Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Метод розпізнавання рукописних математичних формул

(тема)

Виконав

студент II курсу, групи КСМм-22-1

Шаповалов М.І.
(прізвище, ініціали)

Спеціальність

123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма

Комп’ютерні системи та мережі
(повна назва освітньої програми)

Керівник: проф. Кучук Н.Г.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп’ютерні системи та мережі
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Шаповалову Максиму Ігоровичу
(прізвище, ім’я, по батькові)

1. Тема роботи Метод розпізнавання рукописних математичних формул

затверджена наказом по університету від “ 06 ” листопада 2023 р. № 1298 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 15 січня 2024 р.

3. Вхідні дані до роботи 1) Файли формату *.py, *.pb, *.txt, які містять послідовність виконуваних функцій. 2) Операційна система –Windows 11.
3) середовища розробки – JetBrains. PyCharm Community Edition

4. Перелік питань, що потрібно опрацювати у роботі
1) Аналітичний огляд.
2) Розробка системи контролю ходу виконання завдань.
3) Програмна реалізація системи та її тестування.
4) Тестування розробленої ЗНМ
5) Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів)

Слайд-презентація – 13 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної області	07.11.23-13.11.23	
2	Вибір та обгрунтування методики дослідження	14.11.23-20.11.23	
3	Вибір інструментальних засобів	21.11.23-23.11.23	
4	Розробка ЗНМ	24.11.23-06.12.23	
5	Проведення експериментів	07.12.23-23.12.23	
6	Оформлення матеріалів кваліфікаційної роботи	26.12.23-02.01.24	
7	Подання кваліфікаційної роботи керівникові та	03.01.24-06.01.24	
	її попередній захист		
8	Подання кваліфікаційної роботи на рецензу-	09.01.24-12.01.24	
	вання		

Дата видачі завдання 06 листопада 2023 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Кучук Н.Г.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 66 с., 28 рис., 1 табл., 3 дод., 20 джерел.

РОЗПІЗНАВАННЯ РУКОПИСНИХ МАТЕМАТИЧНИХ ФОРМУЛ, ЗАСТОСУНОК, НЕЙРОННА МЕРЕЖА, ЗГОРТАЛЬНА НЕЙРОННА МЕРЕЖА.

Метою кваліфікаційної роботи є розробка застосунку для розпізнавання рукописних математичних формул, що полегшить переведення рукописів у друкований вигляд та допоможе людям зекономити власний час. Програмний застосунок розроблявся з використанням МП Python, бібліотек TensorFlow, Keras, NumPy, OpenCV.

У ході виконання кваліфікаційної роботи була розглянута тема розпізнавання рукописних математичниї формул. В результаті виконання був створений десктопний застосунок, що дозволяє розпізнати написаі власноруч математичні формули за допомогою згорткових нейроних мереж, мови програмування Python та бібліотек, пов’язаних із штучнним інтелектом та нейронними мережами.

ABSTRACT

Master’s thesis: 66 pages, 28 figures, 1 tables, 3 appendices, 20sources.

FIREWALL, RECOGNITION OF HANDWRITTEN MATHEMATICAL FORMULAS. EXPRESSIONS, APPLICATION, NEURAL NETWORK, CONVERGAL NEURAL NETWORK.

The object of research is the process of recognition handwritten texts. The subject of research is the means and methods of building a neural network using modern technologies. The goal of the work is to develop an application for recognizing handwritten mathematical expressions, which will facilitate the translation of manuscripts into printed form and help people save their own time. The software application was developed using MP Python, TensorFlow, Keras, NumPy, OpenCV libraries.

During the master’s thesis the topic of recognizing handwritten mathematical formulas was considered. As a result of the implementation, a desktop application was created that allows you to recognize inscriptions and mathematical formulas themselves using convolutional neural networks, the Python programming language and libraries related to artificial intelligence and neural networks.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

АЗПП – алгоритм зворотного поширення помилки

БД – база даних

БЖ – блок живлення

ЕОМ – електроно-обчислювальна машин

ЗНМ – згорткові нейроні мережі

ІТ – інформаційні технології

МП –мова програмування

ОС – операційна система

ООП – об’єкто-орієнтоване програмування

ПЗ – програмне забезпечення

ПП – програмний продукт

ЦП –центральний процесор

ВСТУП

У наш час використання такої галузі штучного інтелекту, як нейронні мережі, набуває все більшої і більшої популярності та використовується у великій кількості областях. Починаючи від обробки зображень і закінчуючи обробкою та аналізом великих масивів даних від космічних телескопів. Наразі нейромережі активно використовуються у сфері розпізнавання рукописних текстів, що дає змогу екомити час, що міг бути витрачений на переписування тексту, можливість швидкого розуміння написаного, якщо почерк людини є не дуже розбірливим, можливість редагування рукописного тексту під час його оцифрування та ін. Серед розпізнавання текстів можна виділити розпізнавання математичних формул та формулви, що дозволяє швидше оброблювати дані та допомогає оптимізувати час.

Актуальність розробки полягає в тому, що можливість розпізнвати рукописні математичні формули дає можливість зробити зручнішою роботу з розрахунками шляхом їх швидкого переводу з рукописного в печатний вигляд, і так чином допомгає оптимізувати власний час.

Об’єктом дослідження є процес розпінавання рукописного тексту.

Предметом дослідження є засоби та методи побудови нейронної мережі із використанням сучасних техологій.

Метою роботи є розробка застосунку для розпізнавання рукописних математичних формул, що полегшить переведення рукописів у друкований вигляд та допоможе людям зекономити власний час.

Для досягнення поставленої мети були поставлені наступні задачі:

- провести аналітичний огляд існуючих застосунків;
- обрати мову програмування, підходи до розробки застосунку та сучасні технології програмування;
- реалізувати застосунок з розпізнавання рукописних математичних формул;

- виконати тестування розробленого застосунку.

Застосунок розроблявся з використанням МП Python, бібліотек TensorFlow, Keras, NumPy, OpenCV, та ЗНМ.

Практична цінність програмної реалізації застосунку для розпізнавання математичних формул полягає в тому, що люди, пов’язані з математикою чи оцифровкою математичних формул, зможуть оптимізувати власний час.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Переваги та недоліки існуючих застосунків для розпізнавання рукописних математичних формул

Застосунок для розпізнавання рукописних математичних формул – це програмне забезпечення, що дозволяє швидко оцифрувати написані від руки математичні формули, формули та символи. Хоча, як здається, швидше усе передрукувати у смартфон, комп’ютери чи іншу ЕОМ або взагалі вести розрахунки на ній, виникають ситуації, коли під рукою тільки ручка та папір, а те, що написано, хочеться швидко перекинути на ЕОМ у друкованому вигляді. Ось тут і потрібні застосунок для розпізнавання рукописних математичних формул, що полегшить тим людям роботу та зекономлять час, який може бути витрачений с користю. Окрім цього, подібні застосунки використовуються для оцифрування давніх рукописів і дуже допомагають, особливо якщо почерк автора рукопису є не дуже зрозумілим.

Тож застосунки для розпізнавання рукописних математичних формул мають свою нішу і стають все більш розповсюдженими.

Перед початком розробки програмного забезпечення (ПЗ) потрібно зробити огляд готових програмних рішень для розпізнавання математичних формул, з’ясувати їхні плюси та мінуси, схожі та відмині риси.

Розглянемо Detexify – це веб-застосунок, що дозволяє намалювати будь-який символ, після чого він розпізнає його та надає найбільш вірогідні символи, що є в його БД [1]. Веб-застосунок є безкоштовним та open-source - йогой сирцевий код є доступним у репозиторії GitHub. Також окремо для MacOS був випущений застосунок на платній основі із розширеним функціоналом. Серед переваг Detexify можна виділити швидку обробку введених рукописних символів, безкоштовний функціонал та відкритість необробленого коду. Серед недоліків Detexify можна виділити здатність

розпізнавати лише один символ за раз та відсутність можливості завантажити власне зображення (рисунок 1.1).

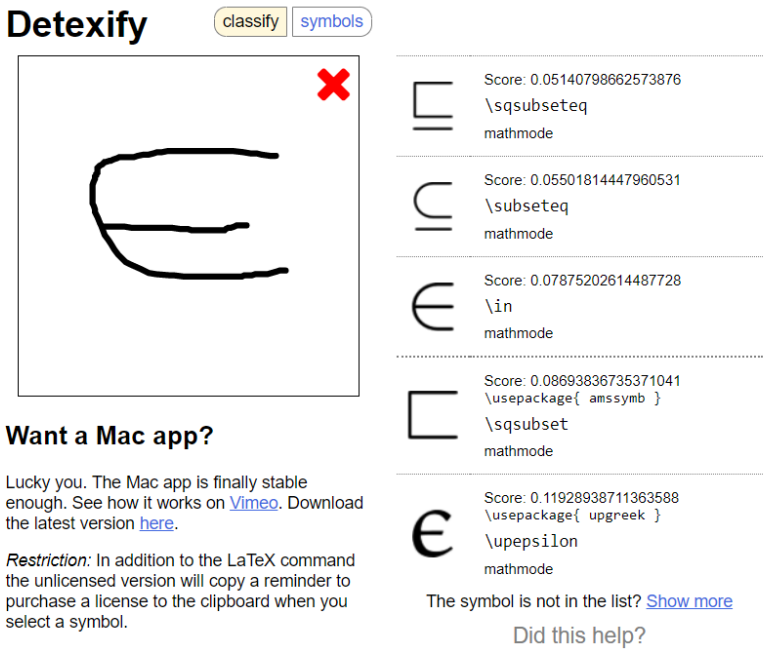

Рисунок 1.1 – Зображення веб-сторінки Detexify

Далі ми розберемо Google Docs – це онлайн-редактор текстових документів, який є частиною офісного пакету Google WorkSpace (рисунок 1.2).

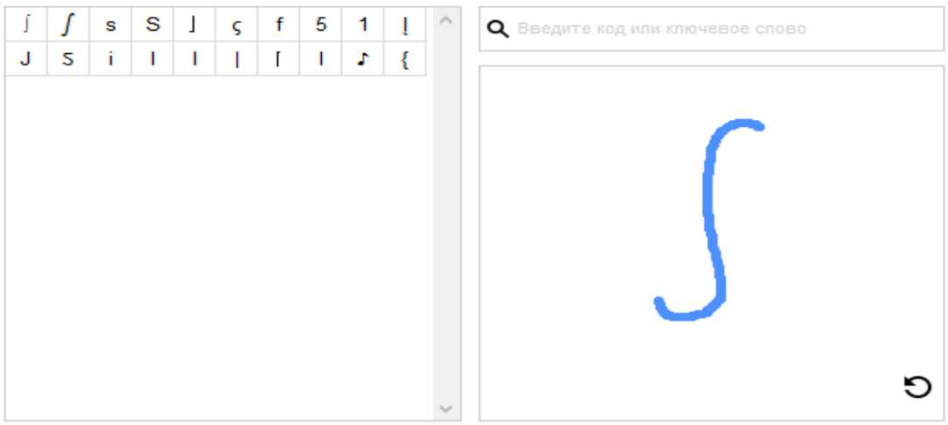

Рисунок 1.2 – Зображення веб-сторінки розпізнавача рукописних символів у Google Docs

Серед багатьох можливостей цей сервіс має функцію, близьку до функціональності розглянутого раніше застосунка – розпізнавання рукописного символу і надання можливих варіантів[2].

Виділимо позитивні сторони Google Docs:

- швидка обробка введених рукописних символів;
- безкоштовний функціонал;
- великий вибір результатів пошуку.

До недоліків Google Docs можна віднести здатність розпізнавати лише один символ за раз.

Щоб намалювати повноцінний математичний вираз, подумайте про розпізнавання математичних виразів. Цей веб-застосунок після обробки вхідного рукопису генерує розпізнані символи у форматах LaTex [3]. Веб-застосунок є безкоштовним із відкритим вихідним кодом – його вихідний код доступний у репозиторії GitHub (рисунок1.3).

