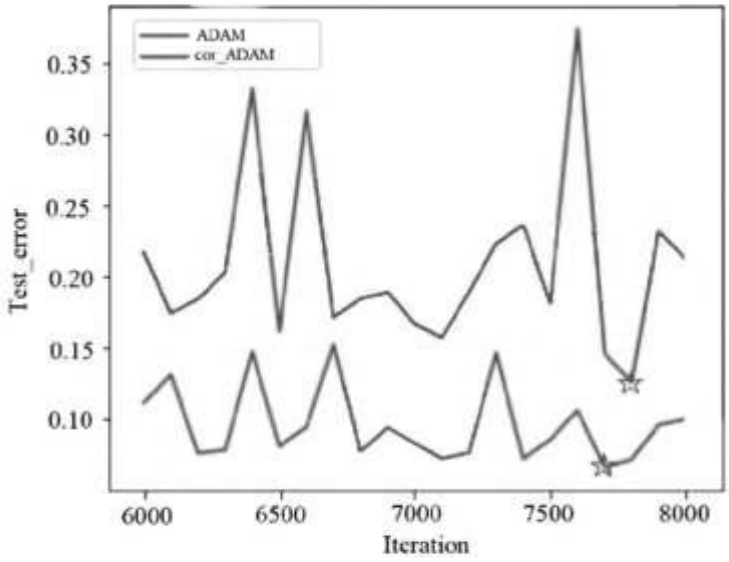

Рисунок 3.5 – Зміна значення функції втрат
на тестових даних (масштабування)

У результаті роботи програмного забезпечення було побудовано графік точності розпізнавання окремих класів об'єктів-чисел. Як показує графік, відображений на рисунку 3.6, розпізнавання об'єкту кожного класу здійснюється з достовірністю понад 95%, що свідчить про правильну роботу нейронної мережі.

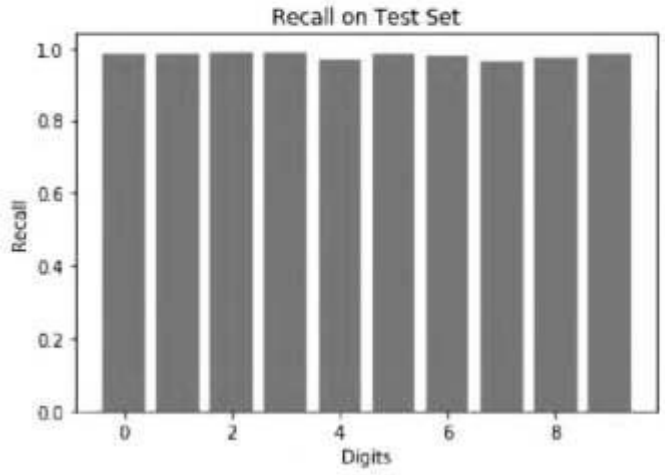

Рисунок 3.6 – Точність розпізнавання окремих класів об'єктів

Отже, згорткова нейронна мережа має високу здатність до узагальнення, особливо зображень символів, які мають подібне з навчальною множиною походження. Узагальнююча здатність мережі значною мірою визначається стійкістю її архітектури до рівня варіативності топології зображень символів, зокрема, зміни ширини тощо.

Порівняння результатів роботи різних дослідників показано в таблиці 3.1, в якій відображено низку моделей для роботи з задачею мультикласифікації на вибірці MNIST та їх точність на тестовій вибірці.

Таблиця 3.1 – Порівняння результатів

Дослідник	Модель	Точність на тестовій вибірці, %
С. Раві [26]	Нейромережа ProjectionNet	95,00
О.-С. Гранмо [27]	Машина Цетліна	98,20
Е. Дюпон, А. Доше та інші [28]	Модель ANODE	98,20
С. Ан, М. Лі та інші [29]	Неоднорідний ансамбль з простими CNN	99,91
Дана робота	Згорткова нейронна мережа (5CNN) з удосконаленим алгоритмом ADAM	99,27

У результаті проведеної експериментальної роботи оптимізатор cor_ADAM досягає кращих результатів, порівняно з оптимізатором ADAM. При цьому запропонована модифікація використовує більше пам'яті,

зменшуючи максимальну помилку розпізнавання, маючи вищу точність та швидшу збіжність до оптимуму.

3.3 Висновки

У даному розділі описано структуру розробленого програмного забезпечення та протестовано роботу удосконаленого алгоритму оптимізації ADAM. Удосконалення дозволяє покращити збіжність та узагальнюючі властивості методу ADAM без значного збільшення обчислювальної складності. Недоліком запропонованого рішення є те, що використовується більше пам'яті, ніж у ході роботи базового методу.

4 РОЗДІЛ ЕКОНОМІКИ

4.1 Технологічний аудит удосконаленого алгоритму оптимізації

Метою проведення комерційного і технологічного аудиту дослідження за темою «Удосконалений алгоритм ADAM для навчання нейромережі» є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в джерелі [30].

Таблиця 4.1 – Критерії оцінювання рівня комерційного потенціалу розробки та їх бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів

Продовження таблиці 4.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти на навчання	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій > 10-ти рр.	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій > 5-ти рр.	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти рр.	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років

Продовження таблиці 4.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки пові-домлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Для проведення технологічного аудиту залучено 3-х незалежних експертів:

– кандидат технічних наук, доцент Гармаш В.В. Автор низки наукових робіт в області обробки та розпізнавання зображень;

– доктор технічних наук, професор Бісікало О.В. Автор наукових праць в галузі штучного інтелекту та інтелектуального аналізу даних;

– кандидат технічних наук, доцент Кривогубченко С.Г. Наукові інтереси пов'язані з адаптивними алгоритмами оптимізації комп'ютерних систем.