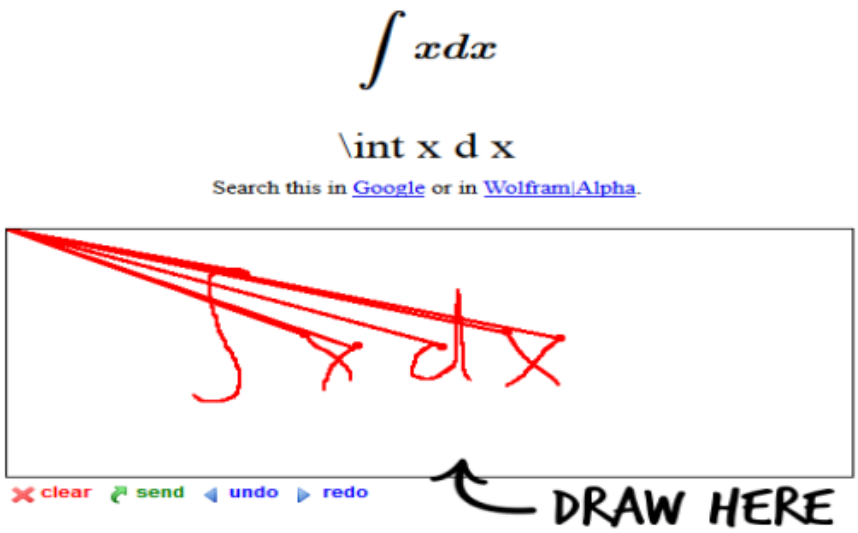

Рисунок 1.3 – Зображення веб-сторінки розпізнавача рукописних символів у Mathematical Expression Recognition

Переваги розпізнавання математичних виразів включають уміння опрацьовувати повноцінну математичну формулу, безкоштовний функціонал, можливість копіювати перекладений вираз у форматі LaTex.

Серед недоліків розпізнавання математичних виразів можна виділити перебої в роботі веб-застосунку, відсутність перекладу рукопису у формат Unicode.

Photomath – це програма для iOS та Android, розроблена Даміром Саболом і реалізована на основі Photomath, яка дозволяє розпізнавати та розв’язувати прості математичні формули, фотографуючи рукописні формули [4]. Dodakok розпізнає шаблон, за яким написаний вираз, і пропонує коротке рішення. Повний доступ, необмежену кількість програм і доступ до освітніх професійних відеоуроків можна отримати, підписавшись на Photomath Plus на місяць або рік.

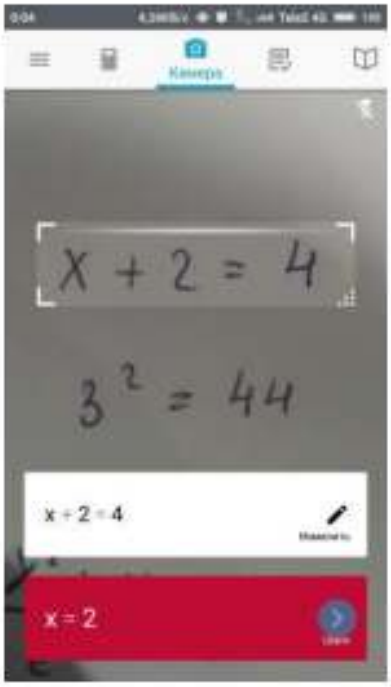

Рисунок 1.4 – Зображення Android-версії застосунку застосунку Photomath під час розв’язку математичної формули

Переваги розпізнавання математичних виразів включають уміння опрацьовувати повноцінну математичну формулу, за допомогою фотоапарата надрукувати рукописну фразу, уміння розгадувати рукопис.

Серед недоліків розпізнавання математичних виразів можна виділити обмежений функціонал у безкоштовній версії.

Далі виділимо Mathpix [5] – це крос-платформний застосунок (Windows, MacOS, Linux, iOS, Android, веб-застосунок), розроблений і підтримуваний Mathpix. Розпізнає рукописний текст, включаючи математичні формули та формули, і переводить його у формати LaTex, Word тощо, відображаючи при цьому правильне написання формул. Отримати розширений функціонал у вигляді збільшення кількості оцифровок рукописних текстів, створення PDF-файлів, отримання індивідуальних консультацій, кількості пристроїв, підключених до одного облікового запису, можливо, придбавши певний тип підписки.

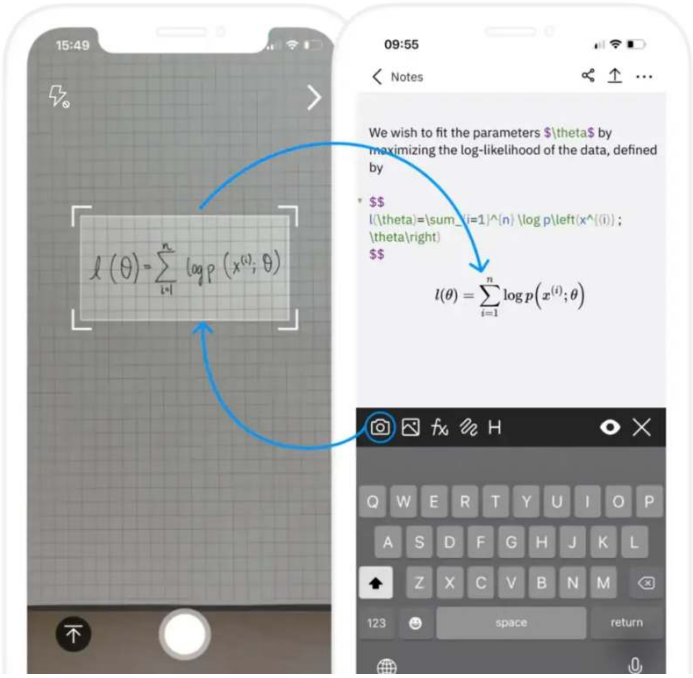

Рисунок 1.5 – Зображення iOS-версії застосунку Mathpix під час оцифрування математичної формули

Серед переваг Mathpix можна виділити:

- кросплатформенність;
- можливість використання різних способів введення рукописного тексту (за допомогою фотоапарата, надсилання фото чи скану);

- підтримка розробників;
- конвертація оцифрованого рукописного тексту в різні формати.

Серед недоліків розпізнавання математичних виразів можна виділити обмежений функціонал у безкоштовній версії.

У таблиці 1.1 наведено порівняльний аналіз представлених попередніх аналогів.

Таблиця 1.1 – Порівняння аналогів

Критерій	Аналог				
	Detexify	Google Docs	MER	Photomath	Mathpix
Онлайн	+	+	+	-	+
Офлайн	-	-	-	+	+
LaTex	+	-	-	+	+
Unicode	-	+	-	-	+
Розпізнавання декількох символів	-	-	+	+	+
Розпізнавання дробних формул	-	-	+	+	+
Розпізнавання формул з інтегралами	-	-	+	+	+
Покроковий роз’вязок	-	-	-	+	-
Наявність преміум-функцій	+-	-	-	+	+

Як видно, онлайн-спосіб обробки рукописного тексту є більш поширеним серед аналогів, тоді як програма Photomath єдина, яка розпізнає рукописний текст офлайн, а програма MathPix має як веб-версію, так і програми, які мають говорити про легітимність Саме веб-застосунок сьогодні

доводить, що не варто покладатися виключно на веб-застосунки, а потрібні рішення, які можуть працювати незалежно від Інтернету.

1.2 Особливості розпізнавання рукописного тексту

Розпізнавання рукописного тексту – це здатність комп'ютера отримувати та інтерпретувати рукописну інформацію з таких носіїв, як паперові документи, фотографії, сенсорні екрани тощо. Розпізнавання ділиться на два методи онлайн розпізнавання рукописного введення та офлайн розпізнавання рукописного введення.

Онлайн-розпізнавання рукописного тексту – перетворюйте рукописний текст, зчитуючи текст, що зберігається на пристроях введення: сенсорних екранах, планшетах для малювання тощо, у режимі реального часу, збираючи інформацію, таку як час і координати кожного рядка рукописного тексту. Цей метод реалізується за допомогою спеціального апаратного або програмного забезпечення та може бути розділений на такі етапи:

- повторна обробка;
- виділення ознак;
- кваліфікація.

Розпізнавання рукописного тексту в автономному режимі – перетворення тексту, розміщеного на зображенні, на літерні коди, які можна використовувати для подальшої обробки тексту на комп'ютері. Цей спосіб повністю перетворює вже написану від руки інформацію в друковану форму. Цей метод має різні способи реалізації:

- розпізнавання символів;
- виділення символів;
- виділення ознак;
- сегментація на основі машинного навчання;
- сегментація за допомогою згорткових нейронних мереж [6].

1.3 Особливості розпізнавання математичних виразів

Розпізнавання математичних виразів є складнішим видом розпізнавання, ніж розпізнавання символів, оскільки крім розпізнавання безпосередньо математичного символу необхідно розпізнати структуру математичного висловлювання.

В он-лайн розпізнаванні рукописного математичного виразу зазвичай виділяють такі стадії:

- збір та попередня обробка даних;
- поділ виразу на символи (сегментація виразу);
- розпізнавання окремих символів;
- розпізнавання структури математичного виразу.

Першим кроком є збір та попередня обробка даних. Навчальна множина для конкретного класу символів складається із множини символів. Символ є впорядкованим набором розчерків, а розчерк є впорядкованим набором пар (x, y), де перша пара відповідає точці торкання пера, а остання точці відриву. Кожен символ малюється певною кількістю розчерків у певному порядку.

Метою попередньої обробки даних є перетворення «сирих» даних розчерків у формат, який дозволяє створити модель символу з його подальшої класифікації. Для спрощення моделі необхідно видалити інформацію, яка не потрібна при класифікації. Початкова множина розчерків перетворюється на вектор певної та жорстко фіксованої розмірності.

Масштабування, зсув та зміна кількості опорних точок відбувається наступним чином. Кожна точка кожного розчерку зсувається так, що початкова точка є верхнім лівим кутом обмежуючої рамки символу. Після цього всі точки масштабуються зі збереженням відношення ширини до висоти так, що символ міститься всередині квадрата. Далі набір точок кожного розчерку змінюється так, щоб точки розташовувалися рівновіддалено (передискретизація по довжині дуги методом лінійної

інтерполяції). В результаті отримано набір векторів однієї розмірності, де кожному класу символів відповідає певна кількість векторів. Цю інформацію можна використовувати, наприклад, для навчання нейронної мережі, або для розпізнавання символу алгоритмом k-найближчих сусідів.

Розмірність вектора можна зменшити в кілька разів, при цьому не втративши ніякої інформації з використанням методу основних компонент.

У результаті створюється масив для навчання класифікатора.

На наступному етапі проводиться сегментація та розпізнавання окремих символів. Класифікатор розпізнає символи окремо. Однак, коли необхідно розпізнавати потокове введення, він не може заздалегідь знати, які розчерки становлять символ, і скільки символів введено. Отже, можливості класифікатора необхідно доповнити визначенням кількості та розташування символів виразу. Вирішення цієї проблеми еквівалентно найкращому угрупованню розчерків у символи, або сегментацією виразу.

Припустимо, що введені розчерки впорядковані за часом, тобто при введенні символу, що складається з трьох розчерків, послідовно вводяться розчерки символу. Наприклад, не допускається ставити крапку над символом «і» після написання символів, які йдуть після «і». З урахуванням цього припущення, визначення символів, які складаються з кількох розчерків, необхідно послідовно розглядати групи розчерків. Якщо символ з кількох розчерків розпізнається класифікатором з помилкою, меншою за певний поріг, то віддається перевага цьому символу. Якщо жоден з результатів класифікації не перевищує граничного значення, граничне значення збільшується і крок повторюється, або виникає помилка розпізнавання.

Слід зазначити, деякі символи, які неможливо розпізнати як різні (наприклад, знаки мінуса і дроба), позначаються як символ (горизонтальна риска), який перейменується під час структурного аналізу.

Алгоритм сегментації можна покращити, якщо для кожної кількості розчерків визначити допустимі символи. Так, два розчерки не можуть бути розпізнані як символ, що складається з одного або трьох розчерків.

Для того, щоб запобігти можливим помилкам сегментації, потрібен інтерфейс, що дозволяє миттєво виправити некоректний результат, або скасувати введення останнього символу.