За результатами оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки, наведених у таблиці 4.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в таблиці 4.3 [31]. Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Удосконалений алгоритм ADAM для навчання нейромережі» становить 41,3 бала, що, відповідно до таблиці 4.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки високий).

Таблиця 4.2 – Результати оцінювання комерційного потенціалу розробки

Критерії	Експерт		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	5	5	5
2. Ринкові переваги (наявність аналогів)	1	1	1
3. Ринкові переваги (ціна продукту)	2	2	2
4. Ринкові переваги (технічні властивості)	3	3	3
5. Ринкові переваги (експлуатаційні витрати)	2	2	2
6. Ринкові перспективи (розмір ринку)	3	2	3
7. Ринкові перспективи (конкуренція)	3	2	3
8. Практична здійсненність (наявність фахівців)	5	5	5
9. Практична здійсненність (наявність фінансів)	4	4	3
10. Практична здійсненність (необхідність матеріалів)	5	5	5
11. Практична здійсненність (термін реалізації)	5	4	4
12. Практична здійсненність (розробка документів)	5	5	5
Сума балів	43	40	41
Середньоарифметична сума балів $СБ_с$	41,3		

Таблиця 4.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів, розрахована на основі висновків експертів	Рівень наукового та комерційного потенціалу розробки
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

Розробка актуальна в сферах бізнесу, науки, освіти тощо. Оскільки розглядаються специфічні питання наукового характеру, то можна відмітити, що існує недостатньо відкритих програм-аналогів. Можна виділити функціонал математичних середовищ MatLab (Deep Learning Toolbox), Mathematica (Wolfram Neural Network Framework) та Python (Keras, TensorFlow). Розроблений програмний продукт використовує більше пам'яті на збереження додаткових даних, але дозволяє отримати кращу точність.

4.2 Розрахунок витрат на удосконалення алгоритму навчання нейромережі

При виконанні магістерської кваліфікаційної роботи було зроблено такі витрати [31]:

1. Витрати на основну заробітну плату дослідників розраховуємо у відповідності до посадових окладів працівників, за формулою:

$$З_о = \sum_{i=1}^{k} \frac{М_{мі} \cdot t_i}{T_р} = 15150{,}00 \cdot 48 / 24 = 30300{,}00 \; грн, \qquad (4.1)$$

де k – кількість посад дослідників залучених до процесу досліджень; $М_{мі}$ – місячний посадовий оклад конкретного дослідника, грн; t_i – кількість днів роботи конкретного дослідника, дні; $T_р$ – середня кількість робочих днів в місяці, $T_р$ = 24 дні.

Проведені розрахунки зведемо до таблиці 4.4.

2. Витрати на основну заробітну плату робітників за відповідними найменуваннями робіт НДР розраховуємо за формулою:

$$З_p = \sum_{i=1}^{n} C_i \cdot t_i, \qquad (4.2)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн./год.; t_i – час роботи робітника при виконанні визначеної роботи, год.

Таблиця 4.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Кількість днів роботи	Витрати на заробітну плату, грн
Керівник НДР з удосконалення алгоритму ADAM навчання нейромережі	15150,00	631,25	48	30300,00
Інженер-програміст 1-ї категорії	14550,00	606,25	24	14550,00
Інженер-дослідник нейромереж	14600,00	608,33	40	24333,33
Технік 1-ї категорії	7450,00	310,42	20	6208,33
Всього				75391,67

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \qquad (4.3)$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $M_м$ = 6700,00 грн.; K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки

робітнику відповідного розряду; K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати; T_p – середнє число робочих днів в місяці, приблизно T_p = 24 дні; $t_{зм}$ – тривалість зміни, год.

Результати запишемо в таблицю 4.5. Тоді маємо:

$$C_1 = 6700{,}00 \cdot 1{,}10 \cdot 1{,}65 / (24 \cdot 8) = 63{,}34 \text{ грн.}$$
$$З_р = 63{,}34 \cdot 16{,}00 = 1013{,}38 \text{ грн.}$$

3. Додаткову заробітну плату розраховуємо як 10…12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%} = (75391{,}67 + 8970{,}38) \cdot 11 / 100\% = 9279{,}83 \text{ грн.}, \quad (4.4)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати; приймемо 11%.

4. Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} = (75391{,}67 + 8970{,}38 + 9279{,}83) \cdot \frac{22}{100} = 20601{,}21 \text{ грн.}, \quad (4.5)$$

де $Н_{зп}$ – норма нарахування на заробітну плату; приймаємо 22%.

Таблиця 4.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год.	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн.	Величина оплати на робітника, грн.
Підготовка робочого місця розробника програмного забезпечення	16,00	2	1,10	63,34	1013,38
Підготовка обладнання моделювання та навчання нейромережі	12,00	3	1,35	77,73	932,77
Інсталяція програмного забезпечення для моделювання та розробки нейромережі	6,20	4	1,50	86,37	535,48
Компіляція програмних блоків	7,35	5	1,70	97,88	719,44
Налагодження програмних блоків	6,00	5	1,70	97,88	587,30
Тестування системи	15,00	4	1,50	86,37	1295,51
Тренування системи	50,00	3	1,35	77,73	3886,52
Всього					8970,38

5. Витрати на матеріали розраховуються окремо по кожному виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot K_j - \sum_{j=1}^{n} B_j \cdot Ц_{вj} = 3{,}0 \cdot 273{,}00 \cdot 1{,}12 - 0 \cdot 0 = 917{,}28 \text{ грн.}, \quad (4.6)$$

де H_j – норма витрат матеріалу j-го найменування, кг; n – кількість видів матеріалів; $Ц_j$ – вартість матеріалу j-го найменування, грн./кг; K_j – коефіцієнт транспортних витрат, (K_j = 1,1 … 1,15); B_j – маса відходів j-го найменування, кг; $Ц_{вj}$ – вартість відходів j-го найменування, грн./кг.

Проведені розрахунки зведемо до таблиці 4.6.

Таблиця 4.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Офісний папір 500 80 г\м	273,00	3,0	0	0	917,28
Папір для записів 100 70 г\м	155,00	5,0	0	0	868,00
Органайзер офісний	174,00	2,0	0	0	389,76
Канцелярське приладдя	222,00	3,0	0	0	745,92
Картридж для принтера	1465,00	1,0	0	0	1640,80
Диск оптичний CD-R	23,25	3,0	0	0	78,12
Flesh-пам'ять 64 GB	625,00	1,0	0	0	700,00
Тека для паперів	97,00	4,0	0	0	434,56
Всього					5774,44

6. Витрати на комплектуючі, які використовують при проведенні НДР, розраховуємо, згідно з їхньою номенклатурою, за формулою:

$$K_в = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot K_j = 1 \cdot 12345{,}00 \cdot 1{,}12 = 13826{,}40 \text{ грн.}, \quad (4.7)$$

де H_j – кількість комплектуючих j-го виду, шт.; $Ц_j$ – покупна ціна комплектуючих j-го виду, грн; K_j – коефіцієнт транспортних витрат, K_j = 1,1 … 1,15.

7. Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i = 44800{,}00 \cdot 1 \cdot 1{,}12 = 50176{,}00 \text{ грн.}, \quad (4.8)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн.; $C_{пр.i}$ – кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.; K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, K_i = 1,10…1,12; k – кількість найменувань устаткування.

8. Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i = 9650{,}00 \cdot 1 \cdot 1{,}12 = 10808{,}00 \text{ грн.}, \quad (4.9)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн.; $C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.; K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, K_i = 1,10…1,12; k – кількість найменувань програмних засобів.

Отримані результати зведемо до таблиці 4.7.

Таблиця 4.7 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт.	Ціна за одиницю, грн.	Вартість, грн.
Математичне середовище MatLab (Deep Learning Toolbox)	1	9650,00	10808,00
Прикладне ПЗ Mathematica (Wolfram Neural Network Framework)	1	7690,00	8612,80
Середовище програмування Python (Keras, TensorFlow)	1	8460,00	9475,20
Всього			28896,00

9. У спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12} = (27720{,}00 \cdot 2) / (2 \cdot 12) = 2310{,}00\ грн., \qquad (4.10)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн.; $t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців; $T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

Проведені розрахунки зведемо до таблиці 4.8.

Таблиця 4.8 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн.
Персональний комп'ютер розробника ПЗ	27720,00	2	2	2310,00
Персональний комп'ютер інженера-дослідника нейромереж	24460,00	2	2	2038,33
Робоче місце інженера-програміста	9130,00	5	2	304,33
Робоче місце інженера-дослідника	9500,00	5	2	316,67
Пристрої передачі даних	6600,00	5	2	220,00
Оргтехніка	8360,00	5	2	278,67
Приміщення лабораторії розробки ПЗ	401000,00	25	2	2673,33
ОС Windows	8580,00	2	2	715,00
Прикладний пакет Microsoft Office	7860,00	2	2	655,00
Всього				9511,33

10. Витрати на силову електроенергію розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вні}}{\eta_i} = 0{,}42 \cdot 380{,}0 \cdot 6{,}20 \cdot \frac{0.95}{0.97} = 989{,}52 \ грн., \quad (4.11)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт; t_i – тривалість роботи обладнання на етапі дослідження, год.; $Ц_e$ – вартість 1 кВт-години електроенергії, грн.; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e$ = 6,20 грн.; $K_{вні}$ – коефіцієнт, що враховує використання потужності, $K_{вні} < 1$; η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

Проведені розрахунки зведемо до таблиці 4.9.

Таблиця 4.9 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Персональний комп'ютер розробника ПЗ	0,42	380,0	989,52
Персональний комп'ютер інженера-дослідника нейромереж	0,05	380,0	117,80
Робоче місце інженера-програміста	0,12	300,0	223,20
Робоче місце інженера-дослідника	0,10	300,0	186,00
Пристрої передачі даних	0,05	180,0	55,80
Оргтехніка	0,45	10,0	27,90
Серверне обладнання	0,32	180,0	357,12
Всього			1957,34

11. Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників:

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%} = (75391{,}67 + 8970{,}38) \cdot 20 / 100\% = 16872{,}41 \text{ грн.}, \quad (4.12)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св} = 20\%$.

12. Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%} = (75391{,}67 + 8970{,}38) \cdot 30 / 100\% = 25308{,}62 \text{ грн.}, \quad (4.13)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп} = 30\%$.

13. До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%} = (75391{,}67 + 8970{,}38) \cdot 50 / 100\% = 42181{,}03 \text{ грн.}, \quad (4.14)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів} = 50\%$.

14. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%} = (75391{,}67 + 8970{,}38) \cdot 100 / 100 = 84362{,}05 \text{ грн.}, \quad (4.15)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв} = 100\%$.

15. Витрати на проведення науково-дослідної роботи розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + B_{спец} + B_{прг} + A_{обл} + B_е + B_{св} + B_{от} + I_в + B_{нзв}, \quad (4.16)$$

$$B_{заг} = 75391{,}67 + 8970{,}38 + 9279{,}83 + 20601{,}21277 + 5774{,}44 + 13826{,}40 +$$
$$+ 50176{,}00 + 28896{,}00 + 9511{,}33 + 1957{,}34 + 16872{,}41 + 25308{,}62 +$$
$$+ 42181{,}03 + 84362{,}05 = 393108{,}70 \text{ грн.}$$

16. Загальні витрати на завершення науково-дослідної роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta} = 393108{,}70 / 0{,}9 = 436787{,}45 \text{ грн.}, \quad (4.17)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо $\eta = 0{,}9$.