Багато алгоритмів розпізнавання математичних виразів дозволяють неоднозначності. Класифікатор не може знати, введено символ «мінус» або символ «дріб». Для цього потрібна додаткова перевірка. Так, на певній відстані вгору і вниз від риси символу «мінус» не буде розміщено жодних розчерків, а для символу «дріб» – буде. На підставі цього проводиться перейменування символу "риска" на символ "мінус" або "дріб" відповідно. Практично аналогічно аналізуються такі неоднозначності як, наприклад, десяткова точка та знак множення.

Далі слідує етап розпізнавання структури виразу. Відношення між математичними операторами визначаються залежно від позиції та відносного розміру символу у виразі. Для визначення відносин використовуються просторові області «згори ліворуч», «зверху», «верхній індекс», «нижній індекс», «знизу», «знизу ліворуч» та «підвираз». Наприклад, очікується, що операнди (чисельник і знаменник) горизонтальної лінії (оператора дробу) лежатимуть у областях «згори» та «знизу» щодо горизонтальної лінії.

Виходячи з простих атрибутів (верхня границя, права границя тощо), можна визначити поріг верхнього індексу, поріг нижнього індексу та інші атрибути. Це чисельні атрибути, які використовуються для відокремлення областей навколо символу. Також з простих атрибутів можна визначити центральну точку.

Центральна точка – це точка, яка визначає розташування символу в будь-якій області. Для визначення атрибутів символ класифікується як символ з надрядковим елементом, символ з підрядковим елементом або центральний символ. Кожен клас символів має атрибути, що по-різному визначаються. Це необхідно для того, щоб уникнути неоднозначності при структурному аналізі:

Базова лінія – це список, який представляє горизонтальне розташування символів у виразі. Кожен символ має посилання на інші базові лінії, які задовольняють різним просторовим відношенням.

Домінування визначається таким чином: символ «s» домінує над символом «a», якщо «a» лежить у сфері «s», а «s» лежить у області «a». Однак обидва символи можуть лежати і в областях один одного. Якщо є впорядкований список символів, то можна визначити крайній лівий домінуючий символ у списку.

Домінуюча базова лінія виразу – це базова лінія, на яку не посилається символ. Під час читання математичного виразу зазвичай спочатку проводиться пошук крайнього лівого домінуючого символу, потім наступного за ним домінуючого символу, і так далі, доки не будуть знайдені всі символи базової лінії.

Структура даних, що виражає, називається деревом базових ліній. Дерево базових ліній створюється рекурсивним знаходженням домінуючих базових ліній у виразі, описаному упорядкованим списком символів.

Отримане дерево є структурою математичного висловлювання, і її можна перетворити на код формульного редактора.

1.4 Використання нейронних мереж для розпізнавання рукописних формул

Людина найбільше сприймає інформацію через зір. Це сприймається як належне та щось просте, проте насправді це дуже складний процес, що потребував багато років еволюції в минулому та постійного несвідомого навчання з народження. Усю інформацію, що сприймають наші очі оброблює зорова кора головного мозку, що розміщена на потилиці та складається з приблизно 140 мільйонів нейронів, між якими існують мільярдів зв’язків, що аналізують величезні обсяги інформації та перетворює у те, що ми сприймаємо, коли бачимо щось [7].

На основі вище написаного можна зробити висновок, що ідея нейронних мереж – розробка певної системи, що використовує для навчання великі обсяги інформації з метою автоматичного визначення правил розпізнавання. І якщо збільшити кількість навчальної інформації, то можна нейронну мережу навчити дізнаватись більше, тобто отримувати більше інформації з тих же вхідних даних.

Основою нейронної мережі є перцептрон – структура, в якої декілька двійкових входів та один двійковий вихід, що можна побачити на рисунку 1.6.

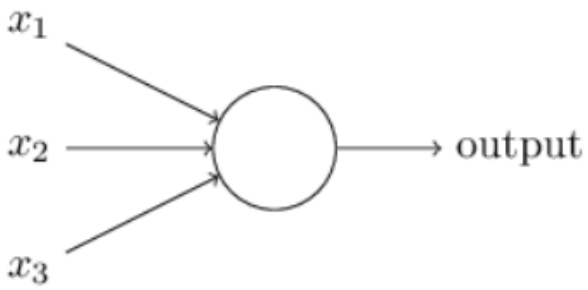

Рисунок 1.6 – Стандартний вигляд перцептрона

Його суть полягає в тому, що він приймає рішення, враховуючи певні свідчення, як от людині потрібно виконати певну дію і перед її виконанням вона зважує певні чинники та фактори.

Ці фактори можна представити у двійковому вигляді. Вихід 0 або 1, визначається тим, чи зважена сума $\Sigma_j w_j x_j$ менше або більше деякого порогового значення. Як і вагові коефіцієнти, поріг – це дійсне число, яке є параметром. Інакше це може бути виражено формулою 1.1:

$$output = \begin{cases} 0, & якщо \sum_{j=1}^{J} w_j x_j \le \delta; \\ 1, & якщо \sum_{j=1}^{J} w_j x_j > \delta, \end{cases} \tag{1.1}$$

де δ – порогове значення.

Під час створення моделі ці фактори – це коефіцієнти, що є вагами умов. Змінюючи ваги певних умов перцептрон може видати якесь рішення, при цьому змінивши деяки коефіцієнти на тій самій моделі можна отримати геть інший результат [8].

Очевидно, що перцептрон не є повною моделлю прийняття рішень людиною. Але приклад ілюструє те, як перцептрон може зважувати різні типи доказів для прийняття рішень. І повинно здатися правдоподібним, що складна мережа перцептронів, що зображена на рисунку 1.7, може приймати досить тонкі рішення.

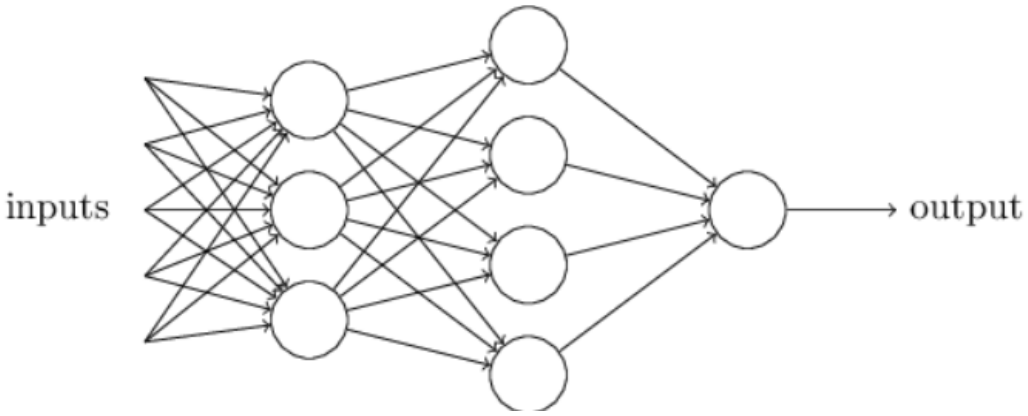

Рисунок 1.7 – Приклад складного перцептрона

Отже математичний апарат нейронних мереж може бути використаним і для вирішення такого складного завдання, як розпізнавання рукописних математичних виразів.

2 ОГЛЯД МОЖЛИВОСТЕЙ НЕЙРОННИХ МЕРЕЖ

2.1 Функції активації нейронних мереж

Кожна функція активації (або нелінійність) приймає одне число і виконує певну фіксовану математичну операцію на ньому. Існує кілька функцій активації, що зустрічаються на практиці:

- сигмоїда (sigmoid);
- тангенсоїда (tahn);
- функція активації ReLU (Rectified Linear Unit) [9].

Розглянемо особливості кожної з цих функцій активації окремо. Почнемо з найбільш відомої – сігмоїди.

Сигмоїдна нелінійність представлена на графіку рисунку 2.1.

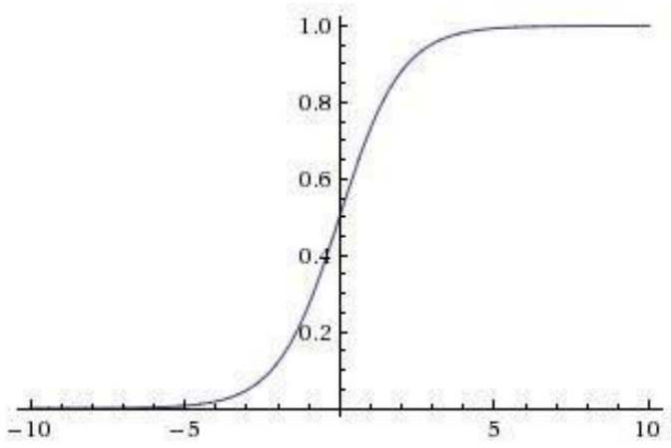

Рисунок 2.1 – Сигмоїдальна нелінійність, яка обмежує дійсні числа до діапазону [0,1]

Сигмоїда приймає дійсне число і "стискує" його в діапазон від 0 до 1. Зокрема, великі негативні числа стають нулями, а великі позитивні числа стають одиницями. Сигмоїдна функція часто зустрічається в історичному відношенні оскільки вона має приємну інтерпретацію як швидкість руху

нейрона: від не спрацювання зовсім (0) до повністю насиченого спрацювання при прийнятній максимальної частоті (1). Сама сигмоїда може виражатися через формулу 2.1:

$$f(s) = \frac{1}{1 + e^{-s}} \tag{2.1}$$

На практиці сигмоїдна нелінійність останнім часом випала з користі, і вона рідко використовується. Вона має два основних недоліки:

Сигмоїда насичується і вбиває градієнти. Дуже небажаним властивістю сигмоєвидного нейрону є те, що коли активація нейрона насичується, у будь-якому хвості від 0 або 1, градієнт у цих областях майже дорівнює нулю. При зворотному розповсюдженні цей (локальний) градієнт буде помножений на градієнт виходу цього нейрону. Тому, якщо локальний градієнт дуже малий, він фактично «вб'є» градієнт, і практично не буде ніякого сигналу протікати через нейрон до його ваг та рекурсивно до його даних. Окрім того, потрібно приділяти особливу обережність при ініціалізації ваг сигмоєвидних нейронів, щоб запобігти насиченість. Наприклад, якщо початкові ваги занадто великі, більшість нейронів стануть насиченими, і мережа ледь навчиться. Виходи сигмоїди не нульові. Це небажано, оскільки нейрони у більш пізніх шарах обробки в нейронній мережі будуть отримувати дані, які не нульові. Це має наслідки для динаміки під час градієнтного спуску, тому що якщо дані, що надходять в нейрон, завжди позитивні (наприклад, x > 0 елементарно в $f = w^T x + b$)), то градієнт на вагах w при зворотному поширенні стане будь-яким з усіх позитивний або усіх негативних (залежно від градієнта всього виразу f). Після того, як ці градієнти будуть додані до частини даних, остаточне оновлення для ваг може мати змінні ознаки, що трохи пом'якшує цю проблему. Тому це є незручно, але має менш серйозні наслідки в порівнянні з насиченою проблемою активації вище.

Далі розглянемо тангенсоїду. Ця функція обмежує дійсне число до діапазону [-1, 1]. Подібно сигмоєвидному нейрону, його активація

насичується, але на відміну від сигмоєвидного нейрона його вихід є відцентрованим відносно нуля. Тому на практиці віддають перевагу тангенційній нелінійності, ніж сигмоподібній нелінійності. Тангенційний нейрон - це масштабований сигмоподібний нейрон, графік зображено на рисунку 2.2.