4.3 Прогнозування комерційних ефектів від комерціалізації розробки

У ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку. Результати дослідження проведені передбачають комерціалізацію впродовж 4-ох років реалізації на ринку. У цьому випадку майбутній економічний ефект буде формуватися на основі:

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик (1-й рік – +150 споживачів; 2-й рік – +300 споживачів; 3-й – +500 споживачів; 4-й – +350 споживачів);

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 500 організацій;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 100000,00 грн;

$\pm\Delta Ц_о$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 50150,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 4-х років, впродовж яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [31]:

$$\Delta П_i = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (4.18)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість; у 2022 році ставка податку на додану вартість складає 20%, а коефіцієнт λ=0,8333; ρ – коефіцієнт, який враховує рентабельність інноваційного продукту; приймемо ρ=40%; ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2022 році ϑ = 18%.

Збільшення чистого прибутку 1-го року:

$$\Delta П_1 = (50150{,}00 \cdot 500{,}00 + 150150{,}00 \cdot 150) \cdot 0{,}83 \cdot 0{,}4 \cdot (1 - 0{,}18) =$$
$$= 12957943{,}40 \text{ грн.}$$

Збільшення чистого прибутку 2-го року:

$$\Delta П_2 = (50150{,}00 \cdot 500{,}00 + 150150{,}00 \cdot 450) \cdot 0{,}83 \cdot 0{,}4 \cdot (1 - 0{,}18) =$$
$$= 25220994{,}20 \text{ грн.}$$

Збільшення чистого прибутку 3-го року:

$$\Delta П_3 = (50150{,}00 \cdot 500{,}00 + 150150{,}00 \cdot 950) \cdot 0{,}83 \cdot 0{,}4 \cdot (1 - 0{,}18) =$$
$$= 45659412{,}20 \text{ грн.}$$

Збільшення чистого прибутку 4-го року:

$$\Delta П_4 = (50150{,}00 \cdot 500{,}00 + 150150{,}00 \cdot 1300) \cdot 0{,}83 \cdot 0{,}4 \cdot (1 - 0{,}18) =$$
$$= 59966304{,}80 \text{ грн.}$$

Приведена вартість збільшення всіх чистих прибутків, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{t=1}^{T} \frac{\Delta П_t}{(1+\tau)^t}, \qquad (4.19)$$

де $\Delta П_t$ – збільшення чистого прибутку у кожному з років, впродовж яких виявляються результати впровадження науково-технічної розробки, грн.; T – період часу, впродовж якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки; τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0{,}19$; t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту

отримання потенційним інвестором додаткових чистих прибутків у цьому році.

Тоді маємо:

$$ПП = 12957943,40 / (1 + 0,19)^1 + 25220994,20 / (1 + 0,19)^2 +$$
$$+ 45659412,20 / (1 + 0,19)^3 + 59966304,80 / (1 + 0,19)^4 =$$
$$= 10889028,07 + 17810178,80 + 27095017,27 + 29903322,34 =$$
$$= 85697546,47 \text{ грн.}$$

Величина початкових інвестицій, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \qquad (4.20)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=3; $ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 436787,45 грн.

Тоді:

$$PV = 3 \cdot 436787,45 = 1310362,35 \text{ грн.}$$

Абсолютний економічний ефект для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV = 85697546,47 - 1310362,35 = 84387184,13 \text{ грн.} \quad (4.21)$$

Внутрішня економічна дохідність інвестицій, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[T_{ж}]{1 + \frac{E_{абс}}{PV}} - 1 = (1 + 84387184{,}13 / 1310362{,}35)^{1/4} - 1 = 1{,}84, \quad (4.22)$$

де $T_{ж}$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

Мінімальна внутрішня економічна дохідність вкладених інвестицій:

$$\tau_{мін} = d + f, \quad (4.23)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2022 році в Україні d = 0,12; f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,5.

Тоді:

$$\tau_{мін} = 0{,}12 + 0{,}5 = 0{,}62 < E_в = 1{,}84,$$

тобто інвестувати в науково-дослідну роботу доцільно.

Період окупності інвестицій, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в} = 1 / 1{,}84 = 0{,}54 \text{ роки}. \quad (4.24)$$

Оскільки $T_{ок}$ < 3-ох років, що свідчить про комерційну привабливість науково-технічної розробки, то це може спонукати потенційного інвестора профінансувати впровадження та виведення її на ринок.

4.4 Висновки

Аналіз комерційного потенціалу розробки показав, що вона за своїми характеристиками випереджає аналоги, що підтверджує її перспективність. Результати економічної частини магістерської кваліфікаційної роботи зведено у таблицю 4.10. Отже, основні техніко-економічні показники удосконаленого алгоритму, визначені у технічному завданні, виконані.

Таблиця 4.10 – Результати економічної частини

Показники	Задані у технічному завданні	Досягнуті у магістерській кваліфікаційній роботі	Висновок
1. Витрати на розробку удосконаленого алгоритму	не більше 450 000 грн.	436 787,45 грн.	Виконано
2. Абсолютний ефект від впровадження	не менше 15 000 грн. за копію	21 096,79 грн. за копію	Виконано
3. Внутрішня дохідність інвестицій	не менше 60 %	62 %	Виконано
4. Термін окупності потенційних інвестицій	до 3-ох років	0,54 роки	Виконано

ВИСНОВКИ

У теоретичній частині роботи проаналізовано особливості функціонування нейронних мереж та алгоритмів їх навчання, розглянуто математичні основи прямого та зворотного проходження нейронної мережі. Хороший оптимізатор дозволяє зменшити вплив сторонніх чинників на результат роботи мережі. Для складних проектів, в яких важлива ефективність роботи мережі, рекомендується використовувати більш ефективний оптимізатор, такий як ADAN та його модифікації.

Запропоновано удосконалення алгоритму оптимізації Adam, яке використовує корегуючу складову та аналізує інформацію з попередніх кроків. Запропоноване удосконалення використовує більше пам'яті, але має меншу максимальну помилку навчання мережі, вищу точність та швидшу збіжність до оптимуму. Точність розпізнавання, отримана в результаті використання розробленого програмного забезпечення для навчання нейронної мережі, є достатньою для його подальшого застосування.

Також проведено технологічний аудит запропонованого алгоритму. Аналіз комерційного потенціалу програмного забезпечення показав, що розробка є перспективною.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Тимощук П.В. Штучні нейронні мережі. Львів: Видавництво НУ "Львівська політехніка", 2011. 441 с.

2. Bishop C.M. Neural Networks for Pattern Recognition. UK: Oxford University Press, 1995. 504 p.

3. Нейронні мережі – шлях до глибинного навчання. URL: https://codeguida.com/post/739/ (дата звернення 19.09.2022).

4. Venkatesan R., Li B. Convolutional Neural Networks. *Convolutional Neural Networks in Visual Computing*. 2017. P. 89-116.

5. Чопорова О.В., Кривохата А. Г. Оптимізація згорткових нейронних мереж та їх ансамблів. *Вісник Запорізького національного університету. Фізико-математичні науки.* № 1. 2019. С. 107-115.

6. Gradient Descent Optimization Algorithms. URL: https://www.kdnuggets.com/2019/06/gradient-descent-algorithms-cheat-sheet.html (дата звернення 10.09.2022).

7. Ruder S. An Overview of Gradient Descent Optimization. Cornell University Library. URL: https://arxiv.org/abs/1609.04747 (дата звернення 16.09.2022).

8. Zhang A., Lipton Z.C., Li M., Smola A.J. Dive into Deep Learning. Optimization Algorithms. 682 p. URL: https://d2l.ai/chapter_optimization/index.html (дата звернення 10.09.2022).

9. Kingma D.P., Ba J.L. Adam: A Method for Stochastic Optimization. 2014 – 15 p. URL: https://arxiv.org/pdf/1412.6980.pdf (дата звернення 16.09.2022).

10. Manzil Z., Reddi J.S., Sachan D. and others. Adaptive Methods for Nonconvex Optimization. *Advances in neural information processing systems.* Vol. 31. 2018. 11 p.

11. Wang Y., Xiao Z., Cao G. A Convolutional Neural Network Method Based on Adam Optimizer with Power-Exponential Learning Rate for Bearing Fault Diagnosis. *Journal of Vibroengineering*. Vol. 24 (4). 2022. P. 666-678.

12. Данилевич М.О., Барабан М.В. Розробка нейронної мережі розпізнавання об'єктів на фото. *Науково-технічна конференція підрозділів ВНТУ*: матер. L науково-технічної конференції. Вінниця: ВНТУ, 2021. С. 1277-1279.

13. Данилевич М.О., Бабій Р.О., Іванов Ю.Ю. та інші. Модифікований алгоритм оптимізації функції втрат нейромережі. *Актуальні задачі медичної, біологічної фізики та інформатики*: матер. науково-практичної конференції з міжнародною участю. Вінниця: ВНМУ ім. М.І. Пирогова, 2022. С. 73-74.

14. Функції активації нейромереж. URL: https://neurohive.io/ru/osnovy-data-science/activation-functions/ (дата звернення 19.09.2022).

15. Howard A.G. Some Improvements on Deep Convolutional Neural Network Based Image Classification. 2013. URL: https://arxiv.org/ftp/arxiv/papers/1312/1312.5402.pdf (дата звернення 19.10.2022).

16. XiaoT., Zhang J., Yang K. and others. Error-Driven Incremental Learning in Deep Convolutional Neural Network for Large-Scale Image Classification. *Proceedings of the 22nd ACM International Conference on Multimedia*. USA 2014. P. 177-186.

17. Новотарський М.А., Нестеренко Б.Б. Штучні нейронні мережі: обчислення. Київ: Інститут математики НАН України, 2004. 408 с.

18. Krizhevsky A., Sutskever I., Hinton G.E. ImageNet Classification with Deep Convolutional Neural Networks. *Communications of the ACM*. Vol. 60 (6). 2017. P. 84-90.

19. Bergstra J., Bengio Y. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*. Vol. 13. 2012. P. 281-305. URL: http://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf (дата звернення 19.10.2022).

20. Becherer N., Pecarina J., Nykl S., Hopkinson K. Improving Optimization of Convolutional Neural Networks Through Parameter Fine-Tuning. *Neural Computing and Applications*. Vol. 31 (8). 2017. P. 3469-3479.

21. Kawaguchi K., Huang J., Kaelbling L.P. Effect of Depth and Width on Local Minima in Deep Learning. *Neural Computation*. Vol. 31 (7). 2019. P. 1462-1498.

22. Wu H., Gu X. Max-Pooling Dropout for Regularization of Convolutional Neural Networks. *Neural Information Processing Lecture Notes in Computer Science*. 2015. P. 46-54.

23. van Rossum G. Python Reference Manual. URL: https://docs.python.org/2.4/ref/ref.html (дата звернення 21.10.2022).

24. Abadi M., Barham P., Chen J. and others. TensorFlow: A System for Large-Scale Machine Learning. *Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation*. Savannah (USA): USENIX Association Berkeley, 2016. P. 265-283.

25. The MNIST database. URL: http://yann. lecun.com/exdb/mnist/ (дата звернення 26.10.2022).

26. Ravi S. ProjectionNet: Learning Efficient On-Device Deep Networks Using Neural Projections. 2017. 12 p. URL: https://arxiv.org/pdf/1708.00630.pdf (дата звернення 26.10.2022).