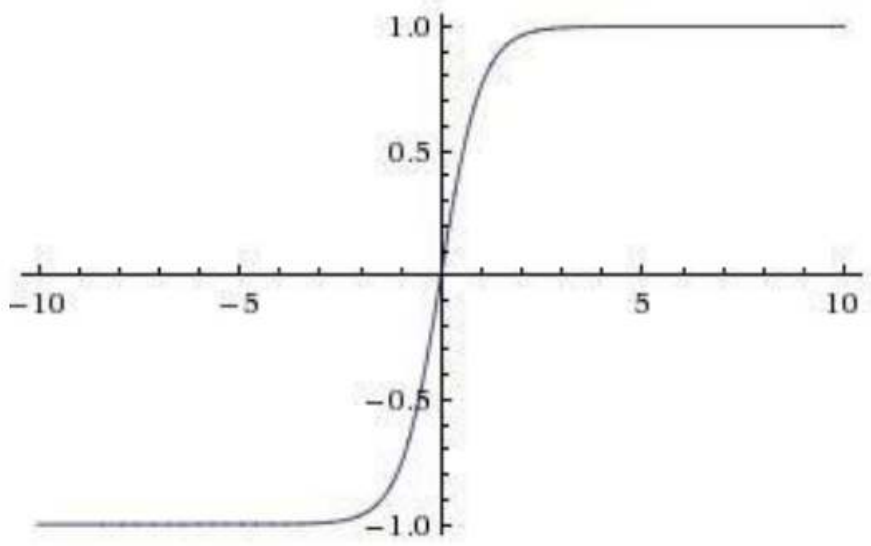

Рисунок 2.2 – Тангенційна нелінійність, яка обмежує дійсні числа до діапазону [-1,1]

Сама тангенсоїда може виражатися через формулу 2.2:

$$f(s)=\frac{e^{2s}-1}{e^{2s}+1};$$
$$f'(s)=1-f^2(s). \qquad (2.2)$$

Активаційна функція ReLU дуже часто використовується останніми роками. Її математична формула має вигляд, який можна побачити на формулі 2.3, а її графік зображено на рисунку 2.3:

$$f(x) = \max(0, x), \qquad (2.3)$$

тобто активація функції відбувається при перетині нуля.

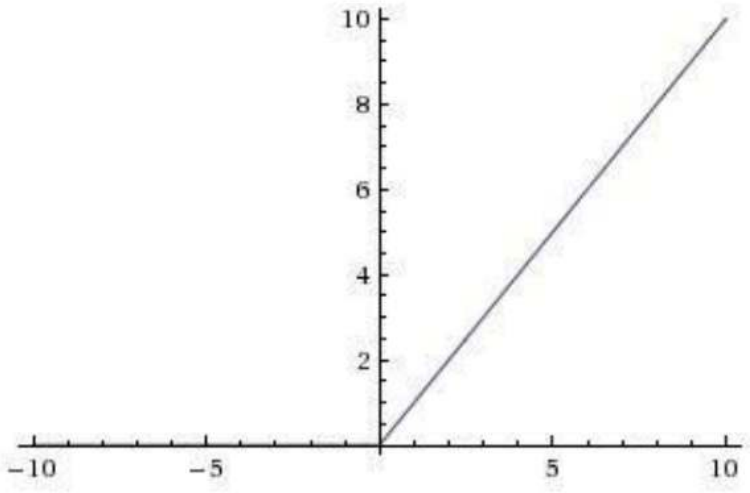

Рисунок 2.3 – Активаційна функція ReLU, яка приймає значення 0, коли x < 0 і є лінійно зростаючою, коли x > 0 [10]

У порівнянні із попередньо розглянутими функціями, нейрони, що будувались на їхній основі, нейрони на основі функції ReLU можуть бути реалізованим шляхом перетинання матриці активації нуля. Проте вони є досить нестабільними та в процесі навчання можуть “відмерти”.

2.2 Згорткові нейроні мережі

Згорткові або конволюційні нейронні мережі (ЗНМ) подібні до звичайних нейронних мереж: вони також побудовані на основі нейронів, які володіють постійно змінюваною вагою і зсувами. Кожен нейрон отримує деякі вхідні дані, виконує скалярний додуток інформації і в окремих ситуаціях супроводжує це нелінійностью. Як і у випадку зі звичайними нейронними мережами, вся ЗНМ висловлює одну диференційовану функцію: з одного боку це необроблені пікселі зображення, з іншого - висновок класу або групи ймовірних класів, які характеризують картинку. Тут також присутня функція втрати на останньому (повністю підключеному) шарі, а всі елементи розроблені і модифіковані з простими нейронними мережами, залишаються справедливими при роботі з ЗНМ.

Архітектура згорткових нейромереж робить явне припущення виду «вхідні дані є зображення», що дозволяє закодувати певні властивості під архітектуру. Завдяки цій особливості, попереднє оголошення можна реалізувати більш ефективно, зменшуючи при цьому кількість параметрів в мережі [11].

Як відомо, нейронні мережі отримують вхідні дані (один вектор), після чого трансформують інформацію, проводячи її через ряд прихованих шарів. Кожен прихований шар складається з безлічі нейронів, де всякий нейрон має стійкий зв'язок з усіма нейронами в попередньому шарі і де нейрони в функції одного шару повністю незалежні один від одного і не мають спільних з'єднань. Останній повнозв'язний шар називається вихідним шаром, і в класифікації він демонструє число класів.

ЗНМ складається з декількох шарів.

Згортковий шар складається з набору карт або матриць, кожна з яких є ядром згортання, при цьому вагові коефіцієнти ядра вираховуються під час навчання, приклад чого можна побачити на рисунку 2.4:

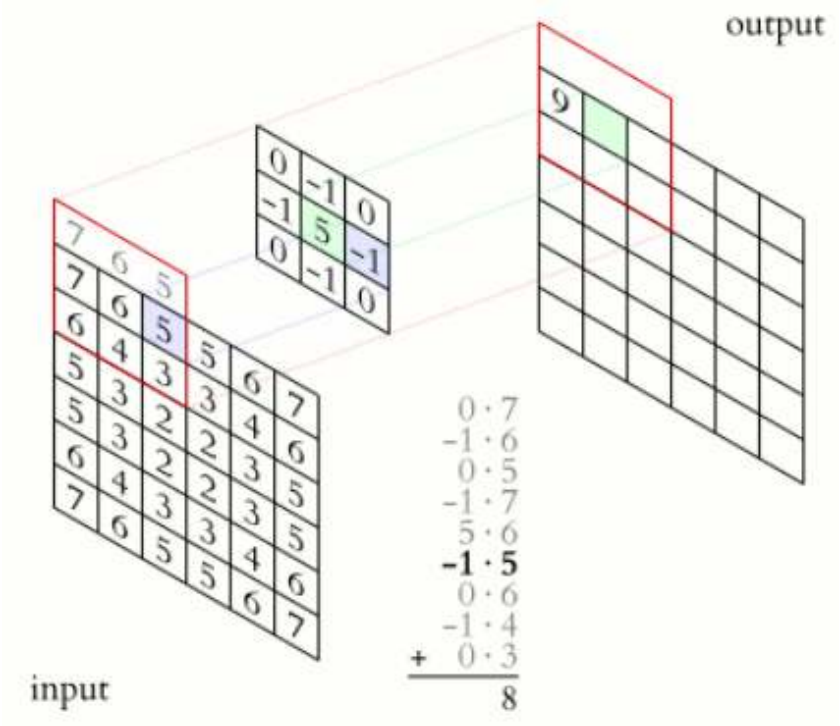

Рисунок 2.4 – Згортання зображення

Шар активації після проходження згортання результат потрапляє у визначену функцію активації;

Шар підвибірні – зменшують розмірність згортальних карт відповідно до виявлених ознак, приклад чого можна побачити на рисунку 2.5:

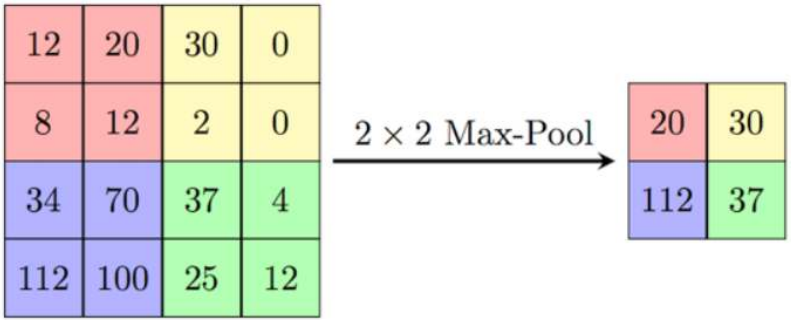

Рисунок 2.5 – Формування нової карти підвибірного шару на основі попередної згортальної карти

Повнозв'язний персептрон, зазвичай багатошаровий, що класифікує та моделює складу нелінійну функцію, яка шляхом оптимізації покращую якість розпізнавання [9].

Серед переваг ЗНМ виділяють наступні:

- один з найкраших алгоритмів з розпізнавання та класифікації зображень;
- в порівнянні із повнозв'язною нейронною мережею має досить меншу кількість вагових коефіцієнтів, що налаштовуються, бо одно ядро використовується цілком для всього зображення, замість того, щоб робити для кожного пікселя зображення власні вагові коефіцієнти, що підштовхує нейромережу під час навчання до узагальнення інформації;
- розпаралелення обчислень;
- стійкість до раптових поворотів чи зсувів вхідного зображення;
- здатність навчатись за допомогою методу зворотньої поширеності помилки.

Проте ЗНМ має занадто багато параметрів, що потребують застосункових ресурсів.

2.3 Алгоритм зворотнього поширення помилки

ЗНМ за своїм типом поширення активаційного сигналу між нейронами є прямими, тому застосування алгоритму зворотного поширення помилки (АЗПП) є цілком доречним до таких мереж. Даний алгоритм ще називають алгоритмом градієнтного спуску через те, що стратегія підбору такого важливого параметру, як вага для кожного нейрона багатошарової мережі базується на градієтному методі. Безперервна цільова функція як показник успішності мережі в загальному випадку визначається як квадратична різниця суми між фактичним результатом і очікуваним вихідним [12].

АЗПП при навчанні використовує два поширення мережею – пряме та зворотнє. На самому початку алгоритму відбувається саме прямий прохід де вхідні дані у вигляді вектору реалізують поширення між шарами, від початкових до останніх.

В результаті прямого проходу генерується набір вихідних сигналів, який і визначає реакцією мережі на вхідні дані. Під час прямого проходу усі синаптичні ваги мережі є фіксованими. Другим етапом алгоритму є зворотній прохід де усі вагові коефіцієнти налаштовуються відповідно за правилами корекції помилок. Суть згаданого правила наступна: від очікуваних вихідних значень віднімається отримане значення фактичного виходу і в результаті такої операції формується сигнал помилки. Сигнал помилки поширюється, як відлуння, в протилежному синаптичних зв'язків, тому алгоритм і отримав таку назву.

А вагові коефіцієнти у свою чергу підлаштовуються з метою максимального наближення вихідного сигналу мережі очікуваного результату. Недоліком алгоритму зворотного поширення помилки є те, що він не дозволяє в загальному випадку досягти глобального мінімуму. Тому і постають основні труднощі навчання нейронних мереж, що полягають в методах виходу з локальних мінімумів.