27. Granmo O.-C. The Tsetlin Machine: A Game Theoretic Bandit Driven Approach to Optimal Pattern Recognition with Propositional Logic. 2018. 36 p. URL: https://arxiv.org/pdf/1804.01508v10.pdf (дата звернення 26.10.2022).

28. Dupont E., Doucet A., The W.Y. Augmented Neural ODEs. *Conference on Neural Information Processing Systems*. Vancouver, 2020. 20 p. URL: https://arxiv.org/pdf/2008.10400.pdf (дата звернення 26.10.2022).

29. An S., Lee M., Park S. and others. An Ensemble of Simple Convolutional Neural Network Models for MNIST Digit Recognition 2020. 10 p. URL: https://arxiv.org/pdf/2008.10400.pdf (дата звернення 26.10.2022).

30. Кавецький В.В., Козловський В.О., Причепа І.В. Економічне обґрунтування інноваційних рішень. Вінниця: ВНТУ, 2016. 113 с.

31. Козловський В.О., Лесько О.Й., Кавецький В.В. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт. Вінниця: ВНТУ, 2021. 42 с.

ДОДАТКИ

Додаток А

(обов'язковий)

Технічне завдання

ЗАТВЕРДЖУЮ

в. о. завідувача кафедри АІІТ

д.т.н., проф. Бісікало О. В.

«___»_____ 2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на магістерську кваліфікаційну роботу

«Удосконалений алгоритм ADAM для навчання нейромережі»

08-31.МКР.002.02.000 ТЗ

Керівник роботи:

к.т.н., доц. каф. АІІТ

Іванов Ю. Ю.

«__» _____ 20__ р.

Виконавець:

ст. гр. 1ІСТ-21м

Данилевич М. О.

«__» _____ 20__ р.

Вінниця ВНТУ – 2022

1. Назва та галузь застосування

Оскільки всі нейрооптимізатори відрізняються своїми властивостями і особливостями реалізації, часто виникає проблема визначення найбільш ефективного алгоритму для пошуку екстремуму функції помилки, що гарантує кращі результати при вирішенні конкретної задачі. У роботі розглядається найбільш поширений на даний час оптимізатор ADAM, який за рахунок швидкої збіжності та високої точності вигідно відрізняється від інших алгоритмів оптимізації нейромереж.

2. Підстави для розробки

Розробку системи здійснювати на підставі наказу по університету №203 від 14.09.2022 та завдання до магістерської кваліфікаційної роботи, складеного та затвердженого кафедрою «Автоматизації та інтелектуальних інформаційних технологій».

3. Мета та призначення розробки

Метою роботи є підвищення ефективності розв'язання задачі навчання штучної нейромережі за рахунок використання удосконаленого алгоритму оптимізації ADAM.

4. Джерела розробки

1. Kingma D.P., Ba J.L. Adam: A Method for Stochastic Optimization. 2014 – 15 p. URL: https://arxiv.org/pdf/1412.6980.pdf (дата звернення 16.09.2022).

2. Ruder S. An Overview of Gradient Descent Optimization. Cornell University Library. URL: https://arxiv.org/abs/1609.04747 (дата звернення 16.09.2022).

3. Чопорова О.В., Кривохата А. Г. Оптимізація згорткових нейронних мереж та їх ансамблів. *Вісник Запорізького національного університету. Фізико-математичні науки.* № 1. 2019. С. 107-115.

4. Zhang A., Lipton Z.C., Li M., Smola A.J. Dive into Deep Learning. Optimization Algorithms. 682 p. URL: https://d2l.ai/chapter_optimization/index.html (дата звернення 10.09.2022).

5. Показники призначення

Розроблено програмне забезпечення, яке дозволяє підвищити ефективність розв'язання задачі оптимізації нейромережі.

Вихідні дані для роботи програми: вибірка даних; архітектура нейромережі (кількість нейронів та прошарків; структурні особливості); оптимізатор на основі адаптивної оцінки моментів (гіперпараметри: коефіцієнти навчання α та згладжування ε; оцінки моментів β_1 та β_2).

Основні технічні вимоги та мінімальні системні вимоги до програми: операційна система *Windows* 7, 8 або 10; підключення клавіатури та миші; кількість ядер комп'ютера – не менше 4 (*Core* i5 або *Ryzen* 5); оперативна пам'ять – не менше 8 ГБ; відеопам'ять – 512 Mb; жорсткий диск – 500 ГБ і більше; встановлене програмне забезпечення *Python Anaconda*.

Результати роботи програми: точність *accuracy* та втрати *loss*.

6. Економічні показники

До економічних показників входять:

- витрати на розробку – не більше 450 тис. грн.;
- вартість копії продукту – не менше 15 тис. грн.;
- мінімальна дохідність – 60 %;
- термін окупності – не менше 3-ох років;
- інші економічні переваги у порівнянні з аналогами.

7. Стадії розробки

1. Розділ 1 «Огляд предметної області роботи» має бути виконаний до 10.10.22.

2. Розділ 2 «Розробка математичної моделі удосконаленого алгоритму ADAM» має бути виконаний до 25.10.22.

3. Розділ 3 «Розробка програмного забезпечення та експериментальні дослідження» має бути виконаний до 10.11.22.

4. Економічний розділ має бути виконаний до 20.11.22.

8. Порядок контролю та приймання

1. Рубіжний контроль провести до 07.12.22.

2. Попередній захист магістерської кваліфікаційної роботи провести до 07.12.22.

3. Захист магістерської кваліфікаційної роботи провести до 23.12.22.

Додаток Б

(обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

УДОСКОНАЛЕНИЙ АЛГОРИТМ ADAM
ДЛЯ НАВЧАННЯ НЕЙРОМЕРЕЖІ

В.о. зав. кафедри АІІТ _____ д-р техн. наук, професор каф. АІІТ
Бісікало О. В.
(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Керівник роботи _____ канд. техн. наук, доцент каф. АІІТ
Іванов Ю. Ю.
(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Тех. контроль _____ канд. техн. наук, доцент каф. АІІТ
Іванов Ю. Ю.
(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Нормоконтроль _____ канд. техн. наук, доцент каф. АІІТ
Іванов Ю. Ю.
(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Опонент _____ _____
_____
(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Студент гр. 1ІСТ-21м _____ Данилевич М. О.
(підпис) (ініціали та прізвище)

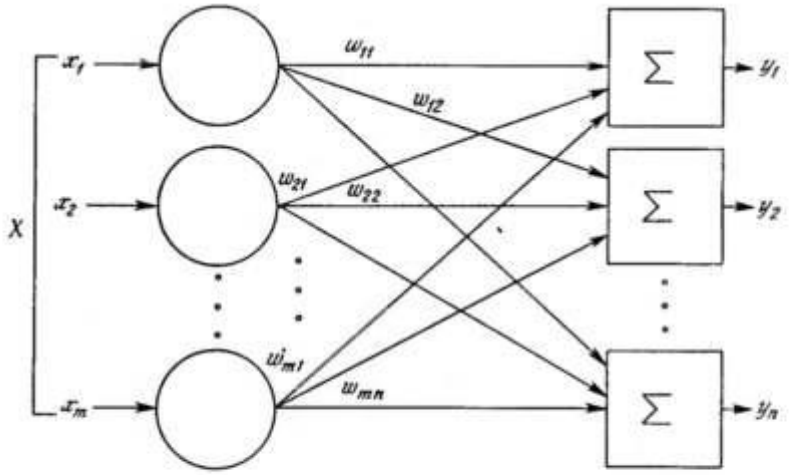

Рисунок Б.1 – Структура одношарової штучної нейромережі

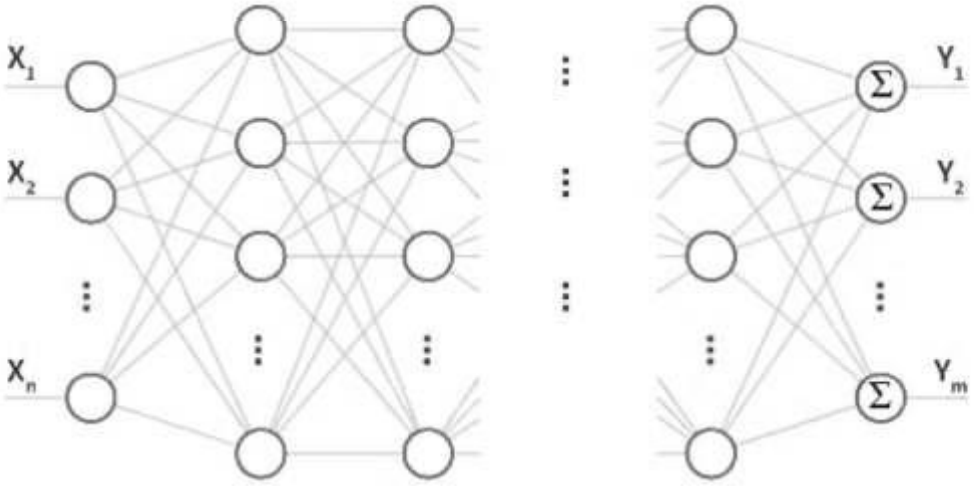

Рисунок Б.2 – Структура багатошарової штучної нейромережі

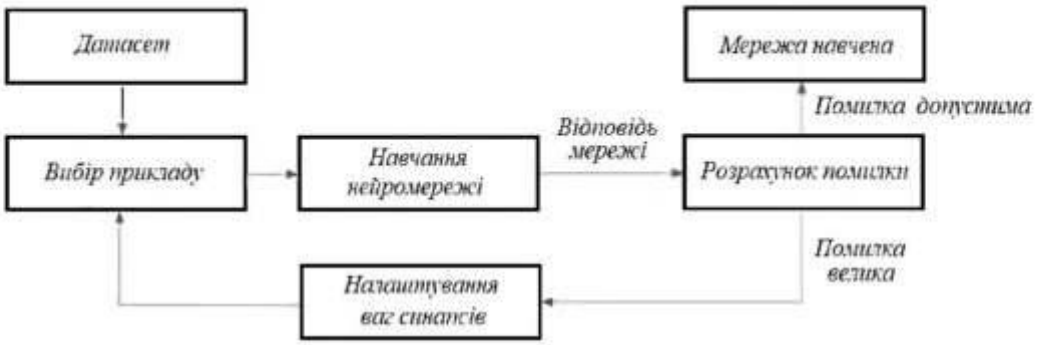

Рисунок Б.3 – Структура для навчання нейромережі

Рисунок Б.4 – Нейромережеві оптимізатори

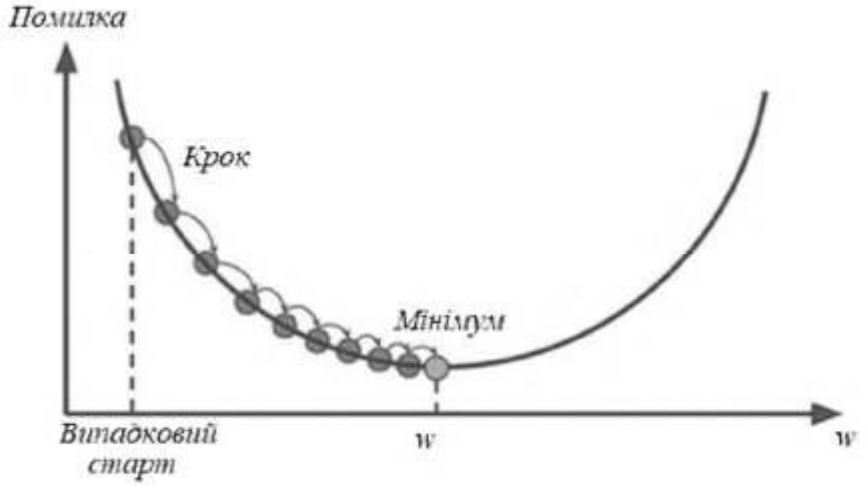

Рисунок Б.5 – Принцип роботи градієнтних алгоритмів оптимізації

Рисунок Б.6 – Схема програми

Додаток В

(обов'язковий)