На рисунку 2.6 можна побачити блок-схему АЗПП:

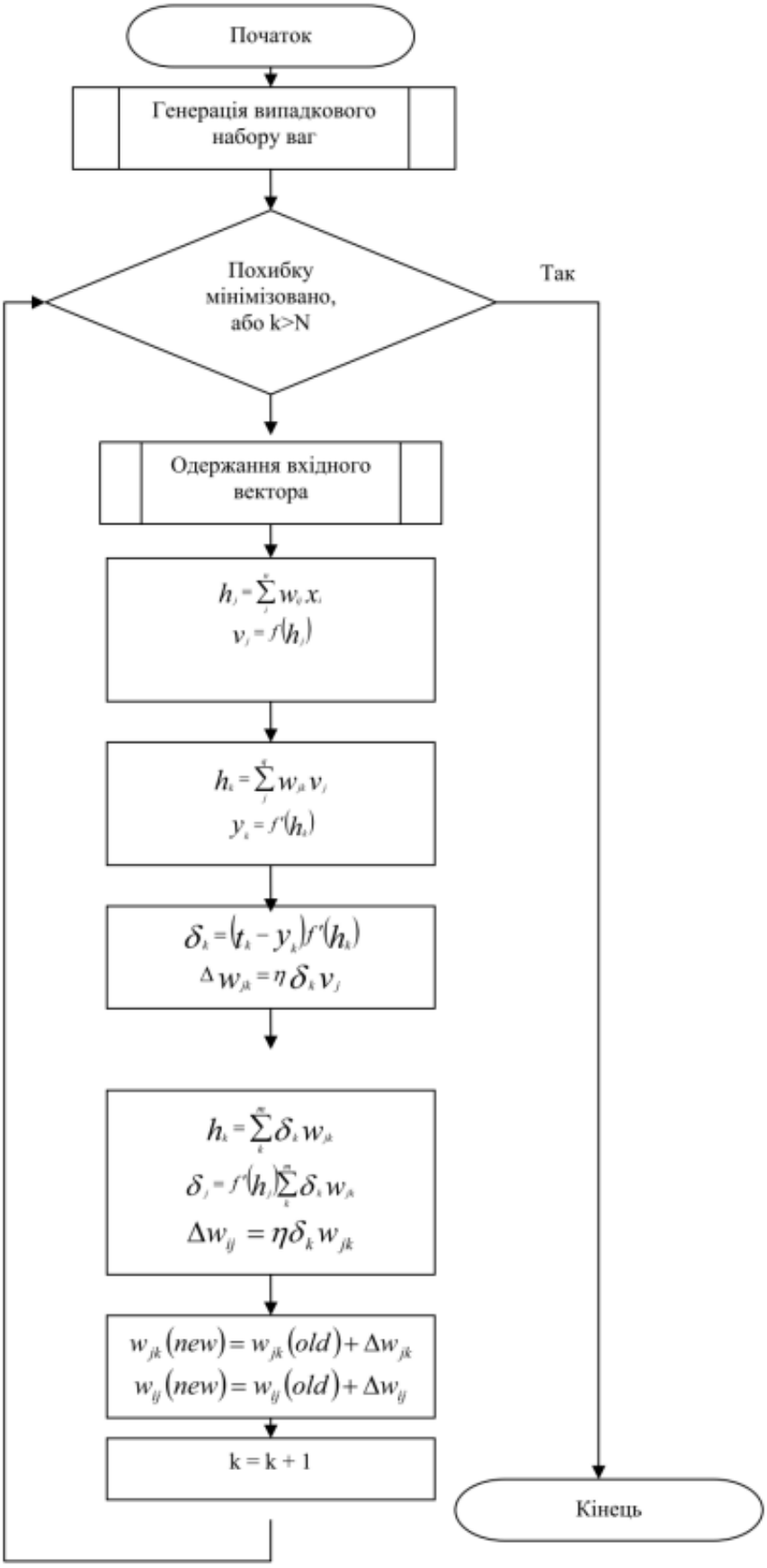

Рисунок 2.6 – Блок-схема АЗПП

Головними недоліками градієнтного спуску або алгоритму зворотнього поширення помилки при навчанні мережі є «Параліч» мережі та розмір кроку.

«Параліч» мережі. Значення ваг мережі в результаті корекції можуть досягти дуже великих величин. Оскільки помилка, що посилається назад впроцесі навчання, пропорційна похідній стискаючій функції, процес навчання може майже зупинитися. Цьому можна запобігти, зменшуючи крок, але процес навчання при цьому буде відбуватися довше [13].

Розмір кроку. Якщо значення кроку не змінюється і воно досить мале, то метод сходитися занадто повільно. Якщо ж крок занадто великий, то може виникнути параліч мережі. Необхідно змінювати значення кроку: збільшувати до тих пір, 50 поки не припиниться поліпшення оцінки в напрямок антиградієнта і зменшувати, якщо оцінка не поліпшується [13].

2.4 Типові завдання з розпізнавання

Нейронна мережа забезпечує вражаючу продуктивність, проте на початку ця продуктивність була незрозумілою та загадковою. Ваги та упередження в мережі були виявлені автоматично. А це означає, що не відразу можна знайти пояснення того, як нейронна мережа працює, на перший поглядю Для цього необхідно знайти спосіб зрозуміти принципи, за якими вона класифікує рукописні символи. І, враховуючи такі принципи, знайти варіанти покращення. Можна припустити, що через кілька десятиліть нейронні мережі ведуть до штучного інтелекту (ШІ). У перші дні досліджень ШІ люди сподівались, що зусилля з побудови ШІ також допоможуть зрозуміти принципи інтелекту та, можливо, функціонування людського мозку. Але, можливо, результатом буде те, що в підсумку ні мозок, ні те, як працює штучний інтелект не будуть до кінця зрозумілими. Щоб вирішити ці питання, можна взяти інтерпретацію штучних нейронів, як засобу зважування доказів. Наприклад, перед нейронною мережею стоїть задача

розпізнати тварину, наприклад шиншилу, яку можна побачити на рисунку 2.7.

Рисунок 2.7 – Зображення шиншили

Зображення на рисунках 2.8 – 2.10 є групою зображень разом із зображенням шиншили, серед яких нейронна мережа має знайти визначену тварину.

Рисунок 2.8 – Зображення горобця

Рисунок 2.9 – Зображення пейзажу

Рисунок 2.10 – Зображення квітки

Щоб зрозуміти як би працювала нейрона мережа, можна спробувати спроєктувати мережу самостійно, вибираючи відповідні ваги та ухили.

Найбільш простим підходом, на мою думку, є евристичний. Для визначення зображення можна розбити задачу на підзадачі – уточнюючі

питання, як от «Чи шиншила знаходиться у центрі зображення?», «Чи в шиншили є хутро?», «Чи передні лапи менші за задні?», «Чи має шиншилла пухнастий хвіст?» і так далі.

Якщо відповіді на кілька з цих запитань – «так» чи «ймовірно так», то можна зробити висновок, що на зображенні дійсно шиншилла. В противному випадку, на зображенні не зображена шиншилла.

Зрозуміло, що в цього підходу є багато недоліків, проте головним було показати, що задачу можна розбити на підзадачі для полегшення її розв'язку. І якщо можна вирішити підзадачі за допомогою нейронних мереж, то можна побудувати нейронну мережу для розпізнавання шиншили, об'єднавши мережі підпроблем. На рисунку 2.11 зображено можливу архітектуру нейронної мережі, що містить підмережі для вирішення підзадач. Ця архітектура не є робочою в повному сенсі, проте дає змогу краще зрозуміти принцип роботи нейронної мережі для розв'язку задачі з розпізнавання шиншили.

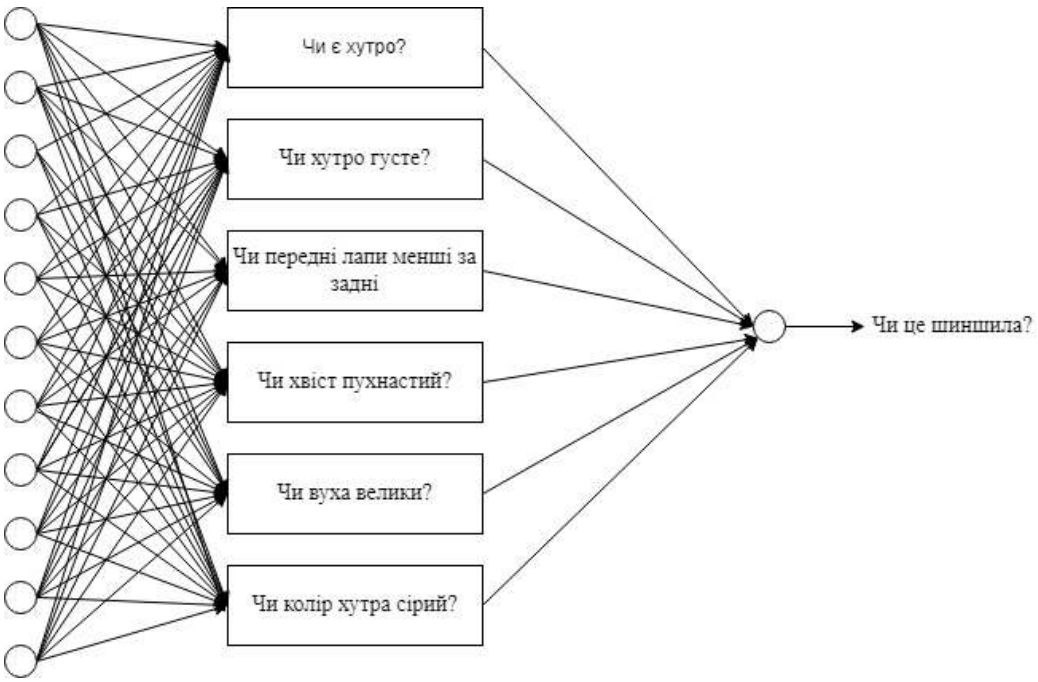

Рисунок 2.11 – Архітектура нейронної мережі для вирішення задачі з пошуку та розпізнавання зображення шиншили

Питання з підзадач теж можна розбити на підзадачі, формуючі застосункові шаріи. Зрештою, усі зійдо до роботи з підзадачами, або підмережами, які відповідатимуть на такі прості запитання, що на них можна легко відповісти на рівні одиничних пікселів. Ці питання можуть, наприклад, стосуватися наявності чи відсутності дуже простих фігур у певних точках зображення. На такі запитання можуть відповісти поодинокі нейрони, підключені до необроблених пікселів на зображенні.

Кінцевим результатом є мережа, яка розбиває дуже складне запитання – показує це зображення шиншилу чи ні – на дуже прості запитання, що відповідають на рівні одиничних пікселів. Він робить це через безліч шарів, причому ранні шари відповідають на дуже прості та конкретні запитання щодо вхідного зображення, а пізніші шари вибудовують ієрархію все більш складних та абстрактних концепцій. Мережі з таким різновидом багатошарової структури – два або більше прихованих шарів – називаються глибокими нейронними мережами. Не має сенсу самостійно розробляти ваги та упередження в мережі. Натомість бажано використовувати алгоритми навчання, щоб мережа могла автоматично вивчати ваги та упередження – і, отже, ієрархію концепцій – з навчальних даних. Дослідники у 1980-х і 1990-х роках намагалися використовувати стохастичний градієнтний спуск і зворотнє поширення для підготовки глибоких мереж. На жаль, за винятком декількох спеціальних архітектур, їм не сильно пощастило. Мережі вчились, але дуже повільно, а на практиці часто занадто повільно, щоб бути корисними [40].

З 2006 року було розроблено набір методик, які дозволяють навчатись у глибоких нейронних мережах. Ці методи глибокого навчання засновані на стохастичному градієнтному спуску та зворотньому розповсюдженні, але також вводять нові ідеї. Ці методи дозволили навчити набагато глибші (і більші) мережі – тепер люди регулярно тренують мережі з 5-10 прихованими рівнями. І, виявляється, вони працюють набагато краще для багатьох проблем, ніж неглибокі нейронні мережі, тобто мережі з одним

єдиним прихованим шаром. Причиною, звичайно, є здатність глибоких мереж будувати складну ієрархію понять. Це трохи схоже на те, як звичайні рукописного тексту програмування використовують модульний дизайн та ідеї про абстракцію для створення складних комп’ютерних програм. Порівняння глибокої мережі з неглибокою мережею трохи схоже на порівняння рукописного тексту програмування з можливістю здійснювати функціональні виклики до розірваного рукописного тексту без можливості здійснення таких дзвінків. Абстракція набуває в нейронних мережах іншої форми, ніж у звичайному програмуванні, але це так само важливо.

3 ПРОЄКТУВАННЯ ТА РОЗРОБКА ЗАСТОСУНКУ З РОЗПІЗНАВАННЯ РУКОПИСНИХ МАТЕМАТИЧНИХ ФОРМУЛ

3.1 Засоби та технологія розробки застосунку

Враховуючи останні події, через які постійно є перебої з електроенергією та інтернетом, було прийнято рішення розробляти застосунок десктопним, що буде зручним для додавання оцифрованих рукописних математичних формул у документи чи зберігання їх у ЕОМ. Результат обробки зберігається у файлові, відкривши який користувач побачить переведений у текстовий варіант рукописний математичний вираз. Рукописний математичний вираз можна ввести в застосунок за допомогою малювання на планшеті чи тачпаді, або завантажити зображення рукописного математичної формули. Розпізнається рукописний математиичний вираз онлайн-методом, що реалізований згорнутою нейронною мережею, яка навчалася на великому масиві даних алгоритмом зворотнього пошуку помилки.

Для реалізації застосунку з розпізнавання рукописних математичних формул було обрано мову програмування Python, оскільки вона має велику кількість інструментів та бібліотек для роботи з нейронними мережами, такі як keras, tensorflow, opencv, та ін.