Лістинг програми

```
##########################################################################
from sklearn.cross_validation import train_test_split
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
n_feature = 2
n_class = 2

def make_network(n_hidden=100):
    model = dict(
    W1=np.random.randn(n_feature, n_hidden),
     W2=np.random.randn(n_hidden, n_class)
    )
    return model

def get_minibatch_grad(model, X_train, y_train):
    xs, hs, errs = [], [], []
    for x, cls_idx in zip(X_train, y_train):
        h, y_pred = forward(x, model)
        y_true = np.zeros(n_class)
        y_true[int(cls_idx)] = 1.
        err = y_true - y_pred
        xs.append(x)
        hs.append(h)
        errs.append(err)
    return backward(model, np.array(xs), np.array(hs), np.array(errs))

def get_minibatch(X, y, minibatch_size):
    minibatches = []
    X, y = shuffle(X, y)
    for i in range(0, X.shape[0], minibatch_size):
        X_mini = X[i:i + minibatch_size]
        y_mini = y[i:i + minibatch_size]
        minibatches.append((X_mini, y_mini))
    return minibatches

def adam(model, X_train, y_train, minibatch_size):
    M = {k: np.zeros_like(v) for k, v in model.items()}
    R = {k: np.zeros_like(v) for k, v in model.items()}
    beta1 = .9
    beta2 = .999
    minibatches = get_minibatch(X_train, y_train, minibatch_size)
```

```
    for iter in range(1, n_iter + 1):
        t = iter
        idx = np.random.randint(0, len(minibatches))
        X_mini, y_mini = minibatches[idx]
        grad = get_minibatch_grad(model, X_mini, y_mini)
        for k in grad:
            M[k] = beta1 * M[k] + (1. - beta1) * grad[k]
            R[k] = beta2 * R[k] + (1. - beta2) * grad[k]**2
            m_k_hat = M[k] / (1. - beta1**(t))
            r_k_hat = R[k] / (1. - beta2**(t))
            model[k] += alpha * m_k_hat / (np.sqrt(r_k_hat) + eps)
    return model

def cor_adam(model, X_train, y_train, minibatch_size):
    M = {k: np.zeros_like(v) for k, v in model.items()}
    R = {k: np.zeros_like(v) for k, v in model.items()}
    D = {k: np.zeros_like(v) for k, v in model.items()}
    beta1 = .9
    beta2 = .999
    minibatches = get_minibatch(X_train, y_train, minibatch_size)
    for iter in range(1, n_iter + 1):
        t = iter
        idx = np.random.randint(0, len(minibatches))
        X_mini, y_mini = minibatches[idx]
        grad = get_minibatch_grad(model, X_mini, y_mini)
        for k in grad:
            M[k] = beta1 * M[k] + (1. - beta1) * grad[k]
            R[k] = beta2 * R[k] + (1. - beta2) * grad[k]**2
            m_k_hat = M[k] / (1. - beta1**(t))
            r_k_hat = R[k] / (1. - beta2**(t))
            D[k] = beta1 * (m_k_hat / (np.sqrt(r_k_hat) + eps)) + (1-beta1)*np.sign(grad[k])*
D[k]
            model[k] += alpha * D[k]
    return model

    X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
    n_iter = 5000
    eps = 1e-8  # Smoothing to avoid division by zero
    alpha =  2e-3
    minibatch_size = 250
    n_experiment = 10
    algos = dict(
        adam = adam,
        mod_adam = mod_adam
    )
    algo_accs = {k: np.zeros(n_experiment) for k in algos}
```

```
for algo_name, algo in algos.items():
    print('Experimenting on {}'.format(algo_name))
    for k in range(n_experiment):
        model = make_network()
        model = algo(model, X_train, y_train, minibatch_size)
        y_pred = np.zeros_like(y_test)
        for i, x in enumerate(X_test):
            _, prob = forward(x, model)
            y = np.argmax(prob)
            y_pred[i] = y
        algo_accs[algo_name][k] = np.mean(y_pred == y_test)
print()
for k, v in algo_accs.items():
    print('{} => mean accuracy: {}, std: {}'.format(k, v.mean(), v.std()))
```

##

Додаток Г

(обов'язковий)

Довідка про можливість впровадження розробки

Додаток Д

(обов'язковий)

Протокол перевірки МКР

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Удосконалений алгоритм ADAM для навчання нейромережі.

Тип роботи: магістерська кваліфікаційна робота

Підрозділ: кафедра автоматизації та інтелектуальних інформаційних технологій, факультет інтелектуальних інформаційних технологій та автоматизації

Показники звіту подібності Plagiat.pl (StrikePlagiarism)

Оригінальність 97,9% Схожість 2,1%

Аналіз звіту подібності (відмітити потрібне):

■ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату

☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.

☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень

Особа, відповідальна за перевірку _____ (підпис) Маслій Р. В. (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Plagiat.pl (StrikePlagiarism) щодо роботи.

Автор роботи _____ (підпис) Данилевич М. О. (прізвище, ініціали)

Керівник роботи _____ (підпис) Іванов Ю. Ю. (прізвище, ініціали)