Python був розроблений наприкінці 1980-х років співробітником голандського інституту Centrum Wiskunde & Informatica (CWI) Гвідо ван Россумом. У лютому 1991 року У лютому 1991 року Гвідо опублікував сирцевий код. Мова почала вільно поширюватися через Інтернет і сподобалася іншим програмістам. З 1991 року Python є цілком об'єктно-орієнтованим. Python також запозичив багато рис таких мов, як C, C++, Modula-3 і Icon, й окремі риси функціонального програмування з Лісп.

Python підтримує динамічну типізацію, умовні конструкції, масиви, цикли, кортержі, словники. Також він є об’єктно-орієнтованою мовою програмування, проте навідміну від рештии ООП-мов має певні особливості:

- класи є одночасно об'єктами з усіма нижче наведеними можливостями;
- успадкування, в тому числі множинне;
- поліморфізм (всі функції віртуальні);
- шнкапсуляція (два рівні – загальнодоступні та приховані методи і поля), особливість – приховані члени доступні для використання та помічені як приховані лише особливими іменами;
- спеціальні методи, що керують життєвим циклом об'єкта: конструктори, деструктори, розподільники пам'яті;
- перевантаження операторів (усіх, крім `is, '.', '='` і символьних логічних);
- властивості (імітація поля за допомогою функцій);
- управління доступу до полів (емуляція полів і методів, частковий доступ тощо);
- методи для управління найпоширенішими операціями (істиннісне значення, `len()`, глибоке копіювання, серіалізація, ітерація по об'єкту тощо);
- метапрограмування (управління створенням класів, тригери на створення класів, та ін);
- повна інтроспекція;
- класові та статичні методи, класові поля;
- класи, вкладені у функції та інші класи [14].

TensorFlow – це бібліотека для машинного навчання, групи технологій, яка дозволяє навчати штучний інтелект вирішенню різних завдань. Бібліотека спочатку розроблена для Python і найчастіше використовується з ним. Бібліотека розроблена Google як продовження внутрішньої бібліотеки компанії. TensorFlow безкоштовна, вона має відкритий вихідний код, який можна переглянути на GitHub, її активно підтримує спільнота ентузіастів.

Сама бібліотека включає безліч інструментів для різних напрямків ML, але найчастіше використовується для роботи з нейронними мережами. Це структури, натхненні пристроєм мереж нейронів у людській нервовій системі. Нейронні мережі складаються із програмних елементів-«нейронів» та зв'язків між ними, і такий пристрій дозволяє їм навчатися. TensorFlow працює зі звичайними та глибокими нейронними мережами різних типів: рекурентними, згортковими тощо. Також вона використовується для машинного та глибокого навчання.

У TensorFlow моделі представлені за допомогою графів – математичних абстракцій, які складаються з вершин та шляхів між ними. Граф можна порівняти із схемою доріг між різними точками. У програмуванні це зазвичай потрібно при вирішенні «маршрутних» завдань та при створенні нейронних мереж. TensorFlow працює з тензорами - багатовимірними структурами даних у векторному, тобто спрямованому просторі. Вони використовуються в лінійній алгебрі та фізиці. Звідси походить назва бібліотеки. За допомогою тензорів описуються шляхи графа, а вершини це математичні операції. Обчислення TensorFlow виражаються як потоки даних через граф. Це означає, що інформація «рухається» за графом, передається шляхами від вершини до вершини. Бібліотека може працювати на потужностях звичайного центрального процесора (CPU) або використовувати потужності графічного процесора (GPU). Режим перемикається у коді. Існує спеціальний тензорний процесор TPU, створений розробниками бібліотеки, - ним можна скористатися через хмарні сервіси Google.

TensorFlow має наступні переваги:

- високий рівень абстракції;
- інтерактивність розробки;
- гибкість;
- кросплатформеність;
- велика спільнота.

Проте має і такі недоліки як велике споживання пам’яті та власні стандарти [15].

Keras – це бібліотека для мови програмування Python, призначена для глибокого машинного навчання. Вона дозволяє швидше створювати та налаштовувати моделі — схеми, якими поширюється і підраховується інформація під час навчання. Але складних математичних обчислень Keras не виконує та використовується як надбудова над іншими бібліотеками.

Keras з версії 2.3 є надбудовою над бібліотекою TensorFlow, яка потрібна для машинного навчання. TensorFlow виконує всі низькорівневі обчислення та перетворення і є своєрідним двигуном, математичним ядром. Keras управляє моделями, за якими проходять обчислення.

Keras створювалася як гнучка модульна бібліотека, яку легко налаштовувати та модифікувати. Вона безкоштовна, має відкритий вихідний код, який може подивитися будь-хто. Keras має вузьку спеціалізацію. Це інструмент для фахівців з машинного навчання, які працюють з мовою Python: саме його найчастіше використовують завдяки зручності математичних обчислень. Keras застосовують розробники, які створюють, налаштовують та тестують системи машинного навчання та штучного інтелекту, насамперед нейронні мережі.

Keras повністю написана на Python, щоб код був зрозумілішим і легшим підтримувався. Працює на більшості існуючих платформ: не тільки в ОС Windows і Linux, а й на мікрокомп'ютерах, мобільних пристроях, хмарі або браузері. Підтримує роботу з CPU та GPU - зі звичайним або графічним процесором. Останній часто використовують для складних обчислень коли CPU не справляється. Підтримує різні види нейронних мереж: класичні перцептрони, згорткові та рекурентні мережі, їх комбінації. Сумісна з Python починаючи з версії 2.7 аж до сучасних.

Keras має наступні переваги:

- зручність роботи;
- модульність;

- швидкість;
- велика спільнота.

Проте має і такі недоліки:

- відсутність зворотньої сумісності;
- залежність від TensorFlow;
- складність вивчення [16].

NumPy – це бібліотека Python, що застосовується для математичних обчислень: починаючи з базових функцій і закінчуючи лінійною алгеброю. Повна назва бібліотеки – Numerical Python extensions, або Числові розширення Python.

Ця бібліотека має кілька важливих особливостей, які зробили її популярним інструментом. По-перше, вихідний код у вільному доступі зберігається на GitHub, тому NumPy називають open-source модулем для Python. По-друге, бібліотека написана мовами C та Fortran. Це компілювані мови (мови програмування, текст яких перетворюється на машинний код — набір інструкцій для конкретного типа процесора. Перетворення відбувається за допомогою спеціальної програми-компілятора, завдяки якому обчислення компілюваними мовами відбуваються швидше), на яких обчислення виробляються набагато швидше і ефективніше, ніж мовами, що інтерпретуються (мови програмування, які не заточені під конкретний тип процесора і можуть бути запущені на різних типах пристроїв). До цих мов належить і сам Python.

NumPy користуються вчені для вирішення багатовимірних завдань у математиці та фізиці, біоінформатиці, обчислювальної хімії та навіть когнітивної психології. На основі NumPy з'являються нові типи масивів, можливості яких виходять за межі того, що пропонує бібліотека. Наприклад, бібліотеки Dask, CuPy чи XND. В основі екосистеми для аналізу даних лежить NumPy. Бібліотека використовується на всіх етапах роботи з даними: вилучення та перетворення, аналіз, моделювання та оцінка, репрезентація. Бібліотеки для машинного навчання scikit-learn та SciPy також працюють

завдяки обчислювальним потужностям NumPy. У порівнянні безпосередньо з Python можливості NumPy дозволяють дослідникам візуалізувати набори даних, які набагато більші за розміром [17].

OpenCV (Open Source Computer Vision Library) випущений під ліцензією BSD і, отже, безкоштова як для академічного, так і для комерційного використання. Вона має інтерфейси для мов C ++, Python і Java і підтримує Windows, Linux, Mac OS, iOS та Android. Бібліотека OpenCV була розроблений для задач розпізнавання. Мова на якій вона написана це C/C++, сама ж бібліотека може скористатися багатоядерною обробкою. Бібліотека використовує OpenCL, вона може скористатися апаратним прискоренням базової неоднорідної обчислювальної платформи.

Прийнята по всьому світу, OpenCV налічує понад 47 тисяч користувачів спільноти та приблизну кількість завантажень понад 14 мільйонів. Використання цієї бібліотеки дуже широке від інтерактивного мистецтва до огляду фільмів та воєного обладнання.

OpenCV – це бібліотека з набором методів, алгоритмів та технологій для роботи із розпізнаванням зображень. Бібліотека має в собі готові методи для використання користувачем, тому написати просту програму для розпізнавання не завдає великих труднощів. Сама в собі бібліотека має готові каскади, що дають змогу змогу знаходити певні нам об’єкти. Так стандартна бібліотека дає нам можливість знаходити майже всі елементи лиця, певні частини тіла та все тіло взагалом. Таких можливостей часто достатньо для простих задач [18].

Бібліотека Pillow є продовженням більш старої бібліотеки PIL (Python Imaging Library), що є, в свою чергу, оригінальною бібліотекою Python, призначеної для роботи з растровими зображеннями. Використання PIL було припинено у 2011 році, оскільки підтримувалося лише другою версією мови Python. З появою Pillow розробники стали розглядати цю бібліотеку, як зручніше відгалуження PIL, яке зберігає всі функції старої бібліотеки, але включає підтримку Python 3-ї версії.

Однак бібліотека Pillow і досі залишається надзвичайно важливим і популярним інструментом серед розробників для роботи із зображеннями. Однією з причин такої сильної популярності цієї бібліотеки є досить велика схожість основних її функцій обробки зображень з подібними ж функціями властивими для таких графічних редакторів, як Adobe Photoshop. Найчастіше, саме тому, наш вибір падає на Pillow, де процеси обробки зображень дуже пов'язані з аналогічними процесами рівня графічних редакторів, які у свою чергу не вимагають занадто просунутих навичок у сфері маніпулювання растровими зображеннями. По суті, з цієї ж причини легкості освоєння можливостей бібліотеки за аналогією з графічними редакторами, Pillow набула найширшого поширення і в різних науково-дослідних сферах серед учених, які займаються програмуванням на аматорських засадах.

Усе вищеописане зумовило те, що, маючи досить значні можливості в поєднанні з відносно легким процесом освоєння в порівнянні з іншими аналогічними модулями, бібліотека Pillow, на сьогоднішній день є одним з найбільш популярних інструментаріїв для роботи з растровими зображеннями [19].

Як середовище розробки була обрана кроссплатформенна платформа ***PyCharm***. PyCharm – це кроссплатформенна інтегрована середовище розробки мови програмування Python, розроблена компанією JetBrains на основі IntelliJ IDEA. Надає користувачеві комплекс засобів для графічного налагоджувача та роботи з кодом. PyCharm допомагає розробнику писати код чистіше та швидше. Він автоматизує рутину, виділяє помилки, сам робить виправлення.

Головні особливості розумного редактора:

- підсвічування синтаксису Python і шаблонів Django, колір якого можна змінювати;
- автоматичне додавання відступів, форматування коду;
- можливість вибрати стиль написання коду;
- варіанти автодоповнення;

- підтримка автогенерації коду;
- пошук дублікатів та швидкі виправлення;
- шаблони коду та сніпети.

Допомога не обмежується лише написанням коду. Налагодження, профільування, тестування, розгортання, використання систем контролю версій, віддалена розробка – все це PyCharm пропонує встановленим "з коробки" або у вигляді зручних плагінів.

Існує дві версії Розробники PyCharm: Community Edition – безкоштовна версія, знаходиться під ліцензією Apache License, і Professional Edition – розширена версія продукту, що має застосункову функціональність, є пропрієтарним ПЗ. JetBrains визначають головну різницю так: PyCharm Community призначений для роботи з чистим Python. Версія Pro «з коробки» підтримує технології фронтенду та баз даних, пітонівські фреймворки та профільники, інструменти для Data Science [20].

3.2 Опис реалізації програмного продукту

У розробленому застосунку користувач може виконувати наступні функції:

- рукописне введення математичної формули шляхом малювання мишею чи тачпадом у виділеній області;
- очищення області малювання;
- завантаження зображення із рукописним математичним виразом;
- визначення символів, що були написані;
- оцифрування рукописного математичної формули із занесенням у файл.

Структура застосунку з розпізнавання рукописних математичних формул була реалізована наступним чином, зображеним на рисунку 3.1.

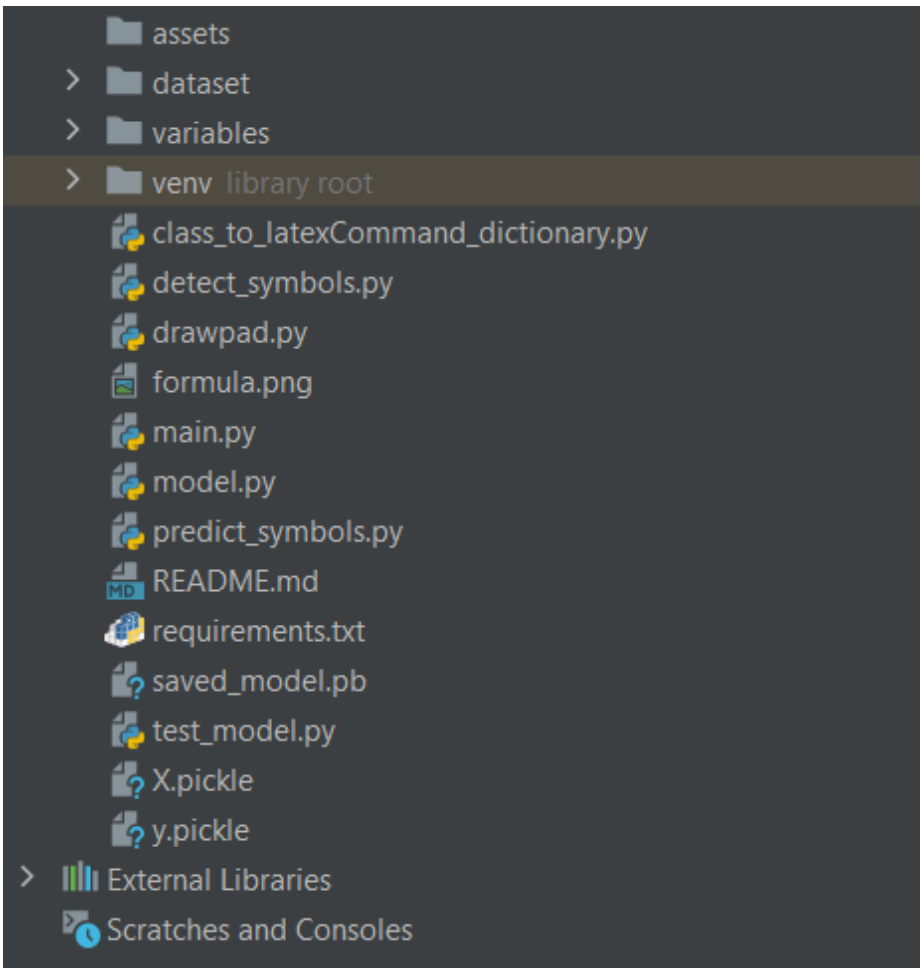

Рисунок 3.1 – Структура застосунку з розпізнавання рукописних математичних формул

Найважливіші файли і папки у структурі наступні:

- dataset – містить в собі набір зображень символів, на яких навчається нейронна мережа, в якості датасету було обрано;
- detect_symbols.py – визначає на зображені математичні символи;
- predict_symbols.py – зчитує визначені математичні символи;
- model.py – навчає нейронну мережу.

Було обрано основний інструментарій для створення десктоп-застосунку з розпізнавання рукописних математичних формул: МП - Pythom, бібліотек для створення нейроннної мережі – TensorFlow із Keras, OpenCV, NumPy, середовища розробки – PyCharm Community Edition, структуру проєкту застосунку та головні модулі.

3.3 Аналіз працездатності розробленого програмного продукту

Навчання моделі реалізовано у файлі model.py. Обробка відсканованого зображення, які були перетворені у матриці з використанням бібліотеки NumPy у функції створення тренувальних даних. Для навчання ЗНМ була використана бібліотека Keras, що є надбудовою для бібліотеки TensorFlow, про що було написано раніше.

Використовуючи модель Sequental для створення послідовної моделі із двома згортковими шарами, що мають 32 ядра розміром 3 x 3 кожен, за якими йдуть відсівні шари розміром 2 x 2, а також відсівний шар із коефіцієнтом 0,25 для уникнення переобладнання. При такої конфігурації число ядра і ряд класів символів однакові. Далі отримана модель згортається з двовимірної до одновимірної , після чого створюється шар, що виконує класифікацію класів.

Після цього було почато тренування послідовної ЗНМ протягом 15 епох чи повторень.

Серед 28251 зображення 26906 використовуютьс для навчання, а 1346 використовуються для перевірки (5%). Значення втрати та точності, а також тестових значень після кожної епохи навчання можна побачити на рисунку 3.2.

Середнє значення втрат становить. 0.03, тестове значення – 1.3. В той же час значення точності становить 0.98, а тестове значення 0.86.

На рисунках 3.3 та 3.4 можна побачити криві втрати та точності відповідно.

```
Epoch 1/15
26906/26906 [==============================] - 12550s 466ms/step - loss: 0.2991 - accuracy: 0.8996 - val_loss: 0.2064 - val_accuracy: 0.9513
Epoch 2/15
26906/26906 [==============================] - 16974s 631ms/step - loss: 0.0242 - accuracy: 0.9938 - val_loss: 0.2041 - val_accuracy: 0.9463
Epoch 3/15
26906/26906 [==============================] - 18499s 688ms/step - loss: 0.0118 - accuracy: 0.9970 - val_loss: 0.0681 - val_accuracy: 0.9823
Epoch 4/15
26906/26906 [==============================] - 17422s 648ms/step - loss: 0.0082 - accuracy: 0.9979 - val_loss: 0.0691 - val_accuracy: 0.9812
Epoch 5/15
26906/26906 [==============================] - 14331s 533ms/step - loss: 0.0065 - accuracy: 0.9983 - val_loss: 0.1376 - val_accuracy: 0.9791
Epoch 6/15
26906/26906 [==============================] - 16997s 632ms/step - loss: 0.0068 - accuracy: 0.9986 - val_loss: 0.0478 - val_accuracy: 0.9922
Epoch 7/15
26906/26906 [==============================] - 17957s 667ms/step - loss: 0.0058 - accuracy: 0.9988 - val_loss: 0.0340 - val_accuracy: 0.9941
Epoch 8/15
26906/26906 [==============================] - 18677s 694ms/step - loss: 0.0058 - accuracy: 0.9989 - val_loss: 0.1393 - val_accuracy: 0.9845
Epoch 9/15
26906/26906 [==============================] - 17867s 664ms/step - loss: 0.0058 - accuracy: 0.9989 - val_loss: 0.0143 - val_accuracy: 0.9981
Epoch 10/15
26906/26906 [==============================] - 15432s 574ms/step - loss: 0.0032 - accuracy: 0.9994 - val_loss: 0.5774 - val_accuracy: 0.9595
Epoch 11/15
26906/26906 [==============================] - 14789s 550ms/step - loss: 0.0045 - accuracy: 0.9990 - val_loss: 0.1409 - val_accuracy: 0.9856
Epoch 12/15
26906/26906 [==============================] - 15589s 579ms/step - loss: 0.0065 - accuracy: 0.9991 - val_loss: 0.1745 - val_accuracy: 0.9800
Epoch 13/15
26906/26906 [==============================] - 15458s 575ms/step - loss: 0.0056 - accuracy: 0.9991 - val_loss: 0.0964 - val_accuracy: 0.9941
Epoch 14/15
```

Рисунок 3.2 – Результати навчання ЗНМ протягом п’ятнадцяти епох

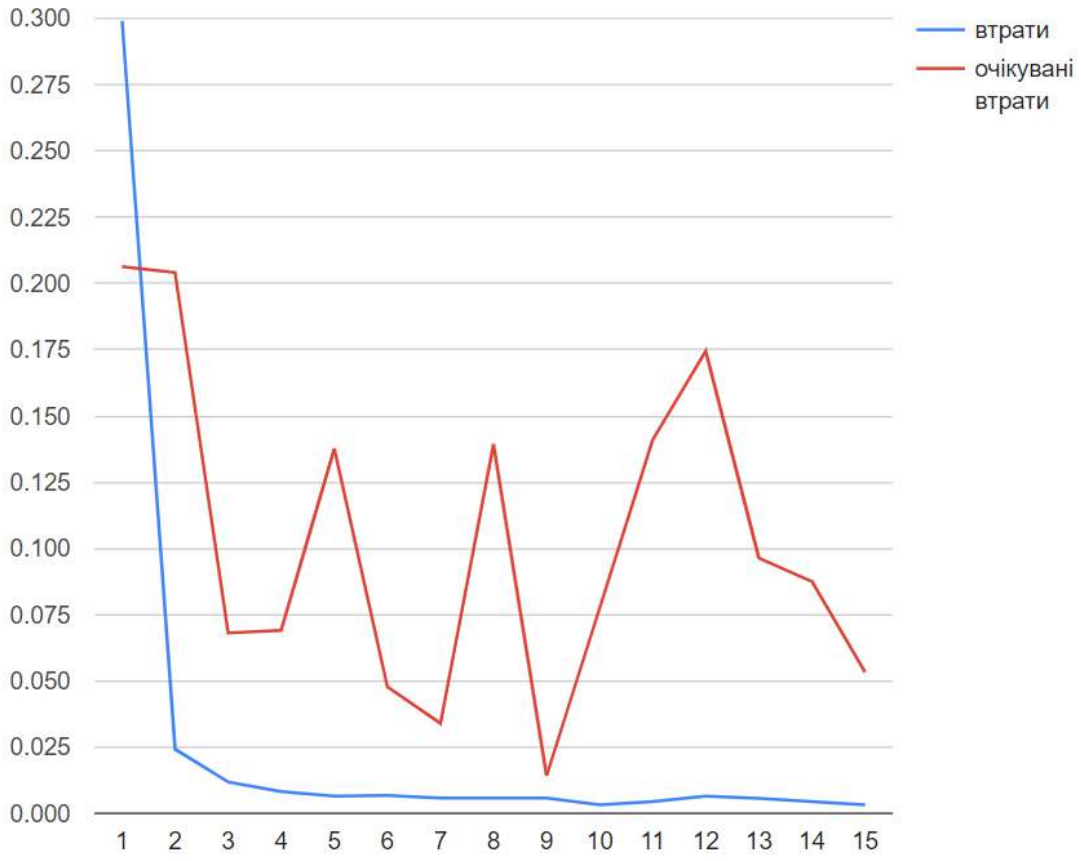

Рисунок 3.3 – Крива втрат

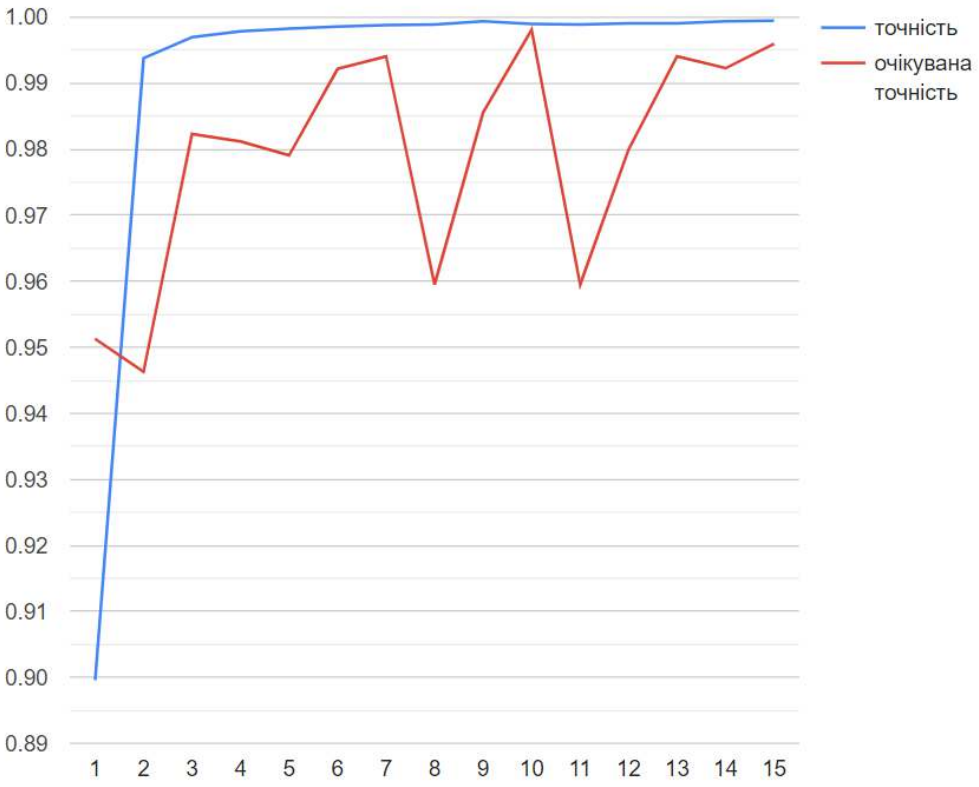

Рисунок 3.4 – Крива точності

Далі було перевірено розпізнавання рукописного математичної формули шляхом написання мишкою, що можна побачити на рисунку 3.5.

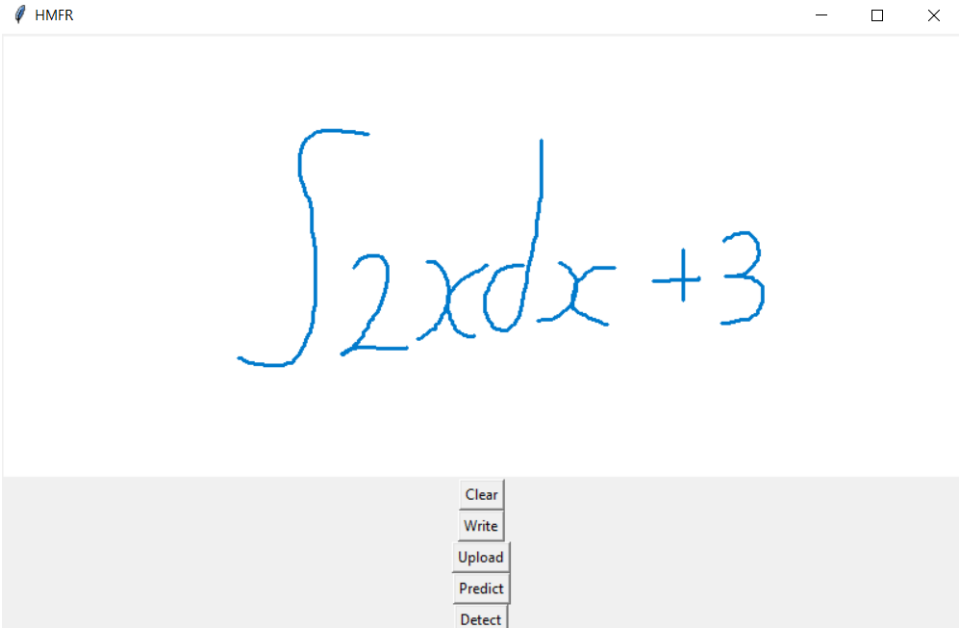

Рисунок 3.5 – Написаний рукою математичний вираз у застосунку

Після натискання на кнопку “Detect” програма визначає на зображені математичний вираз, а після натискання кнопки “Predict” починає розпізнаватии математичний вираз та записує у форматі unicode к файлі result.txt, що лежить в корневій папці застосунку та зображений на рисунку 3.6.

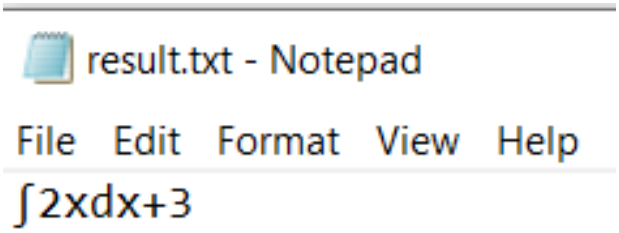

Рисунок 3.6 – Розпізнаний математичний вираз у файлі result.txt

Після натискання на кнопку “Clean” написаний математичний вираз видаляється, що можна побачити на рисунку 3.7.

Рисунок 3.7 – Застосунок після натискання кнопки “Clean”

Далі було перевірено можливість завантаження зображеня математичної формули для його подальшого розпознавання, який можна побачити на рисунку 3.8.

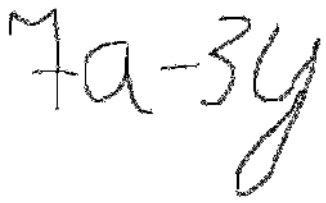

Рисунок 3.8 – Зображнння рукописного математичної формули

Після цього було натиснуто на кнопку “Upload” та відчинено вікно вибору файлу, що зображено на рисунку 3.9.

Далі файл було завантажено, натиснуто кнопки “Detect” та “Predict”, після чого результат розпізнавання був записаний у файл result.txt. Що можна побачити на рисунку 3.10.

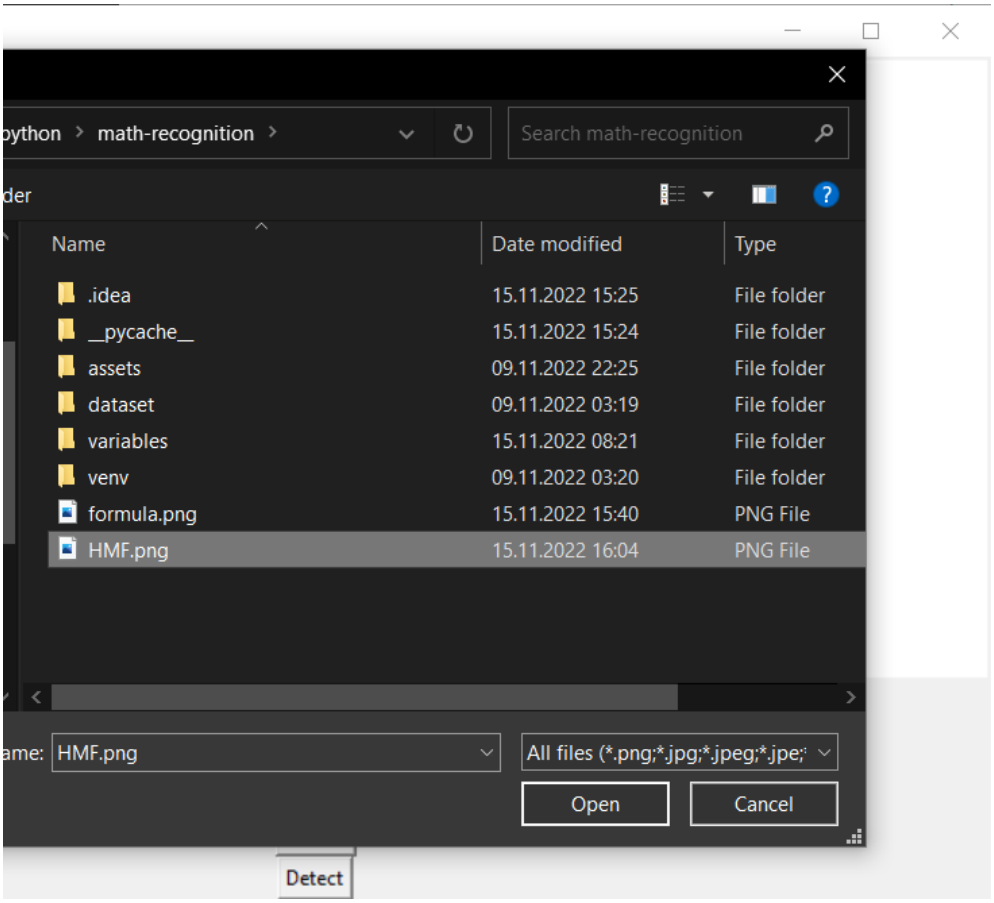

Рисунок 3.9 – Вибір файлу HMF.png із рукописним математичним виразом для подальшого розпізнавання

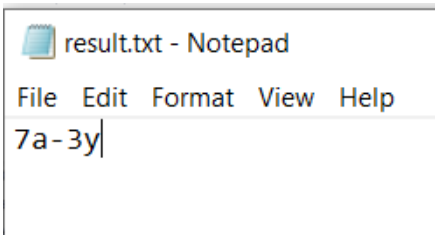

Рисунок 3.10 – Розпізнаний математичний вираз у файлі result.txt

Було навчено розроблену модель нейронної мережі розпізнавати рукописні математичні формули та вирази, перевірено працездатність розробленого застосунку з розпізнавання рукописного тексту шляхом самостійного введення за допомогою тачпаду та завантаження вже існуючого зображення.

ВИСНОВКИ

У кваліфікаційній роботі була розглянута тема розпізнавання рукописних математичних формул. В результаті виконання кваліфікаційної роботи був створений десктопний застосунок, що дозволяє розпізнати написані власноруч математичні формули за допомогою ЗНМ, мови програмування Python та бібліотек, пов’язаних із штучним інтелектом та нейронними мережами.

У першому розділі було ознайомлено із декількома застосунками, що виконують першочергово одну функцію – розпізнавання рукописних математичних формул, а після їх аналізу було зрозуміло що потрібно реалізувати у розробленому застосунку.

У другому розділі було викладено теорію нейронних мереж та особливо згорткової нейронної мережі та алгоритму зворотнього поширення помилки, що його навчає.

У третьому розділі було обрано інструментарій для реалізації можливості розпізнавання рукописних математичних формул, було описано створення згорткової нейронної мережі, її навчання та показ застосунку у роботі.

Тож були досягнуті задачі з розробки застосунку з розпізнавання рукописних математичних формул та реалізації нейронної мережі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Шаповалов М.І., Кучук Н.Г. Методи розпізнавання рукописних формул. Проблеми інформатизації. Тези доповідей 11 міжнародної науково-технічної конференції, Т.1, Баку – Бельсько-Бяла – Харків, 15-16 листопада 2023.– С.72.

2. Detexify LaTeX handwritten symbol recognition. [Електронний ресурс] – Режим доступу: URL: http://detexify.kirelabs.org/

3. Google Документи [Електронний ресурс]. – Режим доступу: URL: https://docs.google.com/

4. Mathematical Expression Recognition. [Електронний ресурс]. – Режим доступу: http://cat.prhlt.upv.es/mer/

5. Photomath [Електронний ресурс]. – Режим доступу: https://photomath.com/en

6. Mathpix [Електронний ресурс]. – Режим доступу: https://mathpix.com/

7. Handwriting recognition [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Handwriting_recognition#:~:text=Handwriting%20recognition%20(HWR),%20also,touch-screens%20and%20other%20devices.

8. Combining Local Appearance and Holistic View: Dual-Source Deep Neural Networks for Human Pose Estimation cc Режим доступу: https://arxiv.org/pdf/1504.07159.pdf

9. Understanding Perceptron: The Founding Element of Newral Networks [Електронний ресурс]. –Режим доступу: https://www.analytixlabs.co.in/blog/what-is-perceptron/

10. Сверточная нейронная сеть, часть 1: структура, топология, функции активации и обучающее множество [Електронний ресурс]. – 2018. – Режим доступу: https://habr.com/ru/post/348000/

11. Розпізнавання образів та їх класифікація [Електронний ресурс]. –

2017. – Режим доступу: https://studopedia.com.ua/1_42779_zagalna-harakteristika-zadach-rozpiznavannya-obraziv-ta-matematichna-model-zadachi.ht

12. Convolutional newral network [Електронний ресурс]. – https://www.ibm.com/cloud/learn/convolutional-neural-networks

13. Алгоритм зворотного поширення помилки [Електронний ресурс] — Режим доступу: https://studfile.net/preview/4494757/page:5/

14. Метод зворотного поширення помилки [Електронний ресурс]. – 2023. Режим доступу: https://www.wiki.uk-ua.nina.az.

15. Python [Електронний ресурс] — Режим доступу: https://en.wikipedia.org/wiki/Python_(programming_language)

16. TensorFlow. [Електронний ресурс] — Режим доступу: https://www.tensorflow.org/

17. Keras. Офіційна документація [Електронний ресурс] — Режим доступу: https://keras.io/

18. NumPy. Офіційна документація [Електронний ресурс] — Режим доступу: http://www.numpy.org/

19. OpenCV. Офіційна документація [Електронний ресурс] — Режим доступу: http://www.opencv.org/

20. Pillow. Офіційна документація [Електронний ресурс] — Режим доступу: http://pillow.readthedocs.io/